Integrated Access 2013 Labs
for Database Design, Application Development, and Administration

Michael V. Mannino
The Business School, Campus Box 165, P. O. Box 173364
University of Colorado Denver, Denver, CO 80217-3364
Michael.Mannino@ucdenver.edu

February 2014

Table of Contents

CHAPTER 7: PIVOT TABLES AND SPECIALIZED ACCESS FORMS

CHAPTER 8: NAVIGATION STRUCTURE AND MACRO LAB

Chapter 1: Introduction to Microsoft Access 2013

Learning Objectives

This chapter begins your study of application development using Microsoft Access. After this chapter, you should have acquired the knowledge and skills to

- Become familiar with the Access user interface.
- Identify the objects of an Access database and object interactions.
- Understand the importance of different views of Access objects.
- Understand Access data types and field properties.
- Understand the purpose of wizards and expression builders.
- Distinguish between table and field properties.
- Understand the Access representation of primary keys and foreign keys.
- Become familiar with the tables and the relationships of the automobile repair database.
- Become familiar with the Access database maintenance tools.

Overview

In Chapter 3 of the textbook, you learned about the principles of the relational model and SQL statements to create tables. Most database management systems (DBMSs) have more convenient ways to create databases than using SQL statements. In addition, SQL statements for creating tables are not standard because data types and field properties vary. Thus, creating a database involves knowledge specific to each DBMS.

This chapter complements the conceptual background of textbook Chapter 3 by providing practical knowledge about creating database applications in Microsoft Access 2013, a powerful and popular desktop DBMS. In this chapter, you will learn about the objects in an Access database, the tools used to create databases, and the properties for tables, fields, and relationships. With this background, you can proceed to lab Chapter 2 for specific skills and practice with creating Access databases. You also may want to refer to this chapter when creating queries, forms, and reports described in later lab chapters.

In addition to learning about Microsoft Access, you will learn about the automobile repair database used in the other lab chapters. This fictional database supports order processing of a moderate-size auto repair shop. This database, although moderate in size, depicts many features of Microsoft Access 2013.

Microsoft Access 2013 is an incremental revision of the previous versions, Microsoft Access 2010 and Access 2007. The most visible change in Access 2007 was a new user interface, consistent with the other members of the Microsoft Office 2007 family. Microsoft Access 2010 has made incremental changes to the user interface featured in Access 2007. Access 2013 also uses the new user interface and ribbon that were introduced in Access 2007 although the tabs and tab items have been somewhat revised in Access 2010 and slightly modified in Access 2013.Access 2013 still supports the data macro feature, a major extension to application macros in earlier versions. Data macros provide a capability similar to triggers (see textbook Chapter 11) so that event driven coding can be attached to tables. In previous Access versions, macros could be associated only with events on application objects such as forms. Along with data macros, Microsoft Access 2013 also substantially extends the development environment and specification of macros.

Access 2013 substantially extends Web usage of Access databases by allowing a database application (tables and application objects) to be hosted by a Microsoft Sharepoint server and Microsoft Azure SQL cloud service. Access 2013 allows users to only open, design and publish pre existing Access 2010 web database but not creating new Access 2010 web database. Access 2013 substantially extends template tables in Access 2007 with complete database templates (tables and associated application objects), application parts (table and associated application objects), and sample fields. Template databases in Access 2013 can be hosted on a Microsoft Sharepoint server and Microsoft Azure.Access 2013 is also compatible with the new Microsoft service "Office365" small business and enterprise editions. As part of Web usage of Access, Microsoft has superseded the switchboard formwith navigation forms. The switchboard form is still supported in Access 2013, but it is no longer featured in the default ribbon. Navigation forms have replaced the switchboard form as the preferred method of navigation. Navigation forms provide a tabbed appearance consistent with Web interfaces for navigation.

Other extensions are more incremental. Access 2013 extends the expression builder with Intellisense, a feature providing context dependent completion of expressions. Intellisense is available in other Microsoft development tools so it is not a new Microsoft feature. Access 2013 provides a new data type, the calculated field data type for fields containing values calculated from other fields in the same table. Access 2013 only supports Access 2000 database formats or later versions. Access 97 databases and earlier must be converted to a compatible versions prior opening them in Access 2013. Finally, Access 2013 does not allow bypassing the ribbon interface to switch to the Access 2003 toolbars and menus. However, the Access 2003 toolbars and menus can be added to a custom ribbon group or the Quick Access Toolbar.

Conventions Used in the Lab Chapters

To facilitate your reading of this chapter and the other lab chapters, some conventions are useful. The conventions enable you to distinguish between different kinds of Access terms. Table 1 lists the conventions along with examples.

Table 1: Conventions Used

Term	Convention	Example
Ribbon tabs, command buttons, and drop down lists	Bold font	**Create** tab, **Create→Macro** command on the ribbon, and **Create→Navigation** drop down list
Menu command	Bold font	**Build Event …** command on a drop-down right button menu
Table and column names	Italics font	*Customer* table
Property names	Italics font	*Data Type* property
Property values	Double quotes	"Yes"
Function keys	Bold font	**F1** key
Code (SQL, VBA, …)	Courier font	`SELECT CustNo, …`

1.1 Objects in Access Database Applications

This section introduces you to the objects in Access database applications. To create a database application, you first define the tables before using them to define queries, forms, and reports. You can use macros and modules to customize the forms and reports in your database application. The need to define

tables before using them contrasts with other desktop software such as spreadsheet programs that do not make a sharp distinction between defining a spreadsheet and using it. Access, as well as most database software, emphasizes careful planning of tables as a cornerstone of application development.

1.1.1 Tables

Access supports most of the relational database concepts presented in textbook Chapter 3. Access uses the terminology of table, field, and record rather than table, column, and row used in textbook Chapter 3. Each field has a specified data type along with a number of other properties specific to each data type. Access supports entity integrity using primary keys and referential integrity using relationships and lookup fields. You can designate rules about referenced rows for relationships. As you will see, Access provides a number of powerful and convenient tools to define tables.

You should not create a database until it has been carefully designed. If you are using the textbook sequentially, you will not cover database design until Part 3. Therefore, the design of the database used in the lab chapters is given to you. If you are covering Part 3 before query formulation in Part 1, you should understand the process of database design. Database creation does not occur until after you have finished the conceptual data modeling (textbook Chapters 5 and 6) and logical design (textbook Chapter 7) phases of database development.

Populating a database (initial data entry) is an important part of the database creation process. Often, an information system cannot be used until its database is initially populated. Because of the importance of database population, Access enables you to enter data directly into a datasheet and/or import data from other sources. Both methods are described in lab Chapter 2.

1.1.2 Application Objects

After your tables are created, queries can be formulated to extract data. A query is a request for data to satisfy decision-making requirements. Access performs searches in the database using the specifications in a query. Access provides the industry standard Structured Query Language (SQL) as well as the visual tool Query Design to formulate queries. Both tools are described in Chapter 3.

Queries are powerful in Access because they can be embedded in applications. An important kind of application in Access is a data entry form. A form provides convenient data entry especially as compared to tediously entering data directly into the rows and columns of a datasheet. Access provides a special kind of form known as a hierarchical form to allow data entry consisting of a fixed and a variable part. Forms, especially hierarchical forms, can have complex relationships to tables. To specify complex relationships, Access provides a wizard that generates a queries using specifications given in the wizard. You also can write the queries directly in SQL or Query Design.

After entering data, a user may view the data as a report either displayed or printed. Reports can be formatted precisely to meet decision-making requirements. Reports can utilize nesting so that a user can follow data relationships easier than in a flat datasheet. Like forms, reports can utilize data from many tables. A query can be used to specify relationships between the data on a report and the underlying tables. Access provides a wizard to generate the query or you can use SQL or Query Design to write the query directly.

In the chapters ahead you will be creating queries, forms, and reports. These lab chapters complement the concepts presented in Chapters 3, 4, and 5 of the textbook.

1.1.3 Advanced Application Development Objects

To handle more complex application requirements, Access provides macros and modules to customize your applications. A macro can eliminate cumbersome repetitive procedures by replacing a set of commands with a single command. A module consists of program statements and procedures written in the Visual Basic for Applications (VBA) language. A macro is faster to develop but slower to execute than a module. However, a module provides more control than a macro. Macros are covered in Chapter 8. Modules are no longer covered in the lab book because macros can be developed to perform many tasks that previously required module development due to major macro extensions in Access 2010 and 2013.

1.1.4 Viewing Database Objects in the Access 2013 Ribbon

In Office 2007 and later, a ribbon is a functional grouping of buttons and drop-down lists that are relevant to particular tasks. Figure 1 shows the Access 2013 ribbon for a database with five tables and no other objects. The **Create** tab displays buttons relevant to table creation and maintenance. The buttons are grouped into four categories: Tables, Forms, Reports, and Other. The Other group allows creation of queries and macros.

The navigation pane below the ribbon in Figure 1 lists the objects in a database. Since this database only contains tables, other objects are not viewable. Other objects that can be viewed are queries, forms, reports, macros, modules, and class modules. The last three objects are coding objects. To view functions relevant to a database object, you select the object and right click on the mouse to show a menu of relevant commands. For example, if you right click on the *Customer* table, you can open the table in Design view as shown in Figure 2. Access provides several views for most objects. For tables, Access provides the Datasheet view to see the rows of a table as shown in Figure 3.

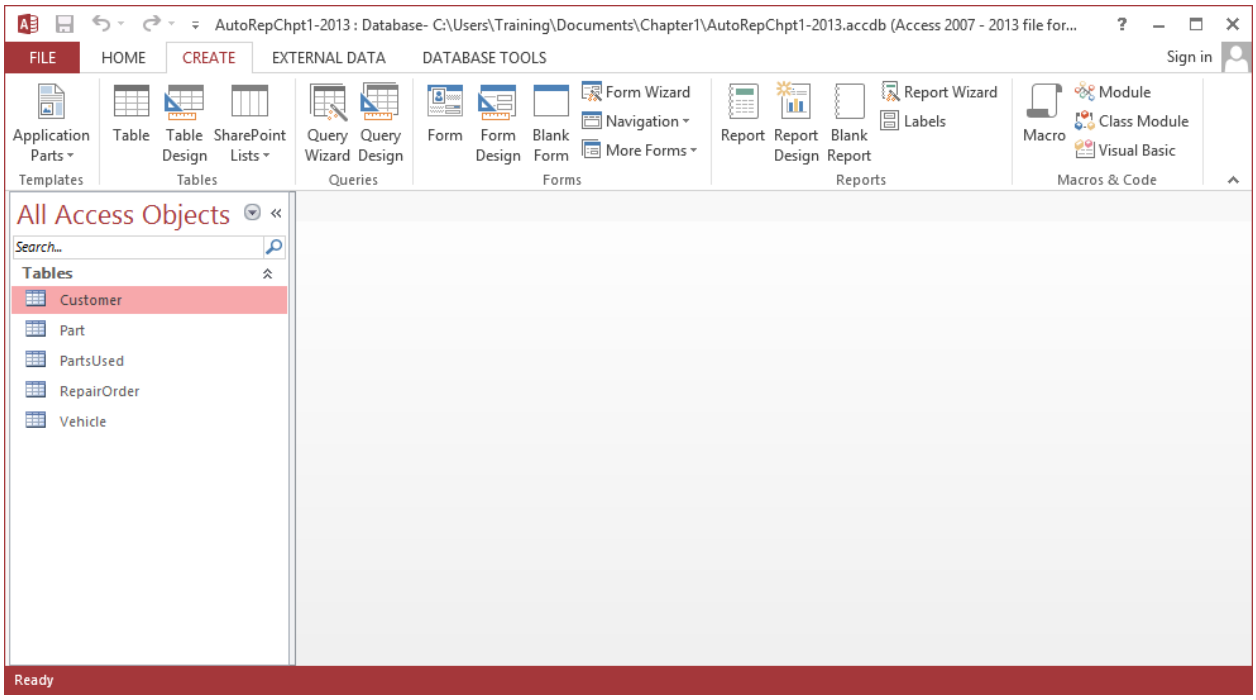


Figure 1: Access 2013 Database Window with Ribbon and Navigation Pane

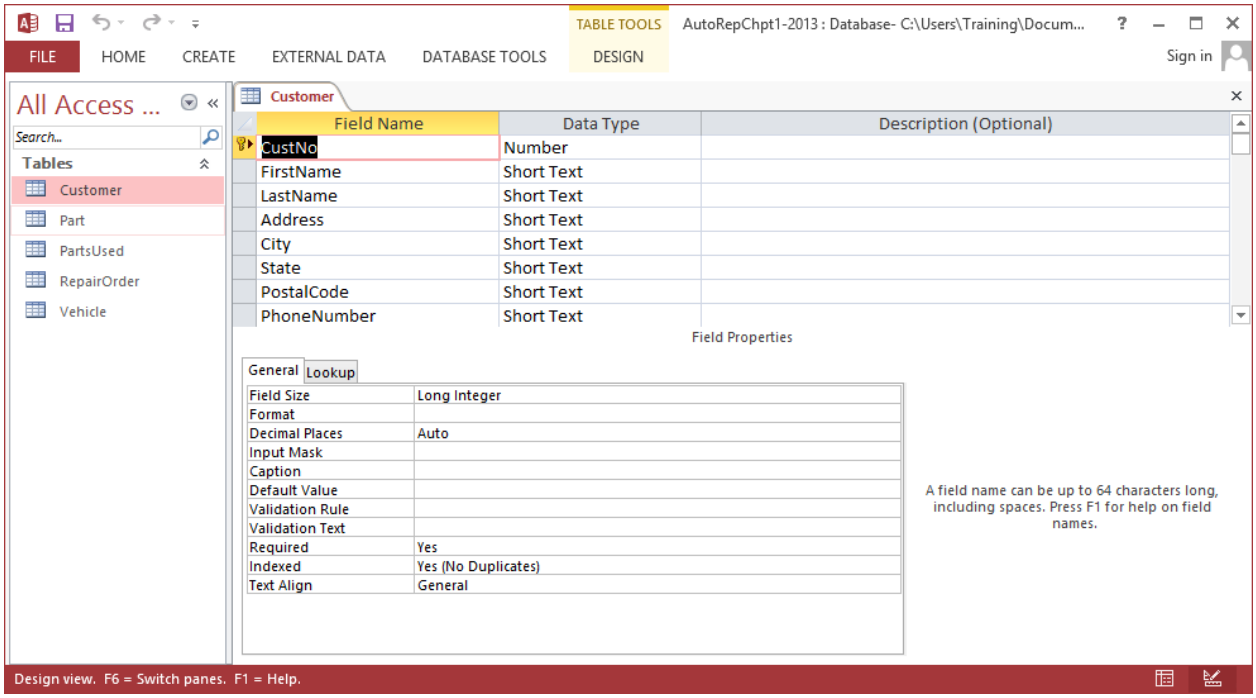


Figure 2: Customer Table in Design View

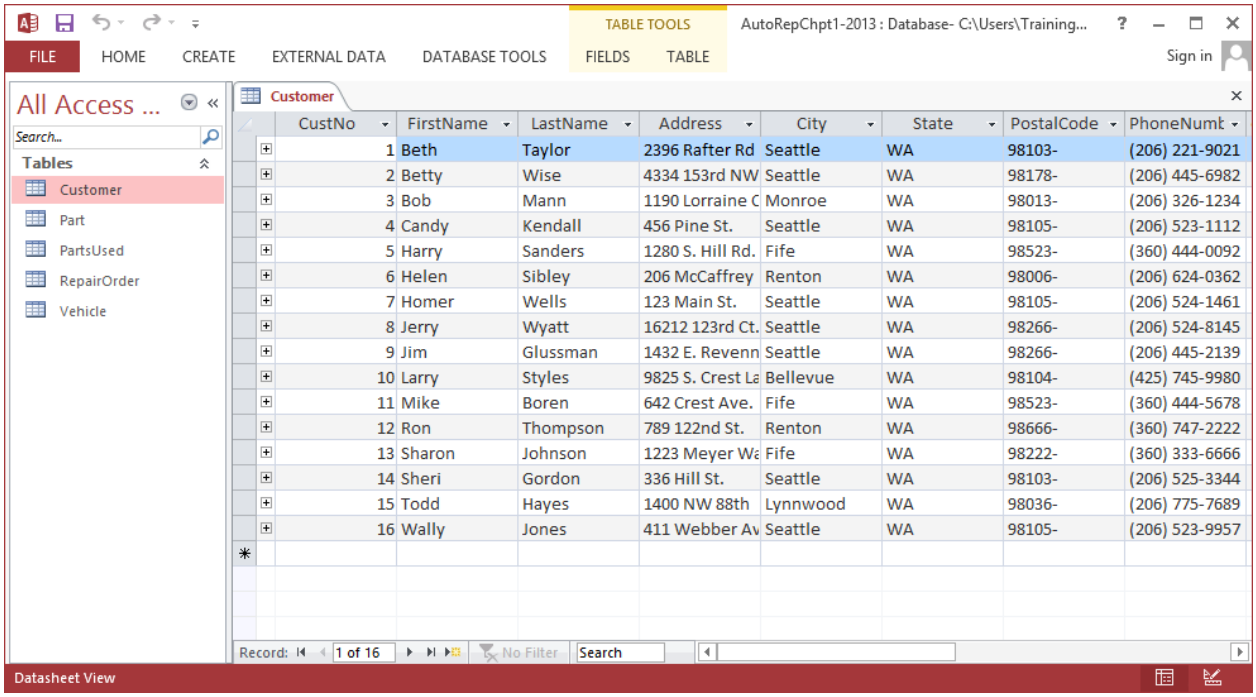


Figure 3: Customer Table in Datasheet View

1.2 Field and Table Properties

Access provides numerous properties for tables and fields. These properties allow you to specify how users can interact with your database. Before you create tables, you should be familiar with these properties so that you can specify values for them. This section discusses important properties that you will use in Chapter 2.

1.2.1 Basic Field Properties

The most important field properties, the *Field Name*, *Data Type*, and *Description*, are visible in the upper half of the Table Design window (Figure 2). Field names are usually short (less than 20 characters) because they appear in datasheets and queries. Descriptions provide more detailed documentation about fields. Data types determine the kinds of data accepted and the operations that can be performed. Here are more details about these properties:

- The first character in a field name cannot be a blank. In addition, you cannot use a period (.), an exclamation mark (!), or a square bracket ([ ]) in field name. Standard with all relational DBMSs, field names within a table must be unique.
- The *Data Type* property provides a list of choices. Some of these choices are specific to Access databases. For example, the "AutoNumber" data type is unique to Access.
- Although the description is optional, you should complete it for most fields. Descriptions do not appear in datasheets.

Description of Data Types

Choosing an appropriate data type for a field is an important choice. An inappropriate choice can allow incorrect values in a field and make a field difficult to manipulate. For example, if you choose "Text" for a field containing dates, you will not be able use the Access functions for date arithmetic. Table 2 describes Access data types appearing in the pull-down list. To learn more about these data types, while in table design view, move the mouse arrow to the data type column and type the **F1** key to access the reference window for data types.

Table 2: Brief Description of Access Data Types

Data Type	Description
Short Text	For text entries as well as numeric entries that do not require calculation such as addresses and phone numbers. This field size can be up to 255 characters.
Long Text	For extended text such as notes or comments in the field. A Long Text field can contain a maximum of a gigabyte. Also known as Memo.
Number	For numbers when numbers require mathematical operations. For example, use the Number data type for inventory quantities or expense amounts. The Number data type has Field Size options for floating point (Single and Double) and integer (Byte, Integer, and Long Integer) numbers.
Date/Time	For entering dates and times. For example, entering a customer's last order date.
Currency	For entering monetary amounts. These numbers will be displayed with $ signs.
AutoNumber	An integer data type in which Access automatically generates numbers. This data type is considered a "number" data type for linking fields. This data type is most often used for primary key fields.
Yes/No	For a field to accept a Yes or No answer from a user
OLE Object	The OLE Object data type is supported for compatibility with older Access versions. The Attachment data type replaces the OLE Object data type.
Hyperlink	For linking an object to another application or website by entering its path or web address in the field.
Attachment	For storing images, spreadsheet files, documents, charts and other types of supported files. Attachment fields are compressed to save space. New feature in Microsoft Access 2007
Calculated	Results calculated from one or more fields of the same table. New data type in Access 2010
Lookup Wizard	For enabling a combo box or list to appear in the field, from which a user selects search criteria. Used for foreign key fields and fields with codes such as state abbreviations.

1.2.2 Other Field Properties

Most other field properties depend on the data type selected. When a field name and a data type are entered in the Table Design window, a list of field properties appears at the bottom (Figure 4). Access provides default values for some field properties. Based on the decisions you make in the database design process, you may decide to override these default values.

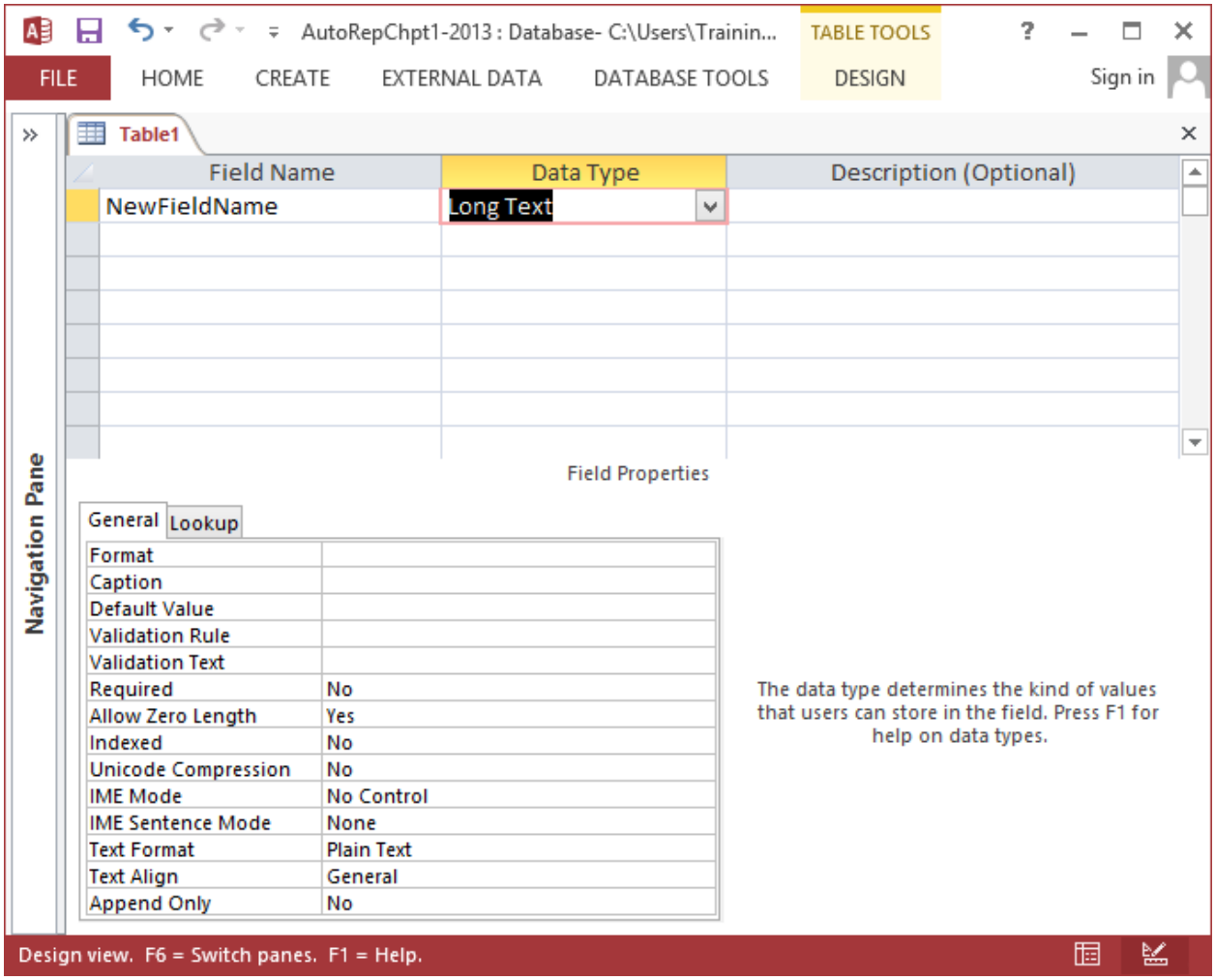


Figure 4: Table Design Window with Field Properties for the Text Data Type

The **General** and **Lookup** tabs contain the field property options related to the chosen data type. The **General** tab is used for all data types. The **Lookup** tab is for lookup fields only. When you click on any field property value area, a description appears on the right side of the window. In this chapter, we will be describing options listed under the **General** tab unless mentioned otherwise.

In the remainder of this section, the more complex field properties are described. Appendix A contains a brief summary of each field property and a cross-reference of field properties by data type.

Field Size

For the "Text" data type, the value is a number from 0–255. If the data type is "AutoNumber", a pull-down list provides options of "Long Integer" and "Replication ID". For the data type "Number", there are seven choices based on the kind of number and the range of values.

Input Mask

An input mask specifies a data entry pattern. Access uses input masks to inform users of the pattern of data entry and to ensure that the data entered conform to the pattern. You can use input masks for fields containing dates, times, social security numbers, and phone numbers.

Format

The *Format* property controls the way data are displayed in a datasheet, a form, or a report. The choices for the *Format* property depend on the data type. For example, there are a variety of numeric formats such as "Currency", "Percent", and "Scientific".

You should be aware that the values you select for the *Format* and the *Input Mask* properties can conflict. For example, an input mask for entering dates as short dates ("mm/dd/yy") conflicts with a format to display the field as a medium date ("day-month-year"). If these properties conflict, users will not be able to modify data in the table.

Default Value

The *Default Value* property allows you to specify a value when the user does not provide one. This property can be specified as a constant or an expression involving functions, operators, and constants. For example, if you set the value of this property to "WA" (for Washington state), the value "WA" automatically appears without the user entering it.

You also can use functions and expressions such as the `Date()` function for a column that stores a date and the expression `Year(Date())` for a column that stores the year. Expressions are combinations of operators, functions, constants, and field names that provide a value when evaluated. Note that you can use the expression builder tool (invoked by clicking on the ellipsis [**…**] button) to help formulate expressions.

Validation Rule

The *Validation Rule* property allows you to specify simple rules that restrict values entered by a user. Validation rules are logical expressions (i.e., result in a true or a false value) built from Boolean combinations (AND, OR, NOT...) of comparisons. For example to restrict a field containing year values, you can define a validation rule such as `BETWEEN 1990 AND 2013`. To restrict a numeric field to be greater than zero, use the validation rule `> 0`. Note that the field name is not mentioned in field validation rules as demonstrated in the two examples. You also can use the expression builder to help formulate validation rules by clicking the ellipsis (**…**) button.

Allow Zero Length and Required

The *Allow Zero Length* and the *Required* properties are similar. A zero-length string has no characters. It is used for text fields when the value does not exist. A zero-length string is specified by double quote marks with nothing inside "". The *Required* property determines whether null values are permitted. A null value in a field indicates that the value exists but is unknown at this time. Setting the *Required* property to "No" allows null values. The *Required* and the *Allow Zero Length* properties can interact in subtle ways. For a good explanation of the interaction, refer to the Access help documentation by clicking the **F1** key in the property value area of either property.

1.2.3 Table Properties

Table properties are not as numerous as field properties. You will use two table properties, primary keys and validation rules, in Chapter 2. This section provides background about these properties.

Primary Keys

Access supports primary keys, as discussed in textbook Chapter 3. Recall that a primary key is a field or combination of fields with unique values. Primary keys ensure entity integrity because each record can be identified through its primary key value. As previously mentioned, Access indirectly supports primary keys with the "AutoNumber" data type. Fields with the "AutoNumber" data type are guaranteed to be unique. Access directly supports primary keys through the primary key button. Lab Chapter 2 describes how to use the primary key button to designate primary keys.

After selecting the primary key for a table, Access shows the primary key using a small key icon in the left margin. For combined primary keys, the key icon appears by both fields as shown in Figure 5. The key icon highlighted for multiple fields means that the combination of fields is unique, not the individual fields.

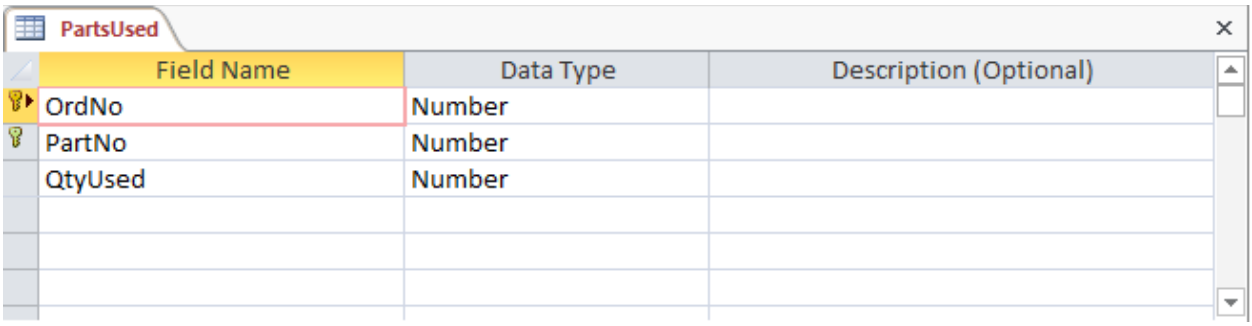


Figure 5: Combined Primary Key Fields

Table Validation Rules

Access distinguishes between field validation and table validation rules. A field validation rule can involve only one field. For example, a field validation rule can be used to restrict an amount field to be greater than 0. A table validation rule can involve more than one field. For example, a table validation is necessary to require a date-authorized field to be later than a date-requested field. In Chapter 2, you will use a table validation to make a similar table validation rule.

The *Table Validation Rule* property is available through the Table Properties window (Figure 6). You can invoke this window through the **Table Properties** command as described in Chapter 2.

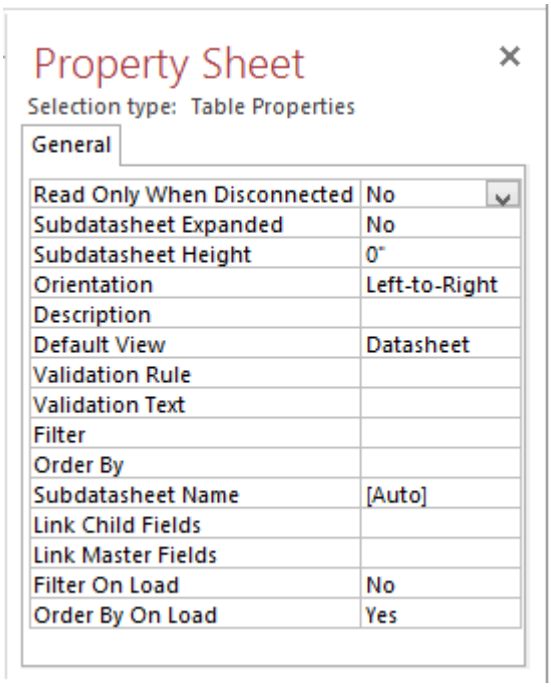


Figure 6: Table Properties Window

1.3 Tools for Object and Property Creation

Access provides wizards and the expression builder to help you create certain objects and properties. A wizard is a sequence of windows that guide you to create an object or property value. Access has wizards for creating many objects such as tables, forms, and reports as well as complex properties such as input masks. The result of a wizard is an object or property value that you can further customize. The expression builder does not guide you like a wizard. Instead, it provides a structured window for building complex expressions. The wizards discussed here are the most common ones that you will be using to create the auto repair database in Chapter 2.

1.3.1 Field Property Wizards

The Input Mask Wizard allows you to specify several parts of an input mask. Examples of input masks for common fields such as zip codes and phone numbers are provided. Instructions are provided in Chapter 2 where the Input Mask Wizard is used.

With the Lookup Wizard you can create a list of choices for a field. The list of choices comes from either another field or a fixed list of values. The Lookup Wizard allows you to enter the values in the list or select a field from another table with allowable values. Instructions are provided in Chapter 2 where the Lookup Wizard is used.

For both wizards, you must be careful to install them. The typical installation option that Access provides does not install all of the wizards. If a wizard does not appear as described in later chapters, you need to install it using your Access CD-ROM. Insert your CD-ROM and perform an incremental installation to add new components. Make sure that all of the wizard components are checked.

1.3.2 Tools for Table Creation

Application Parts allow you to create a predefined table and related objects for selected domains. In the **Application Parts** dropdown list in the **Create** tab, Access provides a list of application parts from which to choose. You may customize the table and application objects further to your requirements.

The Import Wizards guide you to adding a new table or importing data to an existing table. The import wizards are available as buttons in the **External Data** tab of the Access ribbon. There are buttons for importing data from other Access databases, Excel spreadsheets, text files, XML files, Sharepoint files, and other types of databases. Each import wizard walks you through the steps to locate a file with data and then load it into the database. When the wizard is finished, you may customize the table further to your own requirements. Full instructions are given in Chapter 2.

1.3.3 Expression Builder

The expression builder provides a window of structured choices to define complex expressions (combinations of operators, functions, fields, and constants). The expression builder can be used to create properties in most objects. For example, the expression builder can be used for the *Default Value* and the *Validation Rule* properties. As you work through subsequent chapters, you will be using the expression builder frequently. The following is an overview about how to use this important tool:

- When you click in an area where the expression builder is available, three dots appear **"…".** Clicking on these dots invokes the Expression Builder window.
- The Expression Builder window appears on your screen containing an empty window to type expressions (Figure 7). Buttons directly below the window aid you in writing the expression. Clicking a selection in the leftmost pane displays a different set of choices in the rightmost pane.
- Figure 8 shows a date string expression appropriate for a *Default Value* property. The value in the expression area was created after double-clicking the `Date$` function in the rightmost pane.

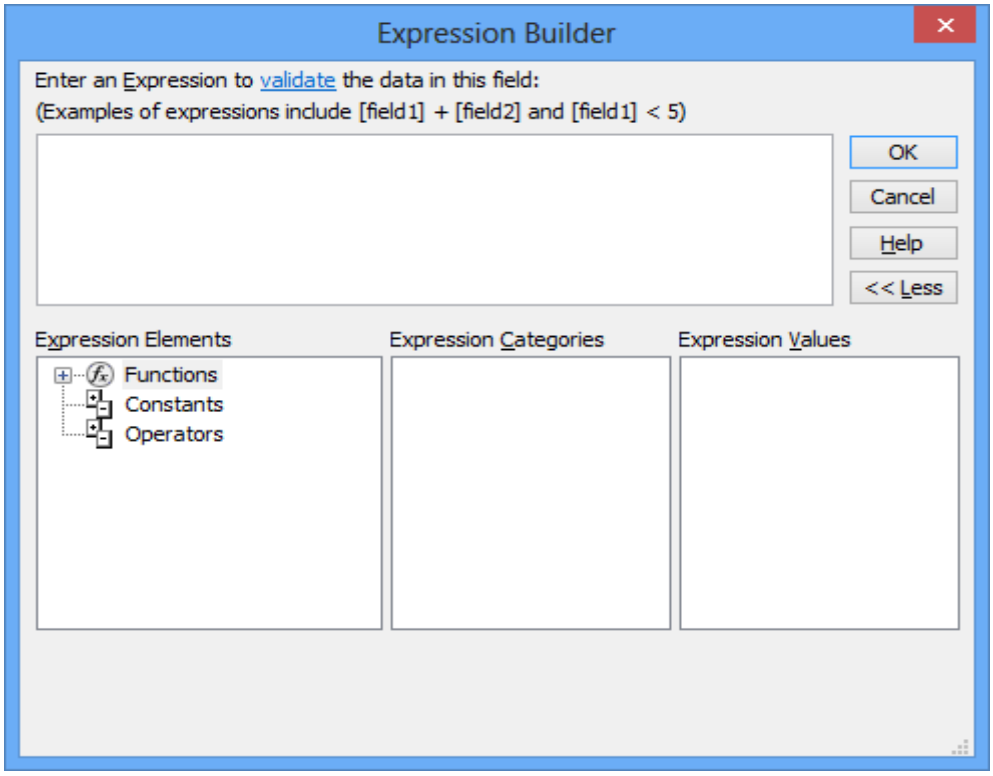


Figure 7: Expression Builder with Empty Window

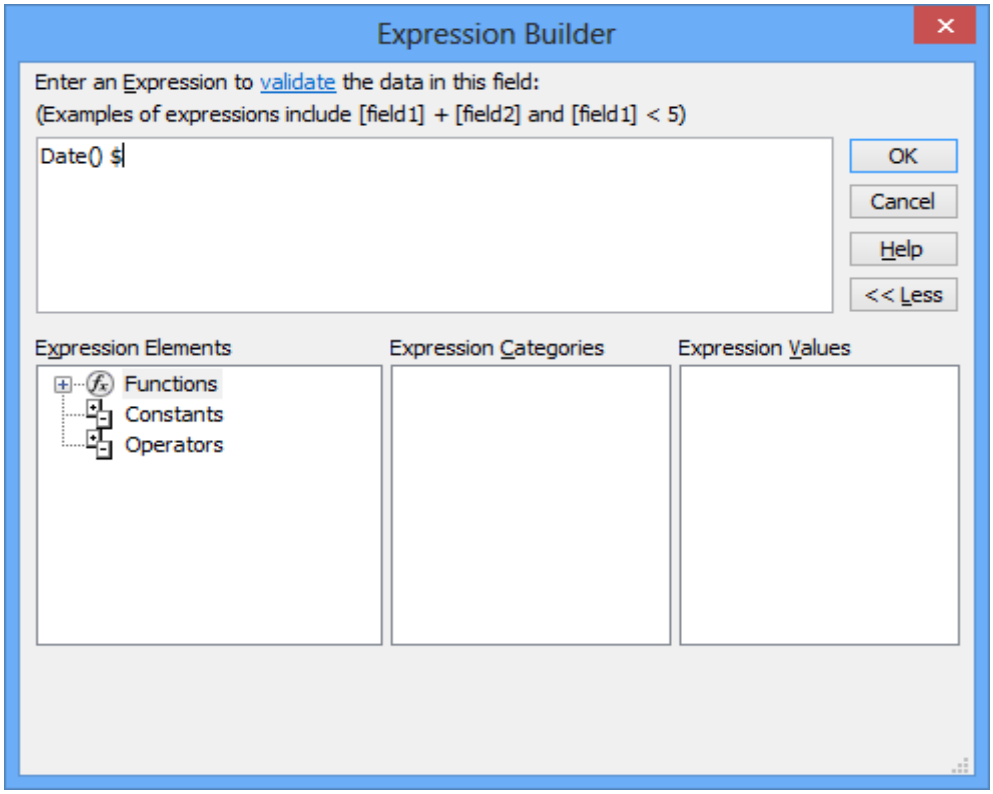


Figure 8: Expression Builder Containing the Date$ Function.

1.4 Access Interface

The interface has changed less since the previous version (Access 2010). Access 2013 has a similar interface to the other Microsoft Office 2013 applications. This similarity allows most users to quickly become productive with Access. To provide an overview of some of the details that may be new to you, this section discusses several parts of the Access interface that you will be using in subsequent chapters.

1.4.1 Ribbon and Navigation Pane

Access uses the ribbon, a new interface feature in Office 2007, to manage tools for database creation and usage. A ribbon is a functional grouping of buttons and drop-down lists that are relevant to particular tasks. Figure 9 shows the Access 2013 Ribbon for a database with five tables and no other objects. The Access 2013 interface also uses the Quick Access toolbar and the Navigation pane. The Quick Access toolbar supports fast access to common tools. The Quick Access toolbar can be customized by using the drop down list to the right of the toolbar. Objects appear in the Navigation Pane. Figure 9 shows a list of tables, the only objects in the database. When other types of objects are added, the Navigation pane displays them. The Navigation pane can display lists of tables, forms, reports, and coding objects (macros and modules).

Some users find the Ribbon too distracting or using too much space at times. You can toggle the Ribbon by using one of these methods:

- Press CTRL-F1.
- Double-click on any of the tab labels.
- Right-click on the row of tab labels or any button within a tab and choose **Minimize the Ribbon** from the shortcut menu.

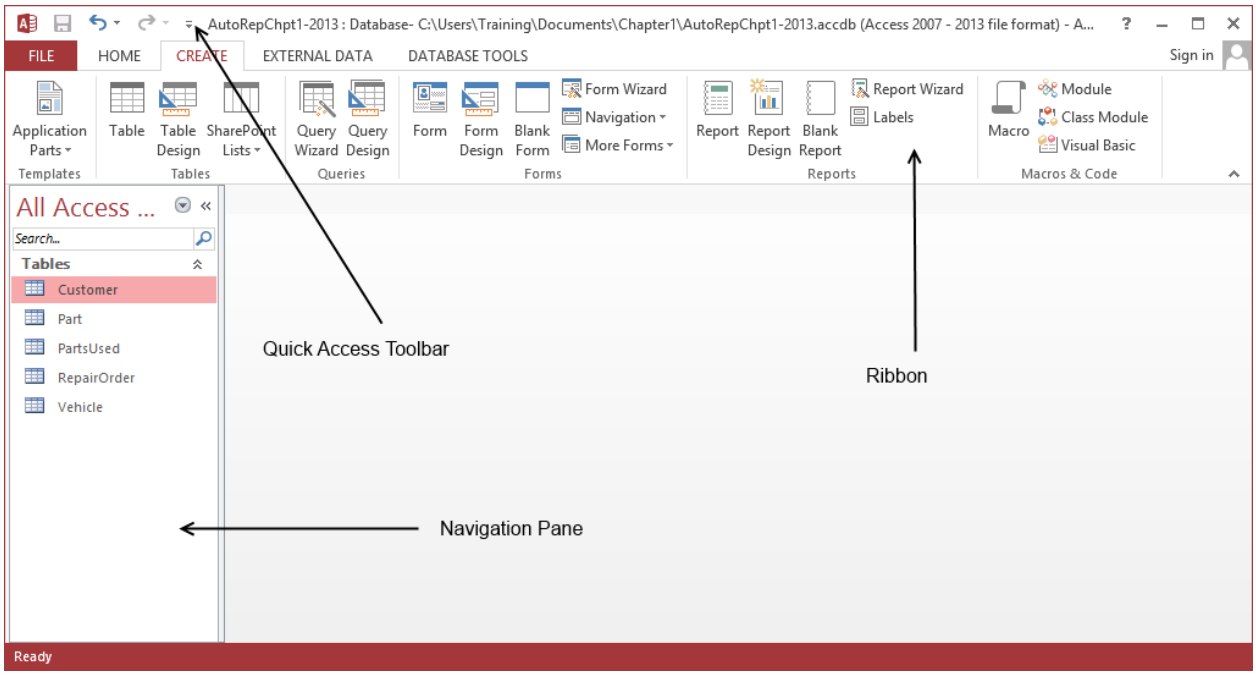


Figure 9: Basic Elements of the Access 2010 Interface

In Figure 10, the **Home** tab displays buttons relevant to most database objects. The buttons are grouped into seven groups: Views, Clipboard, Sort & Filter, Records, Find, Window, and Text Formatting. You can see a caption for a button by dragging the mouse over it. The buttons with small arrows at the bottom are dropdown lists providing choices after the arrow is selected.

The ribbon is context sensitive so that tabs and buttons are relevant to the object selected. In Figure 10, only items in the Records and Window groups are relevant to the current selection. If a table is opened in Design view, other buttons become relevant as shown in Figure 11. The **View** item is now relevant because

the *Customer* table is open in Figure 11. Chapter 2 depicts the buttons and lists that apply to tables. Subsequent chapters depict the buttons and lists that apply to other objects.

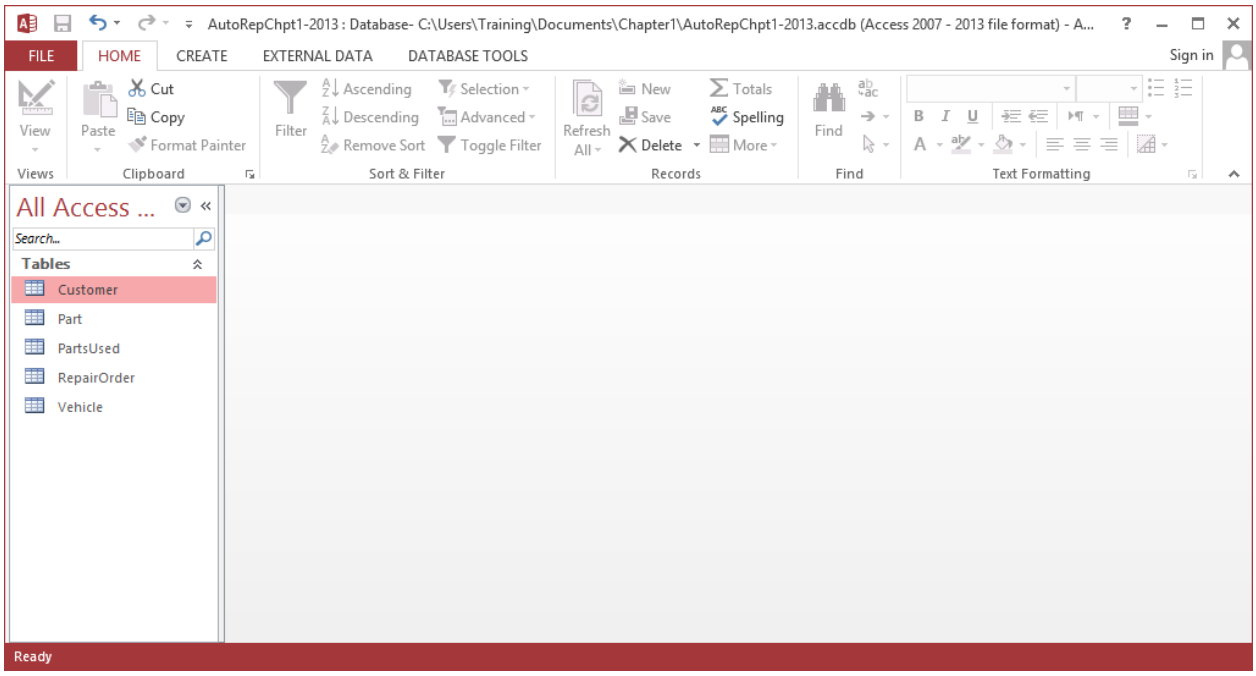


Figure 10: Home Tab in the Ribbon f or Selected Table

The View Toolbar () appears in the bottom right part of the window as shown in Figure 11. The views appearing in the View Toolbar are relevant to the object manipulated in the view pane. In Figure 11, the *Customer* table is opened in Design view. The Design view icon () is highlighted in the View Toolbar. You can switch views by selecting an alternative view in the View Toobar.

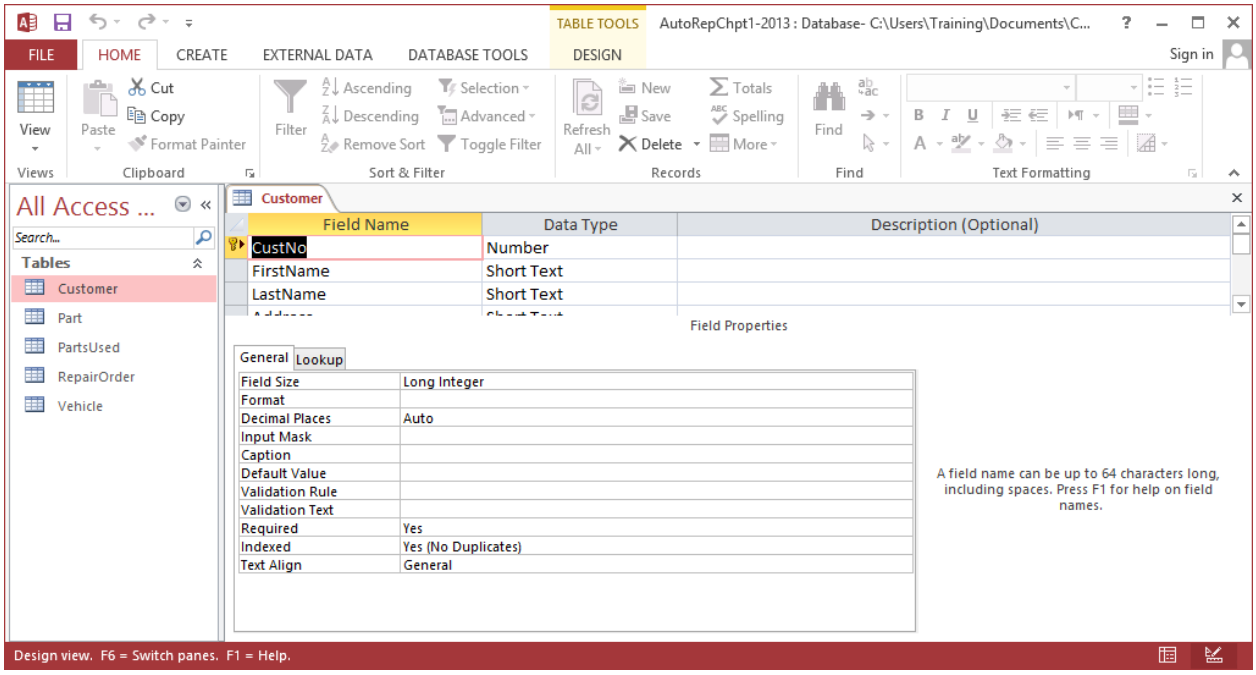


Figure 11: Access Window with Table Open in Design View

1.4.2 Navigating and Editing Tips

Navigation in Access is a little different than in other Office applications. The following tips may be useful to help you with the new parts of navigation.

Dragging and Dropping

This is a Microsoft Windows feature that enables one or more items to be moved by dragging it from one place and dropping into another place. First you select the item to be moved. Next, point the mouse arrow at the item and hold the mouse button down while moving the item to another place. After it has been moved, release the mouse button.

Correcting Mistakes

In table design view or open view, you can move the cursor to a field to correct it. Alternatively, you can use the **Tab** key and arrow keys. The yellow rectangle in the field name area and the yellow square in the margin indicate the current row as shown in Figure 12. When a property becomes highlighted, you may either use the **Delete** key or put the cursor where the correction belongs. You also may use editing buttons in the **Home** tab in the Ribbon.

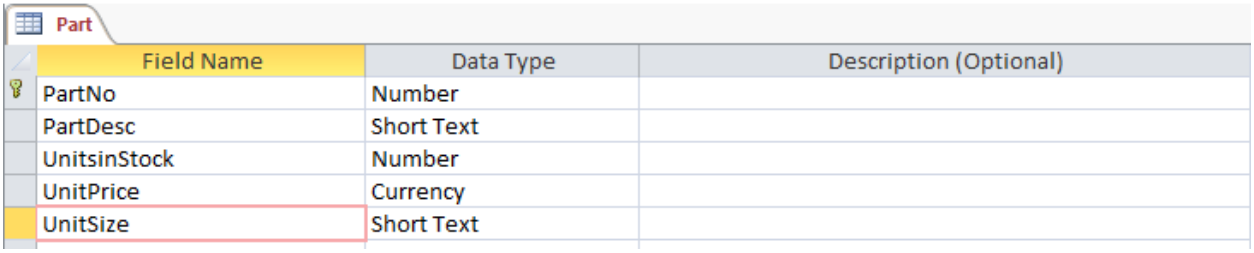


Figure 12: Cursor in *UnitSize* Field in Table Design View

Adjusting Column Widths

If a field seems too narrow or wide in a datasheet (Figure 13), you can adjust the width with the mouse. Begin by pointing at the column-line in the title bar. When the crossbar (double left-right arrow) appears, hold down the left mouse button and move the line to adjust the field size. You also can adjust the field width with the *Field Size* property.

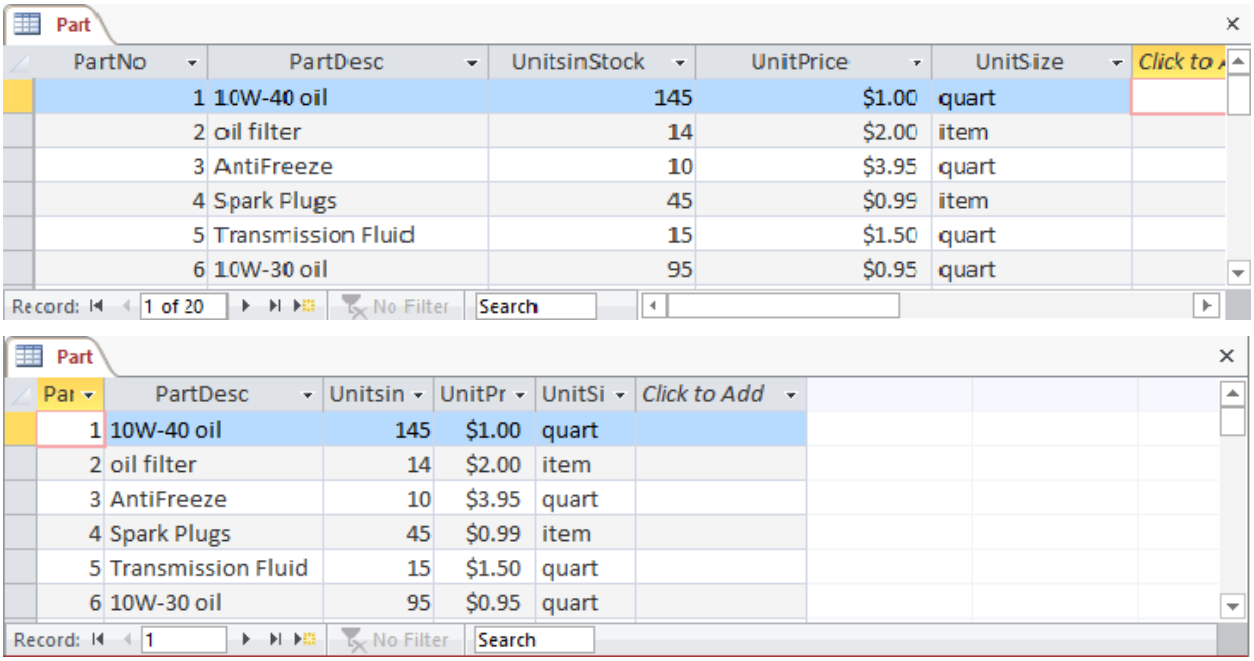


Figure 13: Narrowing Fields in a Datasheet

Additional Formatting

You also may use buttons in the Font group of the Home tab to change the appearance of a field value. Alternatively, you can click on the gray left margin to highlight an entire row (or the gray top title bar to highlight an entire column) and click the right mouse button to display a menu.

Moving Columns

To move a column on a datasheet, click the mouse on the field title. A downward arrow appears and the entire column becomes highlighted, as demonstrated in Figure 14. Click the column again and a bounding rectangle appears. While holding the mouse down, move the rectangle and the column will go to the desired position, then release the mouse. You also can cut and paste a column to move it. Note that you cannot move records in a datasheet. Cutting a record is the same as deleting it.

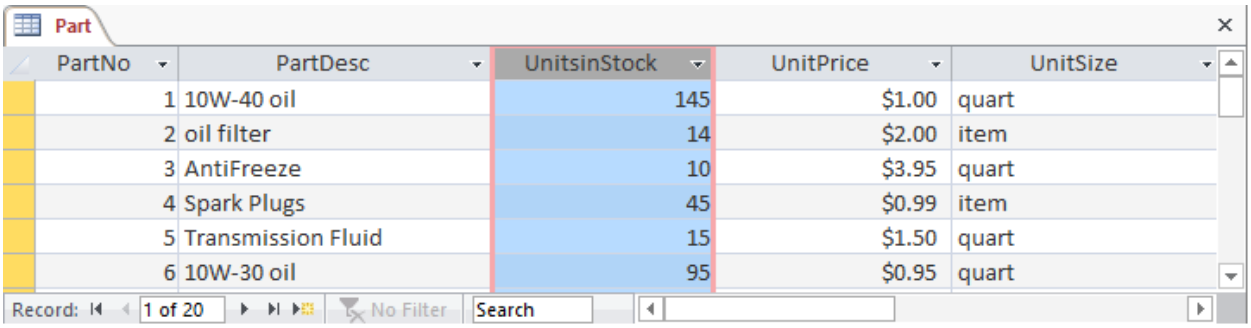


Figure 14: Highlighted Column Ready to Move

Navigating a Datasheet

Navigating a datasheet is easy with a little practice. To navigate among records in a datasheet with too many records to fit in the window, you can use the navigation bar at the bottom of the datasheet, as shown in Figure 15. Using the navigation bar is often faster than using the **Tab** and arrows keys. You also can type **Ctrl Home** (press the **Ctrl** and **Home** keys simultaneously) to go directly to the first record. Similarly you can type **Ctrl End** to go directly to the last record. You also use buttons available in the Records, Sort & Filter, and Find groups in the **Home** tab to manipulate records in a datasheet. To display a field menu, you can click the right mouse button when the cursor is in a field.

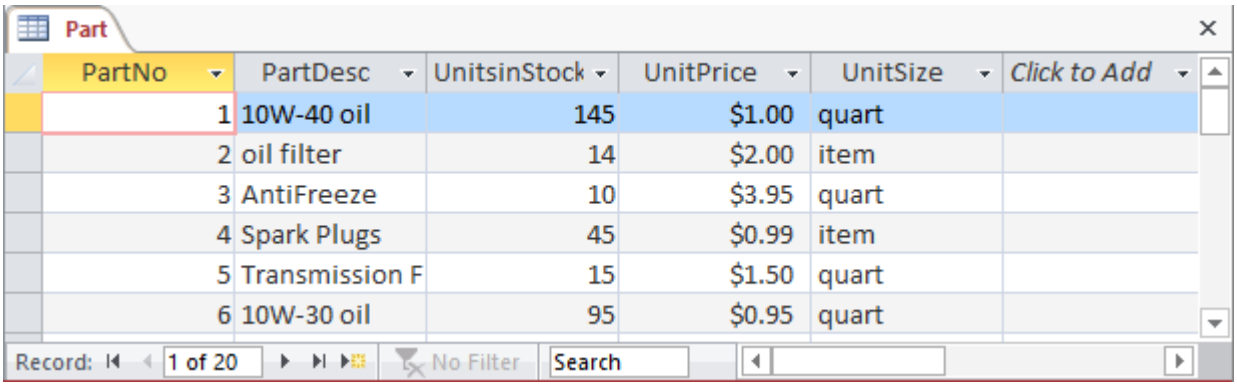


Figure 15: Datasheet with Navigation Bar at the Bottom

Freezing Columns

If you want one or more columns in a datasheet to stay in view while scrolling, you can freeze the columns. Retaining columns is especially useful for identifying columns. In the top datasheet of Figure 15, the first three columns (*CustNo*, *FirstName*, and *LastName*) are not in view. After selecting the column(s) to freeze, the entire column(s) becomes highlighted. Click the right mouse button to use the **Freeze Column** command. The bottom datasheet in Figure 16 shows the first three columns frozen.

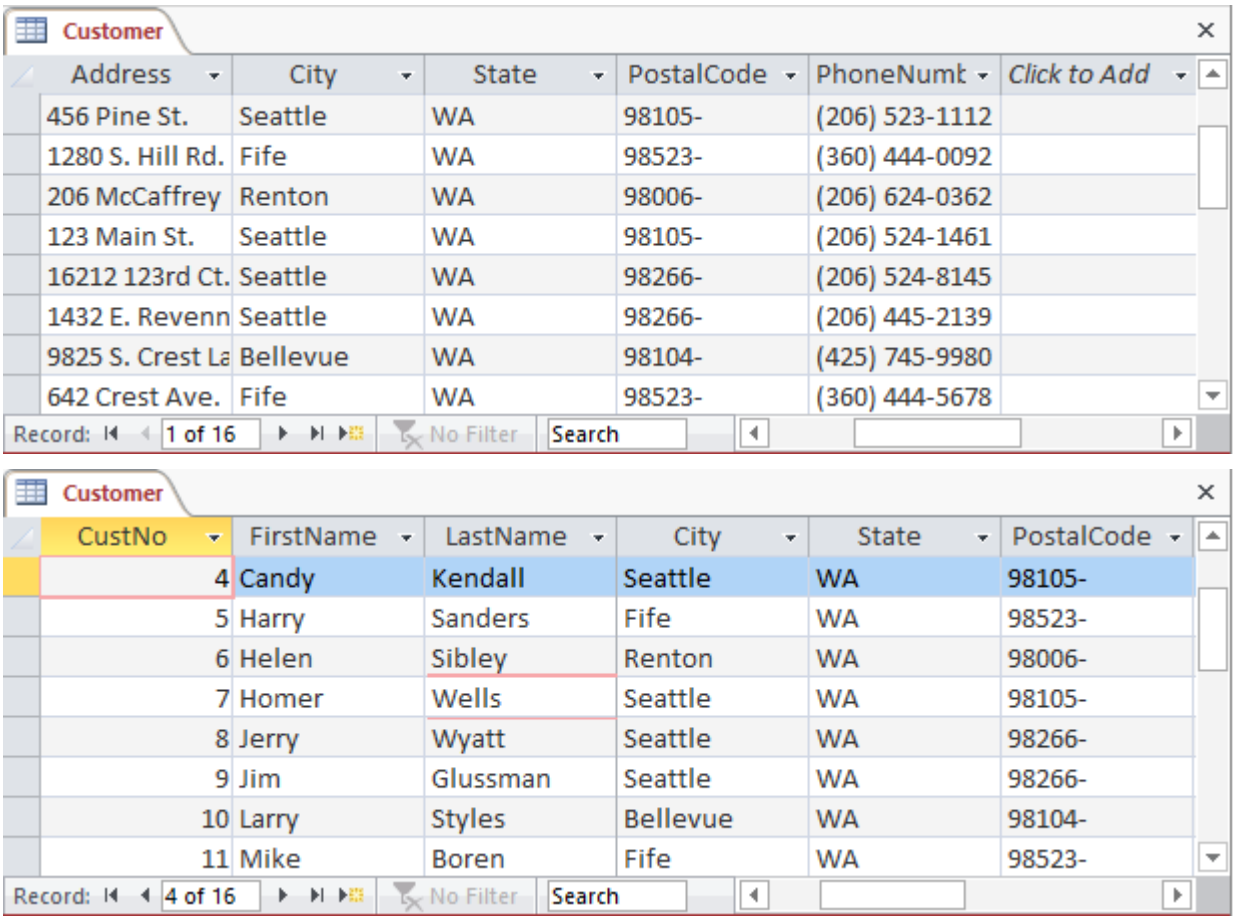


Figure 16: Freezing Several Columns in a Datasheet

1.5 Overview of the Auto Repair Database

The auto repair database is used throughout the lab chapters. To help you become familiar with this database, this section describes the tables and the relationships as well as the functions supported by the database.

1.5.1 Functions Supported by the Auto Repair Database

The auto repair database supports the operations of a small repair shop. Customers bring vehicles to repair. The repair shop completes a service order with a list of parts required to perform the necessary work. As each part of a service order is completed, the number of parts used is recorded. When a service order is finished, the completion time and the total charge are noted. To support marketing efforts, a list of customers and vehicles is maintained. If a service order is for a new customer or vehicle, the customer is added to the database. To facilitate inventory, a list of parts is maintained. Parts are usually added to the database if many service orders require parts not stocked by the repair shop.

1.5.2 Auto Repair Tables

The auto repair database consists of five tables. The tables are described below with the table names italicized:

1. *Customer*: This table consists of data such as the customer's name, unique customer number, address, and phone number.

2. *Vehicle*: Vehicle data include the serial number of the car, its year and model, the license number, the state, the number of cylinders, and the customer number of its owner.
3. *Part*: This table contains the part description, the part number, the cost, the size, and the quantity in stock.
4. *RepairOrder*: This table consists of data taken directly off a repair order form including the repair order number, the date, the car odometer reading, the car serial number, the time brought into the shop, and the time out of the shop.
5. *PartsUsed*: This table contains the repair order number, the part number, and the quantity used.

1.5.3 Auto Repair Relationships

As you also may recall from textbook Chapter 3, common columns link tables. These common columns, known as foreign keys, are primary keys in other tables. For example, the foreign key *CustNo* in the *Vehicle* table is the primary key of the *Customer* table. Foreign keys obey the referential integrity rule described in textbook Chapter 3. A foreign key value can be either a value found in the primary key table or null.

An Access relationship establishes a referential integrity constraint. Access provides the Relationship window to define relationships. Figure 17 shows the relationships for the auto repair database. You establish a relationship by dragging the primary key column of the table to be linked from (the primary key table) into the table to be linked to (the foreign key table). In addition to establishing relationships, Access allows you to define rules for referenced rows, as described in textbook Chapter 3. Lab Chapter 2 provides practice with establishing relationships and referenced row rules for the auto repair database.

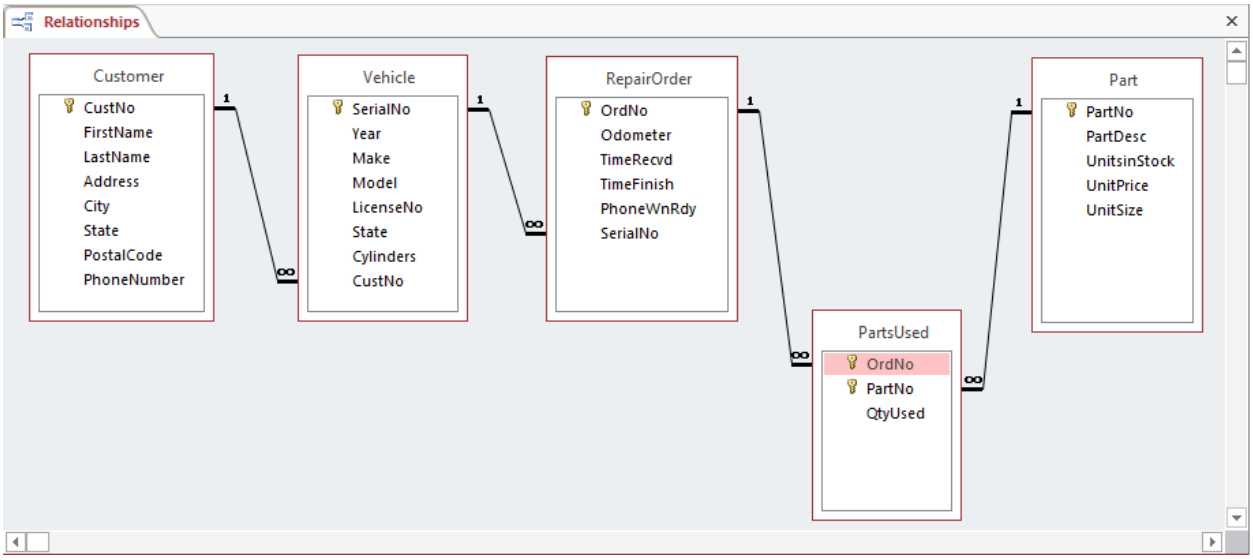


Figure 17: Relationship Window for the Auto Repair Database

Students new to databases sometimes think that simply designing a *query* will link the tables (queries are covered in Chapter 3). A query does link tables, but this link exists only for that particular query. On the other hand, when you relate tables, you are linking these tables permanently (or at least until you remove the links).

1.6 Database Maintenance, Printing, and File Formats

The compaction and repair features may help you maintain a database in its most efficient state. These commands are found in the **Info** menu under the **File** tab (see Figure 18). In addition, Access allows you to print most database objects and components.

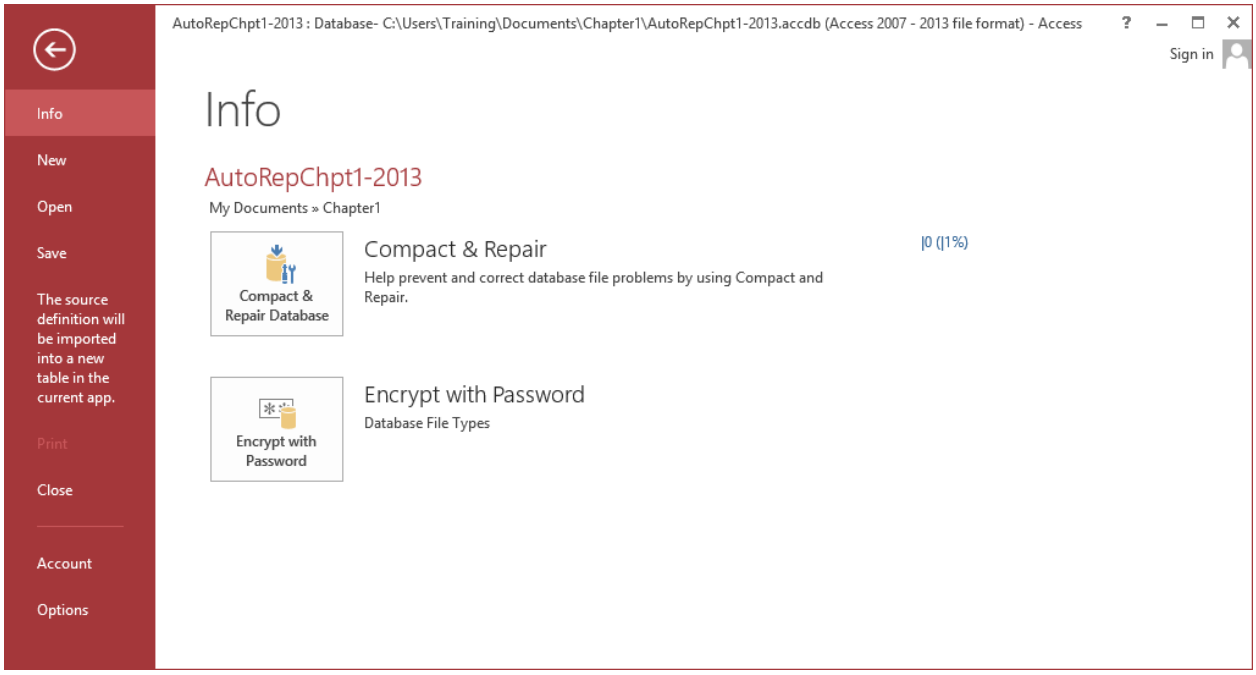


Figure 18: Info Menu

1.6.1 Compacting and Repairing a Database

When you delete tables in a database, the database will eventually become fragmented. This results in inefficient use of hard drive disk space. The reason for compacting a database is to defragment the database file to release disk space on the computer's hard drive. The compact feature makes a copy of the database and reallocates the database on the hard drive disk. Compacting a database is very easy. Simply click **File → Info → Compact & Repair Database…**. Note that during compaction, Access will reset the AutoNumber value if a record had been deleted from the database that contained an AutoNumber field.

Also, if your database is damaged, Access will detect this and give you a message when you attempt to open or compact the database. Access will then ask you if you want to repair it. However, there may be other times when Access may not detect a damaged database. If you feel that a database is behaving incorrectly, you also can repair the database by clicking **File → Info → Compact & Repair Database…**.

1.6.2 Printing Database Objects

You can print the contents of database objects by clicking the **File → Print** menu command. For example to print a table's datasheet, select the table in the Navigation pane and use the **File → Print** menu command. To print details about an object's design, you can use the **Database Tools → Database Documenter** command. This command allows you to select the objects and object properties to print.

1.6.3 Database File Formats

Access 2013 provides multiple file formats along with the ability to set the default file format and convert between file formats. You can set the default file format using the **General** button in the Access Options window (**File → Options…** command) as shown in Figure 19. By default, the format is Access 2007-2013 as shown in the "Default File Format" combo box in Figure 19. You can convert a database to a different file format using the **File → Save & Publish** menu. You can convert the current file format to the other two database file formats (Access 2000 and Access 2002 – 2003) as well as advanced formats for saving the database with a digital signature (Package and Sign), backing up the database, compiling the database into an executable form (Make ACCDE), and publishing the database to a Microsoft Sharepoint server (Sharepoint).

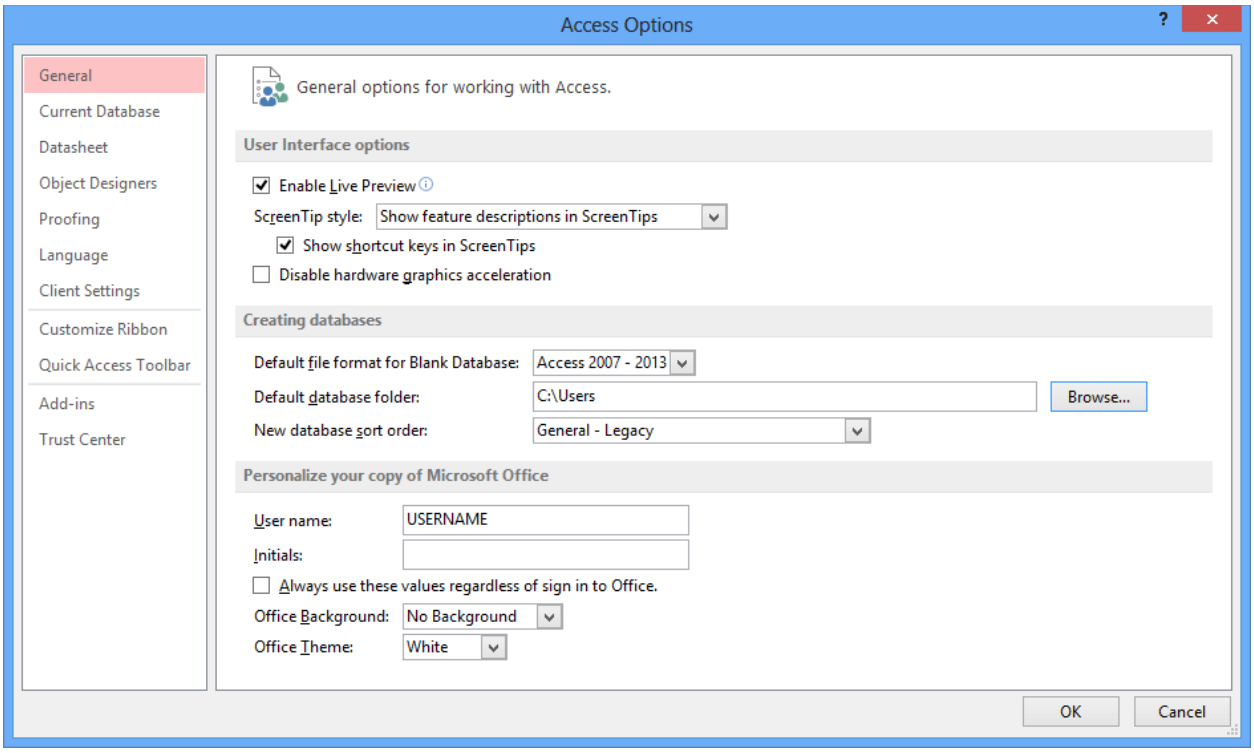


Figure 19: General Tab of the Options Window

You should determine the file format after you initially create a database. If you are creating a new database and do not need compatibility with previous versions of Access, you should use the Access 2007-2013 format. The Access 2007-2013 format will be compatible with future Access versions so you should have no problems converting to future versions. If you need to distribute your database to users with previous versions of Access, you can convert the database using the **Save & Publish** menu as previously described. However, you should be aware that Access 2013 features will be lost after converting to a previous version.

Table 3 lists features that may be unavailable if you do not work with the Access 2007-2013 format. In addition, there may be problems with libraries used in modules that you have written. You should consult the Access help documentation under the topic "File Format" for more details about file formats.

Table 3: New Features in the Access 2007-2013 File Format

Feature	Description
Multivalued fields	Most database programs, including earlier versions of Access, allow you to store only a single value in each field. In Access 2013, however, you can create a lookup field that allows you to store more than one value in a field.
Attachment data type	The Attachment data type replaces the OLE Object data type in earlier Access versions. The Attachment data type supports storage of all types of documents and binary files in a database in a compressed format.
Integration	Integration enables Access databases to work more fully with Windows SharePoint Services and Office Outlook 2007, 2010 and 2013. It also allows antivirus programs to inspect Access database files much more easily.
Offline data	You can take a SharePoint lists offline with one click by using Access 2013. You can work on your data in Access and then synchronize your changes, or reconnect with the SharePoint site at a later time.
History tracking	In Access 2007, you can set a property (AppendOnly) that forces Access to retain a history of all changes to a Long Text field. This feature also supports the versioning feature in Windows SharePoint Services 3.0 so that you can use Access to track changes in a "multiple lines of text" field that is stored in a SharePoint list.

Closing Thoughts

Chapter 1 has introduced Microsoft Access 2013 as a prelude to the labs presented in subsequent chapters. Microsoft Access 2013 is a powerful and easy-to-use desktop DBMS. In this chapter, you gained background knowledge about the objects, tools, and properties provided by Access. Tables are the fundamental objects around which an Access database is built. Access also provides other objects so that you can develop complete information systems. You also learned about the auto repair database, the database featured throughout the lab chapters. In subsequent chapters you will put this background knowledge into practice.

Chapter Reference

The chapter reference summarizes procedures described in the chapter. Since this chapter was mostly reference, only a few procedure summaries are provided.

Procedure 1: Drag and Drop (Section 1.4.2)

1. Select the item to move.
2. Point at the item and hold the mouse down while moving the item to another place.
3. After it has been moved, release the mouse button.

Procedure 2: Adjust Column Width (Section 1.4.2)

1. If a column seems too narrow or wide in a datasheet, you can adjust the width with the mouse.
2. Begin by pointing at the column-line at the gray title bar. When the crossbar (double left-right arrow) appears, hold down the left mouse button and move the line to adjust the field width.
3. You also can adjust the field width with the *Field Size* property.

Procedure 3: Moving a Column (Section 1.4.2)

1. To move a column on a datasheet, click the mouse on the field title. A downward arrow appears and the entire column becomes highlighted. Click the column again and a little box appears. While holding the mouse down, move the small box and the column will go to the desired position, then release the mouse.
2. You also can cut and paste a column to move it.

Note: You cannot move records in a datasheet. Cutting a record is the same as deleting it.

Procedure 4: Navigating a Datasheet (Section 1.4.2)

1. When there are too many records to navigate in a single datasheet window, you can use the navigation bar at the bottom of the datasheet to scroll among the records. Using the navigation bar is often faster than using the **Tab** and arrows keys.
2. You also can type **Ctrl Home** (press the **Ctrl** and **Home** keys simultaneously) to go directly to the first record.
3. Similarly you can type **Ctrl End** to go directly to the last record.
4. You also can use commands in the Records, Sort & Filter, and Find groups in the **Home** tab of the Ribbon to operate on rows.
5. Alternatively, you can click the right mouse button when the cursor is in a field to display a menu.

Procedure 5: Freezing a Column (Section 1.4.2)

1. Freezing a column(s) locks one or more datasheet columns to keep it in view while you scroll.
2. Select the column(s) to freeze by clicking the mouse on the field title. A downward arrow appears and the entire column becomes highlighted.
3. Right click to select the **Freeze Column** command.

Appendix A: Summary of Data Types and Field Properties

Table A.1: Field Property Descriptions

Property	**Description**
Field Size	Sets size of data the field may accept. (See below)
New Values	Tells Access how to increment AutoNumber fields when new records are added to the table. A drop-down menu gives a choice of "Increment" or "Random".
Format	Controls the way that fields are displayed in a form or report.
Decimal Place	Number of digits to the right of the decimal point. Click the arrow to reveal the drop-down menu of decimal place choices.
Input Mask	Allows you to specify a data entry pattern. An input mask informs the user about the pattern of data expected and ensures that data conform to the pattern. Uses the Input Mask Wizard tool.
Caption	Provides information to a user. Carries over to the other database objects such as labels in forms, buttons, and report titles. If no caption is entered, then the field name will be the caption.
Default Value	Allows you to specify a value for a column when the user does not provide one.
Validation Rule	Allows you to specify simple rules to restrict what a user enters into a field.
Validation Text	Gives a message to the user when a validation rule is encountered.
Required	Choose "Yes" if data are required in the field. Choose "No" if null values are permitted.
Allow Zero Length	This setting allows or disallows the field to accept zero-length strings.
Indexed	If set to "Yes", the field supports searching and sorting. A primary key field defaults to "Yes (No Duplicates)".
Unicode Compression	Specifies compression for the field. Microsoft Access 2000 or later uses the Unicode character-encoding scheme to represent the data in a Text, Memo, or Hyperlink field.

Table A.2: Field Properties for Selected Data Types

PROPERTIES	DATA TYPES						
	Short Text	**Long Text**	**Number**	**Date/Time**	**Currency**	**AutoNumber**	**Y/N**
Field Size	D		D			D	
New Values						D	
Format	X	X	X	X	D	X	D
Decimal Place			D		D		
Input Mask	X		X	X	X		
Caption	X	X	X	X	X	X	X
Default Value	X	X	D	X	D		X
Validation Rule	X	X	X	X	X		X
Validation Text	X	X	X	X	X		X
Required	D	D	D	D	D		D
Allow Zero Length	D	D					
Indexed	D		D	D	D	D	

Note: "X"—Property is applicable to data type; "D"—Property has a default value.

Chapter 2: Database Creation Lab

Learning Objectives

This chapter is the first in a series of laboratory exercises enabling you to build a database and associated applications using Microsoft Access 2013. After this chapter, you should have acquired the knowledge and skills to

- Use Design view to create a table.
- Define values for field properties including data types, input masks, validation rules, and formats.
- Create single field and combined primary keys.
- Use wizards and the expression builder.
- Enter data into a table using a datasheet.
- Import data into a table.
- Link tables using the Relationship window.

Overview

In Chapter 1, you gained background about Access databases. You learned about objects in an Access database application, tools to create the objects, and properties of the objects. This background knowledge complemented the concepts in textbook Chapter 3 about relational databases.

This chapter enables you to put the concepts and background knowledge into practice by guiding you to create the auto repair shop database. To gain maximum benefit from this chapter, you should follow along using your copy of Access and the files provided in the website. You will practice creating tables using Design view and the Import Text Wizard. To define certain properties, you will practice using wizards (Input Mask Wizard and Lookup Wizard) and the expression builder. To initially populate your tables, you will use the datasheet and the Import Text Wizard. To finish the creation process, you will use the Relationships window to define referential integrity constraints and rules about referenced rows. After finishing this chapter, you should be ready to create your own databases in Access.

2.1 Creating the Auto Repair Database

This section guides you to creating an empty database. It is assumed that you have already installed Microsoft Access 2013 and you have your computer turned on.

Begin by selecting Microsoft Access 2013 from the **Program** menu. When you launch Access 2013, the opening window differs from its appearance in the previous 2010 version (Figure 1). The opening window displays different options for creating a database, e.g. blank, blank web, and various template options. It also displays the recently opened databases. If the **Blank Database** button is not selected, select it. In the File Name area in the new window, enter "Auto Repair" as the name for the database as shown in Figure 2. The default location to save the file is your personal My Documents folder. To choose a different location, click the folder icon to show the File New Database dialog box, browse to the folder you want (Figure 3), and then click **OK**. Click the **Create** button at the bottom-right of the Access window to save the database.

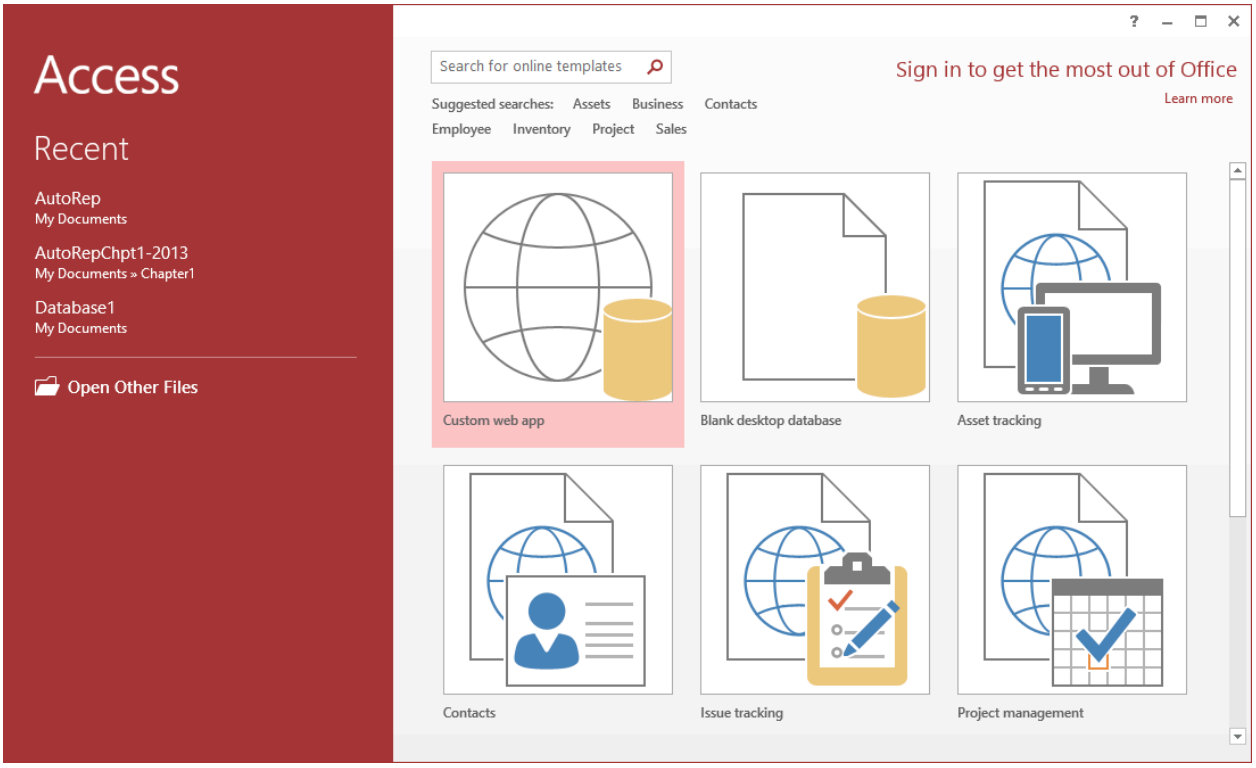


Figure 1: Access 2013 Getting Started Page

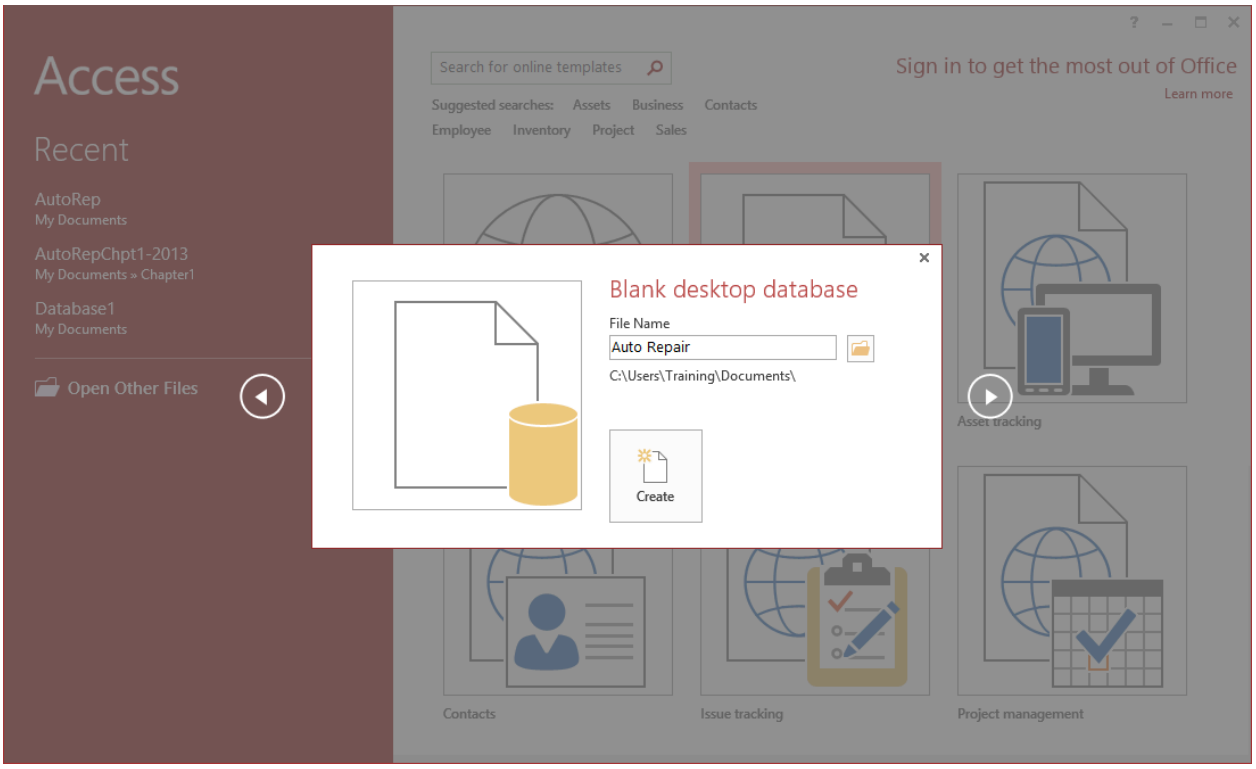


Figure 2: Access 2013 Getting Started Page with File Name Entry

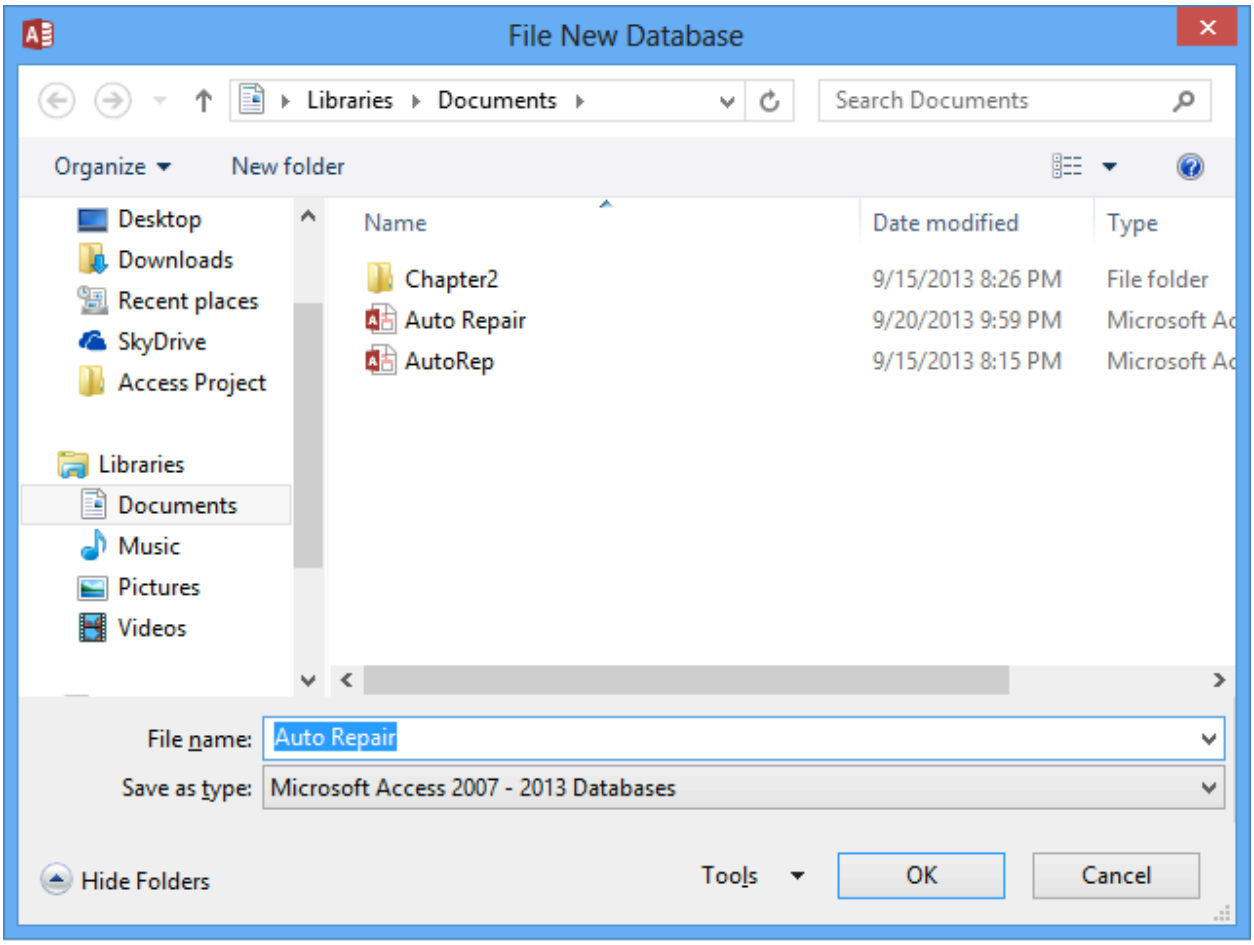


Figure 3: File New Database Window

After the database has been saved, the Database window appears, as depicted in Figure 4. The Ribbon appears at the top of your screen, and a navigation pane appears on the left. The Database window defaults to the **Tables** section since this is where the database creation process begins. The **Tables** section shows the default *Table1* because you have not created any tables.

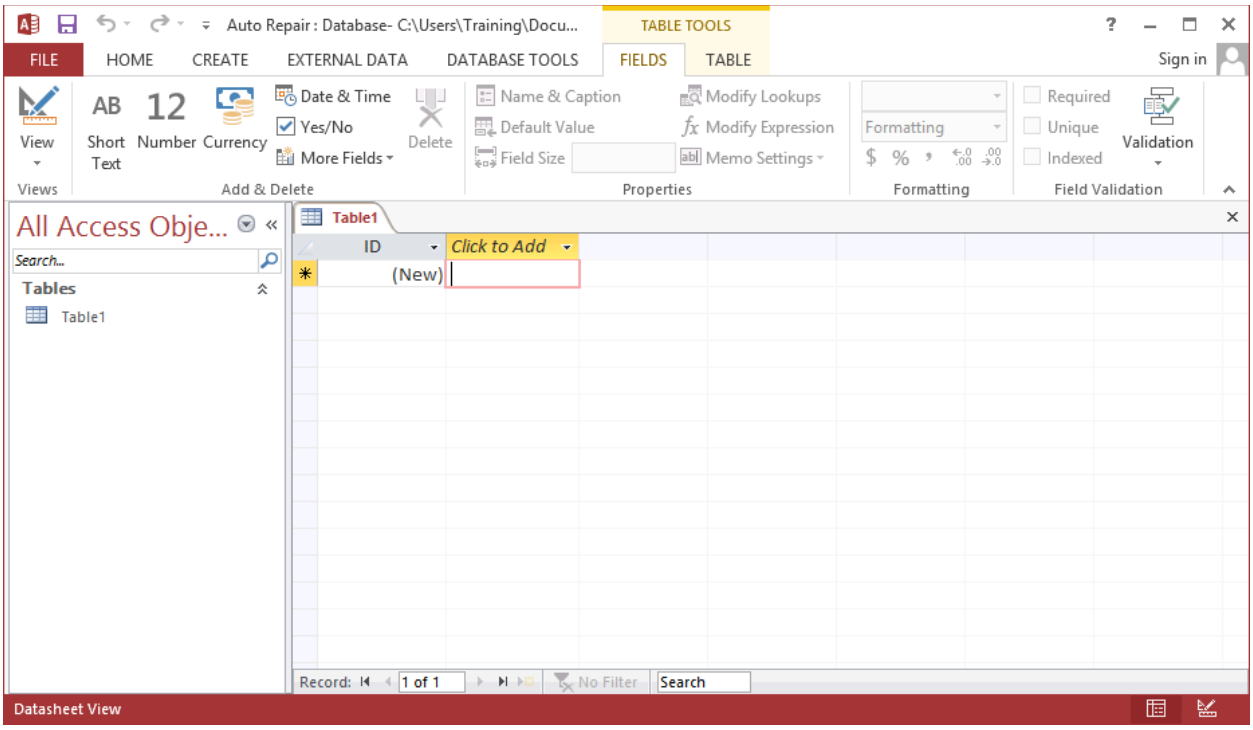


Figure 4: Empty Database Window

2.2 Creating Tables in Access

Access supports a number of ways to create a new table or to add an existing table into a database. This section explains how to use Design view to create a table. Other sections depict the Get External Data Wizard and Table Templates. You also can use the SQL CREATE TABLE statements as presented in textbook Chapter 3. Because you can use the CREATE TABLE statement only inside a module or data definition query, we will not discuss it in this chapter.

2.2.1 Creating the First Table

You will follow the instructions below to create the *Customer* table from the Database window. If you have exited Access, start Access again and select the *AutoRepair* database. Otherwise, the *AutoRepair* Database window should remain on the screen with the **Tables** section open (Figure 4). To proceed, follow these instructions:

- Click the **View** button on the **Home** ribbon. Select "Design View".
- You will need to specify a table name. Type "Customer" as the table name to replace the default table name (*Table1*).
- The Design View window will appear in view pane (Figure 5).

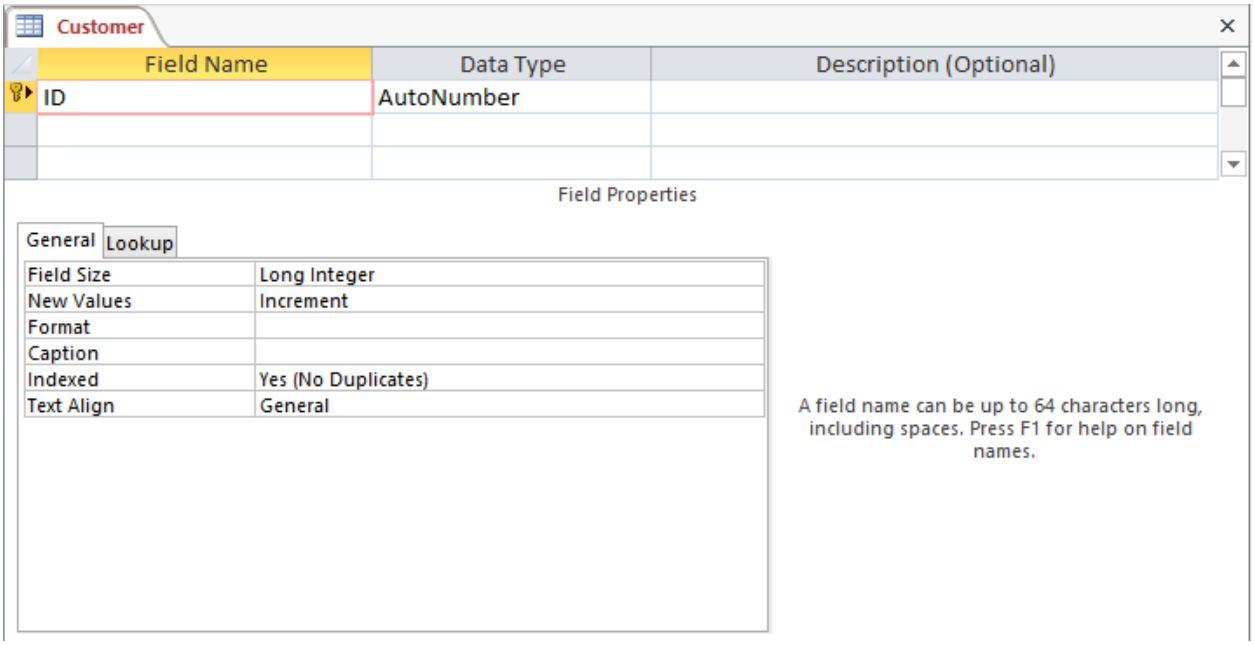


Figure 5: Design View for the Empty Customer Table

2.2.2 Defining the First Field and Its Properties

You should still have the empty design view (Figure 5) open. The following instructions direct you in defining properties for the seven columns of the *Customer* table.

1. Rename the First Field: You should change the default name "ID" of the first field. Position the cursor in the first row of the *Field Name* column and type "CustNo" (abbreviation for Customer Number) as shown in Figure 6. It is a good idea to keep field names short or abbreviate them so they will fit easier in a datasheet.

2. Set the Data Type: Use the **Tab** key to move the cursor over to the *Data Type* column. Choose "AutoNumber" as the data type instead of the default data type of "Text". Recall from Chapter 1 that the "AutoNumber" data type allows Access to generate unique customer numbers.

3. Type the Optional Description: Use the **Tab** key again to move the cursor to the *Description* column. If you desire, type a description for the *CustNo* field name. Typing a description is optional if the field name meaning is obvious. Because the meaning may not be obvious here, type "Unique Customer Number" in the text area.

4. Set Other Field Properties: The bottom of the window lists the field property options and the default values for the corresponding data type (AutoNumber). To change default values in the field property frame, you either type changes or click an arrow to select from a list of values. Follow these steps:

   - For the *Field Size* property, do not change the default value of "Long Integer" because it allows a large number of unique values. You may want to browse the selections in the list to see other choices.
   - For the *New Values* property, do not change the default value of "Increment" because you want customer number values generated sequentially. You may want to browse the selections in the list to see other choices.
   - Move the cursor on the *Indexed* property and click on it. An explanation appears on the right. The option "Yes (No Duplicates)" is already selected from the list. An index allows Access to rapidly perform search and sort operations. Textbook Chapter 8 describes the purposes of indexes in detail.

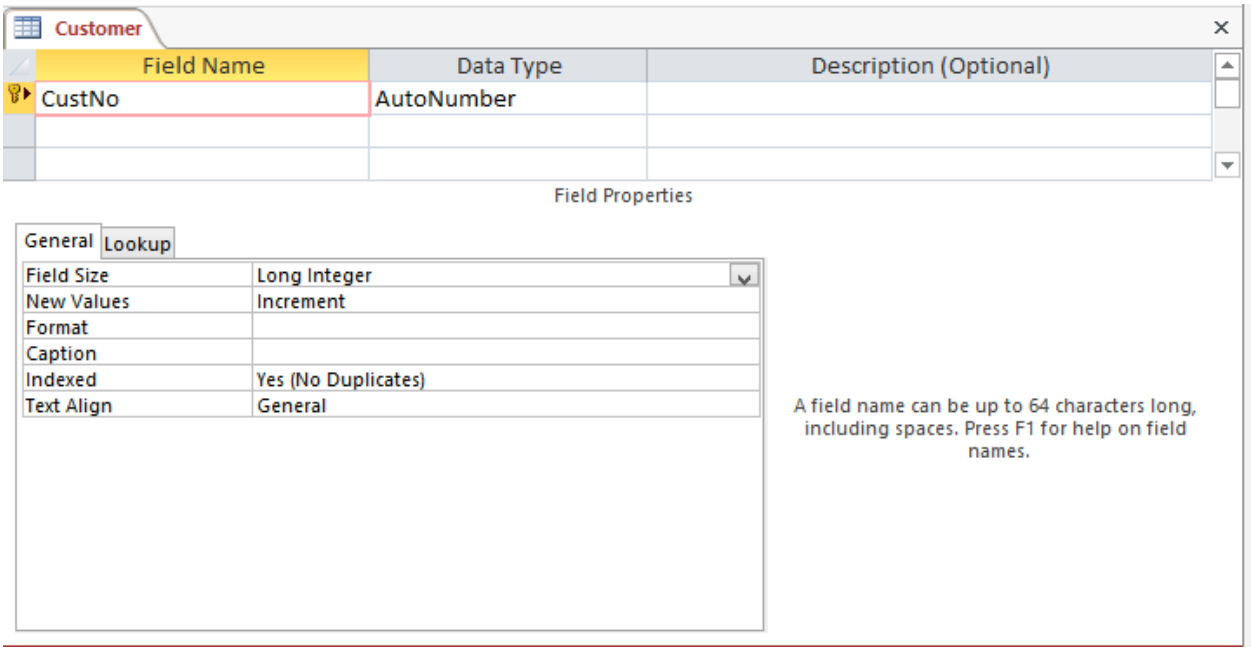


Figure 6: Design View Showing Properties of the *CustNo* Field

5. Set the Primary Key: By default, Access sets the first column as the primary key so you do not need to set *CustNo* as the primary key. In other tables, you will need to set the primary key. You may designate a field as the primary key by either (a) clicking the small **Key** icon on the Access toolbar above or (b) clicking the right mouse button and selecting the **Primary Key** item from the drop-down menu. After designating the primary key, a small key icon appears in the left margin a shown in Figure 6.

6. Save the Table: Select **File → Save** or type **Ctrl S** to save the table. Since you already renamed the table as "Customer", you do not need to change the table name.

2.2.3 Defining the Remaining Fields and Properties

1. Name the Second Field: Position the cursor in the second row of the *Field Name* column and type in the field name *FirstName*, as shown in Figure 7.

2. Set the Data Type: Use the **Tab** key to move the cursor over to the *Data Type* column. Do not change the value from the default "Short Text" value.

3. Type Optional Description: Use the **Tab** key again to move the cursor to the *Description* column. If you desire, type a description for the *FirstName* field.

4. Set Other Field Properties: Use Figure 7 as a guide to change some of the default field properties as suggested in the following list:

   - Change the *Field Size* property to 20 since 255 characters is excessive.
   - Set the cursor on the *Required* property. Change the default selection to "Yes" to force the user to enter a value. Null values are not valid for this field.
   - Set the crusor on the *Allow Zero Length* property. Change the default selection to "No". The ("No") value ensures that the user types a string of one or more characters.
   - Retain the default value ("No") for the *Indexed* property. This field is not a good index choice because it is rather long (20 characters). See textbook Chapter 8 for more details about index selection.
   - When you are finished, the field property option list should appear as the bottom of Figure 7.

- Repeat the above steps for the *LastName* field since the data types and the field properties are the same.

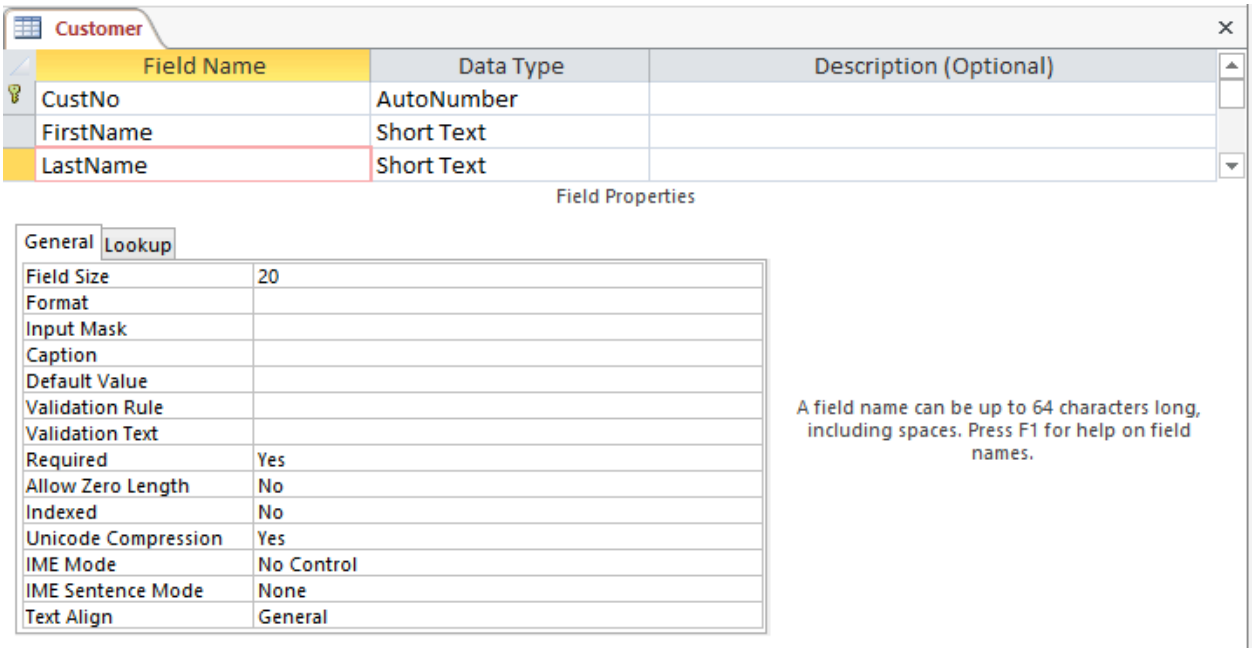


Figure 7: Design View Window for the *FirstName* and the *LastName* Fields

5. Complete the Remaining Fields: Name the remaining fields and set the data types and field properties according to Table 1. To set properties for the last two fields, you will invoke the Input Mask Wizard that is explained after Table 1.

Table 1: Property Values for the Remaining Fields of the *Customer* Table

Field Name	**Data Type**	**Field Property Values**
Address	Text	*Field Size*: "40" *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"
City	Text	*Field Size*: "20" *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"
State	Text	*Field Size*: "2" *Default Value*: "WA" *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"
PostalCode	Text	*Field Size*: "10" *Input Mask*: "Zip Code" (Save without symbols) *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"
PhoneNumber	Text	*Field Size*: "15" *Input Mask*: "Phone Number" (Save without symbols) *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"

6. Using the Input Mask Wizard: When you click in the *Input Mask* property area, three dots (**…**) appear indicating that a wizard is available. Click on these dots to invoke the Input Mask Wizard. If the three dots do not appear, the Input Mask Wizard has not been installed on your copy of Access. You will need to add the Input Mask Wizard in the installation process.

   - If you have not saved the table after making the changes in Table 1, the Input Mask Wizard initially asks you to save the table first (click **Yes).**
   - The first wizard window (Figure 8) appears containing the input mask choices available. You select the input mask according to the type of data contained in the field. You can begin by selecting "Zip Code". Type an example in the Try It area below to experiment with the input mask and click the **Next** button.

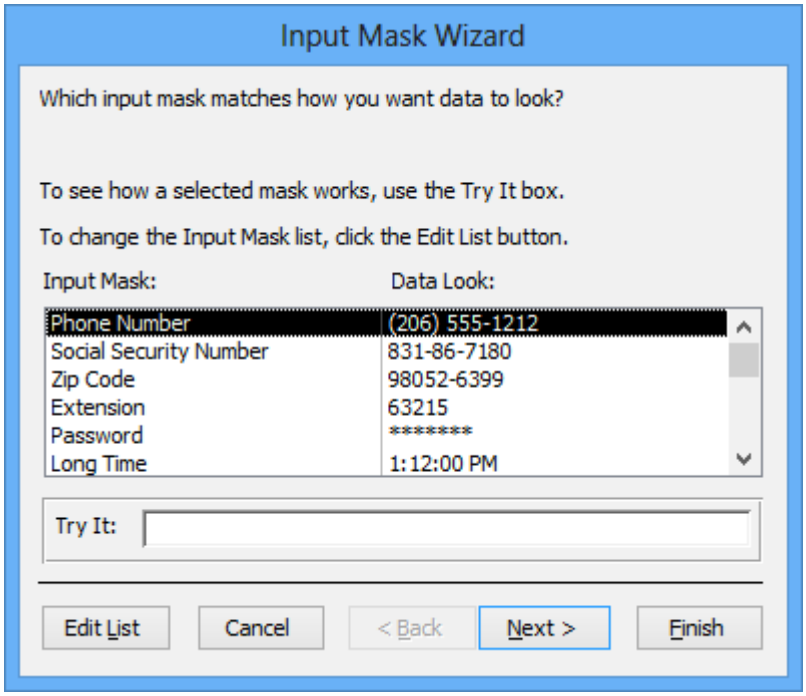


Figure 8: First Window of the Input Mask Wizard

- The next wizard window (Figure 9) allows you to make changes in the selected input mask. An example would be to put in a local default zip code. For this example, leave it alone and click the **Next** button.

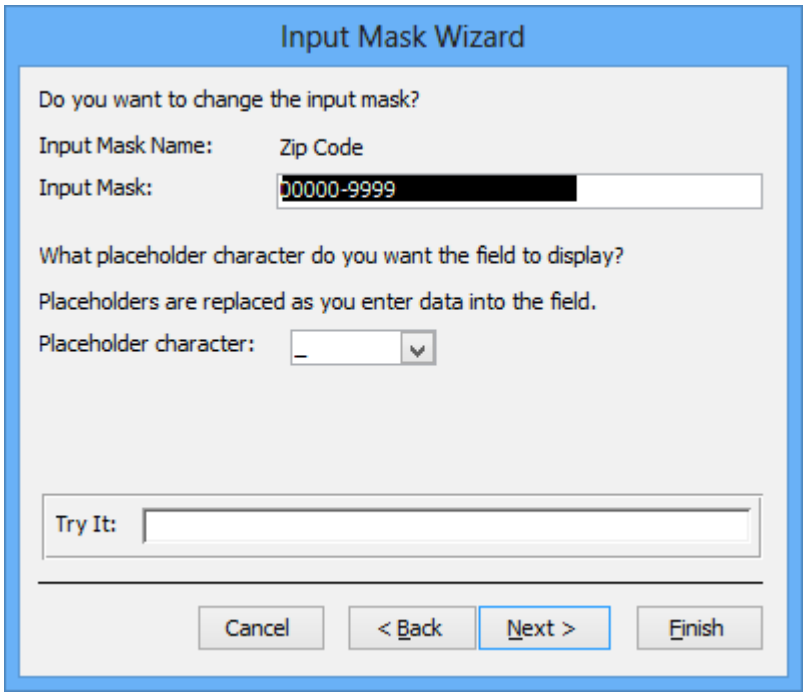


Figure 9: Change Mask Window of the Input Mask Wizard

- The final step asks how to save the data, with or without the input mask symbols (Figure 10). It is recommended to store the data without the symbols to save memory and obtain faster processing.

The input symbols will still appear in your tables and forms. Click **Next** and then the **Finish** button in the last dialog window.

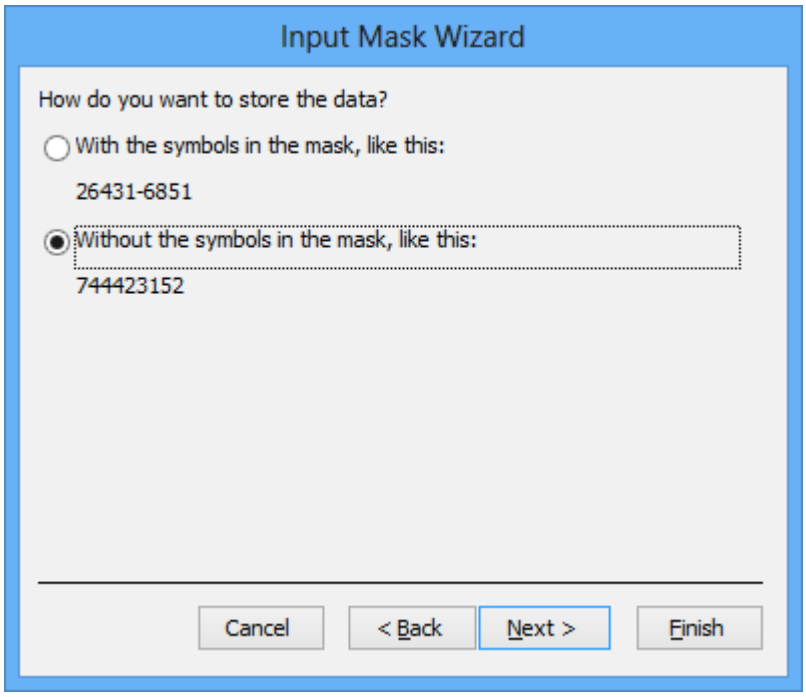


Figure 10: Storing Data Window of the Input Wizard

7. After finishing the Input Mask Wizard, you can view the *Input Mask* property as shown in Figure 11. You can customize the value of the *Input Mask* property by typing in the text area. You will want to read the help documentation (type the **F1** function key while the cursor is inside the text area of the *Input Mask* property and enter "input mask" as the search string) to learn about the details.
8. Repeat the previous steps for setting an *Input Mask* for the *PhoneNumber* field except that you want to use the "Phone Number" input mask.
9. Your completed table should appear as in Figure 11. Since you have already named and saved the table, you can close the table by clicking on the **Close** (**X**) button of the Design View window. Click **Yes** when Access asks you if you want to save your changes.

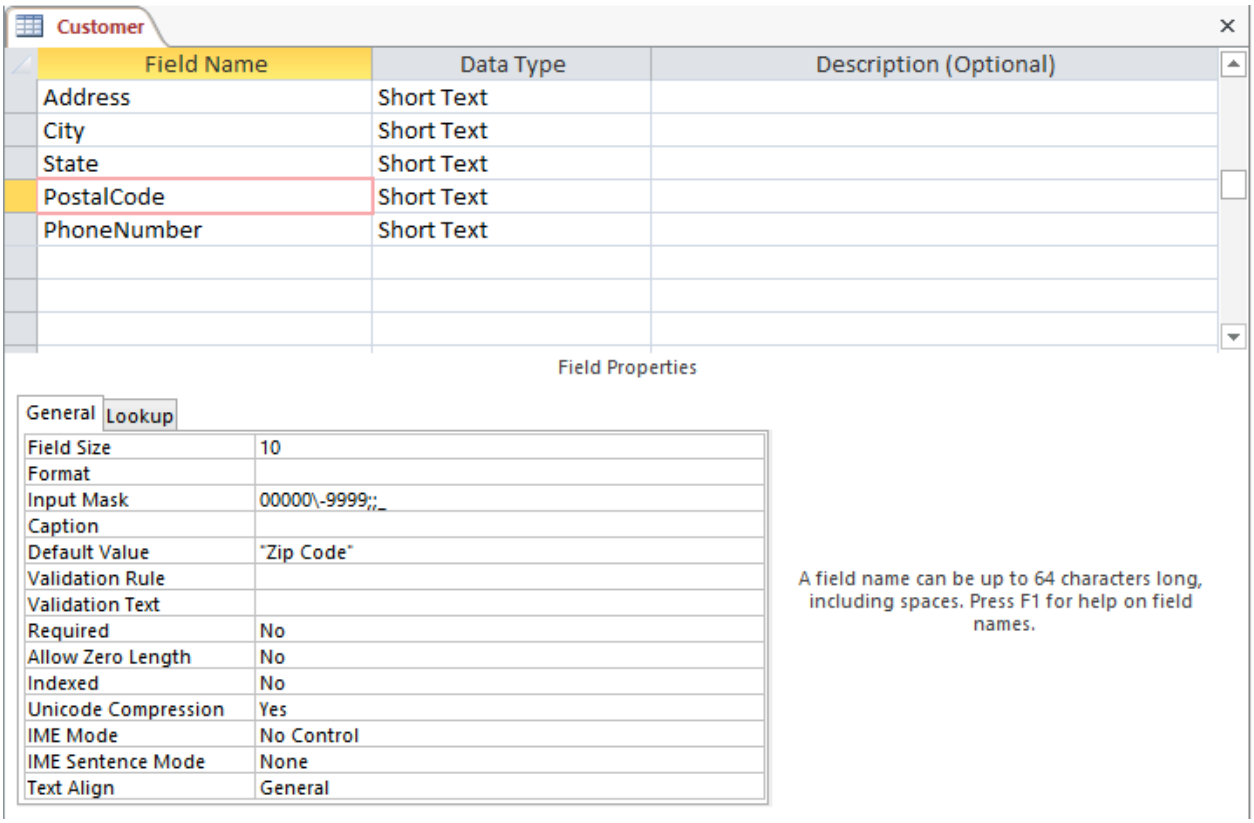


Figure 11: Design View Window for the *Customer* Table

Before we move on to another table, let's take a quick look at the *Customer* table in Datasheet view:

- While the *Customer* table name is selected in the **Tables** section of the Navigation pane, double-click on the object.
- The Datasheet view appears with the fields named (see Figure 12).
- Toggle between views: Tables have two views: Design view and Datasheet view. Design view allows you to add controls and modify existing controls and datasheet view lets you view the data in an organized columnar manner. To change views, right-click the tab name (like "Customer: Table"), and then choose the view you want.
- In the **Home** tab of the Ribbon, the **View** list allows you to switch between views. In this chapter, you can use the **View** list to switch between the **Datasheet** icon  and the **Design View** icon  .
- Close the Datasheet window by clicking on the **Close** button at the right top corner of the window.

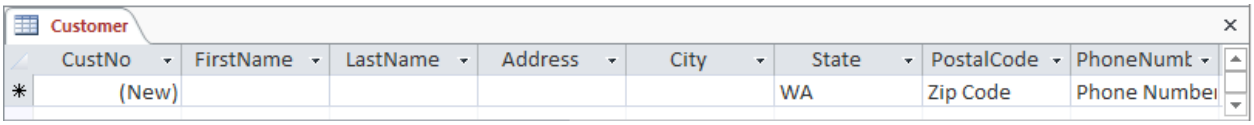


Figure 12: Datasheet View for the Empty *Customer* Table

2.3 Completing the Next Three Tables

The next three tables will follow the steps performed above except for a few different field properties. The table list should remain in the navigation pane.

2.3.1 The RepairOrder *Table*

To gain additional practice with Design View, you will use it to define the *RepairOrder* table as described in the following steps:

1. Add New Table: Click **Create → Table** (Figure 13).
2. Select Design View: Select "Design View" from the New Table window.
3. Add the Fields: Fill in the field names and the properties according to Table 2. The last field is a foreign key.
4. Set the Primary Key: When you are finished, set the *OrdNo* field as the primary key.
5. Name and Save the Table: Your table should appear as in Figure 14. You can close the table by clicking on the **Close** button. Click the **Yes** button to save your changes and type "RepairOrder" as the table name. After you click **OK**, the Database window appears with the *RepairOrder* table.

Table 2: Fields for the *RepairOrder* Table

Field Name	**Data Type**	**Field Properties**
OrdNo (Set as the primary key; Abbreviation for Order Number)	AutoNumber	*Field Size*: "Long Integer" *New Values*: "Increment" *Indexed*: "Yes (No Duplicates)"
Odometer	Number	*Field Size*: "Long Integer" *Decimal Place*: "Auto" *Validation Rule*: "> 0" *Validation Text*: "Odometer value must be greater than 0." *Required*: "Yes" *Indexed*: "No"
TimeRecvd (Denotes when the auto was received into the shop for service. *Default Value* property uses an expression with the Now() function.)	Date/Time	*Format*: "General Date" *Default Value*: "Now()" *Required*: "Yes" *Indexed*: "No"
TimeFinish (Denotes when the auto was finished in the shop for service)	Date/Time	*Format*: "General Date" *Required*: "No" *Indexed*: "No"
PhoneWnRdy (Abbreviation for phone customer when car is ready to be picked up).	Yes/No	*Format*: "Yes/No" *Indexed*: "No"
SerialNo (Serial number of auto being repaired; Foreign key from the *Vehicle* table)	Short Text	*Field Size*: "25" *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"

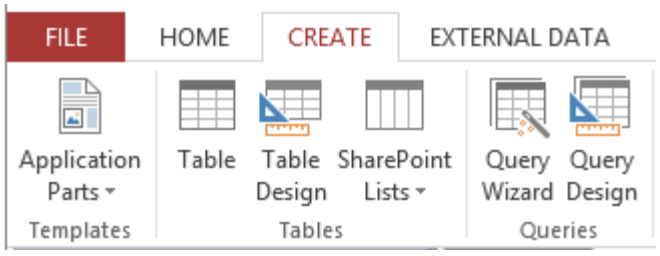


Figure 13: **Create** Tab Showing the **Table** Button

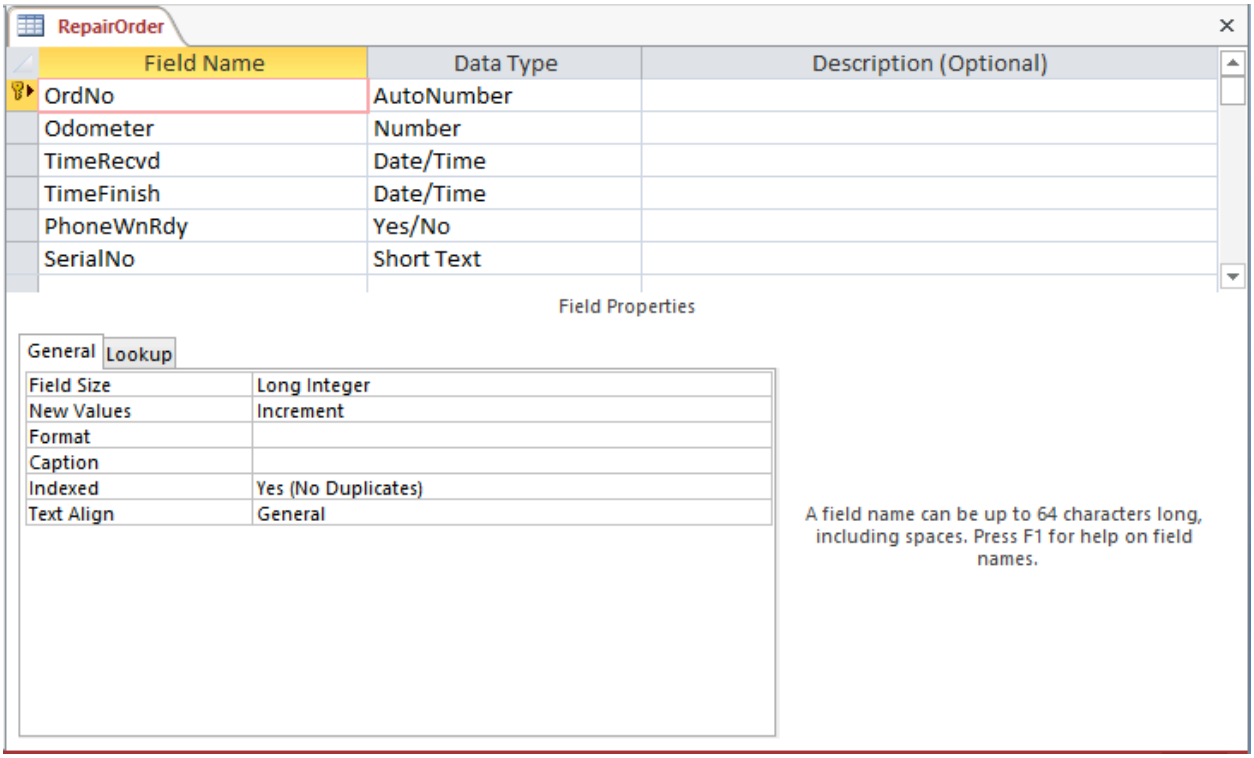


Figure 14: Design View Window for the Completed *RepairOrder* Table

Using Table 2, you defined the *Validation Rule* and the *Validation Text* properties for the *Odometer* field. A validation rule for a field does not use the field name because it is implied. For example, the *Validation Rule* property for Odometer is "> 0". Validation rules for fields cannot reference other fields. To define a validation rule that references other fields, you need a table validation rule, as explained next.

2.3.2 Assigning a Table Property for the RepairOrder *Table*

The *RepairOrder* table needs a validation rule to ensure that the time a repair order was completed (*TimeFinish*) is greater than or equal to the time the repair order was received (*TimeRecvd*). If a repair order has not been completed, the rule should allow a null value for the *TimeFinish* field. This rule must be defined as a table property because it involves two fields. Follow the steps below to define a table property for the *RepairOrder* table:

1. Open the Table Properties Window: With the Design view open (Figure 15), select **Design → Property Sheet** (Figure 16) and a blank Table Properties window appears (Figure 17).

> Technical Note: Alternatively, when the mouse is inside the Design View window, you can right-mouse-click and choose **Properties** from the shortcut menu.

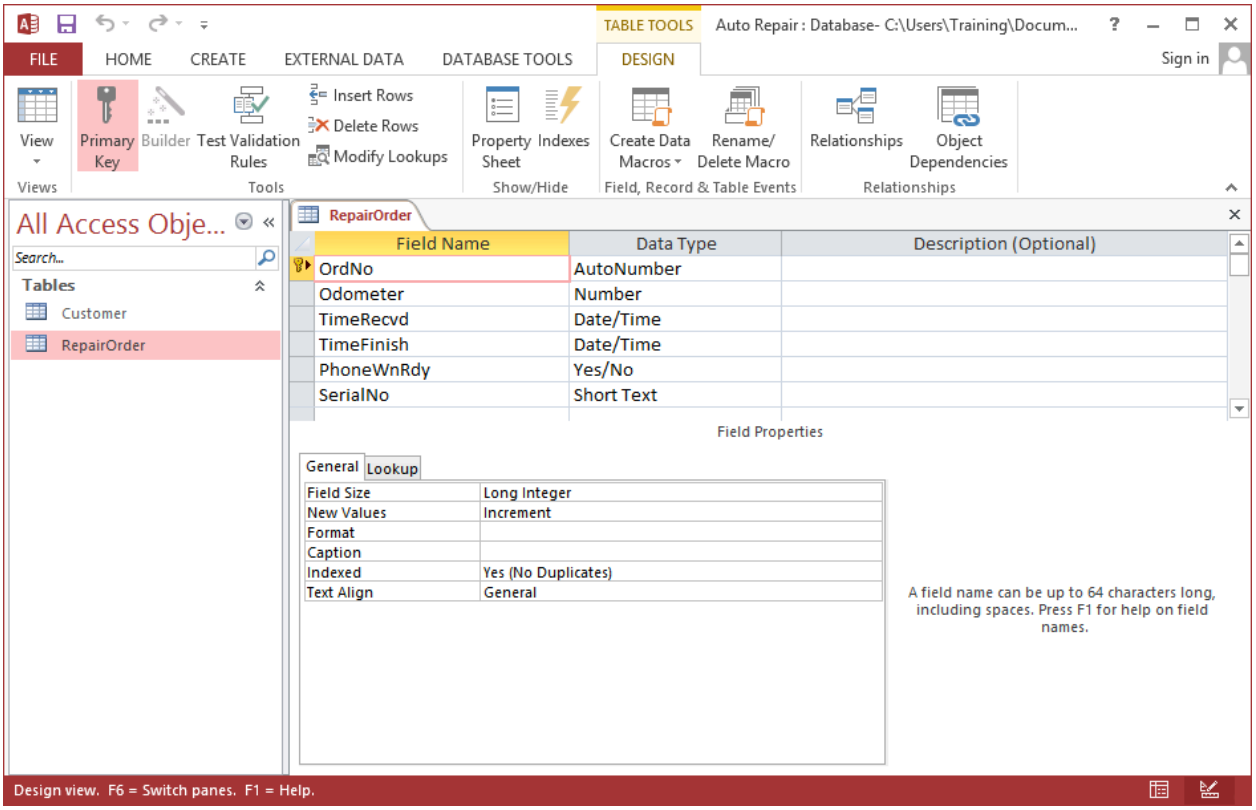


Figure 15: Design View with the design tap in ribbon

2. Open the Expression Builder: Click in the *Validation Rule* area and then click the ellipsis (**…**) button to invoke the Expression Builder window (Figure 18). If necessary, refer back to Chapter 1 to review instructions about using the expression builder.

3. Complete the Expression: Fill in the Expression Builder window to match Figure 18. You may either type the expression or use the expression builder to select certain parts. For example, you can select field names from the middle, bottom pane. Type the comparison operator (>=) because it cannot be selected from the operator buttons below.[1] When you are finished, click **OK** to return to the Table Properties window.

4. Define Other Table Properties: In the Table Properties window, define the *Description* and the *Validation Text* properties to match Figure 19.

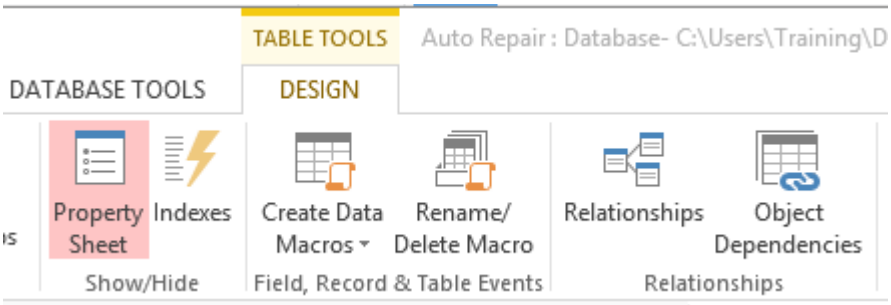


Figure 16: Design Tab in the Ribbon Showing the Properties Button

[1] You can select the > operator and the = operator but not the >= operator.

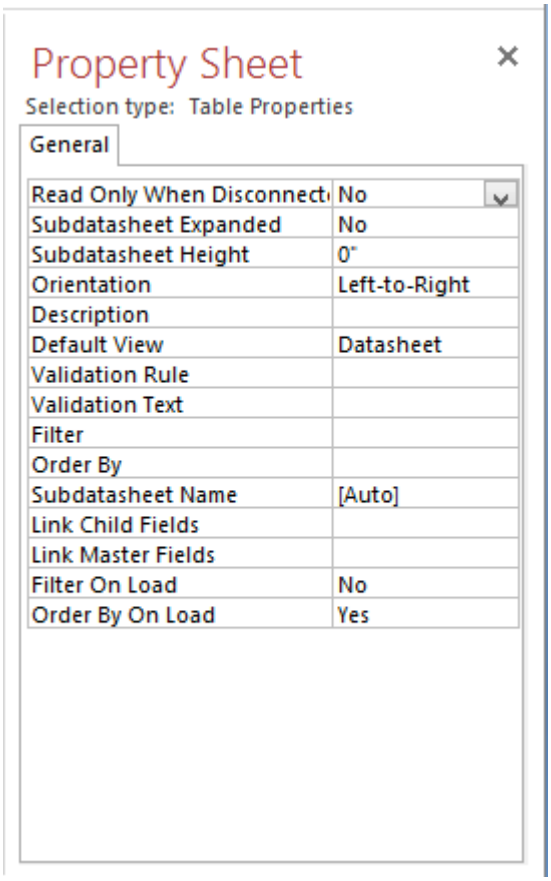


Figure 17: Empty Table Properties Window

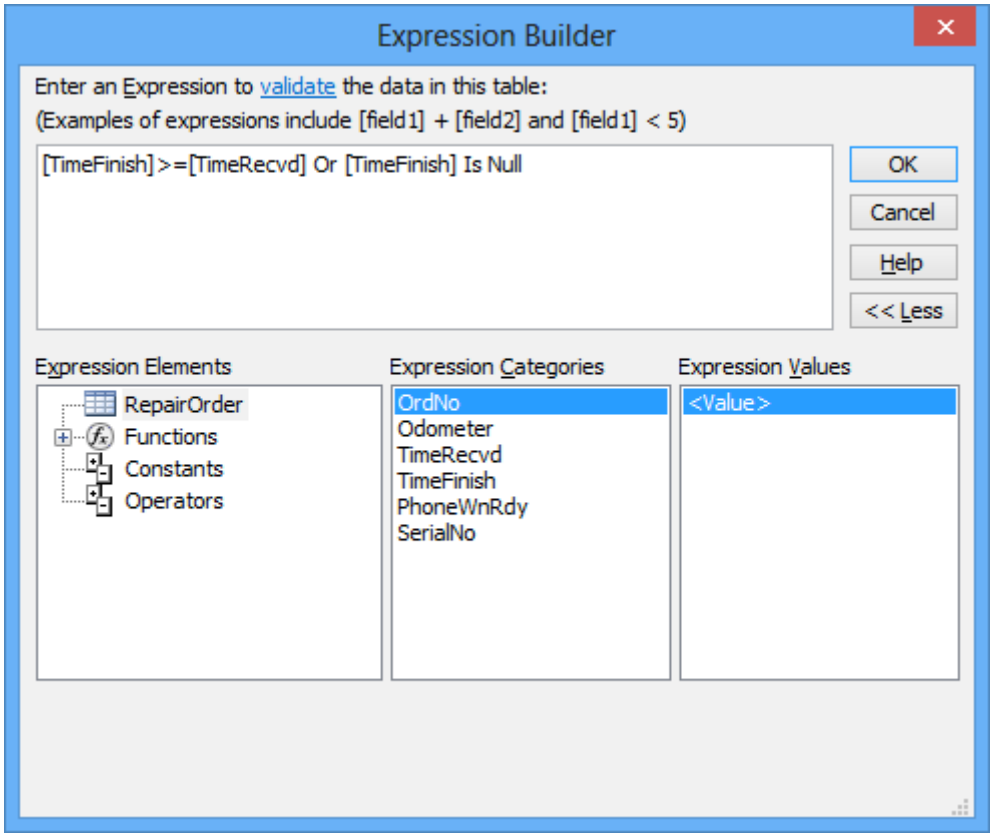


Figure 18: Expression Builder Window Showing the Table Validation Rule

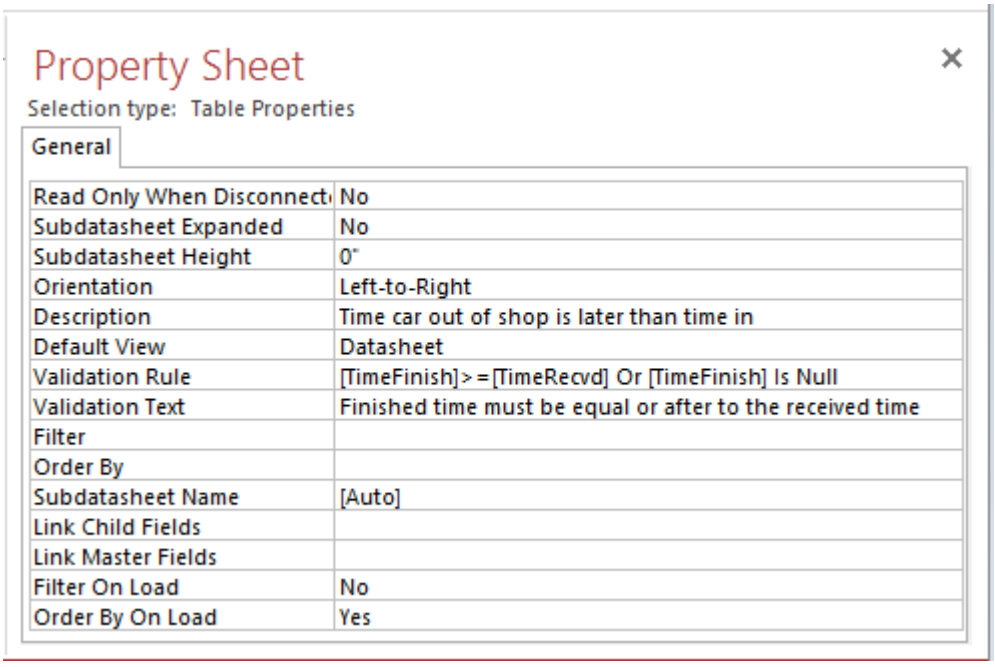


Figure 19: Table Properties Window with a Validation Rule

2.3.3 The Part *Table*

You will follow a similar procedure to create the *Part* table. You should use Table 3 as a guide to define the fields in the *Part* table. When you are complete, the *Part* table should appear as shown in Design View in Figure 20.

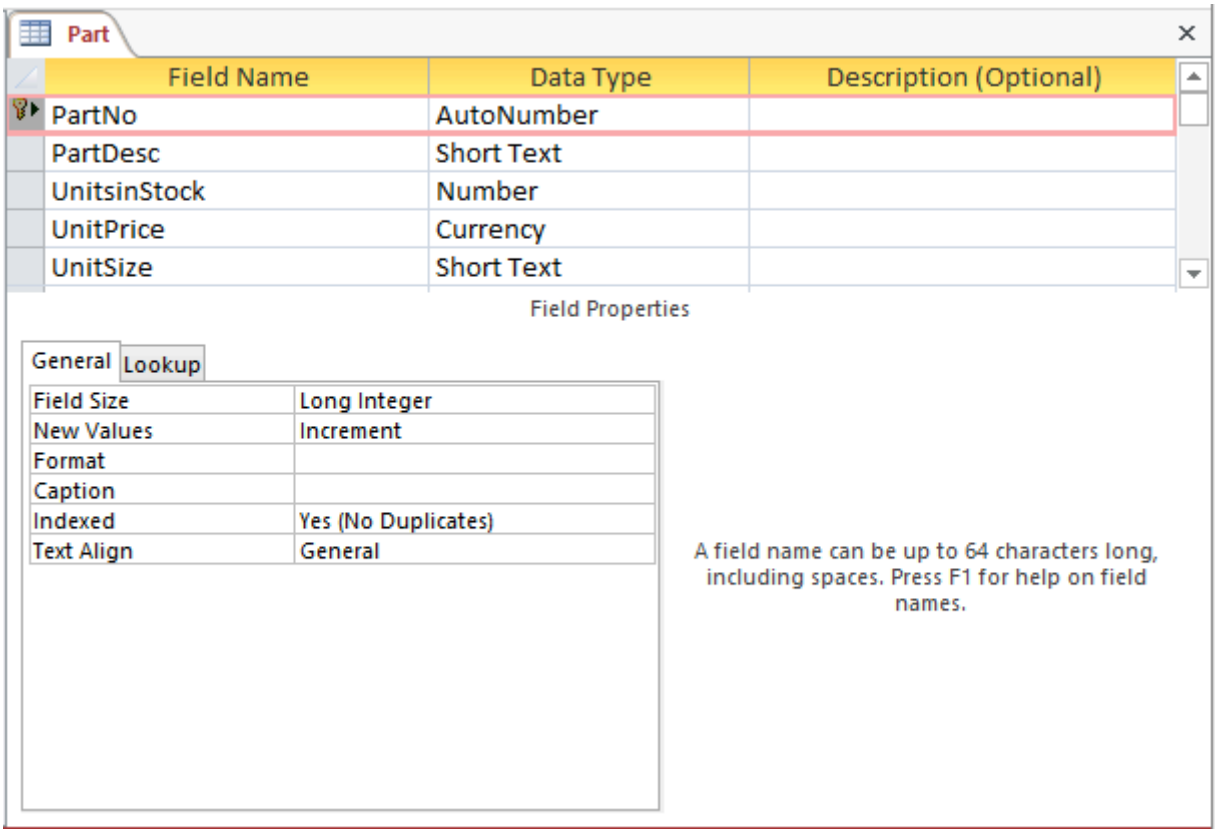


Figure 20: Design View for the Completed *Part* Table.

Table 3: Fields for the *Part* Table

Field Name	Data Type	Field Properties
PartNo	AutoNumber	*Field Size*: "Long Integer" *New Values*: "Increment" *Indexed*: "Yes (No Duplicates)"
PartDesc (Abbreviation for "Part Description")	Short Text	*Field Size*: "40" *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"
UnitsInStock	Number	*Field Size*: "Long Integer" *Decimal Places*: "Auto" *Required*: "No" *Indexed*: "No"
UnitPrice	Currency	*Format*: "Currency" *Decimal Places*: "Auto" *Required*: "No" *Indexed*: "No"
UnitSize (Add this field)	Short Text	*Field Size*: "10" *Required*: "Yes" *Allow Zero Length*: "No" *Indexed*: "No"

2.3.4 The PartsUsed *Table*

The *PartsUsed* table differs from the other tables in several ways. First, it contains a combined primary key consisting of the *PartNo* and the *OrdNo* fields. Second, these two fields are also foreign keys linking to the *Part* and the *RepairOrder* tables, respectively. The following instructions describe creation of a combined primary key and usage of the Lookup Wizard for foreign keys. In Access 2013, the Lookup Wizard allows linking the tables as well as supporting selection of values of a foreign key field from another table. However, we will not use the Lookup Wizard to link the tables. The result is a combo box appearing when you open a datasheet or form (Figure 21). Later, you will use the Relationships window to actually link the tables with referential integrity constraints.

Figure 21: Combo Boxes in a Datasheet and Form

Initial Steps

1. Add New Table: Click **Create** → **Table**.
2. Select Design View: Select "Design View" from the **View** button on the **Home** Ribbon. Save the table as "PartsUsed" when prompted by the *Save as* window.

3. Name the Fields: Name the first two fields according to Table 4.

Table 4: Initial Fields of the *PartsUsed* table

Field Name	**Data Type**	**Field Properties**
OrdNo (Abbreviation for Order Number from the *RepairOrder* table)	Number	*Field Size*: "Long Integer" *Decimal Places*: "Auto" *Default Value*: None (delete 0) *Required*: "Yes" *Indexed*: "No"
QtyUsed (Abbreviation for Quantity of Parts Used.)	Number	*Field Size*: "Integer" *Decimal Places*: "Auto" *Default Value*: "1" *Validation Rule:* "> 0" *Validation Text*: "Qty Used must be greater than 0." *Required*: "Yes" *Indexed*: "No"

4. Select the Data Type for a New Field: Click in the *Data Type* column of the next empty row. Select the "Number" data type for the new field as shown in Figure 22. Leave the Field Name area blank for now.

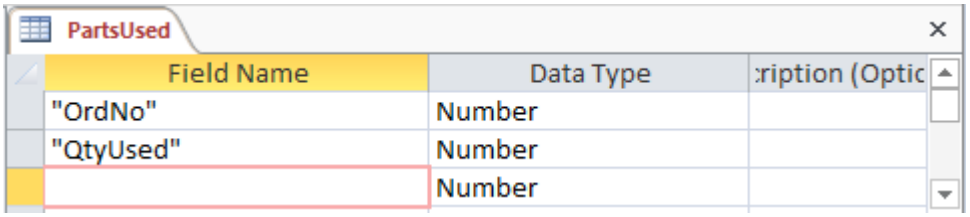


Figure 22: Selecting the Data Type in the Next Empty Row

Using the Lookup Wizard

After you select the data type, go back into the data type list and select **Lookup Wizard …** to invoke the initial dialog box of the Lookup Wizard (Figure 23). Remember that the Lookup Wizard creates foreign key fields so values may be selected from another table. This enables a combo box or list to appear in a form field for a user to select criteria. The following steps guide you in the choices of the Lookup Wizard.

1. Choose the Kind of Value Source: You have two choices; choose the first one, "…look up the values in a table or query.", and click **Next**.

2. Choose the Table: Figure 24 appears on the screen allowing you to select a table from which the lookup entries come. Select *Part* and click **Next**.

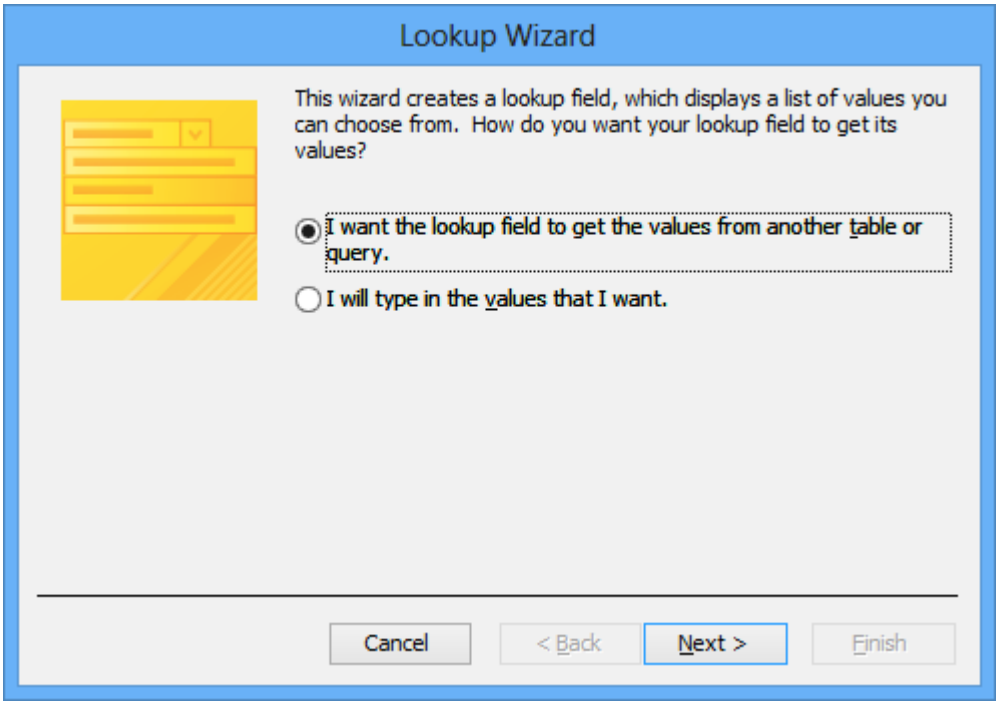


Figure 23: Initial Window of the Lookup Wizard

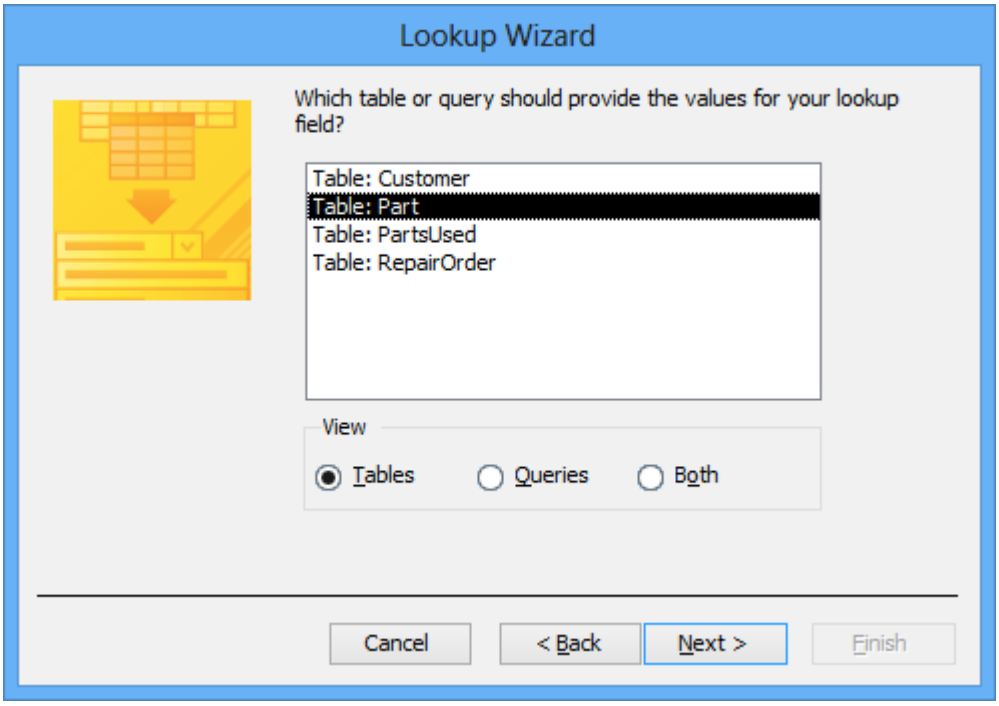


Figure 24: Choose Table Window of the Lookup Wizard

3. Choose the Lookup Field: The wizard window lets you choose the field from the *Part* table to appear as a list (Figure 25). Select the *PartNo* field and click the > button to bring it to the right side. Click **Next** to continue.

4. Choose the Sort Order: The next wizard window allows you to choose the sort order. Select ascending *PartNo* as the sort order as shown in Figure 26.

5. Examine List Appearance: Figure 27 shows how the part numbers in the list appear. Since there are no data in the *Part.Partno* field yet, the list is empty except for the field name. If there were data, you also would be able to adjust the column width as desired. So for now, just click **Next** to continue.

6. Name the Lookup Field: The wizard now prompts you to name the lookup field, as shown in Figure 28. Type "PartNo" and click **Finish**.

7. Save the Table: Click the **Yes** button in response to the dialog box. You are then returned to the Design View window with the lookup properties created for the *PartNo* field.

8. Make *OrdNo* a Lookup Field: Repeat a similar process for the *OrdNo* field. Use the Lookup Wizard to make *RepairOrder.OrdNo* the source of values.

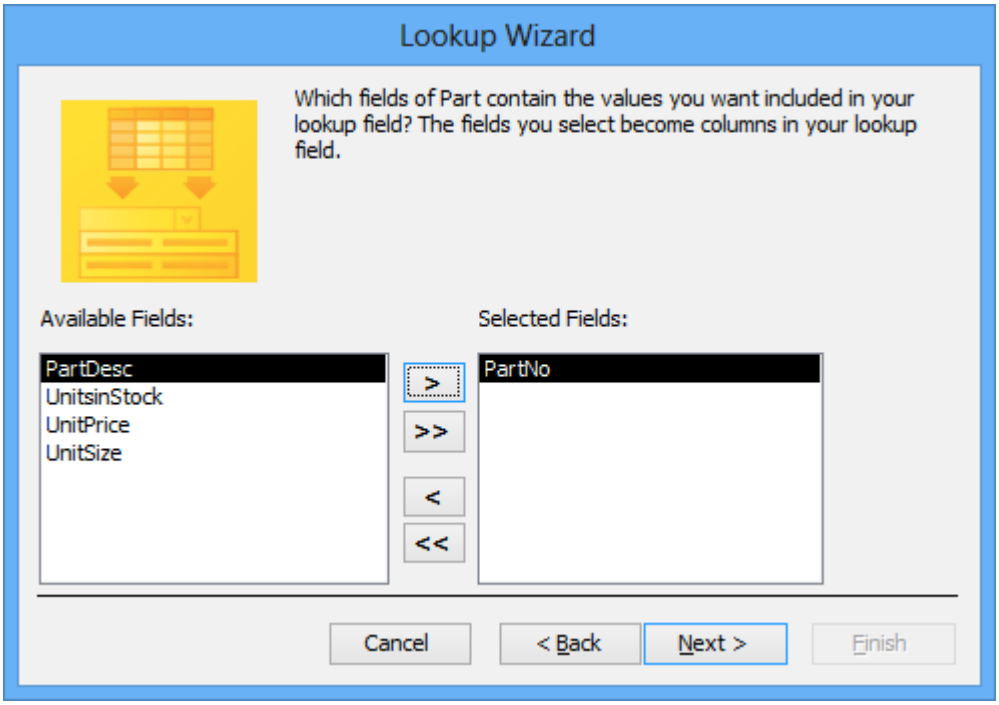


Figure 25: Choosing Fields in the Lookup Wizard

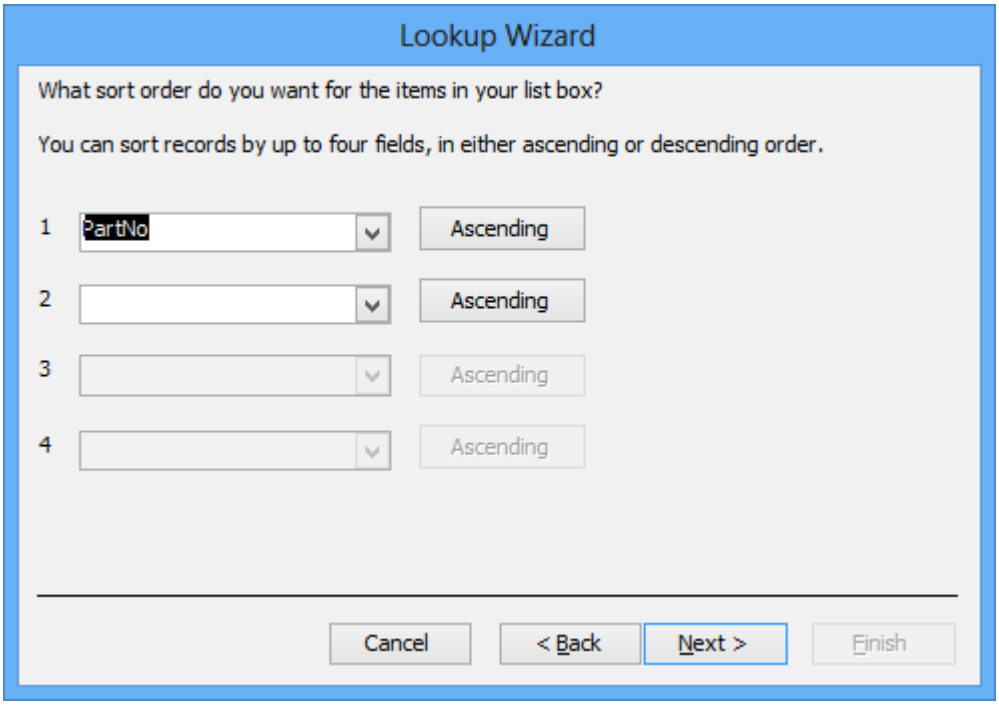


Figure 26: Specify Sort Order

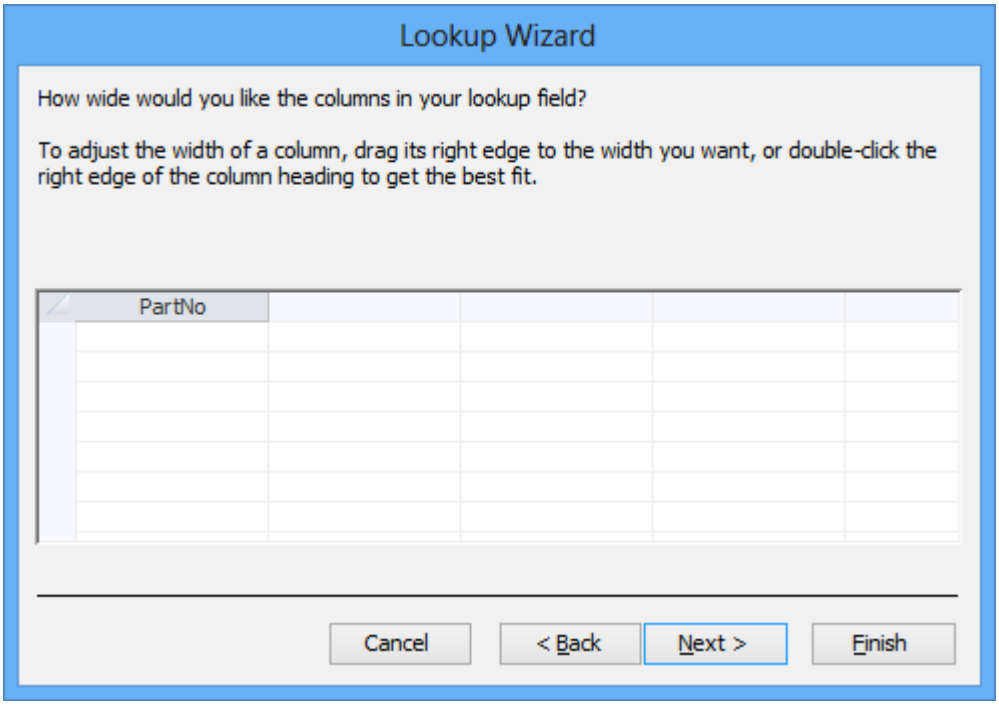


Figure 27: Adjust Column Width

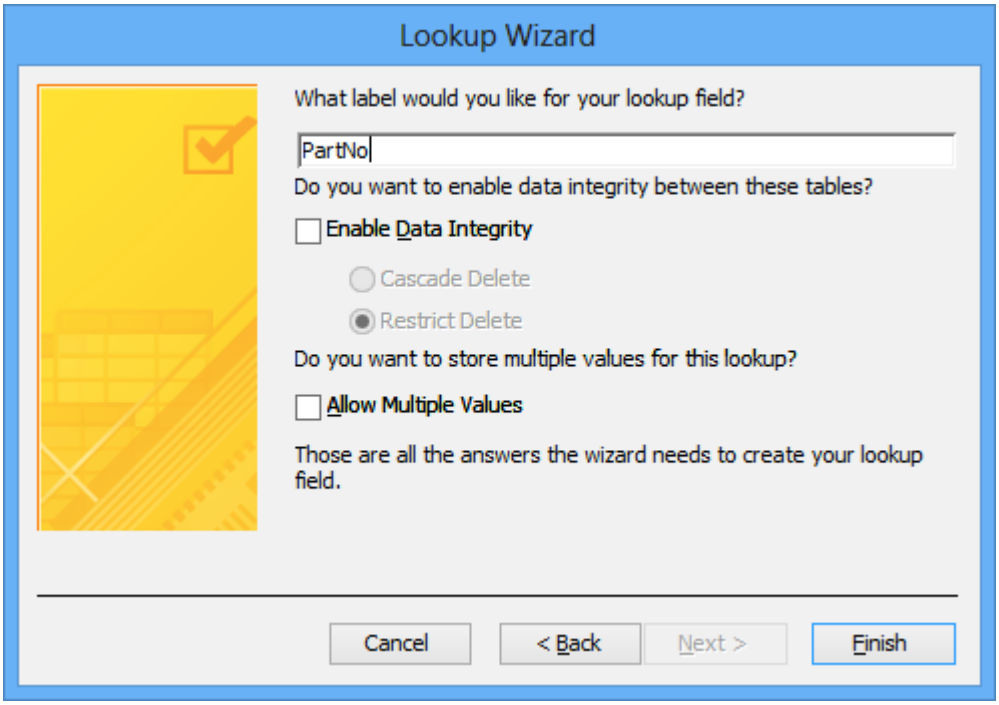


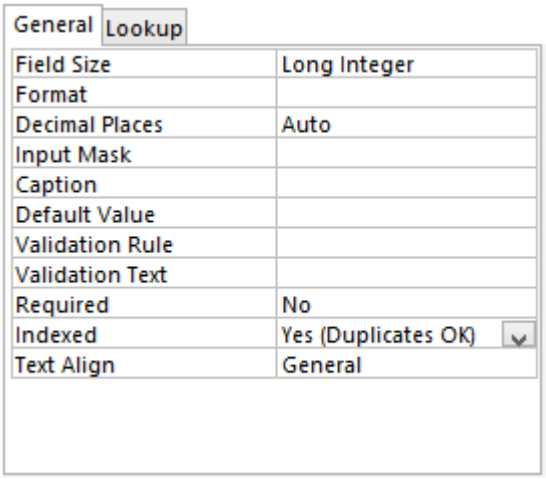
Figure 28: Final Lookup Wizard Window

Final Steps

After finishing the Lookup Wizard for both fields, you can make a few additional changes to complete the *PartsUsed* table. You need to change the *Indexed* property for *PartNo* and make a combined primary key, as described in the following steps.

1. Change the *Indexed* Property: The **General** section or tab (Figure 29) contains the default field property settings. Change the *Indexed* property to "Yes (Duplicates OK)".

2. Examine the **Lookup** Tab: To understand the results of using the Lookup Wizard, refer to the **Lookup** tab behind the **General** tab (Figure 29). You can make adjustments to the *Lookup* properties here. In this case, no changes are needed.



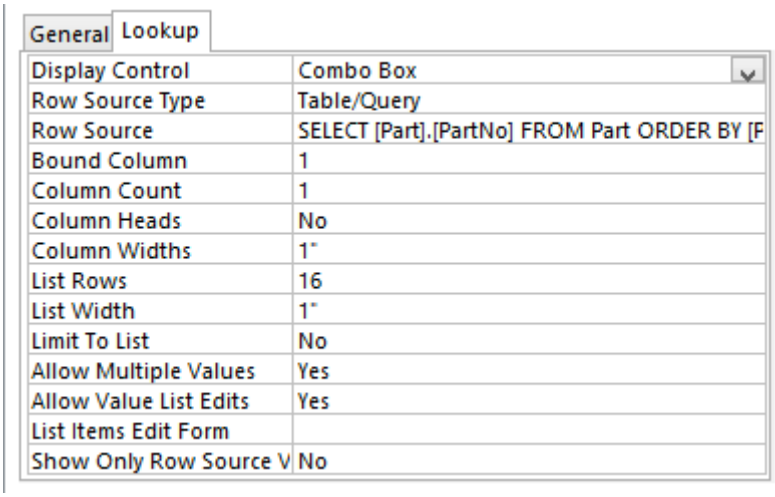


Figure 29: **Lookup** and **General** Tabs for *PartNo* Showing Field Properties

3. Set the Combined Primary Key: You will make a combined primary key consisting of the *PartNo* field and the *OrdNo* field.
   - Select the row with the *OrdNo* field. Access highlights the row after you select it.
   - While holding down the **Ctrl** key, select the row containing the *PartNo* field.
   - Click the **Key** icon 

in the toolbar to select both fields as the combined primary key. Now two rows have the key icon shown. Access considers these two fields as one combined primary key.
   - Move the row containing the *PartNo* field adjacent to the *OrdNo* field. To move the row, first select the row with the *PartNo* field. Then, drag and drop it so that it appears above the *QtyUsed* field. You need to drag it from the outside part of the row.
4. Examine the Design View: The final Design View should appear as in Figure 30. After you have inspected your window, save the table and close the view.

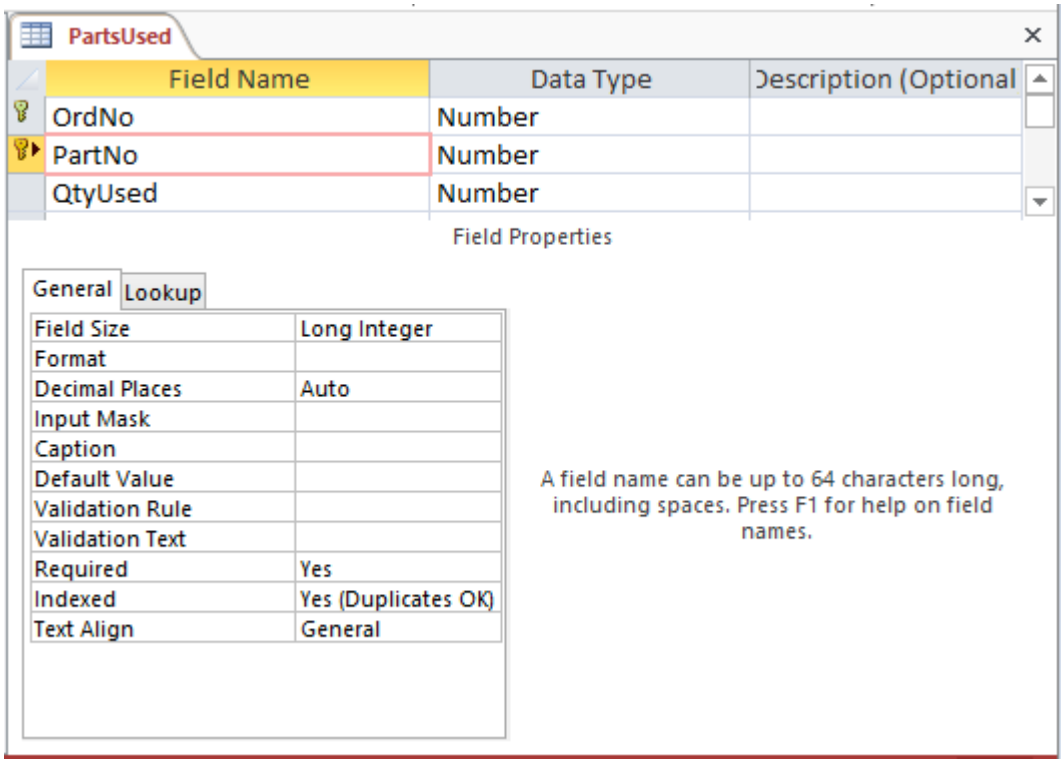


Figure 30: Design View Showing the Completed *PartsUsed* Table

2.4 Entering Data into the Tables

Entering data allows you to better understand the tables that you have created. Entering data may reveal the need for modifications and additions to your table design. This section demonstrates two ways for information systems professionals to enter initial data. Chapters 4 and 5 demonstrate data entry forms for end users.

2.4.1 Entering Data

This section demonstrates using a datasheet for entering a small number of records. Follow the steps below to begin entering data into the *Customer* table datasheet:

1. Open the Datasheet: Select the *Customer* table in the Database window and click on the **Open** button to bring up the datasheet. The empty datasheet should appear with the appropriate field names.
2. Refer to Figure 31: Use the completed datasheet as a guide to enter data into the *Customer* datasheet on your screen.
3. Enter Records: When the datasheet opens on your screen, the cursor will be positioned in the *CustNo* field ready for you to enter the first record (refer to Figure 12). Since Access fills the first field automatically (remember that *CustNo* is an "AutoNumber" data type), you must tab over to the next field and begin typing. A small pencil cursor appears on the left gray margin to indicate the current record. When you are finished entering data in a field, use the **Tab** key to advance to the next field. Each time you add a record, Access automatically saves the updated datasheet on your hard drive.
4. Close the Datasheet: When you are finished entering all records, your datasheet and Figure 30 should appear the same. Close the datasheet by clicking the **Close** button.

Customer

CustNo	FirstName	LastName	Address	City	State	PostalCode	PhoneNumber	Click to Add
1	Beth	Taylor	2396 Rafter Rd	Seattle	WA	98103-	(206) 221-9021	
2	Betty	Wise	4334 153rd NW	Seattle	WA	98178-	(206) 445-6982	
3	Bob	Mann	1190 Lorraine Cir.	Monroe	WA	98013-	(206) 326-1234	
4	Candy	Kendall	456 Pine St.	Seattle	WA	98105-	(206) 523-1112	
5	Harry	Sanders	1280 S. Hill Rd.	Fife	WA	98523-	(360) 444-0092	
6	Helen	Sibley	206 McCaffrey	Renton	WA	98006-	(206) 624-0362	
7	Homer	Wells	123 Main St.	Seattle	WA	98105-	(206) 524-1461	
8	Jerry	Wyatt	16212 123rd Ct.	Seattle	WA	98266-	(206) 524-8145	
9	Jim	Glussman	1432 E. Revenna	Seattle	WA	98266-	(206) 445-2139	
10	Larry	Styles	9825 S. Crest Lane	Bellevue	WA	98104-	(425) 745-9980	
11	Mike	Boren	642 Crest Ave.	Fife	WA	98523-	(360) 444-5678	
12	Ron	Thompson	789 122nd St.	Renton	WA	98666-	(360) 747-2222	
13	Sharon	Johnson	1223 Meyer Way	Fife	WA	98222-	(360) 333-6666	
14	Sheri	Gordon	336 Hill St.	Seattle	WA	98103-	(206) 525-3344	
15	Todd	Hayes	1400 NW 88th	Lynnwood	WA	98036-	(206) 775-7689	
16	Wally	Jones	411 Webber Ave.	Seattle	WA	98105-	(206) 523-9957	
(New)								

Record: 1 of 16 No Filter Search

Figure 31: Completed *Customer* Table Datasheet

2.4.2 Importing Data into the Tables

The data for the next three tables are imported from the lab book's web site. You will use the Get External Data Wizard that enables you to create table definitions as well as load data. In this section, you will use the wizard only to load data. In the next section, you will use the wizard to create table definitions as well as to load data.

The Get External Data Wizard is quite useful because it allows you to load data from a variety of data sources including other Access databases, Excel spreadsheets, Sharepoint Lists, text files, and XML files into an Access database. Often, when you create a new database, you need to import data from other sources. Sometimes database creation is part of a process to convert from an old system to a new system.

The data to populate your database are contained in the ASCII files RepairOrder.txt, Part.txt, and PartsUsed.txt available in the lab book's website. Follow the instructions below to import these files. After you import the *RepairOrder* data, repeat the steps to import the data for the *Part* table and then the *PartsUsed* table.

- Select the Ribbon's **External Data → Text File** button from the Import & Link group.

- The initial window of the Get External Data Wizard window appears (Figure 32). Select the "RepairOrder" file using the **Browse …** button. You also need make the decision of whether the data go into a new table (this allows you to import a complete table) or into an existing table. Select the second choice "Append a copy of the records to the table". The select the *RepairOrder* table from the drop down list as shown in Figure 32. Click **OK** to continue.

- In the next window (Figure 33), choose "Delimited" and click the **Next** button.

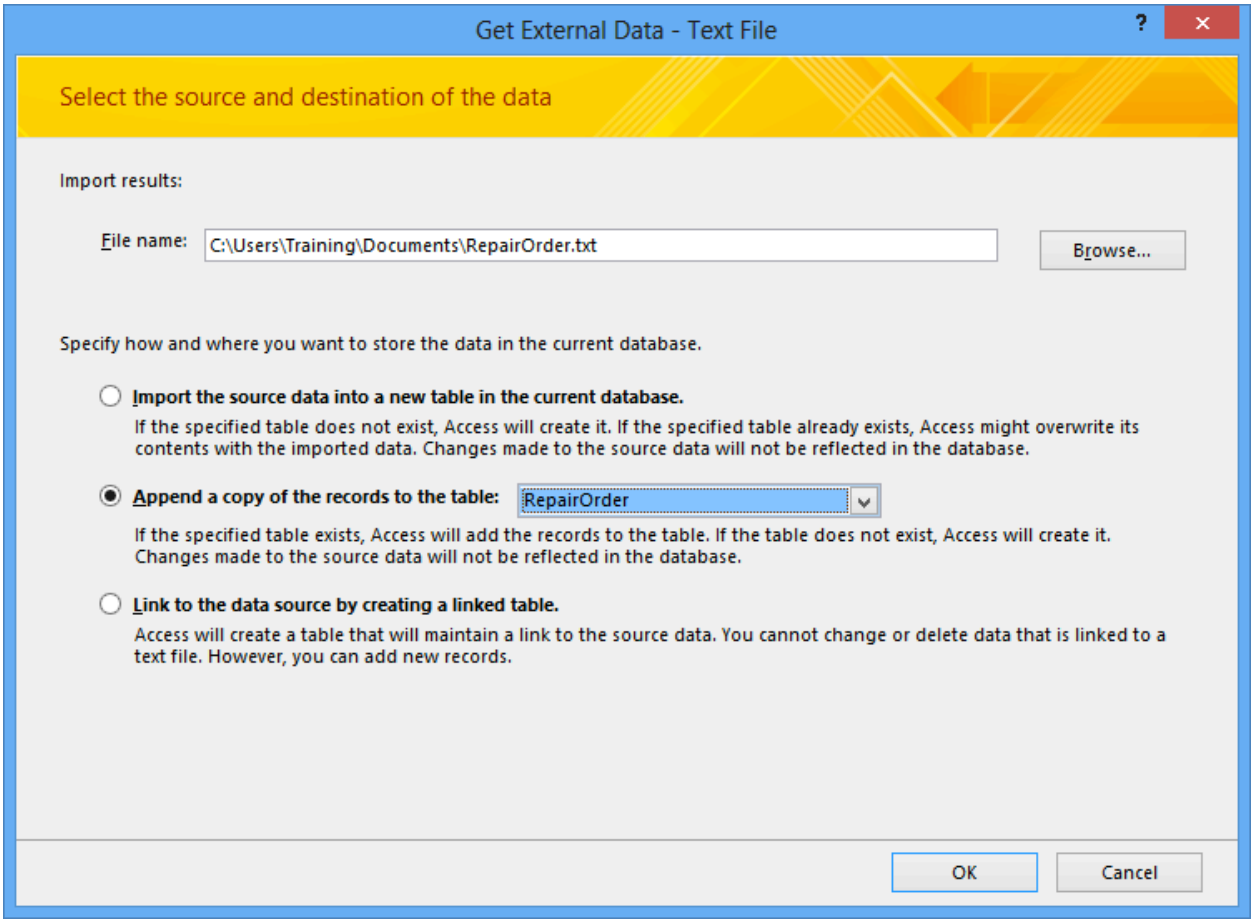


Figure 32: Initial Window of the Get External Data Wizard for Text Files

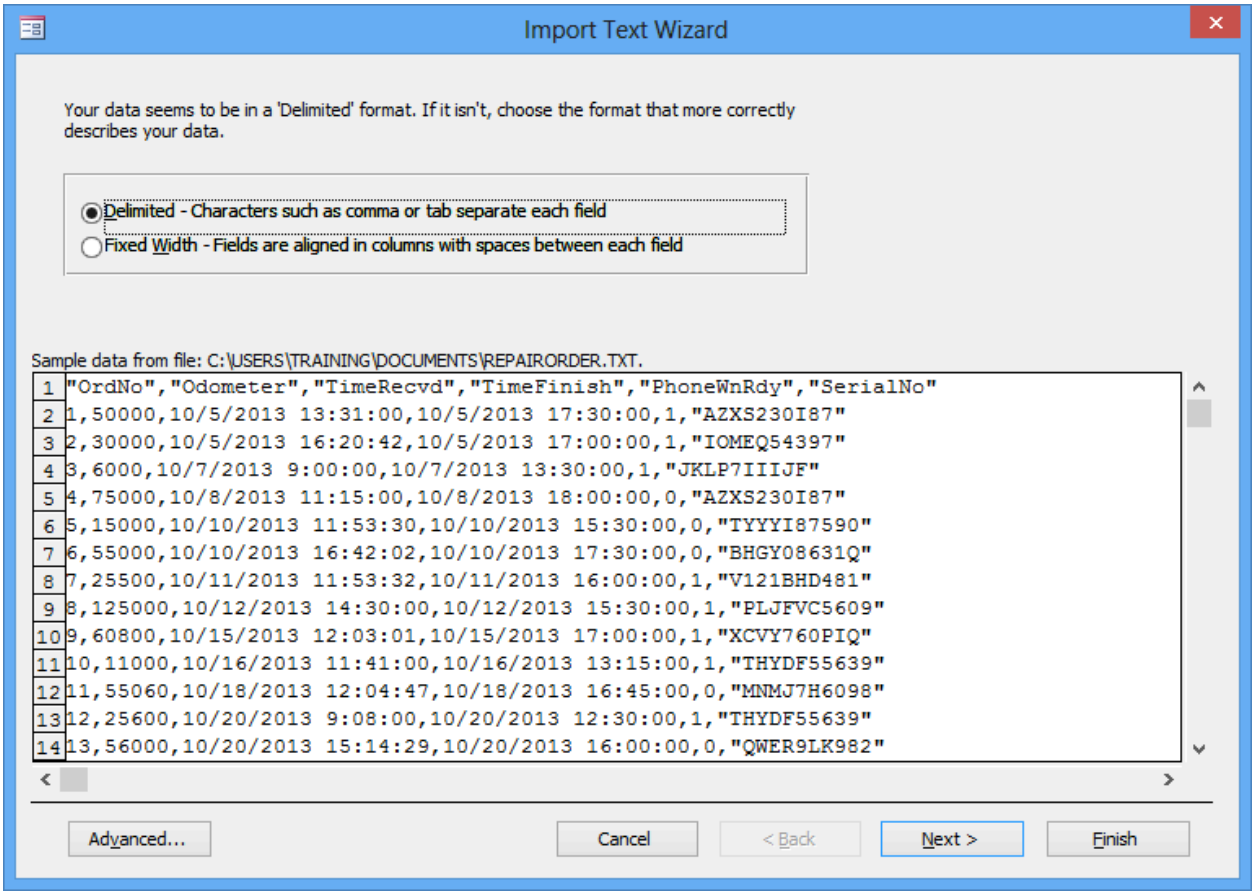


Figure 33: Data Format Window of the Get External Data Wizard for Text Files

- Next, select the following options as shown in Figure 34: (1) "Comma" as the field separator, (2) "First Row Contains Field Names", and (3) double quotation mark (") as the text qualifier, and click **Next** to continue.
- The last wizard window, Figure 35, asks you where to import the data. It should already say *RepairOrder*, so click **Finish**.

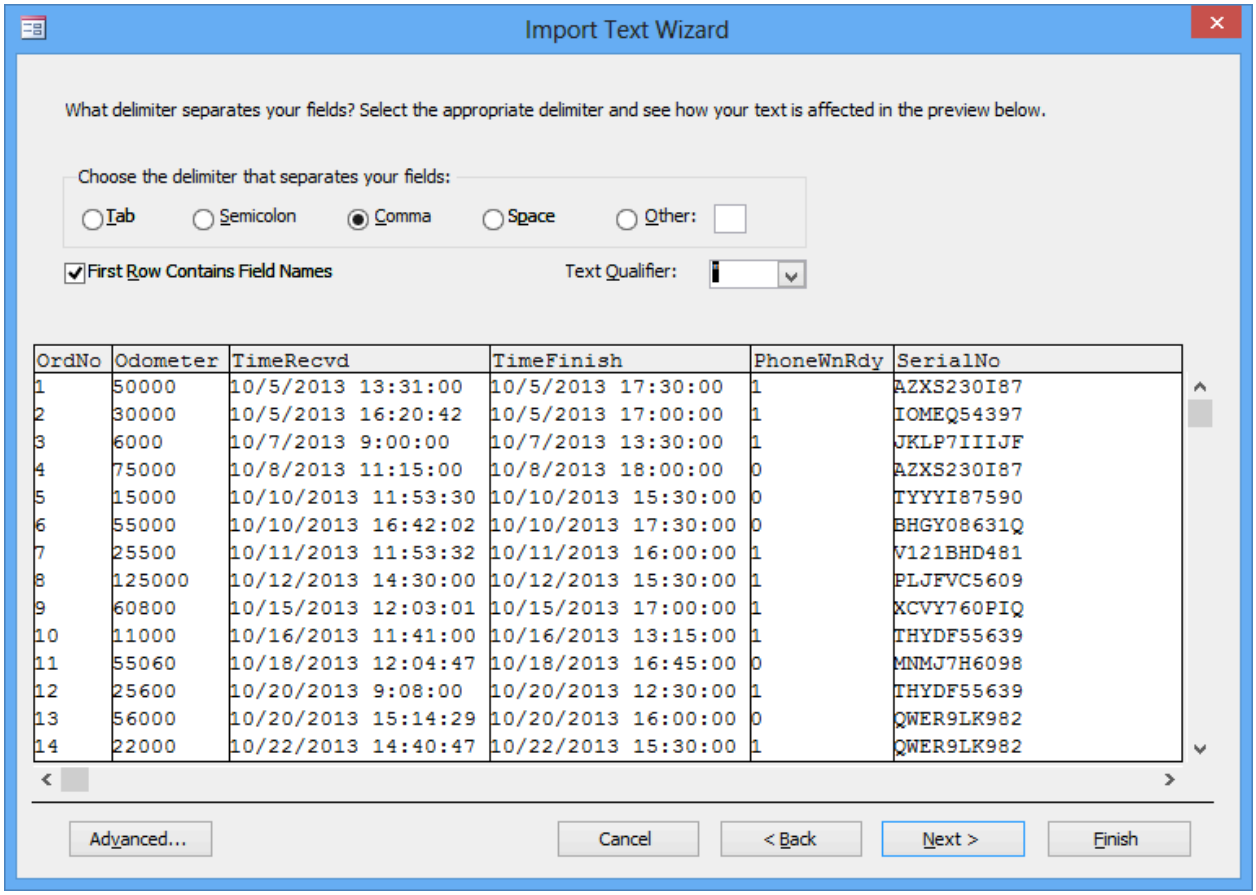


Figure 34: Delimiter Option Window in the Get External Data Wizard for Text Files

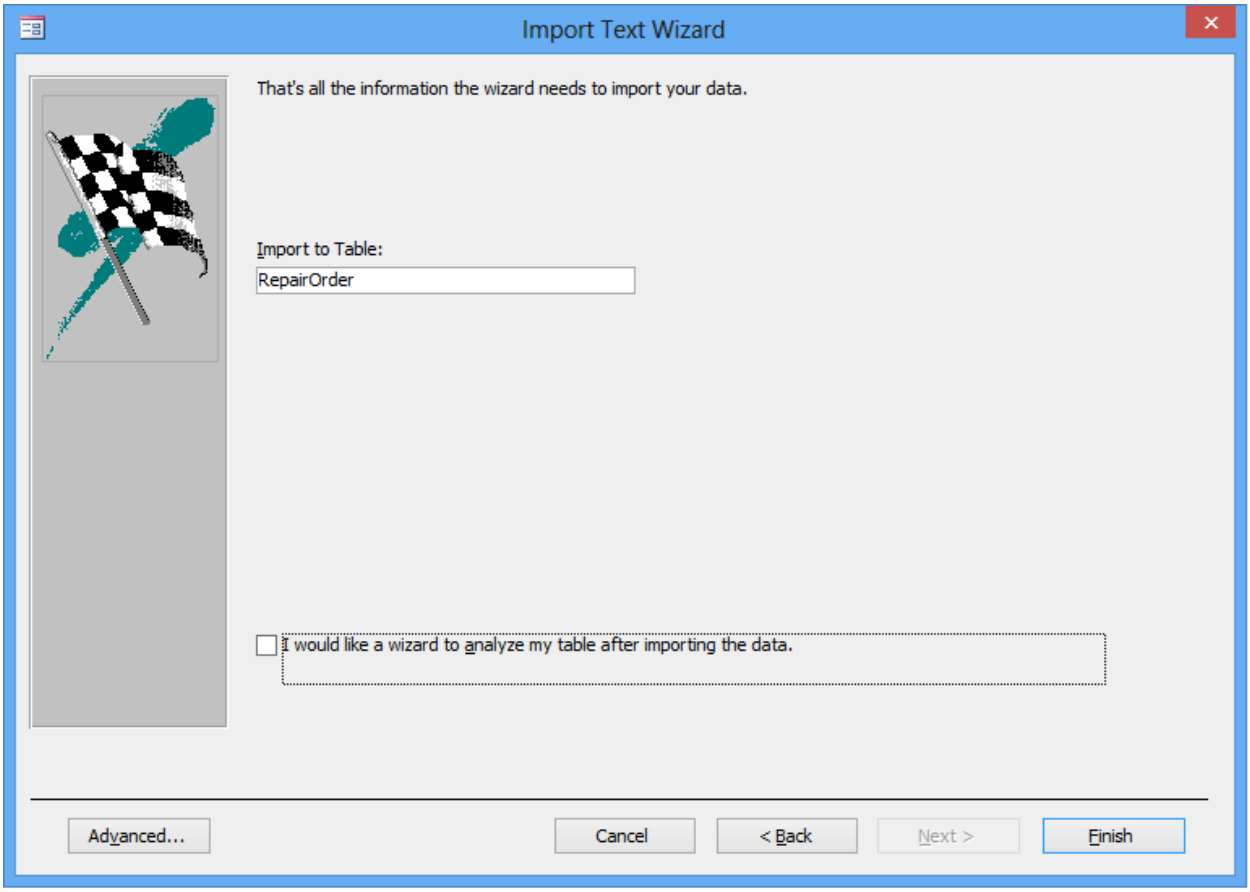


Figure 35: Final Window of the Get External Data Wizard for Text Files

- When you are finished importing the data for the three tables, your completed datasheets of each table should appear as in Figures 36, 37, and 38.

RepairOrder

OrdNo	Odometer	TimeRecvd	TimeFinish	PhoneWnRdy	SerialNo
1	50000	10/5/2013 1:31:00 PM	10/5/2013 5:30:00 PM	☑	AZXS230I87
2	30000	10/5/2013 4:20:42 PM	10/5/2013 5:00:00 PM	☑	IOMEQ54397
3	6000	10/7/2013 9:00:00 AM	10/7/2013 1:30:00 PM	☑	JKLP7IIIJF
4	75000	10/8/2013 11:15:00 AM	10/8/2013 6:00:00 PM	☐	AZXS230I87
5	15000	10/10/2013 11:53:30 AM	10/10/2013 3:30:00 PM	☐	TYYYI87590
6	55000	10/10/2013 4:42:02 PM	10/10/2013 5:30:00 PM	☐	BHGY08631Q
7	25500	10/11/2013 11:53:32 AM	10/11/2013 4:00:00 PM	☑	V121BHD481
8	125000	10/12/2013 2:30:00 PM	10/12/2013 3:30:00 PM	☑	PLJFVC5609
9	60800	10/15/2013 12:03:01 PM	10/15/2013 5:00:00 PM	☑	XCVY760PIQ
10	11000	10/16/2013 11:41:00 AM	10/16/2013 1:15:00 PM	☑	THYDF55639
11	55060	10/18/2013 12:04:47 PM	10/18/2013 4:45:00 PM	☐	MNMJ7H6098
12	25600	10/20/2013 9:08:00 AM	10/20/2013 12:30:00 PM	☑	THYDF55639
13	56000	10/20/2013 3:14:29 PM	10/20/2013 4:00:00 PM	☐	QWER9LK982
14	22000	10/22/2013 2:40:47 PM	10/22/2013 3:30:00 PM	☑	QWER9LK982
15	14800	10/25/2013 9:05:00 AM	10/25/2013 10:15:00 AM	☐	JKLP7IIIJF
16	46000	10/25/2013 11:15:00 AM	10/25/2013 12:05:00 PM	☐	THYDF55639

Record: 1 of 32 No Filter Search

Figure 36: *RepairOrder* Datasheet Showing the First 10 Records

Part

PartNo	PartDesc	UnitsinStock	UnitPrice	UnitSize
1	10W-40 oil	145	$1.00	quart
2	oil filter	14	$2.00	item
3	AntiFreeze	10	$3.95	quart
4	Spark Plugs	45	$0.99	item
5	Transmission Fluid	15	$1.50	quart
6	10W-30 oil	95	$0.95	quart
7	Brake Lining	25	$5.00	quart
8	Shock	75	$6.00	item
9	Muffler	10	$15.00	item
10	Tail Pipe	85	$4.00	item
11	Head Gasket	32	$9.00	dozen
12	Timing Chain	6	$22.50	item
13	Battery	13	$55.00	item
14	Radiator	3	$60.00	item
15	Radiator Hose	24	$3.00	dozen
16	Rotor	4	$16.00	item

Record: 1 of 20 No Filter Search

Figure 37: *Part* Datasheet Showing the First 8 Records

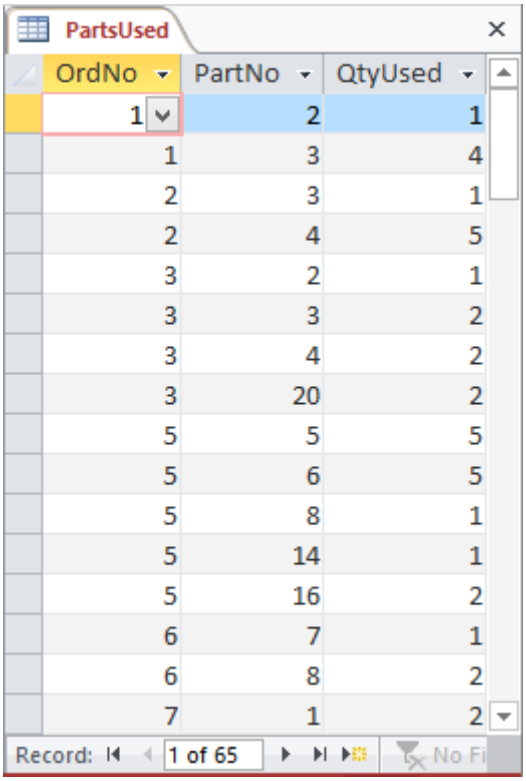


Figure 38: *PartsUsed* Datasheet Showing a Subset of Records

2.5 Importing the Last Auto Repair Table

To complete the database tables, you will import the last table. In the previous section, you used the Get External Data Wizard for Text Files to load data into an existing table. In this section, you will use the wizard to load the table structure (table and field names) as well as to load the data. The first subsection guides you through the steps of the Get External Data Wizard for Text Files. The second subsection guides you to make adjustments to field properties.

2.5.1 Importing the Vehicle *Table*

1. Start the Get External Data Wizard: Select the Ribbon's **External Data → Text File** button from the Import group.
2. Choose the Text File to Import: In the first wizard window, select the "Vehicle.txt" file using the **Browse …** button.
3. Select the Data Storage Option: Choose "Import the source data into a new table in the current database." as shown in Figure 39. Click **Next**.
4. Select the Data Format Option: In the next window, choose "Delimited" and click the **Next** button. This is similar to what you did in (Figure 33).
5. Select the Delimiter Type: Next, select the following options: (1) "Comma" as the field separator, (2) "First Row Contains Field Names", and (3) double quotation mark (") as the text qualifier, and click **Next** to continue. See (Figure 34) above.
6. Select the Fields: In the next window (Figure 40), you should keep all fields since you can make changes later in the Design View window. Click the **Next >** button.

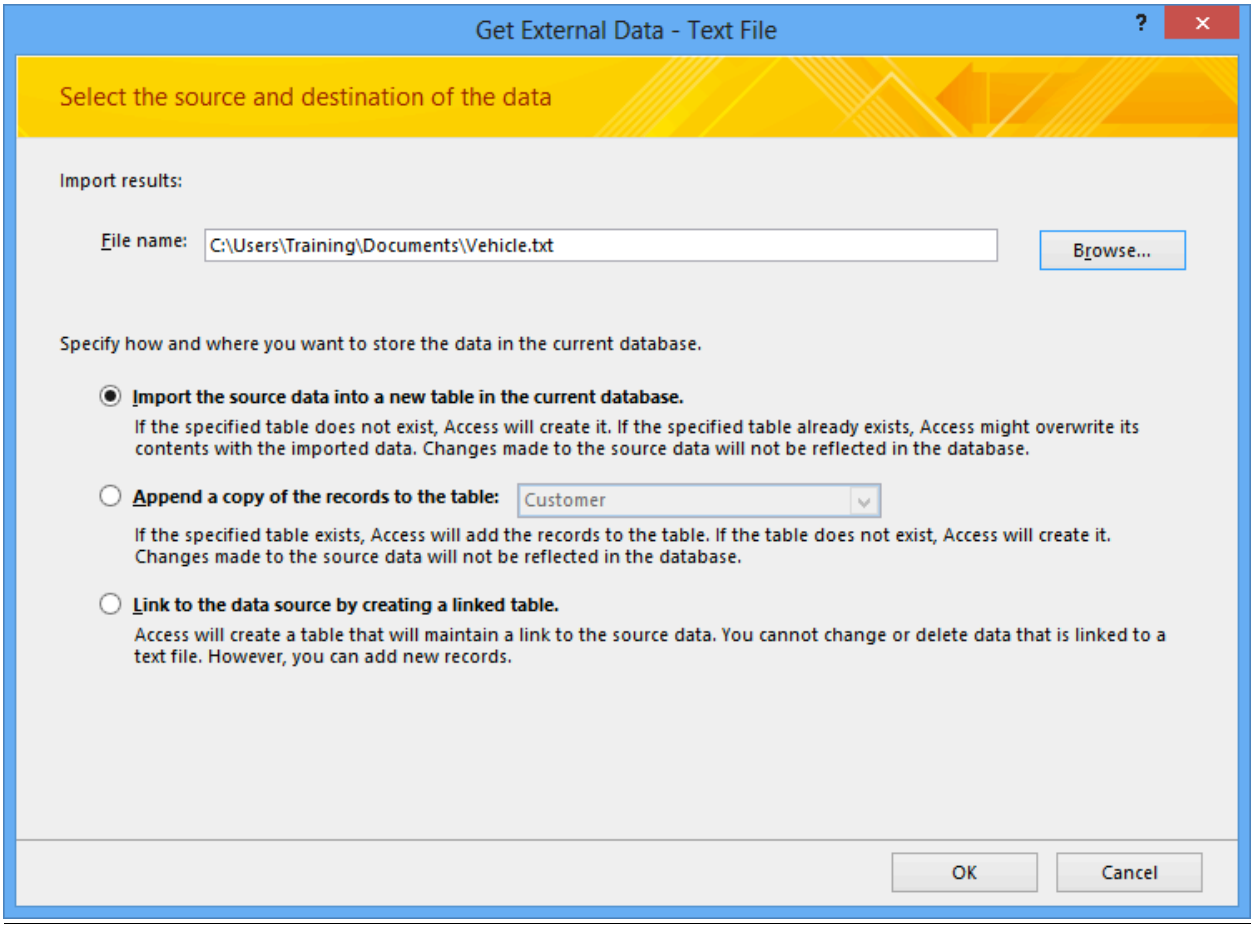


Figure 39: Source and Destination Options in the Get External Data Wizard

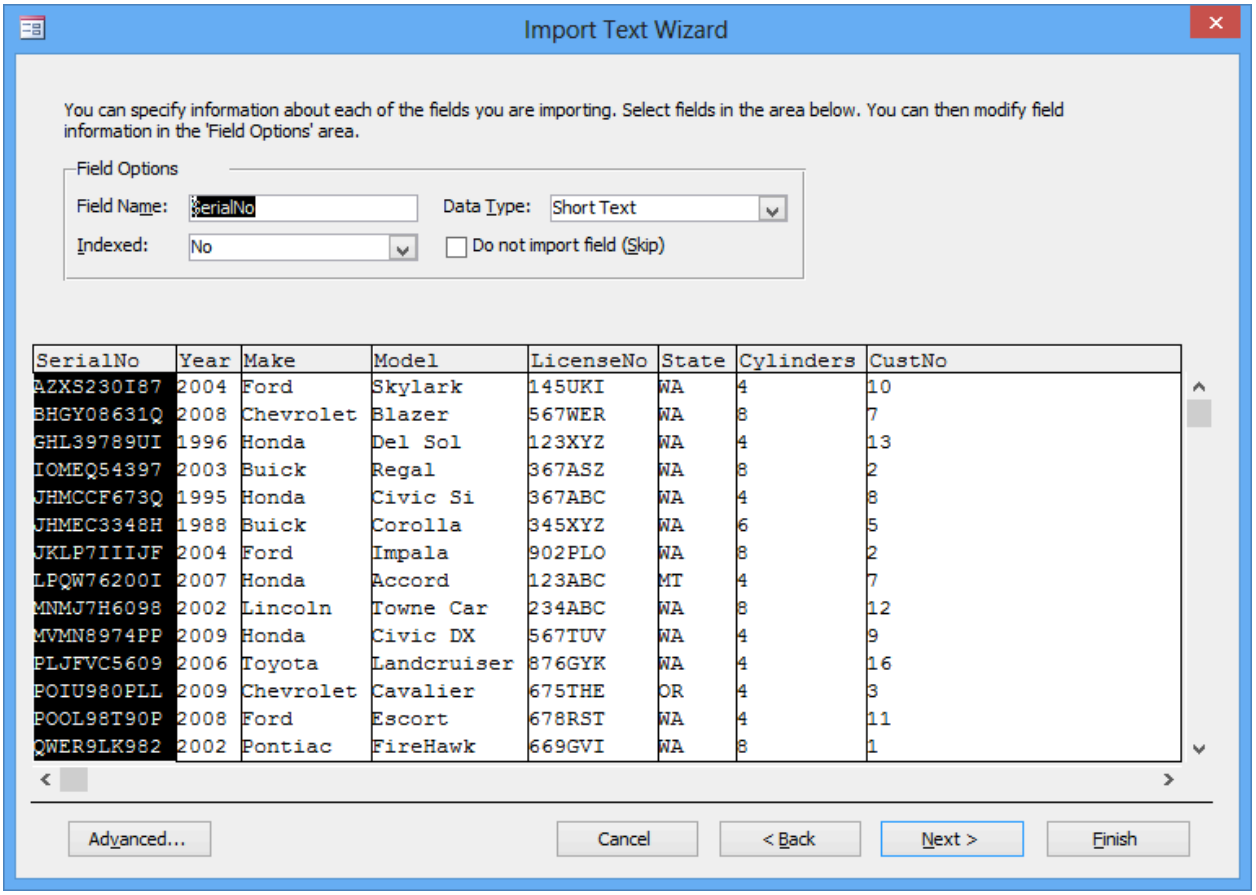


Figure 40: Field Modification in the Import Text Wizard

7. Choose the Primary Key: In the next window (Figure 41) select "Choose my own Primary Key". Make sure that the *SerialNo* field is selected.

8. Designate the Table Name: In the last window (Figure 42) make sure that *Vehicle* is the table name since Access uses the imported file name as the default name. Click the **Finish** button to import your table with data.

9. Correct Errors: If you make an error, do not worry. Access is very forgiving at this stage. If you are still inside the wizard, use the **Back <** button to retrace your steps. If you have finished the wizard, you can proceed to Design View. If you want to try importing again, simply delete the table and import the table again.

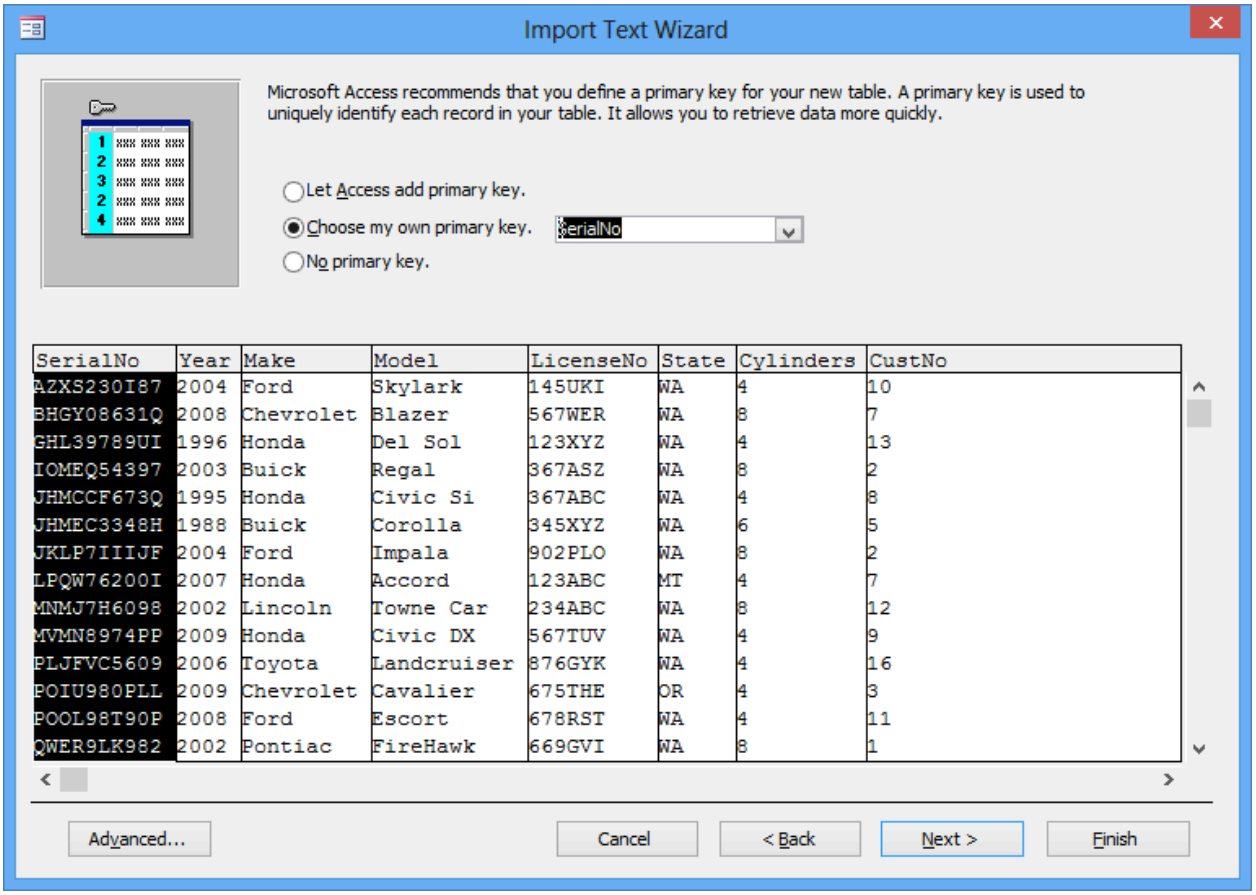


Figure 41: Choosing the Primary Key in the Import Text Wizard

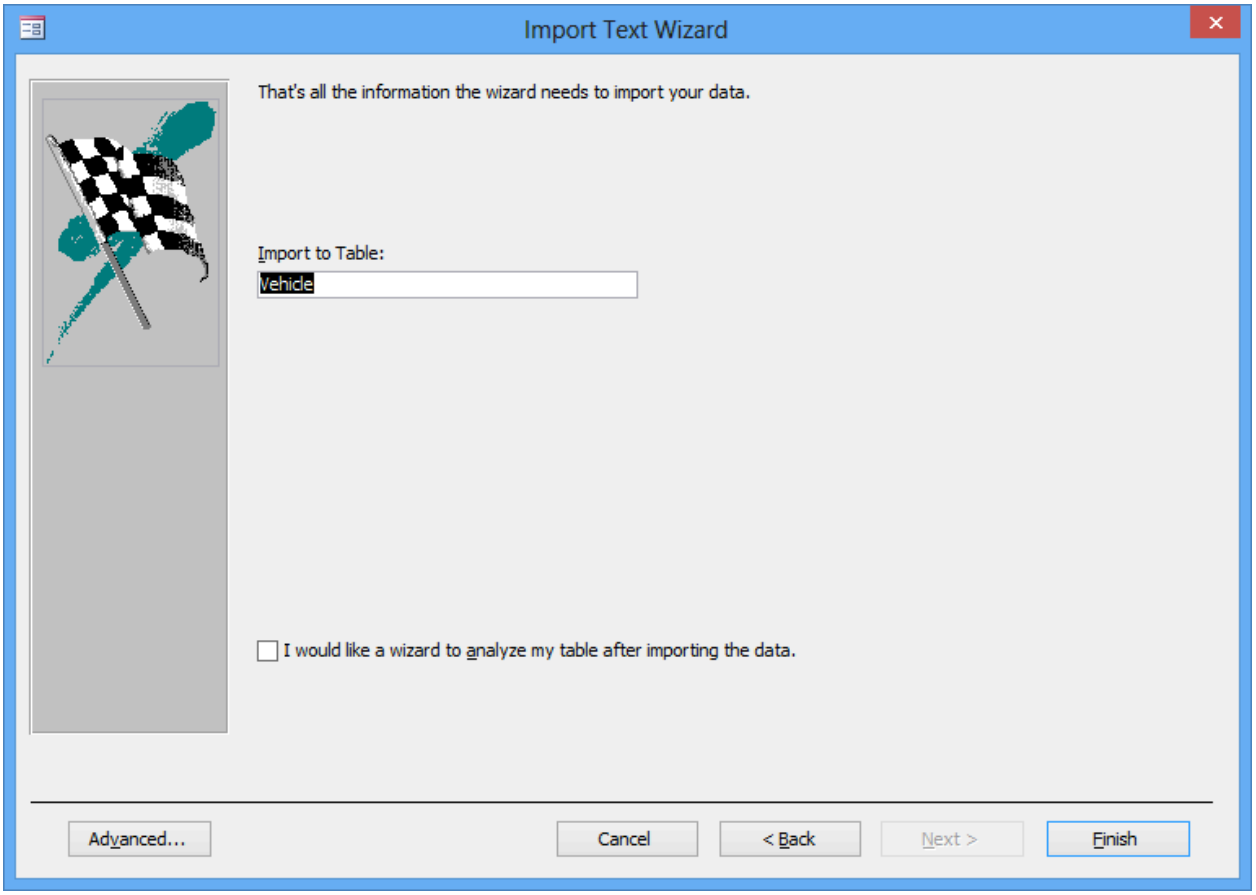


Figure 42: Final Wizard Window for Importing the *Vehicle* Table

2.5.2 Modify Table Definitions

After importing the *Vehicle* table, you may need to change field names, data types, and field properties. For each field, the Import Wizard chooses a data type based on the data that were loaded. You will usually need to make some changes even though the Import Wizard does a good job in most cases. You also may want to change tables related to the imported table. In this case, you should return to the *RepairOrder* table and assign the "Lookup" data type to the *SerialNo* field since it is a foreign key linking the *RepairOrder* table to the *Vehicle* table.

- Select the *RepairOrder* table in the navigation pane and open it in Design View. In the *SerialNo* field, select "Lookup Wizard" from the data type list. When you are finished, return to this field and continue to modify the *Vehicle* table.
- Modify the *Vehicle* table using Table 5 so that your table appears as in Figure 43. Note that you will use the Lookup Wizard for the *CustNo* field since it is a foreign key field. You also will use the Lookup Wizard for the *State* field, except that the data source will be a list of values you type in the wizard instead of a field (see following section).
- When you are finished, save the table. Open the table in datasheet view (Figure 44) to see the changes.

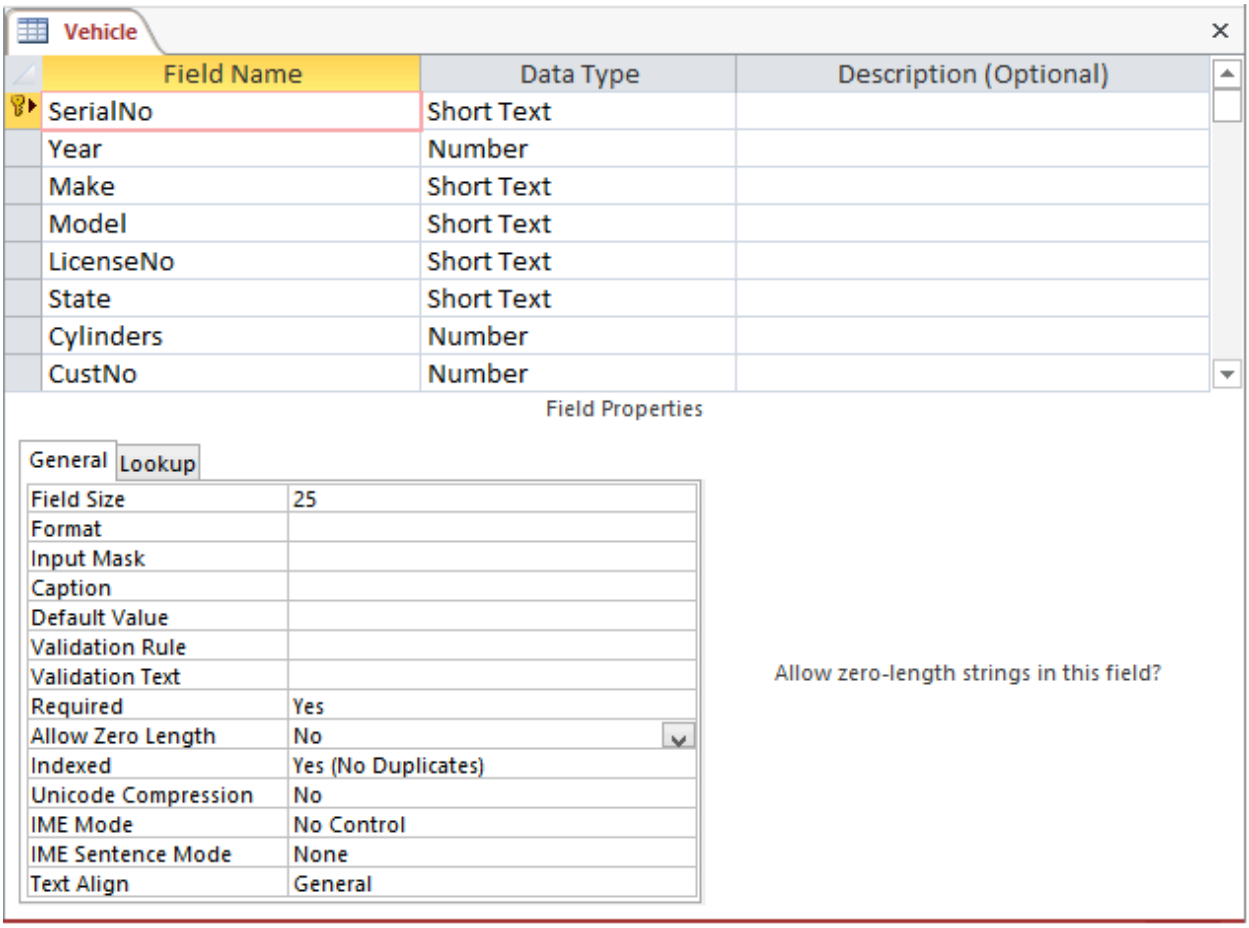


Figure 43: Design View Window of the Completed *Vehicle* Table

Table 5: Fields in the *Vehicle* Table

Field Name	**Data Type**	**Field Properties**
SerialNo (Serial number of auto being repaired)	Short Text (Although this is the primary key field, the data type is set to "Short Text" since many auto serial numbers also contain a few letters.)	*Field Size*: "25" *Required*: "Yes" *Allow Zero Length*: "No" *Indexed*: "Yes (No Duplicates)"
Year	Number	*Field Size*: "Long Integer" *Decimal Place*: "0" *Validation Rule*: "> 1900" *Validation Text*: "Year must be greater than 1900." *Required*: "No" *Indexed*: "No"
Make	Short Text	*Field Size*: "12" *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"
Model	Short Text	*Field Size*: "15" *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"
LicenseNo	Short Text	*Field Size*: "6" *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"
State	Short Text (Assign Lookup feature)	*Field Size*: "5" *Default Value*: WA *Required*: "No" *Allow Zero Length*: "No" *Indexed*: "No"
Cylinders (The number of cylinders of the auto; the *Validation Rule* property uses the In function.)	Number	*Field Size*: "Integer" *Default Value*: 4 *Required*: "Yes" *Validation Rule*: "In(4,6,8)" *Validation Text*: "Cylinders must be 4, 6, or 8."

		Indexed: "No"
CustNo (Foreign key from the *Customer* table)	Number (Assign Lookup feature)	*Field Size*: "Long Integer" *Decimal Place*: "Auto" *Default Value*: "0" *Required*: "Yes" *Indexed*: "No"

Vehicle

SerialNo	Year	Make	Model	LicenseNo	State	Cylinders	CustNo
AZXS230I87	2004	Ford	Skylark	145UKI	WA	4	10
BHGY08631Q	2008	Chevrolet	Blazer	567WER	WA	8	7
GHL39789UI	1996	Honda	Del Sol	123XYZ	WA	4	13
IOMEQ54397	2003	Buick	Regal	367ASZ	WA	8	2
JHMCCF673Q	1995	Honda	Civic Si	367ABC	WA	4	8
JHMEC3348H	1988	Buick	Corolla	345XYZ	WA	6	5
JKLP7IIIJF	2004	Ford	Impala	902PLO	WA	8	2
LPQW76200I	2007	Honda	Accord	123ABC	MT	4	7
MNMJ7H6098	2002	Lincoln	Towne Car	234ABC	WA	8	12
MVMN8974PP	2009	Honda	Civic DX	567TUV	WA	4	9

Record: 1 of 23 No Filter Search

Figure 44: *Vehicle* Datasheet Showing the First 10 Records

Adding Your Own Values to the Lookup Wizard

Go back into the data type list and select "Lookup Wizard" for the *State* field to invoke the initial window of the Lookup Wizard. The following steps guide you in the choices of the Lookup Wizard.

1. Choose the Kind of Value Source: You have two choices; choose the second one, "I will type in the values that I want" (Figure 45). Click **Next**.

2. Type Values into the *Lookup* Column: Leave the column setting at "1". Click in the first column and type "WA". Use the **Tab** or **Arrow Down** key (do not hit **Enter**) and go to the next row and type "OR". In the following rows type "CA", "MT", and "ID" (see Figure 46). Click **Next**.

3. Name the *Lookup* Column: The final wizard window asks you to name the *Lookup* Column. Type "State" and click **Finish** (Figure 47). Also view the appearance of the field property **Lookup** tab (Figure 48). Figures 43 and 44 show the final design view and the datasheet of the *Vehicle* table.

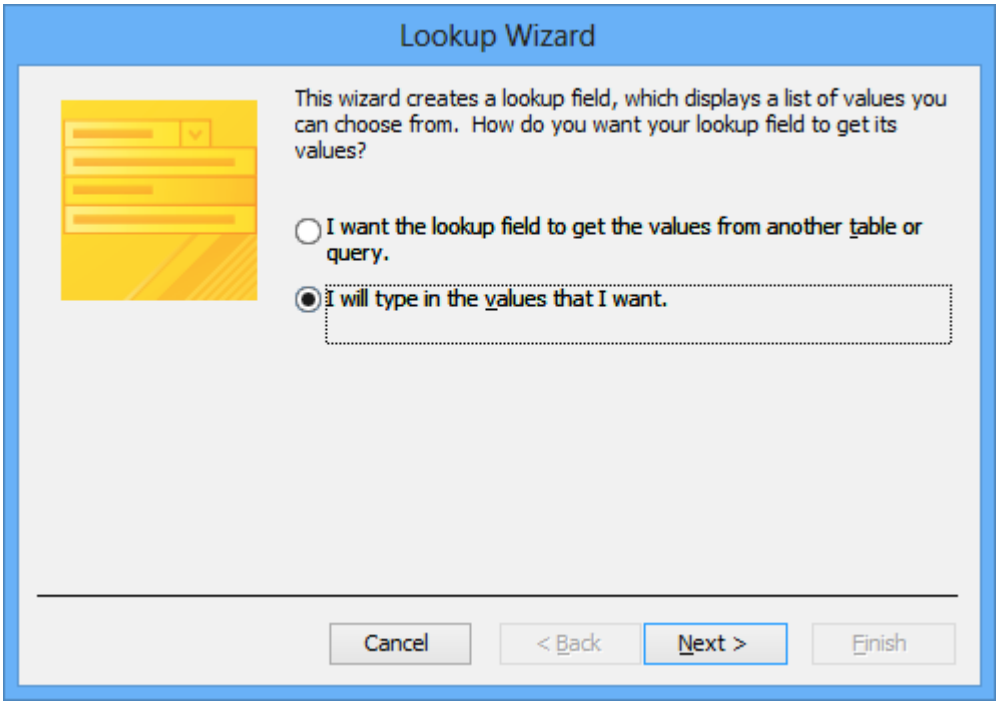


Figure 45: Initial Lookup Wizard Window

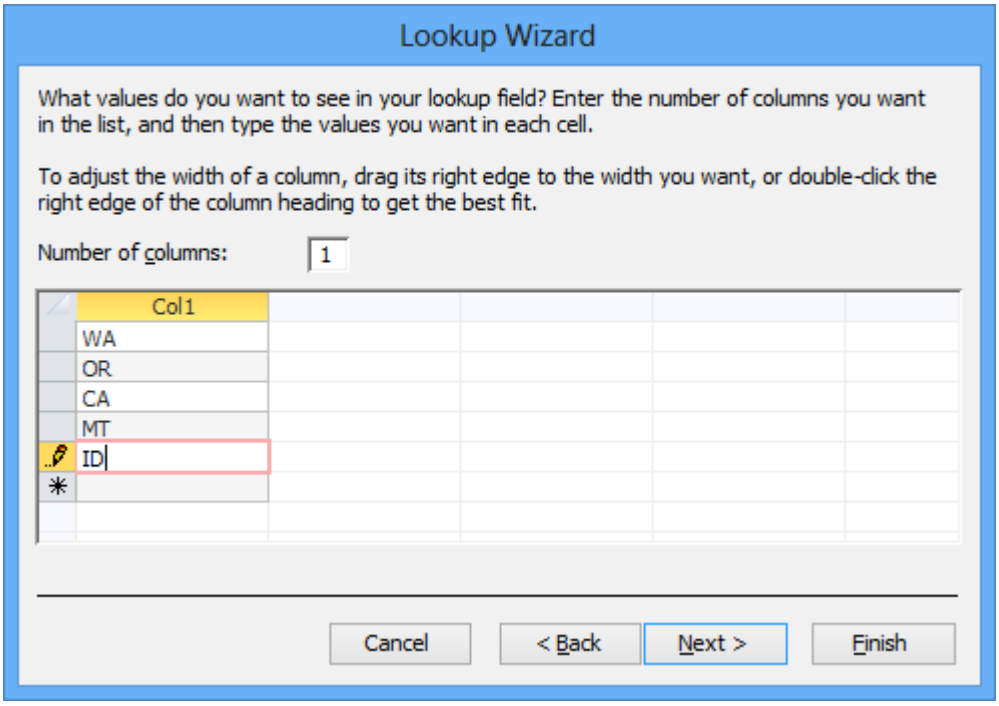


Figure 46: Data Typed into the Column

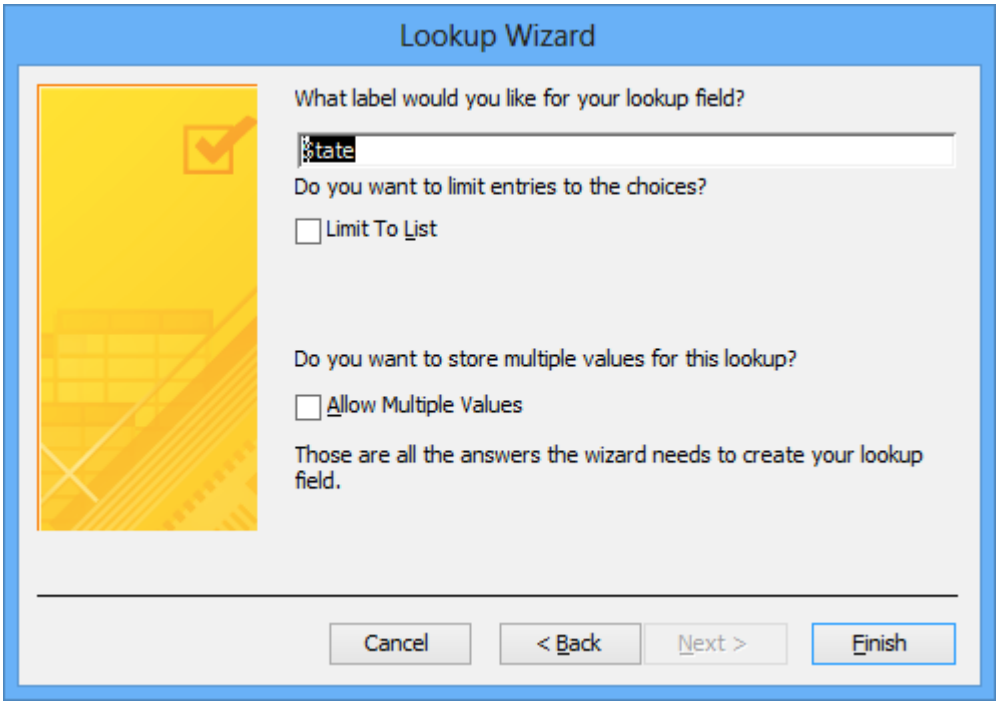


Figure 47: Final Lookup Wizard Window

General | Lookup

Display Control	Combo Box
Row Source Type	Value List
Row Source	"WA";"OR";"CA";"MT";"ID"
Bound Column	1
Column Count	1
Column Heads	No
Column Widths	1"
List Rows	16
List Width	1"
Limit To List	No
Allow Multiple Values	No
Allow Value List Edits	No
List Items Edit Form	
Show Only Row Source V	No

Figure 48: Field Property Lookup Tab

2.6 Creating Relationships

After creating the tables, you can link them together. Typically, you should not link the tables until the database design is complete. You would begin this process by adding tables from the Show Table window into the Relationships window. An exception to this would be if the Lookup Wizard was used to create a field containing values from another table. In this case, those tables automatically appear in the Relationships window while the remaining tables must be added. The following steps explain how to create relationships and rules about referenced rows using the Relationships window.

Creating Relationships with Tables Already Linked

1. Open the Relationships Window: In the ribbon's **Database Tools**, click the **Relationships** button to open the Relationships window. If the Lookup Wizard has been used, tables will already appear in the window. However, they may be poorly organized with crossing connection lines (Figure 49).

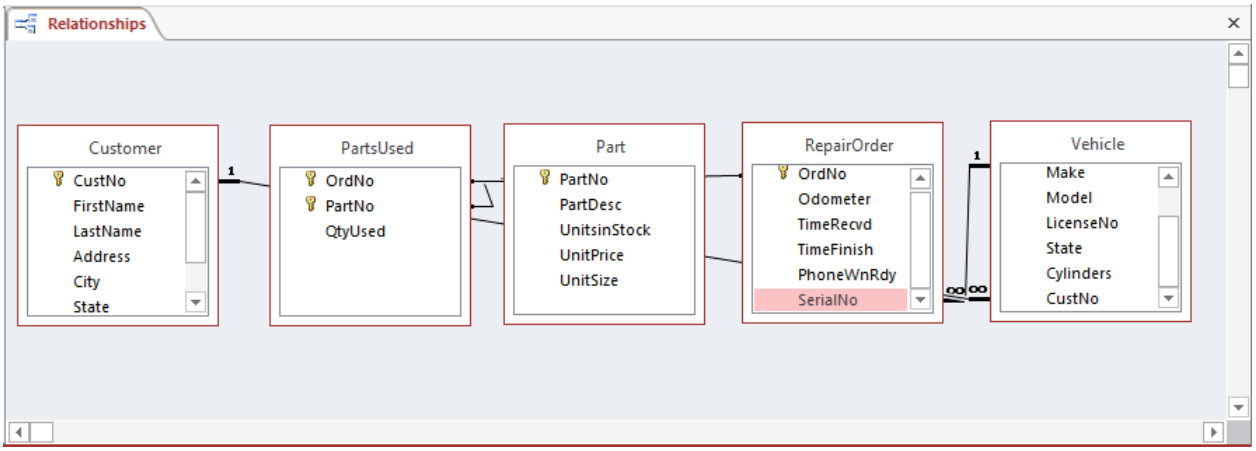


Figure 49: Disorganized Table Order and Connecting Lines

2. Add Tables Not Shown: You may still need to add the remaining tables not shown in the Relationships window. In the ribbon's Design group, click **Show Table** and a window will appear listing the available tables as shown in Figure 50. For each table that you want to add, select it and click the **Add** button. (For this exercise, this step is not necessary.)

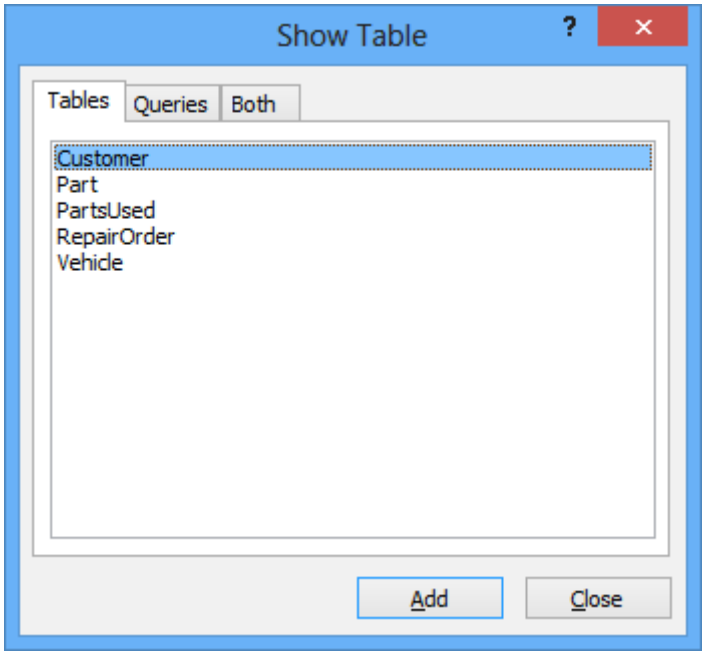


Figure 50: Show Table Window

3. Drag the Tables into Proper Order: Hold the mouse button down and drag the tables beside each other in the following order: *Customer, Vehicle, RepairOrder, PartsUsed,* and *Part* (Figure 51). If the tables are already in order, disregard this step.

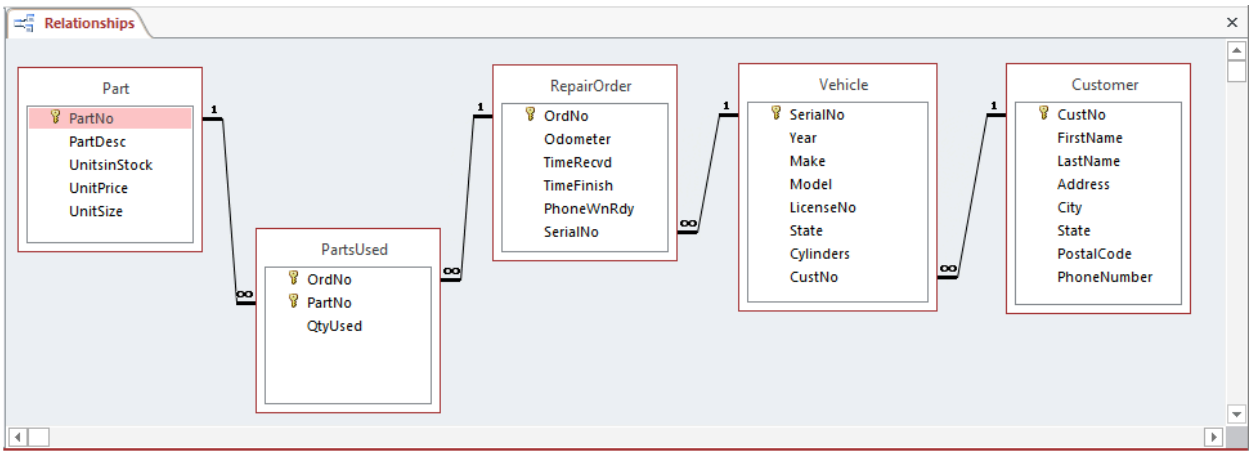


Figure 51: Relationships Window after Rearranging Tables

4. Link Tables: When all of the tables are in the Relationships window, there may not be connecting lines between all of the tables. (In this situation, lines connect all of the tables.) But if you had to create relationships between the tables, drag the primary key field from the parent table to the child table that contains the parent table's primary key. For example, (although these tables are already linked) you would have dragged the *Customer* table's *CustNo* primary key field to the *Vehicle* table's *CustNo* foreign key field. A connecting line would appear as in Figure 50. Each time you do this, the Edit Relationships dialog appears (Figure 52).

5. Enforce Referential Integrity: When the tables and connecting lines are organized, you must enforce referential integrity for each relationship. Beginning with the line connecting the *Customer* table and the *Vehicle* table, point at the line and double-click. Each time you do this, a dialog box appears asking you questions (Figure 52). Choose "Enforce Referential Integrity" for each relationship. (When you close the dialog box, a bold line will appear connecting the tables along with the 1-to-M relationship symbols, as shown in Figure 53.)

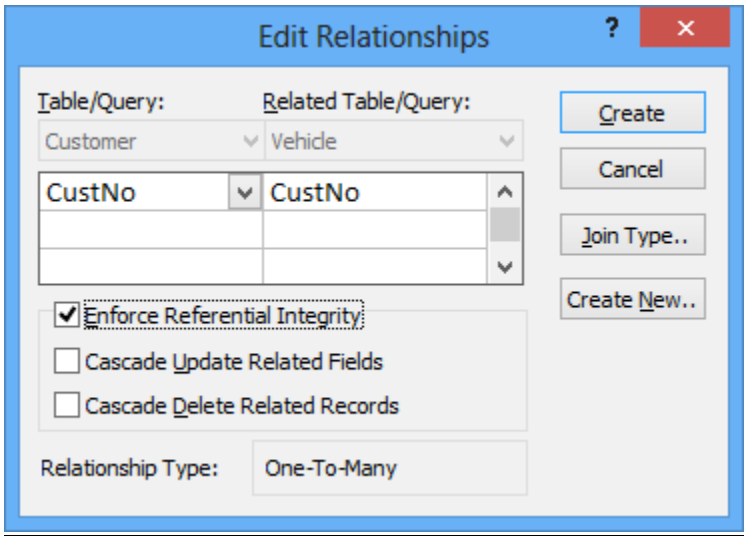


Figure 52: Dialog Box to Specify Details of a Relationship

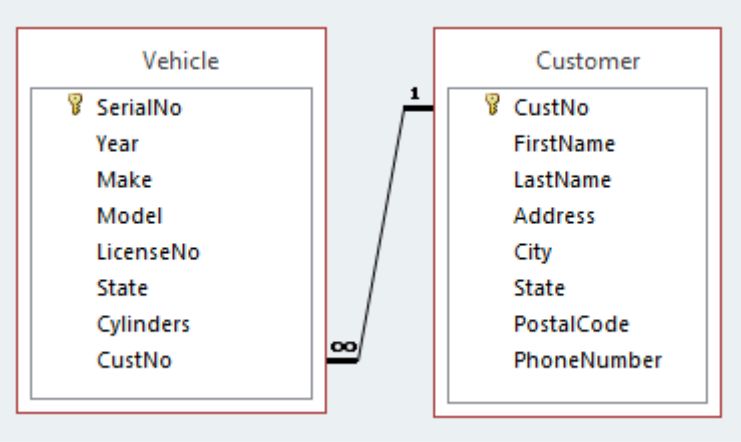


Figure 53: Solid Lines Appear after Enforcing Referential Integrity

- For the relationships between the *RepairOrder* and the *PartsUsed* tables (a many-to-many relationship), you also choose both "Cascade…" check boxes (see Figure 54). You may want to refer back to textbook Chapter 3 to review the concepts of referential integrity and the rules about referenced rows.

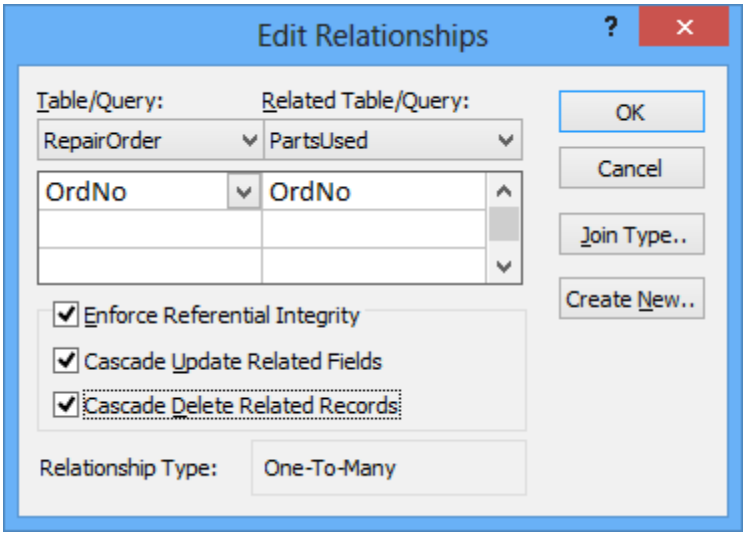


Figure 54: Dialog Box with All Three Details of a Relationship Selected

6. Inspect the Final Result: After enforcing referential integrity, your window should appear as in Figure 55.

If the Relationships window shows duplicate tables and relationships, you must delete the duplicate relationships first and then delete the duplicate tables. To delete a relationship or table, select it and press the **Delete** key. If you delete the duplicate tables first, you will not fix the problem. The next time you open the Relationships window, you will see the duplicate tables and relationships again.

Creating Relationships without Tables in the Relationships Window

If you are in the situation where there are no tables in the Relationships window, follow the steps below.

1. Open the Relationships Window: In the ribbon's Design group, click **Show Table**. A dialog box appears listing the available tables, as shown in Figure 50. For each table, select it and click the **Add** button.

2. Link Tables: When all of the tables are in the Relationships window, there will be no connecting lines showing relationships. To link the tables, you will drag the primary key field of the "one" table to the table to be linked (the "many" table) that contains the "one" table's primary key. Begin by dragging the *Customer* table's *CustNo* primary key field to the *Vehicle* table's *CustNo* foreign key field. A connecting line would appear. Each time you do this, a Relationships dialog box appears asking you questions (Figure 52). Choose "Enforce Referential Integrity" for each relationship; if a many-to-many relationship is required, then choose both "Cascade…" check boxes (see Figure 52). When you close the dialog box, a bold line will appear connecting the tables along with the 1-to-M relationship marks as in Figure 53. Continue doing this for the remaining tables.

3. Inspect the Final Result: After enforcing referential integrity, your window should appear as in Figure 55.

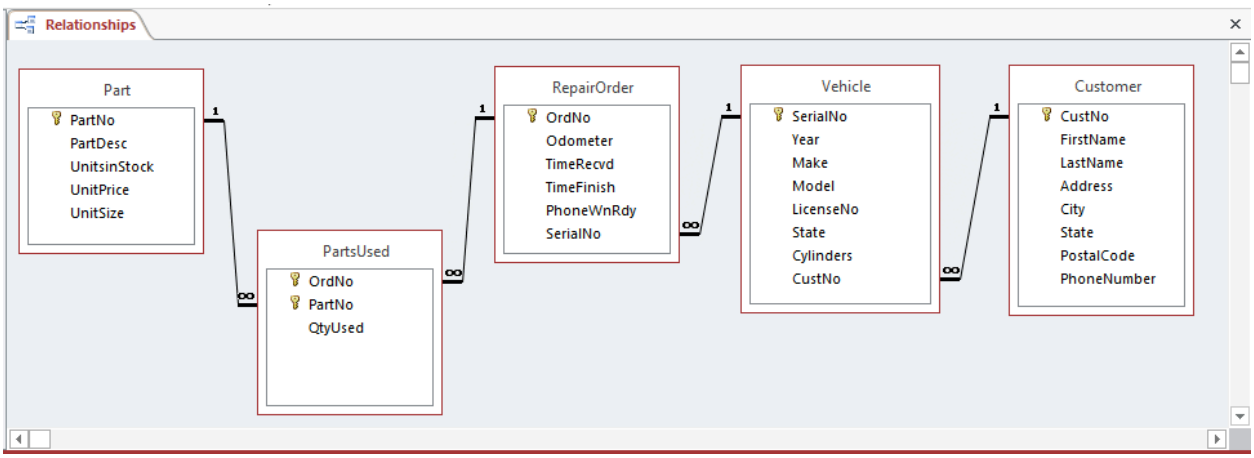


Figure 55: Completed Relationships Window

Creating 1-1 Relationships

Although 1-1 relationships are not common, you should be prepared for them. For example, there could be a 1-1 relationship between an employee table and an office table if each employee resides in at most one office and each office contains at most one employee. In 1-1 relationships, both linking fields (primary key and foreign key) must be unique. In contrast, 1-M relationships involve a primary key (with unique values by definition) and a foreign key containing duplicate values.

In Access, the Edit Relationships window supports 1-1 relationships if the foreign key is unique. A foreign key is unique if it is also the primary key or it has the *Indexed* property set to "Yes (No Duplicates)". If the foreign key is not unique, a 1-1 relationship cannot be established. In the office-employee example, if the employee table contains the foreign key (say OfficeId), the foreign must have its Indexed property set to "Yes (No Duplicates)" as the employee table would have a different field as its primary key (say EmpNo).

2.7 Table Subdatasheets

The subdatasheet feature, a feature introduced in Access 2000, is the opposite of the Lookup Wizard feature. The Lookup Wizard allows values of a foreign key to be selected from a related primary key. The foreign key is in the child table of a 1-M relationship and the primary key is in the related parent table. For example, *Vehicle.CustNo* (a foreign key) allows lookup of *Customer.CustNo* values. Subdatasheets display the opposite direction (the child direction) of 1-M relationships. For example, a subdatasheet in the *Customer* table shows the related records of the *Vehicle* table.

The plus symbol (+) indicates a subdatasheet to display records of a related table. Clicking on the + symbol expands the hidden subdatasheet. For example, clicking on the + symbol in the first *Customer* record in Figure 56 shows the related *Vehicle* records in Figure 56. Because the *Vehicle* table participates in a 1-M

relationship with the *RepairOrder* table, the *Vehicle* subdatasheet also has a subdatasheet that can be exploded.

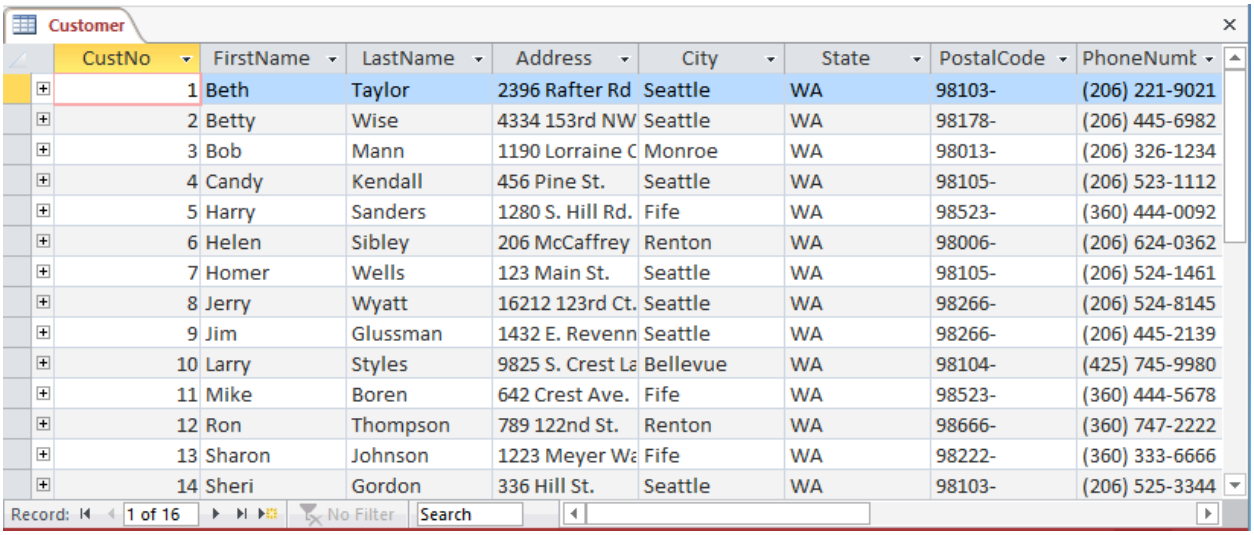


Figure 56: Customer Datasheet with a Hidden Subdatasheet

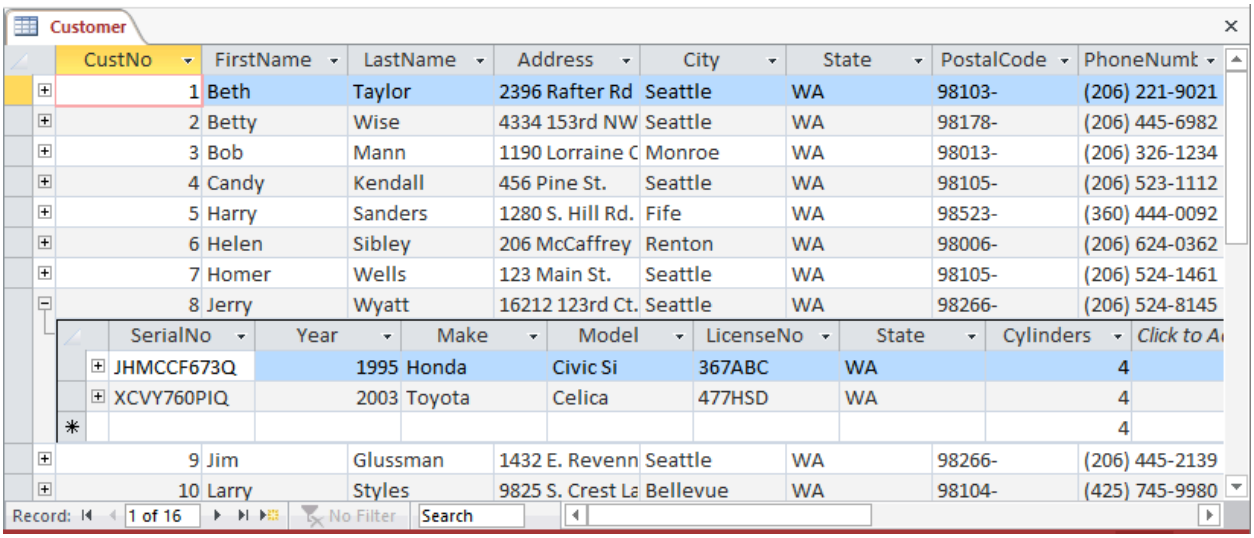


Figure 57: Expanded Subdatasheet Showing Related *Vehicle* Records

You specify a subdatasheet using the *Subdatasheet Name* table property. By default this property has a value of "Auto", as shown in Figure 58. If "Auto" is specified, a subdatasheet only appears if a 1-M relationship was established in the Relationships window.

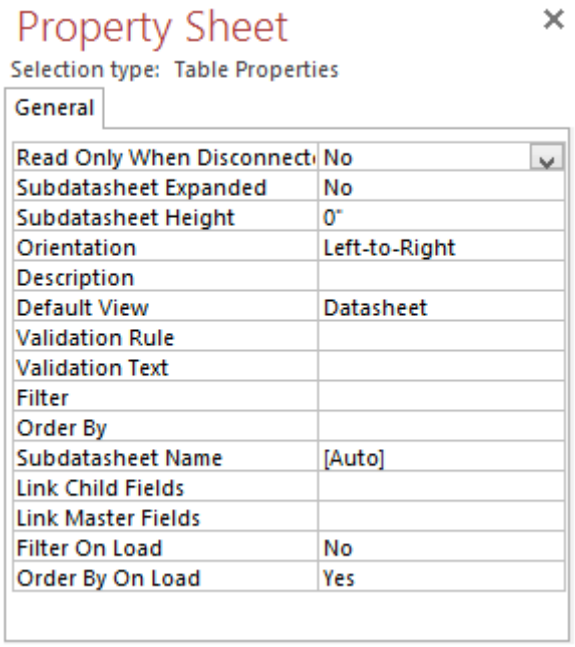


Figure 58: Table Properties Window with "Auto" Default for *Subdatasheet Name* Property

On some occasions, you may want to change the default setting for the *Subdatasheet Name* property. You can change the "Auto" setting to any table or query name in the database by selecting it from the property's drop-down menu (Figure 59). If you override the "Auto" setting, you probably will want to specify values for the *Link Child Fields* and the *Link Master Fields* properties. The *Link Master Fields* property indicates a join field in the parent table of a 1-M relationship. The *Link Child Fields* property indicates the join field in the child table of a 1-M relationship. If you do not want Access to display subdatasheets, you can change the *Subdatasheet Name* property to "None" (a choice in the drop-down menu). The "None" values eliminates the + symbols displayed on a datasheet.

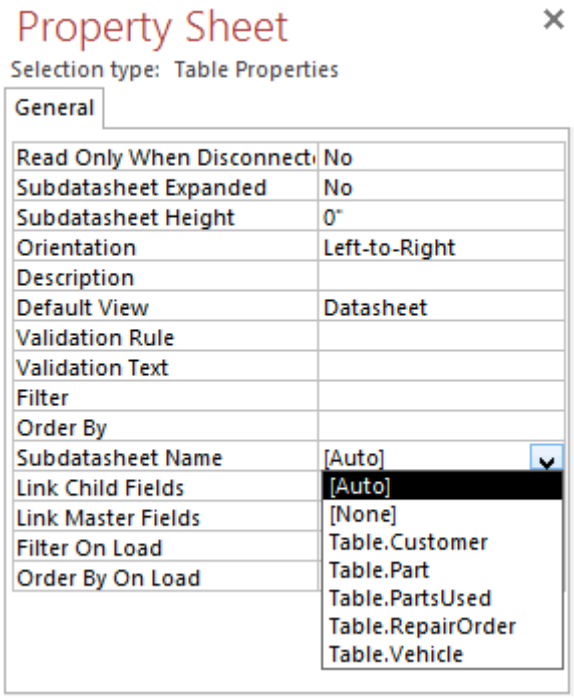


Figure 59: Table Properties Window *Subdatasheet Name* Property Choices

2.8 Database Templates, Application Parts, and Sample Fields

Table templates and field templates were new features in Access 2007. Access 2013 has expanded and changed this feature somewhat. Access 2013 supports database templates, application parts, and sample fields. These Access 2013 features are more comprehensive than table and field templates in Access 2007 and the Table Wizard in Access 2003. This brief section briefly demonstrates the database templates, application parts, and sample fields.

You can create a database from a template when you open Access. In the opening window, a list of database templates appears as shown in Figure 60. Each database template can be hosted on a Microsoft Sharepoint Server providing the ability to use the databases through Web pages. Each template database contains a complete application including tables, forms, reports, queries, macros, and relationships. You can also see the templates listed on a full display of the opening window in Figures 1 and 2.

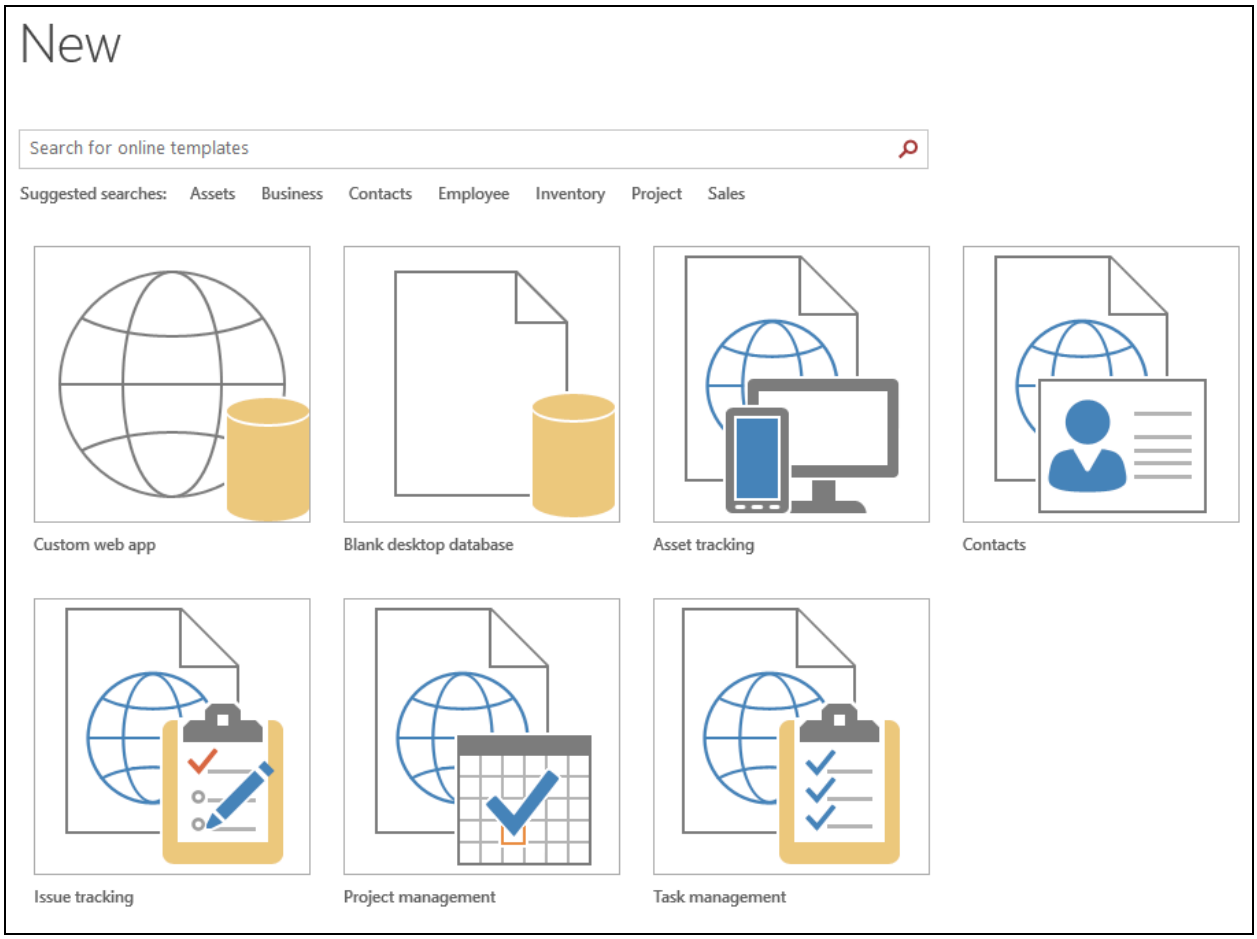


Figure 60: Database Templates in Access 2013

An application part consists of a table along with associated application objects such as forms and reports. Thus, an application part is a subset of a database template. You can create an application part using the **Create→Application Parts** item in the Templates group. As shown in Figure 61, the application parts include forms in the top part of the list and tables in the quick start part of the list. When you select a table item (such as Contacts), Access will create a table and other associated objects in the object list area.

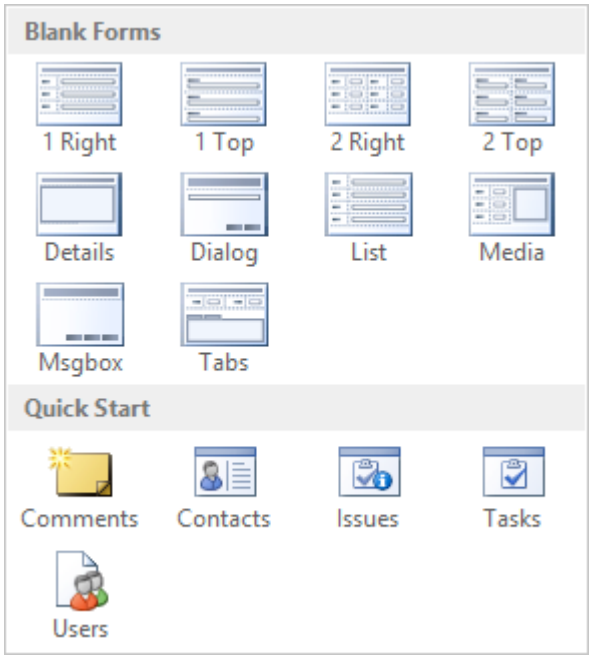


Figure 61: List of Application Parts in the Create Tab

To depict a table in an application parts, Figure 62 displays the *Contacts* table in Datasheet view. You should switch to Design view (Figure 63) to see the details of each field in the Contact template table. You will need to save the table before switching to Design view. The *Contacts* table contains a few additional fields not visible in Figure 63 because of screen size limitations. Each table in an application part has preset properties for each component field.

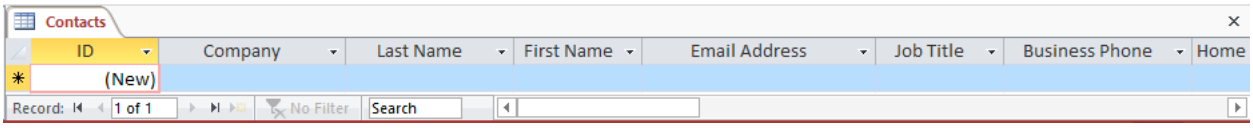


Figure 62: Datasheet View Showing the Partial *Contacts* Template Table

Sample fields are added in Datasheet view for an existing table. You can use a field template on any existing table including a template table. To depict sample fields, open the *Customer* table in Datasheet view (Figure 64). The last empty field is named "Click to Add". To add a new field using a standard data type, click on the arrow in the "Click to Add" field. You will see a list of standard data types including Text, Number, Currency, and others. For additional choices, choose **Fields → More Fields** in the Add & Delete group while the cursor is in the Click to Add area of the datasheet. You can choose from choices in the basic type, number, data and time, yes/no, and quick start categories as shown in Figure 64. To add a sample field as in a table, select the item from the list shown in Figure 65. You can then switch to Design view to edit the field properties.

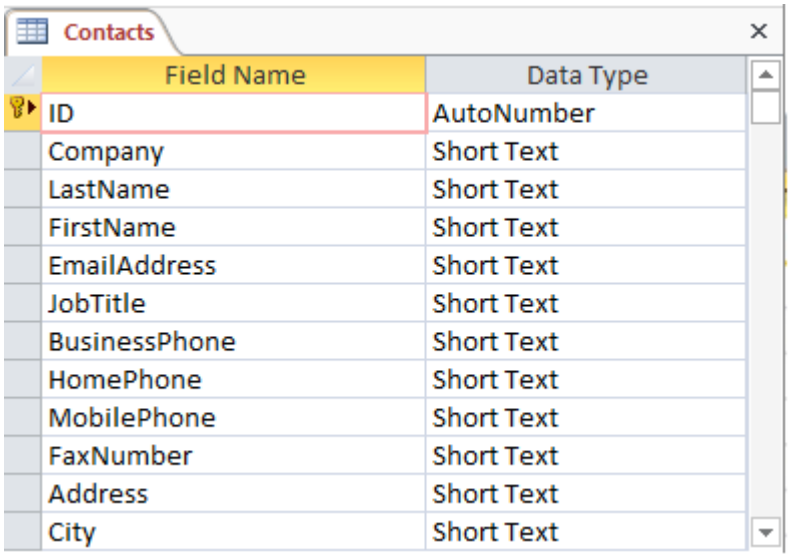


Figure 63: Design View Showing the Partial Contacts Table

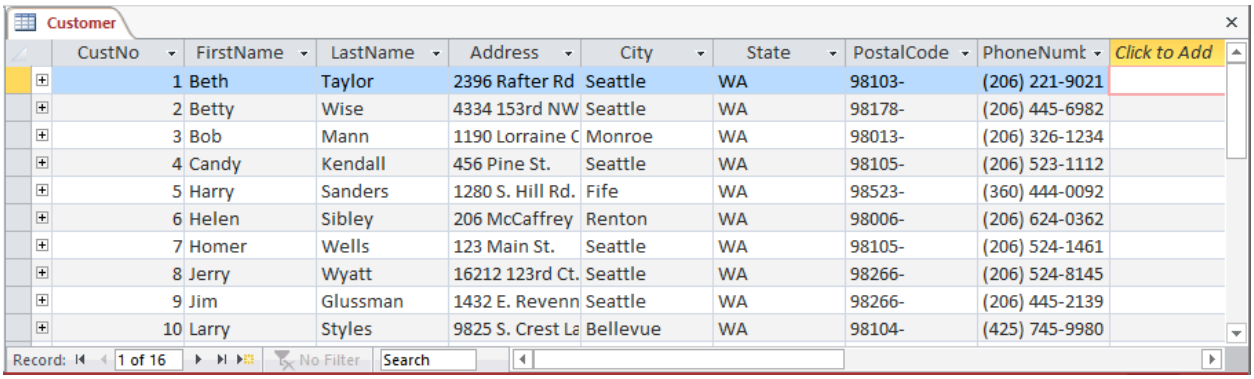


Figure 64: Customer Table in Datasheet View with Add New Field

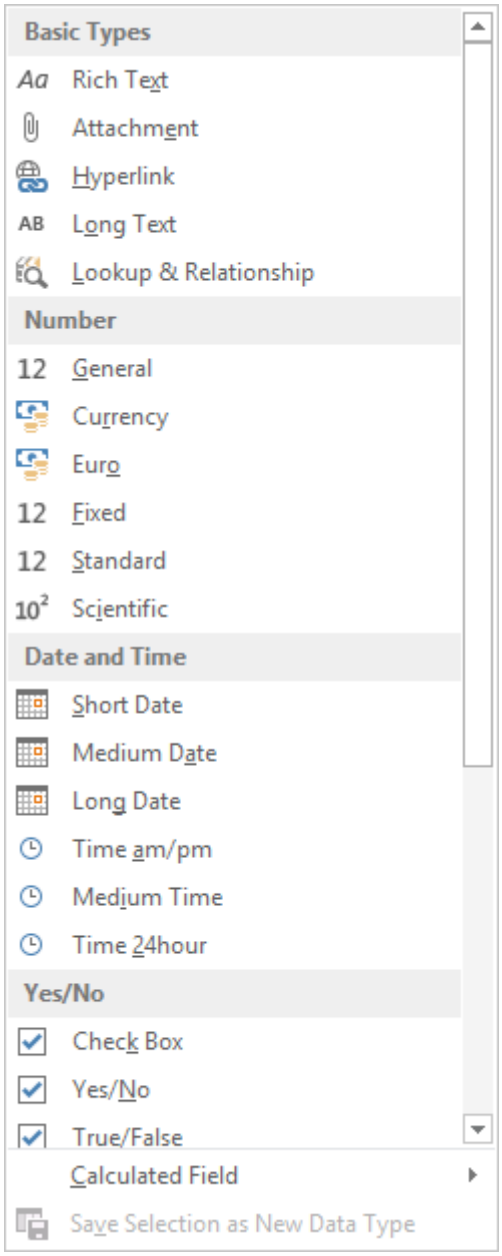


Figure 65: List of Sample Field Templates

Closing Thoughts

Chapter 2 has provided guided instruction about creating databases in Microsoft Access 2013. Access provides a number of convenient tools to create tables, properties, and relationships. These tools are much easier to use than SQL statements as discussed in textbook Chapter 3. In this chapter, you gained practice with several methods to create tables, wizards and the expression builder for defining complex properties, the Import Wizard to load data, and the Relationships window to link tables. This practice should enable you to create your own databases. The practical skills in this chapter complement the concepts in textbook Chapter 3 that emphasize understanding an existing relational database. This chapter also complements the conceptual design skills covered in Part 3 of the textbook. You can use these practical skills to implement a database that you have designed using the methods described in Part 3 of the textbook.

Chapter Reference

The chapter reference summarizes procedures that you practiced. For wizards discussed in the chapter, the procedures highlight important parts of the wizards but do not list all of the steps.

Procedure 1: Create a New Database (Section 2.1)

4. You can start the procedure in by clicking **Microsoft Office →Microsoft Access 2013 …**.to open the main Access window.
5. Provide a file name for the new database in the default location, "My Documents", or browse to select another location, then click **Create**.
6. The default table name appears in the navigation pane in the Tables group.

Procedure 2: Create a New Table in Design View (Section 2.2)

7. In the ribbon's **Create** section, click the **Table** button.
8. Select **View → Design View** from the Datasheet group of the Ribbon. Alternatively, you can click the **Table Design** button in the **Create** section to create a new table and start in Design view.
9. Name the fields, set the data types, type optional descriptions, and set field properties.
10. Designate a primary key by (a) clicking the **Key** button in the Design group of the Ribbon or (b) clicking the right mouse button and selecting the **Primary Key** command from the drop-down menu.

Procedure 3: Tips about Using the Input Mask Wizard (Section 2.2.3)

11. In the *Input Mask* property area, click the three dots (**…**) to invoke the Input Mask Wizard.
12. The Input Mask Wizard asks you to save the table first. If you have not yet named the table, type the table name in response to the next input box.
13. The initial dialog box contains the predefined input masks. Select the input mask according to the type of data contained in the field. Type an example in the Try It area below to see how your values will appear with the input mask.
14. The final window asks how to save the data: with or without the input mask symbols. It is recommended to store the data without the symbols to save memory and obtain faster processing. However, the input symbols will still appear in your tables and forms.
15. After completing the Input Mask Wizard, you can customize the input mask's appearance by changing the codes in the *Input Mask* property. You will need to study carefully the help documentation to understand the codes, though. You also should be aware that Access requires storage of literal characters in a mask for date fields. Literal storage is specified with the second argument in an input mask.

Procedure 4: Defining a Table Validation Rule (Section 2.3.2)

16. Use a table validation rule for integrity rules involving more than one field.
17. To start, you can open the Table Properties window in two ways:
    - With table Design View open, click **Design → Property Sheet**.
    - Click the right mouse button inside the table Design View window. Select **Properties** from the shortcut menu.
18. Click the ellipsis (**…**) button in the *Validation Rule* property to invoke the expression builder. Type the validation rule in the expression text area.
19. Use the *Validation Text* property to display a warning message when a user violates the validation rule.
20. Use the *Description* property to document the validation rule.

Procedure 5: Using the Lookup Wizard (Section 2.3.4)

21. Use the Lookup Wizard for foreign key fields that refer to a field in another table or fields with a small number of values.
22. In the *Data Type* property, select "Lookup Wizard" to invoke the initial dialog box of the Lookup Wizard.
23. Choose the source of the data. You may select a table or query, or type your own values. If you choose to type your own values, you will be asked later to type them into a small datasheet.
24. Choose the field from the table or query to appear in the combo box list. Select the field and click the **>** button to bring it to the right side.
25. Adjust the column width as desired.
26. The wizard prompts you to name the field and to save the table. You are then returned to Design view with the foreign key lookup field inserted.
27. Be aware that the Lookup Wizard creates a relationship if the lookup field is a primary key. When you open the Relationships window, you will see a relationship between the table containing the foreign key and the table containing the primary key.

Procedure 6: Defining a Combined Primary Key (Section 2.3.4)

28. Select the row containing the first field in the combined primary key.
29. Select rows containing the other fields in the combined primary key by using the **Shift** or **Ctrl** keys.
30. When all component fields are selected, click the **Primary Key** button in the Design group of the Ribbon or select the **Primary Key** item from the menu after you right-click.
31. Access considers all component fields as one combined primary key.

Procedure 7: Importing Data into an Existing Table (Section 2.4.2)

32. Choose **External Data → Text File** from the Import group of the Ribbon.
33. When the initial Import Wizard dialog box appears, select the file name.
34. Now you make the decision of whether the data go into a new table (this allows you to import a complete table) or into an existing table. Select "Append a copy of records to the table" and choose which table.
35. In the next window, choose "Delimited" and click the **Next** button.
36. In the next dialog box, select the following options (these are most common, but choose the options that apply to your situation): (1) "Comma" as the field separator, (2) "First Row Contains Field Names", and (3) double quotation mark (") as the text qualifier.

Procedure 8: Importing a New Table (Section 2.5.1)

37. In the ribbon's Import group, click the **External Data → Text File** button (unless it is another file type).
38. In the following Import Wizard windows, follow the same process as you did previously to import data for an existing table. However, in the window asking you where you want to store the data, select "In a New Table" instead of an existing table.
39. In the next window, choose "Delimited" and click the **Next** button.
40. In the next dialog box, select the following options (these are most common, but choose the options that apply to your situation): (1) "Comma" as the field separator, (2) "First Row Contains Field Names", and (3) double quotation mark (") as the text qualifier.
41. In the next window, you should keep all fields since you can make changes later in the Design View window.
42. In the next window, select whether you want Access to add the primary key, you want to choose your own primary key, or you want no primary key. If the table has a combined primary key, choose no primary key.
43. Finally, make sure that the table name is correct since Access uses the imported file name as the default name. Click **Finish** to import your table with data.

Note: If you make an error, do not worry. Access is very forgiving at this stage. If you are still inside the wizard, use the **Back <** key to retrace your steps. If you have finished the wizard, you can proceed to design view. If you want to try importing again, simply delete the table and import the table again.

Procedure 9: Creating Relationships (Section 2.6)

1. If you used the Lookup Wizard, Access has partially created a relationship. You only need to enforce referential integrity and set the appropriate cascade properties.
2. If you did not use the Lookup Wizard, you may need to add tables to the Relationships window before defining the relationships. To create a relationship, drag the primary key from the parent table to the foreign key in the child table. A connection line appears and the Relationships dialog box opens. You should then enforce referential integrity and set the appropriate cascade properties.
3. To create a 1-1 relationship, both linking fields must be unique. In both 1-M and 1-1 relationships, one linking field is usually a primary key. In 1-M relationships, the other linking field is a foreign that can contain duplicate values. In a 1-1 relationship, the other linking field must also be unique. A foreign key is unique if it is also the primary key or it has the Indexed property set to "Yes (No Duplicates)". If the foreign key is not unique, a 1-1 relationship cannot be established.

Additional Practice

For additional practice, you can extend the auto repair database as well as create the university and the order entry databases described in Chapters 2, 3, 9 and 10.

Part 1: Create Additional Tables for the Auto Repair Database

1. Create the *Labor* table using data in the file "labor.txt" available in the lab book's website. For your convenience, the data are shown in Table AP1. The *Labor* table contains the fields labor code (*LabCode*), labor description (*LabDesc*), hourly rate (*HourRate*), and standard time (*StdTime*), representing the standard number of hours to complete the labor.

Table AP1: *Labor* Table

L101		$20.00	0.2
L102		$20.00	0.25
L103		$20.00	0.25
L104		$20.00	0.25
L105		$22.00	0.2
L106		$25.00	1
L107		$25.00	0.5
L108		$22.00	0.4
L109		$20.00	0.1
L110		$22.00	0.75
L111		$25.00	0.5
L112		$25.00	1

2. Create the *RepairLabor* table using data in the file "repairlabor.txt" in the lab book's website. For your convenience, the data are shown in Table AP2. The *RepairLabor* table contains the fields order number (*OrdNo*), labor code (*LaborCode*), and hours worked (*Hours*). Use the Lookup Wizard to define the data types for the fields *OrdNo* (related to *Order.OrdNo*) and *LaborCode* (related to *Labor.LaborCode*).

Table AP2: *RepairLabor* Table

1	L105	0.3
1	L106	1.25
2	L101	0.25
3	L107	0.3
3	L108	0.25
3	L111	0.65
4	L112	1.2
4	L101	0.25
5	L109	0.1
6	L110	0.7
6	L101	2
7	L102	0.3
7	L103	0.25
7	L105	0.5
8	L104	0.3
9	L103	0.5
10	L101	0.3
10	L102	0.3
11	L109	0.1
11	L106	0.5
12	L112	1.5
13	L109	0.3
14	L110	1
14	L106	0.8
20	L112	0.8
20	L109	0.3
20	L110	0.5

3. Modify the relationships by enforcing referential integrity between the *Order* and the *RepairLabor* tables and the *Labor* and the *RepairLabor* tables. Set the *Cascade Update* and *Cascade Delete* properties for the relationship between *Order* and *RepairLabor*.

Part 2: Create the Textbook Databases

1. Create the university database tables as described in textbook Chapter 10. Data for the tables can be found in the text files in the textbook's website. The appendix in textbook Chapter 3 shows data for most of the tables. Textbook section 10.4.2 explains the new tables not presented in textbook Chapter 3.
2. Create the order entry database tables as described in textbook Chapter 10. Data for the tables can be found in the text files in the textbook's website. The problem section in textbook Chapter 4 shows data for most of the tables. The problem section in textbook Chapter 10 explains the new tables not presented in textbook Chapter 4.

Chapter 3: Query Formulation Lab

Learning Objectives

This chapter enables you to gain practical experience building queries utilizing the Query Design and SQL tools of Access 2013. After this chapter, you should have acquired the knowledge and skills to

- Use the Query Design window to create queries.
- Create queries in SQL view.
- Use the Query Wizard and the expression builder.

- Toggle between query views.
- Write queries to summarize data.
- Revise an existing query.
- Recognize and write parameter queries.
- Understand the Append, Update, Delete, and Union queries.
- Gain insight into the standards compliance of Access SQL

Overview

In Chapter 1, you became familiar with Access databases. You learned about the tools to create Access objects and object properties. Chapter 2 enabled you to put these concepts into practice by creating the auto repair database. To begin developing applications using the auto repair database, this chapter describes the tools available to create queries. In addition, this chapter complements the conceptual background in textbook Chapter 4 about using SQL statements to create queries.

This chapter demonstrates two tools to formulate queries in Access. The first part of this chapter gives you practice using the Query Design window. Query Design is a visual tool that allows you to create queries without writing much code. In addition to learning about various aspects of Query Design, you will practice using the Simple Query Wizard and the expression builder. The second part of this chapter demonstrates the SQL window, a text-based tool. Because SQL is the industry standard language, you should gain practice using the SQL window. In addition, some queries can be formulated more easily in SQL. You also will learn to toggle between SQL and Query Design to obtain the best of both tools. When you finish this chapter, you should be ready to create your own queries using both Query Design and SQL.

3.1 Tools to Create Queries

Access provides two ways to formulate queries. Query Design provides a visual way to formulate queries. The SQL window allows you to write SQL statements using the syntax described in textbook Chapter 4. This section uses a simple query to demonstrate the basics of both Query Design and the SQL window.

When you want to create a new query, you must start in Query Design. You can formulate the query by selecting tables and fields, joining tables, and specifying conditions to restrict query results. You then may switch to SQL view to modify the SQL statement for the query created in Query Design.

> Technical Note: Alternatively, you can switch immediately to SQL view to write the entire query. This section introduces you to both tools of query formulation and toggling between the tools.

3.1.1 Creating a Query in Query Design

The first query is a simple one that involves only one table, the *Customer* table. The purpose of this query is to make a phone list to remind existing customers of their vehicle's next service. Therefore, we are going to ask the query to give us a list of customers' first name, last name, and phone number:

1. Open the Auto Repair Database: Start Access and open the auto repair database (AutoRepair.accdb) that you created in Chapter 2. The navigation pane should show the tables that you created in Chapter 2 (Figure 1). Choose **Create → Query Design** in the Queries group to open Query Design in the view pane (Figure 2).

2. Select a Table/Query: Query Design appears in the view pane with the Show Table window on top of it (Figure 2). You must select a table or existing query to include in the new query. Select the *Customer* table, click the **Add** button, and then click the **Close** button. Figure 3 shows the Query Design pane containing the *Customer* table.

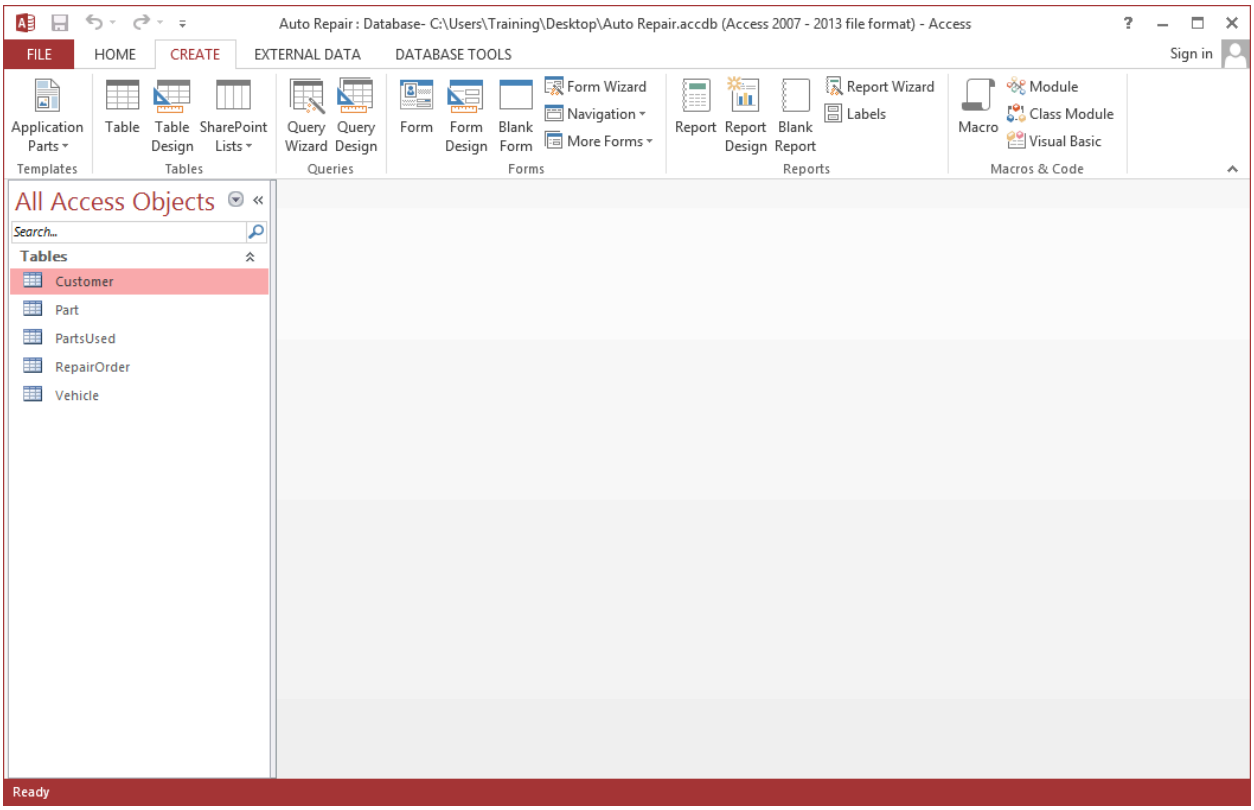


Figure 1: Navigation Pane with Empty View Pane

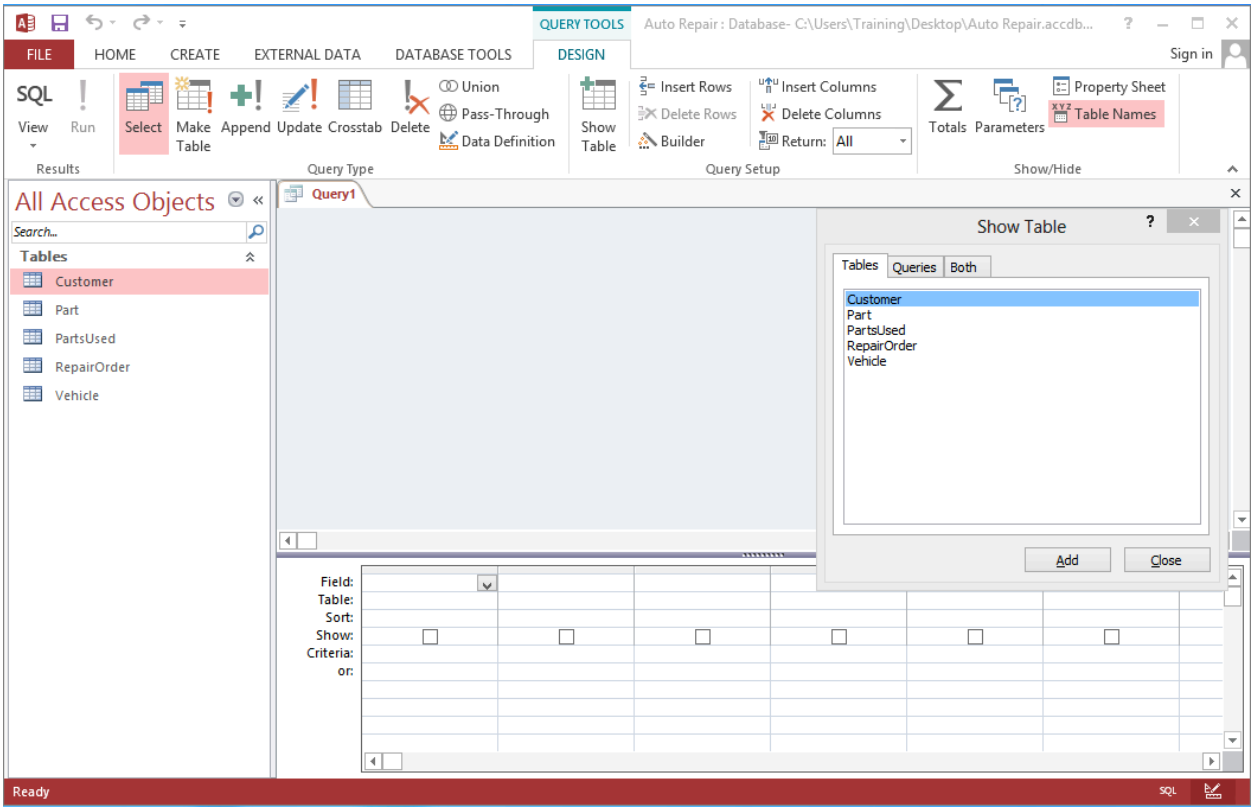


Figure 2: Query Design Pane and Show Table Window

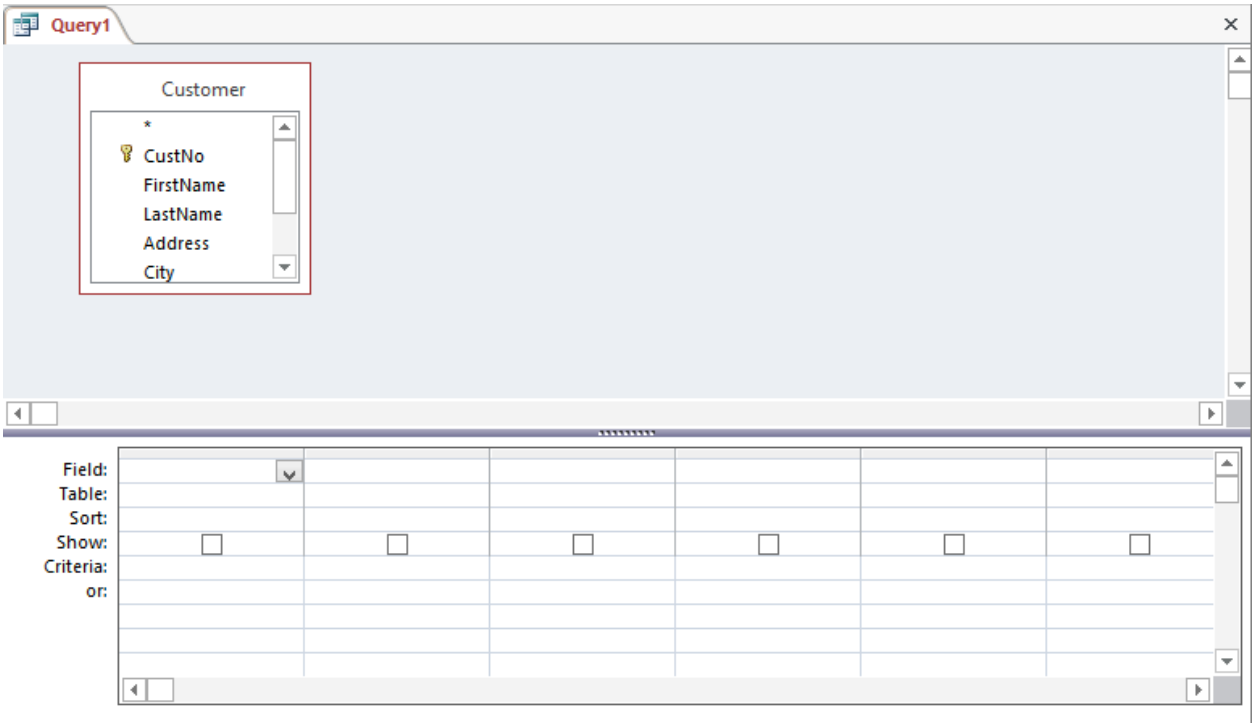


Figure 3: Query Design Area with *Customer* Table

4. Drag Fields into the New Query: To obtain query results, you must put some *Customer* field names into the empty field spaces of the query table. Drag and drop[2] the *CustNo* field name from the *Customer* table to the first column and row in the empty query table. After releasing the mouse button, the field name appears in the grid. Repeat the process to place the *FirstName* field in the next empty column of the first row (see Figure 4).

5. Use a Different Field Selection Technique: To place the next field, *LastName*, use an alternative method. This time, just click into the next empty column of the first row and a small gray arrow appears (see Figure 4). Click the arrow and a list of the *Customer* fields appears. Select *LastName* from the list.

6. Select the Last Field: Select the *PhoneNo* field using the technique demonstrated in step 5. Note that you have to move the small scroll bar down to reveal the *PhoneNo* field. After you have finished, your Query Design window should appear as in Figure 5.

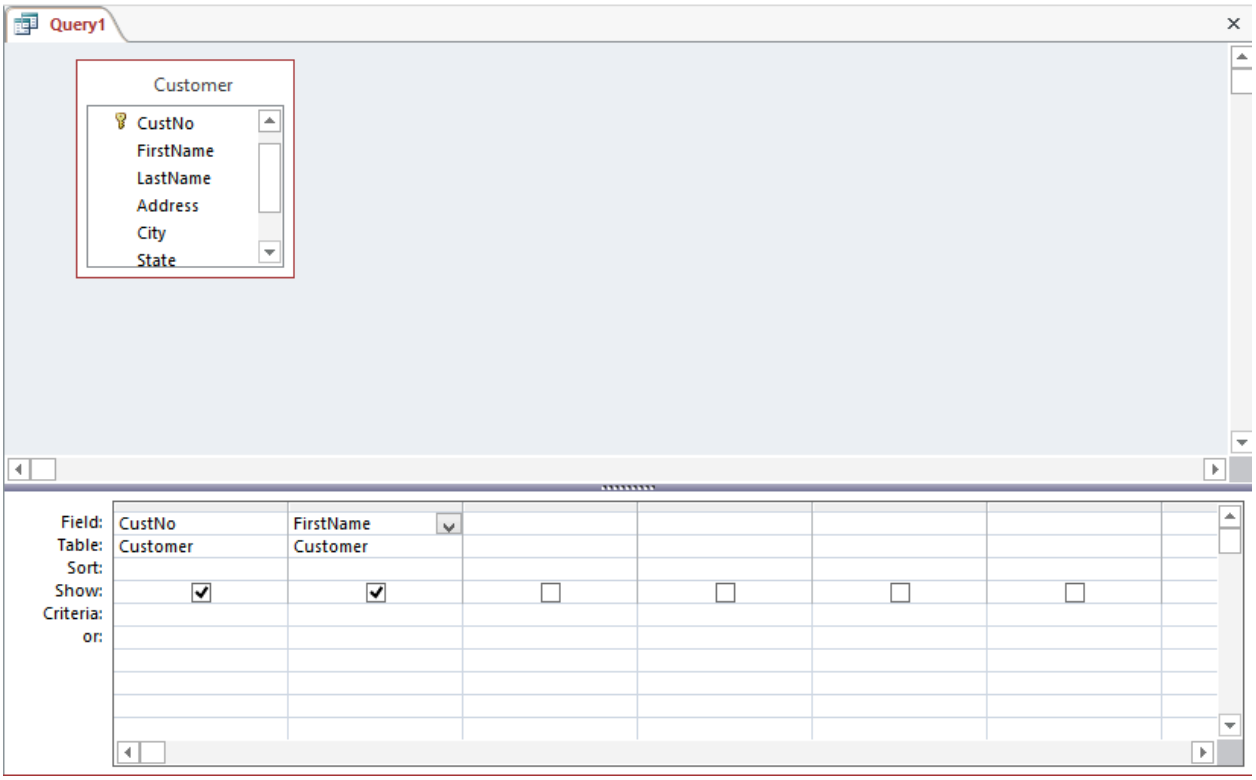


Figure 4: First Two Fields in Query Design View

[2] Drag and drop means that you hold the mouse down while moving it to another part of the window. After the item has been moved, you release the mouse button.

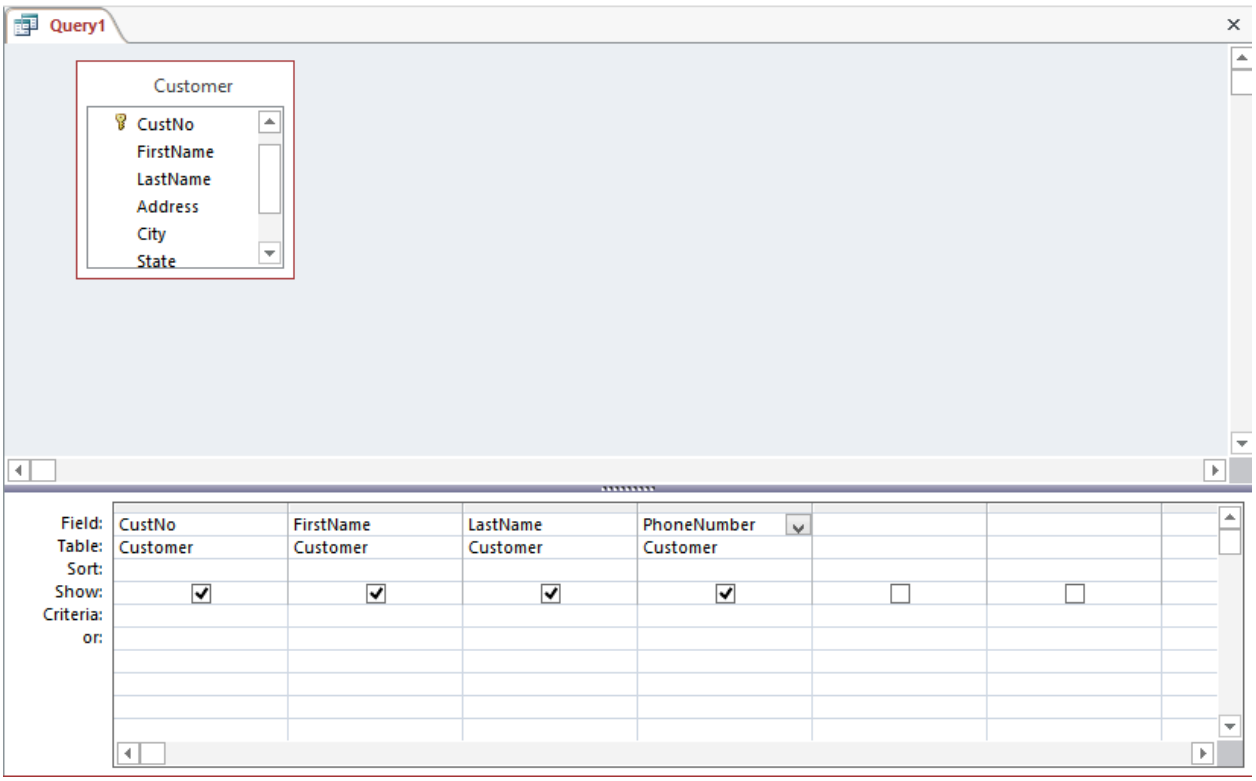


Figure 5: Completed Query Design View

3.1.2 SQL View

In the second part of this chapter you will create queries directly in the SQL view. At this point, it is important to know that queries created in Query Design may also be viewed in the SQL view. To demonstrate switching between query views, the following steps demonstrate ways to access the SQL window as well as to allow you to save your query.

1. Accessing the SQL Window: With the Query Design view open, click **View → SQL View** (see Figure 6) in the **Design** tab of the Ribbon. The SQL view appears (Figure 7) in the view pane.

> Technical Note: You also may right-click the mouse in the blue Query Design window pane. Then select **SQL View** from the shortcut menu that appears.

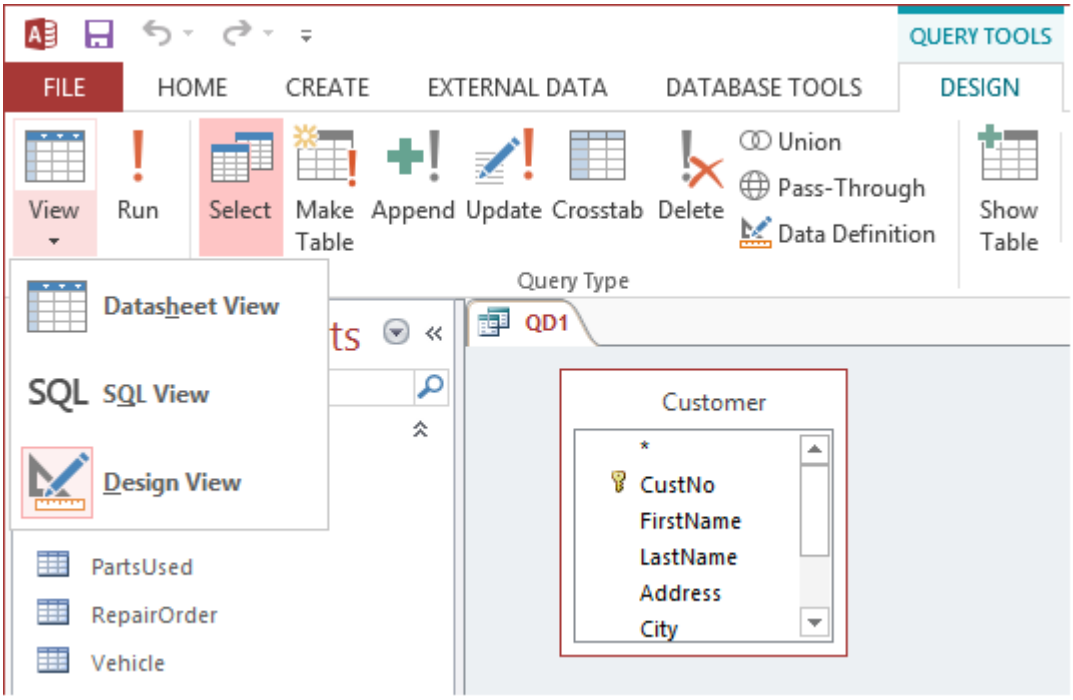


Figure 6: View Menu Showing the SQL View Command

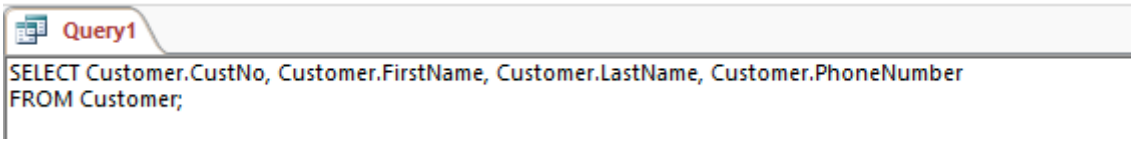


Figure 7: SQL View for "QD1"

2. Close and Save the Query: Close the SQL View pane by clicking the **Close** button (**X** button at the top of the window). Click **Yes** to save the query.

3. Name the Query: Type "QD1" as the name of the query in the next dialog box. Note that "QD" stands for "Query Design". Click the **OK** button to finish. Your first query now appears in the Queries section of your navigation pane.

3.1.3 Execute the Query

As you know, the purpose of a query is to display the answer to a question required by a database user. When running or executing a query, the results are displayed as a datasheet. Now that the "QD1" query is completed, you can run it to see if you obtain the expected results. A query may be executed from either the SQL or the Query Design view:

1. Open the Query Design View or the SQL View: If you are using the standard All Tables view, then the query appears under the table that it uses. You can launch the query at any time by double-clicking it. To open the query in Query Design view, right-click it in the navigation pane, and then choose **Design View**.

2. Execute/Run the Query: Next, choose **Query Tools Design → Run** from the Results group in the Ribbon (Figure 8).

3. View the Result: The datasheet appears with the phone list of the auto repair shop customers (Figure 9). Close the datasheet to return to the navigation pane.

In addition to executing the query per the previous steps, the datasheet also may be accessed by toggling to the datasheet view. Refer to the next section to learn how to toggle between query views.

Figure 8: Ribbon Showing the Results Group with the **!Run** Item

QD1

CustNo	FirstName	LastName	PhoneNumber
1	Beth	Taylor	(206) 221-9021
2	Betty	Wise	(206) 445-6982
3	Bob	Mann	(206) 326-1234
4	Candy	Kendall	(206) 523-1112
5	Harry	Sanders	(360) 444-0092
6	Helen	Sibley	(206) 624-0362
7	Homer	Wells	(206) 524-1461
8	Jerry	Wyatt	(206) 524-8145
9	Jim	Glussman	(206) 445-2139
10	Larry	Styles	(425) 745-9980
11	Mike	Boren	(360) 444-5678
12	Ron	Thompson	(360) 747-2222
13	Sharon	Johnson	(360) 333-6666
14	Sheri	Gordon	(206) 525-3344
15	Todd	Hayes	(206) 775-7689
16	Wally	Jones	(206) 523-9957
*	(New)		

Record: 1 of 16 No Filter Search

Figure 9: "QD1" Datasheet

3.1.4 Toggling between Views

You have just learned about the three query views: design view, SQL view, and datasheet view. For convenience, Access has a number of ways to navigate between these views. You can navigate among views whenever a query is open in the view pane. The different ways to switch between the three views are explained in the following steps:

1. Use the query tools: With Query Design view open, you will see Query Tools in the ribbon. Choose **Query Tools | Design → View** in the Results group (Figure 6) and the three view choices appear.
2. View Bar: When a query view appears in the view pane, the view bar 

appears in the lower right corner underneath the view pane. You can select a button in the view bar to change views. You can select design view (), datasheet view (), or SQL view (

).

Technical Note: You also can switch to other views by clicking on the right mouse button (right-click) when the mouse is over the blue window frame in any view. A shortcut menu appears showing the remaining two view choices.

3.1.5 Using the Simple Query Wizard

Another useful tool for creating queries is the Simple Query Wizard. The Simple Query Wizard guides you using a sequence of windows to create a query. Follow the steps below to use the wizard to create another phone list. You will create a similar query as before, except that you will use the Query Wizard and add a condition to limit the query result to customers in Seattle.

1. Open the New Query Window: Choose **Create → Query Wizard** in the Queries group to open the New Query window (Figure 10). Click the **OK** button with the Simple Query Wizard highlighted.
2. Select a Table/Query: In the initial window (Figure 11), you first need to select from the Tables/Queries list. The Simple Query Wizard uses the selected table/query as a starting point in your new query. The default is always the previous query. Since you are basing this query on the *Customer* table as in the previous query, select *Customer*. Notice that when you make a selection, the list of available fields in the window below changes accordingly.
3. Select Fields to Include: Click the **>** button to move the fields shown in Figure 12 to the right. Select the fields shown in Figure 12 and click **Next**.
4. Finish the Query Wizard: In the final wizard window (Figure 13), name the query "QD2". Below you are asked if you want to open the query or modify its design. Choose the second option, "Modify the query design.", and click **Finish.** The design view of "QD2" (Figure 14) appears ready for you to modify.

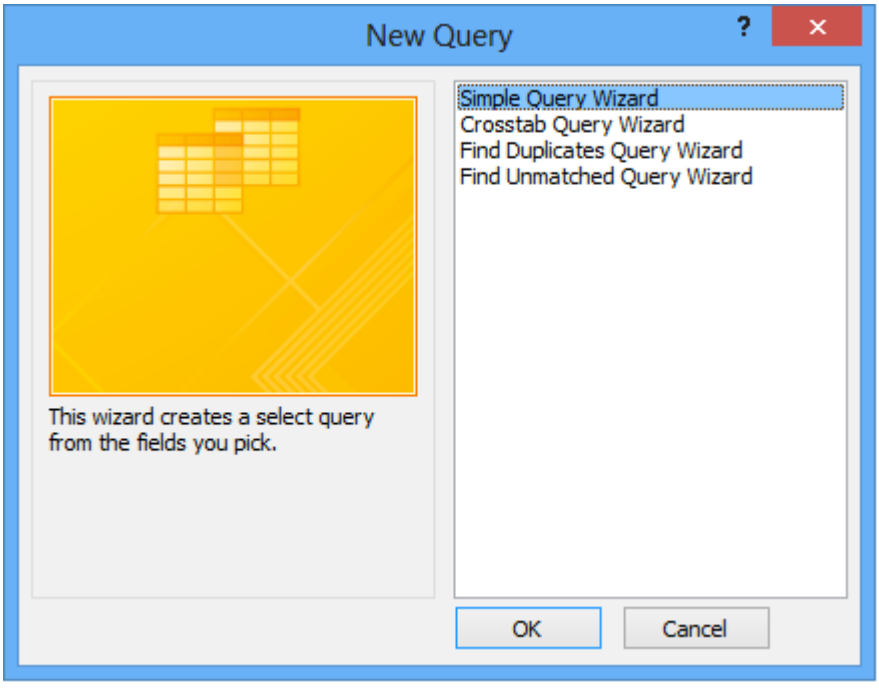


Figure 10: New Query Window of the Simple Query Wizard

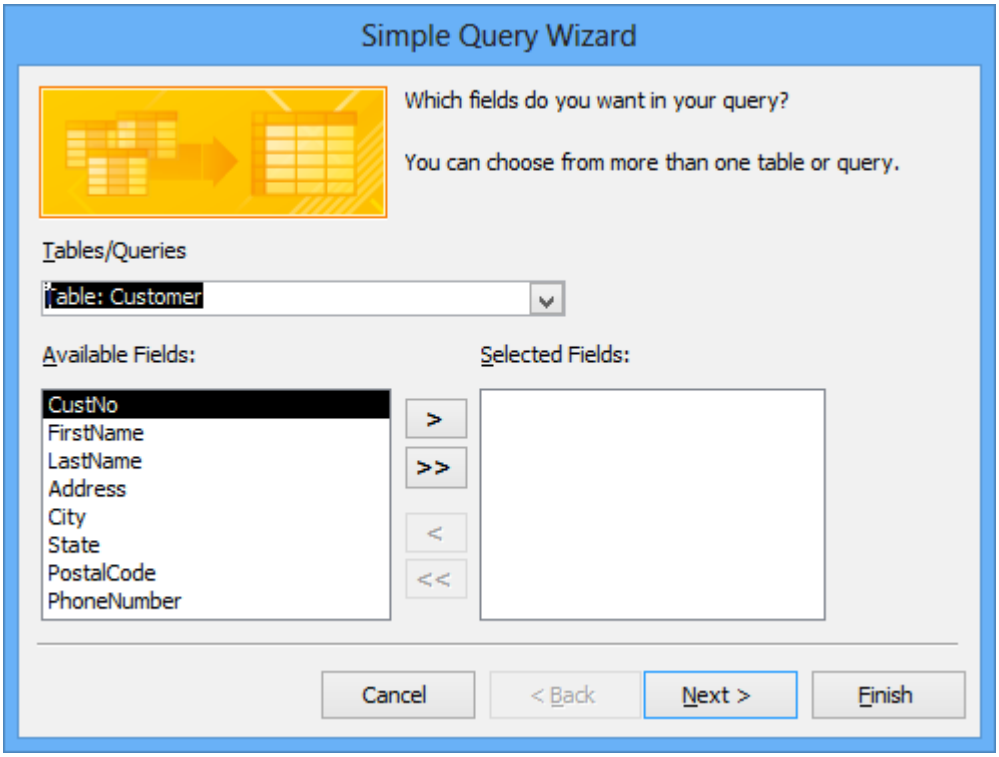


Figure 11: Initial Window of the Simple Query Wizard

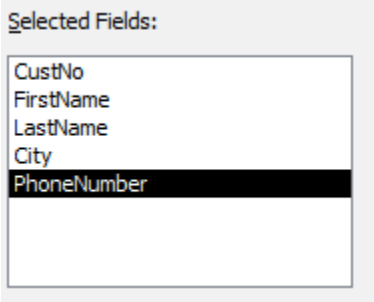


Figure 12: Fields Selected for the New Query

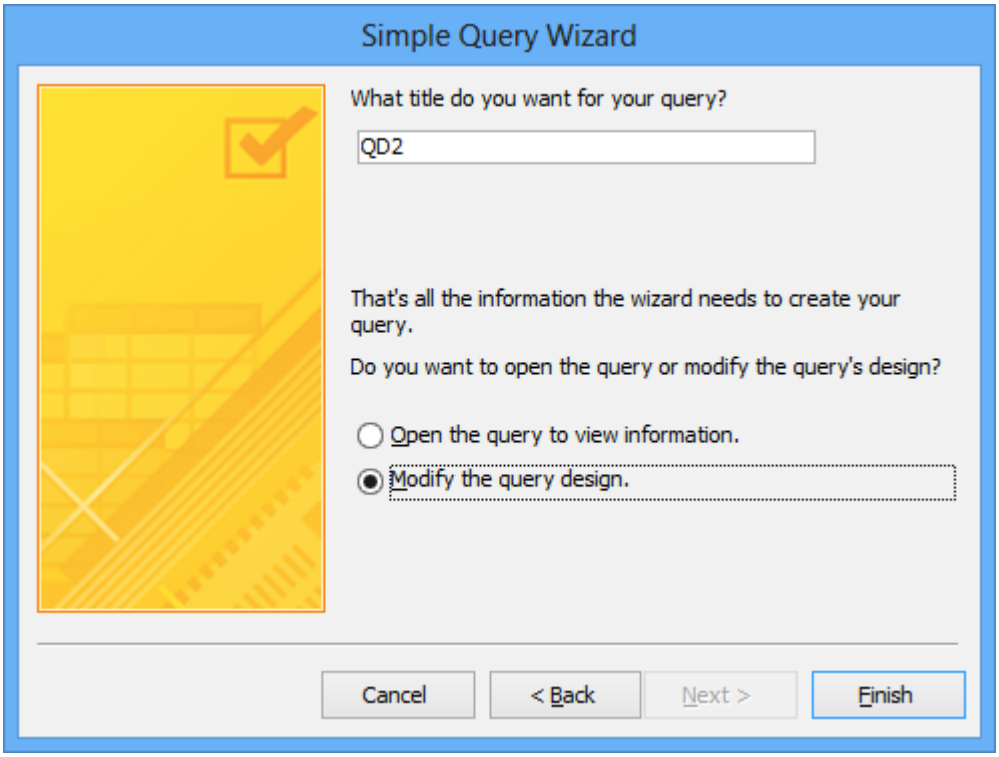


Figure 13: Final Wizard Window

5. Enter a Condition: Next you will insert a condition to limit the list to only customers in Seattle. To accomplish this task, type in the *City* column and the *Criteria* row "Seattle". Be sure to enclose it in double quotes (see Figure 15).

6. Execute the Query: Now execute the query by clicking on the **!Run** button on the ribbon. The datasheet appears as in Figure 16. Notice that the datasheet shows an asterisk (*) in the last row. The asterisk means that the query is "updatable". For now, you can ignore the last row if it contains an asterisk. Updatability is an important concept for data entry forms. See textbook Chapter 10 for view updatability concepts and rules. Close the datasheet when you are finished viewing it.

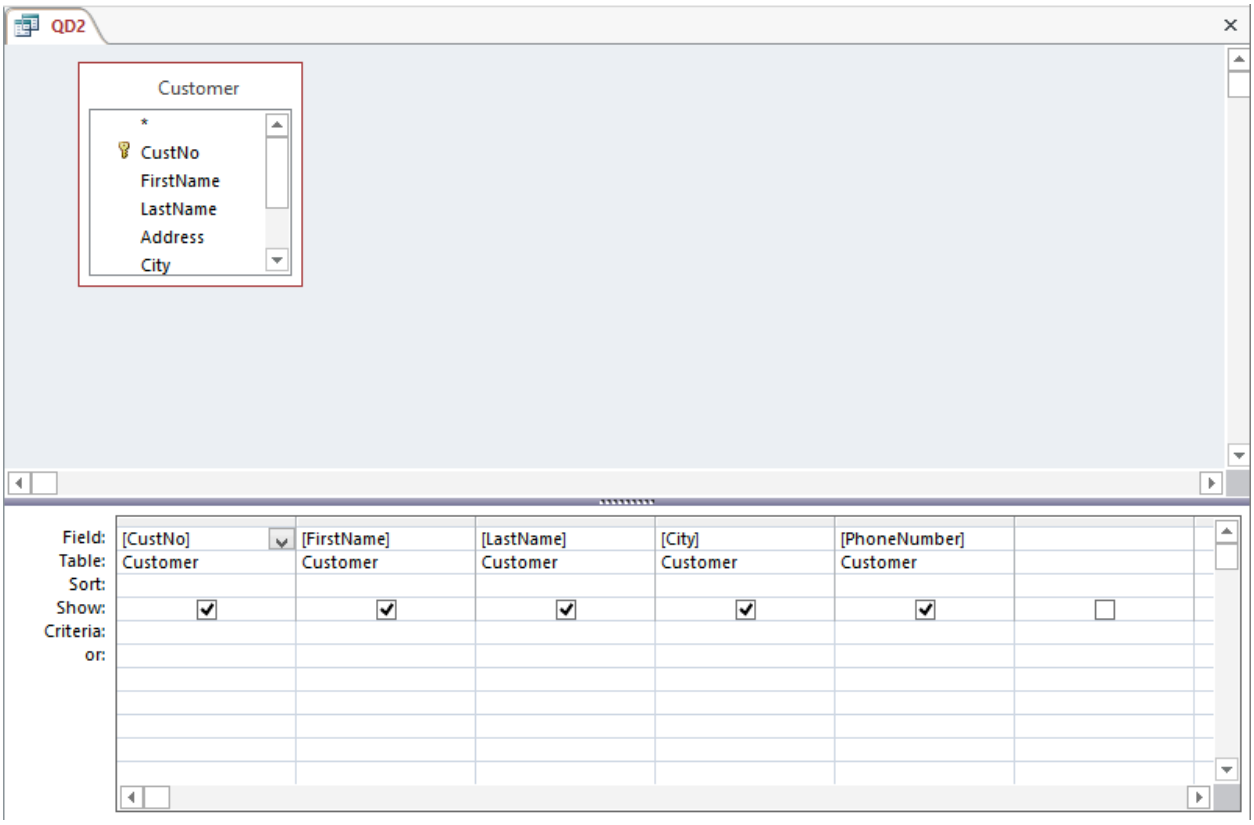


Figure 14: Query Design Window of “QD2”

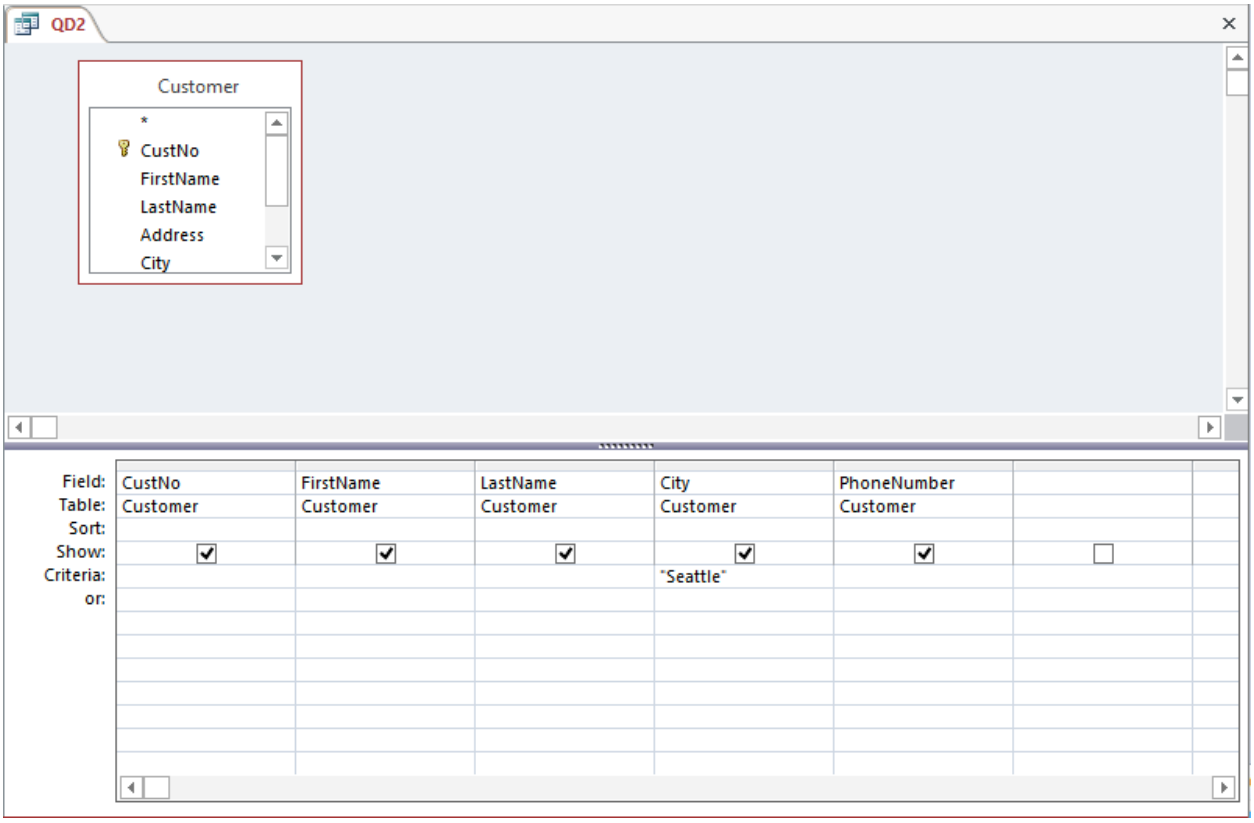


Figure 15: "QD2" with the "Seattle" Criterion

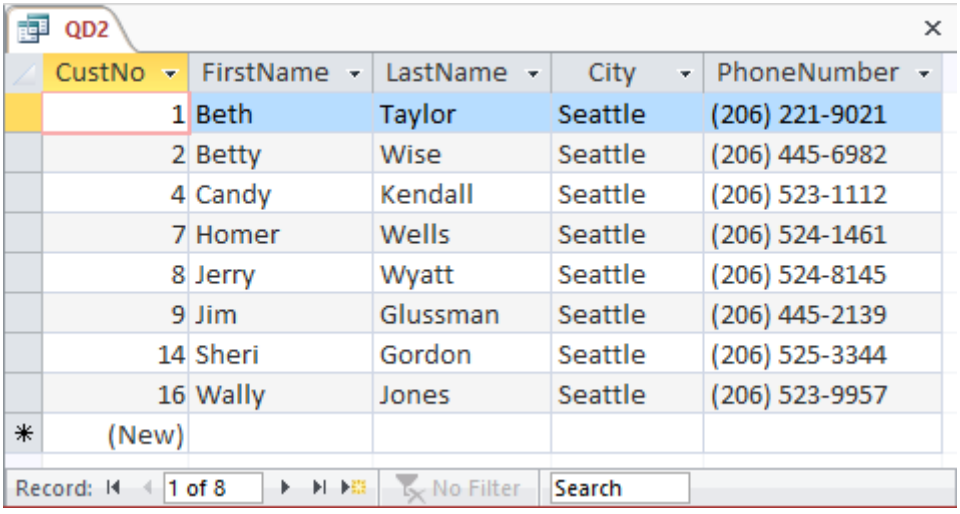


Figure 16: Datasheet of "QD2" with the "Seattle" Criterion

3.2 Creating Additional Queries in Design View

In the previous section you created a simple query in design view and saw how it appeared in SQL view and datasheet view. In this section you will gain additional practice with design view using various tools to help formulate more advanced queries. Remember that you still may toggle between design view and SQL view to examine your query.

3.2.1 Query with More than One Condition

The last query you created ("QD2") had a single condition to limit the result of the query to only customers residing in Seattle. For this new query, your list also will include customers who reside in the city of Renton. To accomplish this, you will copy and paste the query "QD2" and add a second condition to the query. The revised query will allow the phone list to include customers residing in the cities Seattle and Renton.

1. Copy and Paste an Existing Query: In the navigation pane, select the "QD2" query. From the ribbon (or point the mouse on the highlighted "QD2" and right-click), select **Copy.** Then again from the ribbon select **Paste** (or by pointing the mouse under the highlighted "QD2" and right-clicking, select **Paste**). Type "QD3" as the name of the new query when prompted.
2. Open the Query Design Window: Right-click the query in the navigation pane. The design view of "QD3" appears on the screen containing the copied "QD2" query ready for you to change (refer back to Figure 14).
3. Enter a Condition: Type the city name of "Renton" directly under "Seattle" in the *City* column (Figure 17). Adding a criterion in another row indicates an OR connection among the conditions. Thus, the query includes customers from either the city of Seattle or the city of Renton.
4. Execute the Query: Now, execute the query to be sure it is correct (Figure 18). After viewing it, close the Datasheet window and save changes when prompted.

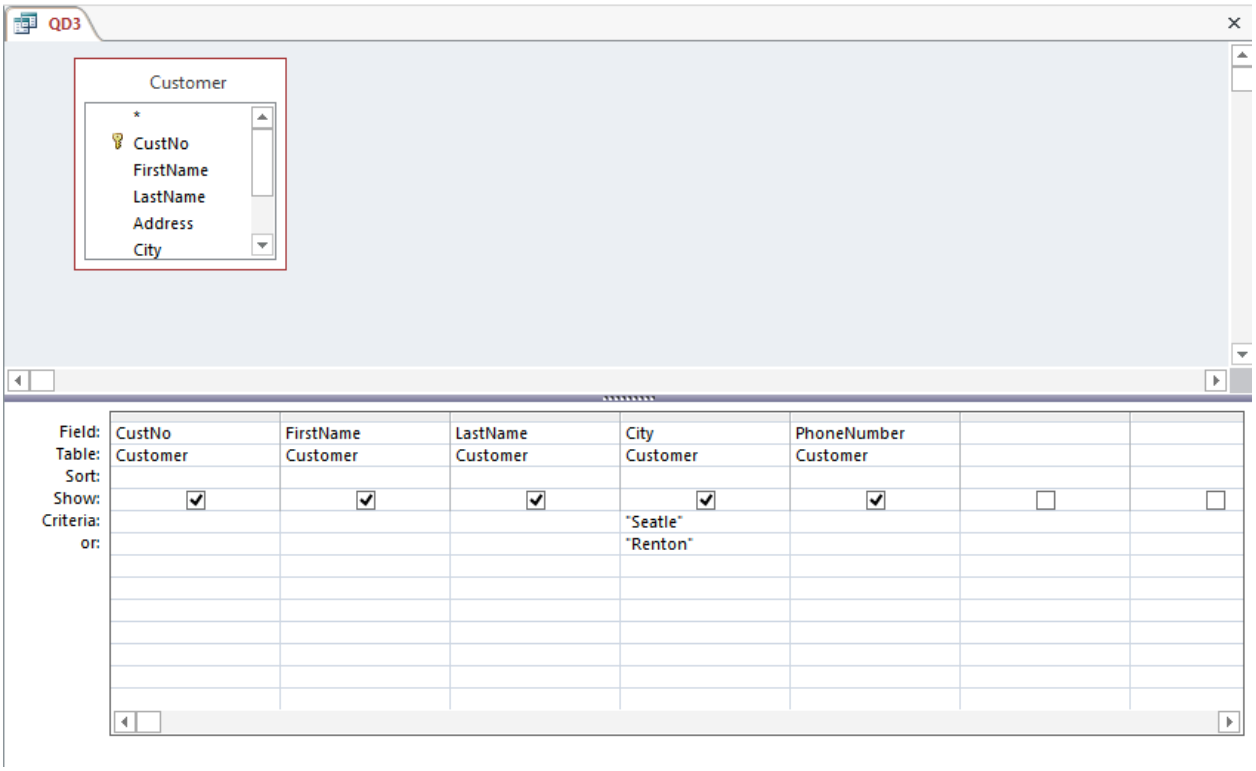


Figure 17: "QD3" with the "Seattle" OR "Renton" Criteria

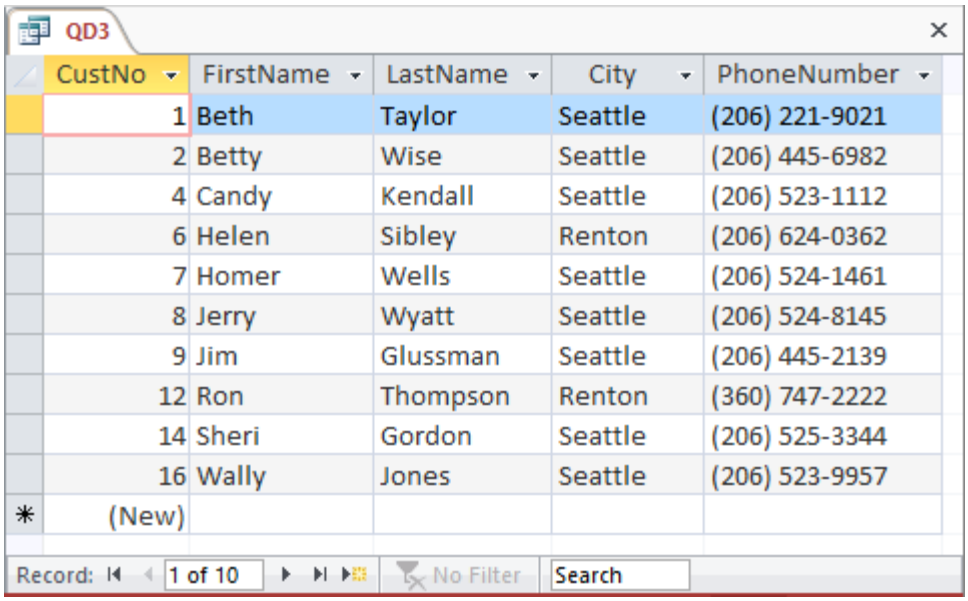


Figure 18: Datasheet of "QD3" with the "Seattle" OR "Renton" Criteria

3.2.2 Using the Expression Builder with Query Design

The next query uses the *Part* table. The auto repair shop needs a parts list to access the impact of price changes. The *Part* table that you created in Chapter 2 contains the current price, not an inflated price. To compute an inflated price, you will use the expression builder to type an expression. In addition, you will set the *Format* property to make the field display as a monetary value.

1. Create a New Query: Choose **Create → Query Design** from the Queries group to open the New Query window.
2. Select the *Part* Table: When the Show Table window appears, select the *Part* table, click **Add,** and then click **Close**.
3. Select Fields: In the Query Design window, drag and drop all five fields from the *Part* table to fill the query table (see Figure 19).

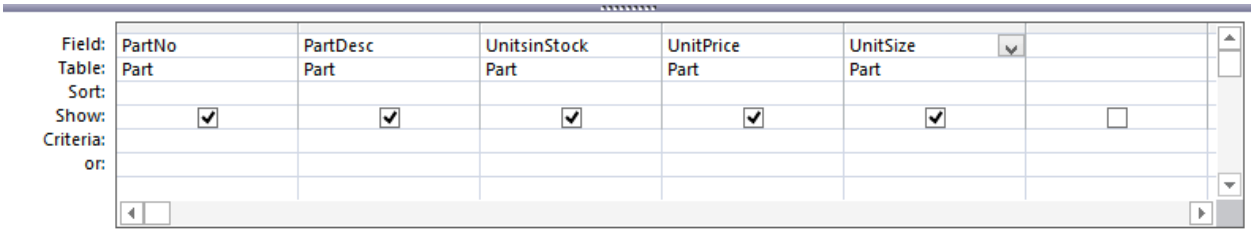


Figure 19: Lower Pane of the Completed Query Design View

4. Using the Expression Builder: Now, you need to change the name of the *UnitPrice* field to *InflatedPrice* and type an expression to inflate the price by 10 percent. You will access the expression builder to accomplish these tasks. However, for the expression builder to read the fields in the query, the query must first be named and saved. So, on the toolbar, click **Save**. When asked to name the query, type "QD4" and click **OK**.
5. Open the Expression Builder: Next, position the mouse in the *UnitPrice* field and click the right mouse button to reveal a shortcut menu. Click on **Build…** and the Expression Builder window appears with the field name *UnitPrice* in the text area.

6. Type into the Text Area: Type the expression as shown in the text area in Figure 20:

InflatedPrice: [UnitPrice]*1.1

When you are finished, click **OK**. Your *UnitPrice* field in design view should now appear as Figure 21. If you scroll in the cell, you can see the entire expression that you typed.

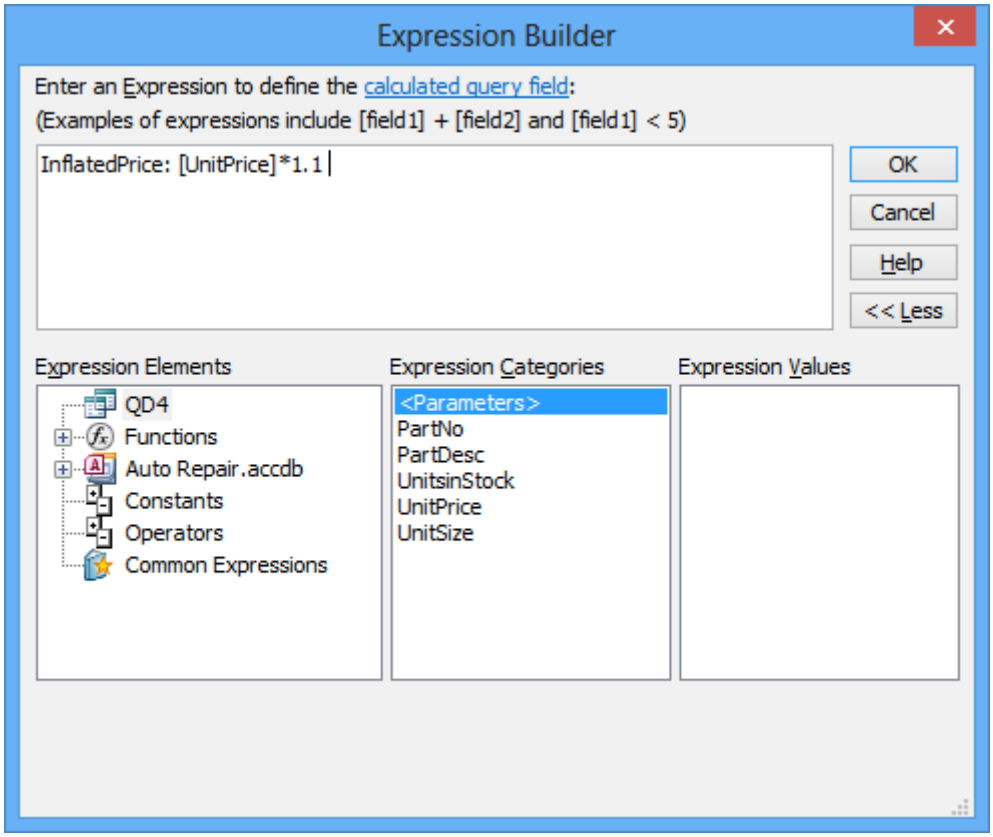


Figure 20: Expression Builder for "QD4"

7. Open the Field Properties Window: Next, you will set a query field property. To open the property sheet for a query field, position the mouse in the *UnitPrice* field and click **Query Tools | Design → Property Sheet** from the Show/Hide group. The Property Sheet for Field Properties appears on your screen (Figure 22).

> Technical Note: Alternatively, you can position the mouse in the *InflatedPrice* field and right-click to reveal a shortcut menu and select **Properties**.

8. Set the *Format* Property: Click the mouse into the text area of the *Format* property to reveal a small arrow. Click the arrow to reveal a list of properties. Scroll down and select the "Currency" value. Click the **Close** button to close the Properties window. Set the Format property for the InflatedPrice file (computed field) to currency just like UnitPrice field.

9. Execute the Query: Execute the query to be sure it is correct. Your datasheet should appear as in Figure 23. When you are finished viewing it, close it and save the changes when prompted.

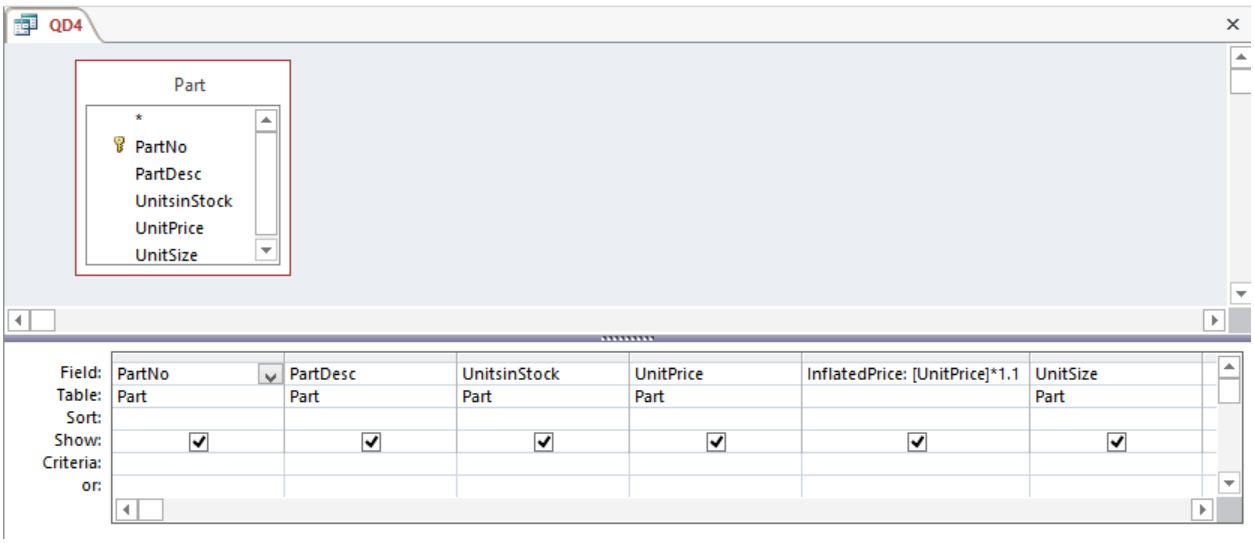


Figure 21: Completed "QD4"

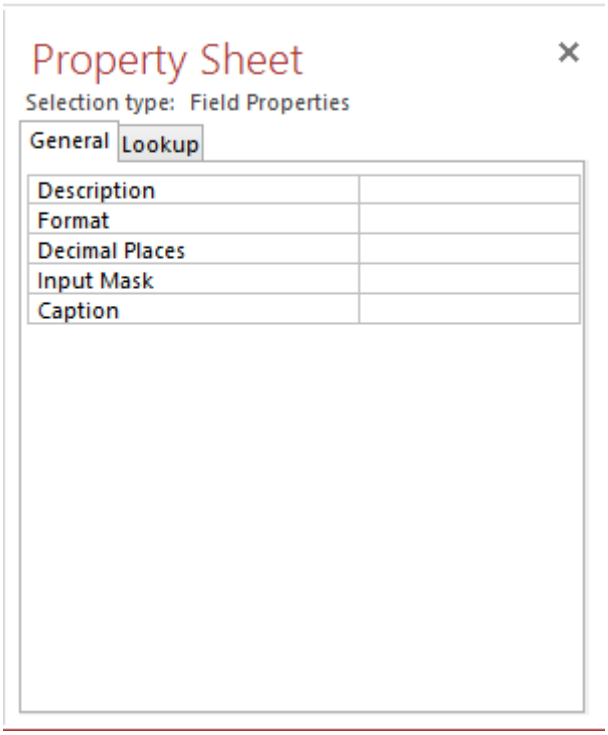


Figure 22: Field Property Sheet Window

QD4

PartNo	PartDesc	UnitsinStock	UnitPrice	InflatedPrice	UnitSize
1	10W-40 oil	145	$1.00	$1.10	quart
2	oil filter	14	$2.00	$2.20	item
3	AntiFreeze	10	$3.95	$4.35	quart
4	Spark Plugs	45	$0.99	$1.09	item
5	Transmission Fluid	15	$1.50	$1.65	quart
6	10W-30 oil	95	$0.95	$1.05	quart
7	Brake Lining	25	$5.00	$5.50	quart
8	Shock	75	$6.00	$6.60	item
9	Muffler	10	$15.00	$16.50	item
10	Tail Pipe	85	$4.00	$4.40	item
11	Head Gasket	32	$9.00	$9.90	dozen
12	Timing Chain	6	$22.50	$24.75	item
13	Battery	13	$55.00	$60.50	item
14	Radiator	3	$60.00	$66.00	item
15	Radiator Hose	24	$3.00	$3.30	dozen
16	Rotor	4	$16.00	$17.60	item
17	Tire	36	$46.00	$50.60	item
18	Headlight	6	$5.00	$5.50	case
19	Tail Light	7	$3.00	$3.30	dozen
20	GearBox	2	$25.00	$27.50	item
(New)					

Record: 1 of 20 No Filter Search

Figure 23: Datasheet of "QD4"

3.2.3 Connecting Conditions by AND

To make the parts list more useful, it should be restricted to frequently used parts. Since the auto repair shop performs many oil changes, restricting the list to the oil inventory is appropriate. In addition, the list should only display an item if the quantity in stock is greater than 10 so the inventory will not be depleted.

1. Copy and Paste an Exisitng Query: To begin, copy "QD4" and paste it as "QD5".
2. Open the Query Design Window: Next, open "QD5" in design view. It should appear on your screen the same as "QD4" (refer back to Figure 21).
3. Open the Expression Builder: You will use the expression builder to set the criteria for the parts list. Therefore, click in the *Criteria* row of the *PartDesc* field column to open the Expression Builder window.
4. Type into the Text Area: Type the expression as shown in the text area of Figure 24:

   ```
   LIKE "*oil*"
   ```

   When you are finished, click **OK**.
5. Set Criteria: Set the criteria for the number of units in stock. Click in the *Criteria* row of the *UnitsInStock* field and type the following as in Figure 25: `>10`

6. Execute the Query: When you are finished, execute the query to see if it is correct, then close and save changes when prompted. The datasheet should appear as in Figure 26.

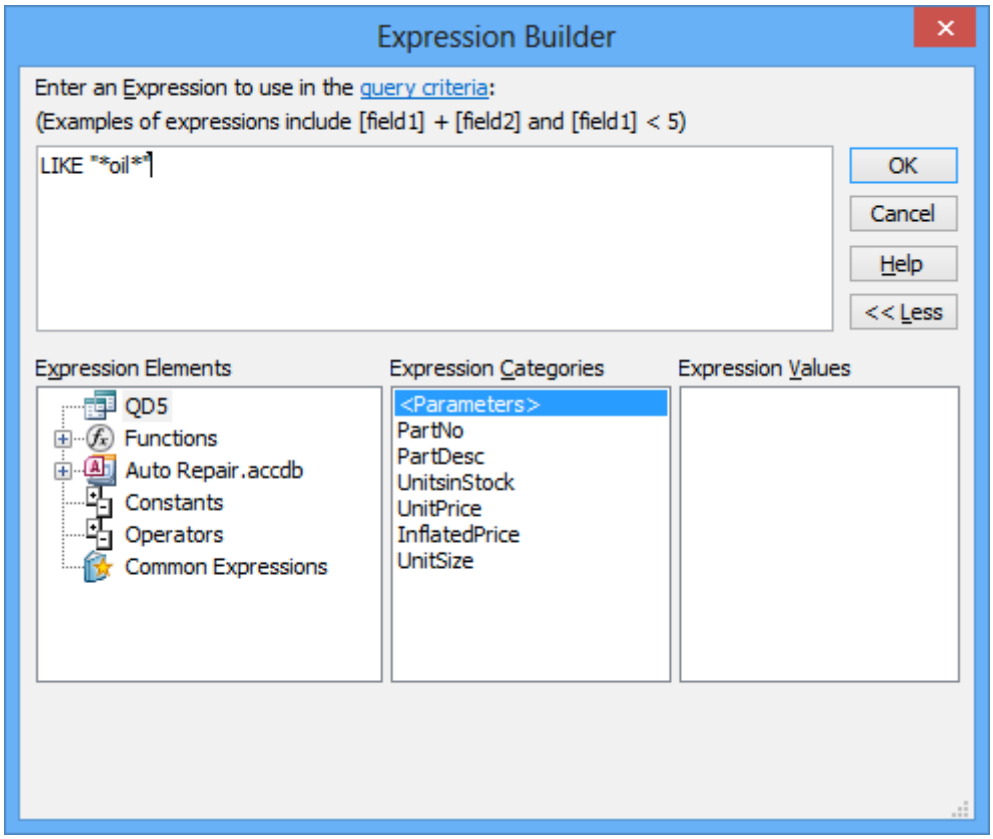


Figure 24: Expression Builder Window for "QD5"

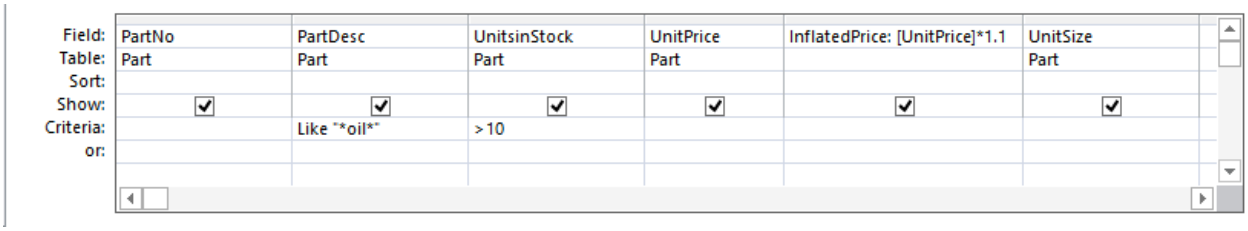


Figure 25: Completed "QD5"

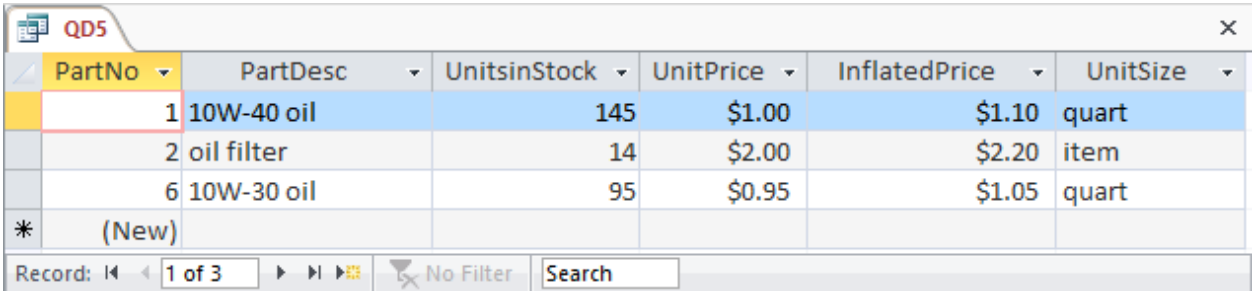


Figure 26: Datasheet of "QD5"

3.2.4 Creating a Join Query

Many customers who bring their vehicles into the auto repair shop own more than one vehicle. Therefore, it would be convenient for the employees to have a list of the customers and their vehicles. To accomplish this task, a query will be created to join the *Customer* and the *Vehicle* tables. Query Design makes the join connection automatically, as the following steps depict.

1. Create a New Query: Choose **Create → Query Design** from the Queries group to open the New Query window.
2. Select Two Tables: When the Show Table window appears, select the *Customer* and the *Vehicle* tables from the list by using the **Ctrl** key. Click the **Add** button to transfer both tables to the Query Design window. Click the **Close** button to finish.
3. View the Join Properties Window: A line appears connecting the two tables as they appeared in the Relationships window from Chapter 2.[3] Double-click on this line to reveal the Join Properties window (Figure 27). A join operator should connect these tables. Later in this chapter, you will use the outer join operator. Confirm that the first choice (join) is selected and click **OK**.
4. Place Fields: In the Query Design window, drag the first three fields from the *Customer* table and the first four fields from the *Vehicle* table to the Query Design table (Figure 28). Note that you have to scroll the table to the right to access additional fields to fill.
5. Execute the Query: You should see seven fields filled with data as in Figure 29. When you are finished viewing, close and save the query as "QD6" when prompted.

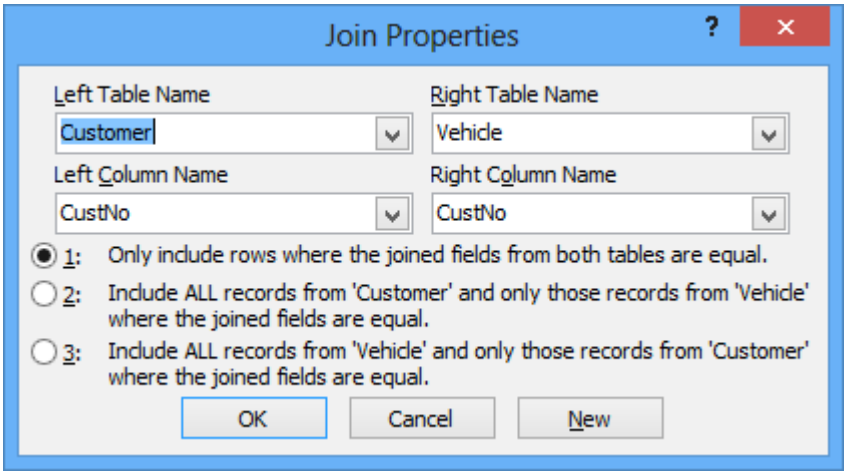


Figure 27: Join Properties Window

[3] If your Relationships window does not show thick lines (referential integrity is not supported), the lines in Figure 28 will appear as thin lines. You must enforce referential integrity in the Relationships window to make thick lines in the Join window (Figure 28).

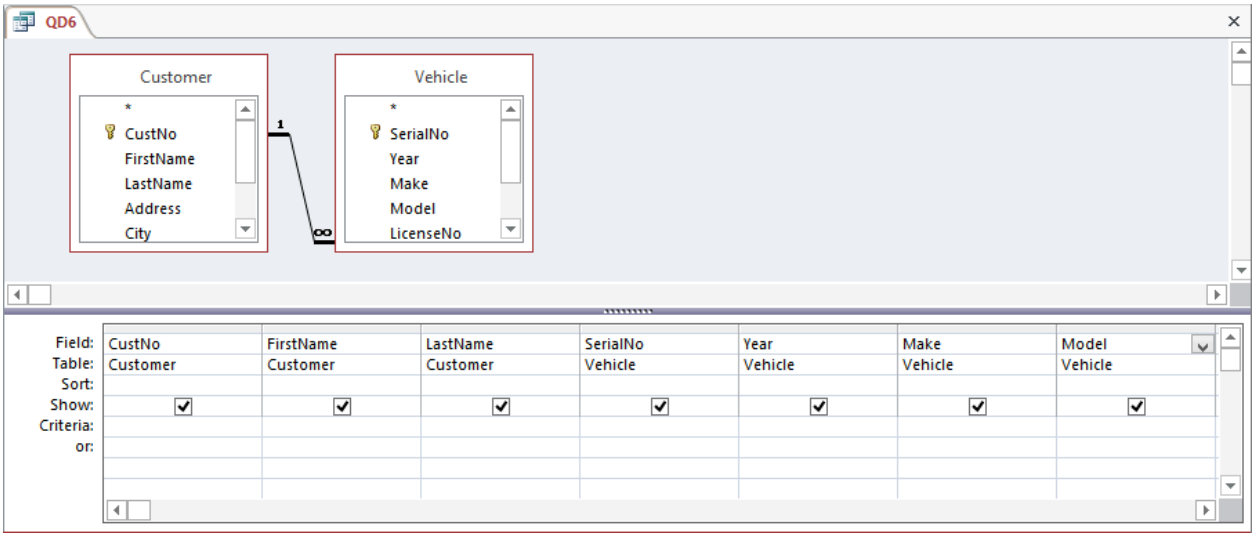


Figure 28: Completed "QD6"

QD6

CustNo	FirstName	LastName	SerialNo	Year	Make	Model
1	Beth	Taylor	QWER9LK982	2002	Pontiac	FireHawk
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal
2	Betty	Wise	JKLP7IIIJF	2004	Ford	Impala
3	Bob	Mann	POIU980PLL	2009	Chevrolet	Cavalier
4	Candy	Kendall	SWQW345RT6	2012	BMW	320i
5	Harry	Sanders	JHMEC3348H	1988	Buick	Corolla
5	Harry	Sanders	THYDF55639	2012	Ford	F-100
6	Helen	Sibley	TYYYI87590	2005	Toyota	Celica
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer
7	Homer	Wells	LPQW76200I	2007	Honda	Accord
8	Jerry	Wyatt	JHMCCF673Q	1995	Honda	Civic Si
8	Jerry	Wyatt	XCVY760PIQ	2003	Toyota	Celica
9	Jim	Glussman	MVMN8974PP	2009	Honda	Civic DX
9	Jim	Glussman	V121BHD481	2011	Toyota	Pick-up
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark
10	Larry	Styles	WWQAA452P0	1999	Buick	Regal

Record: 1 of 23 No Filter Search

Figure 29: Datasheet of "QD6" (first 16 rows)

3.2.5 Adding Another Table to the Join Query

In addition to knowing what customers own which vehicles, it also would be convenient to know what repair jobs have been performed for each customer. To accomplish this task, the *RepairOrder* table will be added to the previous query ("QD6"). Again, Query Design automatically connects the three tables, as the following steps describe.

1. Copy and Paste an Existing Query: To begin, copy "QD6" and rename it "QD7".
2. Open the Query Design Window: Next, open "QD7" in design view. It should appear the same as "QD6" (refer back to Figure 28).

3. Select an Additional Table: Next, click **Design → Show Table** from the Query Setup group. When the Show Table window appears, add the *RepairOrder* table and then click **Close**.

4. Open the Join Properties Window: In the Query Design window, double-click on the line connecting the *RepairOrder* and the *Vehicle* tables to open the Join Properties window. Confirm that the first choice is selected and click **OK**.

> Technical Note: In the future, you can avoid this step by remembering that the join operator is the default connection. You also can visually see that the connection is a join operator because of the 1 and ∞ symbols appearing in the line.

5. Place Additional Fields: In the Query Design window, drag the first three fields from the *RepairOrder* table to the Query Design table. Note that you will have to scroll the table to the right to access additional fields to fill.

6. Execute the Query: You should see ten fields filled with data as shown in Figure 30. When you are finished, close and save changes when prompted.

QD7

CustNo	FirstName	LastName	SerialNo	Year	Make	Model	OrdNo	Odometer	TimeRecvd
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	1	50000	10/5/2013 1:31:00 PM
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	4	75000	10/8/2013 11:15:00 AM
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	29	20600	11/25/2013 9:15:00 AM
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	6	55000	10/10/2013 4:42:02 PM
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	30	22590	11/27/2013 4:15:00 PM
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	31	23000	2/3/2002 10:20:00 AM
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	32	15000	12/1/2013 3:40:00 AM
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal	2	30000	10/5/2013 4:20:42 PM
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal	21	72700	11/5/2013 1:50:00 PM
5	Harry	Sanders	JHMEC3348H	1988	Buick	Corolla	25	61800	11/15/2013 8:10:00 AM
2	Betty	Wise	JKLP7IIIJF	2004	Ford	Impala	3	6000	10/7/2013 9:00:00 AM
2	Betty	Wise	JKLP7IIIJF	2004	Ford	Impala	15	14800	10/25/2013 9:05:00 AM
12	Ron	Thompson	MNMJ7H6098	2002	Lincoln	Towne Car	11	55060	10/18/2013 12:04:47 PM
12	Ron	Thompson	MNMJ7H6098	2002	Lincoln	Towne Car	17	71500	10/30/2013 2:35:00 PM
9	Jim	Glussman	MVMN8974PP	2009	Honda	Civic DX	18	15800	11/1/2013 12:10:00 PM
16	Wally	Jones	PLJFVC5609	2006	Toyota	Landcruiser	8	125000	10/12/2013 2:30:00 PM

Record: 1 of 32 No Filter Search

Figure 30: Datasheet of "QD7" (first 16 rows)

3.2.6 Creating an Outer Join Query

The goal of the next query is to create a list that matches repair orders with associated vehicles. This list is especially handy for employees since, as previously mentioned, many customers own more than one vehicle. To build this list, this query should join the *RepairOrder* and the *Vehicle* tables. But note that a difficulty arises because employees would like to see vehicles that have never been repaired. The join operator (as described in textbook Chapters 3 and 4) excludes nonmatching rows. The outer join operator is needed to include nonmatching rows. A one-sided outer join operator that preserves the *Vehicle* rows is needed. You might want to review the definition of the outer join operator in textbook Chapter 3 if you do not remember it. Access directly supports the one-sided outer join operator, as the following steps describe.

1. Create a New Query: Choose **Create → Query Design** from the Queries group to open the New Query window.

2. Select Two Tables: When the Show Table window appears, select the *RepairOrder* and the *Vehicle* tables, click the **Add** button**,** then click the **Close** button.

3. Change the Connection Type: In the Query Design window, double-click on the line connecting the tables to open the Join Properties window (refer back to Figure 27). Select the second choice, "Include ALL records from 'Vehicle' and only those records from 'RepairOrder' where the joined fields are equal.", and then click **OK**. This choice connects the tables with a one-sided join that preserves the *Vehicle* records. In Figure 31, note the arrow from the *Vehicle* to the *RepairOrder* table indicating a one-sided outer join.

4. Place Fields: In the Query Design window, drag the first three fields from the *RepairOrder* table and the first four fields from the *Vehicle* table to the Query Design table.

5. Sort the Result: In the *Sort* row of the *OrdNo* field, select "Ascending" from the list.

6. Execute the Query: You should see seven fields with data, as shown in Figure 32. Because of the sorting, the unmatched rows appear first. Unmatched rows have no values for fields from the *RepairOrder* table. When you are finished viewing, close and save the query as "QD8" when prompted.

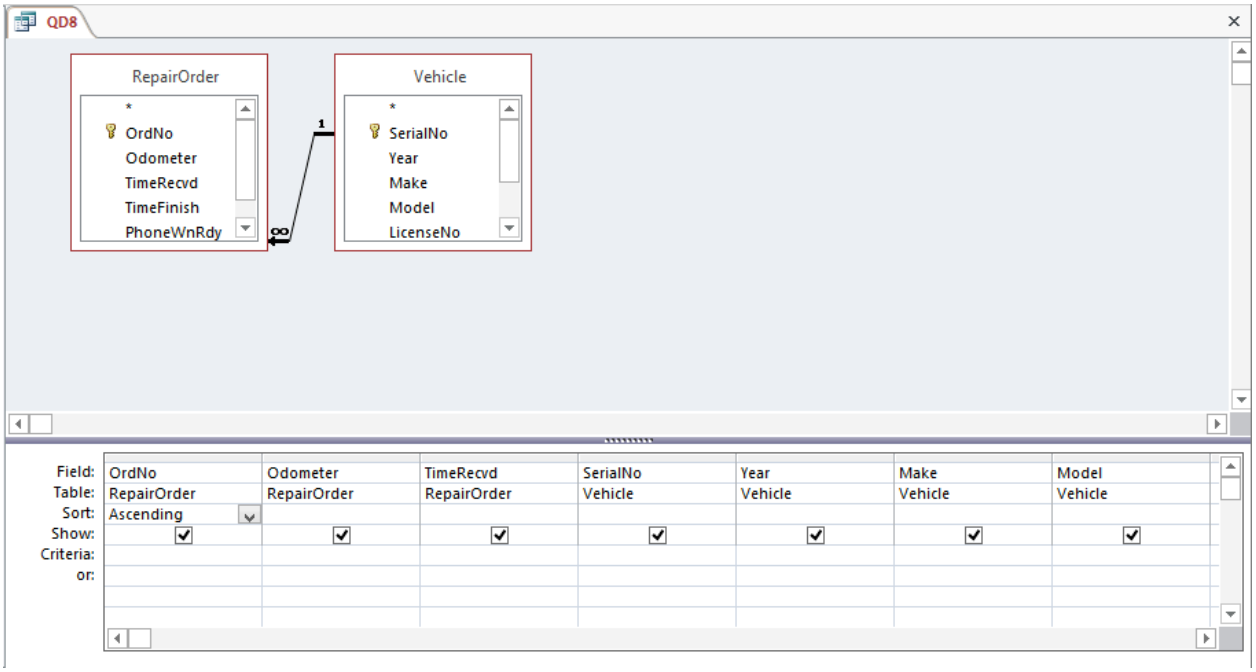


Figure 31: Query Design Window of Completed "QD8"

QD8

OrdNo	Odometer	TimeRecvd	SerialNo	Year	Make	Model
			GHL39789UI	1996	Honda	Del Sol
			LPQW76200I	2007	Honda	Accord
			JHMCCF673Q	1995	Honda	Civic Si
			VHJK009L88	2000	Ford	Taurus
1	50000	10/5/2013 1:31:00 PM	AZXS230I87	2004	Ford	Skylark
2	30000	10/5/2013 4:20:42 PM	IOMEQ54397	2003	Buick	Regal
3	6000	10/7/2013 9:00:00 AM	JKLP7IIIJF	2004	Ford	Impala
4	75000	10/8/2013 11:15:00 AM	AZXS230I87	2004	Ford	Skylark
5	15000	10/10/2013 11:53:30 AM	TYYYI87590	2005	Toyota	Celica
6	55000	10/10/2013 4:42:02 PM	BHGY08631Q	2008	Chevrolet	Blazer
7	25500	10/11/2013 11:53:32 AM	V121BHD481	2011	Toyota	Pick-up
8	125000	10/12/2013 2:30:00 PM	PLJFVC5609	2006	Toyota	Landcruiser
9	60800	10/15/2013 12:03:01 PM	XCVY760PIQ	2003	Toyota	Celica
10	11000	10/16/2013 11:41:00 AM	THYDF55639	2012	Ford	F-100
11	55060	10/18/2013 12:04:47 PM	MNMJ7H6098	2002	Lincoln	Towne Car
12	25600	10/20/2013 9:08:00 AM	THYDF55639	2012	Ford	F-100

Record: 16 of 36 No Filter Search

Figure 32: Datasheet of "QD8" (first 16 rows)

3.3 Creating Queries in SQL View

With a tool as convenient as Query Design, you may wonder why you would ever use SQL. Access provides SQL because it is the industry standard language. Query Design is specific to Access. In addition, some queries are either difficult or impossible to formulate in Query Design. Thus, SQL is preferred for some queries and the only alternative for other queries.

This section provides you additional practice creating queries in SQL view. Note that you will be creating queries by typing statements directly into SQL view since the expression builder is not available in SQL view. The SQL view is a much simpler tool than Query Design.

To access SQL view for a new query, you first must open the query in design view. When the Design View window appears with the Show Table dialog box, just click the **Close** button without adding any tables. Then toggle to SQL View via the **Query Tools | Design → SQL** command in the Results group or the SQL button in the view bar. Remember that you still may toggle between SQL and design view to examine your query.

3.3.1 Another Modification to the Parts List

Now you will return to the parts list with which you were working previously. To help employees in reordering decisions, the new query should display parts that have quantities less than 10.

1. Create a New Query: Create a new query in design view and then open the SQL View window. Type the following statement in the SQL window as shown in Figure 33. After typing the statement, execute it and return to SQL view.

```
SELECT *
 FROM Part
 WHERE UnitsInStock < 10;
```

2. Correcting Mistakes: If you had made a mistake when typing the query, Access would let you know when you try to run the query. To demonstrate an error, insert an extra letter in the field name *UnitsInStock*, such as an extra "k" at the end: *UnitsInStockk*. Now try to run the query. Instead, you see a dialog asking you for a parameter value (Figure 34). In this case, a parameter dialog signifies an error, so click **Cancel** and correct the mistake. Later in this chapter, you will write a query that uses a parameter value. Unless you use parameters, the parameter dialog signifies an error, usually a misspelled field name.

3. Execute the Query Again: After correcting the mistake, you should see the fields filled with data as in Figure 35. When you are finished viewing, close and save the query as "SQL1" when prompted.

Figure 33: SQL View

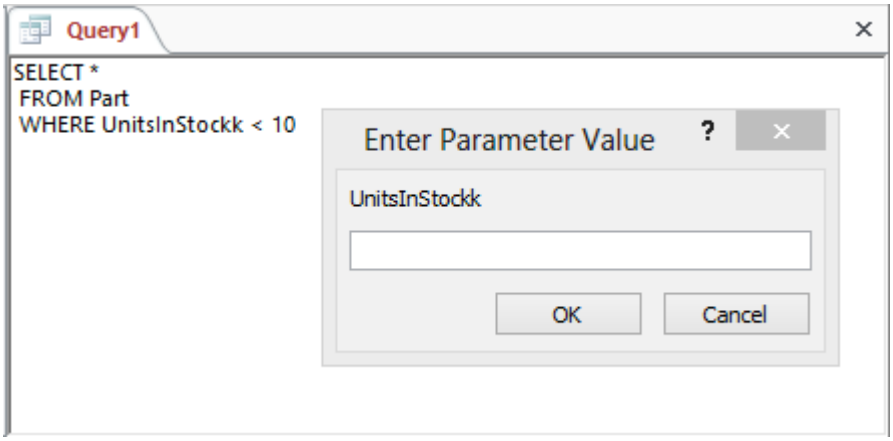


Figure 34: Parameter Dialog Indicating an Error

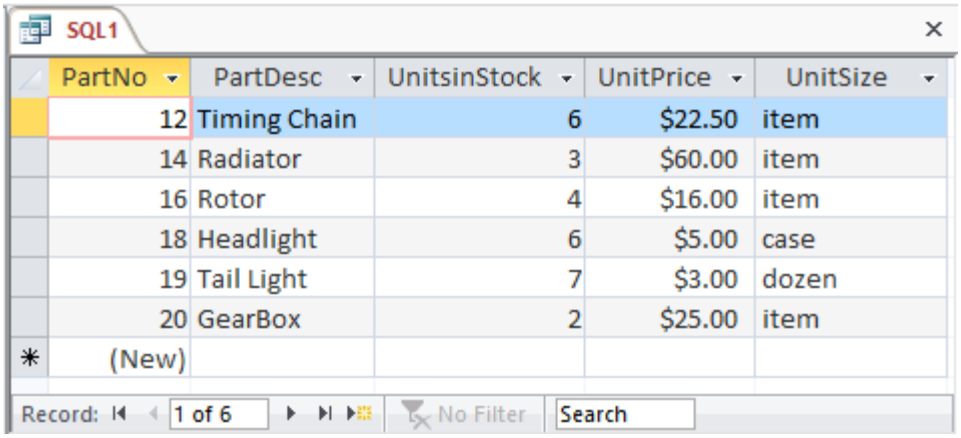


Figure 35: Datasheet of "SQL1"

3.3.2 Adding a Parameter Name to a Query

Next, you are going to refine the parts list to make it even more flexible. Instead of displaying a list of parts that have been depleted to 10 or less, you will use a parameter query to enter any desired cutoff value. When the query is run, a dialog box will appear prompting the user to enter a parameter value.

1. Create a new query: Create a new query in design view and then open the SQL View window. Type the following statement:

```
SELECT *
FROM Part
WHERE UnitsInStock < [Reorder Point:]
```

Note that square brackets "[]" enclose the parameter name. The square brackets must be used whenever a field name or parameter name contains spaces. Since *Reorder Point* is not a field name in the auto repair database, Access treats it as a parameter, as demonstrated next.

2. Execute the Query: When you execute the query, the dialog box in Figure 36 appears. Type in a number (5 is typed in Figure 36) and click **OK**. The datasheet displays the result as shown in Figure 37.

3. Close and Save the Query: When you are finished viewing, close and save the query as "SQL2" when prompted.

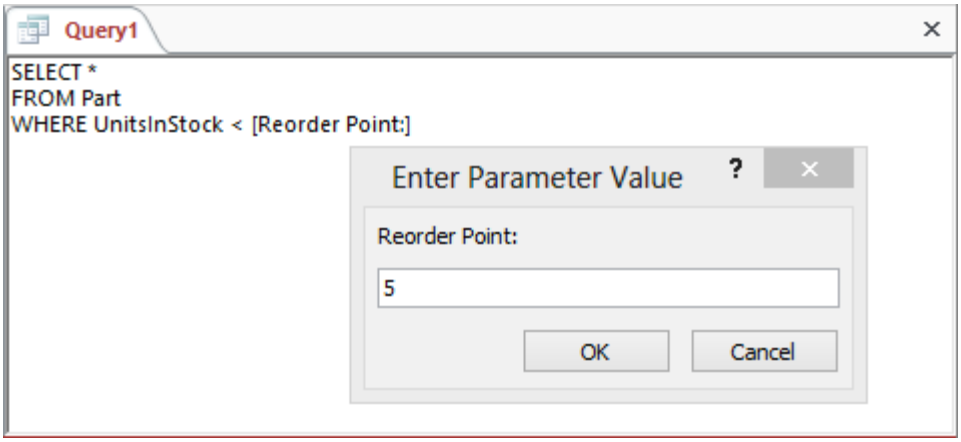


Figure 36: Parameter Value Dialog

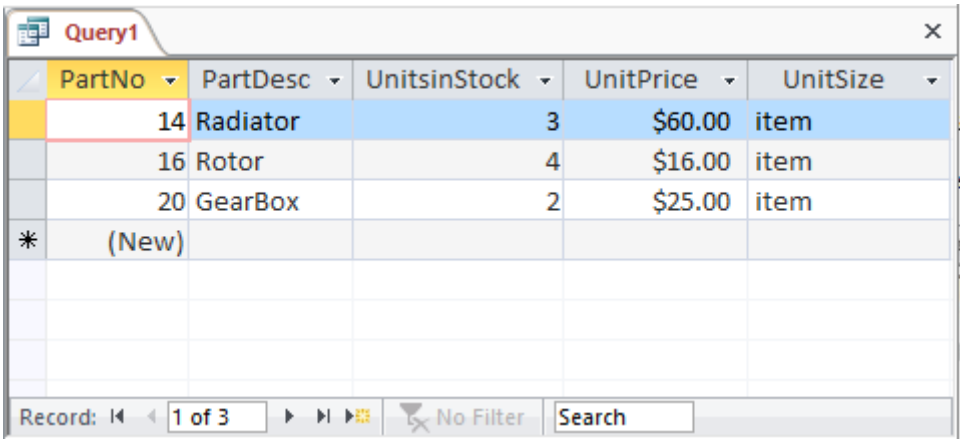


Figure 37: Datasheet of "SQL2"

3.3.3 Using a Query to Summarize Data

For month-end record keeping, the accounting department needs to know how many parts were used on each repair job. This task requires a query with the *Part* and *PartsUsed* tables along with a GROUP BY to perform the summarization. You will create this query in SQL view, but then toggle to observe the representation in design view.

1. Create a New Query: Create a new query in the Query Design window and then open the SQL window. Type in the following expression:

```
SELECT Part.PartNo, PartDesc, Count(*) AS
 [Total Repairs], Sum(QtyUsed) AS [Total Quantity]
FROM Part INNER JOIN PartsUsed
      ON Part.PartNo = PartsUsed.PartNo
GROUP BY Part.PartNo, PartDesc;
```

Note that this SQL statement uses the join operator style discussed in textbook Chapter 3. This style is used here because Query Design recognizes it. The cross product style used in textbook Chapter 4 is not understood as a join operation by Query Design.

2. Execute the Query: The datasheet displays the result as in Figure 38.
3. Toggle to Design View: To view this query in design view, click **Query Tools | Design → View → Design View** from the Results group or use one of the methods explained earlier in this chapter.
4. View Query Design Window: The Query Design window shows the join operation and the group by operation (Figure 39). When you are finished viewing, close and save the query as "SQL3" when prompted.

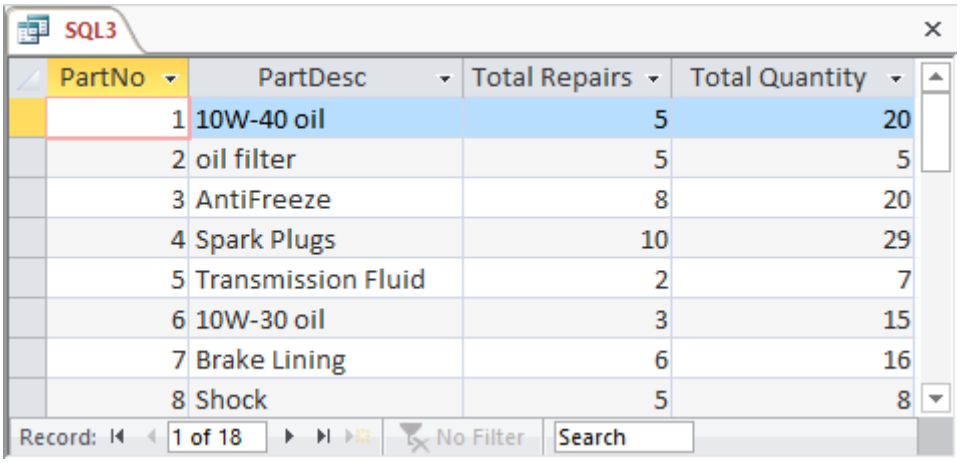


Figure 38: Datasheet of "SQL3" (first 8 rows)

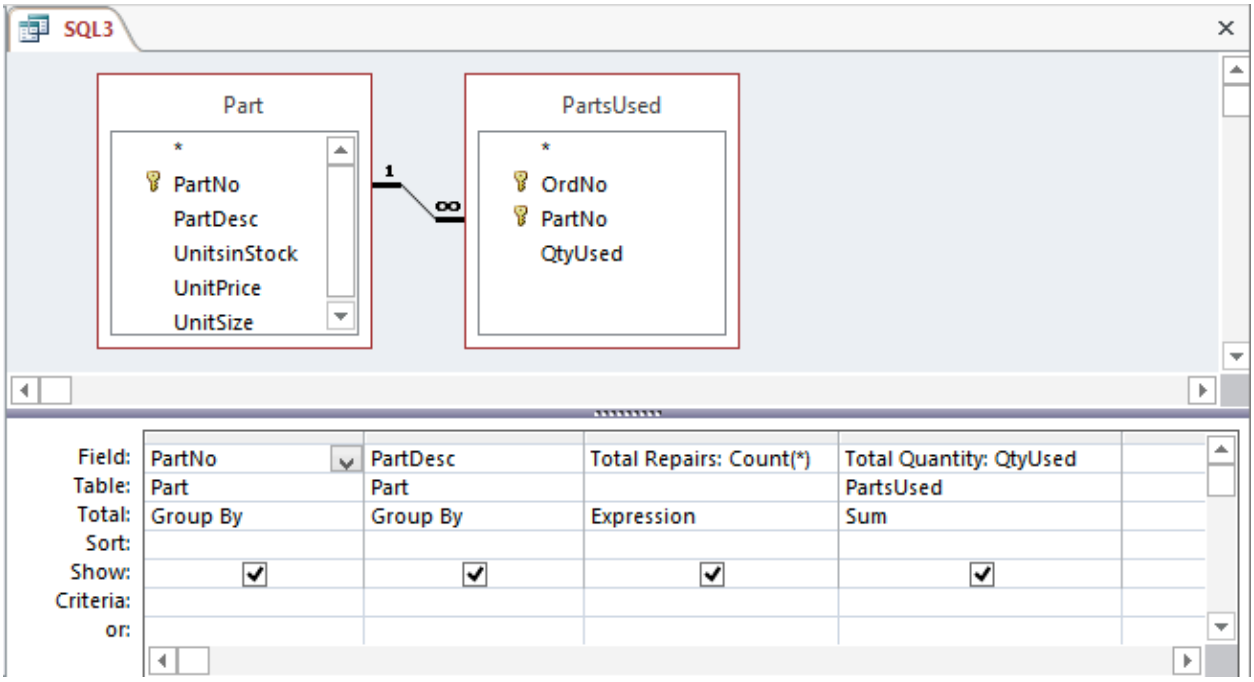


Figure 39: "SQL3" in Design View

3.3.4 Adding a Table to an Existing Query

The next query adds *PartsUsed* data to the list we made earlier showing repair jobs for each customer ("QD7"). You accomplished this task by adding the *RepairOrder* table to a previous query that contained only the *Customer* and the *Vehicle* tables. This time you will add a fourth table, the *PartsUsed* table, to the query "QD7". In addition, this query demonstrates using SQL to modify a query created in design view.

1. Copy and Paste an Existing Query: To begin, copy "QD7" and rename it as "SQL4".
2. Open the Query Design Window: Next, open "SQL4" in design view. It should appear on your screen the same as "QD7".
3. View the SQL Query: Open the SQL View window and view the code as shown below:

```
SELECT Customer.CustNo, Customer.FirstName,
 Customer.LastName, Vehicle.SerialNo, Vehicle.Year,
 Vehicle.Make, Vehicle.Model, RepairOrder.OrdNo,
 RepairOrder.TimeRecvd, RepairOrder.Odometer
FROM (Customer INNER JOIN Vehicle
 ON Customer.CustNo = Vehicle.CustNo)
 INNER JOIN RepairOrder
 ON RepairOrder.SerialNo = Vehicle.SerialNo;
```

4. Add a Table to the Query: To add the *PartsUsed* table to the query, type the additional code seen underlined below. The full SQL statement also appears in Figure 40. Note that you should add the *PartNo* and *QtyUsed* fields from the *PartsUsed* table since *OrdNo* is already contained in the query from the *RepairOrder* table.

```
SELECT Customer.CustNo, Customer.FirstName,
 Customer.LastName, Vehicle.SerialNo, Vehicle.Year,
 Vehicle.Make, Vehicle.Model, RepairOrder.OrdNo,
 RepairOrder.TimeRecvd, RepairOrder.Odometer,
 PartsUsed.PartNo, PartsUsed.QtyUsed
FROM ((Customer INNER JOIN Vehicle
 ON Customer.CustNo = Vehicle.CustNo)
 INNER JOIN RepairOrder
 ON RepairOrder.SerialNo = Vehicle.SerialNo)
 INNER JOIN PartsUsed
 ON RepairOrder.OrdNo = PartsUsed.OrdNo;
```

5. Execute the Query: Execute the query to see if everything is typed in correctly. The datasheet should have 12 fields across as in Figure 41. When you are finished viewing, close and save the changes when prompted.

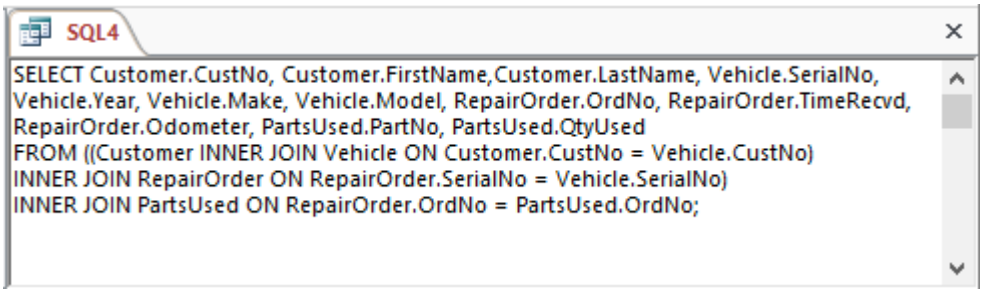
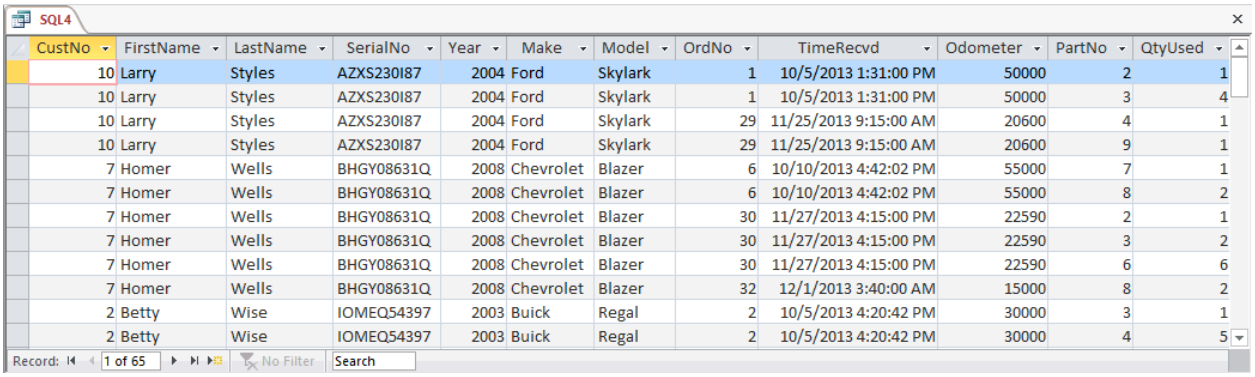


Figure 40: SQL Window Containing "SQL4"



SQL4

CustNo	FirstName	LastName	SerialNo	Year	Make	Model	OrdNo	TimeRecvd	Odometer	PartNo	QtyUsed
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	1	10/5/2013 1:31:00 PM	50000	2	1
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	1	10/5/2013 1:31:00 PM	50000	3	4
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	29	11/25/2013 9:15:00 AM	20600	4	1
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	29	11/25/2013 9:15:00 AM	20600	9	1
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	6	10/10/2013 4:42:02 PM	55000	7	1
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	6	10/10/2013 4:42:02 PM	55000	8	2
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	30	11/27/2013 4:15:00 PM	22590	2	1
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	30	11/27/2013 4:15:00 PM	22590	3	2
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	30	11/27/2013 4:15:00 PM	22590	6	6
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	32	12/1/2013 3:40:00 AM	15000	8	2
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal	2	10/5/2013 4:20:42 PM	30000	3	1
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal	2	10/5/2013 4:20:42 PM	30000	4	5

Record: 1 of 65 No Filter Search

Figure 41: Datasheet of "SQL4"

3.4 Creating Special Queries

In addition to the queries described above, there are a few queries with specific uses that are explained below. They are the Append query, the Update query, the Delete query, the Union query, and the Make Table query. The SQL formulation for these kinds of queries is shown because Query Design does not fully support formulations of them.

3.4.1 The Append Query

An Append query inserts records into a table. You create an Append query using the SQL INSERT statement as discussed in textbook Chapter 4. An Append query cannot be viewed in Design view after it is created in SQL view. Query Design supports Append queries in which a collection of records from one table is inserted into another table. You will use an Append query to insert an additional record into the *Part* table.

1. Create a New Query: Create a new query in design view and open the SQL window. Type in the following statement:

```
INSERT INTO Part ( PartDesc, UnitsInStock, UnitPrice,
 UnitSize )
SELECT "Gas Filter" AS Expr1, 10 AS Expr2,
 15.5 AS Expr3, "item" AS Expr4
```

Note that this syntax differs from that shown in textbook Chapter 4. Using the syntax of textbook Chapter 4, the INSERT statement appears as shown below. If you use this syntax be warned that Access changes it to the syntax with the `SELECT` keyword.

```
INSERT INTO Part (PartDesc, UnitsInStock, UnitPrice,
 UnitSize )
VALUES ("Gas Filter", 10, 15.5, "item");
```

2. Execute the Query: When you try to execute the query, the message in Figure 42 appears. Click the **Yes** button to insert the new record. You should only execute this query once. Executing it more than one time will insert duplicate records, except for the unique *PartNo* value generated by Access. Close and save the query as "SQL5" when prompted.

3. View the New Record: Now, return to the **Tables** section of the Database window and open the *Part* table to see the additional row (Figure 43).

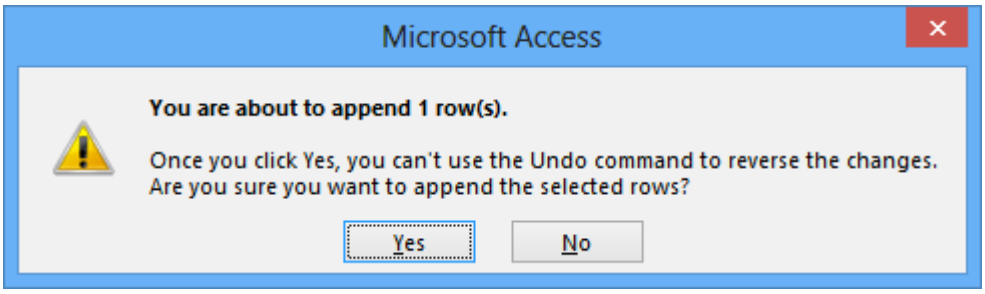


Figure 42: Append Message Box

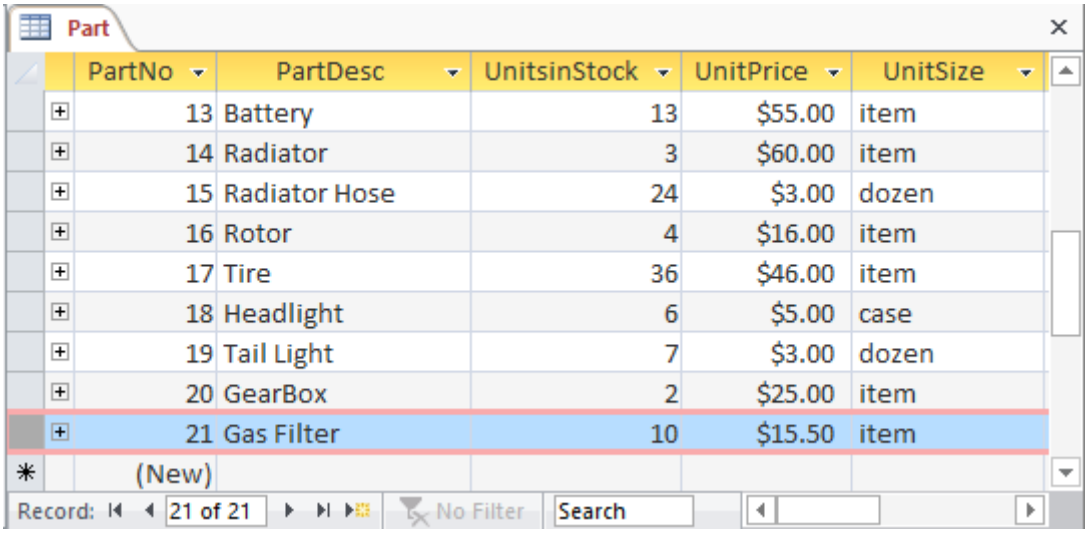


Figure 43: Bottom Part of Datasheet (*Part* Table) of "SQL5"

3.4.2 The Update Query

An Update query changes the values of individual fields of selected records. You create an Update query using the SQL UPDATE statement as discussed in textbook Chapter 4 or using Query Design. Unlike an Append query, an UPDATE query can be viewed in design view after it is created in SQL view. You will use an Update query to decrease the quantity of gas filters in the *Part* table.

1. Create a New Query: Create a new query in design view and open the SQL window. Type the following statement:

```
UPDATE Part SET UnitsInStock = UnitsInStock - 1
 WHERE PartDesc = "Gas Filter";
```

2. Execute the Query: When you try to execute the query, the message in Figure 44 appears. Click the **Yes** button to update the record. Update queries can be repeatedly executed if you desire. If you execute this query two more times, the *UnitsInStock* field is reduced by 2. Close and save the query as "SQL6" when prompted.

3. View the Changes: Return to the **Tables** section of the Database window and open the *Part* table. You can see that the number of gas filters in Figure 45 has decreased from 10 to 9 in comparison to the datasheet in Figure 43 above.

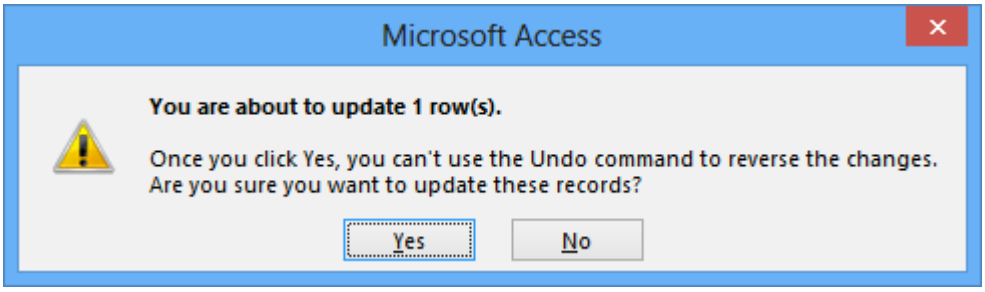


Figure 44: Update Message Box

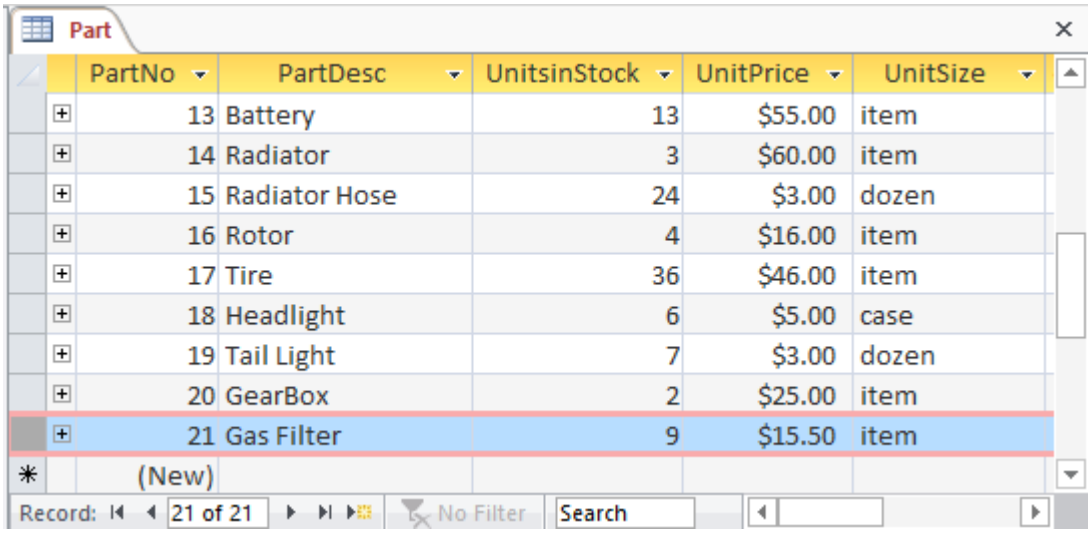


Figure 45: Bottom Part of Datasheet (*Part* Table) of "SQL6"

3.4.3 The Delete Query

A Delete query removes selected rows from the database. You create a Delete query using either the SQL DELETE statement, as discussed in textbook Chapter 4, or Query Design. Like Append queries, Delete queries cannot be viewed in design view after created in SQL view. You will use a Delete query to remove the gas filter record from the *Part* table.

1. Create a New Query: Create a new query in design view and open the SQL window. Type the following statement:

```
DELETE *
FROM Part
WHERE PartDesc = "Gas Filter";
```

2. Execute the Query: When you try to execute the query, the message in Figure 46 appears. Click **Yes** to delete the row. You should only perform Delete queries once. If you perform them more than one time, you will generate an error message about the record not existing. Close and save the query as "SQL7" when prompted.

3. View the Changed *Part* Table: Return to the **Tables** part of the navigation pane. Open the *Part* table to see that the gas filter row has been deleted (Figure 47).

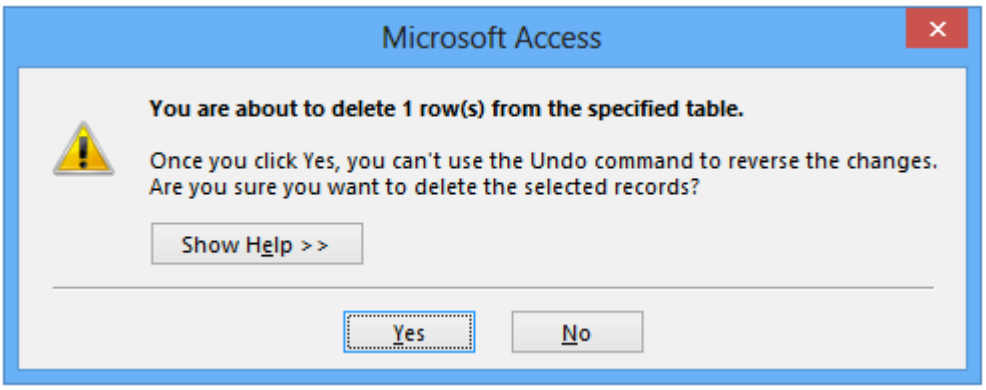


Figure 46: Delete Message Box

Part

PartNo	PartDesc	UnitsinStock	UnitPrice	UnitSize
12	Timing Chain	6	$22.50	item
13	Battery	13	$55.00	item
14	Radiator	3	$60.00	item
15	Radiator Hose	24	$3.00	dozen
16	Rotor	4	$16.00	item
17	Tire	36	$46.00	item
18	Headlight	6	$5.00	case
19	Tail Light	7	$3.00	dozen
20	GearBox	2	$25.00	item
(New)				

Record: 20 of 20 No Filter Search

Figure 47: Bottom Part of Datasheet (*Part* Table) of "SQL7"

3.4.4 The Union Query

A Union query combines the results of two or more queries using the union operator. The union operator in SQL, like the union operator in relational algebra, requires "union compatible" tables. Recall from textbook Chapter 3 that two tables are union compatible if they have the same number of columns and each corresponding column has a compatible data type. You will use a Union query to display a vehicle list containing Honda or Toyota models. Notice that each SELECT part of the query is union compatible. A Union query cannot be viewed in design view after it is created in SQL view.

1. Create a New Query: Create a new query in design view and open the SQL window. Type the following statement in the SQL window:

```
SELECT * FROM Vehicle WHERE Make = "Honda"
 UNION
SELECT * FROM Vehicle WHERE Make = "Toyota";
```

2. Execute the Query: The datasheet displays the result as shown in Figure 48. When you are finished viewing the result, close and save the query as "SQL8" when prompted.

SQL8

SerialNo	Year	Make	Model	LicenseNo	State	Cylinders	CustNo
GHL39789UI	1996	Honda	Del Sol	123XYZ	WA	4	13
JHMCCF673Q	1995	Honda	Civic Si	367ABC	WA	4	8
LPQW76200I	2007	Honda	Accord	123ABC	MT	4	7
MVMN8974PP	2009	Honda	Civic DX	567TUV	WA	4	9
PLJFVC5609	2006	Toyota	Landcruiser	876GYK	WA	4	16
TGVYY76R43	2008	Honda	Civic	980QAW	WA	4	15
TYYYI87590	2005	Toyota	Celica	890YUI	ID	6	6
V121BHD481	2011	Toyota	Pick-up	190PGF	WA	6	9
XCVY760PIQ	2003	Toyota	Celica	477HSD	WA	4	8

Record: 1 of 9 No Filter Search

Figure 48: Datasheet of "SQL8"

3.4.5 Make-Table Query

Microsoft Access 2013 provides an additional special query known as a Make-Table Query. Using a Make-Table Query, you can copy selected rows of an existing table or query into a new table. To create a Make-Table query, you select the **Query Tools | Design → Make-Table**… command in the Query Type group for an empty query open in SQL or design view. Then you provide a table name in the Make Table dialog window. Access generates a SELECT … INTO statement with the specified table name inserted after the INTO clause. For more information, you should review the SELECT … INTO statement in the Help documentation.

The SELECT … INTO statement[4] is not standard SQL although it is a useful variant of the SELECT statement. Standard SQL includes the INSERT … SELECT statement (see textbook Chapter 4) to insert a set of rows from another table into an existing table. Thus, the SELECT … INTO statement creates a new table whereas the INSERT … SELECT statement must use an existing table. If you are developing an application that may need to be converted to another DBMS, you should avoid the SELECT … INTO statement.

3.4.6 Summary of Special Queries

Because of the special nature of the Append, Update, Delete, and Union queries, Access provides separate icons for them. In Figure 49, you can see the + icon for the Append query (SQL5), the pencil icon for the Update query (SQL6), the × icon for the Delete query (SQL7), and the intersecting rings icon for the Union query (SQL8).

[4] The Access SELECT … INTO statement is different than the SELECT … INTO statement for embedded SQL. The embedded SQL SELECT … INTO, part of standard SQL, supports retrieval of single-row results into programming language variables. Textbook Chapter 11 describes the embedded SQL SELECT … INTO statement in the presentation of stored procedures.

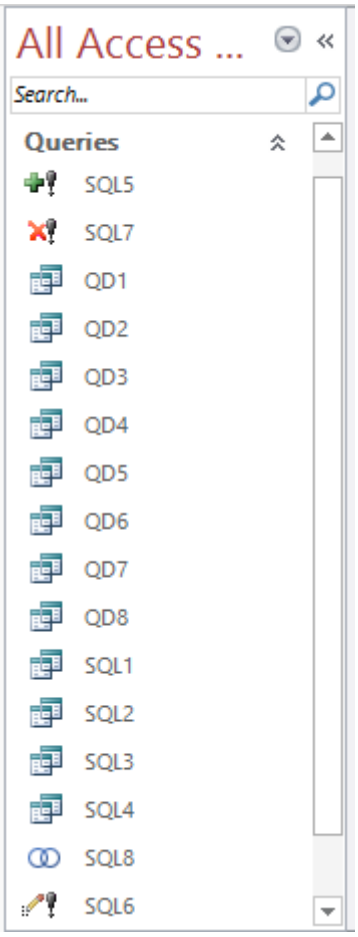


Figure 49: List of Queries Used in This Chapter

3.5 Query Subdatasheets

As you may have noticed, none of the query datasheets in this chapter displayed the + symbol indicating an available subdatasheet. Although query datasheets are capable of displaying subdatasheets, there is no default setting as there was for table datasheets. A subdatasheet may be displayed by setting the desired table or query name in the *Subdatasheet Name* query property in the Query Properties window as you did for table datasheets. You also need to set the *Link Master Fields* and the *Link Child Fields* properties to establish a connection between the datasheet and the subdatasheet.

As an example, you can open the QD1 query in design view and set a property to display a subdatasheet. Remember that QD1 contains customer information, so when you are finished you will have a subdatasheet to show the customer's vehicle information.

- To open the Query Properties window, right-mouse-click in an empty area in the Query Design pane and choose **Properties…** (Figure 50).
- Scroll down to locate the *Subdatasheet Name* property. Click in that property to reveal a drop-down menu and select "Table.Vehicle" (Figure 51).
- Toggle to datasheet view (Figure 52) and click on a + sign to expand the subdatasheet. All vehicles are shown for each customer because no connection has been established between the datasheet and the subdatasheet.

- Toggle back to Query Design and locate the *Subdatasheet Name* property again. Set the *Link Master Fields* and the *Link Child Fields* properties to "CustNo" as shown in Figure 53.
- Toggle to datasheet view again and click on a + sign to expand the subdatasheet as shown in Figure 54. Now the subdatasheet displays only the vehicles of the given customer, not all vehicles in the database.
- Toggle to back to Query Design and locate the *Subdatasheet Name* property again. Delete the value so that it is blank (its original state before this example). Now when you toggle to datasheet view, the + symbols have disappeared. Close the datasheet without saving changes.

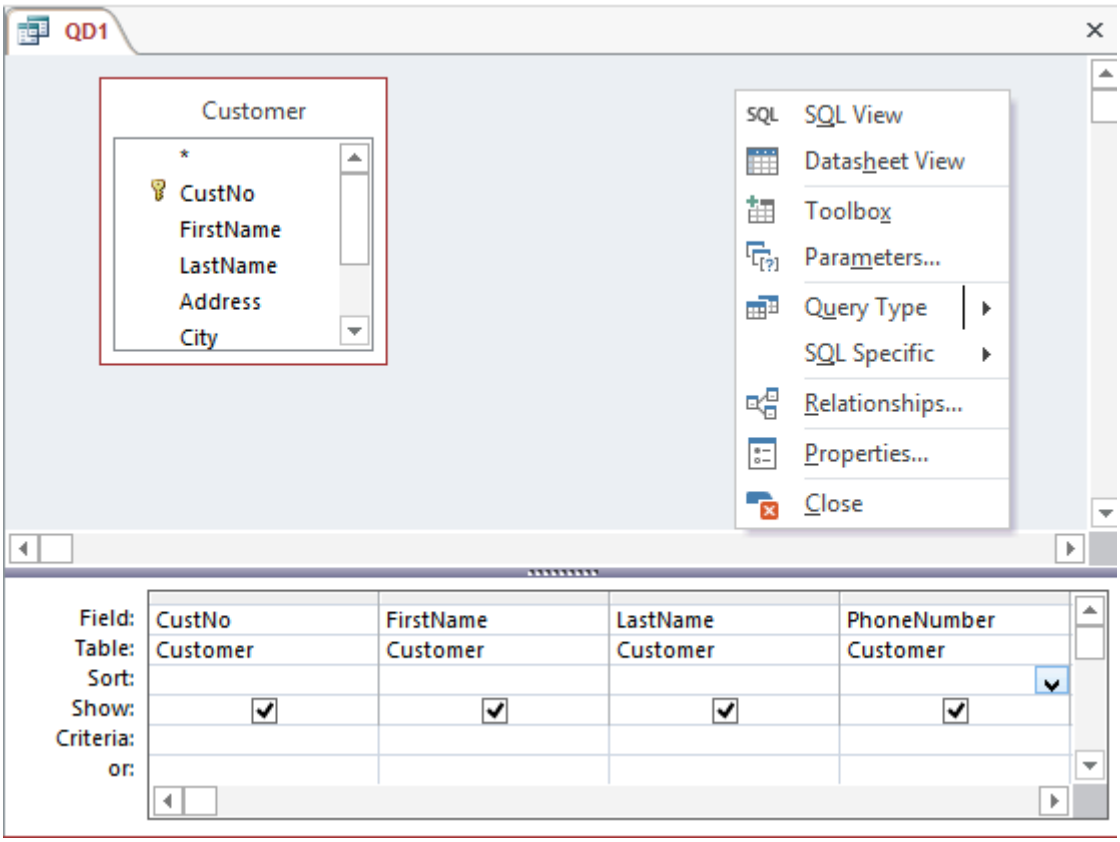


Figure 50: Design View and the Right Mouse Shortcut Menu Showing **Properties…**

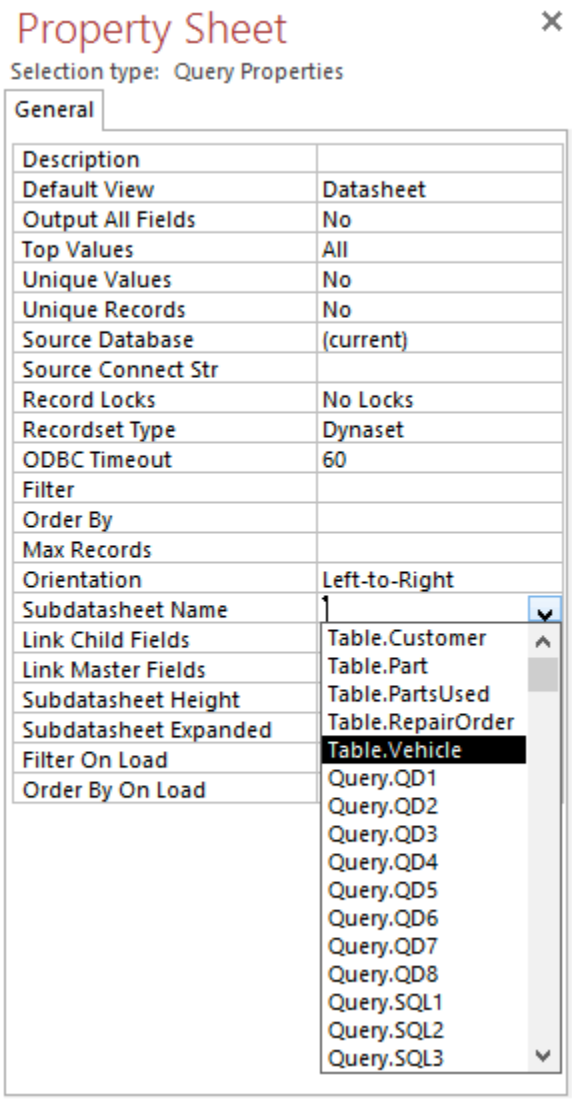


Figure 51: Query Properties Window Showing *Subdatasheet Name* Property

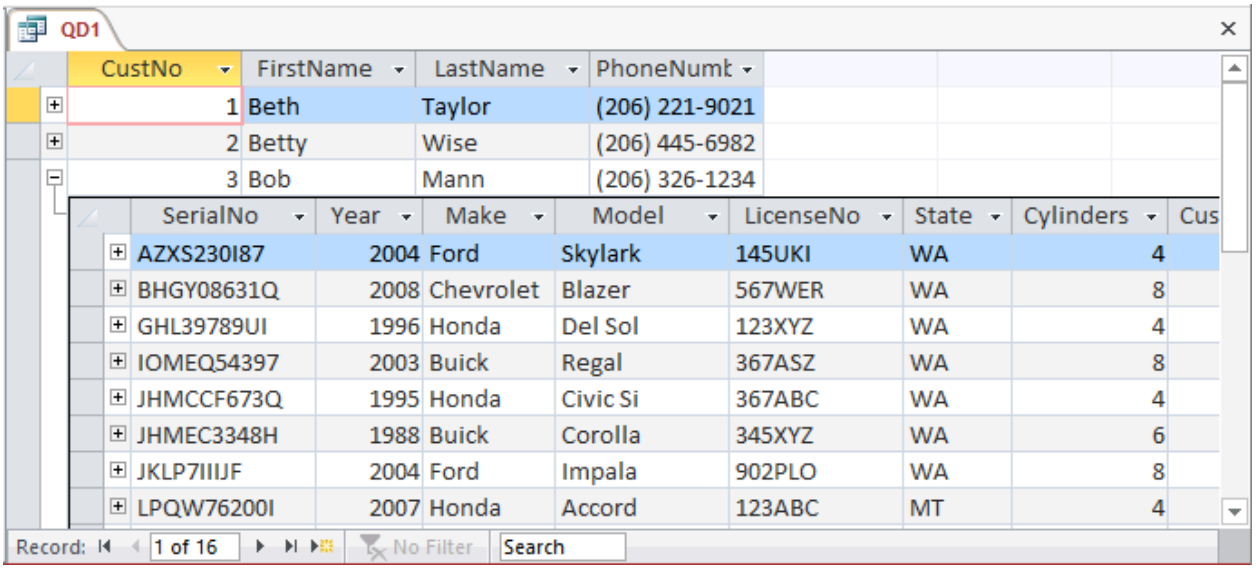


Figure 52: Query Datasheet with Hidden Subdatasheet

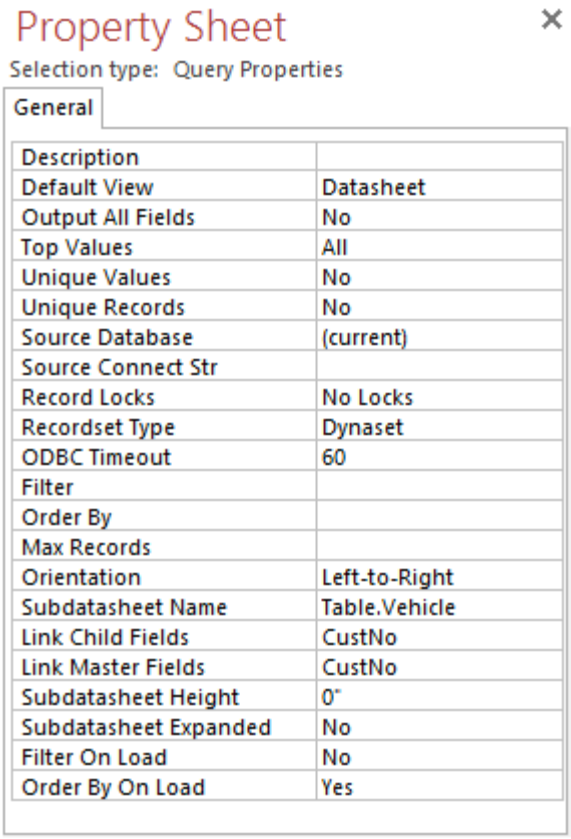


Figure 53: Query Properties Window Showing the Linking Properties

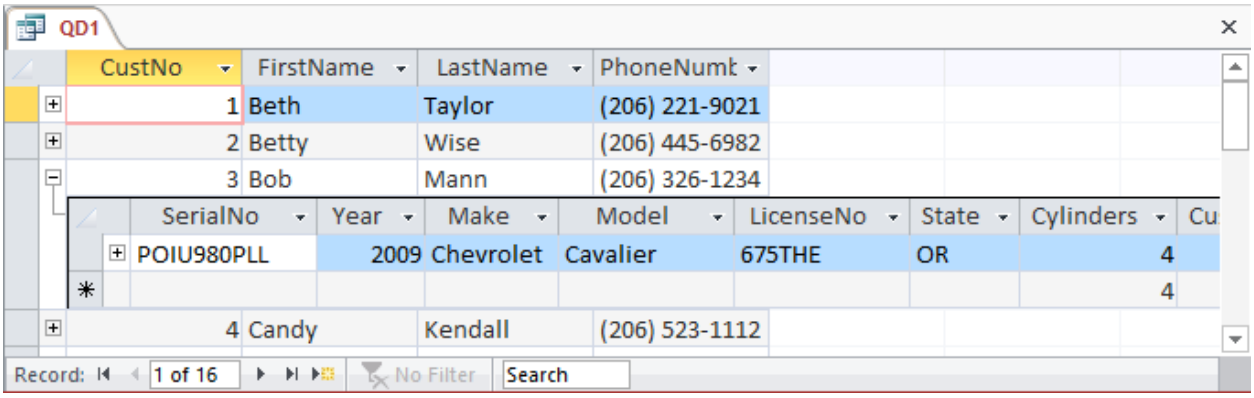


Figure 54: Expanded Subdatasheet after Setting the Linking Properties

3.6 Standards Compliance of Access SQL

As explained in textbook chapters 3, 4, and 9, international standards organizations such as the American National Standards Institute (ANSI) have specified a number of SQL standards. The most recent standards are known as SQL-89, SQL-92, and SQL:1999 in reference to the years in which the standards were formally accepted. The SQL:1999 standard has been updated with new parts in the SQL:2003, SQL:2008, and SQL:2011 standards. In each subsequent standard, the size of the specification has grown. For example, the SQL-92 specification contains about 600 pages whereas the SQL:1999 specification contains about 2,000 pages. Most relational DBMSs (including Access) support a subset of a standard (typically SQL-92) while adding proprietary extensions.

This section supplements the presentation in textbook chapters 3, 4, and 9 by providing insight into the standards compliance of Access SQL. The compliance of Access is more complex than other relational DBMSs because native Access databases using the Jet database engine support both the SQL-89 and the SQL-92 specifications along with proprietary extensions. Although Access does support both SQL specifications, it does not satisfy level 1 compliance for either specification. In addition, Access supports Microsoft SQL Server syntax after converting an Access database to a SQL Server database. SQL Server supports a subset of SQL-92 with some proprietary extensions.

3.6.1 ANSI Query Mode

Access 2013 supports the choice of SQL-89 or SQL-92 query mode using the Object Designers item in the Access Options window (**File → Options**). The choice can be made in the bottom right of the window in the "SQL Server Compatible Syntax (ANSI 92)" area. You can choose to make SQL-92 the query mode for the current database (checked in Figure 55) and the default for new databases (not checked in Figure 55). By default, SQL89 is the default query mode for Access 2013 format databases.

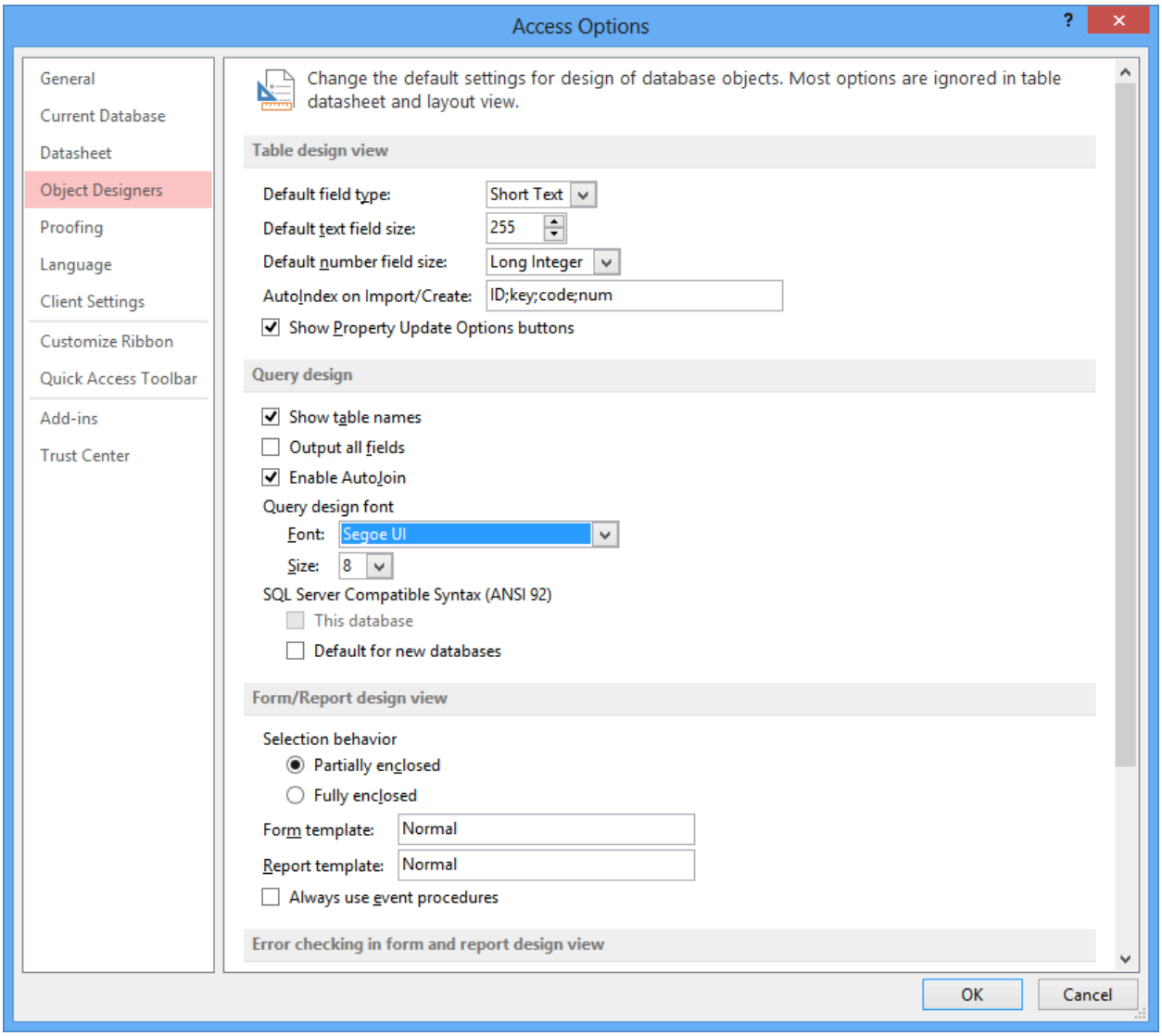


Figure 55: Advanced Tab of the Options Window Showing Query Mode Choices

Note that SQL-92 syntax can only be specified for the Access 2013, Access 2010 and Access 2003 file formats. If you convert to a previous Access format, some SQL statements may have syntax errors or produce incorrect results. Thus, you should not use SQL-92 query mode if you intend to convert a database to a previous Access format.

In practice, the SQL-92 query mode has little impact on data manipulation statements (SELECT, INSERT, UPDATE, and DELETE). However, it has a significant impact on data definition statements as presented in the following section. For SELECT statements, SQL-89 and SQL-92 seem to differ only on meta characters used with the LIKE operator and on duplicate field and alias names used in a query. Because using duplicate field and alias names is poor practice, the differences will not be presented here. You should be aware of the different meta characters used in SQL-89 and SQL-92, however. The two most widely used meta characters are the wildcard symbols for multiple characters (* in SQL-89 and % in SQL-92) and single characters (? in SQL-89 and _ in SQL-92). Examples 3.1 and 3.2 depict the SQL-89 and SQL-92 wildcard characters.

Example 3.1: **Using the meta character % in SQL-92 query mode**

Retrieve customers with a last name beginning with B. If this statement is executed in SQL89 mode, no records are returned because the meta character (%) has no special meaning.

```
SELECT *
 FROM Customer
 WHERE LastName LIKE 'B%'
```

Example 3.2: **Using the meta character * in SQL-89 query mode**

Retrieve customers with a last name beginning with B. If this statement is executed in SQL92 mode, no records are returned because the meta character (*) has no special meaning.

```
SELECT *
 FROM Customer
 WHERE LastName LIKE 'B*'
```

Access, like most relational DBMSs, supports a subset of the SQL standards with some proprietary extensions. For example, neither SQL-89 nor SQL-92 query modes supports the DISTINCT operator inside aggregate functions such as the COUNT function. As an example of a supported proprietary extension, Access provides the TRANSFORM statement to produce a pivot table used in data warehouse applications.

In addition to proprietary extensions, both SQL-89 and SQL-92 query modes support elements of the SQL:1999 standard. The most prominent part of the SQL:1999 standard supported is the use of a SELECT statement in the FROM clause as demonstrated in Example 3.3. Note that after you save the query, Access slightly varies the syntax of the nested query similar to the way in which the syntax is changed for the INSERT statement presented in Section 3.4.1.

Example 3.3: **Using a nested query inside the FROM clause**

Retrieve the list of vehicle serial numbers along with the count of unique parts repaired. The SELECT statement inside the FROM clause extracts the unique combinations of *Vehicle.SerialNo* and *PartNo*.

```
SELECT T1.SerialNo, COUNT(*) AS NumUniqueParts
 FROM
 (SELECT DISTINCT Vehicle.SerialNo, PartNo
   FROM RepairOrder, PartsUsed, Vehicle
   WHERE RepairOrder.OrdNo = PartsUsed.OrdNo
    AND Vehicle.SerialNo = RepairOrder.SerialNo) AS T1
 GROUP BY T1.SerialNo
```

Example 3.3 could be accomplished without a SELECT statement in the FROM clause if Access supported the DISTINCT keyword inside of the COUNT function. Without using a nested query in the FROM clause, two queries must be used as shown in Example 3.4. The second statement produces the list of serial numbers along with the count of the unique parts repaired using the name of the stored query in the FROM clause.

Example 3.4: **Using two queries instead of a nested query in the FROM clause**

Retrieve the list of vehicle serial numbers along with the count of unique parts repaired. The first SELECT statement, saved as “Temp3_4” produces the unique combinations of *Vehicle.SerialNo* and *PartNo*. The second SELECT statement uses the *Temp3_4* in the FROM clause to produce the result.

Temp3_4:

```
SELECT DISTINCT Vehicle.SerialNo, PartNo
 FROM RepairOrder, PartsUsed, Vehicle
 WHERE RepairOrder.OrdNo = PartsUsed.OrdNo
  AND Vehicle.SerialNo = RepairOrder.SerialNo

SELECT SerialNo, COUNT(*) AS NumUniqueParts
 FROM Temp3-4
 GROUP BY T1.SerialNo
```

3.6.2 Data Definition Statements

Although Access provides powerful and convenient tools for data definition, you can also use SQL data definition statements such as the CREATE TABLE statement. You can create a data definition query in the SQL window. You can also use the **Query Tools | Design → Data Definition** command in the Query Type group to open a data definition query in SQL view after you have created a new query. You can find documentation about the data definition statements in the subtopic "Create or modify tables or indexes" in the main topic "Designing Applications" in the table of contents of the Access help documentation. The details about SQL data types are not contained in this help documentation, a major omission if you want to write CREATE TABLE statements. The web page (http://msdn.microsoft.com/en-us/library/ff841694.aspx) is another source of documentation about data definition statements in Access SQL.

If you intend to use SQL data definition statements, SQL-92 query mode provides more coverage than SQL-89 mode. Table 1 summarizes the differences in data definition statements in SQL-89 and SQL-92 query modes. SQL-92 query mode supports most of the SQL data definition statements discussed in textbook chapters 3, 10, and 14.

The treatment of the CREATE VIEW statement in SQL-92 query mode is unconventional. The CREATE VIEW statement is covered in textbook Chapter 10. Access saves a view as an object in the list of queries. When a view is executed, Access creates another query containing the query part of the view definition. Example 3.5 shows a CREATE VIEW statement[5] for the query in QD7. Example 3.6 shows the result of executing the CREATE VIEW statement in Example 3.5. Example 3.7 shows a query using the view in Example 3.5. The net result is that the CREATE VIEW statement does not provide any benefits beyond just saving the query directly. The CREATE VIEW statement adds another object to the query list in addition to the query that would be saved anyway.

[5] Note that Example 3.5 will only execute successfully in SQL-92 query mode.

Table 1: Summary of Data Definition Support in SQL-89 and SQL-92 Query Modes

Statement	SQL-89	SQL-92
CREATE TABLE	No support for many features.	Added support for defaults, check constraints, cascading referential integrity, foreign keys, and custom AutoNumber seed and increment values.
ALTER TABLE	No support for altering column definition.	Added support for altering column definition.
CREATE/DROP PROCEDURE	No support	Supported
CREATE/DROP VIEW	No support	Supported
GRANT/REVOKE	No support	Supported
CREATE/DROP USER	No support	Supported
CREATE/DROP GROUP	No support	Supported
CREATE INDEX	No support	Supported

Example 3.5: **CREATE VIEW Statement for QD7**

Retrieve details from the *Customer*, *Vehicle*, and *RepairOrder* tables.

```
CREATE VIEW CustVehRepOrdView
AS
SELECT Customer.CustNo, Customer.FirstName,
  Customer.LastName, Vehicle.SerialNo, Vehicle.Year,
  Vehicle.Make, Vehicle.Model, RepairOrder.OrdNo,
  RepairOrder.TimeRecvd, RepairOrder.Odometer
 FROM (Customer INNER JOIN Vehicle
   ON Customer.CustNo = Vehicle.CustNo)
  INNER JOIN RepairOrder
   ON RepairOrder.SerialNo = Vehicle.SerialNo
```

Example 3.6: **Generated SQL Statement for CustVehRepOrdView**

This query is created after executing the CustVehRepOrdView. Access saves the query as CustVehRepOrdView.

```
SELECT Customer.CustNo, Customer.FirstName,
    Customer.LastName, Vehicle.SerialNo,
    Vehicle.Year, Vehicle.Make, Vehicle.Model,
    RepairOrder.OrdNo, RepairOrder.TimeRecvd,
    RepairOrder.Odometer
 FROM (Customer INNER JOIN Vehicle
    ON Customer.CustNo = Vehicle.CustNo)
  INNER JOIN RepairOrder
      ON Vehicle.SerialNo = RepairOrder.SerialNo
```

Example 3.7: **SQL SELECT Statement using CustVehRepOrdView**

This query uses the CustVehRepOrdView.

```
SELECT *
 FROM CustVehRepOrdView
 WHERE Make = "Toyota"
```

Closing Thoughts

Chapter 3 has provided guided instruction about query formulation in Microsoft Access 2013. Access provides two tools to create queries, the visual Query Design tool and the text-oriented SQL view. You learned how to use the Query Design tool to specify computed columns, row conditions, joins, and outer joins. You gained practice with the SQL view to formulate the kind of queries presented in textbook Chapters 4 and 9. To enhance usage of these tools, Access provides convenient ways to switch between them. You also saw how to make a subdatasheet open in a Query datasheet view. The subdatasheet feature allows a user to view additional details about related records. To provide insight about the SQL specifications supported by Access, the final section described the differences between the SQL-89 and the SQL-92 query modes supported in Access.

After completing this lab, you should be ready to formulate the queries in textbook Chapters 4 and 9. To enhance understanding of query formulation, you should use the tools presented in this chapter rather than just formulating queries on paper. By using the SQL view, you should also gain confidence about using SQL on DBMSs besides Microsoft Access. You will find that Query Design is particularly useful when formulating queries for data entry forms in the next two lab chapters. The emphasis in these kinds of queries is on the tables needed and how to combine the tables. The visual nature of Query Design allows you to focus on these aspects.

Chapter Reference

The chapter reference summarizes procedures that you practiced. For wizards discussed in the chapter, the procedures highlight important parts of the wizards but do not list all of the steps.

Procedure 1: Using Query Design (Section 3.1.1)

44. Choose **Create → Query Design** in the Queries group to open the New Query window.
45. The Query Design view appears with the Show Table window on top of it. Select tables to include in the new query.
46. To add fields in the query grid, you can use one of the following techniques:
    - Drag and drop field names into the query grid.
    - Click in the first row of an empty column and a small gray arrow appears. Click the arrow and select from the list of field names.
47. If the query grid is full, use the scroll bar to reveal other fields.

Procedure 2: Using the SQL Window (Section 3.1.2)

1. Queries created in Query Design also can be viewed in SQL view. You can toggle to the SQL window by one of these methods:
    - Clicking **View → SQL View**.
    - Clicking on the right mouse button (right-click) when the mouse is over the blue window frame in any view. A shortcut menu appears showing the remaining two view choices.
2. While in SQL view, type the SQL statement. Unfortunately, you do not have the use of the expression builder in the SQL window. Thus, you must type expressions without the help of the expression builder.
3. To use the SQL view for a new query, you must first start a new query in Query Design. Close the Show Table window and toggle to SQL view.

Procedure 3: Copy and Paste an Existing Query (Section 3.2)

48. Copy and paste is a good way to begin a new query that is similar to an existing query.
49. In the **Queries** part of the navigation pane, select the query to be copied. From the ribbon's **Home** tab (or by pointing the mouse on the selected query and clicking the right mouse button), select **Copy.**
50. Then again from the ribbon's **Home** tab select **Paste** (or by pointing the mouse under the selected query and clicking the right mouse button, select **Paste**). Type a name for the new query.

Procedure 4: Using the Expression Builder with Query Design (Section 3.2.2)

51. Create a new query or open an existing query in design view.
52. To use the expression builder, the query first must be named and saved.
53. In the query grid, position the mouse in the appropiate field and click the right mouse

button to reveal a shortcut menu.

54. Click on **Build** and the Expression Builder window appears with the field name in the text area.
55. Type the required expression into the Expression Builder window. When you are finished, click **OK**. If you scroll in the cell, you can see the entire expression that you typed.
56. To create a name for a computed field, type "FieldName:" before the expression where "FieldName" is the name you give to the computed field.

Procedure 5: Setting a Property for a Query Field (Section 3.2.2)

57. You can open the Field Properties window using one of the following techniques:
    - Click in the field (in the query grid) to select it and click **Design → Property Sheet**.
    - Position the mouse in the field and right-mouse-click to open a shortcut menu and select **Properties**.
58. Set the desired properties such as the *Format* property.

Procedure 6: Creating a Join Query (Section 3.2.4)

59. Create a new query.
60. Select two or more tables from the Show Table window.
61. If the tables have a relationship, a line appears connecting the two tables. Double-click on this line to reveal the Join Properties window. A join operator should connect these tables. Confirm that the first choice (join) is selected and click **OK**.
62. To add another table, click **Design → Show Table**. When the Show Table window appears, add the desired table (or tables) and repeat the step above.

Procedure 7: Creating an Outer Join Query (Section 3.2.6)

63. The join operator (as described in textbook Chapters 3 and 4) excludes nonmatching records. The outer join operator includes both matching and nonmatching records.
64. Select two or more tables when the Show Table window appears.
65. If the tables have a relationship, a line appears connecting the two tables. Double-click on the line connecting the tables to open the Join Properties window. Select the second or the third choice. This choice connects the tables with a one-sided join. An arrow from one table points to the other table indicating a one-sided outer join.

Additional Practice

The following problems provide additional practice with Query Design and the SQL window. Parts 1 and 2 use the auto repair order database extended with the *Labor* and the *RepairLabor* tables (see the practice problems of Chapter 2). After formulating a query with one of the tools, toggle to see its representation in the other tool.

Part 1: Query Design

1. List all of the columns of the *Labor* table in which the hourly rate is greater than $20.
2. List the labor description, the standard time, and the hourly rate of the *Labor* table in which the labor description begins with "I". Sort the result in ascending order by standard time and hourly rate.
3. List the labor description, the standard time, and the hourly rate of the *Labor* table in which the hourly rate is greater than $20 and the standard time is less than 0.3 hour or the hourly rate is less than $25 and the standard time is greater than 0.4 hour.
4. List the labor code, the labor description, the standard time (from the *Labor* table), and the actual hours (from the *RepairLabor* table). Include a row in the result if its hourly rate is less than $25.
5. List the labor code, the labor description, and the difference between the labor standard time (from the *Labor* table) and the actual hours (from the *RepairLabor* table). Include a row in the result if the difference is greater than 0.2 hour.
6. List the repair order number, the labor code, the labor description, the time received, and the labor cost (product of the actual labor hours times the hourly rate). Include a row in the result if the time received is in October 2010. In the query result, format the labor cost as currency.

Part 2: SQL View

1. List all of the columns from the *Labor* table in which the standard time is greater than or equal to 0.5 hour.
2. List the order number, the labor code, the labor description, and the labor amount (repair labor hours times hourly rate of labor). Include a record in the result if its labor amount is greater than $30. Rename the labor amount expression as "LaborAmt" in the result.
3. Perform problem (2) using a different join method (see textbook Chapter 9).
4. For each repair order in October 2010, list the count of the labor repairs. The result should include the order number, the date that the repair was received (not the date and time), and the number of labor repairs. (Hint: you need to use a function to list the date without the time.)
5. For each repair order in October 2010, list the count of the labor repairs and the sum of the labor amount. The labor amount is computed as the repair labor hours times the hourly rate of labor. Only include a row in the result if the sum of its labor amount is greater than $55. Rename the computed result column. The result should include the order number, the date that the repair was received (not the date and time), the number of labor repairs, and the sum of the labor amount.

Part 3: University Database Examples

Try the examples in textbook Chapters 4 and 9 on the university database from textbook Chapter 10. You may need to convert some of these examples to the textbook Chapter 10 database. The university database in textbook Chapter 3 differs slightly from the university database in textbook Chapter 10.

Part 4: Order Entry Database Problems

Try the problems in textbook Chapters 4, 9, and 10 (problems 10.1 to 10.21) on the extended order entry database from textbook Chapter 10. For each problem, decide whether SQL or Query Design is an easier tool. You may need to convert some of these problems in textbook Chapters 4 and 9 to the textbook

Chapter 10 database. The order entry database in textbook Chapter 4 differs slightly from the extended order entry database in textbook Chapter 10.

Chapter 3: Query Formulation Lab

Learning Objectives

This chapter enables you to gain practical experience building queries utilizing the Query Design and SQL tools of Access 2013. After this chapter, you should have acquired the knowledge and skills to

- Use the Query Design window to create queries.
- Create queries in SQL view.
- Use the Query Wizard and the expression builder.
- Toggle between query views.
- Write queries to summarize data.
- Revise an existing query.
- Recognize and write parameter queries.
- Understand the Append, Update, Delete, and Union queries.
- Gain insight into the standards compliance of Access SQL

Overview

In Chapter 1, you became familiar with Access databases. You learned about the tools to create Access objects and object properties. Chapter 2 enabled you to put these concepts into practice by creating the auto repair database. To begin developing applications using the auto repair database, this chapter describes the tools available to create queries. In addition, this chapter complements the conceptual background in textbook Chapter 4 about using SQL statements to create queries.

This chapter demonstrates two tools to formulate queries in Access. The first part of this chapter gives you practice using the Query Design window. Query Design is a visual tool that allows you to create queries without writing much code. In addition to learning about various aspects of Query Design, you will practice using the Simple Query Wizard and the expression builder. The second part of this chapter demonstrates the SQL window, a text-based tool. Because SQL is the industry standard language, you should gain practice using the SQL window. In addition, some queries can be formulated more easily in SQL. You also will learn to toggle between SQL and Query Design to obtain the best of both tools. When you finish this chapter, you should be ready to create your own queries using both Query Design and SQL.

3.1 Tools to Create Queries

Access provides two ways to formulate queries. Query Design provides a visual way to formulate queries. The SQL window allows you to write SQL statements using the syntax described in textbook Chapter 4. This section uses a simple query to demonstrate the basics of both Query Design and the SQL window.

When you want to create a new query, you must start in Query Design. You can formulate the query by selecting tables and fields, joining tables, and specifying conditions to restrict query results. You then may switch to SQL view to modify the SQL statement for the query created in Query Design.

> Technical Note: Alternatively, you can switch immediately to SQL view to write the entire query. This section introduces you to both tools of query formulation and toggling between the tools.

3.1.1 Creating a Query in Query Design

The first query is a simple one that involves only one table, the *Customer* table. The purpose of this query is to make a phone list to remind existing customers of their vehicle's next service. Therefore, we are going to ask the query to give us a list of customers' first name, last name, and phone number:

3. Open the Auto Repair Database: Start Access and open the auto repair database (AutoRepair.accdb) that you created in Chapter 2. The navigation pane should show the tables that you created in Chapter 2 (Figure 1). Choose **Create → Query Design** in the Queries group to open Query Design in the view pane (Figure 2).

4. Select a Table/Query: Query Design appears in the view pane with the Show Table window on top of it (Figure 2). You must select a table or existing query to include in the new query. Select the *Customer* table, click the **Add** button, and then click the **Close** button. Figure 3 shows the Query Design pane containing the *Customer* table.

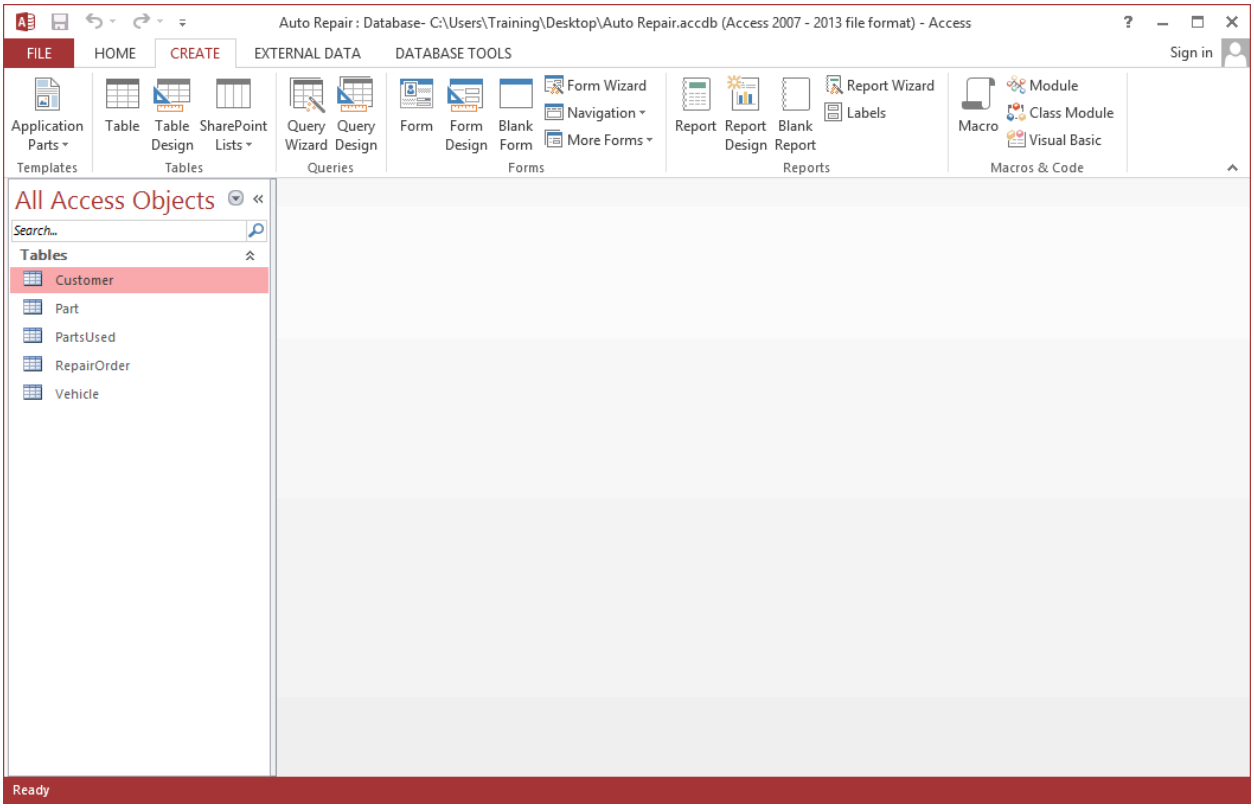


Figure 1: Navigation Pane with Empty View Pane

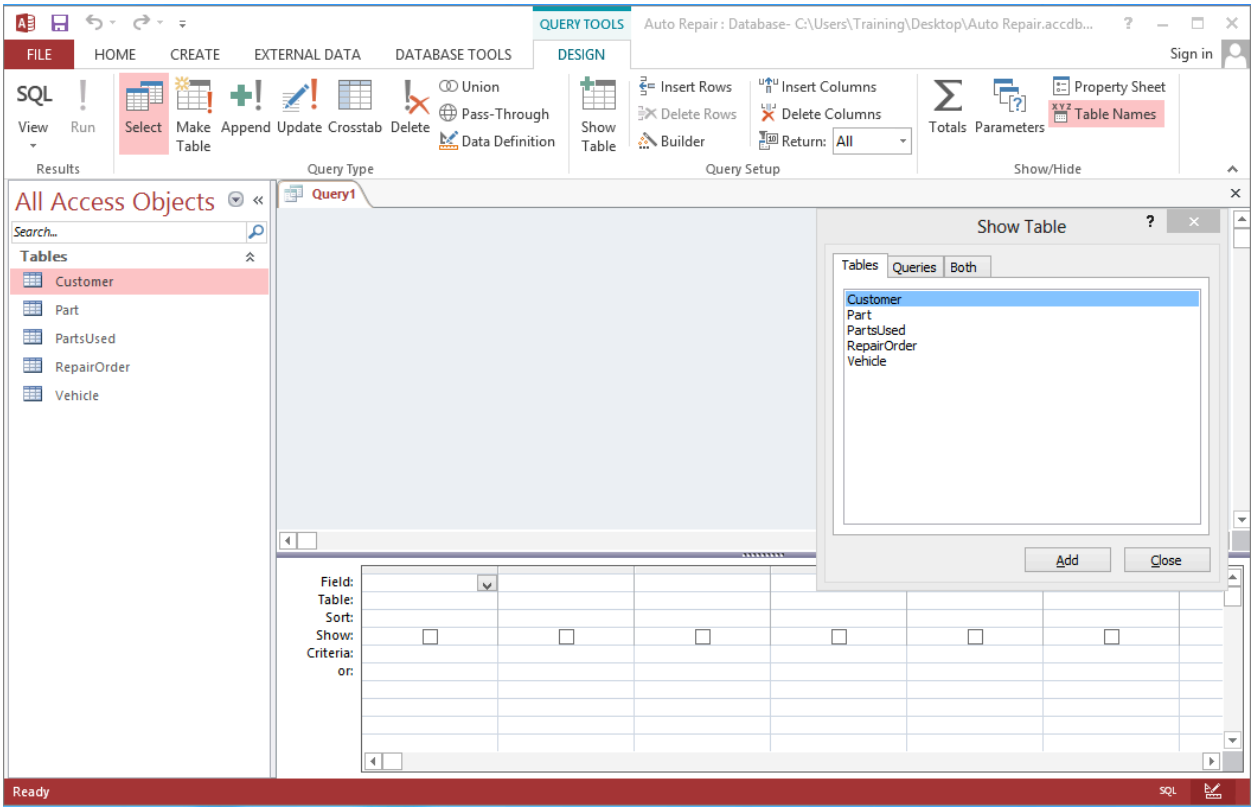


Figure 2: Query Design Pane and Show Table Window

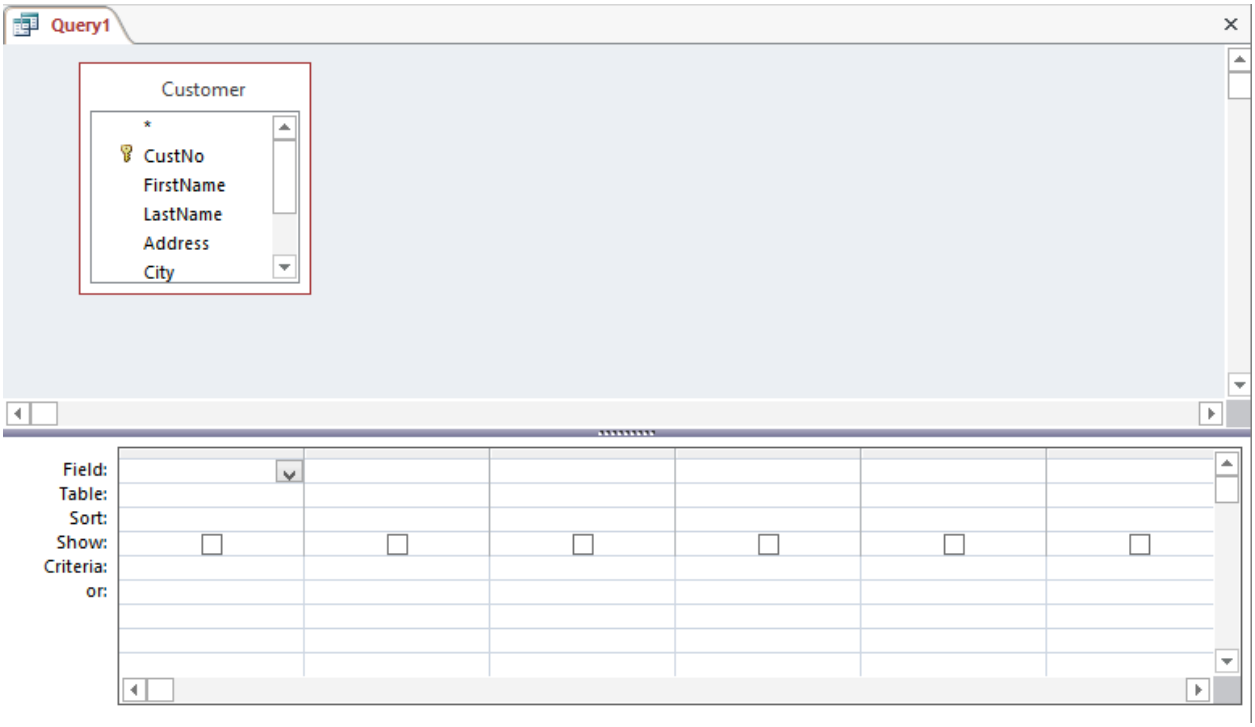


Figure 3: Query Design Area with *Customer* Table

6. Drag Fields into the New Query: To obtain query results, you must put some *Customer* field names into the empty field spaces of the query table. Drag and drop[6] the *CustNo* field name from the *Customer* table to the first column and row in the empty query table. After releasing the mouse button, the field name appears in the grid. Repeat the process to place the *FirstName* field in the next empty column of the first row (see Figure 4).

7. Use a Different Field Selection Technique: To place the next field, *LastName*, use an alternative method. This time, just click into the next empty column of the first row and a small gray arrow appears (see Figure 4). Click the arrow and a list of the *Customer* fields appears. Select *LastName* from the list.

7. Select the Last Field: Select the *PhoneNo* field using the technique demonstrated in step 5. Note that you have to move the small scroll bar down to reveal the *PhoneNo* field. After you have finished, your Query Design window should appear as in Figure 5.

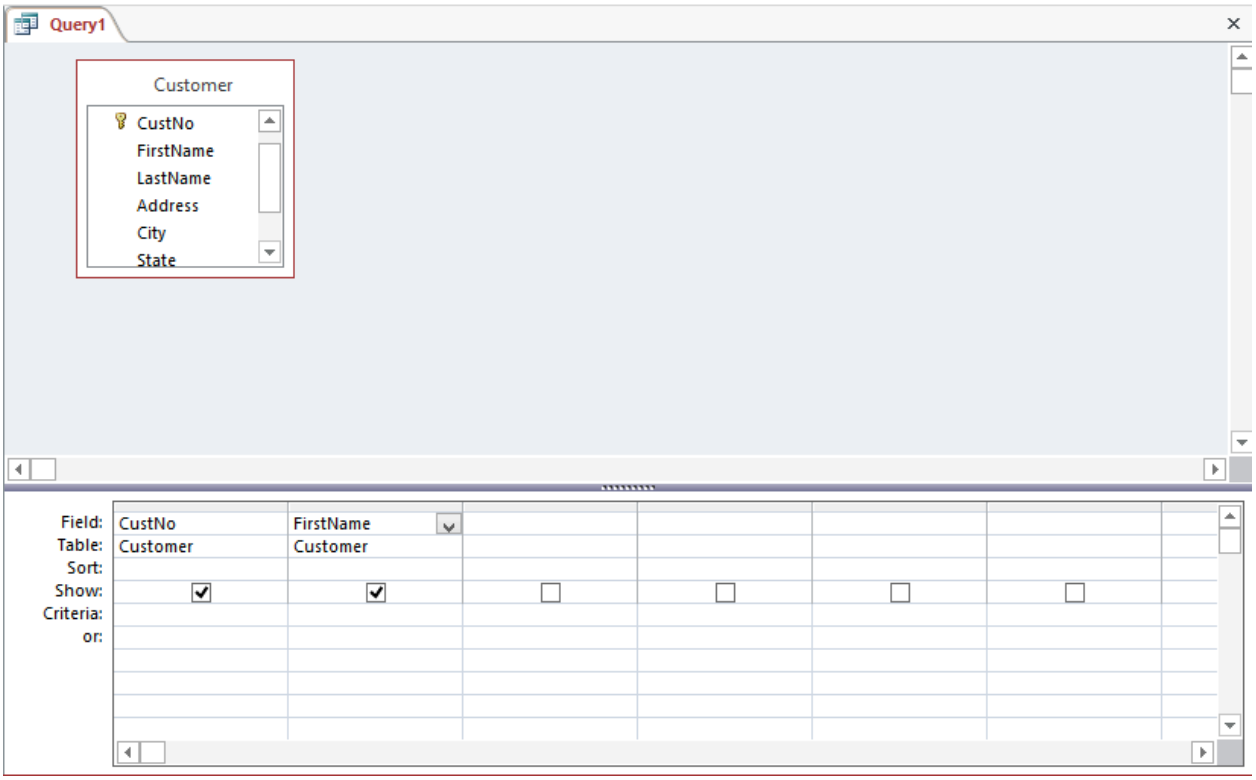


Figure 4: First Two Fields in Query Design View

[6] Drag and drop means that you hold the mouse down while moving it to another part of the window. After the item has been moved, you release the mouse button.

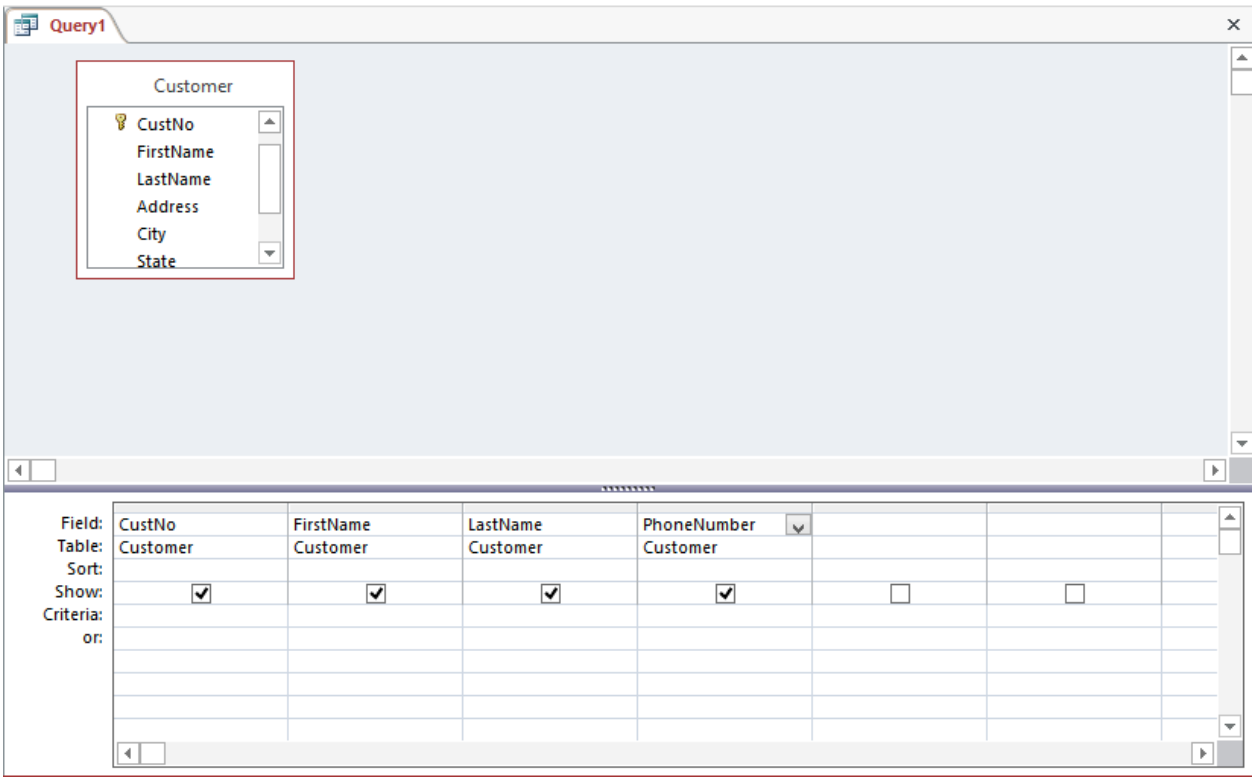


Figure 5: Completed Query Design View

3.1.2 SQL View

In the second part of this chapter you will create queries directly in the SQL view. At this point, it is important to know that queries created in Query Design may also be viewed in the SQL view. To demonstrate switching between query views, the following steps demonstrate ways to access the SQL window as well as to allow you to save your query.

2. Accessing the SQL Window: With the Query Design view open, click **View → SQL View** (see Figure 6) in the **Design** tab of the Ribbon. The SQL view appears (Figure 7) in the view pane.

> Technical Note: You also may right-click the mouse in the blue Query Design window pane. Then select **SQL View** from the shortcut menu that appears.

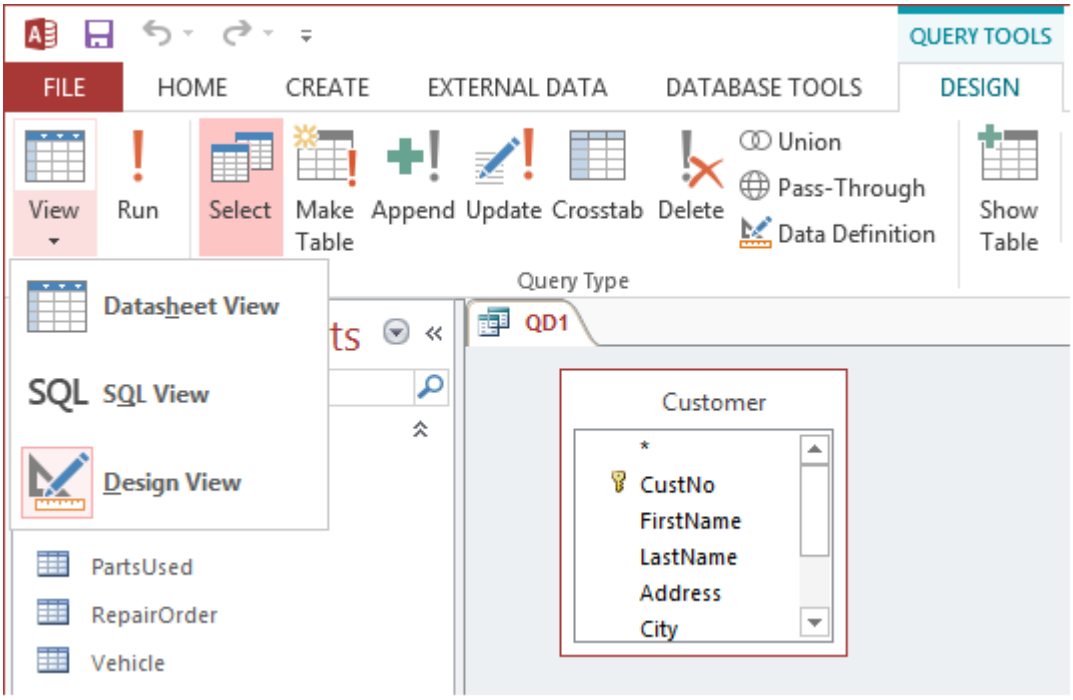


Figure 6: View Menu Showing the SQL View Command

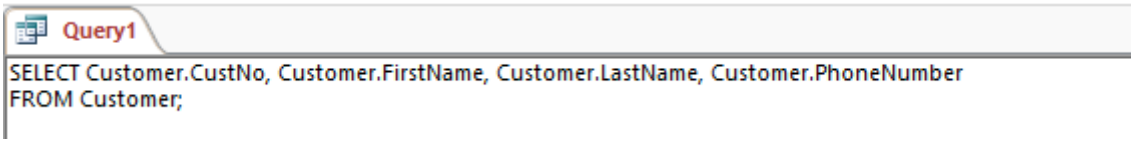


Figure 7: SQL View for "QD1"

4. Close and Save the Query: Close the SQL View pane by clicking the **Close** button (**X** button at the top of the window). Click **Yes** to save the query.

5. Name the Query: Type "QD1" as the name of the query in the next dialog box. Note that "QD" stands for "Query Design". Click the **OK** button to finish. Your first query now appears in the Queries section of your navigation pane.

3.1.3 Execute the Query

As you know, the purpose of a query is to display the answer to a question required by a database user. When running or executing a query, the results are displayed as a datasheet. Now that the "QD1" query is completed, you can run it to see if you obtain the expected results. A query may be executed from either the SQL or the Query Design view:

4. Open the Query Design View or the SQL View: If you are using the standard All Tables view, then the query appears under the table that it uses. You can launch the query at any time by double-clicking it. To open the query in Query Design view, right-click it in the navigation pane, and then choose **Design View**.

5. Execute/Run the Query: Next, choose **Query Tools Design → Run** from the Results group in the Ribbon (Figure 8).

6. View the Result: The datasheet appears with the phone list of the auto repair shop customers (Figure 9). Close the datasheet to return to the navigation pane.

In addition to executing the query per the previous steps, the datasheet also may be accessed by toggling to the datasheet view. Refer to the next section to learn how to toggle between query views.

Figure 8: Ribbon Showing the Results Group with the **!Run** Item

QD1

CustNo	FirstName	LastName	PhoneNumber
1	Beth	Taylor	(206) 221-9021
2	Betty	Wise	(206) 445-6982
3	Bob	Mann	(206) 326-1234
4	Candy	Kendall	(206) 523-1112
5	Harry	Sanders	(360) 444-0092
6	Helen	Sibley	(206) 624-0362
7	Homer	Wells	(206) 524-1461
8	Jerry	Wyatt	(206) 524-8145
9	Jim	Glussman	(206) 445-2139
10	Larry	Styles	(425) 745-9980
11	Mike	Boren	(360) 444-5678
12	Ron	Thompson	(360) 747-2222
13	Sharon	Johnson	(360) 333-6666
14	Sheri	Gordon	(206) 525-3344
15	Todd	Hayes	(206) 775-7689
16	Wally	Jones	(206) 523-9957
*	(New)		

Record: 1 of 16 No Filter Search

Figure 9: "QD1" Datasheet

3.1.4 Toggling between Views

You have just learned about the three query views: design view, SQL view, and datasheet view. For convenience, Access has a number of ways to navigate between these views. You can navigate among views whenever a query is open in the view pane. The different ways to switch between the three views are explained in the following steps:

3. Use the query tools: With Query Design view open, you will see Query Tools in the ribbon. Choose **Query Tools | Design → View** in the Results group (Figure 6) and the three view choices appear.

4. View Bar: When a query view appears in the view pane, the view bar 

appears in the lower right corner underneath the view pane. You can select a button in the view bar to change views. You can select design view (), datasheet view (), or SQL view (

).

Technical Note: You also can switch to other views by clicking on the right mouse button (right-click) when the mouse is over the blue window frame in any view. A shortcut menu appears showing the remaining two view choices.

3.1.5 Using the Simple Query Wizard

Another useful tool for creating queries is the Simple Query Wizard. The Simple Query Wizard guides you using a sequence of windows to create a query. Follow the steps below to use the wizard to create another phone list. You will create a similar query as before, except that you will use the Query Wizard and add a condition to limit the query result to customers in Seattle.

5. Open the New Query Window: Choose **Create → Query Wizard** in the Queries group to open the New Query window (Figure 10). Click the **OK** button with the Simple Query Wizard highlighted.
6. Select a Table/Query: In the initial window (Figure 11), you first need to select from the Tables/Queries list. The Simple Query Wizard uses the selected table/query as a starting point in your new query. The default is always the previous query. Since you are basing this query on the *Customer* table as in the previous query, select *Customer*. Notice that when you make a selection, the list of available fields in the window below changes accordingly.
7. Select Fields to Include: Click the **>** button to move the fields shown in Figure 12 to the right. Select the fields shown in Figure 12 and click **Next**.
8. Finish the Query Wizard: In the final wizard window (Figure 13), name the query "QD2". Below you are asked if you want to open the query or modify its design. Choose the second option, "Modify the query design.", and click **Finish.** The design view of "QD2" (Figure 14) appears ready for you to modify.

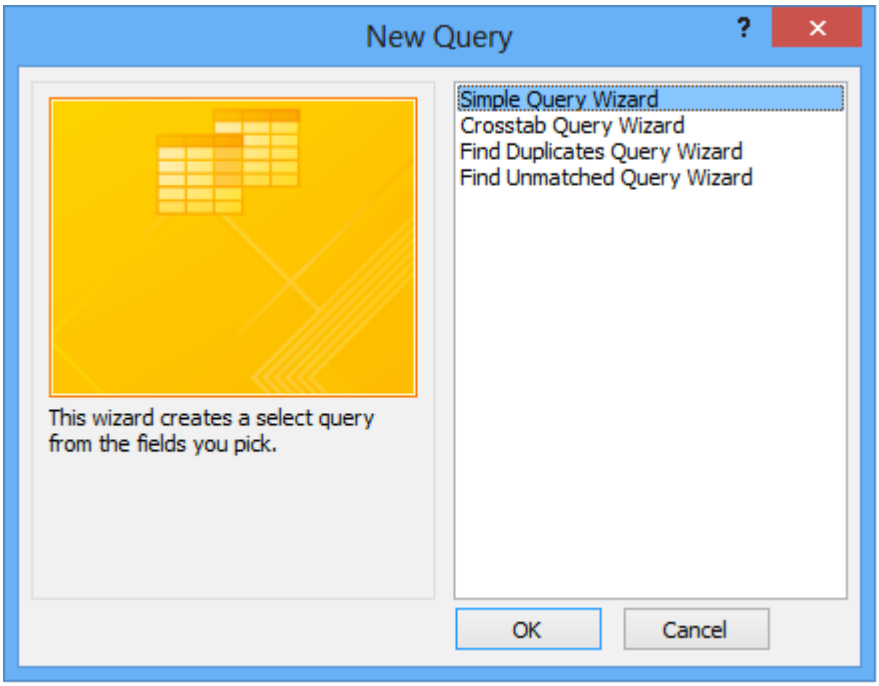


Figure 10: New Query Window of the Simple Query Wizard

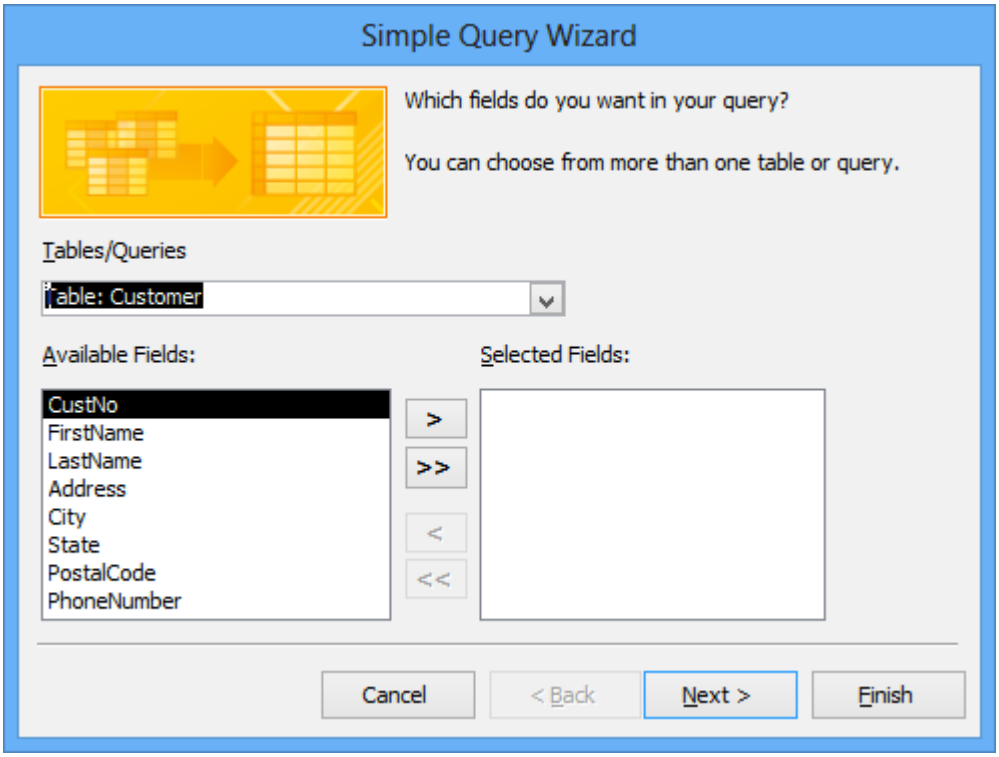


Figure 11: Initial Window of the Simple Query Wizard

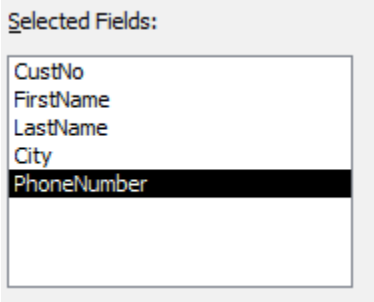


Figure 12: Fields Selected for the New Query

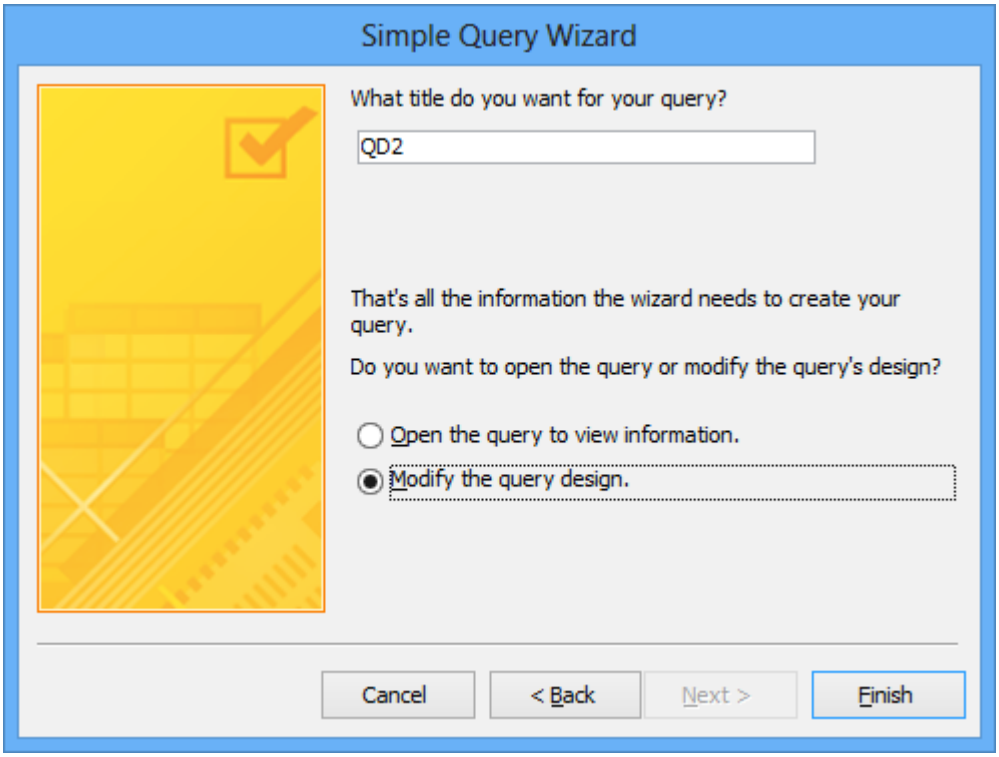


Figure 13: Final Wizard Window

7. Enter a Condition: Next you will insert a condition to limit the list to only customers in Seattle. To accomplish this task, type in the *City* column and the *Criteria* row "Seattle". Be sure to enclose it in double quotes (see Figure 15).

8. Execute the Query: Now execute the query by clicking on the **!Run** button on the ribbon. The datasheet appears as in Figure 16. Notice that the datasheet shows an asterisk (*) in the last row. The asterisk means that the query is "updatable". For now, you can ignore the last row if it contains an asterisk. Updatability is an important concept for data entry forms. See textbook Chapter 10 for view updatability concepts and rules. Close the datasheet when you are finished viewing it.

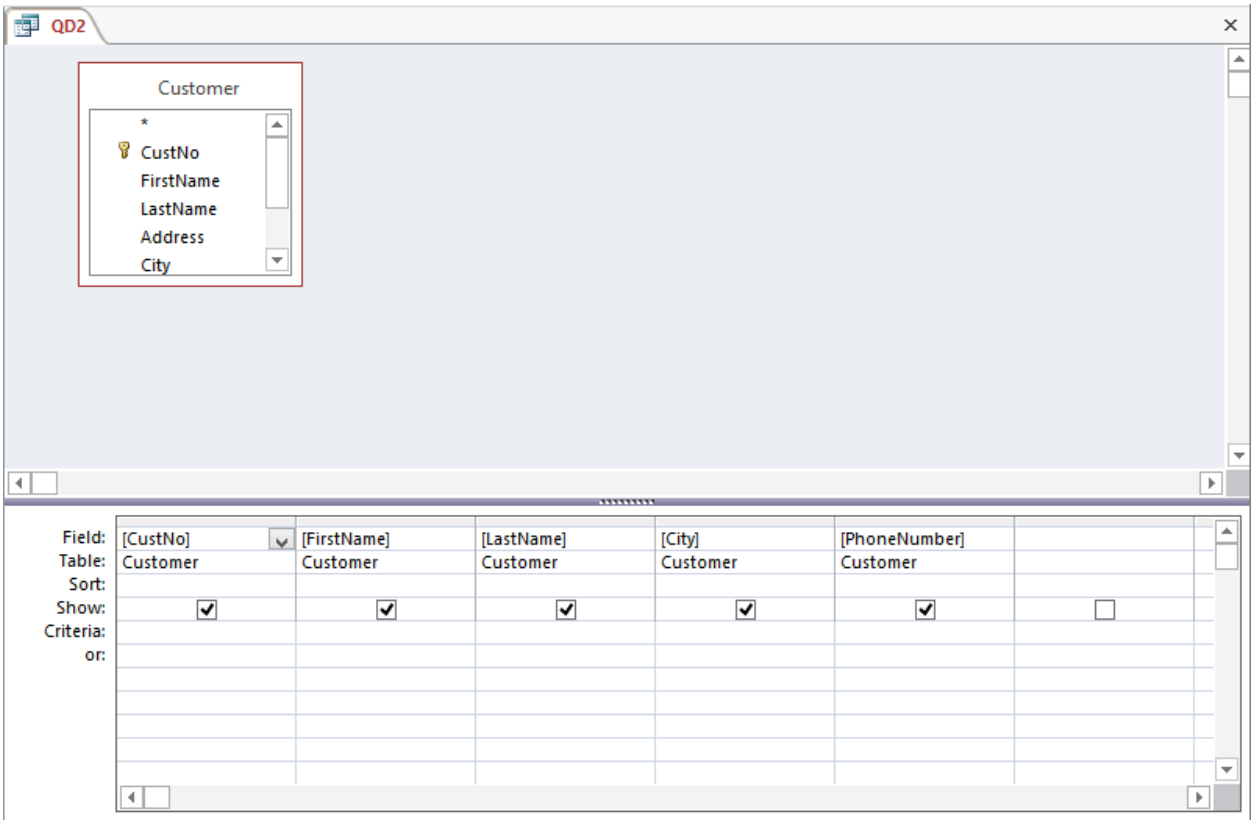


Figure 14: Query Design Window of “QD2”

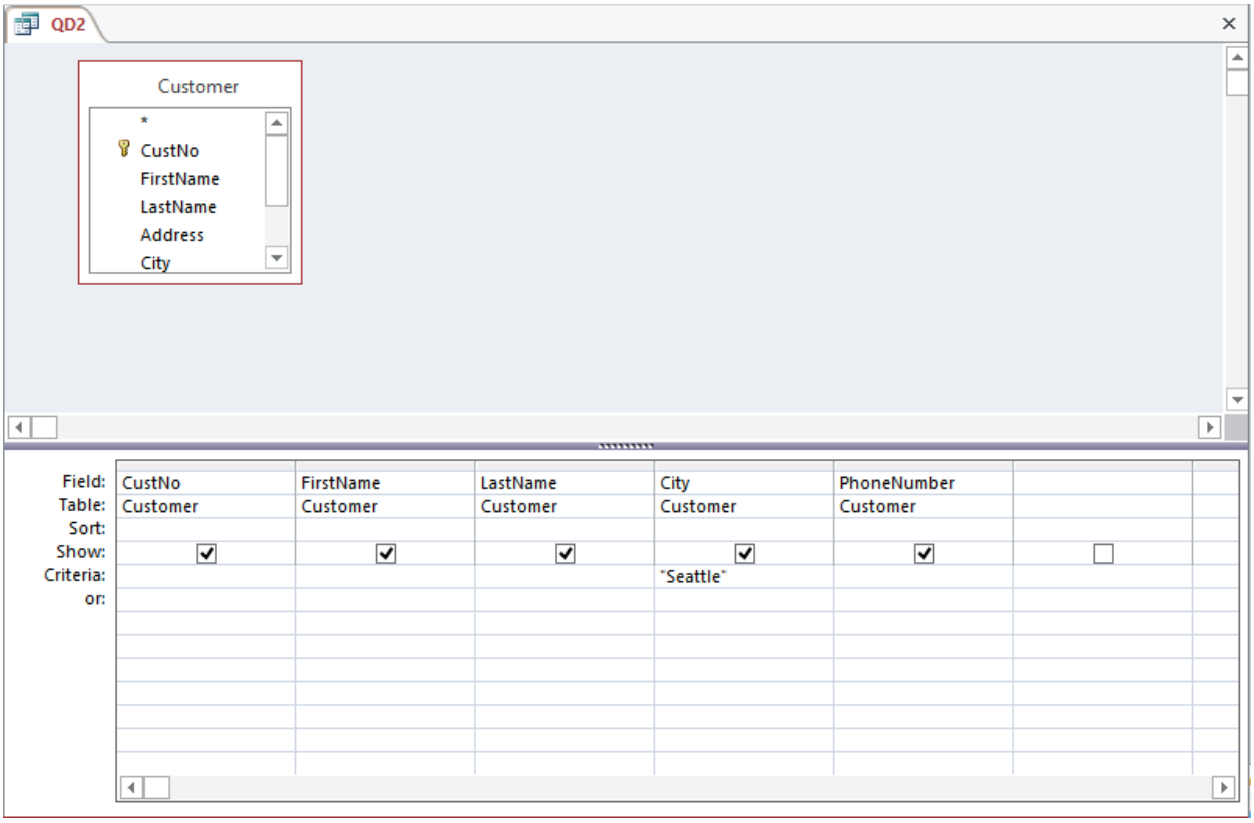


Figure 15: "QD2" with the "Seattle" Criterion

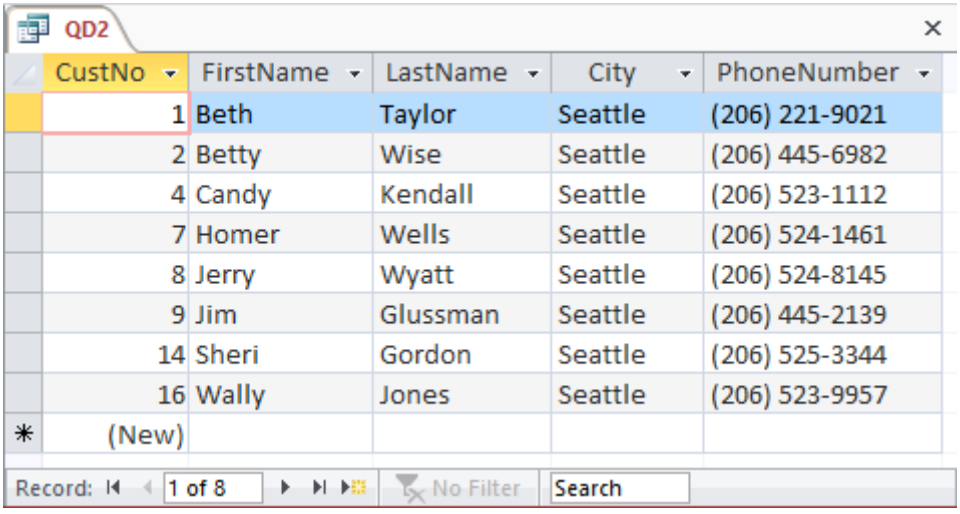


Figure 16: Datasheet of "QD2" with the "Seattle" Criterion

3.2 Creating Additional Queries in Design View

In the previous section you created a simple query in design view and saw how it appeared in SQL view and datasheet view. In this section you will gain additional practice with design view using various tools to help formulate more advanced queries. Remember that you still may toggle between design view and SQL view to examine your query.

3.2.1 Query with More than One Condition

The last query you created ("QD2") had a single condition to limit the result of the query to only customers residing in Seattle. For this new query, your list also will include customers who reside in the city of Renton. To accomplish this, you will copy and paste the query "QD2" and add a second condition to the query. The revised query will allow the phone list to include customers residing in the cities Seattle and Renton.

5. Copy and Paste an Existing Query: In the navigation pane, select the "QD2" query. From the ribbon (or point the mouse on the highlighted "QD2" and right-click), select **Copy.** Then again from the ribbon select **Paste** (or by pointing the mouse under the highlighted "QD2" and right-clicking, select **Paste**). Type "QD3" as the name of the new query when prompted.

6. Open the Query Design Window: Right-click the query in the navigation pane. The design view of "QD3" appears on the screen containing the copied "QD2" query ready for you to change (refer back to Figure 14).

7. Enter a Condition: Type the city name of "Renton" directly under "Seattle" in the *City* column (Figure 17). Adding a criterion in another row indicates an OR connection among the conditions. Thus, the query includes customers from either the city of Seattle or the city of Renton.

8. Execute the Query: Now, execute the query to be sure it is correct (Figure 18). After viewing it, close the Datasheet window and save changes when prompted.

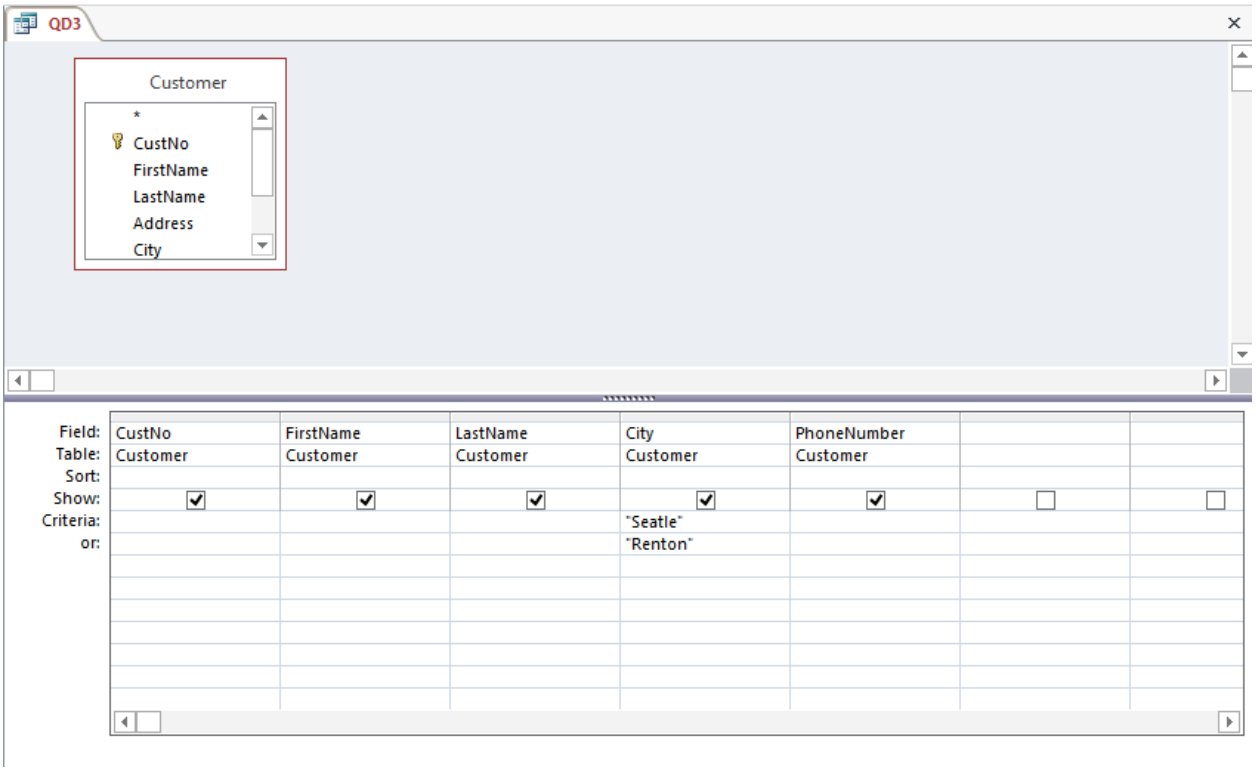


Figure 17: "QD3" with the "Seattle" OR "Renton" Criteria

QD3

CustNo	FirstName	LastName	City	PhoneNumber
1	Beth	Taylor	Seattle	(206) 221-9021
2	Betty	Wise	Seattle	(206) 445-6982
4	Candy	Kendall	Seattle	(206) 523-1112
6	Helen	Sibley	Renton	(206) 624-0362
7	Homer	Wells	Seattle	(206) 524-1461
8	Jerry	Wyatt	Seattle	(206) 524-8145
9	Jim	Glussman	Seattle	(206) 445-2139
12	Ron	Thompson	Renton	(360) 747-2222
14	Sheri	Gordon	Seattle	(206) 525-3344
16	Wally	Jones	Seattle	(206) 523-9957
*	(New)			

Record: 1 of 10 No Filter Search

Figure 18: Datasheet of "QD3" with the "Seattle" OR "Renton" Criteria

3.2.2 Using the Expression Builder with Query Design

The next query uses the *Part* table. The auto repair shop needs a parts list to access the impact of price changes. The *Part* table that you created in Chapter 2 contains the current price, not an inflated price. To compute an inflated price, you will use the expression builder to type an expression. In addition, you will set the *Format* property to make the field display as a monetary value.

4. Create a New Query: Choose **Create → Query Design** from the Queries group to open the New Query window.

5. Select the *Part* Table: When the Show Table window appears, select the *Part* table, click **Add,** and then click **Close**.

6. Select Fields: In the Query Design window, drag and drop all five fields from the *Part* table to fill the query table (see Figure 19).

Field:	PartNo	PartDesc	UnitsinStock	UnitPrice	UnitSize	
Table:	Part	Part	Part	Part	Part	
Sort:						
Show:	☑	☑	☑	☑	☑	☐
Criteria:						
or:						

Figure 19: Lower Pane of the Completed Query Design View

7. Using the Expression Builder: Now, you need to change the name of the *UnitPrice* field to *InflatedPrice* and type an expression to inflate the price by 10 percent. You will access the expression builder to accomplish these tasks. However, for the expression builder to read the fields in the query, the query must first be named and saved. So, on the toolbar, click **Save**. When asked to name the query, type "QD4" and click **OK**.

8. Open the Expression Builder: Next, position the mouse in the *UnitPrice* field and click the right mouse button to reveal a shortcut menu. Click on **Build…** and the Expression Builder window appears with the field name *UnitPrice* in the text area.

9. Type into the Text Area: Type the expression as shown in the text area in Figure 20:

InflatedPrice: [UnitPrice]*1.1

When you are finished, click **OK**. Your *UnitPrice* field in design view should now appear as Figure 21. If you scroll in the cell, you can see the entire expression that you typed.

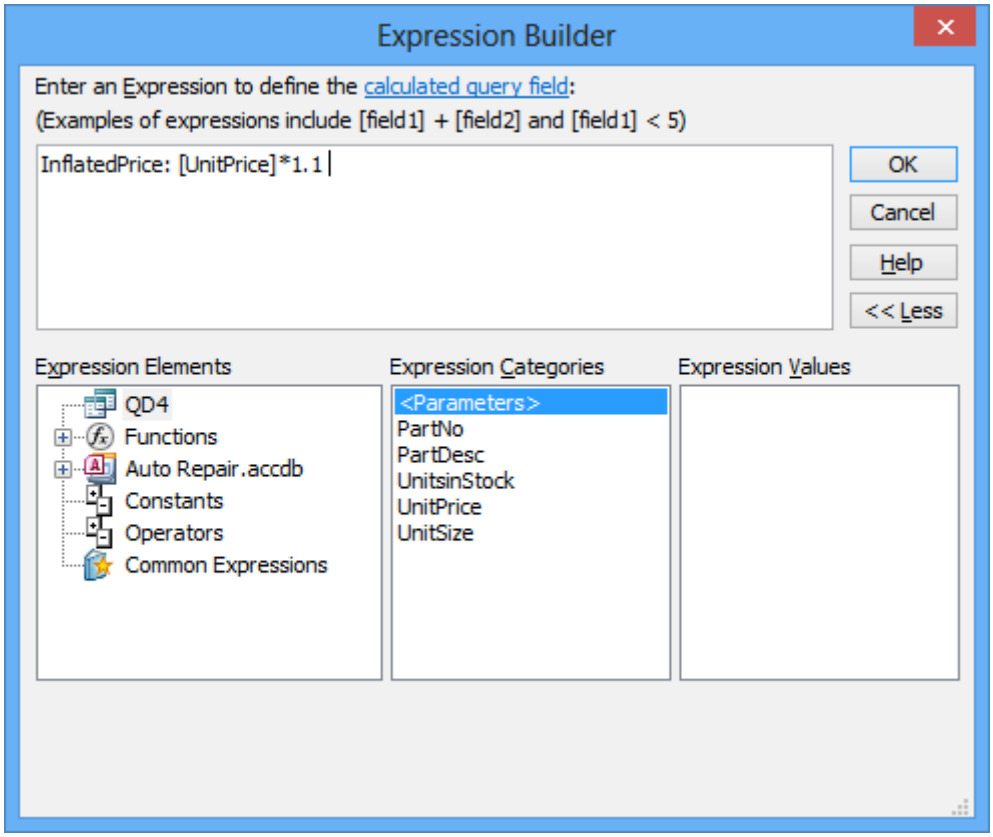


Figure 20: Expression Builder for "QD4"

10. Open the Field Properties Window: Next, you will set a query field property. To open the property sheet for a query field, position the mouse in the *UnitPrice* field and click **Query Tools | Design → Property Sheet** from the Show/Hide group. The Property Sheet for Field Properties appears on your screen (Figure 22).

> Technical Note: Alternatively, you can position the mouse in the *InflatedPrice* field and right-click to reveal a shortcut menu and select **Properties**.

11. Set the *Format* Property: Click the mouse into the text area of the *Format* property to reveal a small arrow. Click the arrow to reveal a list of properties. Scroll down and select the "Currency" value. Click the **Close** button to close the Properties window. Set the Format property for the InflatedPrice file (computed field) to currency just like UnitPrice field.

12. Execute the Query: Execute the query to be sure it is correct. Your datasheet should appear as in Figure 23. When you are finished viewing it, close it and save the changes when prompted.

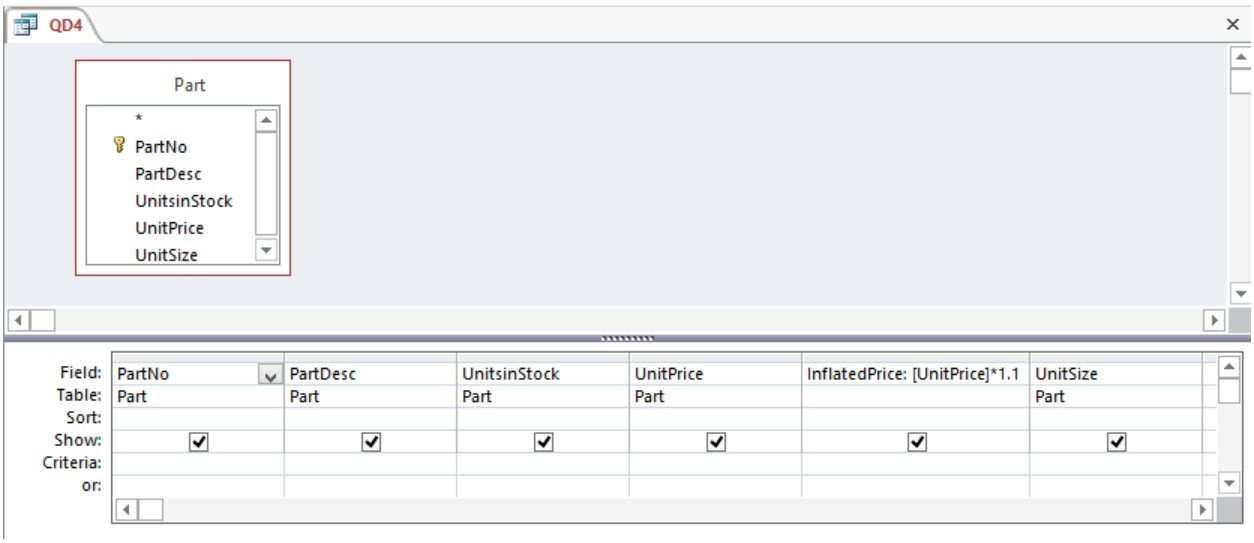


Figure 21: Completed "QD4"

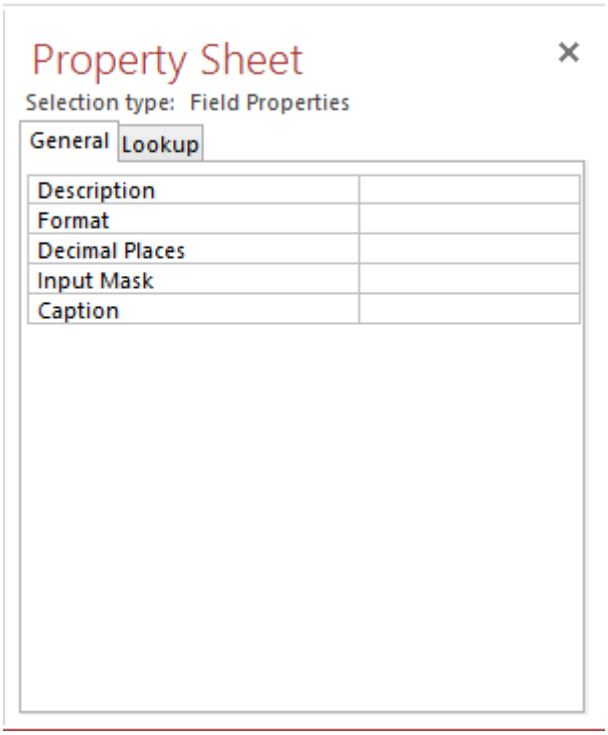


Figure 22: Field Property Sheet Window

QD4

PartNo	PartDesc	UnitsinStock	UnitPrice	InflatedPrice	UnitSize
1	10W-40 oil	145	$1.00	$1.10	quart
2	oil filter	14	$2.00	$2.20	item
3	AntiFreeze	10	$3.95	$4.35	quart
4	Spark Plugs	45	$0.99	$1.09	item
5	Transmission Fluid	15	$1.50	$1.65	quart
6	10W-30 oil	95	$0.95	$1.05	quart
7	Brake Lining	25	$5.00	$5.50	quart
8	Shock	75	$6.00	$6.60	item
9	Muffler	10	$15.00	$16.50	item
10	Tail Pipe	85	$4.00	$4.40	item
11	Head Gasket	32	$9.00	$9.90	dozen
12	Timing Chain	6	$22.50	$24.75	item
13	Battery	13	$55.00	$60.50	item
14	Radiator	3	$60.00	$66.00	item
15	Radiator Hose	24	$3.00	$3.30	dozen
16	Rotor	4	$16.00	$17.60	item
17	Tire	36	$46.00	$50.60	item
18	Headlight	6	$5.00	$5.50	case
19	Tail Light	7	$3.00	$3.30	dozen
20	GearBox	2	$25.00	$27.50	item
(New)					

Record: 1 of 20 No Filter Search

Figure 23: Datasheet of "QD4"

3.2.3 Connecting Conditions by AND

To make the parts list more useful, it should be restricted to frequently used parts. Since the auto repair shop performs many oil changes, restricting the list to the oil inventory is appropriate. In addition, the list should only display an item if the quantity in stock is greater than 10 so the inventory will not be depleted.

7. Copy and Paste an Exisitng Query: To begin, copy "QD4" and paste it as "QD5".

8. Open the Query Design Window: Next, open "QD5" in design view. It should appear on your screen the same as "QD4" (refer back to Figure 21).

9. Open the Expression Builder: You will use the expression builder to set the criteria for the parts list. Therefore, click in the *Criteria* row of the *PartDesc* field column to open the Expression Builder window.

10. Type into the Text Area: Type the expression as shown in the text area of Figure 24:

    LIKE "*oil*"

    When you are finished, click **OK**.

11. Set Criteria: Set the criteria for the number of units in stock. Click in the *Criteria* row of the *UnitsInStock* field and type the following as in Figure 25: `>10`

12. Execute the Query: When you are finished, execute the query to see if it is correct, then close and save changes when prompted. The datasheet should appear as in Figure 26.

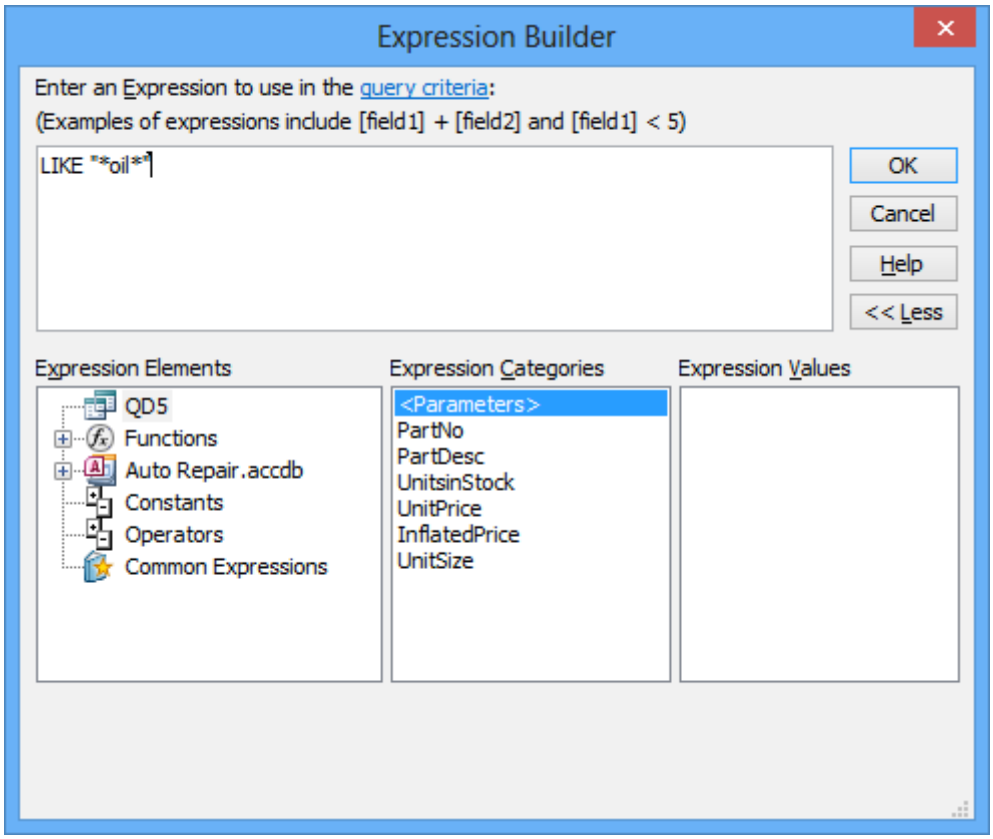


Figure 24: Expression Builder Window for "QD5"

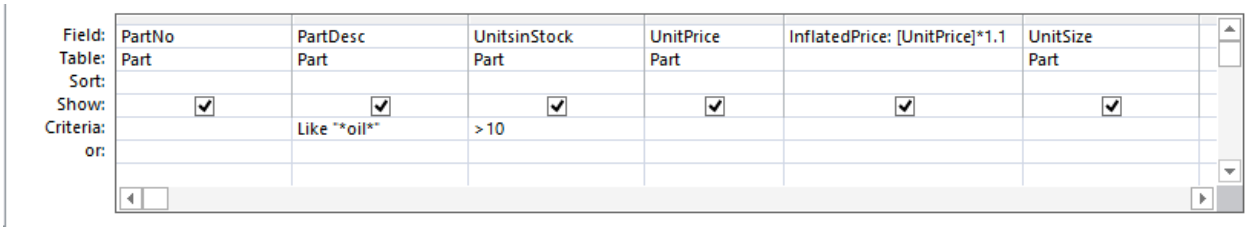


Figure 25: Completed "QD5"

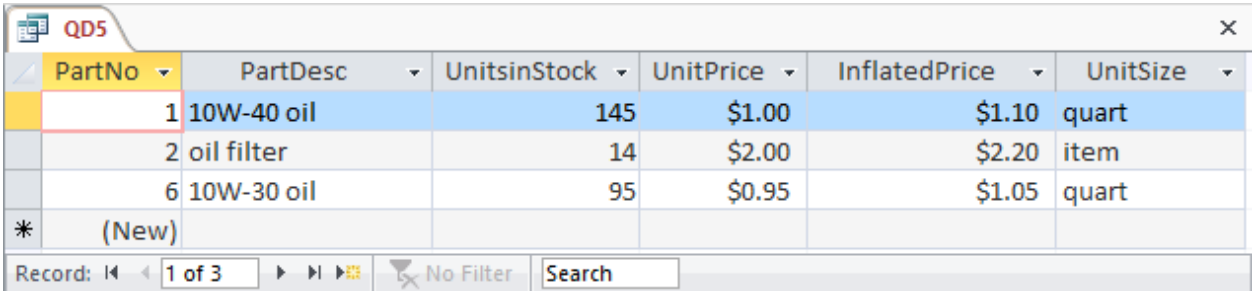


Figure 26: Datasheet of "QD5"

3.2.4 Creating a Join Query

Many customers who bring their vehicles into the auto repair shop own more than one vehicle. Therefore, it would be convenient for the employees to have a list of the customers and their vehicles. To accomplish this task, a query will be created to join the *Customer* and the *Vehicle* tables. Query Design makes the join connection automatically, as the following steps depict.

6. Create a New Query: Choose **Create → Query Design** from the Queries group to open the New Query window.
7. Select Two Tables: When the Show Table window appears, select the *Customer* and the *Vehicle* tables from the list by using the **Ctrl** key. Click the **Add** button to transfer both tables to the Query Design window. Click the **Close** button to finish.
8. View the Join Properties Window: A line appears connecting the two tables as they appeared in the Relationships window from Chapter 2.[7] Double-click on this line to reveal the Join Properties window (Figure 27). A join operator should connect these tables. Later in this chapter, you will use the outer join operator. Confirm that the first choice (join) is selected and click **OK**.
9. Place Fields: In the Query Design window, drag the first three fields from the *Customer* table and the first four fields from the *Vehicle* table to the Query Design table (Figure 28). Note that you have to scroll the table to the right to access additional fields to fill.
10. Execute the Query: You should see seven fields filled with data as in Figure 29. When you are finished viewing, close and save the query as "QD6" when prompted.

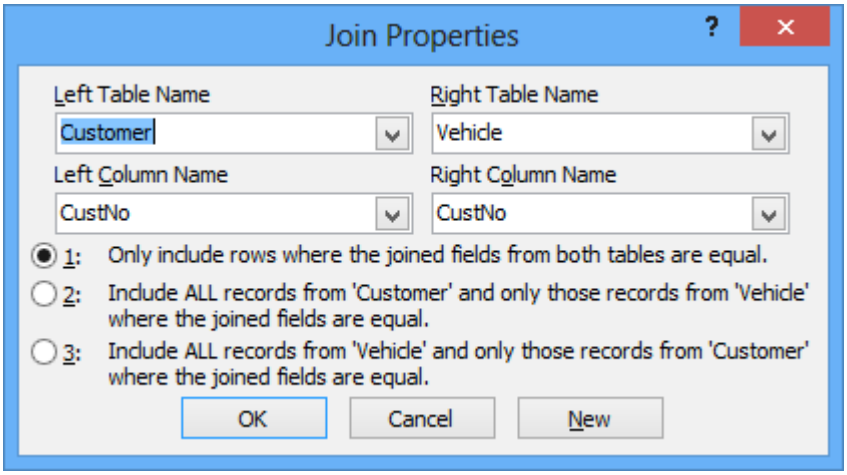


Figure 27: Join Properties Window

[7] If your Relationships window does not show thick lines (referential integrity is not supported), the lines in Figure 28 will appear as thin lines. You must enforce referential integrity in the Relationships window to make thick lines in the Join window (Figure 28).

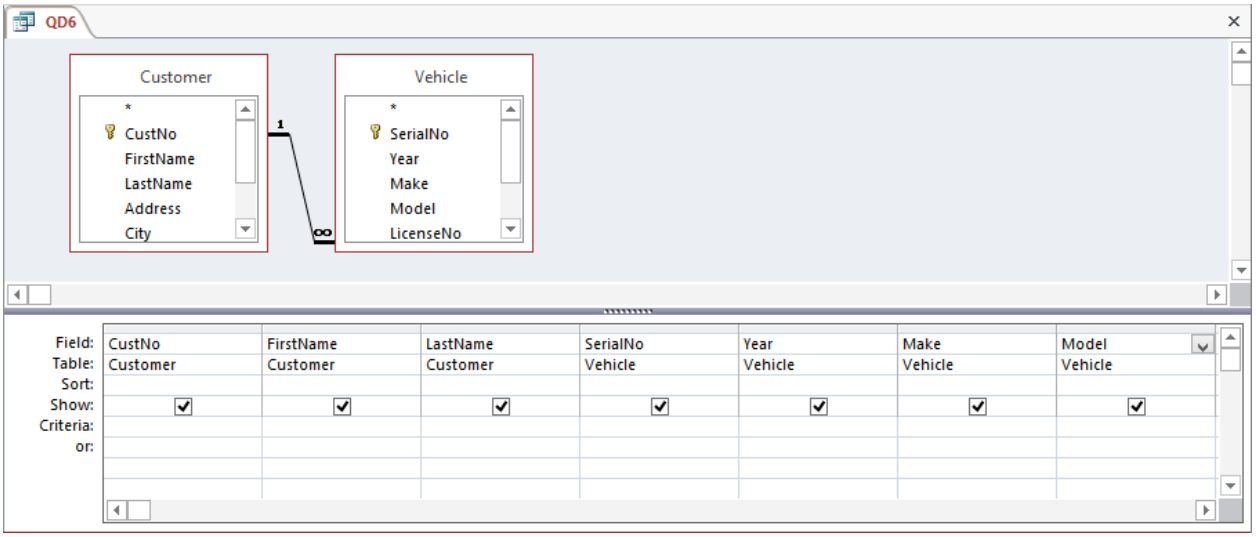


Figure 28: Completed "QD6"

QD6

CustNo	FirstName	LastName	SerialNo	Year	Make	Model
1	Beth	Taylor	QWER9LK982	2002	Pontiac	FireHawk
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal
2	Betty	Wise	JKLP7IIIJF	2004	Ford	Impala
3	Bob	Mann	POIU980PLL	2009	Chevrolet	Cavalier
4	Candy	Kendall	SWQW345RT6	2012	BMW	320i
5	Harry	Sanders	JHMEC3348H	1988	Buick	Corolla
5	Harry	Sanders	THYDF55639	2012	Ford	F-100
6	Helen	Sibley	TYYYI87590	2005	Toyota	Celica
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer
7	Homer	Wells	LPQW76200I	2007	Honda	Accord
8	Jerry	Wyatt	JHMCCF673Q	1995	Honda	Civic Si
8	Jerry	Wyatt	XCVY760PIQ	2003	Toyota	Celica
9	Jim	Glussman	MVMN8974PP	2009	Honda	Civic DX
9	Jim	Glussman	V121BHD481	2011	Toyota	Pick-up
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark
10	Larry	Styles	WWQAA452P0	1999	Buick	Regal

Record: 1 of 23 No Filter Search

Figure 29: Datasheet of "QD6" (first 16 rows)

3.2.5 Adding Another Table to the Join Query

In addition to knowing what customers own which vehicles, it also would be convenient to know what repair jobs have been performed for each customer. To accomplish this task, the *RepairOrder* table will be added to the previous query ("QD6"). Again, Query Design automatically connects the three tables, as the following steps describe.

5. Copy and Paste an Existing Query: To begin, copy "QD6" and rename it "QD7".

6. Open the Query Design Window: Next, open "QD7" in design view. It should appear the same as "QD6" (refer back to Figure 28).

7. Select an Additional Table: Next, click **Design → Show Table** from the Query Setup group. When the Show Table window appears, add the *RepairOrder* table and then click **Close**.

8. Open the Join Properties Window: In the Query Design window, double-click on the line connecting the *RepairOrder* and the *Vehicle* tables to open the Join Properties window. Confirm that the first choice is selected and click **OK**.

> Technical Note: In the future, you can avoid this step by remembering that the join operator is the default connection. You also can visually see that the connection is a join operator because of the 1 and ∞ symbols appearing in the line.

7. Place Additional Fields: In the Query Design window, drag the first three fields from the *RepairOrder* table to the Query Design table. Note that you will have to scroll the table to the right to access additional fields to fill.

8. Execute the Query: You should see ten fields filled with data as shown in Figure 30. When you are finished, close and save changes when prompted.

QD7

CustNo	FirstName	LastName	SerialNo	Year	Make	Model	OrdNo	Odometer	TimeRecvd
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	1	50000	10/5/2013 1:31:00 PM
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	4	75000	10/8/2013 11:15:00 AM
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	29	20600	11/25/2013 9:15:00 AM
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	6	55000	10/10/2013 4:42:02 PM
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	30	22590	11/27/2013 4:15:00 PM
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	31	23000	2/3/2002 10:20:00 AM
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	32	15000	12/1/2013 3:40:00 AM
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal	2	30000	10/5/2013 4:20:42 PM
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal	21	72700	11/5/2013 1:50:00 PM
5	Harry	Sanders	JHMEC3348H	1988	Buick	Corolla	25	61800	11/15/2013 8:10:00 AM
2	Betty	Wise	JKLP7IIIJF	2004	Ford	Impala	3	6000	10/7/2013 9:00:00 AM
2	Betty	Wise	JKLP7IIIJF	2004	Ford	Impala	15	14800	10/25/2013 9:05:00 AM
12	Ron	Thompson	MNMJ7H6098	2002	Lincoln	Towne Car	11	55060	10/18/2013 12:04:47 PM
12	Ron	Thompson	MNMJ7H6098	2002	Lincoln	Towne Car	17	71500	10/30/2013 2:35:00 PM
9	Jim	Glussman	MVMN8974PP	2009	Honda	Civic DX	18	15800	11/1/2013 12:10:00 PM
16	Wally	Jones	PLJFVC5609	2006	Toyota	Landcruiser	8	125000	10/12/2013 2:30:00 PM

Record: 1 of 32 No Filter Search

Figure 30: Datasheet of "QD7" (first 16 rows)

3.2.6 Creating an Outer Join Query

The goal of the next query is to create a list that matches repair orders with associated vehicles. This list is especially handy for employees since, as previously mentioned, many customers own more than one vehicle. To build this list, this query should join the *RepairOrder* and the *Vehicle* tables. But note that a difficulty arises because employees would like to see vehicles that have never been repaired. The join operator (as described in textbook Chapters 3 and 4) excludes nonmatching rows. The outer join operator is needed to include nonmatching rows. A one-sided outer join operator that preserves the *Vehicle* rows is needed. You might want to review the definition of the outer join operator in textbook Chapter 3 if you do not remember it. Access directly supports the one-sided outer join operator, as the following steps describe.

7. Create a New Query: Choose **Create → Query Design** from the Queries group to open the New Query window.

8. Select Two Tables: When the Show Table window appears, select the *RepairOrder* and the *Vehicle* tables, click the **Add** button**,** then click the **Close** button.

9. Change the Connection Type: In the Query Design window, double-click on the line connecting the tables to open the Join Properties window (refer back to Figure 27). Select the second choice, "Include ALL records from 'Vehicle' and only those records from 'RepairOrder' where the joined fields are equal.", and then click **OK**. This choice connects the tables with a one-sided join that preserves the *Vehicle* records. In Figure 31, note the arrow from the *Vehicle* to the *RepairOrder* table indicating a one-sided outer join.

10. Place Fields: In the Query Design window, drag the first three fields from the *RepairOrder* table and the first four fields from the *Vehicle* table to the Query Design table.

11. Sort the Result: In the *Sort* row of the *OrdNo* field, select "Ascending" from the list.

12. Execute the Query: You should see seven fields with data, as shown in Figure 32. Because of the sorting, the unmatched rows appear first. Unmatched rows have no values for fields from the *RepairOrder* table. When you are finished viewing, close and save the query as "QD8" when prompted.

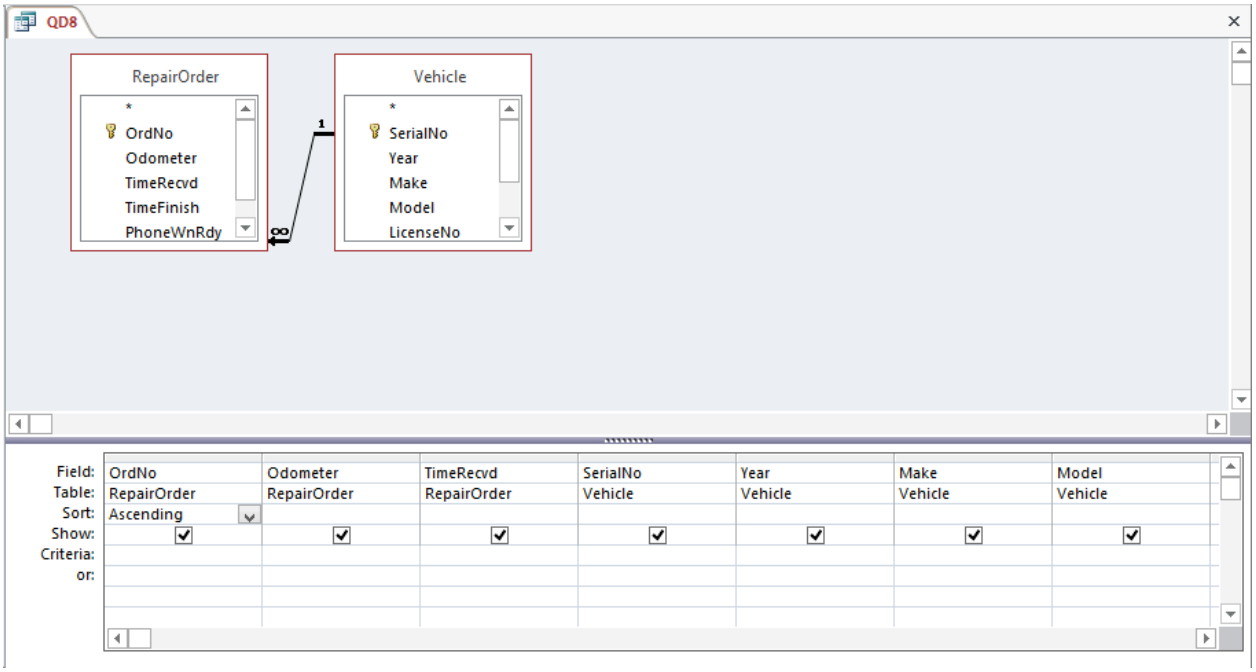


Figure 31: Query Design Window of Completed "QD8"

QD8

OrdNo	Odometer	TimeRecvd	SerialNo	Year	Make	Model
			GHL39789UI	1996	Honda	Del Sol
			LPQW76200I	2007	Honda	Accord
			JHMCCF673Q	1995	Honda	Civic Si
			VHJK009L88	2000	Ford	Taurus
1	50000	10/5/2013 1:31:00 PM	AZXS230I87	2004	Ford	Skylark
2	30000	10/5/2013 4:20:42 PM	IOMEQ54397	2003	Buick	Regal
3	6000	10/7/2013 9:00:00 AM	JKLP7IIIJF	2004	Ford	Impala
4	75000	10/8/2013 11:15:00 AM	AZXS230I87	2004	Ford	Skylark
5	15000	10/10/2013 11:53:30 AM	TYYYI87590	2005	Toyota	Celica
6	55000	10/10/2013 4:42:02 PM	BHGY08631Q	2008	Chevrolet	Blazer
7	25500	10/11/2013 11:53:32 AM	V121BHD481	2011	Toyota	Pick-up
8	125000	10/12/2013 2:30:00 PM	PLJFVC5609	2006	Toyota	Landcruiser
9	60800	10/15/2013 12:03:01 PM	XCVY760PIQ	2003	Toyota	Celica
10	11000	10/16/2013 11:41:00 AM	THYDF55639	2012	Ford	F-100
11	55060	10/18/2013 12:04:47 PM	MNMJ7H6098	2002	Lincoln	Towne Car
12	25600	10/20/2013 9:08:00 AM	THYDF55639	2012	Ford	F-100

Record: 16 of 36 No Filter Search

Figure 32: Datasheet of "QD8" (first 16 rows)

3.3 Creating Queries in SQL View

With a tool as convenient as Query Design, you may wonder why you would ever use SQL. Access provides SQL because it is the industry standard language. Query Design is specific to Access. In addition, some queries are either difficult or impossible to formulate in Query Design. Thus, SQL is preferred for some queries and the only alternative for other queries.

This section provides you additional practice creating queries in SQL view. Note that you will be creating queries by typing statements directly into SQL view since the expression builder is not available in SQL view. The SQL view is a much simpler tool than Query Design.

To access SQL view for a new query, you first must open the query in design view. When the Design View window appears with the Show Table dialog box, just click the **Close** button without adding any tables. Then toggle to SQL View via the **Query Tools | Design → SQL** command in the Results group or the SQL button in the view bar. Remember that you still may toggle between SQL and design view to examine your query.

3.3.1 Another Modification to the Parts List

Now you will return to the parts list with which you were working previously. To help employees in reordering decisions, the new query should display parts that have quantities less than 10.

2. Create a New Query: Create a new query in design view and then open the SQL View window. Type the following statement in the SQL window as shown in Figure 33. After typing the statement, execute it and return to SQL view.

```
SELECT *
 FROM Part
 WHERE UnitsInStock < 10;
```

4. Correcting Mistakes: If you had made a mistake when typing the query, Access would let you know when you try to run the query. To demonstrate an error, insert an extra letter in the field name *UnitsInStock*, such as an extra "k" at the end: *UnitsInStockk*. Now try to run the query. Instead, you see a dialog asking you for a parameter value (Figure 34). In this case, a parameter dialog signifies an error, so click **Cancel** and correct the mistake. Later in this chapter, you will write a query that uses a parameter value. Unless you use parameters, the parameter dialog signifies an error, usually a misspelled field name.

5. Execute the Query Again: After correcting the mistake, you should see the fields filled with data as in Figure 35. When you are finished viewing, close and save the query as "SQL1" when prompted.

Figure 33: SQL View

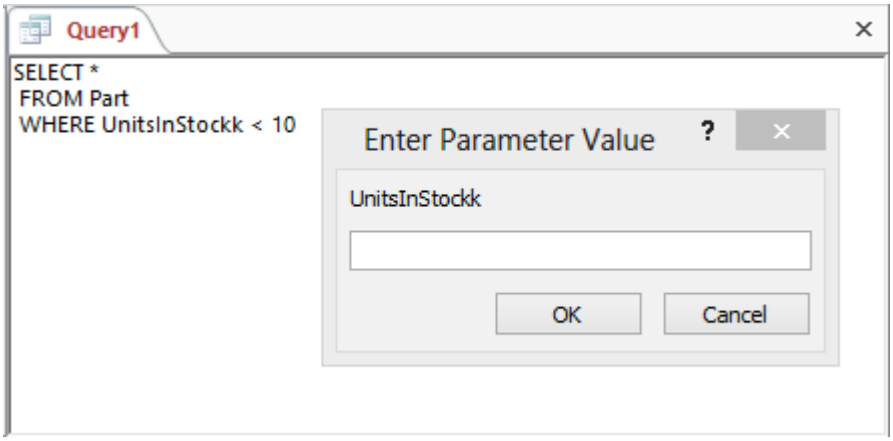


Figure 34: Parameter Dialog Indicating an Error

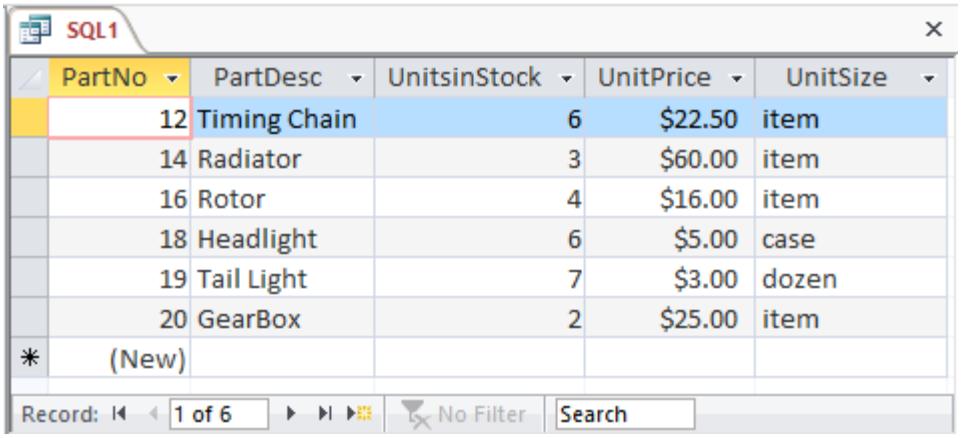


Figure 35: Datasheet of "SQL1"

3.3.2 Adding a Parameter Name to a Query

Next, you are going to refine the parts list to make it even more flexible. Instead of displaying a list of parts that have been depleted to 10 or less, you will use a parameter query to enter any desired cutoff value. When the query is run, a dialog box will appear prompting the user to enter a parameter value.

2. Create a new query: Create a new query in design view and then open the SQL View window. Type the following statement:

```
SELECT *
FROM Part
WHERE UnitsInStock < [Reorder Point:]
```

Note that square brackets "[]" enclose the parameter name. The square brackets must be used whenever a field name or parameter name contains spaces. Since *Reorder Point* is not a field name in the auto repair database, Access treats it as a parameter, as demonstrated next.

4. Execute the Query: When you execute the query, the dialog box in Figure 36 appears. Type in a number (5 is typed in Figure 36) and click **OK**. The datasheet displays the result as shown in Figure 37.

5. Close and Save the Query: When you are finished viewing, close and save the query as "SQL2" when prompted.

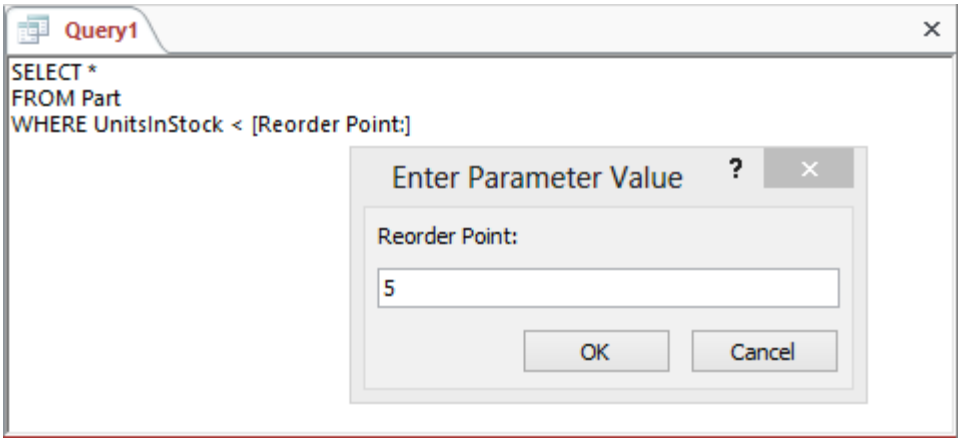


Figure 36: Parameter Value Dialog

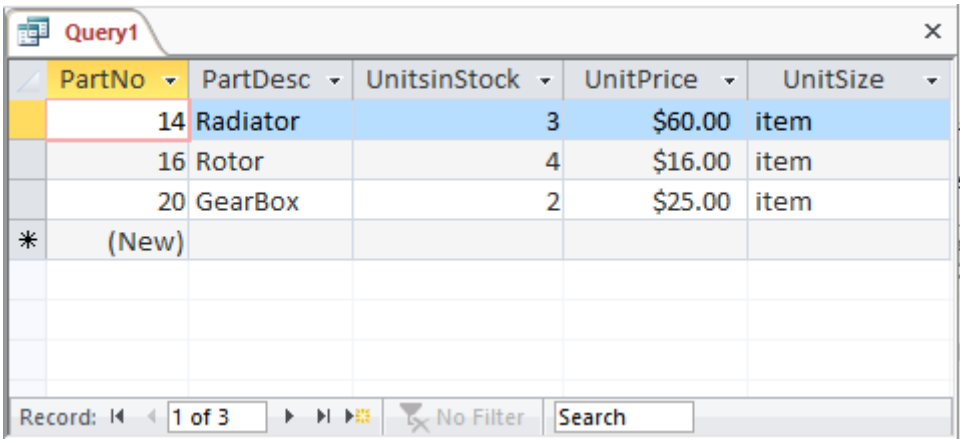


Figure 37: Datasheet of "SQL2"

3.3.3 Using a Query to Summarize Data

For month-end record keeping, the accounting department needs to know how many parts were used on each repair job. This task requires a query with the *Part* and *PartsUsed* tables along with a GROUP BY to perform the summarization. You will create this query in SQL view, but then toggle to observe the representation in design view.

2. Create a New Query: Create a new query in the Query Design window and then open the SQL window. Type in the following expression:

```
SELECT Part.PartNo, PartDesc, Count(*) AS
 [Total Repairs], Sum(QtyUsed) AS [Total Quantity]
FROM Part INNER JOIN PartsUsed
      ON Part.PartNo = PartsUsed.PartNo
GROUP BY Part.PartNo, PartDesc;
```

Note that this SQL statement uses the join operator style discussed in textbook Chapter 3. This style is used here because Query Design recognizes it. The cross product style used in textbook Chapter 4 is not understood as a join operation by Query Design.

5. Execute the Query: The datasheet displays the result as in Figure 38.

6. Toggle to Design View: To view this query in design view, click **Query Tools | Design → View → Design View** from the Results group or use one of the methods explained earlier in this chapter.

7. View Query Design Window: The Query Design window shows the join operation and the group by operation (Figure 39). When you are finished viewing, close and save the query as "SQL3" when prompted.

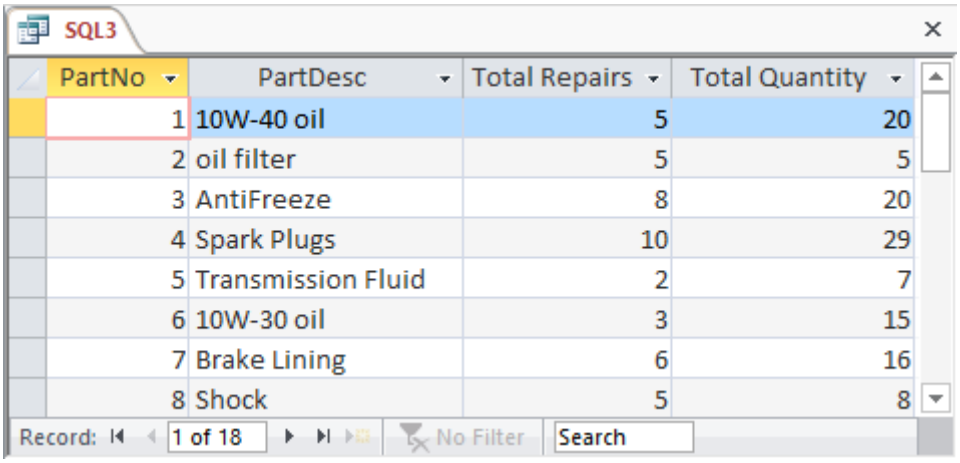


Figure 38: Datasheet of "SQL3" (first 8 rows)

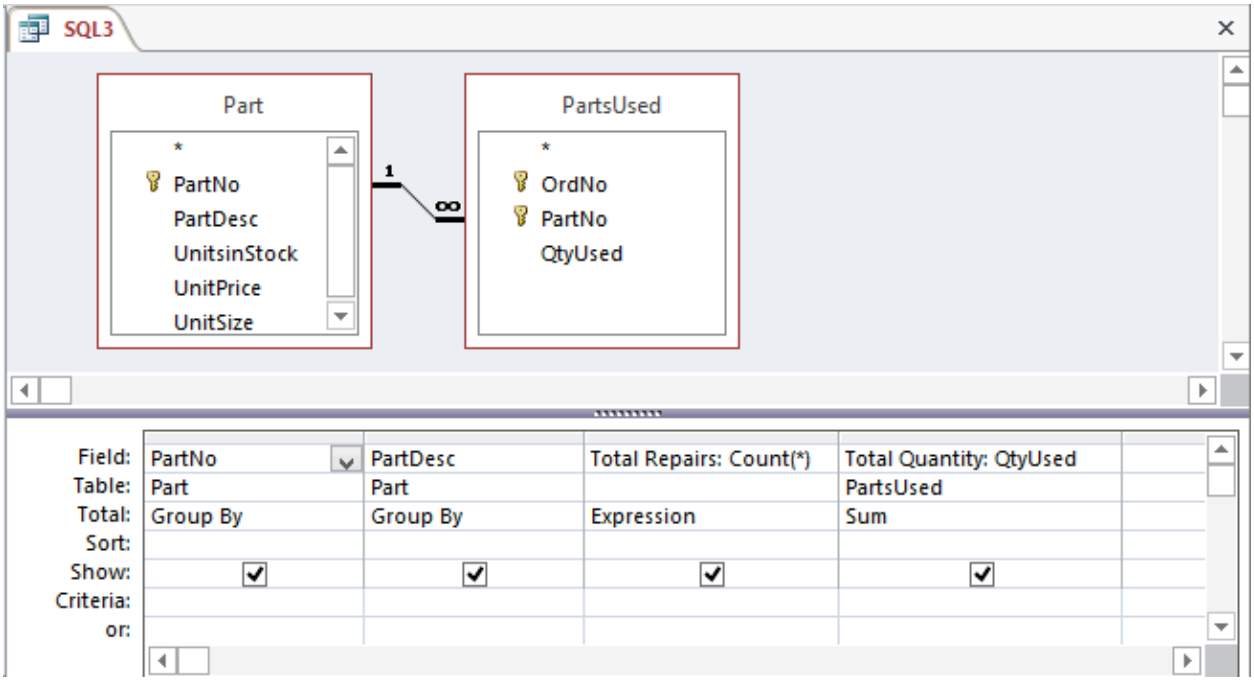


Figure 39: "SQL3" in Design View

3.3.4 Adding a Table to an Existing Query

The next query adds *PartsUsed* data to the list we made earlier showing repair jobs for each customer ("QD7"). You accomplished this task by adding the *RepairOrder* table to a previous query that contained only the *Customer* and the *Vehicle* tables. This time you will add a fourth table, the *PartsUsed* table, to the query "QD7". In addition, this query demonstrates using SQL to modify a query created in design view.

4. Copy and Paste an Existing Query: To begin, copy "QD7" and rename it as "SQL4".

5. Open the Query Design Window: Next, open "SQL4" in design view. It should appear on your screen the same as "QD7".

6. View the SQL Query: Open the SQL View window and view the code as shown below:

```
SELECT Customer.CustNo, Customer.FirstName,
 Customer.LastName, Vehicle.SerialNo, Vehicle.Year,
 Vehicle.Make, Vehicle.Model, RepairOrder.OrdNo,
 RepairOrder.TimeRecvd, RepairOrder.Odometer
FROM (Customer INNER JOIN Vehicle
 ON Customer.CustNo = Vehicle.CustNo)
 INNER JOIN RepairOrder
 ON RepairOrder.SerialNo = Vehicle.SerialNo;
```

6. Add a Table to the Query: To add the *PartsUsed* table to the query, type the additional code seen underlined below. The full SQL statement also appears in Figure 40. Note that you should add the *PartNo* and *QtyUsed* fields from the *PartsUsed* table since *OrdNo* is already contained in the query from the *RepairOrder* table.

```
SELECT Customer.CustNo, Customer.FirstName,
 Customer.LastName, Vehicle.SerialNo, Vehicle.Year,
 Vehicle.Make, Vehicle.Model, RepairOrder.OrdNo,
 RepairOrder.TimeRecvd, RepairOrder.Odometer,
 PartsUsed.PartNo, PartsUsed.QtyUsed
FROM ((Customer INNER JOIN Vehicle
 ON Customer.CustNo = Vehicle.CustNo)
 INNER JOIN RepairOrder
 ON RepairOrder.SerialNo = Vehicle.SerialNo)
 INNER JOIN PartsUsed
 ON RepairOrder.OrdNo = PartsUsed.OrdNo;
```

7. Execute the Query: Execute the query to see if everything is typed in correctly. The datasheet should have 12 fields across as in Figure 41. When you are finished viewing, close and save the changes when prompted.

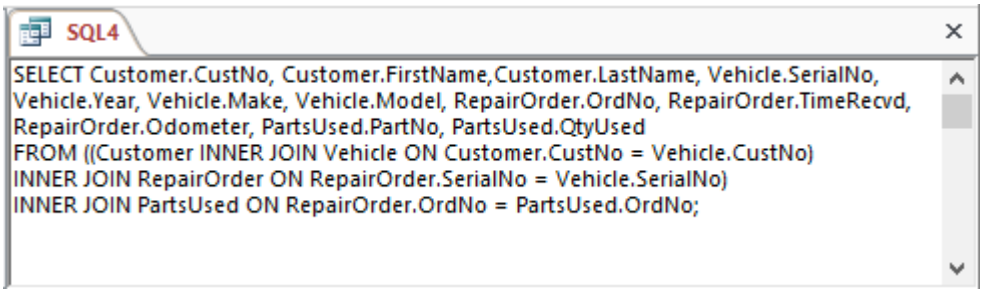


Figure 40: SQL Window Containing "SQL4"

SQL4

CustNo	FirstName	LastName	SerialNo	Year	Make	Model	OrdNo	TimeRecvd	Odometer	PartNo	QtyUsed
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	1	10/5/2013 1:31:00 PM	50000	2	1
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	1	10/5/2013 1:31:00 PM	50000	3	4
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	29	11/25/2013 9:15:00 AM	20600	4	1
10	Larry	Styles	AZXS230I87	2004	Ford	Skylark	29	11/25/2013 9:15:00 AM	20600	9	1
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	6	10/10/2013 4:42:02 PM	55000	7	1
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	6	10/10/2013 4:42:02 PM	55000	8	2
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	30	11/27/2013 4:15:00 PM	22590	2	1
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	30	11/27/2013 4:15:00 PM	22590	3	2
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	30	11/27/2013 4:15:00 PM	22590	6	6
7	Homer	Wells	BHGY08631Q	2008	Chevrolet	Blazer	32	12/1/2013 3:40:00 AM	15000	8	2
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal	2	10/5/2013 4:20:42 PM	30000	3	1
2	Betty	Wise	IOMEQ54397	2003	Buick	Regal	2	10/5/2013 4:20:42 PM	30000	4	5

Record: 1 of 65 No Filter Search

Figure 41: Datasheet of "SQL4"

3.4 Creating Special Queries

In addition to the queries described above, there are a few queries with specific uses that are explained below. They are the Append query, the Update query, the Delete query, the Union query, and the Make Table query. The SQL formulation for these kinds of queries is shown because Query Design does not fully support formulations of them.

3.4.1 The Append Query

An Append query inserts records into a table. You create an Append query using the SQL INSERT statement as discussed in textbook Chapter 4. An Append query cannot be viewed in Design view after it is created in SQL view. Query Design supports Append queries in which a collection of records from one table is inserted into another table. You will use an Append query to insert an additional record into the *Part* table.

2. Create a New Query: Create a new query in design view and open the SQL window. Type in the following statement:

```
INSERT INTO Part ( PartDesc, UnitsInStock, UnitPrice,
 UnitSize )
SELECT "Gas Filter" AS Expr1, 10 AS Expr2,
 15.5 AS Expr3, "item" AS Expr4
```

Note that this syntax differs from that shown in textbook Chapter 4. Using the syntax of textbook Chapter 4, the INSERT statement appears as shown below. If you use this syntax be warned that Access changes it to the syntax with the `SELECT` keyword.

```
INSERT INTO Part (PartDesc, UnitsInStock, UnitPrice,
 UnitSize )
VALUES ("Gas Filter", 10, 15.5, "item");
```

4. Execute the Query: When you try to execute the query, the message in Figure 42 appears. Click the **Yes** button to insert the new record. You should only execute this query once. Executing it more than one time will insert duplicate records, except for the unique *PartNo* value generated by Access. Close and save the query as "SQL5" when prompted.

5. View the New Record: Now, return to the **Tables** section of the Database window and open the *Part* table to see the additional row (Figure 43).

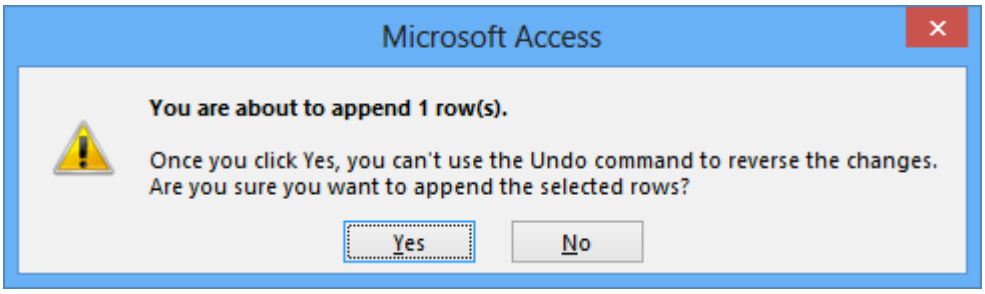


Figure 42: Append Message Box

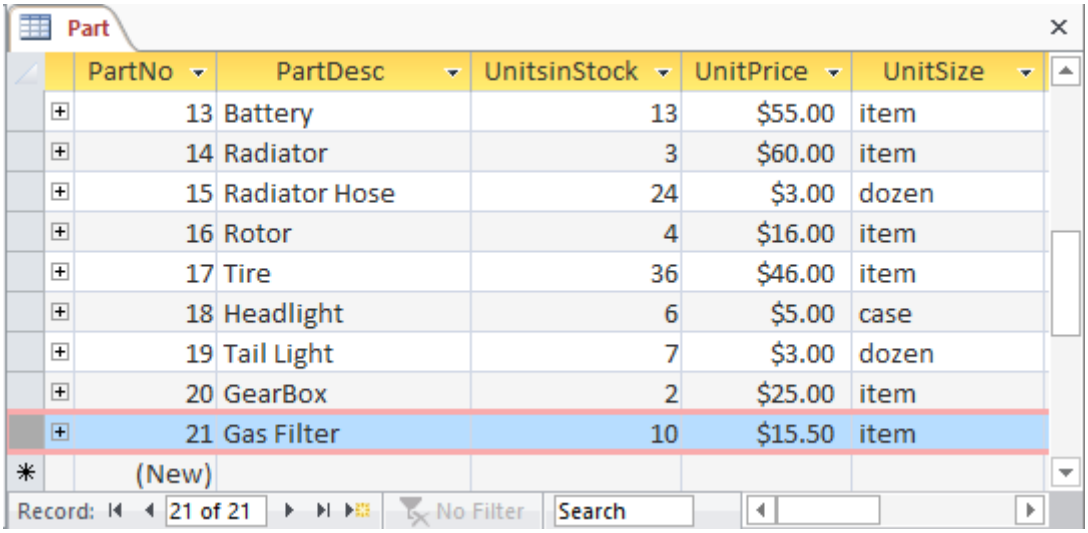


Figure 43: Bottom Part of Datasheet (*Part* Table) of "SQL5"

3.4.2 The Update Query

An Update query changes the values of individual fields of selected records. You create an Update query using the SQL UPDATE statement as discussed in textbook Chapter 4 or using Query Design. Unlike an Append query, an UPDATE query can be viewed in design view after it is created in SQL view. You will use an Update query to decrease the quantity of gas filters in the *Part* table.

2. Create a New Query: Create a new query in design view and open the SQL window. Type the following statement:

```
UPDATE Part SET UnitsInStock = UnitsInStock - 1
 WHERE PartDesc = "Gas Filter";
```

4. Execute the Query: When you try to execute the query, the message in Figure 44 appears. Click the **Yes** button to update the record. Update queries can be repeatedly executed if you desire. If you execute this query two more times, the *UnitsInStock* field is reduced by 2. Close and save the query as "SQL6" when prompted.

5. View the Changes: Return to the **Tables** section of the Database window and open the *Part* table. You can see that the number of gas filters in Figure 45 has decreased from 10 to 9 in comparison to the datasheet in Figure 43 above.

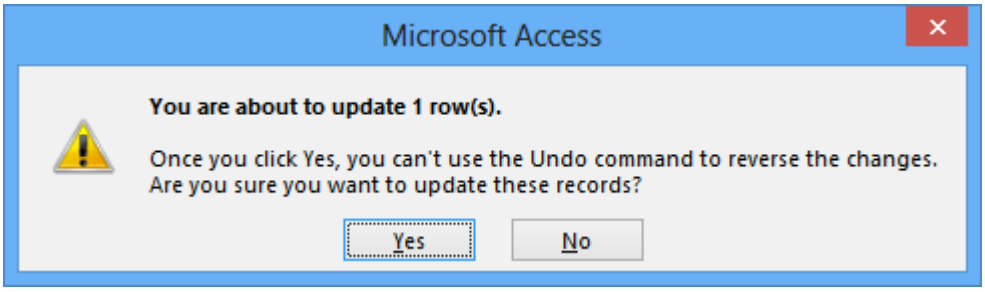


Figure 44: Update Message Box

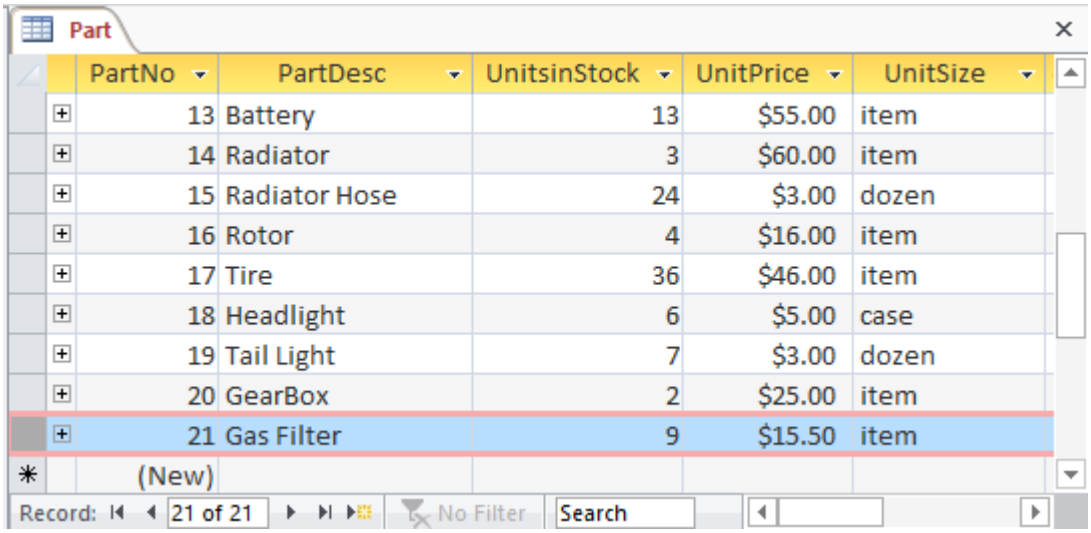


Figure 45: Bottom Part of Datasheet (*Part* Table) of "SQL6"

3.4.3 The Delete Query

A Delete query removes selected rows from the database. You create a Delete query using either the SQL DELETE statement, as discussed in textbook Chapter 4, or Query Design. Like Append queries, Delete queries cannot be viewed in design view after created in SQL view. You will use a Delete query to remove the gas filter record from the *Part* table.

2. Create a New Query: Create a new query in design view and open the SQL window. Type the following statement:

```
DELETE *
FROM Part
WHERE PartDesc = "Gas Filter";
```

4. Execute the Query: When you try to execute the query, the message in Figure 46 appears. Click **Yes** to delete the row. You should only perform Delete queries once. If you perform them more than one time, you will generate an error message about the record not existing. Close and save the query as "SQL7" when prompted.

5. View the Changed *Part* Table: Return to the **Tables** part of the navigation pane. Open the *Part* table to see that the gas filter row has been deleted (Figure 47).

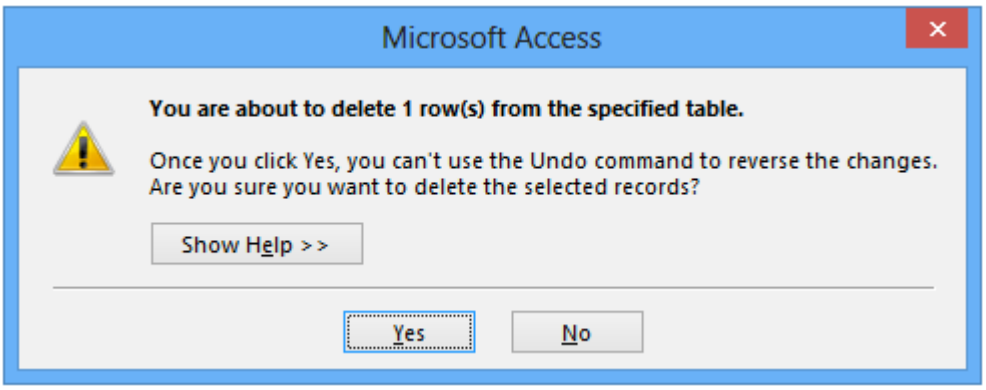


Figure 46: Delete Message Box

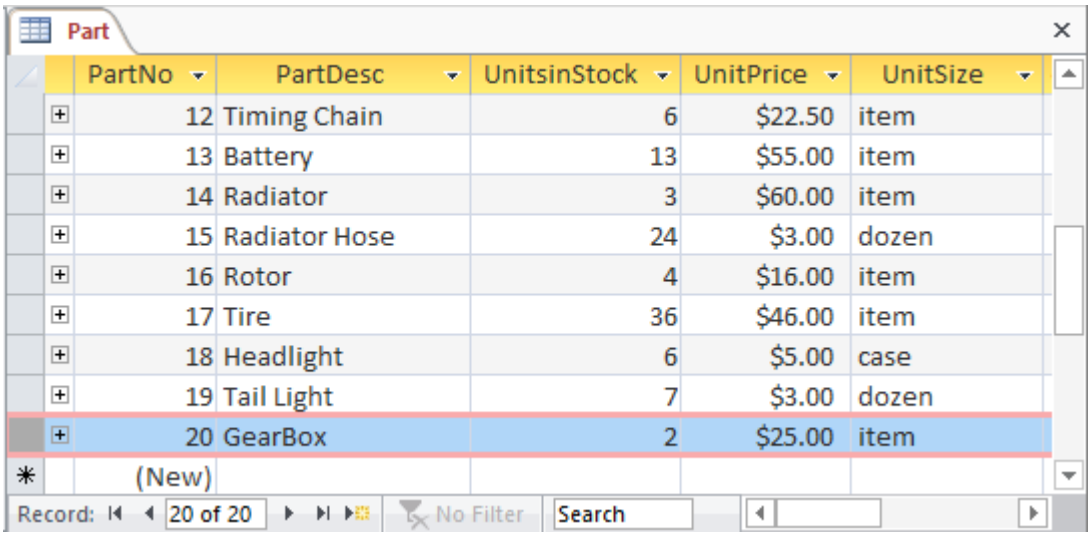


Figure 47: Bottom Part of Datasheet (*Part* Table) of "SQL7"

3.4.4 The Union Query

A Union query combines the results of two or more queries using the union operator. The union operator in SQL, like the union operator in relational algebra, requires "union compatible" tables. Recall from textbook Chapter 3 that two tables are union compatible if they have the same number of columns and each corresponding column has a compatible data type. You will use a Union query to display a vehicle list containing Honda or Toyota models. Notice that each SELECT part of the query is union compatible. A Union query cannot be viewed in design view after it is created in SQL view.

2. Create a New Query: Create a new query in design view and open the SQL window. Type the following statement in the SQL window:

```
SELECT * FROM Vehicle WHERE Make = "Honda"
 UNION
SELECT * FROM Vehicle WHERE Make = "Toyota";
```

3. Execute the Query: The datasheet displays the result as shown in Figure 48. When you are finished viewing the result, close and save the query as "SQL8" when prompted.

SQL8

SerialNo	Year	Make	Model	LicenseNo	State	Cylinders	CustNo
GHL39789UI	1996	Honda	Del Sol	123XYZ	WA	4	13
JHMCCF673Q	1995	Honda	Civic Si	367ABC	WA	4	8
LPQW76200I	2007	Honda	Accord	123ABC	MT	4	7
MVMN8974PP	2009	Honda	Civic DX	567TUV	WA	4	9
PLJFVC5609	2006	Toyota	Landcruiser	876GYK	WA	4	16
TGVYY76R43	2008	Honda	Civic	980QAW	WA	4	15
TYYYI87590	2005	Toyota	Celica	890YUI	ID	6	6
V121BHD481	2011	Toyota	Pick-up	190PGF	WA	6	9
XCVY760PIQ	2003	Toyota	Celica	477HSD	WA	4	8

Record: 1 of 9 No Filter Search

Figure 48: Datasheet of "SQL8"

3.4.5 Make-Table Query

Microsoft Access 2013 provides an additional special query known as a Make-Table Query. Using a Make-Table Query, you can copy selected rows of an existing table or query into a new table. To create a Make-Table query, you select the **Query Tools | Design → Make-Table**… command in the Query Type group for an empty query open in SQL or design view. Then you provide a table name in the Make Table dialog window. Access generates a SELECT … INTO statement with the specified table name inserted after the INTO clause. For more information, you should review the SELECT … INTO statement in the Help documentation.

The SELECT … INTO statement[8] is not standard SQL although it is a useful variant of the SELECT statement. Standard SQL includes the INSERT … SELECT statement (see textbook Chapter 4) to insert a set of rows from another table into an existing table. Thus, the SELECT … INTO statement creates a new table whereas the INSERT … SELECT statement must use an existing table. If you are developing an application that may need to be converted to another DBMS, you should avoid the SELECT … INTO statement.

3.4.6 Summary of Special Queries

Because of the special nature of the Append, Update, Delete, and Union queries, Access provides separate icons for them. In Figure 49, you can see the + icon for the Append query (SQL5), the pencil icon for the Update query (SQL6), the × icon for the Delete query (SQL7), and the intersecting rings icon for the Union query (SQL8).

[8] The Access SELECT … INTO statement is different than the SELECT … INTO statement for embedded SQL. The embedded SQL SELECT … INTO, part of standard SQL, supports retrieval of single-row results into programming language variables. Textbook Chapter 11 describes the embedded SQL SELECT … INTO statement in the presentation of stored procedures.

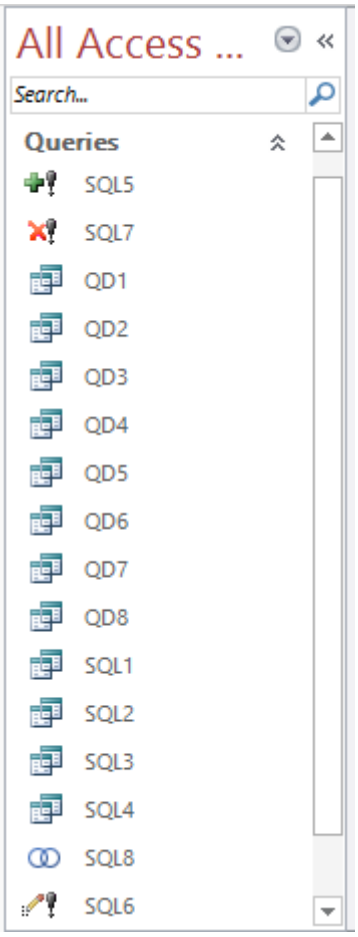


Figure 49: List of Queries Used in This Chapter

3.5 Query Subdatasheets

As you may have noticed, none of the query datasheets in this chapter displayed the + symbol indicating an available subdatasheet. Although query datasheets are capable of displaying subdatasheets, there is no default setting as there was for table datasheets. A subdatasheet may be displayed by setting the desired table or query name in the *Subdatasheet Name* query property in the Query Properties window as you did for table datasheets. You also need to set the *Link Master Fields* and the *Link Child Fields* properties to establish a connection between the datasheet and the subdatasheet.

As an example, you can open the QD1 query in design view and set a property to display a subdatasheet. Remember that QD1 contains customer information, so when you are finished you will have a subdatasheet to show the customer's vehicle information.

- To open the Query Properties window, right-mouse-click in an empty area in the Query Design pane and choose **Properties…** (Figure 50).
- Scroll down to locate the *Subdatasheet Name* property. Click in that property to reveal a drop-down menu and select "Table.Vehicle" (Figure 51).
- Toggle to datasheet view (Figure 52) and click on a + sign to expand the subdatasheet. All vehicles are shown for each customer because no connection has been established between the datasheet and the subdatasheet.

- Toggle back to Query Design and locate the *Subdatasheet Name* property again. Set the *Link Master Fields* and the *Link Child Fields* properties to "CustNo" as shown in Figure 53.
- Toggle to datasheet view again and click on a + sign to expand the subdatasheet as shown in Figure 54. Now the subdatasheet displays only the vehicles of the given customer, not all vehicles in the database.
- Toggle to back to Query Design and locate the *Subdatasheet Name* property again. Delete the value so that it is blank (its original state before this example). Now when you toggle to datasheet view, the + symbols have disappeared. Close the datasheet without saving changes.

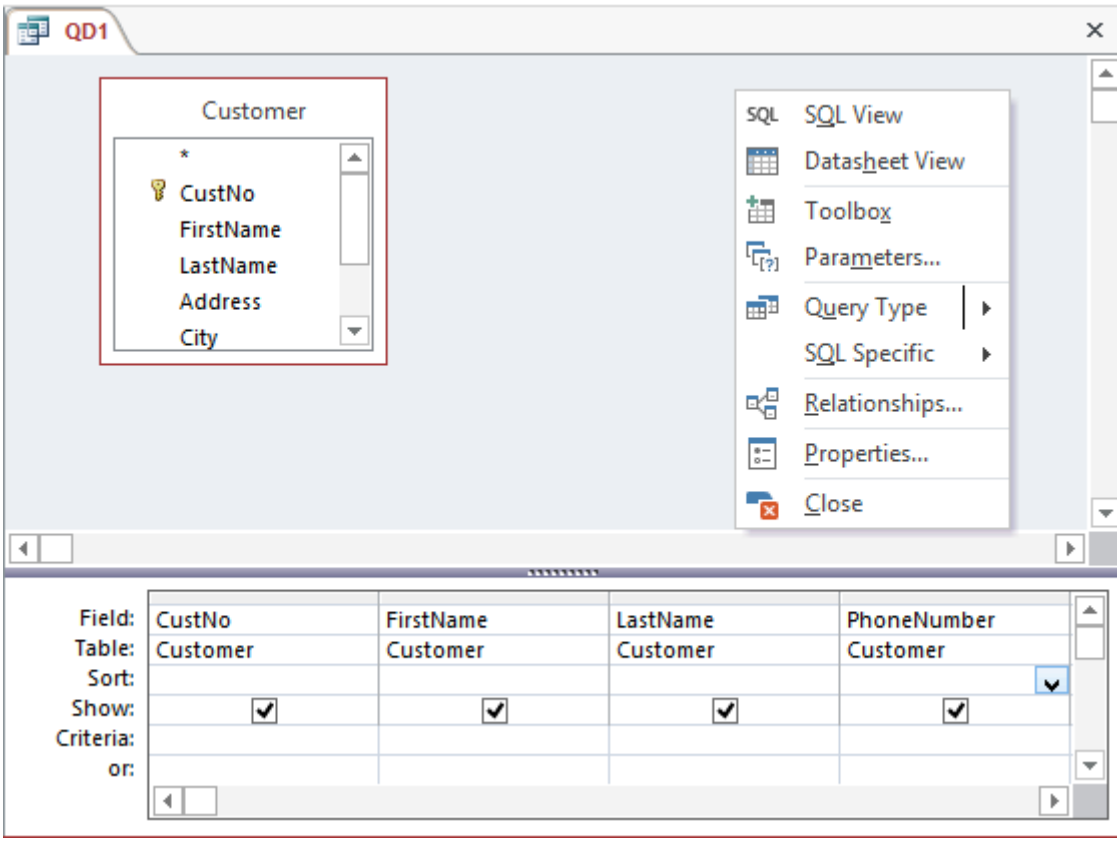


Figure 50: Design View and the Right Mouse Shortcut Menu Showing **Properties…**

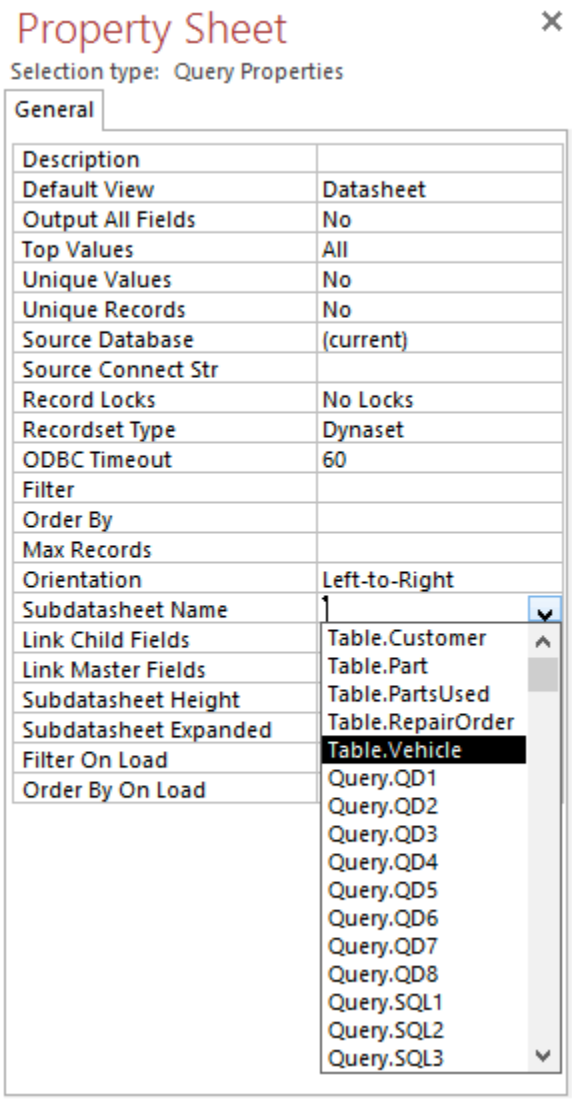


Figure 51: Query Properties Window Showing *Subdatasheet Name* Property

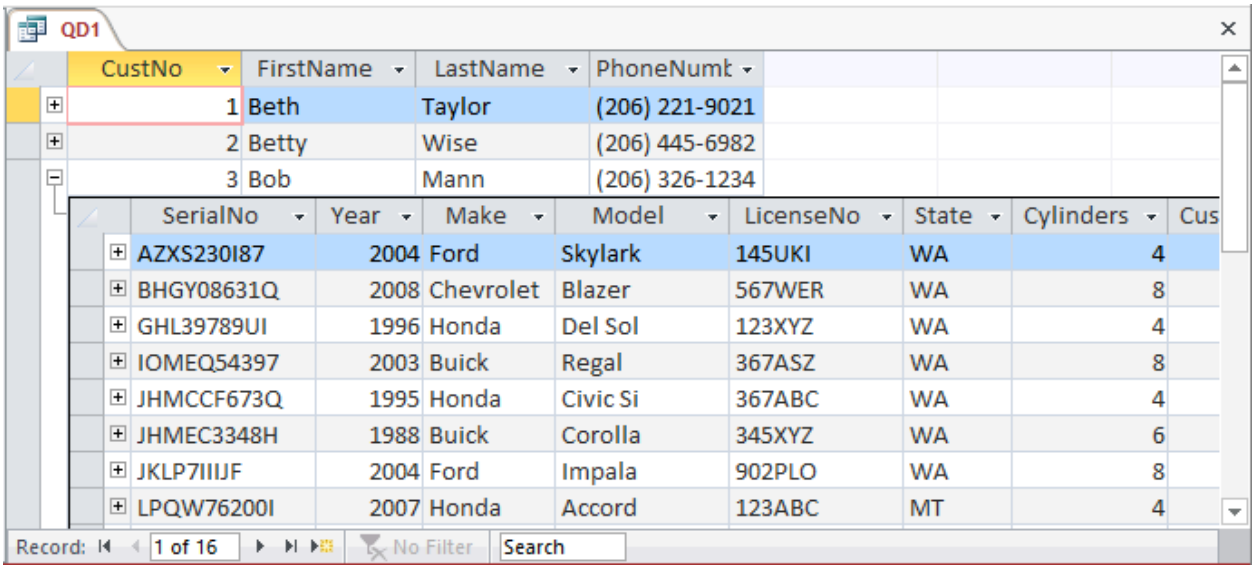


Figure 52: Query Datasheet with Hidden Subdatasheet

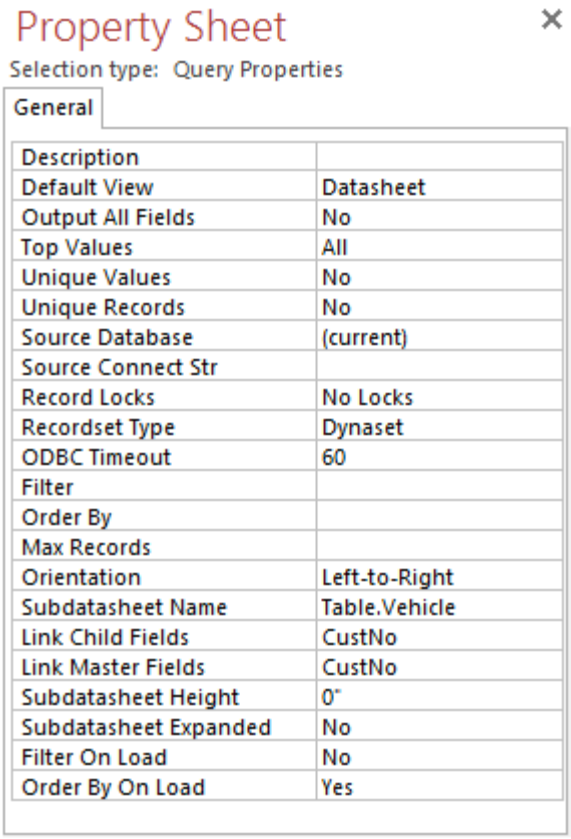


Figure 53: Query Properties Window Showing the Linking Properties

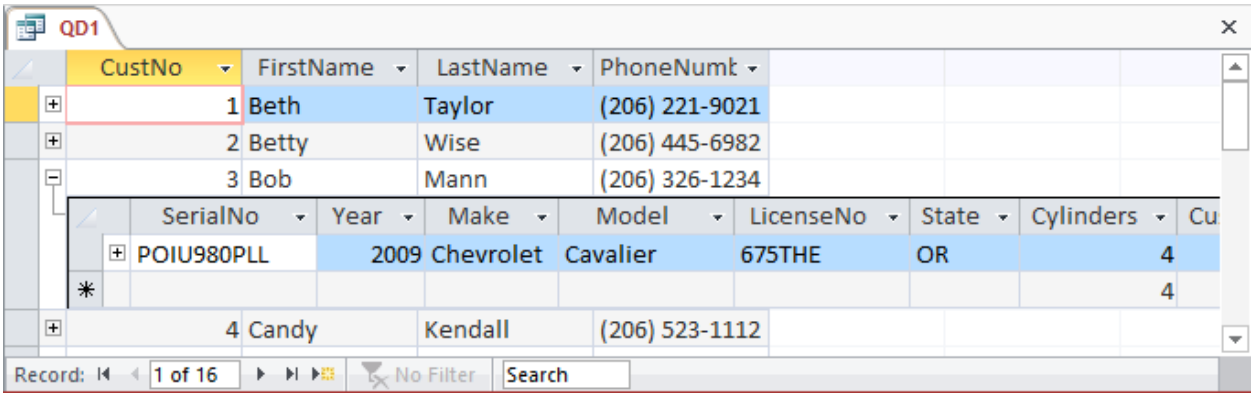


Figure 54: Expanded Subdatasheet after Setting the Linking Properties

3.6 Standards Compliance of Access SQL

As explained in textbook chapters 3, 4, and 9, international standards organizations such as the American National Standards Institute (ANSI) have specified a number of SQL standards. The most recent standards are known as SQL-89, SQL-92, and SQL:1999 in reference to the years in which the standards were formally accepted. The SQL:1999 standard has been updated with new parts in the SQL:2003, SQL:2008, and SQL:2011 standards. In each subsequent standard, the size of the specification has grown. For example, the SQL-92 specification contains about 600 pages whereas the SQL:1999 specification contains about 2,000 pages. Most relational DBMSs (including Access) support a subset of a standard (typically SQL-92) while adding proprietary extensions.

This section supplements the presentation in textbook chapters 3, 4, and 9 by providing insight into the standards compliance of Access SQL. The compliance of Access is more complex than other relational DBMSs because native Access databases using the Jet database engine support both the SQL-89 and the SQL-92 specifications along with proprietary extensions. Although Access does support both SQL specifications, it does not satisfy level 1 compliance for either specification. In addition, Access supports Microsoft SQL Server syntax after converting an Access database to a SQL Server database. SQL Server supports a subset of SQL-92 with some proprietary extensions.

3.6.1 ANSI Query Mode

Access 2013 supports the choice of SQL-89 or SQL-92 query mode using the Object Designers item in the Access Options window (**File → Options**). The choice can be made in the bottom right of the window in the "SQL Server Compatible Syntax (ANSI 92)" area. You can choose to make SQL-92 the query mode for the current database (checked in Figure 55) and the default for new databases (not checked in Figure 55). By default, SQL89 is the default query mode for Access 2013 format databases.

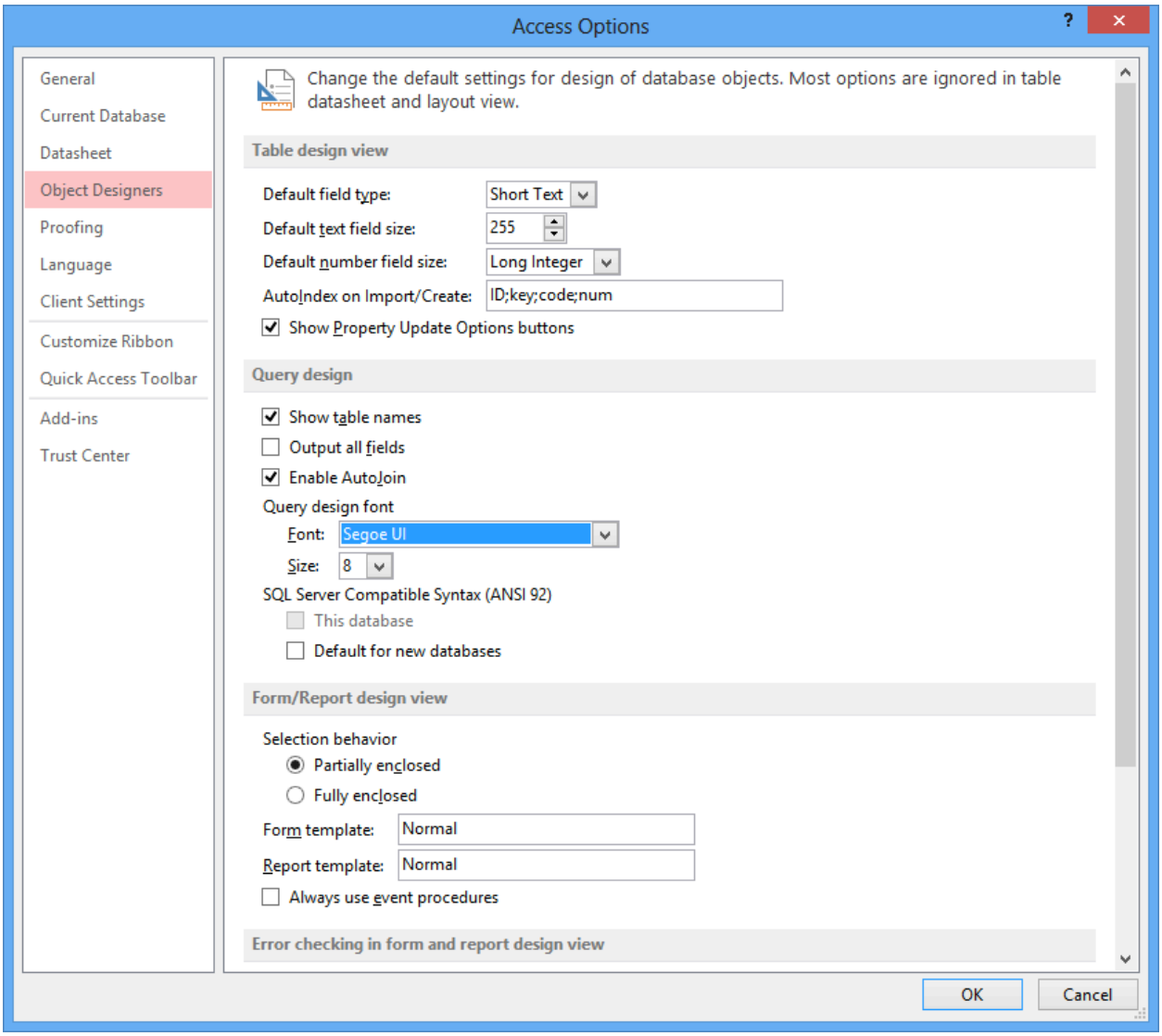


Figure 55: Advanced Tab of the Options Window Showing Query Mode Choices

Note that SQL-92 syntax can only be specified for the Access 2013, Access 2010 and Access 2003 file formats. If you convert to a previous Access format, some SQL statements may have syntax errors or produce incorrect results. Thus, you should not use SQL-92 query mode if you intend to convert a database to a previous Access format.

In practice, the SQL-92 query mode has little impact on data manipulation statements (SELECT, INSERT, UPDATE, and DELETE). However, it has a significant impact on data definition statements as presented in the following section. For SELECT statements, SQL-89 and SQL-92 seem to differ only on meta characters used with the LIKE operator and on duplicate field and alias names used in a query. Because using duplicate field and alias names is poor practice, the differences will not be presented here. You should be aware of the different meta characters used in SQL-89 and SQL-92, however. The two most widely used meta characters are the wildcard symbols for multiple characters (* in SQL-89 and % in SQL-92) and single characters (? in SQL-89 and _ in SQL-92). Examples 3.1 and 3.2 depict the SQL-89 and SQL-92 wildcard characters.

Example 3.1: **Using the meta character % in SQL-92 query mode**

Retrieve customers with a last name beginning with B. If this statement is executed in SQL89 mode, no records are returned because the meta character (%) has no special meaning.

```
SELECT *
 FROM Customer
 WHERE LastName LIKE 'B%'
```

Example 3.2: **Using the meta character * in SQL-89 query mode**

Retrieve customers with a last name beginning with B. If this statement is executed in SQL92 mode, no records are returned because the meta character (*) has no special meaning.

```
SELECT *
 FROM Customer
 WHERE LastName LIKE 'B*'
```

Access, like most relational DBMSs, supports a subset of the SQL standards with some proprietary extensions. For example, neither SQL-89 nor SQL-92 query modes supports the DISTINCT operator inside aggregate functions such as the COUNT function. As an example of a supported proprietary extension, Access provides the TRANSFORM statement to produce a pivot table used in data warehouse applications.

In addition to proprietary extensions, both SQL-89 and SQL-92 query modes support elements of the SQL:1999 standard. The most prominent part of the SQL:1999 standard supported is the use of a SELECT statement in the FROM clause as demonstrated in Example 3.3. Note that after you save the query, Access slightly varies the syntax of the nested query similar to the way in which the syntax is changed for the INSERT statement presented in Section 3.4.1.

Example 3.3: **Using a nested query inside the FROM clause**

Retrieve the list of vehicle serial numbers along with the count of unique parts repaired. The SELECT statement inside the FROM clause extracts the unique combinations of *Vehicle.SerialNo* and *PartNo*.

```
SELECT T1.SerialNo, COUNT(*) AS NumUniqueParts
 FROM
 (SELECT DISTINCT Vehicle.SerialNo, PartNo
   FROM RepairOrder, PartsUsed, Vehicle
   WHERE RepairOrder.OrdNo = PartsUsed.OrdNo
    AND Vehicle.SerialNo = RepairOrder.SerialNo) AS T1
 GROUP BY T1.SerialNo
```

Example 3.3 could be accomplished without a SELECT statement in the FROM clause if Access supported the DISTINCT keyword inside of the COUNT function. Without using a nested query in the FROM clause, two queries must be used as shown in Example 3.4. The second statement produces the list of serial numbers along with the count of the unique parts repaired using the name of the stored query in the FROM clause.

Example 3.4: **Using two queries instead of a nested query in the FROM clause**

Retrieve the list of vehicle serial numbers along with the count of unique parts repaired. The first SELECT statement, saved as "Temp3_4" produces the unique combinations of *Vehicle.SerialNo* and *PartNo*. The second SELECT statement uses the *Temp3_4* in the FROM clause to produce the result.

Temp3_4:

```
SELECT DISTINCT Vehicle.SerialNo, PartNo
 FROM RepairOrder, PartsUsed, Vehicle
 WHERE RepairOrder.OrdNo = PartsUsed.OrdNo
   AND Vehicle.SerialNo = RepairOrder.SerialNo


SELECT SerialNo, COUNT(*) AS NumUniqueParts
 FROM Temp3-4
 GROUP BY T1.SerialNo
```

3.6.2 Data Definition Statements

Although Access provides powerful and convenient tools for data definition, you can also use SQL data definition statements such as the CREATE TABLE statement. You can create a data definition query in the SQL window. You can also use the **Query Tools | Design → Data Definition** command in the Query Type group to open a data definition query in SQL view after you have created a new query. You can find documentation about the data definition statements in the subtopic "Create or modify tables or indexes" in the main topic "Designing Applications" in the table of contents of the Access help documentation. The details about SQL data types are not contained in this help documentation, a major omission if you want to write CREATE TABLE statements. The web page (http://msdn.microsoft.com/en-us/library/ff841694.aspx) is another source of documentation about data definition statements in Access SQL.

If you intend to use SQL data definition statements, SQL-92 query mode provides more coverage than SQL-89 mode. Table 1 summarizes the differences in data definition statements in SQL-89 and SQL-92 query modes. SQL-92 query mode supports most of the SQL data definition statements discussed in textbook chapters 3, 10, and 14.

The treatment of the CREATE VIEW statement in SQL-92 query mode is unconventional. The CREATE VIEW statement is covered in textbook Chapter 10. Access saves a view as an object in the list of queries. When a view is executed, Access creates another query containing the query part of the view definition. Example 3.5 shows a CREATE VIEW statement[9] for the query in QD7. Example 3.6 shows the result of executing the CREATE VIEW statement in Example 3.5. Example 3.7 shows a query using the view in Example 3.5. The net result is that the CREATE VIEW statement does not provide any benefits beyond just saving the query directly. The CREATE VIEW statement adds another object to the query list in addition to the query that would be saved anyway.

[9] Note that Example 3.5 will only execute successfully in SQL-92 query mode.

Table 1: Summary of Data Definition Support in SQL-89 and SQL-92 Query Modes

Statement	SQL-89	SQL-92
CREATE TABLE	No support for many features.	Added support for defaults, check constraints, cascading referential integrity, foreign keys, and custom AutoNumber seed and increment values.
ALTER TABLE	No support for altering column definition.	Added support for altering column definition.
CREATE/DROP PROCEDURE	No support	Supported
CREATE/DROP VIEW	No support	Supported
GRANT/REVOKE	No support	Supported
CREATE/DROP USER	No support	Supported
CREATE/DROP GROUP	No support	Supported
CREATE INDEX	No support	Supported

Example 3.5: **CREATE VIEW Statement for QD7**

Retrieve details from the *Customer*, *Vehicle*, and *RepairOrder* tables.

```
CREATE VIEW CustVehRepOrdView
AS
SELECT Customer.CustNo, Customer.FirstName,
  Customer.LastName, Vehicle.SerialNo, Vehicle.Year,
  Vehicle.Make, Vehicle.Model, RepairOrder.OrdNo,
  RepairOrder.TimeRecvd, RepairOrder.Odometer
 FROM (Customer INNER JOIN Vehicle
   ON Customer.CustNo = Vehicle.CustNo)
  INNER JOIN RepairOrder
   ON RepairOrder.SerialNo = Vehicle.SerialNo
```

Example 3.6: **Generated SQL Statement for CustVehRepOrdView**

This query is created after executing the CustVehRepOrdView. Access saves the query as CustVehRepOrdView.

```
SELECT Customer.CustNo, Customer.FirstName,
    Customer.LastName, Vehicle.SerialNo,
    Vehicle.Year, Vehicle.Make, Vehicle.Model,
    RepairOrder.OrdNo, RepairOrder.TimeRecvd,
    RepairOrder.Odometer
 FROM (Customer INNER JOIN Vehicle
    ON Customer.CustNo = Vehicle.CustNo)
  INNER JOIN RepairOrder
      ON Vehicle.SerialNo = RepairOrder.SerialNo
```

Example 3.7: **SQL SELECT Statement using CustVehRepOrdView**

This query uses the CustVehRepOrdView.

```
SELECT *
 FROM CustVehRepOrdView
 WHERE Make = "Toyota"
```

Closing Thoughts

Chapter 3 has provided guided instruction about query formulation in Microsoft Access 2013. Access provides two tools to create queries, the visual Query Design tool and the text-oriented SQL view. You learned how to use the Query Design tool to specify computed columns, row conditions, joins, and outer joins. You gained practice with the SQL view to formulate the kind of queries presented in textbook Chapters 4 and 9. To enhance usage of these tools, Access provides convenient ways to switch between them. You also saw how to make a subdatasheet open in a Query datasheet view. The subdatasheet feature allows a user to view additional details about related records. To provide insight about the SQL specifications supported by Access, the final section described the differences between the SQL-89 and the SQL-92 query modes supported in Access.

After completing this lab, you should be ready to formulate the queries in textbook Chapters 4 and 9. To enhance understanding of query formulation, you should use the tools presented in this chapter rather than just formulating queries on paper. By using the SQL view, you should also gain confidence about using SQL on DBMSs besides Microsoft Access. You will find that Query Design is particularly useful when formulating queries for data entry forms in the next two lab chapters. The emphasis in these kinds of queries is on the tables needed and how to combine the tables. The visual nature of Query Design allows you to focus on these aspects.

Chapter Reference

The chapter reference summarizes procedures that you practiced. For wizards discussed in the chapter, the procedures highlight important parts of the wizards but do not list all of the steps.

Procedure 1: Using Query Design (Section 3.1.1)

66. Choose **Create → Query Design** in the Queries group to open the New Query window.
67. The Query Design view appears with the Show Table window on top of it. Select tables to include in the new query.
68. To add fields in the query grid, you can use one of the following techniques:
    - Drag and drop field names into the query grid.
    - Click in the first row of an empty column and a small gray arrow appears. Click the arrow and select from the list of field names.
69. If the query grid is full, use the scroll bar to reveal other fields.

Procedure 2: Using the SQL Window (Section 3.1.2)

4. Queries created in Query Design also can be viewed in SQL view. You can toggle to the SQL window by one of these methods:
    - Clicking **View → SQL View**.
    - Clicking on the right mouse button (right-click) when the mouse is over the blue window frame in any view. A shortcut menu appears showing the remaining two view choices.
5. While in SQL view, type the SQL statement. Unfortunately, you do not have the use of the expression builder in the SQL window. Thus, you must type expressions without the help of the expression builder.
6. To use the SQL view for a new query, you must first start a new query in Query Design. Close the Show Table window and toggle to SQL view.

Procedure 3: Copy and Paste an Existing Query (Section 3.2)

70. Copy and paste is a good way to begin a new query that is similar to an existing query.
71. In the **Queries** part of the navigation pane, select the query to be copied. From the ribbon's **Home** tab (or by pointing the mouse on the selected query and clicking the right mouse button), select **Copy.**
72. Then again from the ribbon's **Home** tab select **Paste** (or by pointing the mouse under the selected query and clicking the right mouse button, select **Paste**). Type a name for the new query.

Procedure 4: Using the Expression Builder with Query Design (Section 3.2.2)

73. Create a new query or open an existing query in design view.
74. To use the expression builder, the query first must be named and saved.
75. In the query grid, position the mouse in the appropiate field and click the right mouse

button to reveal a shortcut menu.

76. Click on **Build** and the Expression Builder window appears with the field name in the text area.
77. Type the required expression into the Expression Builder window. When you are finished, click **OK**. If you scroll in the cell, you can see the entire expression that you typed.
78. To create a name for a computed field, type "FieldName:" before the expression where "FieldName" is the name you give to the computed field.

Procedure 5: Setting a Property for a Query Field (Section 3.2.2)

79. You can open the Field Properties window using one of the following techniques:
    - Click in the field (in the query grid) to select it and click **Design → Property Sheet**.
    - Position the mouse in the field and right-mouse-click to open a shortcut menu and select **Properties**.
80. Set the desired properties such as the *Format* property.

Procedure 6: Creating a Join Query (Section 3.2.4)

81. Create a new query.
82. Select two or more tables from the Show Table window.
83. If the tables have a relationship, a line appears connecting the two tables. Double-click on this line to reveal the Join Properties window. A join operator should connect these tables. Confirm that the first choice (join) is selected and click **OK**.
84. To add another table, click **Design → Show Table**. When the Show Table window appears, add the desired table (or tables) and repeat the step above.

Procedure 7: Creating an Outer Join Query (Section 3.2.6)

85. The join operator (as described in textbook Chapters 3 and 4) excludes nonmatching records. The outer join operator includes both matching and nonmatching records.
86. Select two or more tables when the Show Table window appears.
87. If the tables have a relationship, a line appears connecting the two tables. Double-click on the line connecting the tables to open the Join Properties window. Select the second or the third choice. This choice connects the tables with a one-sided join. An arrow from one table points to the other table indicating a one-sided outer join.

Additional Practice

The following problems provide additional practice with Query Design and the SQL window. Parts 1 and 2 use the auto repair order database extended with the *Labor* and the *RepairLabor* tables (see the practice problems of Chapter 2). After formulating a query with one of the tools, toggle to see its representation in the other tool.

Part 1: Query Design

7. List all of the columns of the *Labor* table in which the hourly rate is greater than $20.

8. List the labor description, the standard time, and the hourly rate of the *Labor* table in which the labor description begins with "I". Sort the result in ascending order by standard time and hourly rate.

9. List the labor description, the standard time, and the hourly rate of the *Labor* table in which the hourly rate is greater than $20 and the standard time is less than 0.3 hour or the hourly rate is less than $25 and the standard time is greater than 0.4 hour.

10. List the labor code, the labor description, the standard time (from the *Labor* table), and the actual hours (from the *RepairLabor* table). Include a row in the result if its hourly rate is less than $25.

11. List the labor code, the labor description, and the difference between the labor standard time (from the *Labor* table) and the actual hours (from the *RepairLabor* table). Include a row in the result if the difference is greater than 0.2 hour.

12. List the repair order number, the labor code, the labor description, the time received, and the labor cost (product of the actual labor hours times the hourly rate). Include a row in the result if the time received is in October 2010. In the query result, format the labor cost as currency.

Part 2: SQL View

6. List all of the columns from the *Labor* table in which the standard time is greater than or equal to 0.5 hour.

7. List the order number, the labor code, the labor description, and the labor amount (repair labor hours times hourly rate of labor). Include a record in the result if its labor amount is greater than $30. Rename the labor amount expression as "LaborAmt" in the result.

8. Perform problem (2) using a different join method (see textbook Chapter 9).

9. For each repair order in October 2010, list the count of the labor repairs. The result should include the order number, the date that the repair was received (not the date and time), and the number of labor repairs. (Hint: you need to use a function to list the date without the time.)

10. For each repair order in October 2010, list the count of the labor repairs and the sum of the labor amount. The labor amount is computed as the repair labor hours times the hourly rate of labor. Only include a row in the result if the sum of its labor amount is greater than $55. Rename the computed result column. The result should include the order number, the date that the repair was received (not the date and time), the number of labor repairs, and the sum of the labor amount.

Part 3: University Database Examples

Try the examples in textbook Chapters 4 and 9 on the university database from textbook Chapter 10. You may need to convert some of these examples to the textbook Chapter 10 database. The university database in textbook Chapter 3 differs slightly from the university database in textbook Chapter 10.

Part 4: Order Entry Database Problems

Try the problems in textbook Chapters 4, 9, and 10 (problems 10.1 to 10.21) on the extended order entry database from textbook Chapter 10. For each problem, decide whether SQL or Query Design is an easier tool. You may need to convert some of these problems in textbook Chapters 4 and 9 to the textbook

Chapter 10 database. The order entry database in textbook Chapter 4 differs slightly from the extended order entry database in textbook Chapter 10.

Chapter 4: Single Table Form Lab

Learning Objectives

This chapter provides practice with creating forms for individual tables in Access 2013. After this chapter, you should have acquired the knowledge and skills to

- Create forms using the Form command and the Form Wizard.
- Lay out forms using the Form Design window.
- Toggle between form views.
- Create and modify controls on a form.
- Set form and control properties.
- Use forms for data entry, modification, deletion, and search.

Overview

This chapter takes you beyond the query-building skills of Chapter 3. Although query formulation provides the foundation for building database applications, users do not often manipulate queries directly. Instead, queries are embedded in forms and reports. This chapter provides the basic skills to create and use forms for individual tables. The skills and background emphasized in this chapter provide the foundation to build the more complex forms described in Chapter 5.

Forms play an important role in databases by simplifying how a user enters, edits, and searches for data. Because of the important role of forms, Access provides a number of convenient ways to create forms. The first part of this chapter discusses tools for creating forms and provides practice with the tools. The second part of this chapter focuses on understanding form properties and applying them to the forms created in the first part. The last part of this chapter provides practice with using forms for data entry and searching. Knowledge of form usage will enable you to educate users and improve your form design skills.

4.1 Tools to Create Forms

As it does for tables and queries, Access provides a wizard and a design window to create forms. In addition, Access provides the Form tool for creating default forms. You will find these tools, along with others, listed when you open the New Form window. From this list you have a variety of choices, as explained briefly below. Note that you will not be using the last tool in this chapter:

- Form Design Window: The Form Design window operates similar to the Table Design window and the Query Design window. You can create a form from scratch and toggle between Design view and Form view.
- Form Wizard: The Form Wizard can create simple forms, as described in this chapter. You also can use the Form Wizard to create complex, hierarchical forms, as described in Chapter 5.
- Form: You have fourstyle choices: Columnar, Tabular, Datasheet and Justified. In this chapter, you will primarily use the Columnar style.
- Chart Wizard: This tool creates a form containing a chart.

In the following sections you will build three different forms based on the *Customer* table, the *Vehicle* table, and the *Part* table to demonstrate the methods described above. Later in this chapter you will modify these forms to better fit the requirements of the auto repair shop.

4.1.1 The Form Command

Your first form involves the *Customer* table. This form simplifies data entry when new customers drop off their vehicles for repair. You will use a columnar style for this form as described in the following instructions.

5. Open the Auto Repair Database: Execute Access and open the auto repair database that you created in Chapter 2 and revised in Chapter 3. The Database window should open. If the Tables section is not visible in the Navigation pane, select All Objects from the drop down list in the Navigation page.

6. Open the New Form Window: Select the *Customer* table in the Navigation pane. Choose **Create → Form** from the Forms group of the Ribbon (Figure 1). A new tab appears, with your form in Layout view. You will see the finished form as depicted in Figure 2.

7. The Customer form (Figure 2) has a subdatasheet. Access 2013 automatically adds a subdatasheet when creating forms with tables have a related child table. In this situation, the *Customer* table is the parent table in a 1-M relationship with the *Vehicle* table. Since we do not want the Subdatasheet in the Customer form, you should remove it. You can delete the subdatasheet by switching to Layout view, selecting the datasheet, and then pressing the DELETE key. After deleting the subdatasheet, the Customer form will appear as in Figure 3.

8. Access 2013 provides a convenient way to toggle between form views. Using the view area in the bottom right of the Access window, you can toggle among form view (), layout view (), and design view (). Layout view provides a more visually oriented view than design view although only limited changes can be made in layout view. Design view supports changes to all parts of a form.

9. Save the form as *Customer* when prompted after you click the **Close** button (**X**) on the upper right.

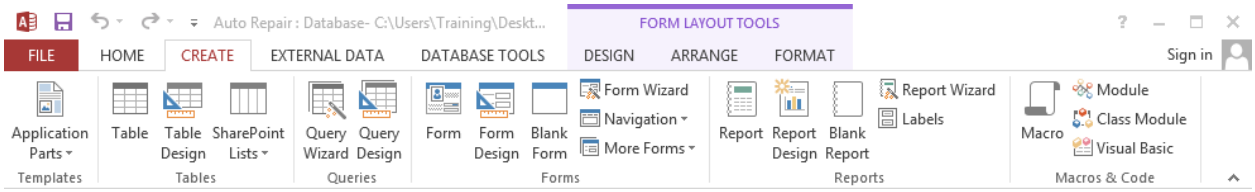


Figure 1: Form Button in the Forms Group of the Ribbon

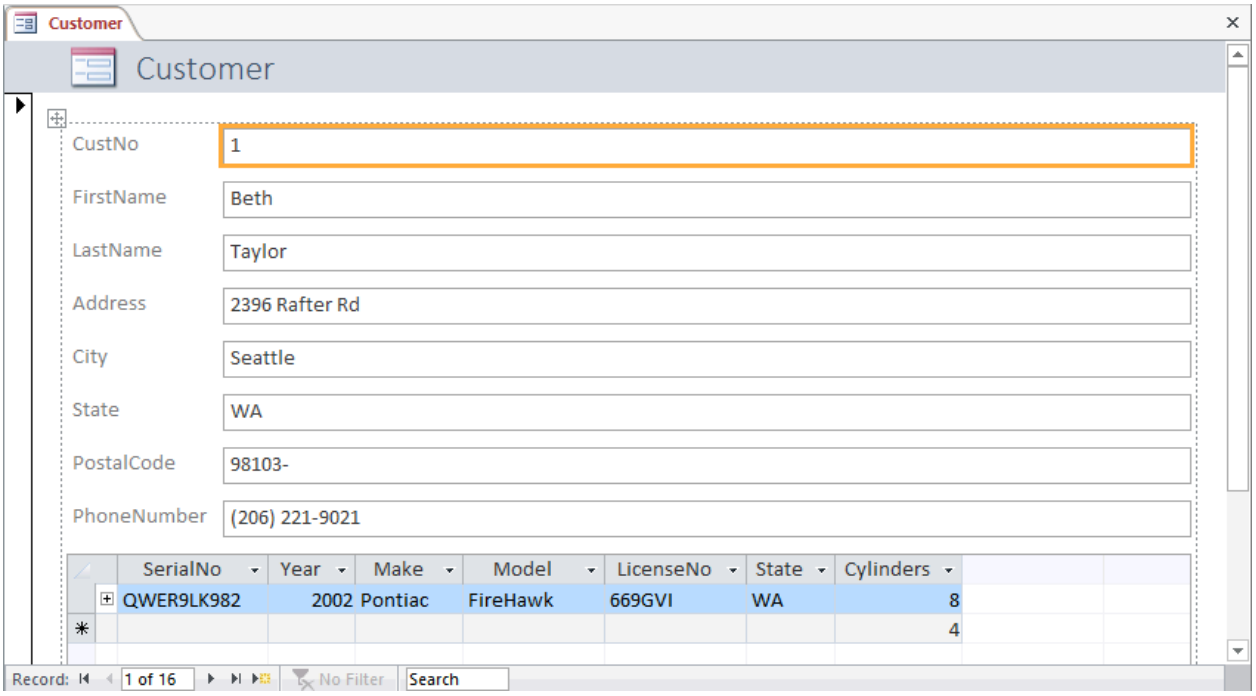


Figure 2: *Customer* Form in Layout View with the Vehicle Subdatasheet

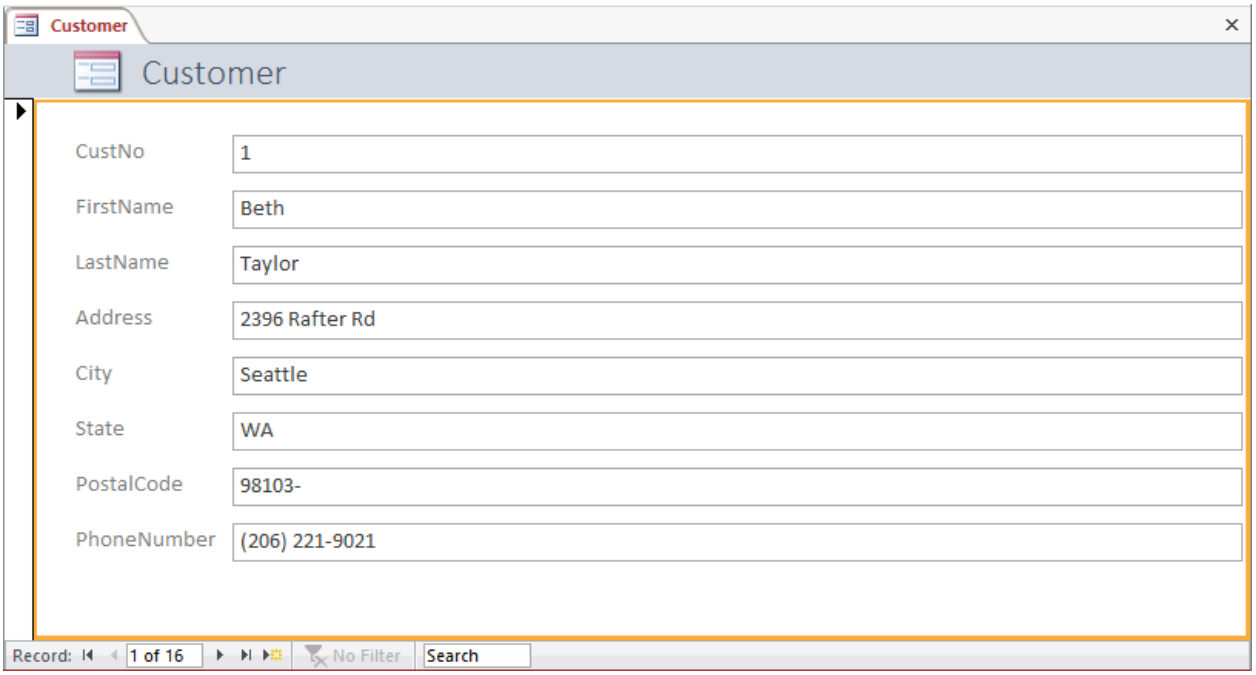


Figure 3: *Customer* Form in Layout View after Deleting the Vehicle Subdatasheet

4.1.2 The Form Wizard

Follow the steps below to create a form for entering vehicle data into the *Vehicle* table. You will create a similar form as before, except that you will use the Form Wizard.

9. Open the New Form Window: Choose **Create → Form Wizard** from the Forms group of the Ribbon.
10. Select Fields to Include: Click the >> button to move all of the fields shown in Figure 4 to the right. Selected fields will appear as in Figure 5. Click the **Next** button when finished.

11. Choose Layout Style: Select "Columnar" in the next window (Figure 6) and click **Next**.

12. Finish the Form Wizard: In the final wizard window (Figure 7), name the form *Vehicle*. Below you are asked if you want to open the form or modify its design. Choose the first option, "Open the form to view or enter information," and click **Finish.** The completed form appears in Figure 8.

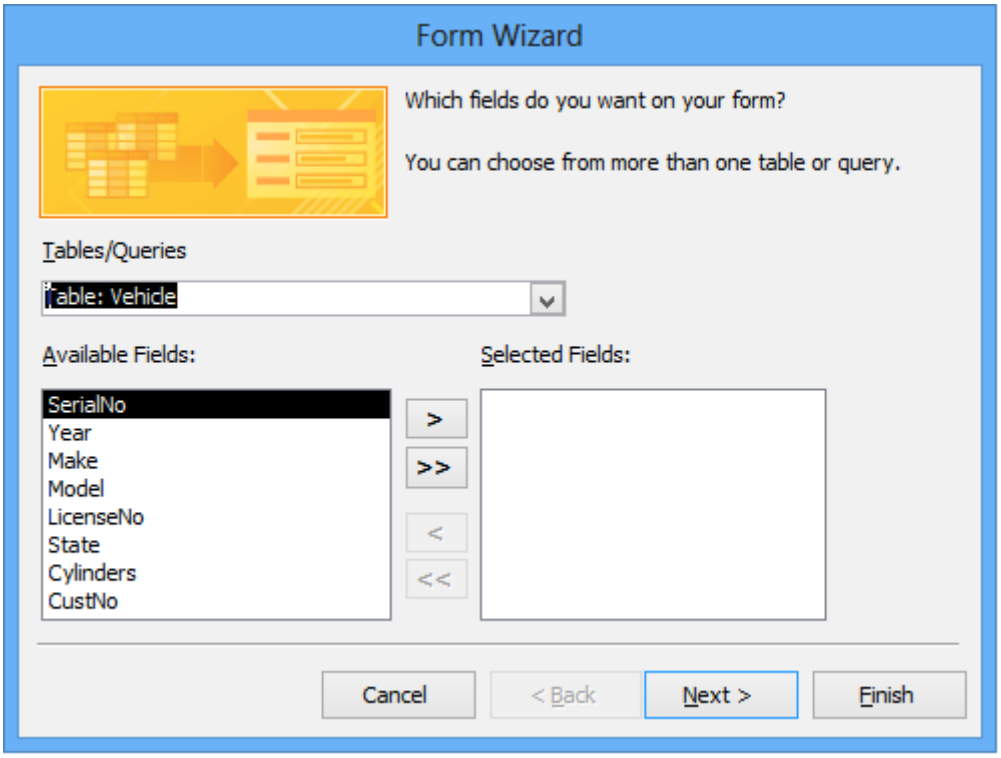


Figure 4: Wizard Window for Choosing Fields

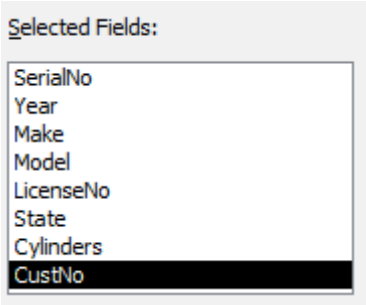


Figure 5: Selected Fields for the New Form

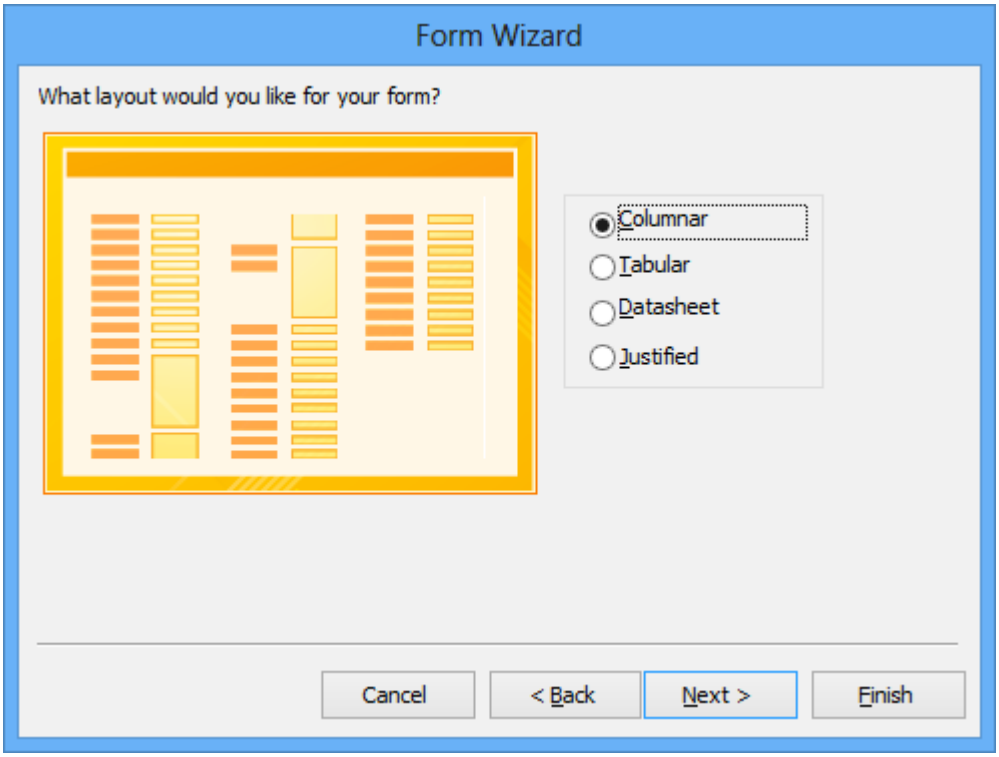


Figure 6: Window to Select the Form Layout

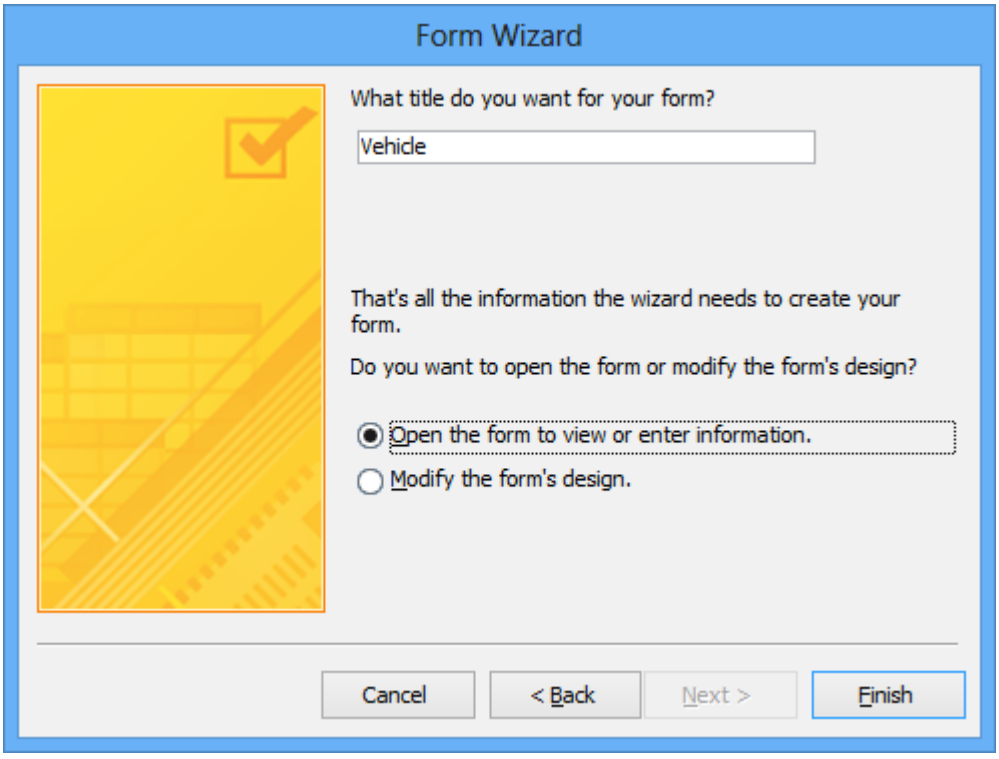


Figure 7: Final Wizard Window

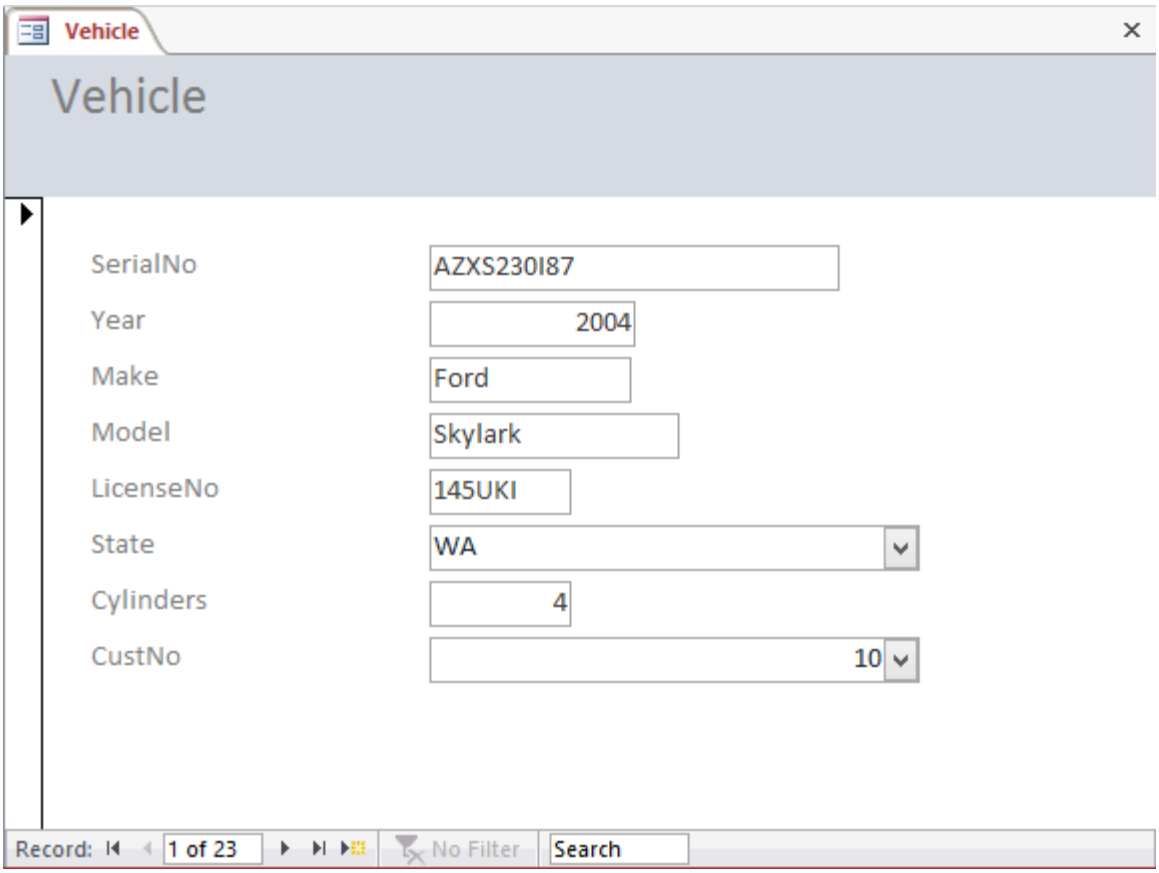


Figure 8: *Vehicle* Form in Form View

4.1.3 Toggling between Views

Before finishing this section on form creation tools, you should understand how to toggle between the different views of a form. In Chapters 2 and 3 you learned about different views for database objects: the design view that utilizes a Design window, an open or run view that displays as a datasheet, and a third view for queries, the SQL view. Similar views apply to forms and can also be accessed by clicking the **Home → View** drop down menu in the Views group of the Ribbon.

A form has two views for designing forms and two views for using forms. Both Design view () and Layout view () allow you to add and modify controls. Layout view is a more visually-oriented view than Design view, but not all tasks can be performed in Layout view. In certain situations, Access displays a message telling you that you must switch to Design view to make a particular change. While viewing a form in Layout view, each control displays real data. Thus, Layout view is very useful for setting the size of controls, or performing many other tasks that affect the appearance of the form.

You can open a form in form view. Figure 8 shows the *Vehicle* form in form view. Datasheet view can be used in a split form (a new feature in Access 2007) or in a subform. A split form provides two views of the same form that are connected to the same data source. You can use the datasheet part of the form to quickly locate a record and the form portion to view or change the record. Datasheet view is useful in subforms to see records related to the record displayed in a main form. Chapter 5 presents details about hierarchical forms and subforms.

Before finishing this section, try toggling among the views. With the *Vehicle* form open (Figure 8), click **Home → View → Design View**. Notice that record navigation does not appear in design view (Figure 9). In Layout view, however, record navigation is available as shown in Figure 10. Layout view differs only slightly from form view. In Figure 10, the *SerialNo* field has a yellow border in layout view and a grey

border in form view (Figure 8). You can toggle among the other views using the **View** drop down menu, the **View** toolbar, or the right mouse click. When you are finished, close the *Vehicle* form.

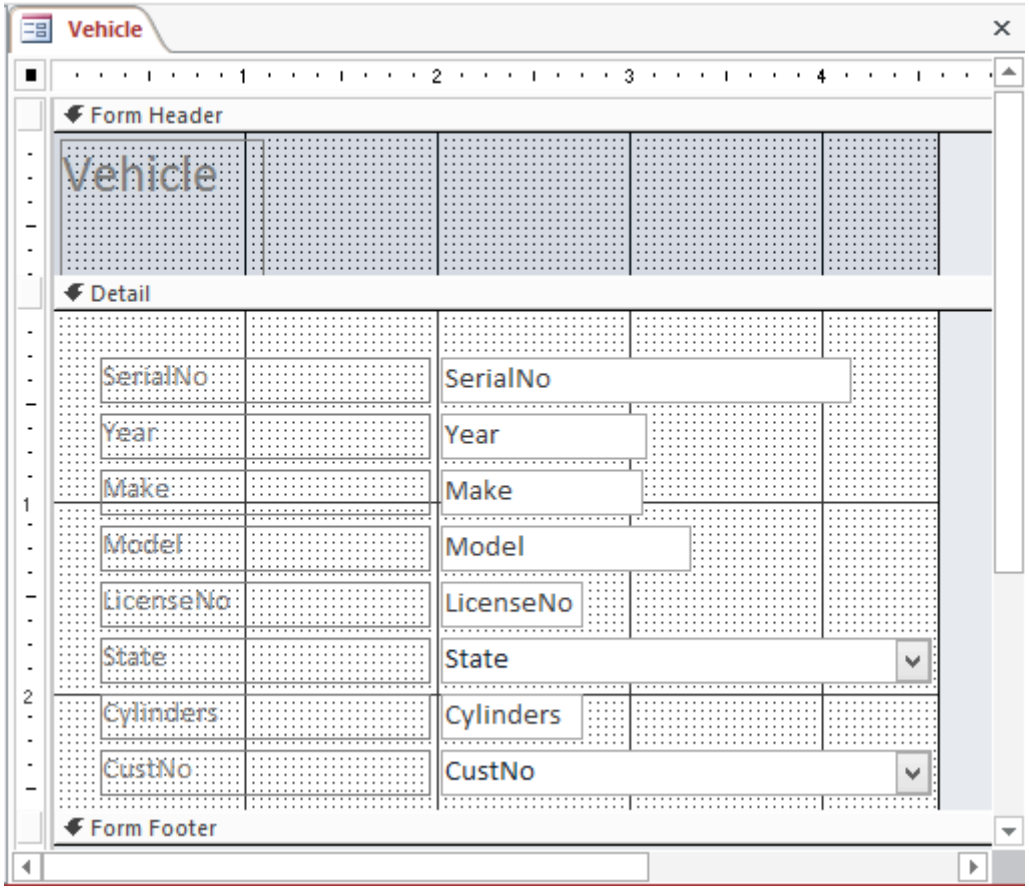


Figure 9: *Vehicle* Form in Design View

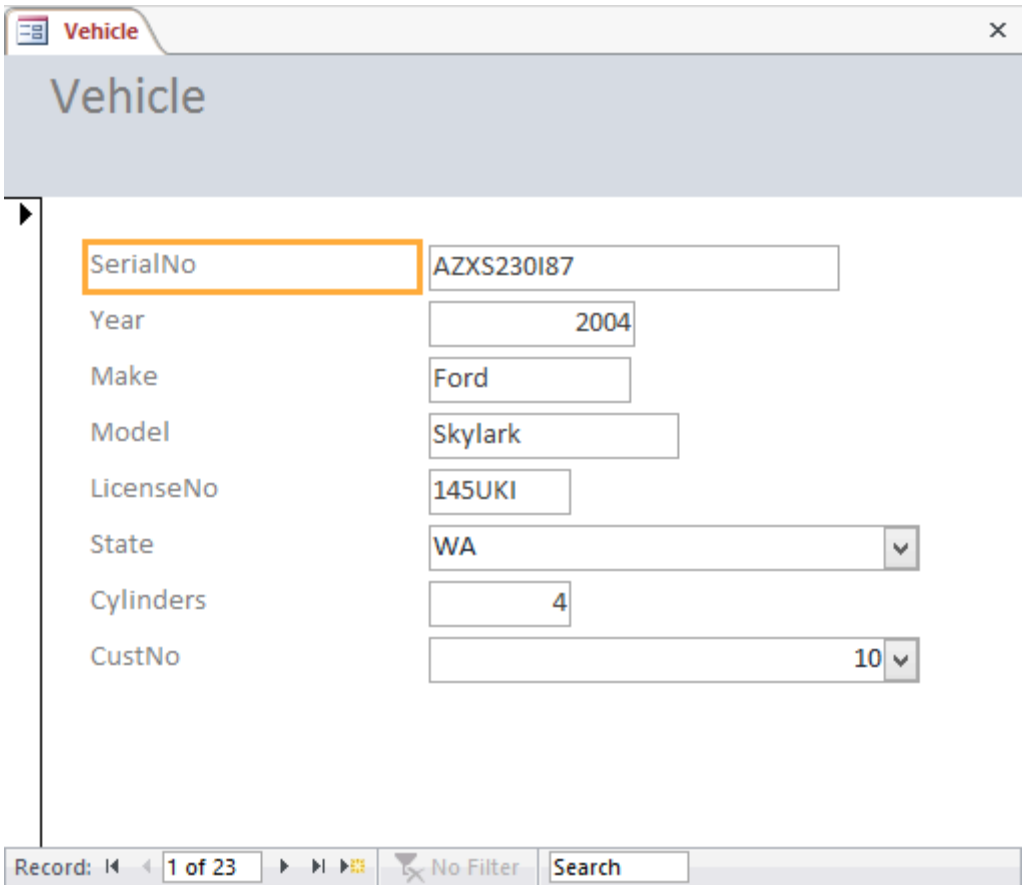


Figure 10: *Vehicle* Form in Layout View

4.2 Working with Form Controls

When you create a form using the Form Wizard or Form tools, Access places controls on the form. Controls allow you to manipulate data or initiate actions. Common controls are textboxes for text and numeric data, check boxes for true/false data, and combo boxes for selecting a choice from a list. Most controls that Access places on a form are bound to fields from a table. Binding a control means identifying a source of data for the control. For example, a textbox bound to the *Customer.CustNo* field means that the textbox displays data from this field and can be used to enter data into this field. Other controls such as command buttons initiate actions rather than bind to database fields. In this chapter, you will mostly use bound controls. In Chapter 8, you will create unbound controls to initiate actions.

When working with textboxes, you should understand the relationship between a textbox and its associated label. When a textbox is placed on a form by the Form Wizard or dragged from the field list in the form window, a label control is automatically attached to identify the textbox. By default, the label contains the field name (since the textbox is bound to the table field) unless another name was entered in the *Caption* field property of the database field. In contrast, an unbound textbox can be created from the toolbox. The new textbox will say "unbound" inside until you bind a field to it, and the accompanying label will have no name until you type one inside. Unbound textboxes will be demonstrated in Chapter 5. A label also can be created by itself from the toolbox for form titles, comments, or labeling other controls.

The same concept of binding exists for combo boxes and check boxes. As you may remember from Chapter 2, you used the Lookup Wizard for foreign key fields in tables such as the *Vehicle* and the *PartsUsed* tables. When controls bound to these fields are added to a form, they will be combo boxes rather than textboxes. Any choice made by a user from the combo box will be made in that table's field. This

binding idea also holds true for check boxes that represent yes/no fields in a table. When a user clicks a check box, a "Yes" is recorded in the table's field.

In the following section you will create the *Part* form from scratch using the Form Design window. This practice will provide you experience working with textboxes. You also will be using the label control to title the forms you have created. Later in the chapter, you will be modifying a combo box in the *Vehicle* form.

4.2.1 Design View

Follow the steps below to create a form for entering data into the *Part* table. You will create a similar form as the previous two, except that you will use the Form Design window to create the form from scratch.

1. Open the New Form Window: Choose **Create → Blank Form** from the Forms group of the Ribbon. Select "Design View" from the View list. An empty Form Design window appears (Figure 11).

2. Place Fields: Access does not know at this point from which tables you want to display and edit data. Choose **Form Design Tools → Design → Add Existing Fields.** The Field List on the right displays a list of each local or linked table. If you click the plus symbol next to the name of a table, Access expands the list and displays the name of every field in that table. To open th*e Part* table field list, click the plus symbox next to it. You can click on a field name in the Field List and drag and drop it onto your form as shown in Figure 12.

> Technical Note: If the fields do not fit comfortably, you can extend the size of the form. To extend the size, move the cursor to the bottom of the form rectangle until the cursor changes to a double arrow with a thick line. You then can drag the mouse to extend the form.

3. Adjust Horizontal and Vertical Spacing: Your form fields may be too tightly spaced, as shown in Figure 13. Using the choices in the **Arrange → Size/Space** of the Sizing & Ordering group (Figure 14), you can increase, decrease, or make the vertical spacing equal. Before using this command, select all fields in the Design window using the mouse and the **Shift** key. With all fields selected, click **Increase Vertical**. Repeat the same procedure using the other choices in the Position group. Figure 15 shows the fields after increasing the horizontal and vertical spacing two times.

4. Align the Fields: The fields in Figure 13 are evenly aligned because you added them to the form at the same time. If you add a new field, the alignment may not be even. You can change the alignment by using the commands in the Control Alignment group in the ribbon. To align controls, select the form fields to align (use the **Shift** key) and then choose the kind of alignment from the Control Alignment group. You can align fields to the left, right, top, bottom, or grid.

5. Move Fields Individually: If the commands in the Control Alignment and Position groups do not suffice, you can move fields individually. To move a control with the mouse, highlight the field by clicking on it and then drag the control to move it. Notice that a bound control and its label move together as long as you drag it at any point except at its anchor in the upper left corner. You also can move a label and bound control separately. To move a label separately, select the control and then drag its anchor (filled rectangle in upper left corner ■) of the control. You can drag the label or bound control separately using the appropriate anchor.

6. Group Fields to Move as a Unit: Select the fields (textboxes or labels) you wish to move as a unit and click **Arrange → Size/Space → Group**  in the Sizing & Ordering group of the Ribbon. This method is a good way to move a group of fields more toward the center of the form. When you are finished, you can click **Arrange → Size/Space → Ungroup**  .

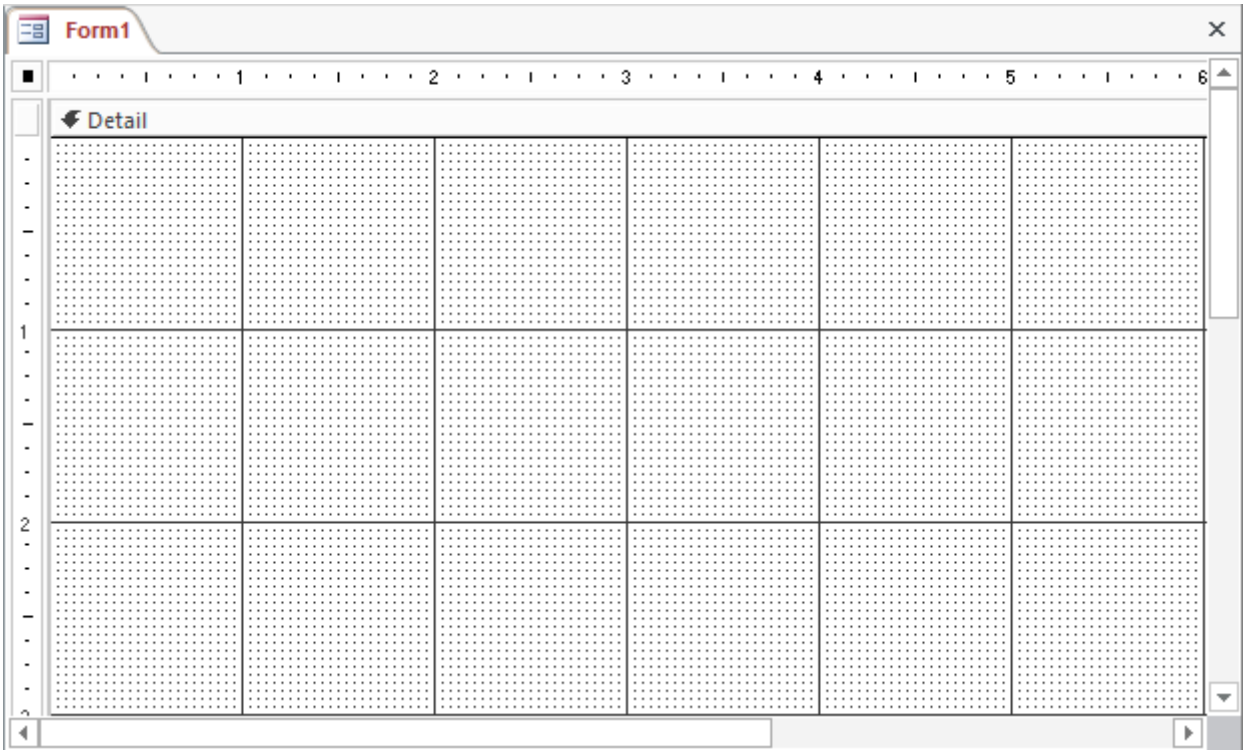


Figure 11: Empty Form Design Window

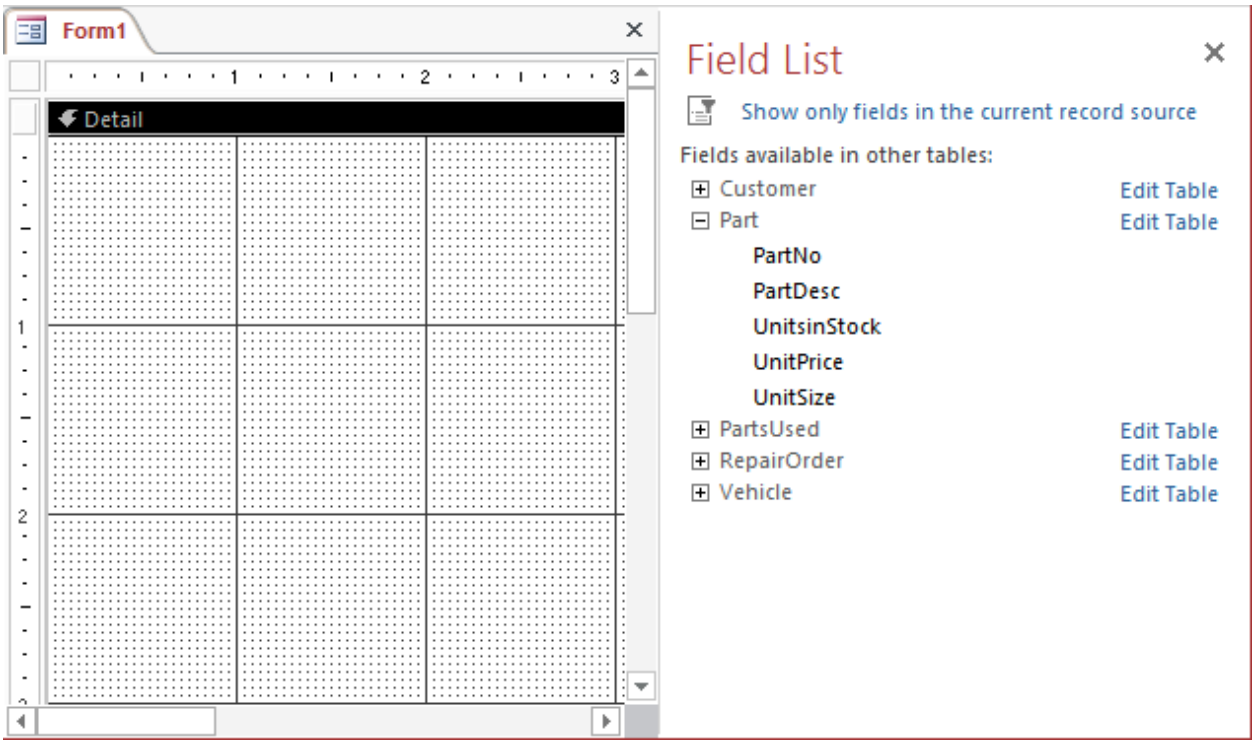


Figure 12: Empty Form Design Window with *Part* Table Field List

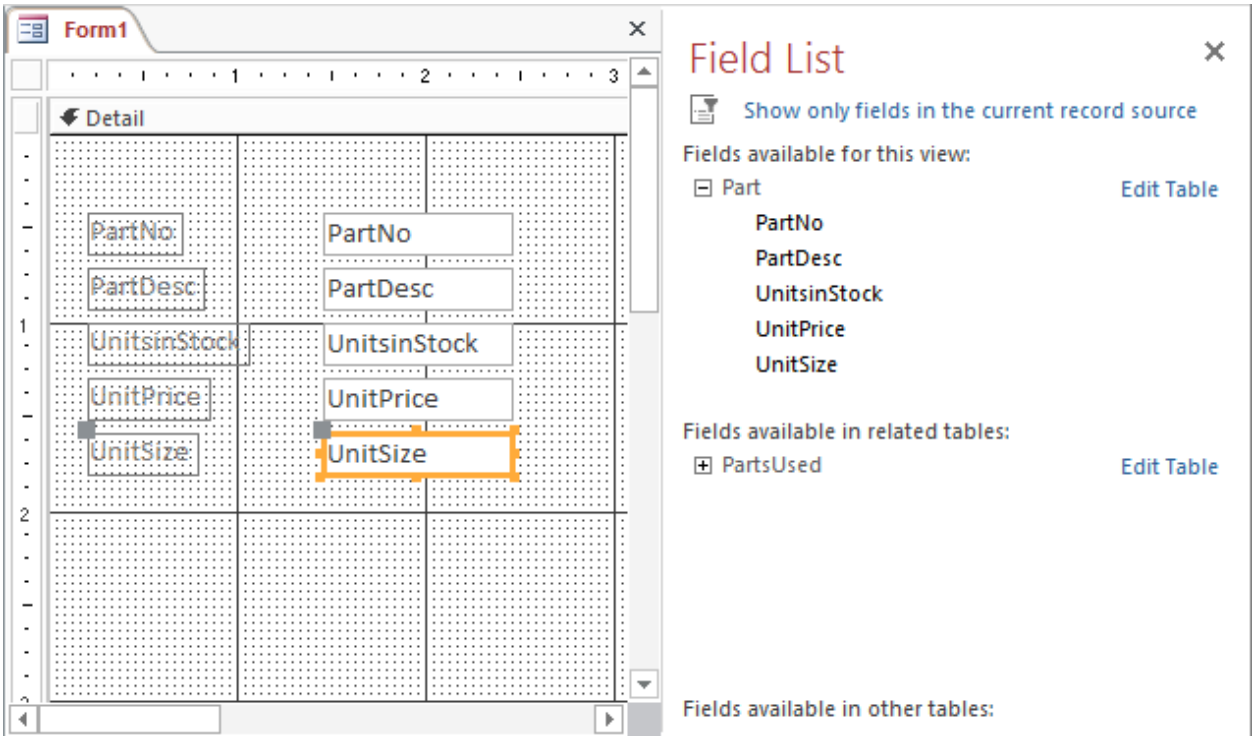


Figure 13: Initial Placement of Fields

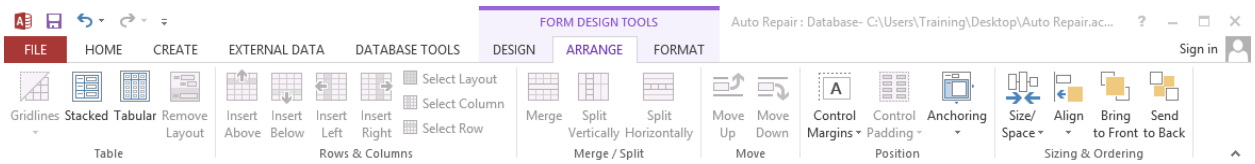


Figure 14: Arrange Tab Choices for Manipulating Controls in Design View

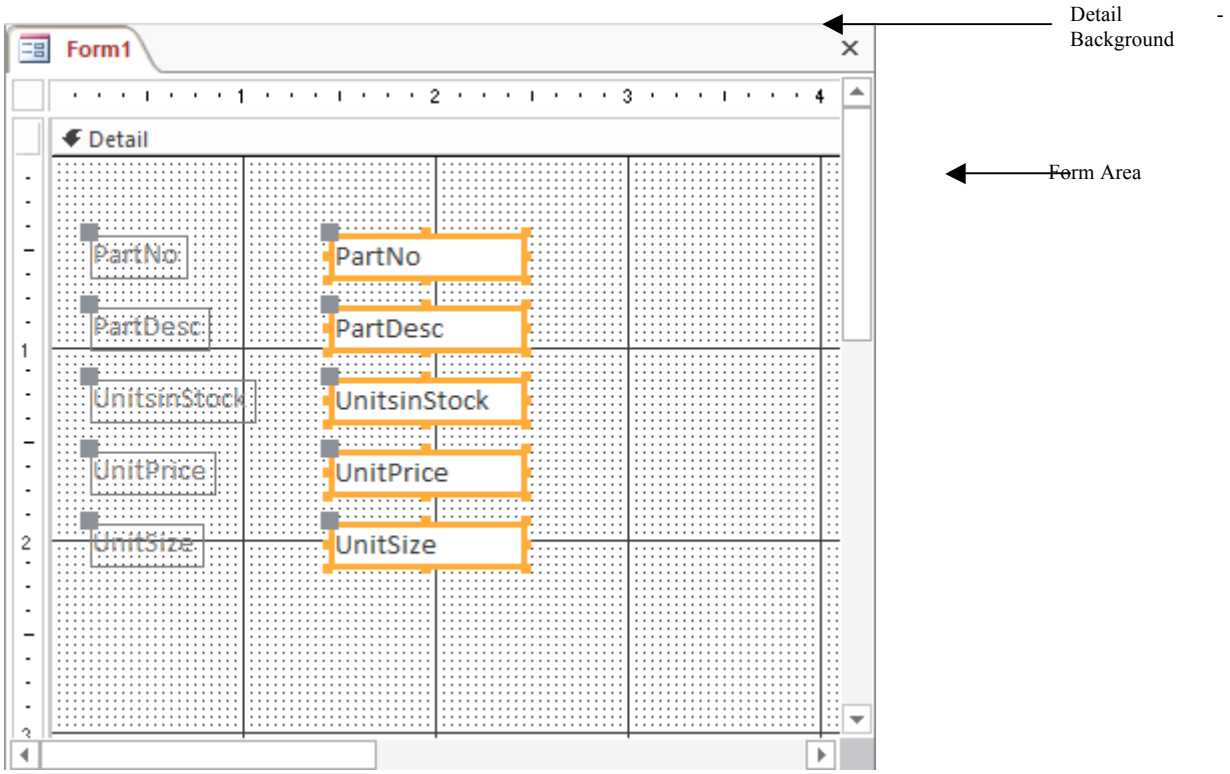


Figure 15: Form Fields after Adjusting Horizontal and Vertical Space

7. Select the Background Color and/or Style: There are three ways to set the background color or style of the form:

   - Select the detail (grey, dotted/grid backgound) section and choose a color in the *Back Color* property, see (a) below and Figures 16 and 17;
   - Select the *Picture* property from the Form Property Sheet. To open the Form Property Sheet, double-click anywhere in the nongrid area. In the **Format** tab (Figure 18), click the *Picture* property to reveal the ellipsis (**…**) button, and click on it.
   - Click **Format → Background Image** in the Background group of the Ribbon and then choose the desired image file from the file dialog. For this exercise, to keep the *Part* form consistent with the other forms so you will not use an image.

8. Add Title: While still in Design view, select **Design → Title** from the Controls group of the Ribbon to add a title. Change the title name to "Part Form" as shown in Figure 19.

9. Toggle to Form View: Change to Form view to see how the form appears (Figure 20). When you are finished, close the form and save changes. When prompted to name the form, type *Part*.

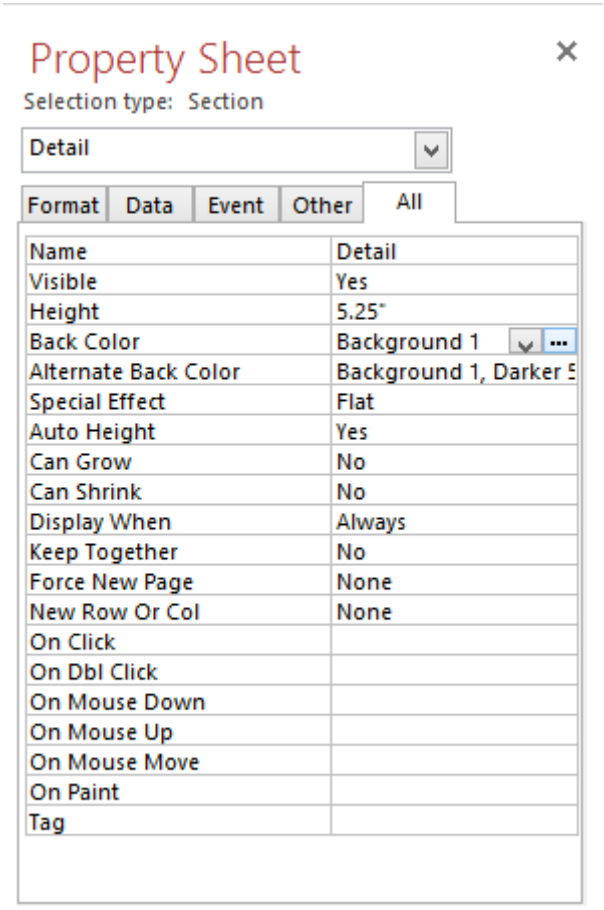


Figure 16: Property Sheet for the Detail Section

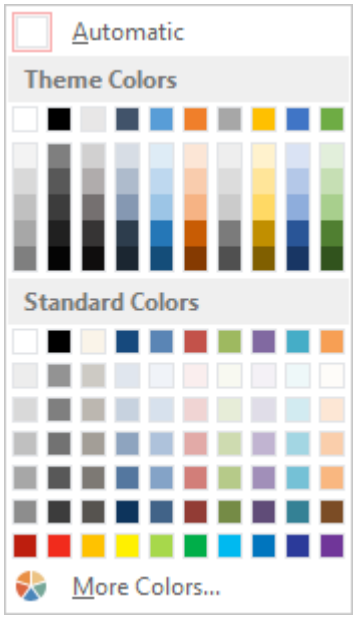


Figure 17: Color Palette

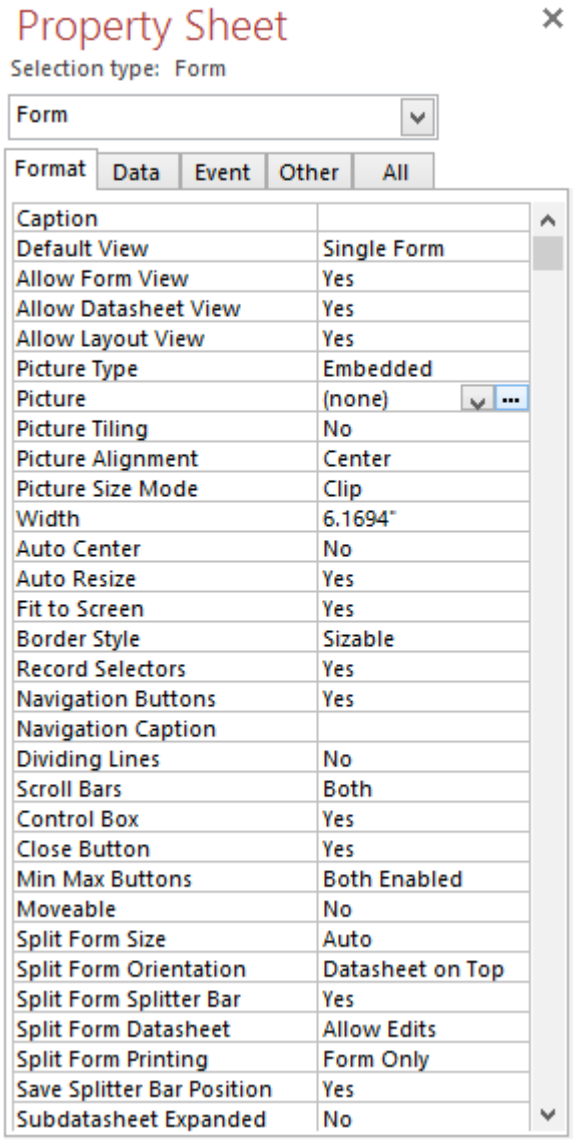


Figure 18: Format Tab in the Property Sheet for the Form

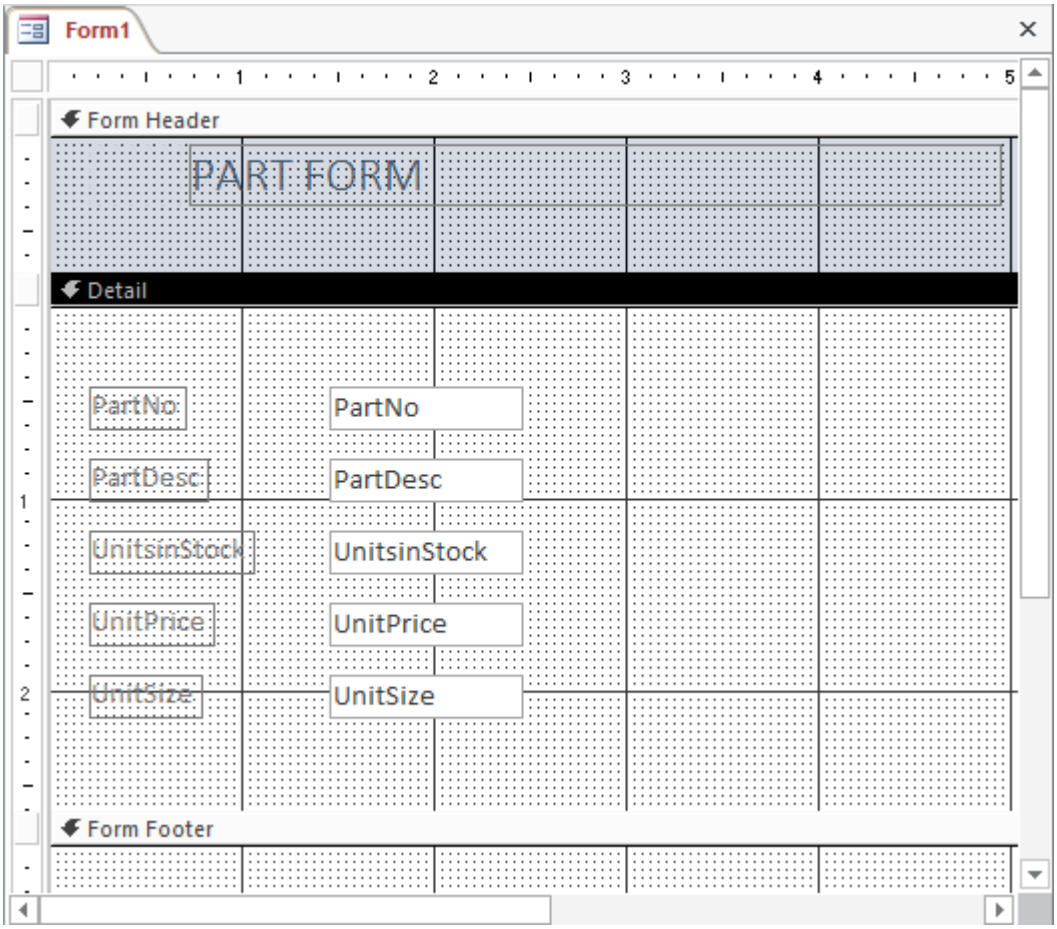


Figure 19: *Part* Form in Design View with Title

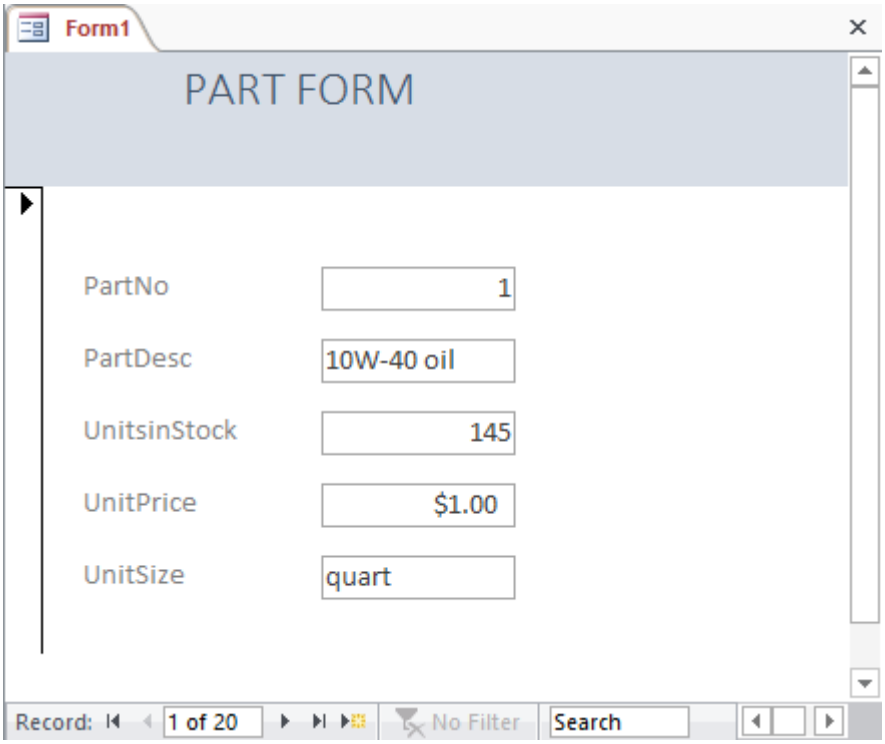


Figure 20: Completed *Part* Form in Form View

4.2.2 Tab Order

The tab order is an important modification that can be made after a form is created. The tab order is important when navigating inside a form. A user has three ways to navigate between controls: clicking the mouse on a control, using the **Tab** key, or using the **Return** key (**Enter**) when completing an entry. When using the **Tab** key or the **Return** key, the sequence or order of controls on a form is extremely important. This order is referred to as the tab order. For example, it would be inconvenient for a user to enter a customer's first name but have to tab far down the form and back again to enter the last name. Therefore, you must pay close attention to the tab order.

You can easily test the tab order for the *Customer*, the *Vehicle*, and the *Part* forms. From the Database window, click the **Open** button for the *Customer* form to open it in form view. Beginning with the *FirstName* textbox, tab through the controls. As you should find, they are in consecutive order. Therefore, no modification is necessary. Close the form and test the tab order of the other two forms. You should find that they do not require modification either.

You can change the tab order of a form by accessing the Tab Order window. In design view, click **Design → Tab Order…** in the Tools group. When the Tab Order window appears, follow the simple directions to rearrange the tab order of the controls. Figure 21 shows the Tab Order window for the *Part* form.

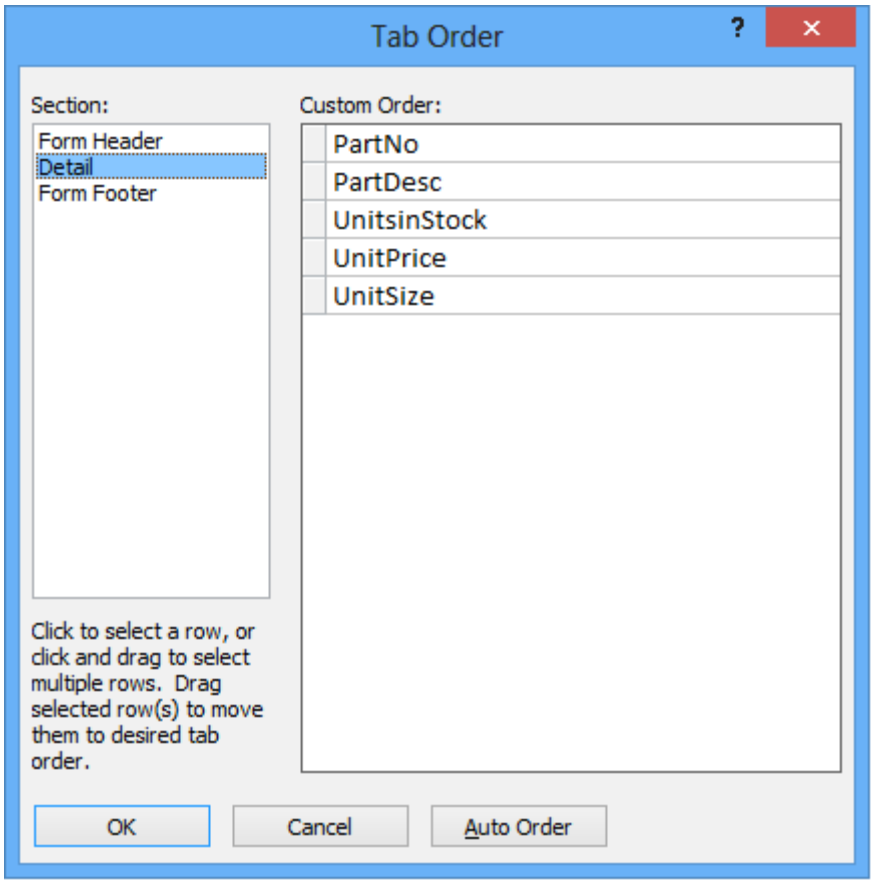


Figure 21: Tab Order Window

4.3 Understanding Form Properties

Forms have properties at the form level, the control level, and the section level (e.g., the detail section). These properties allow you to customize the appearance of a form and change the way that controls react in response to a user's actions. This section describes some important form and control properties.

A property is set inside the Properties window for the specific control or section. A Properties window contains the list of properties specifically for that section or control along with preset default settings. You can open a Properties window in three ways:

- Double-click the component.
- Select the component and click the right mouse button and select **Properties**.
- Select the component and then click **Design → Property Sheet**.

4.3.1 Allow Properties

A form has a number of actions available to a user such as adding data, editing data, deleting data, and viewing existing data. A form may be created to perform any combination of these tasks. Depending on requirements, you may restrict or set limits on actions allowed, such as being able to add data but not delete. This ability protects against a careless user deleting records by mistake. You also can create a form to be read-only, thus preventing a user from making any changes to the data on a form. You may limit actions available to all users by setting a series of allow properties at the form level.

The allow properties do not provide a fine level of control over form usage. For example, the allow properties do not permit one group of users read-only permission while another group of users has addition-only permission. Unfortunately, Access 2013 databases do not support user permissions. If you want to use user permissions, you will need to store your data on a server database (such as Microsoft SQL Server) and manage user permissions through the server DBMS. Alternatively, you can convert your database to Access 2003 file format. After converting a database to Access 2003 file format, you can click the **Manage Users & Permissions** button in the **File → Info** item.

The allow properties are set in a form's Properties window. Depending on these property settings, certain manipulation actions may be restricted. Remember that form view is the view that a user sees. This section explains the significance of setting the allow properties. Table 1 summarizes the properties at the end of the section.

- *Allow Filters*: You can use this property to specify whether records in a form can be filtered. You use a filter to display a subset of records that meet specific conditions. For example, in the *Vehicle* form, you can use the filter to display only Toyotas. For security reasons, a filter may be used to restrict to users records that contain private information. If the property is set to "Yes", records can be filtered. If the property is set to "No", records cannot be filtered and the filter options in the Sort & Filter group of the Home tab are disabled. However, setting the *Allow Filters* property to "No" does not have any effect on the *Filter* and the *Filter On Load* properties. These properties may still be used to set and remove (turn off) filters.

- *Allow Edits*: You can use this property to specify whether a user is allowed to edit existing records. If the property is set to "Yes", the user may edit records. If the property is set to "No", the user cannot edit existing records. In addition, the **Home → Delete** drop down menu in Records group is disabled. If you want to keep a user from changing data in a particular control, then you should set the *Enabled* and *Locked* properties, as described later.

- *Allow Deletions*: You can use this property to specify whether a user is allowed to delete existing records. If the property is set to "Yes", the user may delete records. If the property is set to "No", the user cannot delete existing records although records may still be viewed or edited. Also if this property is set to "No", the **Home → Delete** drop down menu is disabled.

- *Allow Additions*: You can use this property to specify whether a user is allowed to add a record to the database. If the property is set to "Yes", the user may add new records. If the property is set to "No", the user cannot add new records although records may still be viewed or edited.

- *Data Entry*: You can use this property to specify whether a form initially allows data entry. If the property is set to "Yes", the form will open showing a blank record. If the property is set to "No", the form will open showing existing records. However, the *Allow Additions* property must be set to "Yes" for the *Data Entry* property to be active. If the *Allow Additions* property is set to "Yes" and the *Data Entry* property is set to "No", the user may click the **Home → New** button in the Records group to perform data entry.

Table 1: Summary of Allow Properties

Allow Filters	Specifies whether records in a form can be filtered. If set to "Yes", records can be filtered. If set to "No", records cannot be filtered.
Allow Edits	Specifies whether records can be edited. If set to "Yes", the user may edit records. If set to "No", the user cannot edit existing records.
Allow Deletions	Specifies whether existing records can be deleted. If set to "Yes", the user may delete records. If set to "No", the user cannot delete records.
Allow Additions	Specifies whether records may be added to the database. If set to "Yes", records may be added. If set to "No", records cannot be added.
Data Entry	Specifies whether a form initially allows data entry. If set to "Yes", the form will open showing a blank record. If the property is set to "No", the form will open showing existing records.

4.3.2 Other Important Properties

There are a number of view properties (*Allow Form View*, *Allow Datasheet View* and *Allow Layout View*) that determine the views allowed for the user. Each property has "Yes" and "No" choices indicating whether the user should be allowed to use the form in that view. As explained already, having the ability to toggle among views is very useful during the process of form development. However, you may not want the user to have all of these options when the form is finished. Therefore, you can set a view property at the form level to limit the views available to all users. In run-time, a user may be allowed to view the form in form view, datasheet view, or both.

Default values are preset values determined by the form developer that appear when a user adds records in a form. You already have used the *Default Value* property for table fields in Chapter 2. A control on a form, such as a textbox, has its own list of properties including a *Default Value* property. If a textbox on a form is bound to a table field, the default value previously set for the field during table design remains in place although the form control property appears empty. Setting a *Default Value* property for a form control overrides the *Default Value* previously set during table design. If you decide to set the *Default Value* property for a control, it only affects new records that are added to the database. Thus, if you change the *Default Value* form field property, the change is not applied to the existing records.

As with the *Default Value* property, you have had experience with the *Validation Rule* property for fields and tables in Chapter 2. Similar to the *Default Value* property, a *Validation Rule* previously set for a field during table design remains effective if the control property is empty. However, a *Validation Rule* is applied differently depending on where it is set. Validation rules you set for form controls are applied when you enter or edit data. Validation rules set for records (as a table property) are applied when you go to another record. When you set *Validation Rules* for both a table field and a form control, both sets of rules are applied when you edit data.

An important point to keep in mind is that you should set the *Validation Text* property whenever you set the *Validation Rule* property. Otherwise, Access will display its standard error message saying the validation rule was violated instead of a more precise message for the particular error.

4.4 Setting Form Properties

In the following sections you will be setting properties for the *Customer*, the *Vehicle*, and the *Part* forms. However, due to the number of available properties, this section cannot provide practice with all properties. If you are curious about a property, you can click on it anytime and press **F1** on the keyboard to read the help documentation about it.

You should have the Database window open with the Forms section in the Navigation pane containing three forms. Select a form and open it in design view before starting the procedures in this section.

4.4.1 Revising the Customer *Form*

You will be setting various properties for the *Customer* form. You will set font properties for the label control containing the form title in the header, change the size of the form background, and set form properties.

1. Open the Property Sheet for the Label Control in the Header: After opening the Customer form in design view, you can open the Property Sheet by double clicking on the control or selecting the Properties command after right clicking the control.

2. Change the *Name* Property: Replace the existing value in the *Name* property with the new name "FormTitle".

3. Change the *Font Size* Property: The current font size is too small so you will increase it. Locate the *Font Size* property toward the middle of the Properties window and click inside. Next, click the arrow to display a list of font sizes. Select the size "14".

4. Change the *Font Weight* Property: Tab down to the next property and click inside. Next, click the arrow to display a list of font weight choices. Select "Semi-bold" as the *Font Weight* value.

5. Toggle to Form View: You can see how the form title appears in form view (refer back to Figure 20 to see the title for the Part form). You may use any of the methods already explained to open form view. If you are not satisfied with the title's appearance, switch to Design view and modify properties in the Property Sheet of the label control in the header section. Otherwise, proceed to the next steps.

6. Expand the Form Background: You may have noticed in form view that the background does not cover the entire form. Therefore, you need to expand the property settings for the background height and the form width.

    - Expand Background Height: In the detail section, click on the "Blends" background and then click the right mouse button to select its Property Sheet. This opens a Property Sheet for the Detail section (Figure 22). Change the value of the *Name* property to "Background". Next, move down to the *Height* property and change the height to "4.5" (Figure 23).

    - Expand the Form Width: Double-click outside the grid area to open the Property Sheet for the form. Change the *Width* property to "7.5". Note that for some background images, the image width will not expand as the form's width expands.

    - Toggle to Form View: See the expanded form in form view (Figure 24). If you are satisfied with the result, toggle back to design view.

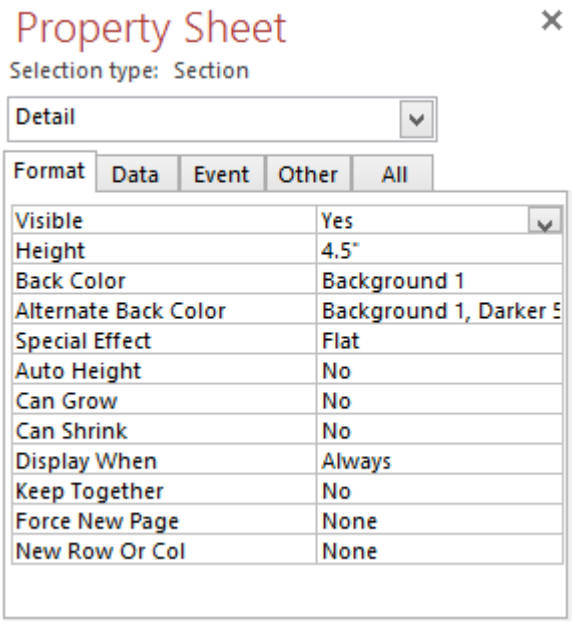


Figure 22: Property Sheet for the Detail Section

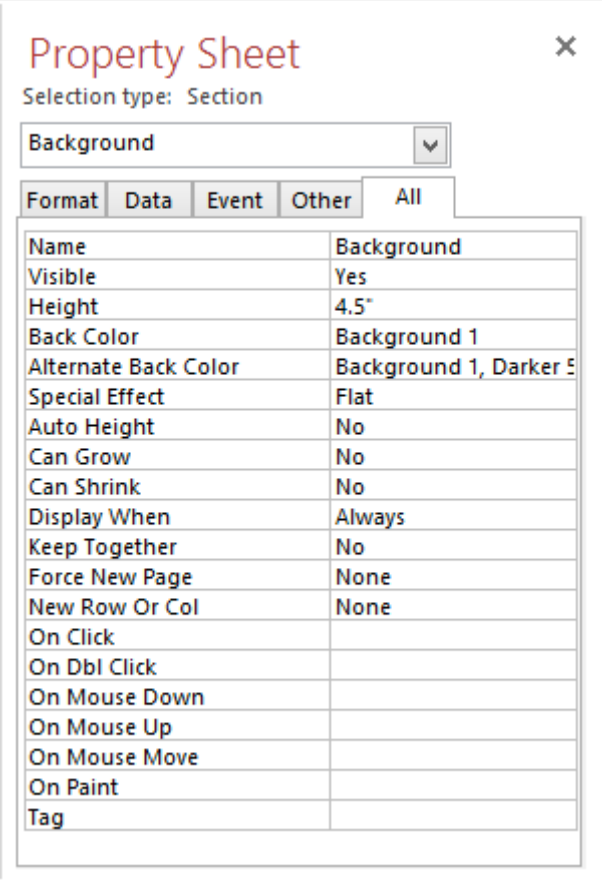


Figure 23: Detail Renamed to Background

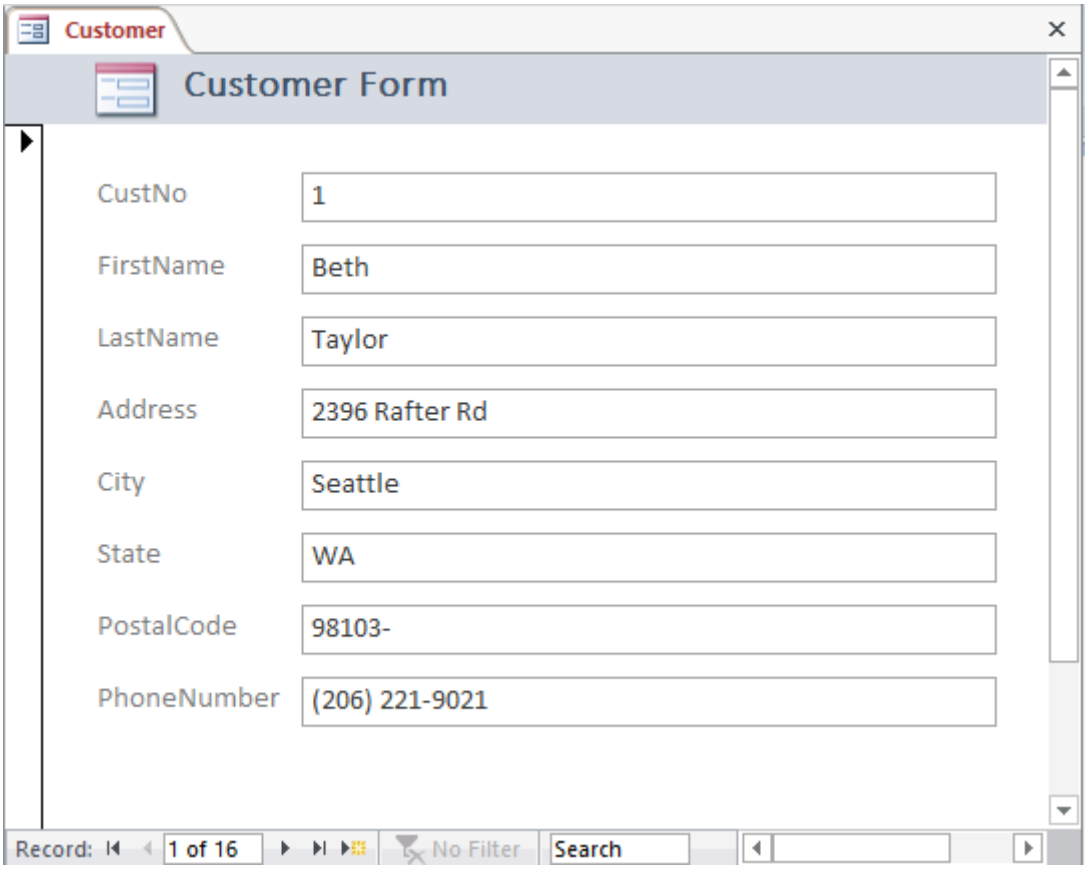


Figure 24: Revised *Customer* Form

7. Set Form Properties: Open the Property Sheet for the form, if it is not open already, using any method previously described. Next, select the All tab if it is not open. Use Table 2 as a guide to set property values. After changing the properties, close the form and save the changes when prompted.

Table 2: *Customer* Form Properties and Settings

Property	**Setting**
Allow Filters	Keep the default, "Yes".
Allow Form View	"Yes"
Default View	Click the arrow in the *Default View* property area to display a list of choices. Select "Single Form". This allows the user to view the form only showing one record at a time.
Allow Datasheet View	Set these properties to "No" to prevent the user from viewing the form in this mode.
Allow Edits	Keep the default, "Yes".
Allow Deletions	Set to "Yes".
Allow Additions	Keep the default, "Yes".
Data Entry	Set to "No".

4.4.2 Revising the Vehicle *Form*

You will be setting properties for the *Vehicle* form similar to the way you did for the *Customer* form. You will set font properties for the label control containing the form title in the header, change the size of the form background, and set form properties. In addition, you will modify the *CustNo* combo box by adding descriptive fields to the combo box list.

1. Set *Font Size* and *Weight* Properties: Repeat the steps performed for the *Customer* form to set the *Font Size* and the *Font Weight* properties for the form title in the header. Also change the *Name* property of the label to "FormTitle".

2. Expand the Form Background: Repeat the steps performed for the *Customer* form to expand the form background and change its name. Return to design view when you are finished.

3. Add Fields to the *CustNo* Combo Box: In Chapter 2, you used the Lookup Wizard for the *CustNo* field in the *Vehicle* table. Therefore, when the Form Wizard created the *Vehicle* form, the *CustNo* field control was made a combo box control rather than a textbox. A combo box allows the user to display a list of choices; in this instance, customer numbers. However, a customer number alone is not enough choice for a user. There also should be some descriptive information to help the user differentiate between the customer numbers. To remedy this, you will include the first and the last names in the list by adding those fields from the *Customer* table.

    - Open the Property Sheet for the Combo Box. Click in the *Row Source* property area and click the ellipsis (**…**) button to open Query Design.
    - Drag the *FirstName* and the *LastName* fields down into the query grid. In the *LastName* column, click in the *Sort* row and click the arrow to choose "Ascending".
    - Close the Query Design window. When prompted, click **Yes** to save these changes.

4. Set Combo Box Properties: Use Table 3 as a guide to set other properties of the combo box.

Table 3: *CustNo* Combo Box Properties and Settings

Property	**Setting**
Column Count	"3"
Column Heads	"Yes"
Column Widths	"0.5"
List Width	"2.5"

5. Toggle to Form View: See how the form looks in form view (Figure 25). If you are satisfied with the result, toggle back to design view.

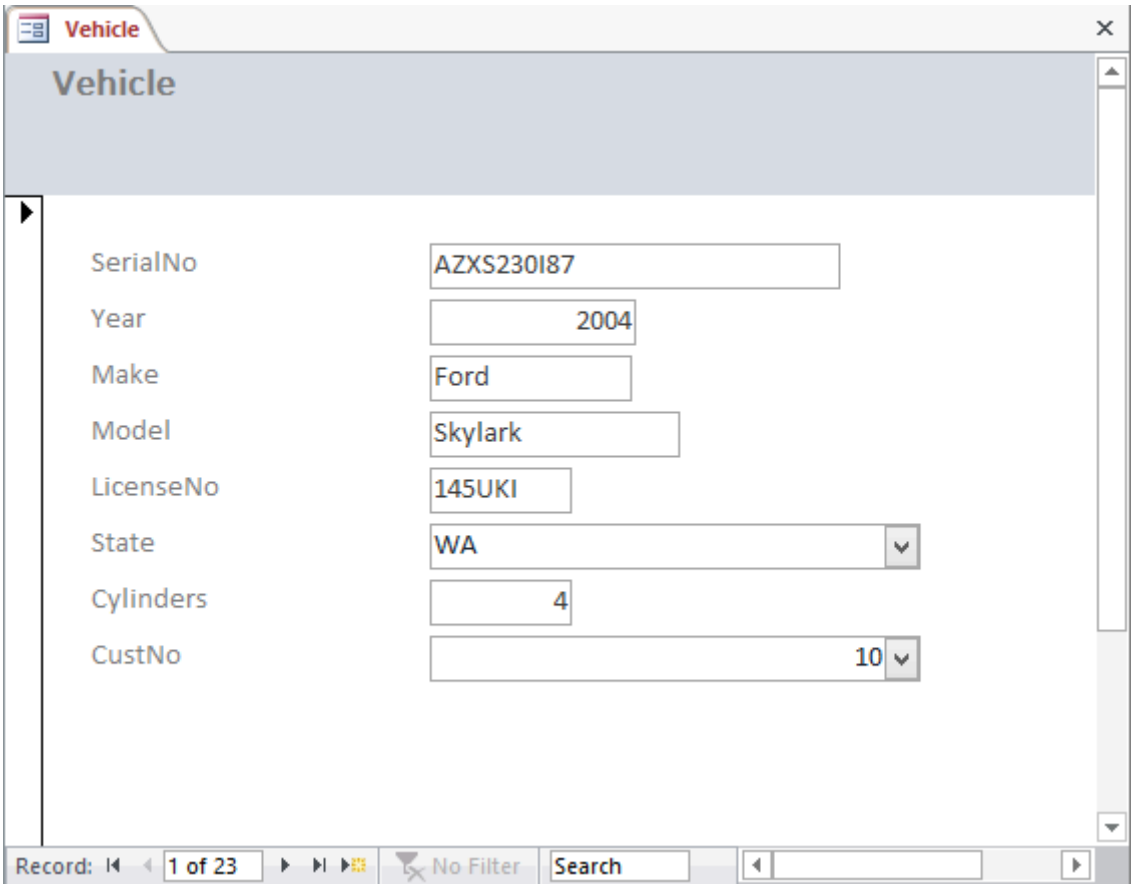


Figure 25: Revised *Vehicle* Form

6. Set Form Properties: Open the Properties window for the form with the **All** tab selected. Use Table 4 as a guide to change selected properties. After changing the properties, close the form and save the changes when prompted.

Table 4: *Vehicle* Form Properties and Settings

Property	**Setting**
Allow Filters	Keep the default, "Yes".
Allow Form View	"Yes"
Allow Datasheet View	"Yes
Allow Edits	Keep the default, "Yes".
Allow Deletions	Set to "Yes".
Allow Additions	Keep the default, "Yes".
Data Entry	Set to "No".

4.4.3 Revising the Part *Form*

As with the *Customer* and the *Vehicle* forms, you will set properties for the label control and the background of the *Part* form.

1. Set *Font Size* and *Weight* Properties: Repeat the steps performed in the *Customer* form to set the *Font Size* and the *Font Weight* properties for the label control in the header. Also change the name of the label to "Form Title".
2. Expand the Form Background: Repeat the steps performed for the *Customer* form to expand the form background, except there is no need to rename the section. Return to form design view when you are finished.
3. Toggle to Form View: See how the form looks in form view (Figure 26). If you are satisfied with the result, toggle back to design view.

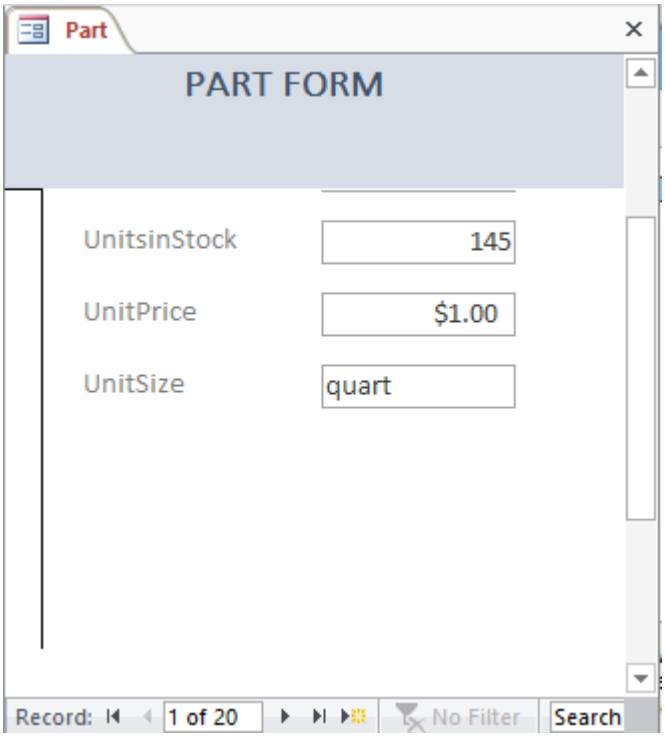


Figure 26: Revised *Part* Form

4. Set Form Properties: Open the Property Sheet for the form with the All tab selected. Use Table 5 as a guide to change selected properties. After changing the properties, close the form and save the changes when prompted.

Table 5: *Part* Form Properties and Settings

Property	**Setting**
Allow Filters	Keep the default, "Yes".
Allow Form View	"Yes"
Allow Datasheet View	"Yes
Allow Edits	Keep the default, "Yes".
Allow Deletions	Set to "Yes".
Allow Additions	Keep the default, "Yes".
Data Entry	Set to "No".

4.4.4 Checking for Form Errors

As you design custom forms, form errors can occur. Access 2013 provides a tool to identify errors in forms and reports. Error checking is controlled through the Error Checking section in the Access Options window (**File → Options** button) as shown in Figure 27. Error checking is limited to individual controls and control properties as shown in Table 6. Access attaches an error indicator (!) next to controls with

errors. Complex errors involving form queries and different parts of a form are not detected by the error checking feature. Appendix A of Chapter 5 explains complex errors that apply to hierarchical forms.

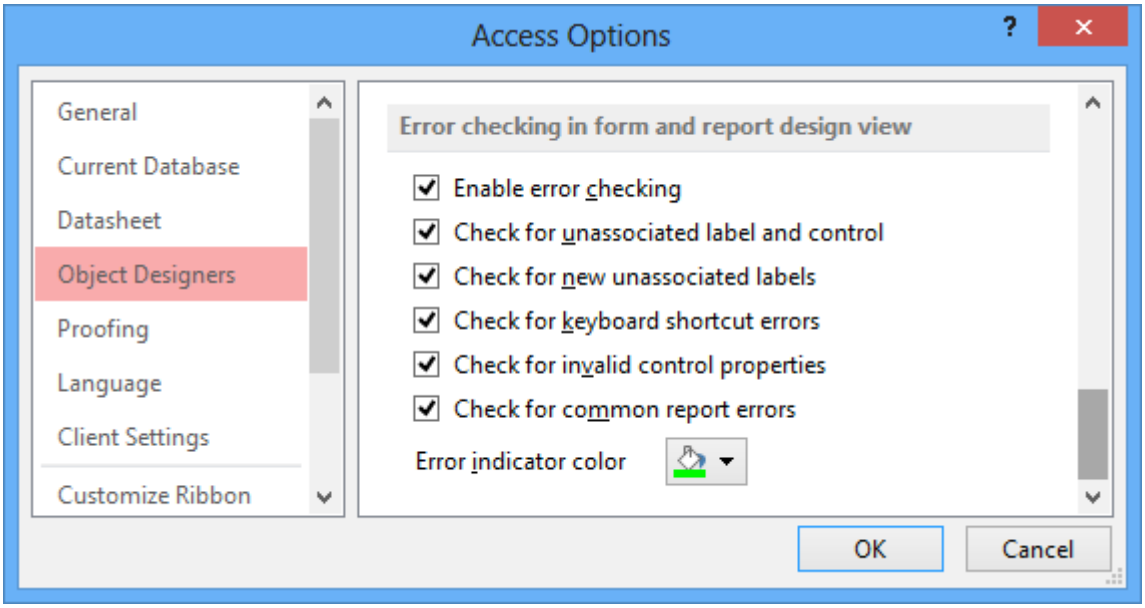


Figure 27: Error Checking Section in the Access Options Window

Table 6: Types of Form Errors Detected by Access

Error Category	**Explanation**
Unassociated label and control	Indicates simultaneous selection of a label and a control, but the label is not associated the control.
New unassociated label	Indicates a new label was added without an associated control.
Keyboard shortcut errors	Indicates a variety of errors involving keyboard shortcuts such as a duplicate shortcut and a shortcut with a space.
Invalid control properties	Indicates a variety of improperly specified properties such as the *Control Source* property not having a valid expression or field name.

4.5 Using Forms for Data Entry, Sorting, and Searching

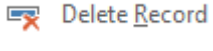
In the earlier sections of this chapter, you have seen how to create single table forms and modify selected controls and properties. As a form developer, you will use these skills often. To gain advanced form development skills, you need to understand how forms are used. This section provides practice with using forms for data entry, sorting, and searching. This practice can provide insight to improve your form development skills as well as to help you educate users about form capabilities.

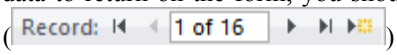
4.5.1 Manipulating Data in a Form

Before manipulating data in a form, you should understand restrictions on form usage. The allow properties presented in Section 4.3 provide one way to restrict form usage. You should be able to associate patterns in the allow properties with restrictions on form usage. To help you recognize patterns, Table 7 lists common patterns and their meanings.

Table 7: Selected Settings of Allow Properties

Form Usage	Allow Property Settings
All activity (no restrictions)	Set all allow properties to "Yes".
Read-only	Set *Allow Additions*, *Allow Deletions*, and *Allow Edits* to "No".
Update and view records	Set *Allow Additions* and *Allow Deletions* to "No". Set other allow properties to "Yes".
Add records only	Set *Data Entry* and *Allow Additions* to "Yes". Set other allow properties to "No".

The instructions assume that the allow properties are set to permit all activity. To begin, open the *Customer* form in form view. The **New**  and Save  buttons become active in the Records group of the **Home** tab in the Ribbon. In addition, you can select **Delete Record**  item in the **Delete** drop down list.

1. Adding Data: When you want to add data, click the **Home → New** button in the Ribbon. The boxes will become blank.
   - Type data in a field and press **Tab** to go to the next field.
   - When you are finished with the entire record, press **Tab** to go to the next record.
   - When you move to another record, Access saves your previous record.
   - When you are finished adding data and want the existing data to return on the form, you should navigate to another record using the record navigation bar () at the bottom left of the view pane.
2. Making Corrections: To edit an existing record, use the **Tab** key or the **Down Arrow** key to move down the form fields. To move up, press **Shift + Tab** or the **Up Arrow** key.
   - Press **Backspace** or the **Delete** key.
   - Use **Esc** to cancel an entry in the current field. To cancel an entire record, press **Esc** again before you move out of the record.
   - When you move to another record, Access saves all changes.
   - Note: Access allows copying (copy and paste) as well as moving (cut and paste) data between fields.
3. Expanding a Textbox: A textbox can be expanded for ease of data entry or editing.
   - Click into the field you want to edit.
   - Press **Shift + F2**.
4. Deleting Data: When you want to delete data, select the record you want to delete by scrolling the record arrows at the bottom of the form.
   - Click the **Delete** button on the ribbon.
   - A dialog box appears asking if you really want to delete the record (Figure 28). Click **Yes** to delete the record.

- To delete data in related tables, make sure referential integrity is enforced and cascade deletions are turned on in the Relationships window.

5. For Practice: Add two records of your choice to the *Customer* form and then delete them.

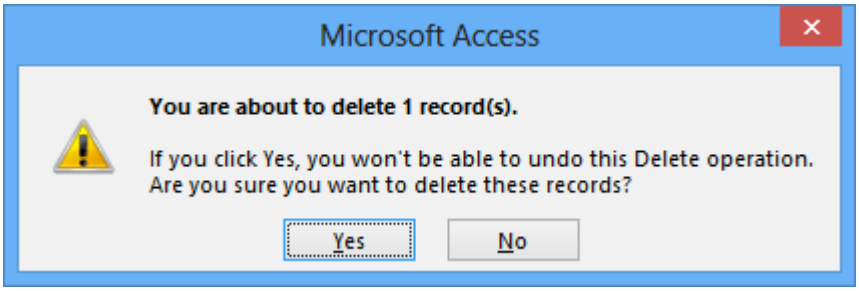


Figure 28: Delete Record Window

4.5.2 Sorting Records

To help see patterns in data and find records, you can sort the data in a form. Access allows sorting in ascending (smallest to largest) and descending (largest to smallest) orders on one or more form fields. In the following instructions, you will use the sort tool for the *Vehicle* form.

1. Open the *Vehicle* Form in Form View: In the Navigation pane, double click the *Vehicle* form to open it in Form view.

2. Select the Sort Fields: To sort the records by year of vehicle, click on the *Year* field.

3. Activate Sort: Click the  (ascending) button in the Home tab of the Sort & Filter group of the Ribbon (Figure 29). Alternatively, you can right-mouse-click on one of the fields or labels and choose **Sort Smallest to Largest**. When you are finished, click the scroll arrows at the bottom of the form to see the ascending order of the records.

4. Reset Records: Reset records back to the beginning of the sort by clicking the  button in the **Home** tab of the Sort & Filter group of the Ribbon (Figure 29).

5. Remove Sort: The sort remains active until you remove it. Thus, if you close the form and open it again, the sort is still active. To remove the sort, click the  (Remove Sort) button in the **Home** tab of the Sort & Filter group of the Ribbon (Figure 29)

6. Return to Sort: After removing the sort, you can return to the same sort by clicking the  button in the Home tab of the Sort & Filter group of the Ribbon. Close the *Vehicle* form when you are finished.

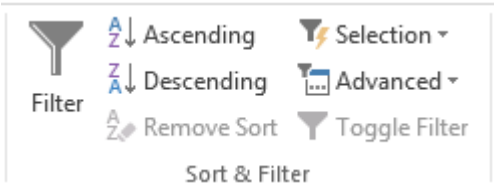


Figure 29: Sort & Filter Group in the Home Tab of the Ribbon

4.5.3 Filtering Records by Form

Filtering records allows only certain records to be displayed based on criteria specified in form fields. You may choose one or more filter criteria from different fields or from within the same field. You can filter by

form, by selection, or by excluding certain selections. In this section you will apply the Filter by Form tool to the *Vehicle* form.

1. Open the *Vehicle* Form in Form View: In the Navigation pane, double-click the *Vehicle* form to open it in the form view.

2. Activate the Filter: To activate the filter tool, select **Home → Advanced → Filter By Form** (Figure 30) in the Sort & Filter group of the Ribbon. The fields become blank except for the *Make* and the *Model* fields, as depicted in Figure 31. The bottom of the window contains tabs titled Look for and Or.

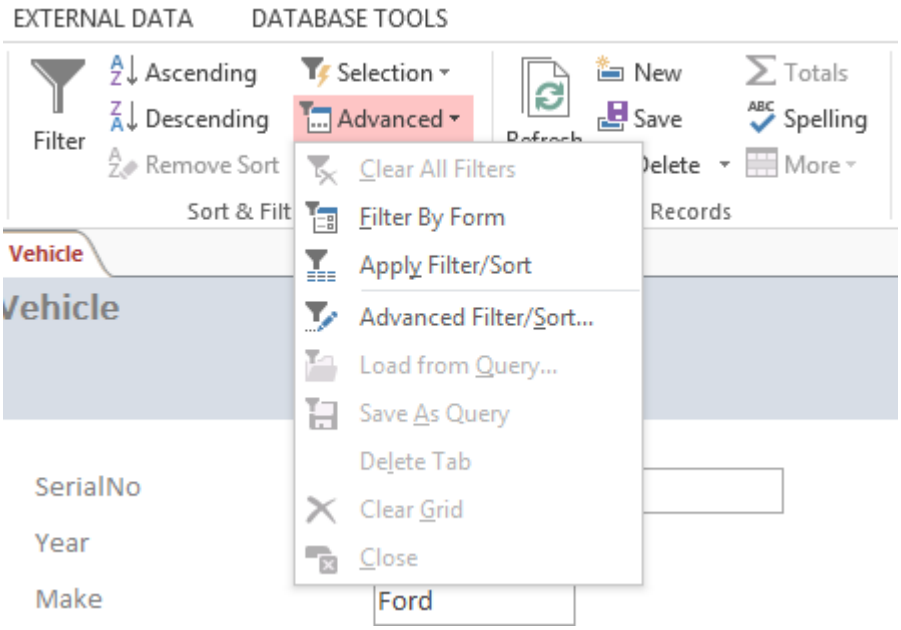


Figure 30: Advanced Menu in the Sort & Filter Group of the Ribbon

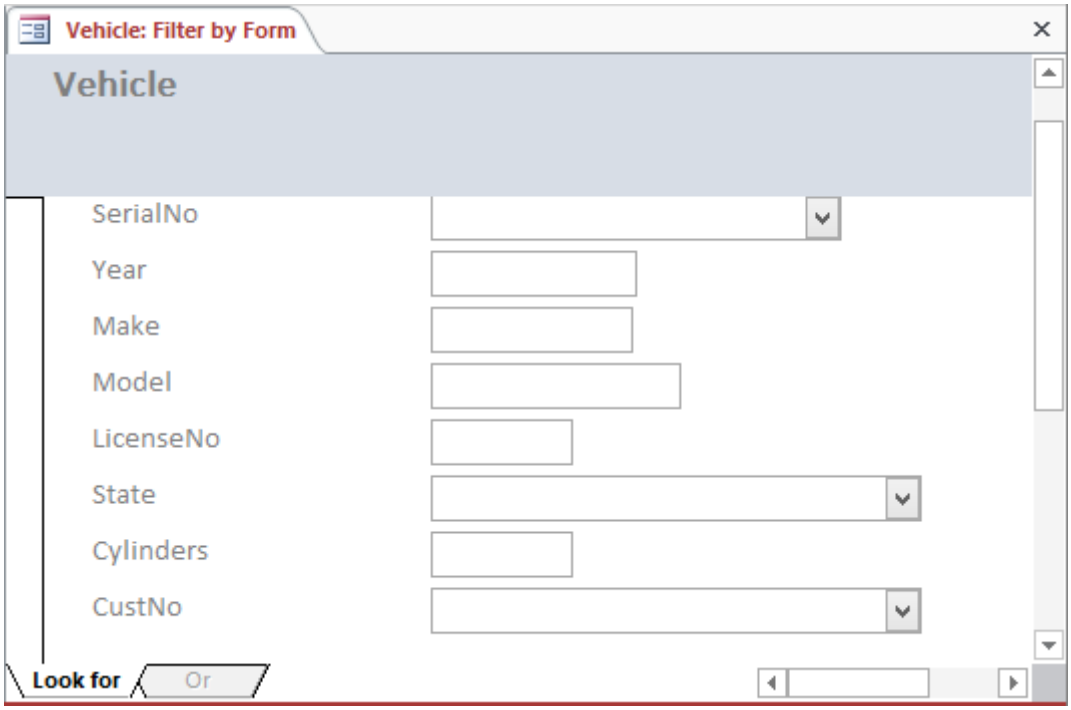


Figure 31: *Vehicle* Form with Filter by Form Activated

3. Define AND Filter Criteria: Click on each field and a drop down arrow appears (Figure 31). For this example, click the *Make* field and click the arrow to reveal a list of vehicle makes. Choose "Toyota" for the first filter criterion (you also can just type in the desired criterion). For the next filter criterion, go to the *Model* field and select or type "Celica". You have told Access to look for vehicles with the *Make* "Toyota" and the *Model* "Celica".

4. Define OR Filter Criteria: Click the **Or** tab at the bottom of the window to choose additional criteria from the same field. After selecting the **Or** tab, the fields become blank again. Return to the *Make* field and choose or type "Honda". Go to the *Model* field and select or type "Accord". You have told Access to look for vehicles that are either Toyota Celicas or Honda Accords.

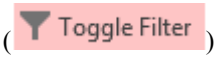
5. Broaden the Search: Each time you click the **Or** tab, another **Or** tab appears so you can keep choosing criteria. The more criteria selected, the broader the search. For example, if you click the **Or** tab again and select "2003" for the *Year* field, the criteria match Toyota Celicas, Buick Regals, or vehicles made in 2003

> Note: Each set of **Or** criteria you select is stored. Click any **Or** tab to see that tab's criterion displayed in the field. If you need to delete a criteria, you must delete that tab by **Home → Advanced → Delete Tab** (Figure 30) in the Sort & Filter group of the Ribbon. Alternatively, you click the right mouse button outside of the form and select the **Delete Tab** command.

6. Correct Mistakes: If you make a mistake choosing criteria for either mode of filter (AND/OR), click in the field and backspace over the data or use the **Cut** button. Otherwise, click the **Delete** button  in the Records group of the **Home** tab of the Ribbon to return the fields to blank. You also can click the right mouse button and select the **Cut** button. If you want to leave the filter mode altogether, click the **Close** button on the window to return the records back to normal.

7. Apply Filter: After your criteria are chosen, it is time to apply the filter. Click **Home → Advanced → Apply Filter/Sort** in the Sort & Filter group. You also can click the right mouse button outside of the form and select the **Apply Filter/Sort** button.

8. View Filtered Records: Click the scroll arrows at the bottom of the form to view the filtered records. Wherever you are in the list of records, you can reset the records back to the beginning by selecting **Apply Filter/Sort** again.

9. Toggle Filter: To view all records, toggle using the **Toggle Filter** button (

) in the Sort & Filter group of the Ribbon. When you are finished viewing the records, toggle back to form view.

10. Remove Filter: The filter remains active until you remove it. To remove the filter, select **Home → Advanced → Clear All Filters**. Unlike the sort tool, if you close the form and open it again, the filter is removed.

4.5.4 Filtering Records by Selection

One of the main differences between filtering records by form and filtering records by selection is that with the latter, Access can only filter by one criterion at a time. However, filtering by selection gives you the option of filtering by excluding a selection. In this section, you will practice filtering by selection and excluding a selection.

1. Open the *Vehicle* Form: In the Database window, select the *Vehicle* form and click the **Open** button.

2. Choose a Field to Filter: Scroll through the records until a record appears with a field by whose criterion you desire to filter. For this example, filter the records by *Year* "2002". Scroll until you come to a record with the *Year* "2002" and click in that field.

3. Filter: To activate the filter tool, choose the **Home → Filter** button in the Sort & Filter group. The Filter menu appears as shown in Figure 32. You also can select a specific value to filter or use a number filter providing a comparison on the selected value. The number filter is also available in the **Home → Selection** list (Selection) in the Sort & Filter group

4. Filter Excluding Selection: Use this mode to find records with the exception of the selected filter criterion. For example, select *Make* and scroll to a record with "Toyota". Choose the exclusion comparison ("Does not equal Toyota") from the **Home → Selection** list. Alternatively, you can see the same filter choices by right clicking in the form field.

5. View Filtered Records: Click the scroll arrows at the bottom of the form to view the filtered records. You can reset the records back to the beginning by selecting **Home → Advanced → Apply Filter/Sort**.

6. Toggle Filter: To view all records, toggle using the **Toggle Filter** button (Toggle Filter) in the Sort & Filter group of the Ribbon.When you are finished viewing the records, toggle back to form view.

7. Remove Filter: The filter remains active until you remove it. To remove the filter, select **Home → Advanced → Clear All Filters**. Unlike the sort tool, if you close the form and open it again, the filter is removed.

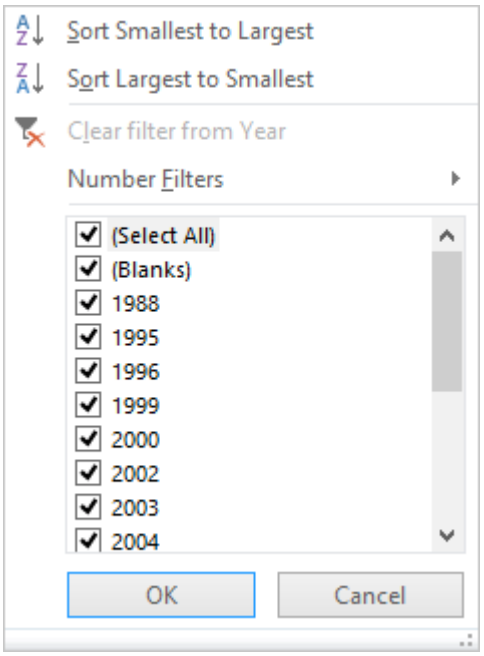


Figure 32: Filter Menu

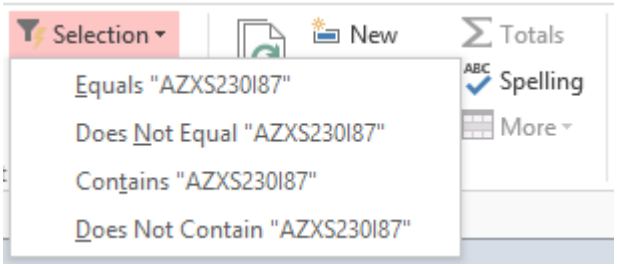


Figure 33: Selection Menu for Toyota Value

4.5.5 Using the Find Tool

The Find tool is a fast and convenient way to locate a specific record. It can be used regardless of the record displayed on the form. You will practice the find tool using the *Customer* form.

1. Open the *Customer* Form: In the Navigation pane, double-click the *Customer* form to open it in the form view.
2. Choose Field to Search: For this example you will search for a specific customer by the last name. Click inside the *LastName* field.
3. Activate the Find Box: Click the **Binocular** button  in the Find group on the ribbon to open the Find window. The *LastName* field appears as depicted in Figure 34.
4. Choose Value to Find: In the Find What area, type a last name such as "Wyatt".
5. Where to Search: In the Search area you can keep the Search default choice "All".
6. What to Match: In the Match area you also can click a drop-down menu to reveal other choices, but you will keep the default choice "Whole Field".
7. Check Box Selections: The check boxes provide further choices for searching. For this example, do not check any boxes.
8. Replace: If you want to replace the record when you find it, click the **Replace** tab at the top of the window. Type into the Replace With area what you want to replace the record with (Figure 35).
9. Click the Find Buttons: The first button to click is the **Find Next** button. This should give you the record you require unless there is more than one, such as more than one customer with the same last name. If more than one record is a possibility, click the **Find Next** button again.
10. View the Record: When the record for which you are searching appears on the form, close the Find window.

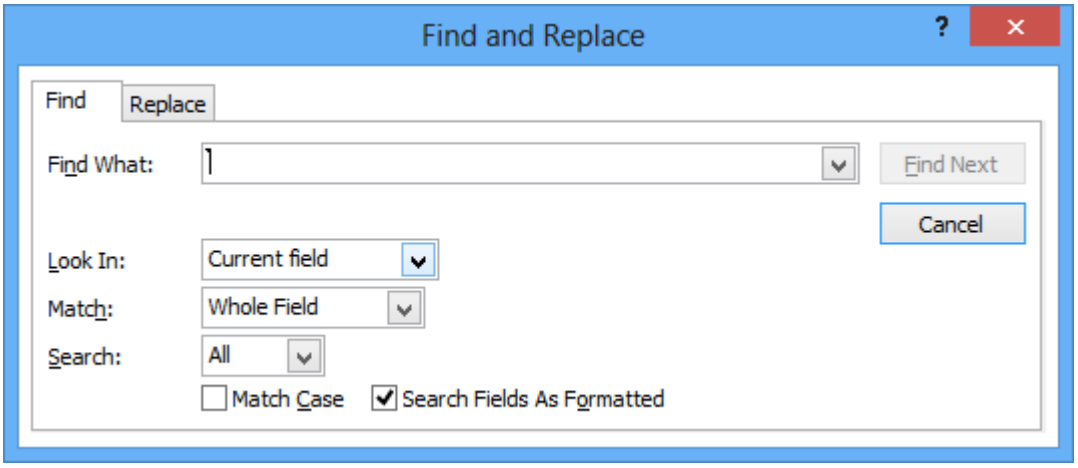


Figure 34: Find Tab of the Find and Replace Window

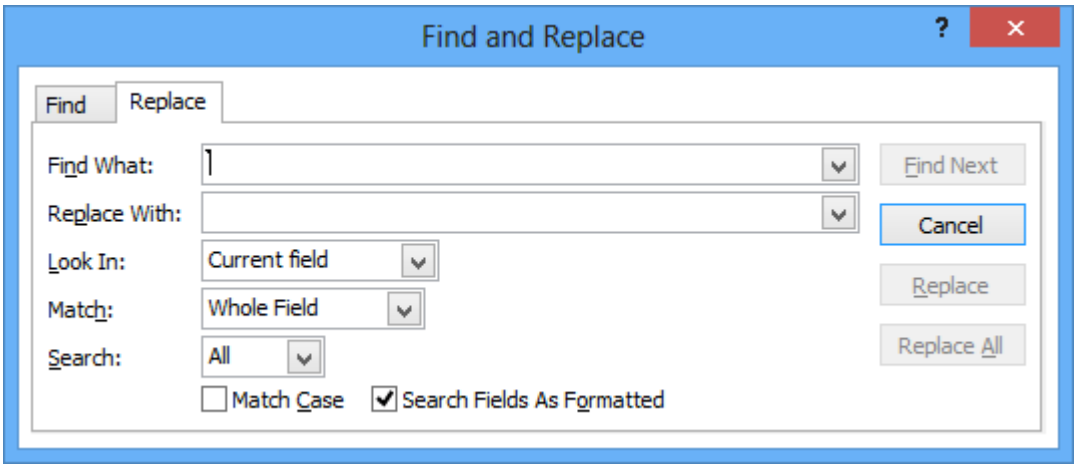


Figure 35: Replace Tab of the Find and Replace Window

Closing Thoughts

Chapter 4 has provided guided instruction about developing and using single table forms in Microsoft Access 2013 Access provides a number of tools to create and refine single table forms. You learned about the Form Wizard, the Form command, and the Form Design view to create single table forms. Forms have several levels of properties to manipulate. You learned about some important form level properties and the relationship of form field properties to related table field properties. Forms can be used to enter new data, change existing data, delete data, and view existing data. To become aware of how users interact with forms, you practiced manipulating data and searching for data. Access provides several convenient tools to search for data without writing a query.

After completing this chapter, you should be ready to tackle the more difficult hierarchical forms in Chapter 5. Because hierarchical forms manipulate more complex data, they can be more difficult to develop. To become proficient in developing hierarchical forms, you need to understand the conceptual material in textbook Chapter 10 as well as the practical material in lab Chapter 5. The background knowledge and skills of this chapter should prepare you for the practical issues covered in Chapter 5.

Chapter Reference

The chapter reference summarizes procedures that you practiced. For wizards discussed in the chapter, the procedures highlight important parts of the wizards but do not list all of the steps.

Procedure 1: Creating a New Form (Section 4.1)

88. You can create a new form using the Form command, the Form Wizard, or the Form Design window.
89. The Form command allows limited choices for the layout of a form. All fields of a data source (table or query) are placed on the form.
90. The Form Wizard allows more choices about the fields on a form and the form's layout.
91. The Form Design view allows complete flexibility about the fields on a form and the form layout. After using the Form command or the Form Wizard, you can customize the appearance of a form in Form Design view.

Procedure 2: Moving Controls (Section 4.2.1)

92. You can use spacing commands in the **Arrange→Size/Spacing** list in the Size & Ordering group to change the space between controls. Before using these commands, select controls using the mouse and the **Shift** key. With all fields selected, you can increase, decrease, or equal spacing options for vertical or horizontal spacing.
93. You can use items in the **Align** list in the Size & Ordering group to align controls. Select the controls to align (use the **Shift** key) and choose the kind of alignment from the **Align** list. You can align fields to the left, right, top, bottom, or grid.
94. You can create a tabular or stacked layout for a group of controls using the **Tabular** or **Stacked** buttons in the Table group. If you want to remove a tabular or stacked layout, you should use the **Remove Layout** button in the Table group. The form creation tools (**Create→Forms**) will create form fields using stacked or tabular layouts so removing the layouts can be useful to override these tools.
95. If these tools do not suffice, you can move controls individually. To move a control with the mouse, highlight the field by clicking on it and then drag the control to move it. Notice that a bound control and its label move together as long as you drag it at any point except the upper left corner.
96. You also can move a label and a bound control separately. To move a label separately, select the control and then drag its anchor (filled rectangle in upper left corner) of the control. You can drag the label or bound control separately using the appropriate anchor.

Procedure 3: Setting the Background Color (Section 4.2.1)

97. Select the detail (backgound) section and choose a color in the *Back Color* property.
98. Double-click on the grey, grid background to open a Property Sheet titled "Section: Background".
99. Go down to the *Back Color* property and click in that area to reveal the ellipsis (**…**) button. Click the ellipsis button to open the color palette, select a color, and click **OK**.
100. Close the Property Sheet. The background color will be applied.

Procedure 4: Setting a Background Image (Section 4.2.1)

101. As an alternative to a background color, you can use an image.
102. Double-click anywhere on the dark gray, nongrid area to open the Form Property Sheet.
103. Click on the **All** tab and click in the *Picture* property to reveal the ellipsis (**…**) button. Click the ellipsis button to open the the Insert Picture window (opens to the default "My Documents").
104. Browse to find the location of the desired picture file and click **OK**.
105. Go to the *Picture Alignment* property and select the area where the picture is best displayed as the background.

Procedure 5: Adjusting the Size of the Detail Section (Section 4.4.1)

1. You can extend the size of the detail section to allow for additional controls or to provide more comfortable placement of existing controls.
2. To extend the height of the detail section, move the cursor to the bottom of the form rectangle until the cursor changes to a double arrow with a thick line. You then can drag the mouse down to extend the height.
3. Use the same method to increase the width of the detail section.

Procedure 6: Using a Form to Add Records (Section 4.5.1)

Although data entry can be performed in form view or datasheet view, the instructions here use form view. The instructions assume that the allow properties are set to permit all activity. To begin, open a form in form view. The **New Record** and **Delete Record** buttons become active on the ribbon.

1. To begin adding data in a form, click the **New** button in the **Records** group on the ribbon. The boxes in the form will become blank.
2. Type data in a field and press **Tab** to go to the next field.
3. When you are finished with the entire record, press **Tab** to go to the next record.
4. When you move to another record, Access saves your previous record.
5. When you are finished adding data and want the existing data to return on the form, you should navigate to another record using the record navigation bar.

Procedure 7: Searching for Records (Sections 4.5.3, 4.5.4, and 4.5.5)

Access provides a number of tools to search for records while viewing form data. You can use the Filter by Form tool, the Filter by Selection tool, and the Filter Excluding Selection tool. These tools are available in form view from the **Home → Sort & Filter** group. In addition, you can use the Find tool to search for specific records.

1. The Filter by Form tool supports multiple criteria connected by the OR operator. Each criterion consists of one or more search conditions connected by the AND operator. After defining search criteria, use **Sort & Filter → Advanced → Apply Filter/Sort** to retrieve records matching the search criteria. To remove the filter and view all records again, use **Sort & Filter → Advanced → Clear All Filters**.
2. The Filter by Selection tool is a simpler but easier-to-use tool than the Filter by Form tool. The Filter by Selection tool supports only a single search condition. You can define a search condition by typing a value in the field to search. To perform the search, click **Sort & Filter → Selection** or the right mouse shortcut menu.
3. The Filter Selection tool allows a variety of filters including exclusion and containment. An exclusion filter finds records that do not satisfy the search condition.
4. The Find tool supports searching for records that match a value in a specific form field or any form field. You can match records that exactly match the search value, match the search value at the start of a field, or contain the search value in any part of a field. You invoke the Find tool using the **Binocular** icon on the ribbon.

Additional Practice

The following problems provide additional practice with the extended auto repair database as well as the textbook databases.

Part 1: *Labor* Table

1. Make a form for the *Labor* table using the Form command.
2. Make a form for the *Labor* table using the Form Wizard.
3. Make a form for the *Labor* table using form design without using the Form command or the Form Wizard. Add a form heading in addition to the form fields.
4. Add two records using one of the forms. Update values in both of the new records. Delete the new records after adding and updating.
5. Use the Filter by Form method to find labor records in which the hourly rate is greater than $20 and the standard time is less than 0.3 hour or the hourly rate is less than $25 and the standard time is greater than 0.4 hour.
6. Use the Filter by Selection tool to find labor records with an hourly rate equal to $20.
7. Use an exclusion filter to find labor records with an hourly rate not equal to $20.

Part 2: University Database

Make forms for the *Student*, the *Faculty*, and the *Course* tables in the university database of textbook Chapter 10. Use any of the tools described in this chapter.

Part 3: Order Entry Database

Make forms for the *Customer*, the *Employee*, the *Supplier*, and the *Product* tables in the extended order entry database of textbook Chapter 10. The form for the *Product* table should not allow insertion of new records. Use any of the tools described in this chapter.

Chapter 5: Hierarchical Form Lab

Learning Objectives

This chapter demonstrates Access 2013 features for hierarchical forms that are more complex than the single table forms you developed in Chapter 4. At the completion of this chapter, you should have acquired the knowledge and skills to

- Create hierarchical forms using the Form Wizard.
- Reuse existing subforms in new hierarchical forms.
- Understand how AutoLookup queries display data on a form.
- Customize combo boxes with additional fields and row restrictions.
- Gain additional practice with aligning and moving controls on a form.
- Use form fields to perform record and aggregate computations.
- Understand how to reference fields in a main form, a subform, and a query.

Overview

This chapter takes you beyond the simple forms of Chapter 4. To develop hierarchical forms[10], you need the practical knowledge and skills of Chapter 4, the concepts of textbook Chapter 10, and the practical skills about hierarchical forms described in this chapter. If you have forgotten the concepts of Sections 10.3 and 10.4 of textbook Chapter 10, you should review them now. In particular, you need to know the rules for 1-M updatable queries, components of a hierarchical form, the relationship between hierarchical forms and tables, and query formulation skills for hierarchical forms. Without understanding these concepts, you should still be able to complete this lab, but you may have difficulty creating hierarchical forms on your own.

This chapter guides you through the steps to create a complex hierarchical form. To make the instructions clear, you will develop the final form through four forms of increasing complexity. In developing these four hierarchical forms, you will learn how to define a form based on a query, link a subform and a main form, understand AutoLookup queries, extend combo boxes with additional fields and row restrictions, and share computations between a main and a subform. You will find that no matter the level of details in the instructions, it often takes a trial-and-error approach to obtain the desired formatting for a form. To cement your understanding of the concepts, you may want to repeat some parts of the lab exercises. In the last part of this chapter, you will learn about four specialized but important features, the Subform Wizard, the tab control, conditional formatting, and object dependencies.

5.1 Creating a Simple Hierarchical Form

The first version of the repair order form is rather simple. It manipulates the *RepairOrder* and *PartsUsed* tables since they have a 1-M relationship (refer to the Relationships window in Figure 1). Because the repair order form only manipulates one table in both the main form and the subform, you do not need to

[10] Beginning in Access 2007, hierarchical forms were called "one-to- many forms." This chapter uses the term "hierarchical form" to be consistent with previous Access versions and the textbook.

write updatable queries as described in textbook Chapter 10. In later sections you will write queries. This section provides guided instruction to create the first repair order form and make some simple modifications to it.

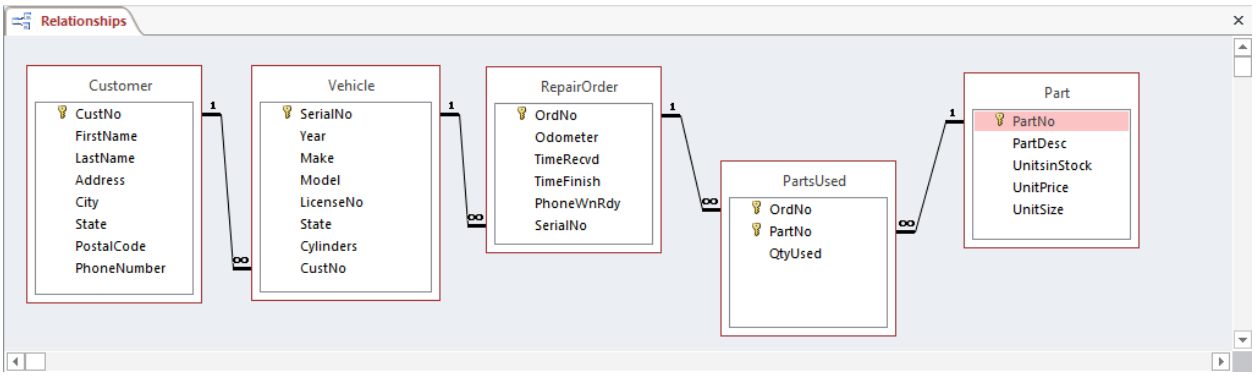


Figure 1: Relationships in the Auto Repair Database

5.1.1 Using the Form Wizard to Create a Hierarchical Form

In Chapter 4 you used the Form Wizard to create single table forms. The Form Wizard also can be used to create hierarchical forms consisting of a main form and an embedded subform. The main form contains data from the parent table and the subform contains data from the child table. You will use the Form Wizard to create the initial version of the repair order form using the following instructions.

1. Open the New Form Window: Choose **Create → Form Wizard** in the Forms group of the Ribbon. Select the *RepairOrder* table as shown in Figure 2.
2. Select Fields to Include: Click the >> button to move all of the fields shown in the left side (Available Fields) to the right side (Selected Fields).
   - In the same wizard window, go up to the Tables/Queries selection box where the *RepairOrder* table name appears and select a different table, the *PartsUsed* table. Now, the fields of the *PartsUsed* table appear under Available Fields. Click the >> button to move all of the fields to the right side (Selected Fields). Note, the Selected Fields window (Figure 3) contains fields from both tables. Click the **Next >** button when finished.

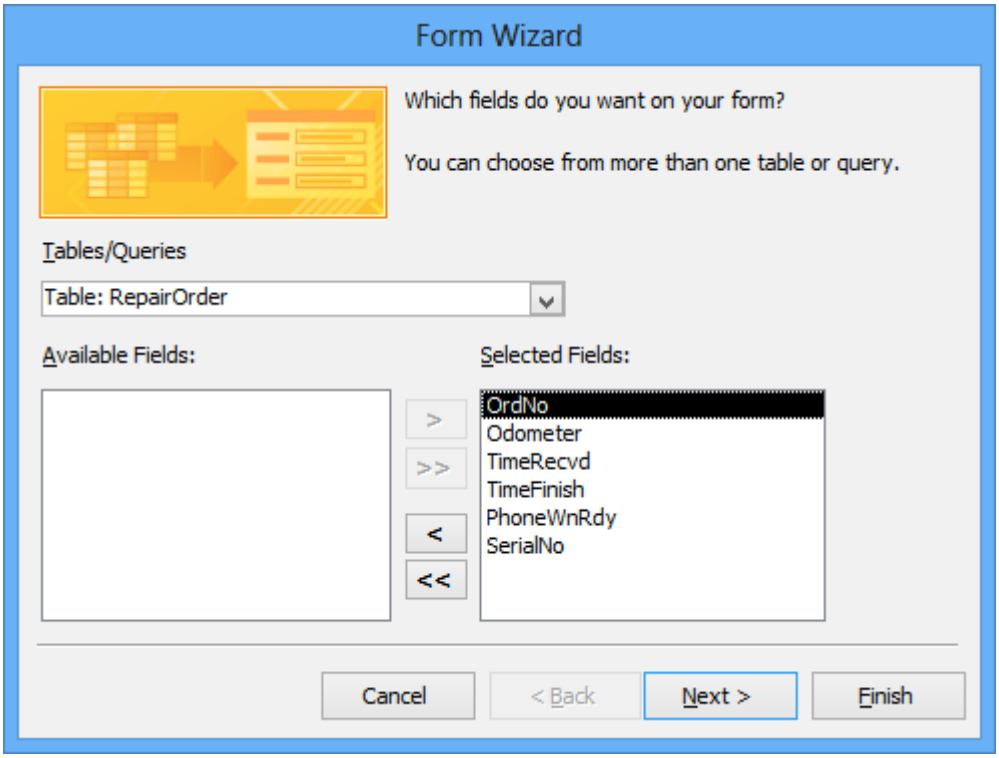


Figure 2: Initial Form Wizard Window with *RepairOrder* Table Selected

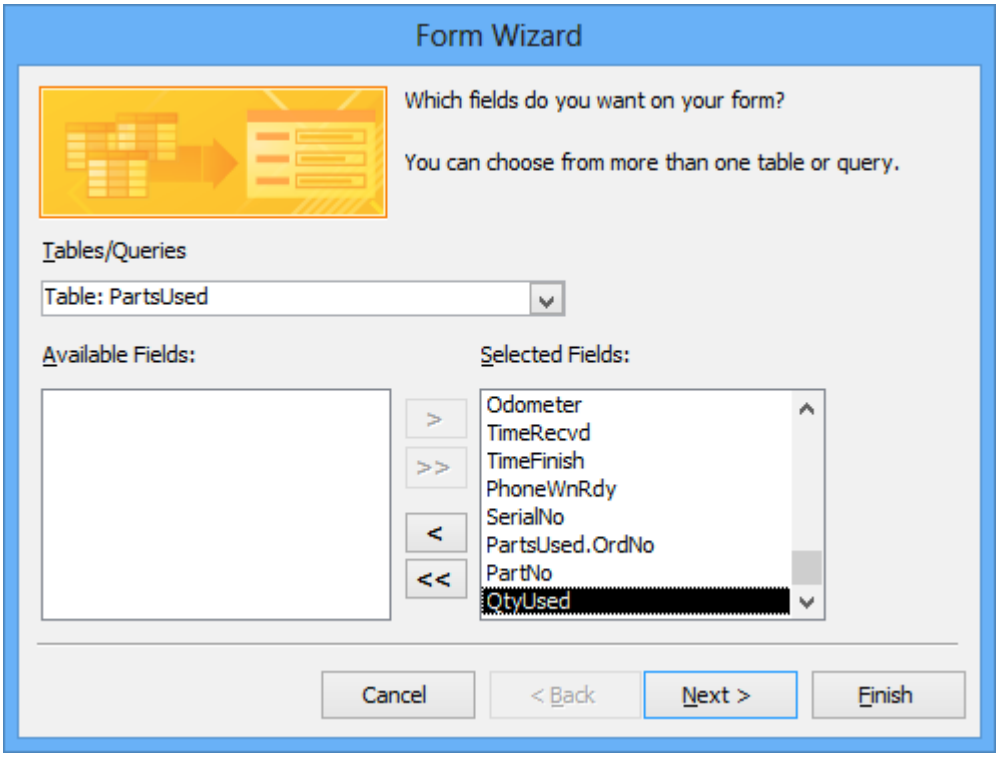


Figure 3: Field Selection Window of the Form Wizard with PartsUsed Fields Selected

- Select Viewing Option: The next window asks how you want to view the data. Select "by RepairOrder" because you want *RepairOrder* data to appear in the main form. The option button "Form with subform(s)" should be selected as shown in Figure 4. Select it if it is not selected and click **Next >**.

Technical Note: If your database does not have a 1-M relationship from the *RepairOrder* table to the *PartsUsed* table, the viewing options window will not appear. Instead, Access will assume that you want a main form only and display the Layout window. If you have completed Chapter 2, your database contains all the 1-M relationships.

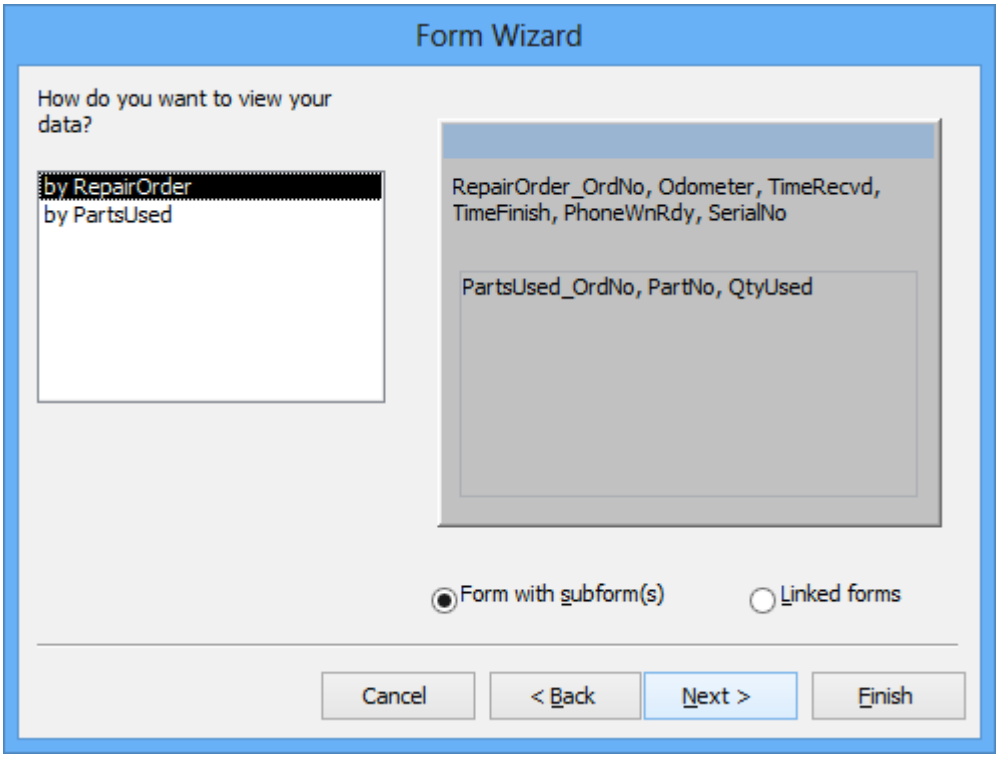


Figure 4: Viewing Options Window of the Form Wizard

- Select the Layout of the Subform: In the next wizard window (Figure 5), select "Datasheet" (it should be the default) and click **Next** >.
- Name the Forms: You must provide names for the main form and the subform. If not entered already, type "RepairOrder" for the main form and "Parts: Subform1" for the subform (Figure 6). In the same window, select the second option, "Modify the form's design", and click **Finish**. The *RepairOrder* form appears in design view as shown in Figure 7.

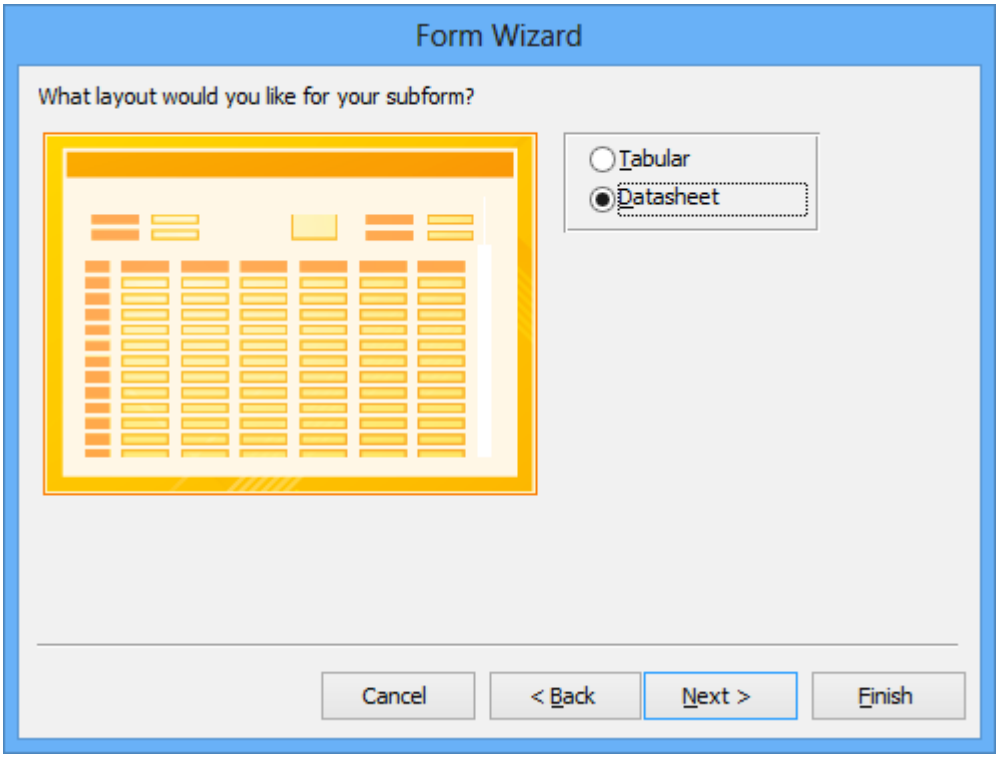


Figure 5: Subform Layout Window of the Form Wizard

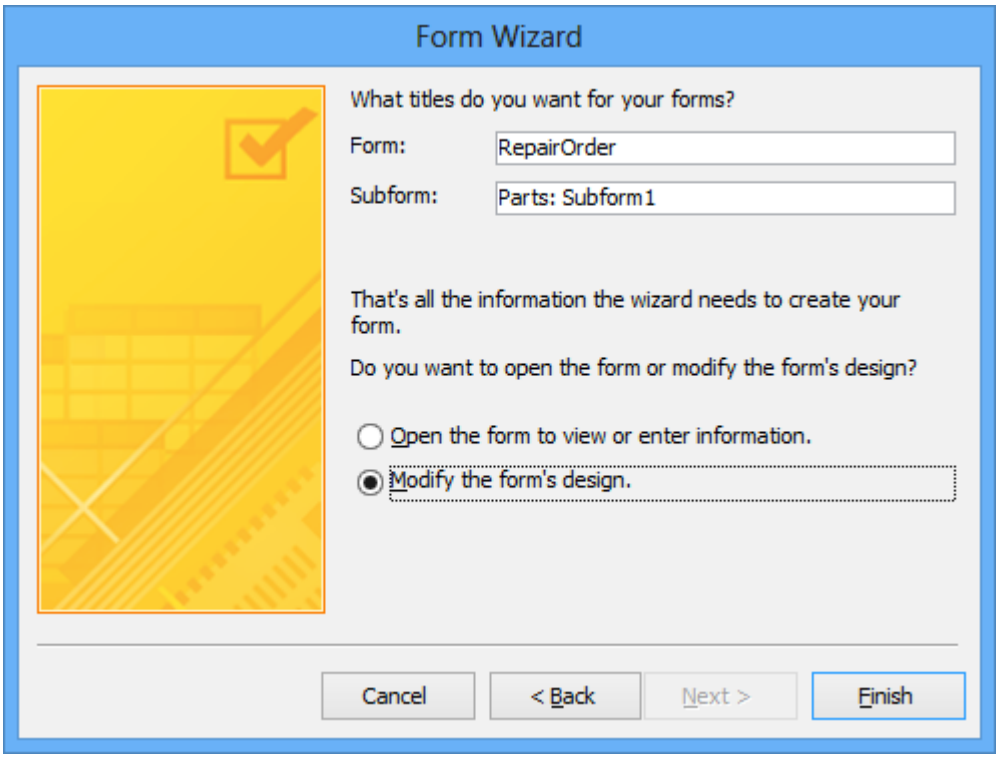


Figure 6: Last Window of the Form Wizard

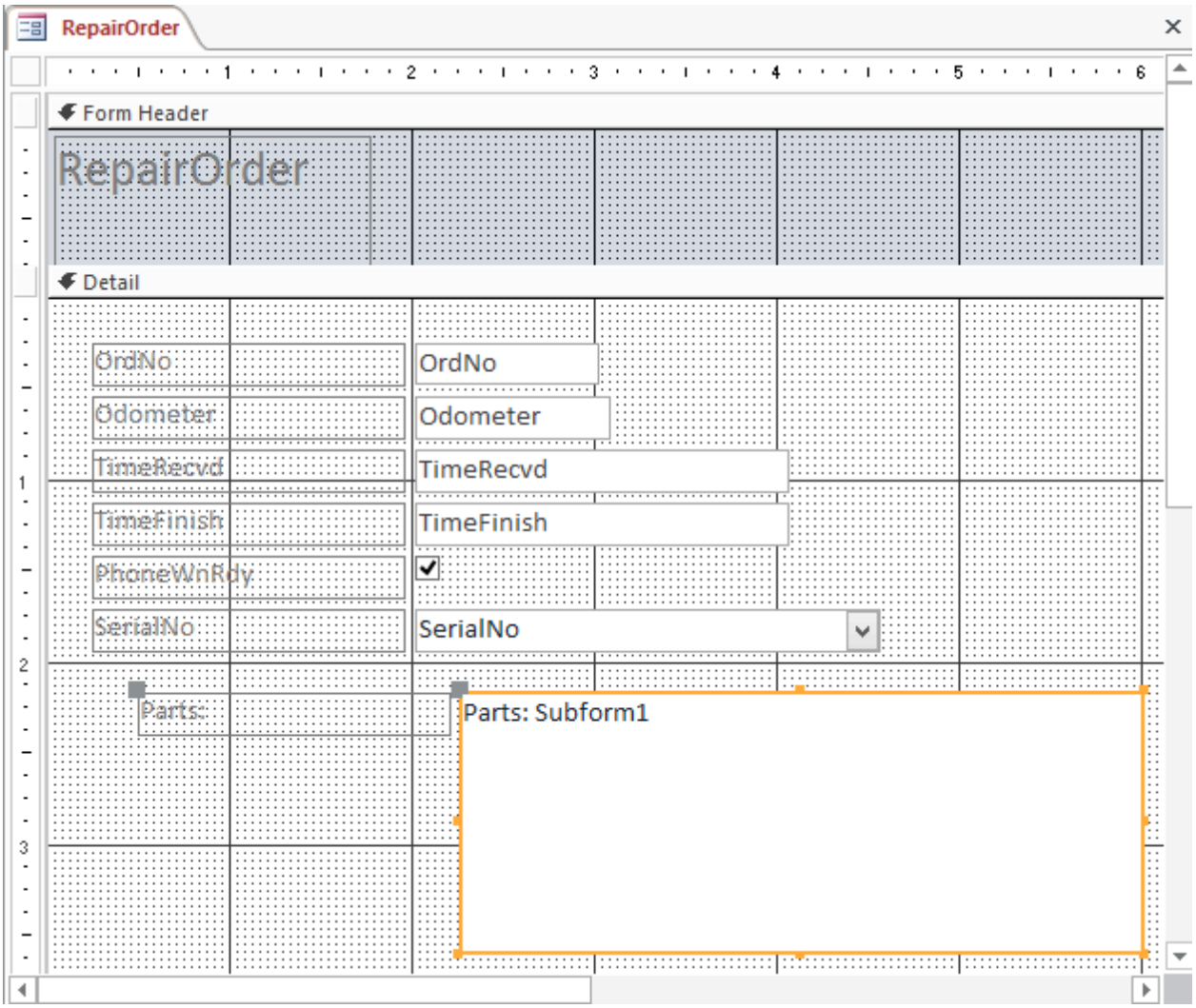


Figure 7: Design View for the RepairOrder Form

5.1.2 Customizing a Form

You will customize this form similar to what you did in Chapter 4. You will put a form title in the form header and expand the form background. You should still have the Design View window of the *RepairOrder* form open.

1. Title the Form: In the form header, modify the form's title. Inside the label, type the name "Repair Order Form". In the Properties window for the label, set the *Name* property to "Form Title", the *Font Size* to 14, and the *Font Weight* to "Semi-bold".

2. Expand the Form Background: Open the Properties window for the detail section by double-clicking on the "Blends" background.

    1. Expand Height in Detail Section: In the same Properties window, change the *Height* property to "5.0". Close the Properties window.

    - Expand Width in Form Background: Double-click on the dark gray area (outside the grid area) to open the form's Properties window. Change the *Width* property to "7.5". Close the Properties window.

3. Toggle to Form View: Scrolling through the hierarchical form (Figure 8) shows a *RepairOrder* record along with its related *PartsUsed* records. Close the form window when you are finished.

- In the Database window you will find a form named *RepairOrder* that displays a hierarchical form when opened. You also will find a form titled *Parts: Subform1* that displays a datasheet when opened. However, the subform is normally opened as part of the hierarchical form, not by itself.

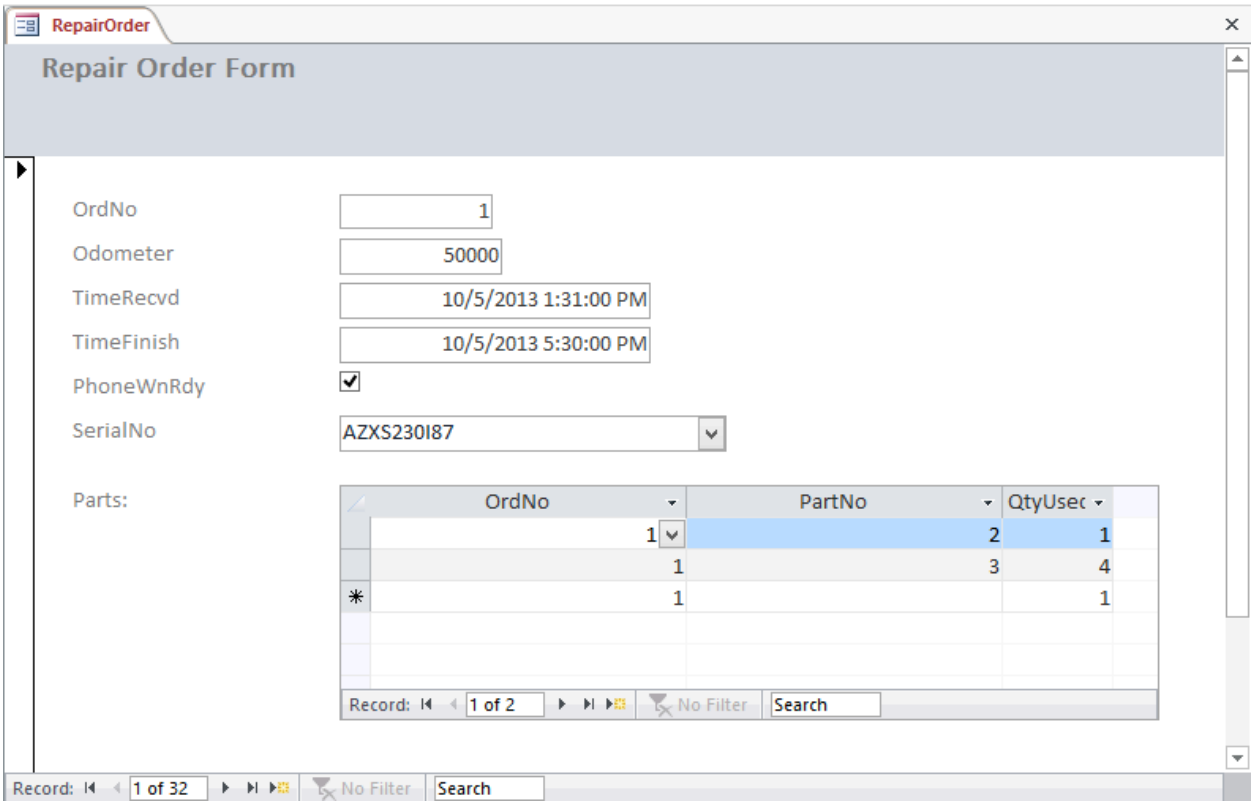


Figure 8: *RepairOrder* Hierarchical Form

5.2 Extending the Subform

The initial version of the *RepairOrder* hierarchical form is obviously inadequate. One improvement is to display additional data in the subform while eliminating the *OrdNo* column (since it is already displayed in the main form). When a user types an order number on the main form, the subform should display part data such as the description and the quantity in stock. To accomplish this, you will place additional fields in the subform by associating the subform with a query.

5.2.1 Creating a Query for the Subform

It is usually more convenient to use Query Design rather than SQL when writing a query for a form. Query Design allows you to focus on the tables and the fields in the query result. (Remember that you still can switch between Query Design and SQL.) Another important advantage to using Query Design for form queries concerns the join style. To make a multiple-table query updatable in Access, the join operator style[11] must be used. Since Query Design always uses the join operator style, you do not have to remember to use the join operator style. If you use SQL directly, make sure to use the join operator style.

[11] See textbook Chapter 4 for a discussion of the join operator style.

To enable the subform to display additional information, you need to associate the subform with a query, not the *PartsUsed* table. Before revising the subform, you should first formulate the underlying query:

5. Open a New Query: Move to the **Queries** section in the Database window. Choose **Create → Query Design** in the Queries group to open the New Query window. In the Show Table window, add the *PartsUsed* and the *Part* tables.

5. Formulate the Query: In the Query Design window, formulate a query as shown in Figure 9. Follow these tips to formulate the query.

   - The query should contain all fields from the *PartsUsed* table: *OrdNo, PartNo,* and *QtyUsed.* In your query, it is important that *PartNo* comes from the *PartsUsed* table, not from the *Part* table.
   - The query should contain all fields from the *Part* table except the *PartNo* field: *PartDesc*, *UnitsInStock*, *UnitPrice*, and *UnitSize.*
   - Note that the query follows the rules for 1-M updatable queries described in textbook Chapter 10. In particular, note that the child table (*PartsUsed*) contains both the foreign keys and its primary key. If the *Part.PartNo* field were used instead of *PartsUsed.PartNo*, the query would not support modifications to the *PartsUsed* table. The subform will not work correctly unless its underlying query supports updates to this table. To verify that the query is updatable, follow the steps described in Appendix C.
   - Save the query as "SubQuery1".

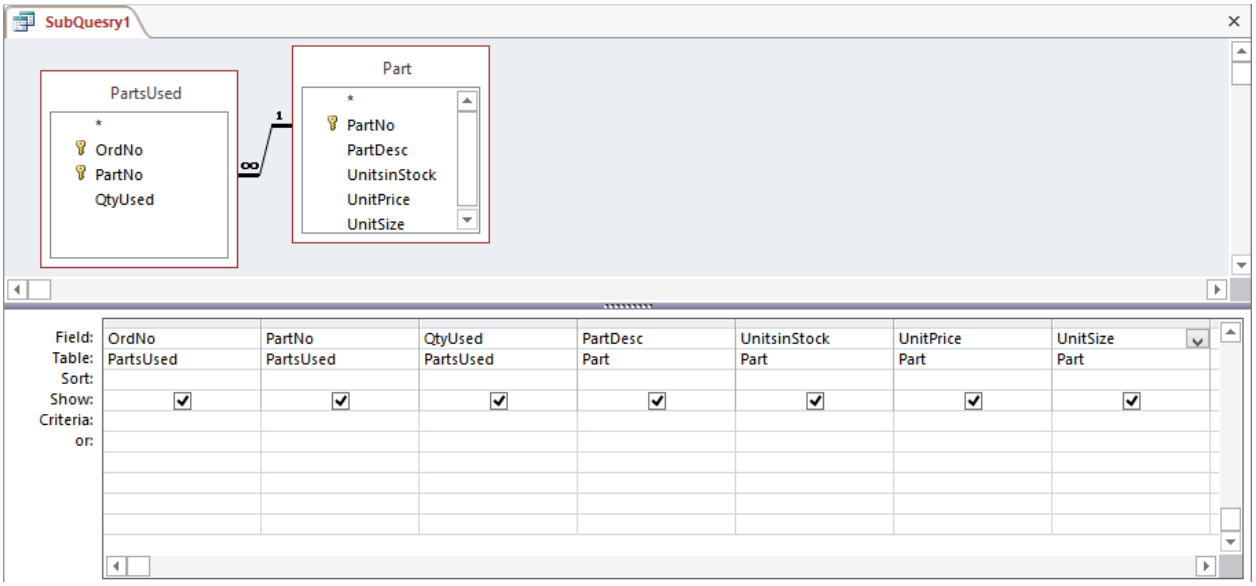


Figure 9: Query Design Window for *SubQuery1*

5.2.2 Creating the Repair Order Form with a New Subform

Create a hierarchical form using the Form Wizard again. Most questions can be answered identically as before. All questions are repeated for your reference.

1. Open the New Form Window: Choose **Create → Form Wizard** in the Forms group to open the initial window of the Form Wizard. Select the *RepairOrder* table where indicated.
2. Select Fields to Include: Click the >> button to move all of the fields shown in the left side (Available Fields) to the right side (Selected Fields).

- In the same wizard window, go up to the Tables/Queries list box where the *RepairOrder* table name appears and select *SubQuery1*.
- Now, the fields of *SubQuery1* appear under Available Fields. Click the single **>** button to move the following fields to the right side (Selected Fields): *PartNo*, *QtyUsed*, *PartDesc*, *UnitsInStock*, *UnitPrice*, and *UnitSize*.
- Note, the Selected Fields window contains fields from both the *RepairOrder* table and *SubQuery1*. Click the **Next >** button when finished.

- Select the Viewing Option**:** The next window asks how you want to view the data. Select "by RepairOrder" because it appears in the main form. The option "Form with subform(s)" also should be selected. Click **Next**.
- Select the Layout for the Subform: In the next window, select "Datasheet" (it should be the default) and click the **Next >** button to advance.
- Name the Form: Type "RepairOrder1" for the main form and "Parts: Subform2" for the subform (there is a space between Parts: and Subform2). In the same window, select the second option, "Modify the form's design", and click **Finish**. The *RepairOrder1* form appears in design view as in Figure 10.

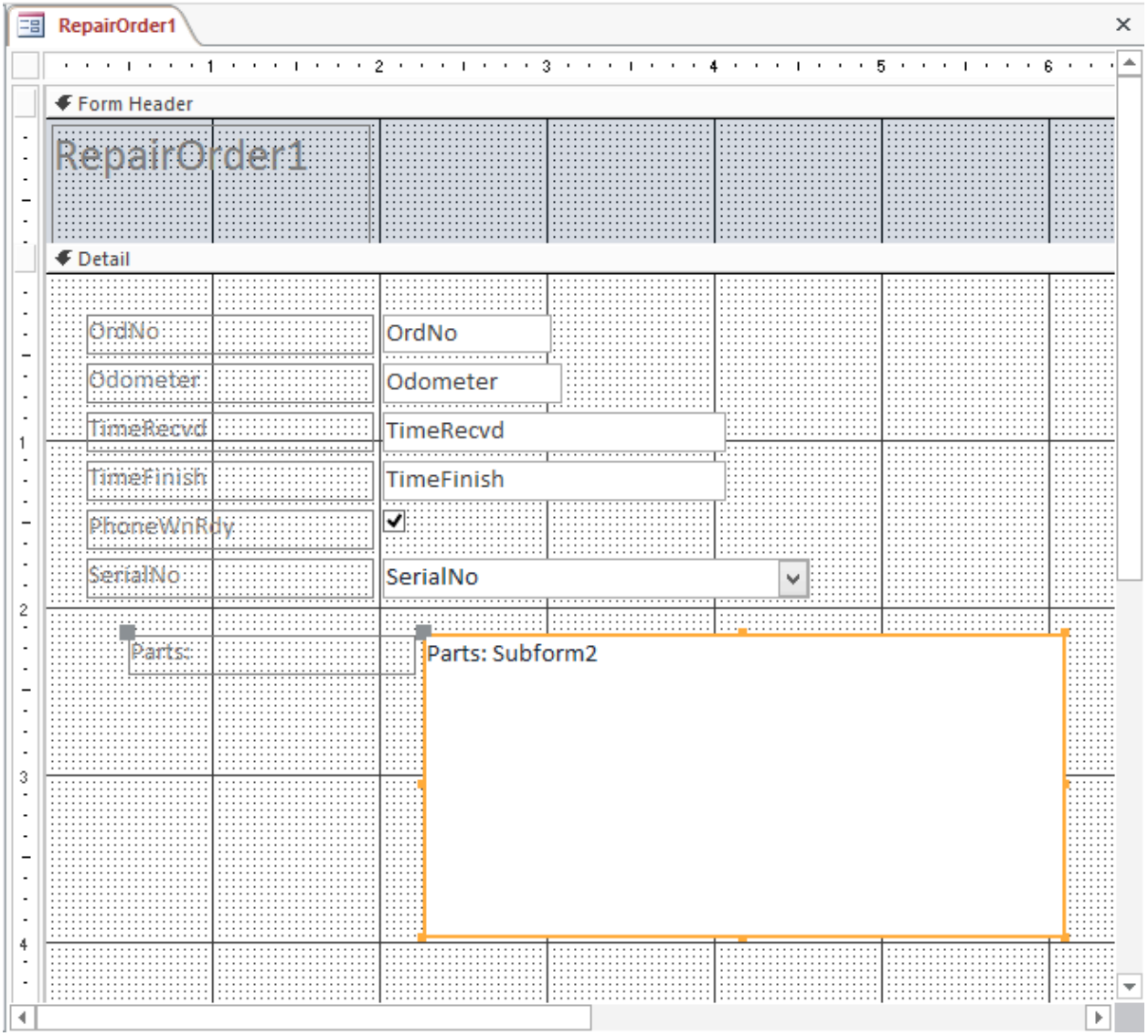


Figure 10: Design View for the *RepairOrder1* Form

5.2.3 Customizing the Main and the Subform

To begin, customize the main form exactly as you did for the initial repair order form. Create a form header, change the title of the form to "Repair Order Form", and expand the form background's height and width.

Before you customize the subform, you must close the main form and then reopen it. For some reason, Access does not allow you to edit the controls inside a subform until the main form is closed and reopened. After reopening the main form, it should appear as in Figure 11. The subform now allows you to manipulate the controls inside of it.

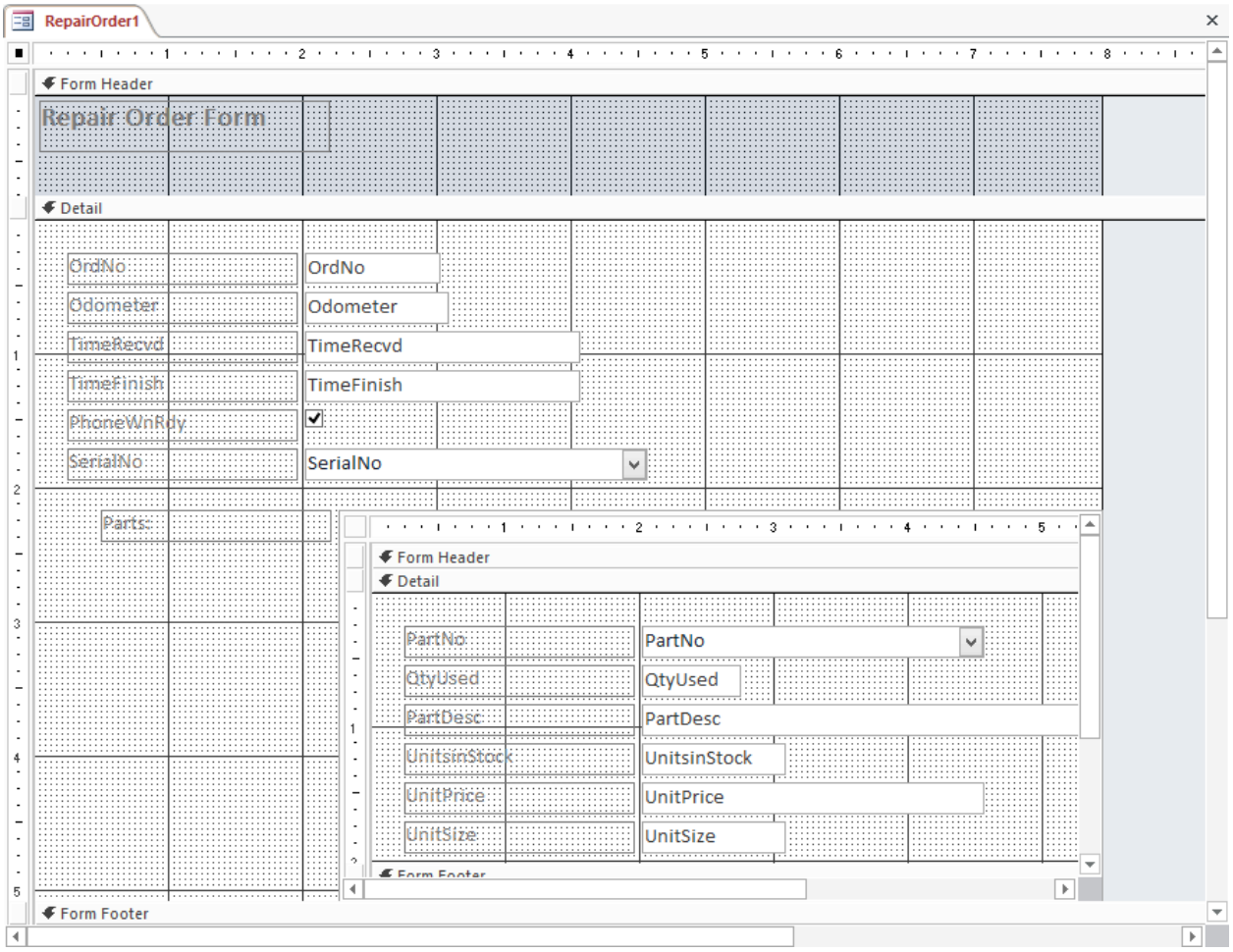


Figure 11: Design View of the *RepairOrder1* Form with the Subform Showing Controls

After reopening the *RepairOrder1* form, you will customize the subform in two ways. First, you will customize the width allocated to the subform in the main form. To perform this customization, you need to open the Property Sheet for the subform as explained below. Second, you will customize the fields inside the subform by opening the subform in design view as explained below.

- Open the Property Sheet of Subform: Select the subform by clicking on one of its edges. Then select **Design → Property Sheet** to open the Property Sheet. Alternatively, click outside the subform if it is already selected. Position the mouse on the edge of the subform, click the right mouse button, and select the **Properties** item. The Property Sheet allows you to change features that apply to the entire subform such as the subform's width.

- Open the Property Sheet for Subform Controls: First click outside of the subform to deselect it, then point inside the subform and double-click on a control inside the subform. You can edit the properties on any control in the subform by double-clicking on it.

Customize the Subform

In the following steps, you need to make changes by opening the Property Sheet of the subform and then opening the subform in design view. Continue with the following steps to customize the subform:

- Expand the Subform: Open the subform's Property Sheet as described previously. Change the *Width* property to "6.5".

- Change Field Names: To change the headings for subform fields in datasheet view, you must change the label's *Caption* property. Note that labels are under the form detail section, not the form header section, as shown in Figure 11. For each label in the subform, double-click on it and change its *Caption* property as indicated in Table 1.

Table 1: Change Label Caption Properties

Label Name	**Label Caption**	**Caption Change**
PartNo_Label	"PartNo"	"Part No"
QtyUsed_Label	"QtyUsed"	"Quantity Used"
PartDesc_Label	"PartDesc"	"Description"
UnitsInStock_Label	"UnitsInStock"	"In Stock"
UnitPrice_Label	"UnitPrice"	"Price"
UnitSize_Label	"UnitSize"	"Size"

Shortcut: Select a label on the form and open its Property Sheet. After changing the *Caption* property, do not close the Property Sheet. While the Property Sheet is still open, select the next label and the Property Sheet changes accordingly.

- Set Column Width: The column width can be set only when the subform is in datasheet view. The easiest way to put the subform in datasheet view is to put the main form in form view. After placing the main form in form view, notice that some of the subform datasheet columns are too wide. To make the column sizes uniform, right-click on the heading of the first column to select the column and open the shortcut menu. Select the **Field Width** command and click the "Standard Width" check box in the window (Figure 12). You can also use the **Best Fit** button for a tighter fit. Repeat this process for each field in the subform datasheet. You may have to use the horizontal scroll bars to access the last few columns.
- Close the Window: When you are finished, your form should appear in Form view as shown in Figure 13. Close the window and save the changes when prompted.
    - In the Database window you should find a form named *RepairOrder1* that displays a hierarchical form when opened.
    - The form named *Parts: Subform2* displays a datasheet when opened.

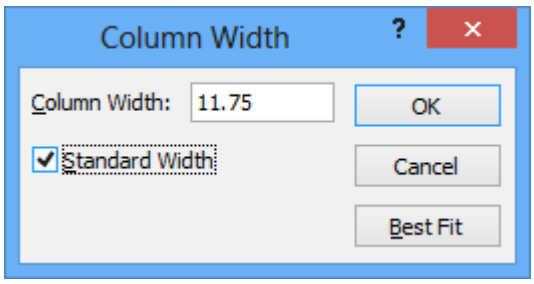


Figure 12: Column Width Window after Selecting the **Field Width** Item

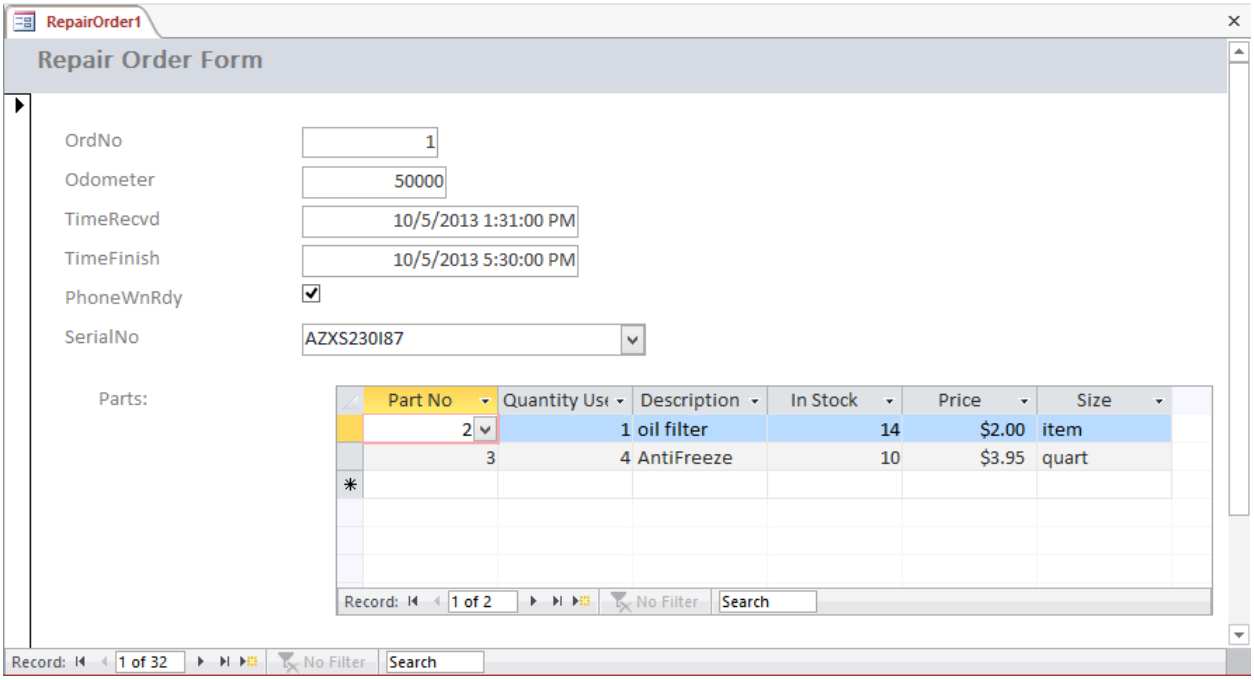


Figure 13: *RepairOrder1* Form

5.2.4 Connecting the Subform with the Main Form

When creating a hierarchical form based on a query, Access may not be able to make the connection between the main form and the subform. The connection is defined by the *Link Master Fields* and the *Link Child Fields* properties. Switch to Design view in the *RepairOrder1* form and examine these properties by opening the Property Sheet for the subform as described in the previous section. The value of both properties should be *OrdNo*. As an experiment, delete the value of both properties and view the form. You should notice that the main form and the subform are not connected. Advancing the records of the main form does not change the records in the subform. Reset the link properties to *OrdNo* and change to form view. The main and the subforms are now connected.

The *Link Master Fields* and the *Link Child Fields* properties are used to alter the query for the subform. When the subform's query executes, a condition is added to the WHERE clause using the Boolean AND operator. For example, assume that the value in the order number field in the main form is "1". The condition `PartsUsed.OrdNo = 1` is added to the subform's query. The alteration of the subform's query occurs automatically by Access without any user action. Without the connection, the subform's query retrieves more records.

The *Default View* and the allow view properties (*Allow Form View*, *Allow Datasheet View* and *Allow Layout View*) control usage of a form. The Default View property has four primary values: one record as a form ("Single Form"), multiple records as a table ("Datasheet"), multiple records as a form ("Continuous Forms"), and split form providing two synchronized views (a single form and a datasheet). For most hierarchical forms, the main form should display as a form and the subform should display as a datasheet. To restrict a hierarchical form in this manner, you need to set the *Default View* and the allow view properties in both the main form and the subform. Starting with the main form in design view, open the Properties window for the form and set the *Default View* property to "Single Form" and the *Allow Form View* property to "Yes". These settings force the main form to show only a single record at a time. On the subform, set the *Default View* property to "Datasheet" and the *Allow Datasheet View* property to "Yes".

5.3 Extending the Main Form

In this section you will revise the main form with a query similar to what you did for the subform. The main form will display data about the customer and the vehicle along with repair order data. You also will be making a few layout changes such as placing a rectangle control around the vehicle and customer data.

5.3.1 Creating the Main Form Query

To enable the main form to display additional data, you need to associate the main form with a query consisting of the *RepairOrder*, the *Vehicle*, and the *Customer* tables. Follow the steps below to formulate the underlying query:

- Open a New Query: Choose **Create → Query Design** in the Queries group. In the Show Table window, add the *RepairOrder*, *Vehicle*, and *Customer* tables.
- Formulate the Query: In Query Design, formulate a query as shown in Figure 14. Follow these tips to develop the query.

1. The query should contain all fields of the *RepairOrder* table.
1. The query should contain the following fields from the *Vehicle* table: *Year*, *Make*, *Model*, *LicenseNo*, *Cylinders*, and *CustNo* (all fields except *State* and *SerialNo*).
1. The query should contain all fields from the *Customer* table except *CustNo*.
2. For practice, verify that the query follows the rules for 1-M updatable queries presented in textbook Chapter 10.
3. Save the query as "MainQuery".

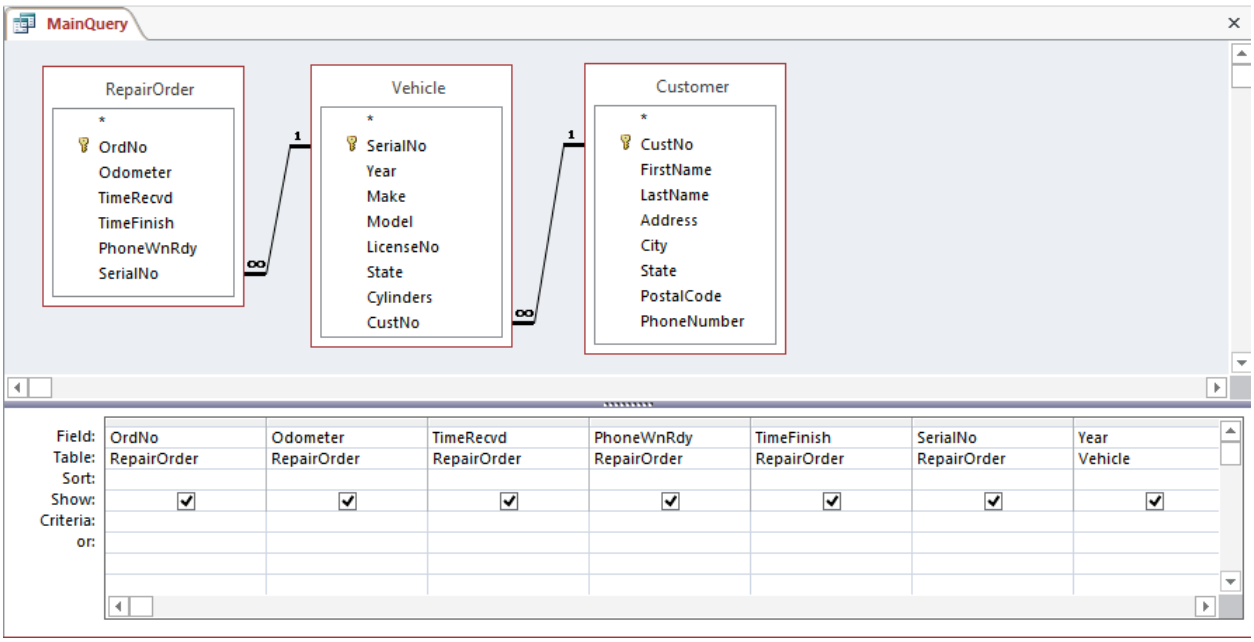


Figure 14: Query Design Window for *MainQuery*

5.3.2 Review of Aligning and Moving Controls

Before revising the main form, you should review ways to align and move controls. Remember from Chapter 4 that you can move a control by selecting it and dragging the control to another location. Revising

the main form consists of moving, aligning, and adding textboxes; changing label and textbox names; and finally enclosing textboxes inside a rectangle control. Here are some additional tips to facilitate formatting, aligning, and moving controls:

- Moving Labels and Textboxes Separately: To move a label separately, select the control and then drag its anchor (filled rectangle in upper left corner) of the control. You can drag the label or bound control separately using the appropriate anchor.
- Copy Label and Textbox Formats with the Format Painter: Select a label or a textbox containing the format that you want to copy. Then click the **Format Painter** icon on the ribbon (**Home → Format Painter** in the Clipboard group), and the cursor changes to a paintbrush. Finally, click on the target label or the textbox to copy the formatting.

- Moving Labels, Textboxes, or Other Controls as One Unit: To group fields as a unit for the purpose of formatting or moving, you can either (a) click the mouse above a corner of the fields to group and drag a grouping rectangle across the fields, (b) simply click on each field while holding the **Shift** key down, or (c) select the controls that you wish to move as a unit and click **Arrange → Size/Shape → Group** in the Sizing & Ordering group of the Ribbon.
  - To Move the Controls: While the unit is still selected, you hold the mouse button down and move the group as one unit to the desired location.
  - To Format the Unit: While the unit is still selected, you can use the toolbar to change fonts or use bold, italics, or underline. Your changes will affect all fields at the same time.
  - To Move Fields as a Unit: While the unit is still selected, drag the unit to the desired location while the cursor is positioned at any point except an anchor in the upper left corner of any control in the unit. If you drag from an anchor, only the individual control will move.
  - Use the **Arrange** Tab: You also can use commands in the Position and Sizing & Ordering groups. You can manipulate the alignment (grid, top, bottom, left, and right), size (tallest, shortest, widest, shortest, narrowest, grid, fit), anchoring (top left or right, bottom left or right, stretch down, stretch across top, stretch across bottom, stretch down and across, and stretch down and right), and spacing (increase, decrease, or equalize horizontal or vertical spacing) of a group of controls.

5.3.3 Modifying the Main Form

Now that you have created the query, you can revise the main form. You can begin the revision of the main form by either (1) using the Form Wizard with a main form query and a subform query or (2) copying and pasting the *RepairOrder1* form and modifying it. The Form Wizard is most applicable when you are starting from scratch with a main form query and a subform query. The copy-and-paste method is easier when you have an existing form that needs a small amount of revision. This section demonstrates the copy-and-paste method because the existing main form only needs a fair amount of revision. Appendix D demonstrates the Form Wizard method using a main form query and a subform query.

1. Copy and Paste the *RepairOrder1* Form: In the Database window, copy *RepairOrder1* and paste it as *RepairOrder2*. Open *RepairOrder2* in design view.

2. Change Form Properties: Open the Form Properties window by clicking **Design → Property Sheet**. In the *Record Source* area click the arrow to reveal a list and choose *MainQuery*. Change the *Caption* property to "RepairOrder2".

3. Move the Subform and the Textboxes: Make room for additional textboxes by moving the subform further down the form.

   1. Select the subform and point the cursor at any border until the arrows cursor (four directional arrows) appears.

2. Drag the subform down about two grid boxes. You may need to expand the bottom of the form at the page footer line. Figure 15 shows the arrangement of textboxes and the subform.

4. Change Label Captions: In the main form, select the label of a textbox and click inside to edit the caption. Change the label names according to Table 2.

Table 2: Changes for Caption Properties

Existing Caption	**New Caption**
OrdNo	"Order No."
Odometer	"Odometer:" (add colon)
TimeRecvd	"Time Received:"
TimeFinish	"Time Finished:"
PhoneWnRdy	"Phone When Ready:"
SerialNo	"Serial No."

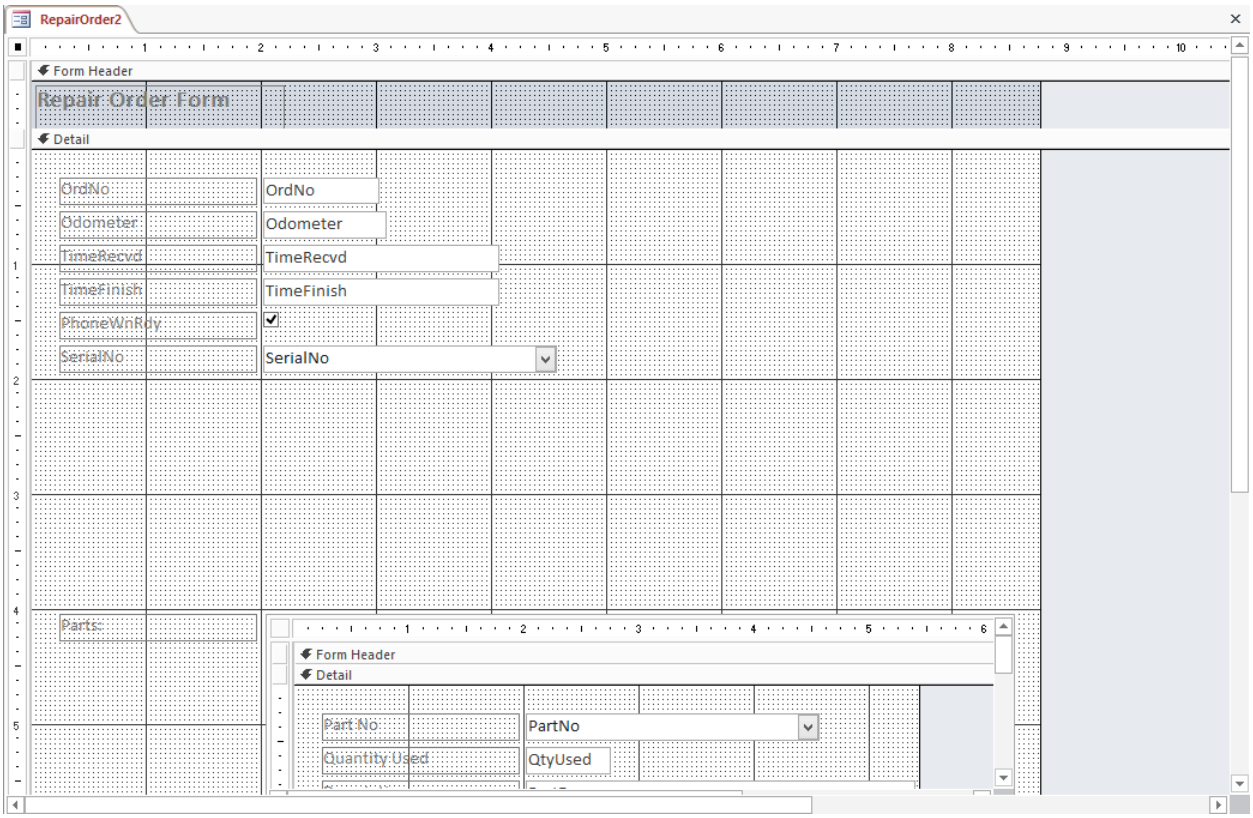


Figure 15: Design View Window for *RepairOrder2* after Changes through Step 6

5. Set Label Sizes: Standardize the label sizes. Select a label and right-click to open the shortcut menu. Select **Size** and choose **To Fit**. Do this for each label on the form.

6. Set Textbox Sizes: Standardize the textbox sizes by opening each Property Sheet and setting the *Width* property to "1.0".

Shortcut: Select the first textbox (*OrdNo*) and double-click to open its Property Sheet. After setting the property, do not close the window. Instead double-click the next textbox and the Property Sheet will change to the next textbox.

7. Add Textboxes: Add textboxes and change label names according to Table 3. Drag a field name from the field list and drop it onto the form. If the field list is not on the screen, click **Design → Add Existing Fields** in the Tools group. You can use the **Arrange → Stacked** command in the Table group to make a neatly formatted layout for each group of controls. Alternatively, you can use the tools in the Size & Ordering and Position groups to modify the display. If necessary, move the subform down further to accommodate the new textboxes. You should try to achieve a layout similar to Figure 14.

Table 3: Changes for Caption Properties

Existing Caption	**New Caption**
CustNo	"Customer No."
FirstName	"First Name:"
LastName	"Last Name:"
Address	"Address:" (add colon)
City	"City:" (add colon)
State	"State:" (add colon)
PostalCode	"Zip Code:"
PhoneNumber	"Phone No."
Year	"Year:"
Make	"Make:"
Model	"Model:"
LicenseNo	"License No:"
Cylinders	"Cylinders:" (add colon)

8. Move *Serial No*. directly above *Year* so the vehicle data are together.

9. Label the Columns: Select the **Label** tool from the toolbox and place a new label above each column (Figure 14). Type "Vehicle Information" and "Customer Information" as the label captions. Change label sizes **To Fit** to make it easier to drag and center the label above the column. While a label is still selected, click the capital **U** on the toolbar to underline the label's text.

10. Enclose Fields in Rectangle Controls: Refer to Figure 16 to add rectangle controls.

    - In the toolbox, click the box with thick lines that denotes the rectangle control. When you move the cursor to the form, it changes to a cross.
    - Point the cursor at a corner of the column and while holding down the mouse button, drag the box large enough to enclose the column. If you need to adjust the box size, point the cursor on one of the small boxes attached to the box. When you see a double arrow, hold the mouse button down while extending (or shrinking) the box.
    - Set *Back Color*: Double-click on the rectangle to open its Properties window. Change the *Back Color* property to white and the *Special Effect* property to "Sunken". The enclosed textboxes disappear as they are hidden under the rectangle. To make them appear, click **Arrange → Send to Back** in the Position group. This action moves the rectangle to the background and the textboxes to the foreground (Figure 16).

11. Set Tab Order: When you start at the *Order No*. field and tab through the textboxes, you will find that the order needs a few changes.

    1. Click **Design → Tab Order…** in the Tools group to open the Tab Order window.
    2. Follow the instructions in the window and move *Parts: Subform2* to the end of the list and place *CustNo* directly after *PhoneWnRdy*. Click **OK** when you are finished.

12. Center the Form Title: Now that the form is wider, it is appropriate to center the form title. Select the label and right-click to open the shortcut menu. Select **Size** and choose **To Fit**. Next, drag the label to the center of the form.

13. Toggle to Form View: Figure 17 shows the completed *RepairOrder2* form. To test the form, tab through the fields and click through a few records using the record scroll arrows below. When you are finished, close the form and save changes when prompted.

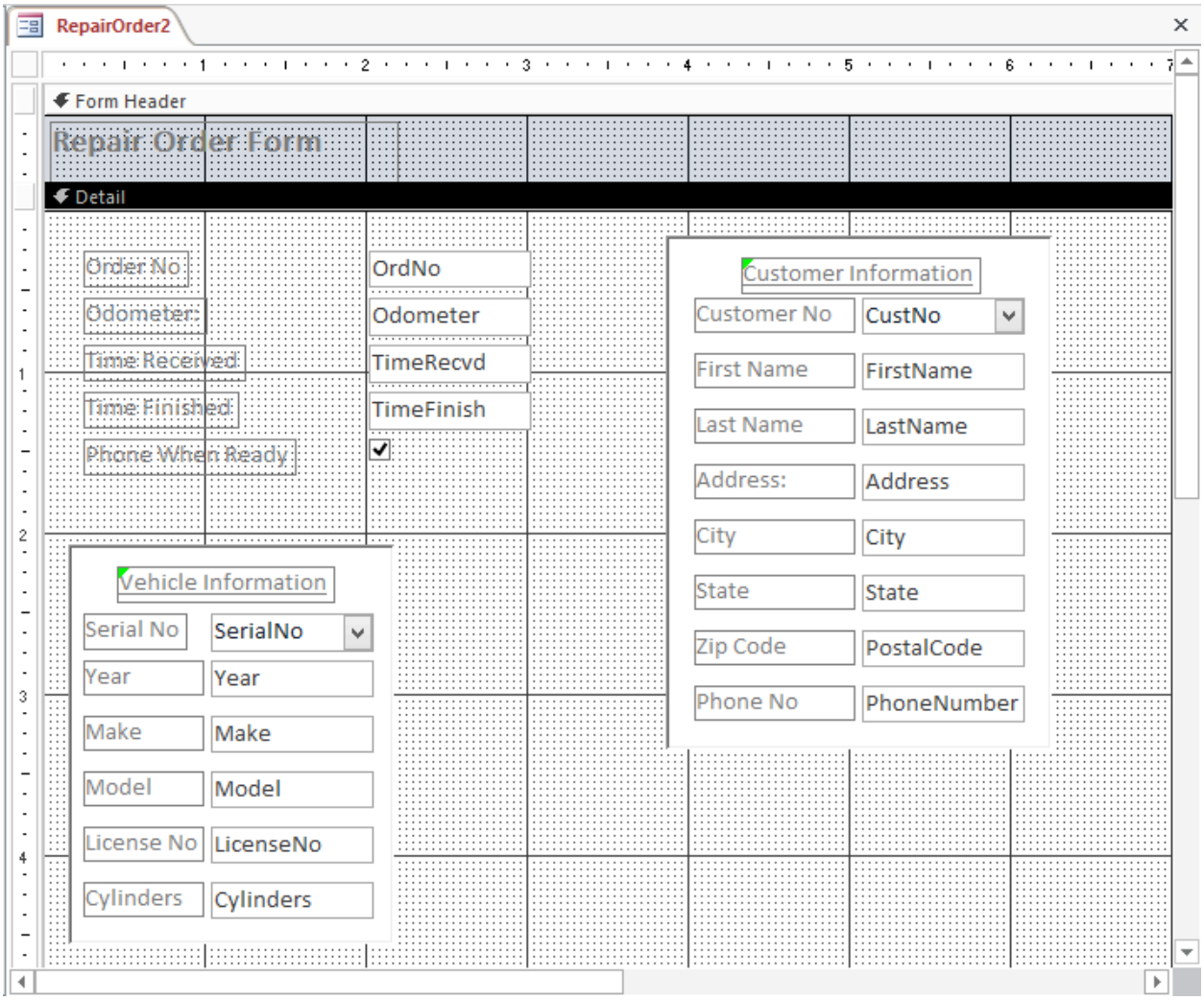


Figure 16: Design View Window for *RepairOrder2*

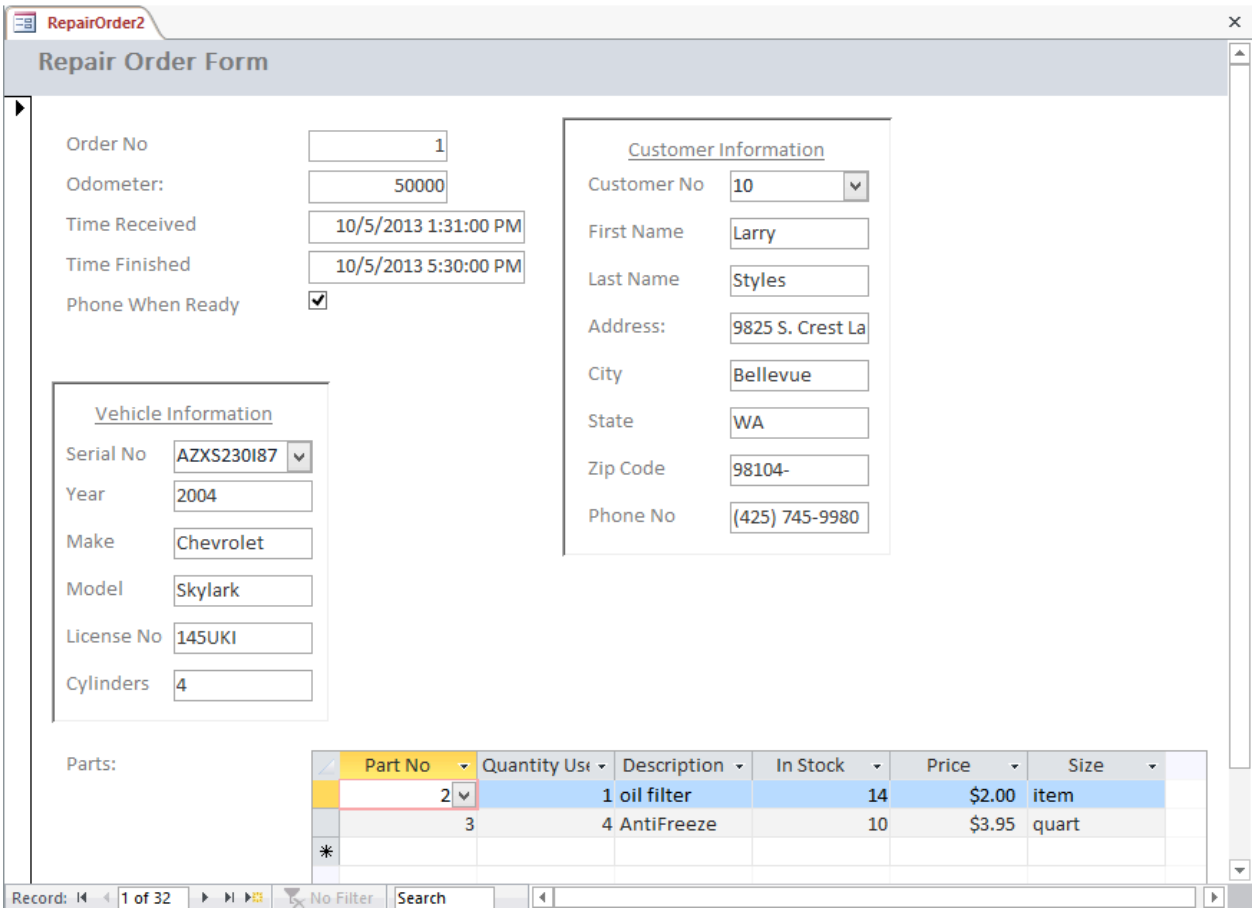


Figure 17: Completed *RepairOrder2* Form

5.3.4 Using the Repair Order Form for Data Manipulation

Before proceeding to other form design issues, you should understand how the *RepairOrder2* form supports data manipulation. As discussed in textbook Chapter 10, you should identify tables that can be changed by using a form. This decision is fundamental to understanding the business process that uses the form. For the *RepairOrder2* form, the *RepairOrder* table can be changed in the main form and the *PartsUsed* table can be changed in the subform. The other tables (*Vehicle*, *Customer*, and *Part*) are read-only.

Even though the fields of the *Vehicle*, *Customer*, and *Part* tables should be read-only, the current form design supports updates to these fields.[12] The only fields in which change should be permitted are the *RepairOrder* fields (*OrdNo*, *RepairDate*, *Odometer*, *TimeRecvd*, *TimeFinish*, *PhoneWnRdy*, and *SerialNo*) in the main form and the *PartsUsed* fields (*PartNo* and *QtyUsed*) in the subform. Note that the *CustNo* field in the main form is read-only.

To prevent changes to the other fields, the user should not be able to establish focus in the read-only fields. If a control accepts focus, the user can enter data into the control. To prevent a user from establishing focus, set the *Enabled* property to "No" and the *Locked* property to "Yes". For the *RepairOrder2* form, you should set these properties for all form fields except the form fields bound to the previously listed fields of the *RepairOrder* and the *PartsUsed* tables.

[12] The underlying queries support updates to fields of the parent table but do not support insert operations to the parent table.

To see the effect of setting the *Enabled* and the *Locked* properties, open the *RepairOrder2* form in Form view. Notice that you cannot establish focus in the form fields bound to fields from the *Vehicle*, the *Customer*, and the *Part* tables.

AutoLookup Queries

You may wonder where read-only form fields obtain values if a user cannot enter values. AutoLookup queries provide values for read-only form fields if these fields are bound to tables participating in a 1-M relationship. An AutoLookup query retrieves data from a parent table when a foreign key value is entered from the corresponding child table. You do not need to write AutoLookup queries. Access automatically creates and executes AutoLookup queries when a user enters a foreign key value.

AutoLookup: An Access feature that provides values for fields of a parent table when the foreign key value is entered in the corresponding child table.

To see results of AutoLookup queries, open *RepairOrder2* in form view with existing records displayed. If you select a different serial number, AutoLookup queries display *Customer* and *Vehicle* data relating to the selected serial number from the *RepairOrder* table. This situation involves two AutoLookup queries. In the first AutoLookup query, the *RepairOrder* table is the child table with *SerialNo* as the foreign key and the *Vehicle* table is the parent table. In the second AutoLookup query, *Vehicle* is the child table with *CustNo* as the foreign key and *Customer* is the parent table. The second AutoLookup query displays *Customer* data after the first AutoLookup query displays *Vehicle* data. Both AutoLookup queries execute so fast that you do not notice.

For another AutoLookup query example, select a different part number in the subform. An AutoLookup query displays the related *Part* fields. The AutoLookup query works in this situation because *PartNo* is the foreign key of the child table (*PartsUsed*) and the subform contains controls bound to the parent table (*Part*).

5.4 Using Combo Boxes

The third revision of the repair order form involves combo box extensions. To provide a context for users, you will add fields to the combo box lists for the *SerialNo* and the *PartNo* form fields. Some users may need to see more than just the primary key value to choose the correct vehicle or part. To avoid mistakes and reduce the length of a combo box list, you will modify the query for the *Serial No.* combo box. When a customer number value is displayed on the form, the *Serial No.* combo box will display only vehicles of that customer. On occasion you may find it necessary to change the kind of control for a form field. To provide practice, you will change the *CustNo* field from a combo box to a textbox because it is a read-only form field.

5.4.1 Adding Fields to a Combo Box

The following instructions guide you to add fields to the *Serial No*. combo box in the main form and the *Part No*. combo box in the subform. To begin, open the *RepairOrder2* form in Design view.

1. Add Fields to the *Serial No.* Combo Box: Double-click on the *Serial No.* combo box to open its Property Sheet:
2. Set the *Row Source* Property: Click in the *Row Source* property area and click on the ellipsis (**…**) button to open Query Design in the view pane.
    1. Drag the *Year, Make*, and *Model* fields down into the query grid. In the *Make* column, click in the *Sort* row and choose "Ascending" (Figure 18).
    2. Close Query Design in the view pane. When prompted, click **Yes** to save these changes.
3. Set Other Properties According to Table 4:

Table 4: *SerialNo* Combo Box Properties and Settings

Property	**Setting**
Column Count	"4"
Column Heads	"Yes"
Column Widths	"1.0"
List Width	"4.0"

4. Toggle to Form View: To see how the form looks, toggle to Form view. If you are satisfied with the result, toggle back to Design view.

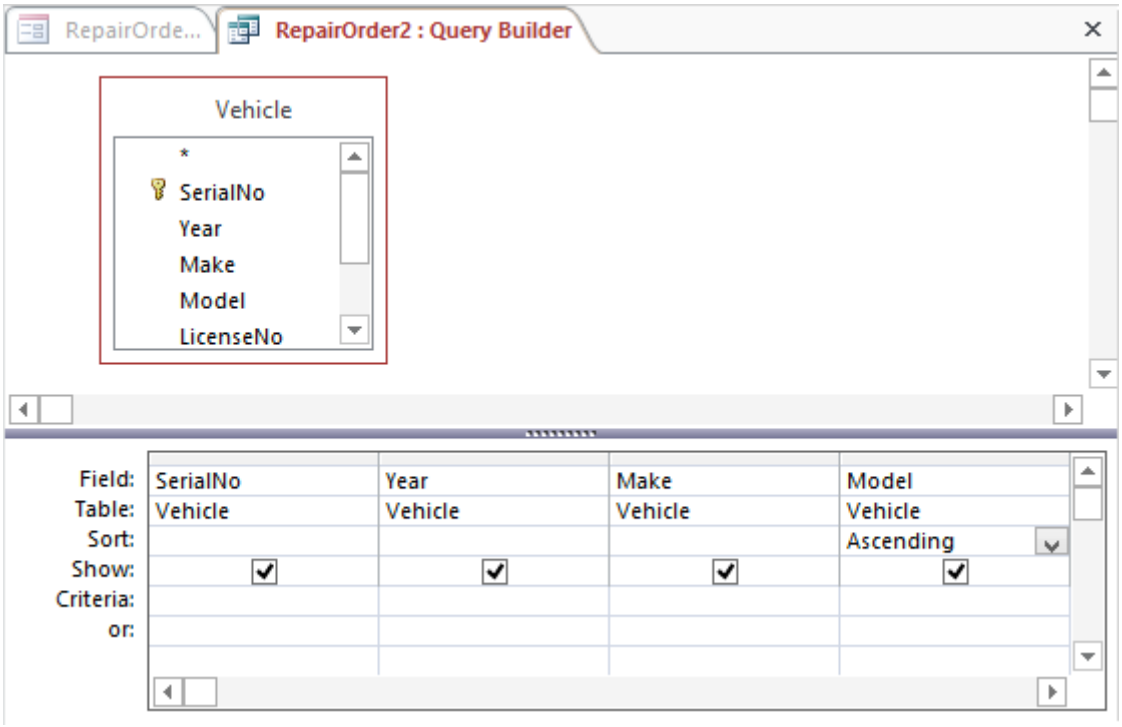


Figure 18: Query Design for the *Serial No.* Combo Box

5. Add Fields to the *Part No.* Combo Box: Double-click to open the subform in design view (open in full view). Double-click on the *PartNo* field to open its Property Sheet.

6. Set *Row Source* Property: Click in the *Row Source* property area and click on the ellipsis (**…**) button to open Query Design in the view pane.

   3. Drag the *PartDesc* field down into the query grid. Click in the *Sort* row and choose "Ascending". (See Figure 19)

   4. Close Query Design in the view pane. When prompted, click **Yes** to save these changes.

7. Set Other Properties: Use Table 5 as a guide to set other properties.

Table 5: *Part No.* Combo Box Properties and Settings

Property	**Setting**
Column Count	"2"
Column Heads	"Yes"
Column Widths	"1.0"
List Width	"2.0"

8. Toggle to Datasheet View: Check the appearance of the subform in datasheet view. Click the combo box to observe the list. If you are satisfied with the result, toggle back to Design view. Then close Design View in the view pane to return to the *RepairOrder2* form.

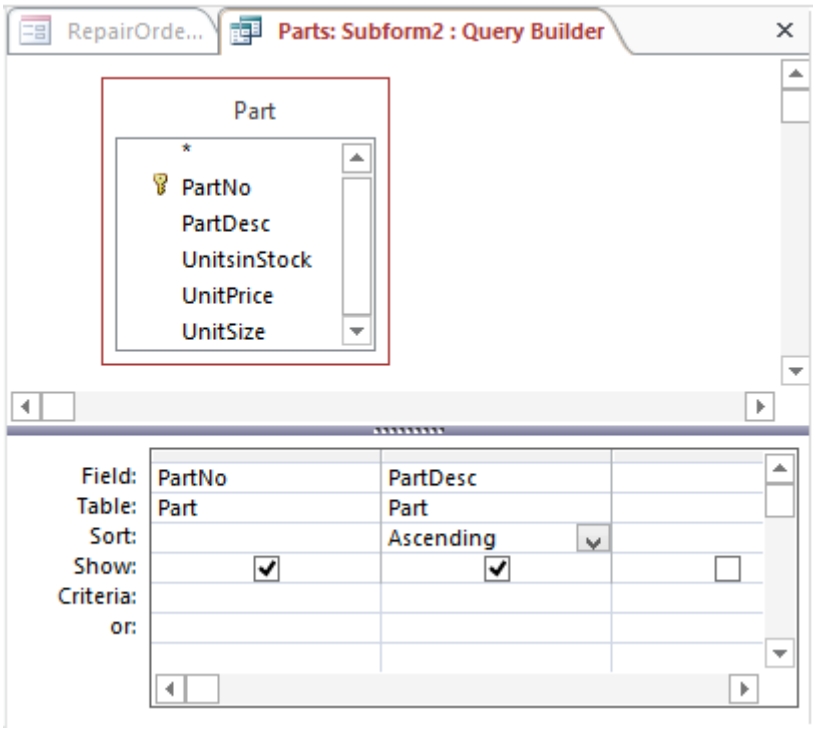


Figure 19: Query for the *Part No* Combo Box

5.4.2 Restricting Values in a Combo Box List

When possible, you should make the values displayed in a combo box consistent with other values in a form. This practice saves a user's time (fewer list values to view) and reduces errors from choosing an incorrect list value. You will apply this principle to the *Serial No.* combo box in the main form. You will modify the combo box query to display only the vehicles for the customer number on the form. In addition, your combo box query should display all vehicles when no customer number is displayed on the main form (data entry mode). With the main form open in design view, use the following instructions to modify the combo box query.

1. Open the Property Sheet: Double-click the *Serial No.* combo box to open its Property Sheet.

2. Edit the *Row Source* Property: Click the ellipsis (**…**) button in the *Row Source* property to open the Query Design window.

3. Enter a Condition for the *CustNo* Field: Drag the *CustNo* field from the *Vehicle* table to the query grid. Edit the query grid properties as follows:

   - Remove the checkmark from the *Show* property because this field is used only for testing a condition.
   - In the *Criteria* property, open the expression builder by right-clicking and selecting the **Build…** item.
   - Expand the Forms folder in the left pane until you have opened the fields of the *RepairOrder2* form, as depicted in Figure 20. You need to expand three folders in the left pane to see the fields.
   - Double-click the *CustNo* field. You may need to scroll to find the *CustNo* field. A reference to the *CustNo* field on the *RepairOrder2* form appears in the text window (Figure 21).
   - Close the Expression Builder window to return to the Query Design window.
   - You have just entered a condition to ensure that the *Vehicle.CustNo* field matches the *CustNo* form field. To see the complete comparison expression, switch to SQL View. You will see that the expression is "`((Vehicle.CustNo)=[Forms]![RepairOrder2]![CustNo])`". After viewing the complete expression, switch back to Design View.

4. Enter a Condition for Null Customer Numbers: The condition on *CustN*o works fine if editing an existing repair order. For data entry, you want all vehicles to display in the combo box list. To show all vehicles in data entry mode, you must add a condition to test for null customer numbers. Use the following guidelines to add a new column in the query grid and an OR condition to test for null customer numbers.

   - Scroll to a new column in the query grid. Instead of selecting a column, open the Expression Builder window using the right mouse button.
   - As before, you need to select the *CustNo* form field. Follow the directions in Figures 20 and 21. When finished, close the Expression Builder window.
   - Remove the checkmark in the *Show* property of this new field.
   - Enter a condition in the *or* property of this new field. Type "Is Null" in the text area.
   - To see the complete comparison expression, you should switch to SQL View. You will see that the expression for the OR condition is "(([Forms]![RepairOrder2]![CustNo]) Is Null)". The entire WHERE clause for the combo box query is

```
((Vehicle.CustNo)=[Forms]![RepairOrder2]![CustNo])          OR
(([Forms]![RepairOrder2]![CustNo]) Is Null)
```

5. Review the Query: The final query appears in Figure 22. Note that the *Criteria* values are truncated because of space limitations. If you open the Expression Builder window, you will see the entire property value. You also can see the entire SELECT statement by toggling to SQL view.

6. Return to the Main Form: Close and save Query Design in the view pane to return to the main form.

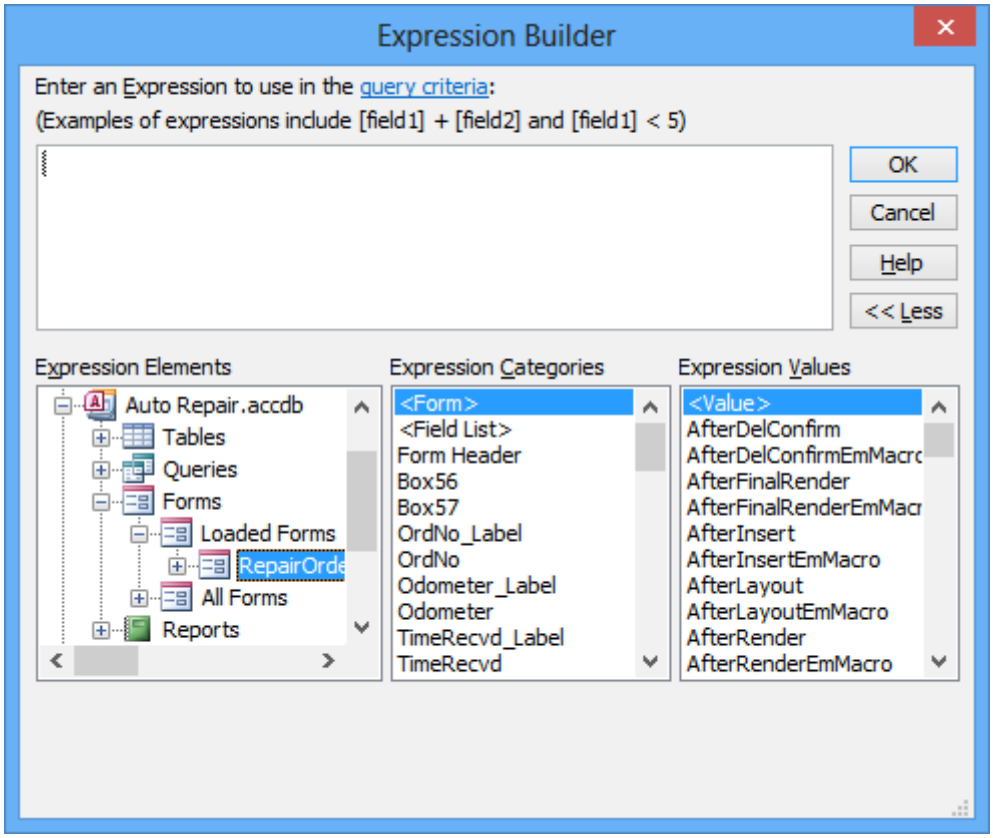


Figure 20: Expression Builder with *RepairOrder2* Fields Selected

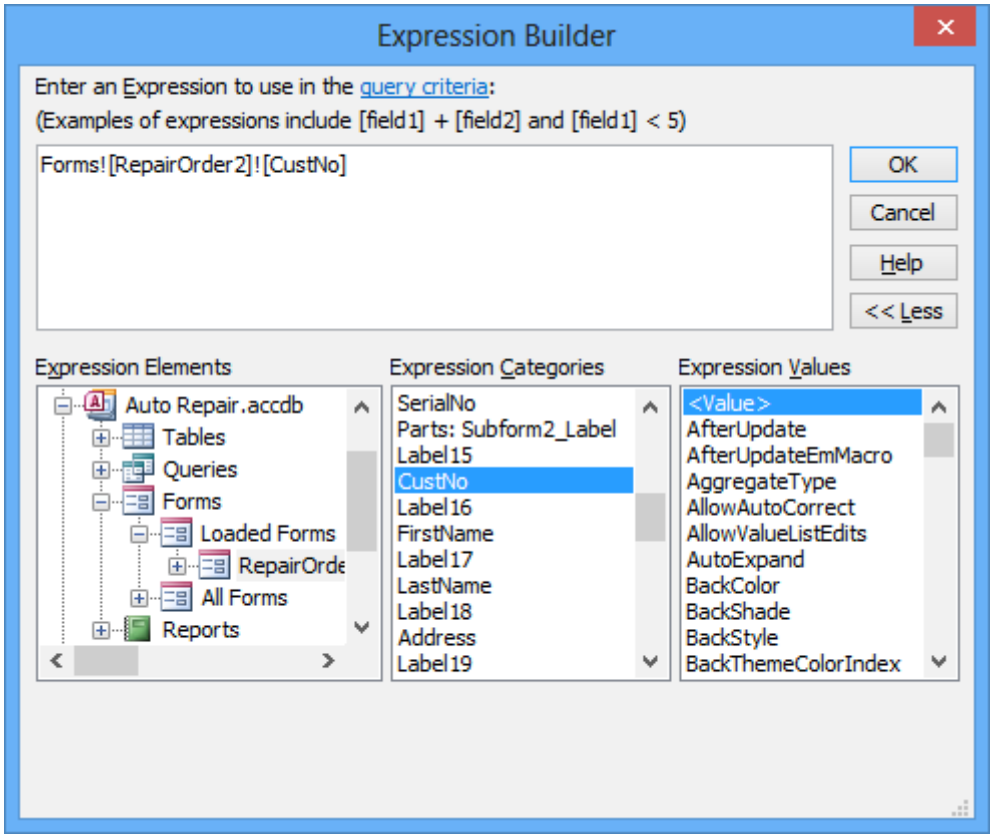


Figure 21: Expression Builder with Reference to *CustNo* Form Field

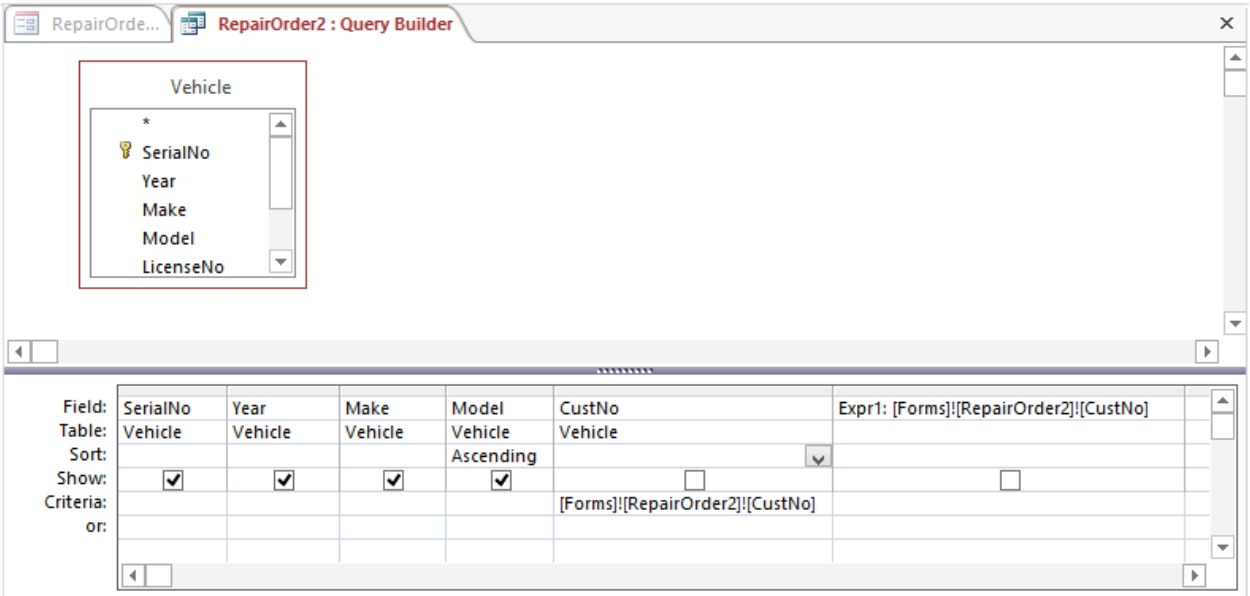


Figure 22: Query Design Showing the Finished Query

To see the effect of your revised query, open *RepairOrder2* in form view. Click the *Serial No.* combo box and notice that the list is now much shorter. The revised query displays only vehicles owned by the

customer on the form. Place the form in data entry mode by clicking **Home → New**. Now all vehicles are displayed when you view the *Serial No.* combo box list. The OR condition in the query has produced the entire list of vehicles.

There is a problem when viewing vehicles in the combo box as you advance among records. Start with the first *RepairOrder* record and note the list of vehicles in the combo box. Advance to another *RepairOrder* record with a different *CustNo* value. Notice that the combo box list has not changed. The combo box query needs to be refreshed, click **Home → Refresh All → Refresh** in the Records group to execute the combo box query. The combo box list now changes to reflect the different customer number.

In Chapter 8, you will write a macro to make the combo box refresh automatically. For now, you need to manually refresh the combo box query.

5.4.3 Changing a Combo Box to a Textbox

The *Customer No.* combo box should be a read-only field. An AutoLookup query will generate the value for customer number when a user selects a vehicle's serial number. A combo box is not a good choice of a control because the user should never select values from the list. Therefore, you will change the combo box to a textbox and make it read-only. (Remember that you make a form field read-only by setting its *Enabled* property to "No" and its *Locked* property to "Yes".)

1. Change the Combo Box to a Textbox: Right-click the *Customer No.* combo box and select **Change To → Text Box** (Figure 23). Now the *CustNo* field is a textbox.
2. Make the Textbox Read-Only: Double-click the *Customer No.* textbox to open its Property Sheet. Change the *Enabled* property to "No" and the *Locked* property to "Yes".
3. Toggle to FormView: Notice that the *Customer No.* combo box is now a textbox, as shown in Figure 24. You cannot obtain focus on the textbox because of the settings for the *Enabled* and the *Locked* properties. If you are satisfied, close the form window and save changes when prompted.

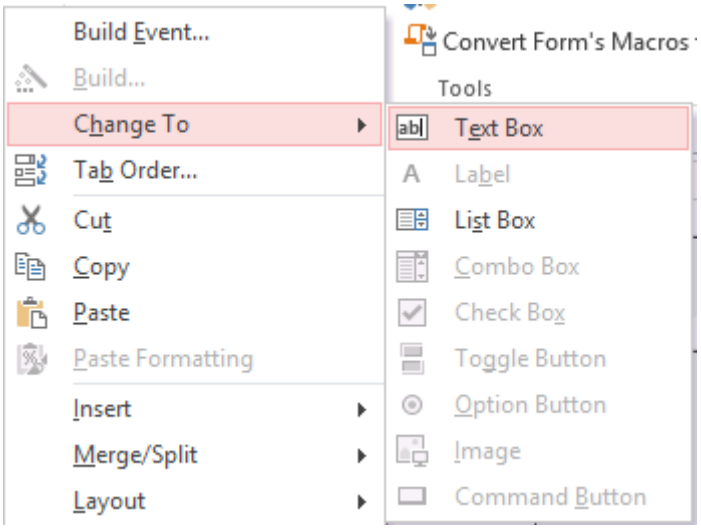


Figure 23: Menu Selection to Change a Form Field to a Textbox

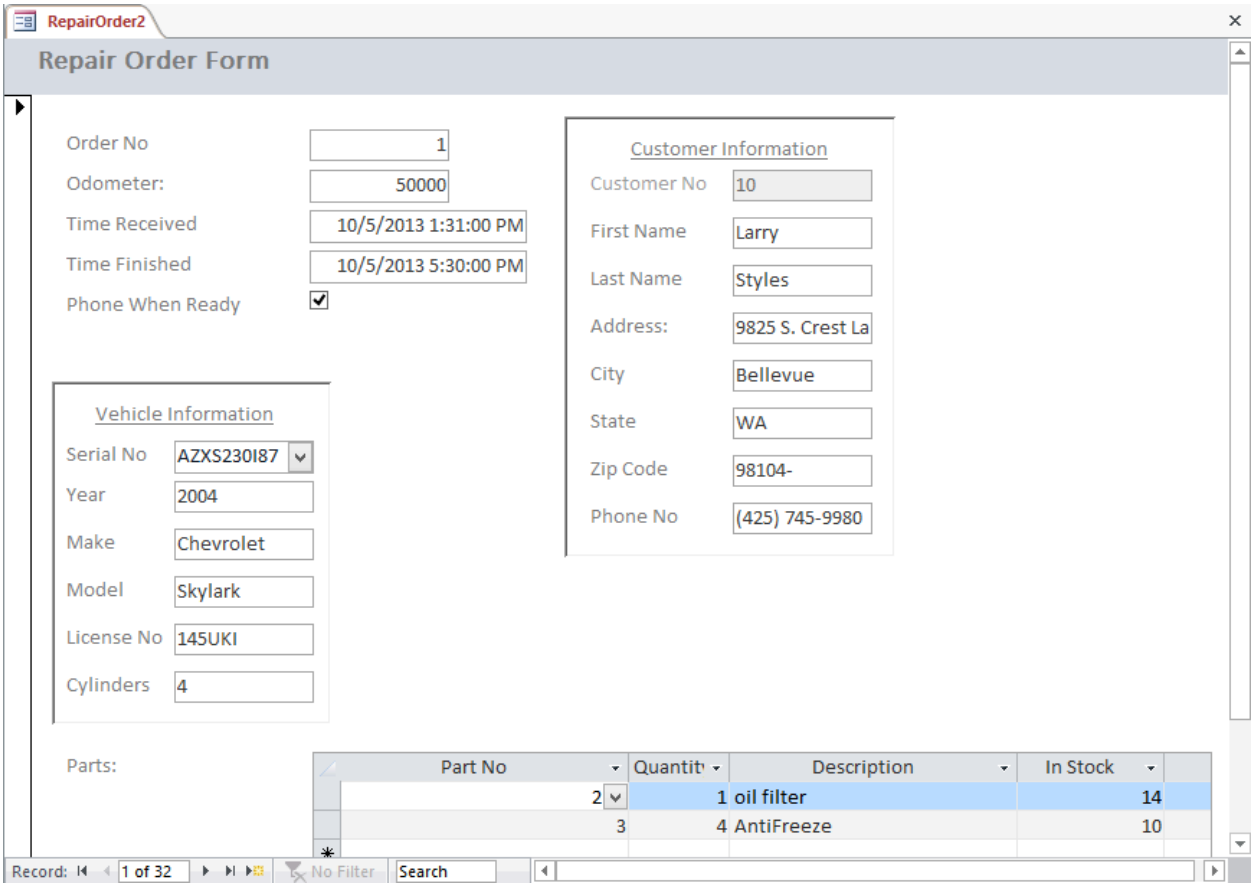


Figure 24: Revised *RepairOrder2* Form

5.5 Sharing Computations between Forms

The last feature to demonstrate involves shared computations between fields of the subform and the main form. You want to compute the total cost for each part on the subform and display the total cost on the main form. You will need to change both the main form and the subform to achieve this goal.

5.5.1 Making Computations in a Subform

To compute the total part cost, you will add a read-only textbox in the form footer of the subform. Since the new textbox references the part cost (quantity used times unit price), you will add a new textbox in the subform to contain the computed part cost. The following steps describe the actions you need to take. To begin the instructions, open the *RepairOrder2* form in Design view.

1. Open the Subform in Design View: When the subform details are exposed in Design view as in Figure 15, you need to use the pull-down menu to open the subform in Design view in a separate window. Click the boundary of the embedded subform and then right click to open the menu. Select **Subform in New Window** to open it in Design view in a separate tabbed window.

2. Add a Textbox to the Subform: Drag a new textbox to the form detail section at the right end of the subform. Note that you may have to expand the width of the form detail section to make room for the textbox. The textbox is initially unbound, as shown in Figure 25.

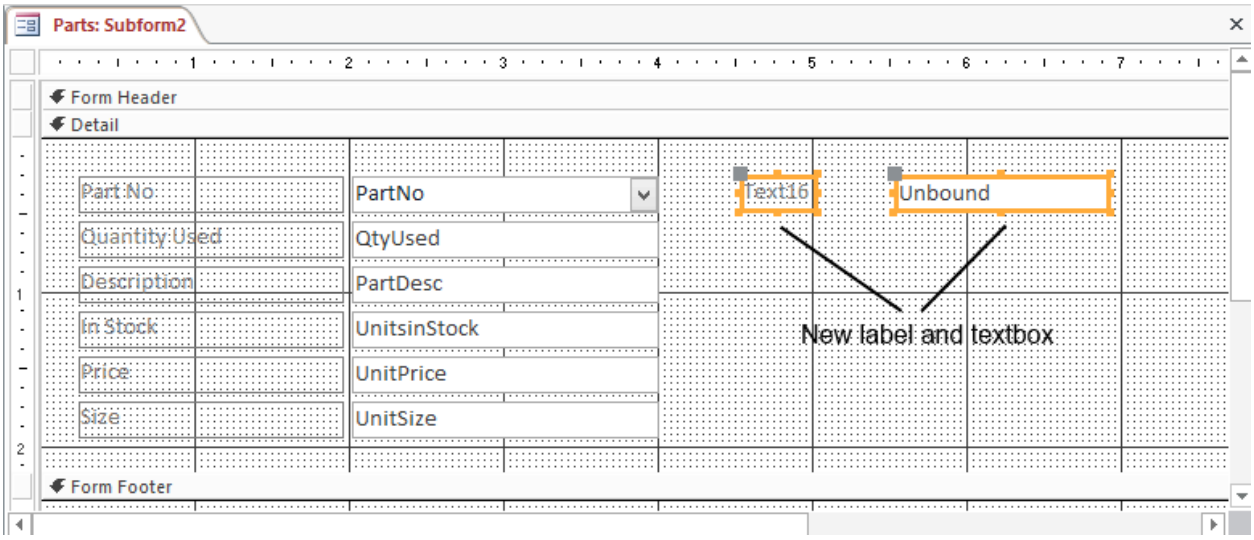


Figure 25: Adding a Textbox to the Subform

3. Delete the Associated Label: Select the associated label and delete it.

4. Set Textbox *Name* Property: Double-click the textbox to open its Property Sheet. Change the *Name* property to "PartCost".

5. Set the *Control Source* Property: The *Control Source* property should be a new field from *SubQuery1*. Remember that this subform (*Parts: Subform1*) uses *SubQuery1* as its *Record Source*.

    1. Open *Subquery1*: Open the Property Sheet of the subform. Then, open Query Design on *Subquery1* by selecting the ellipsis (**…**) button in the *Record Source* property.

    2. Add a Field: Scroll to the next empty field in the query grid and click the right mouse button inside the first row. From the shortcut menu select **Build…** to open the Expression Builder window. Type the following expression in the window: `PartCost: [QtyUsed]*[UnitPrice]`. Close the Expression Builder and the Query Design windows and save the changes.

    3. Set the *Control Source*: Select the new textbox and open its Property Sheet if not already opened. In the *Control Source* property, choose the field *PartCost.*

5. Set Other Properties: With the Property Sheet of the new textbox still open, set other properties according to Table 6.

Table 6: Textbox Properties and Settings

Property	**Setting**
Name	"Part Cost"
Format	"Currency"
Enabled	"No"
Locked	"Yes"

6. Add a Textbox in the Subform Footer: The costs for all the parts must be computed in the subform before the total cost can be referenced in the main form. Since this textbox should not appear in the subform, place it in the form footer of the subform, as depicted in Figure 26. You will find it easier to add the textbox in the footer of the subform if you first open the subform in design view in a separate window.

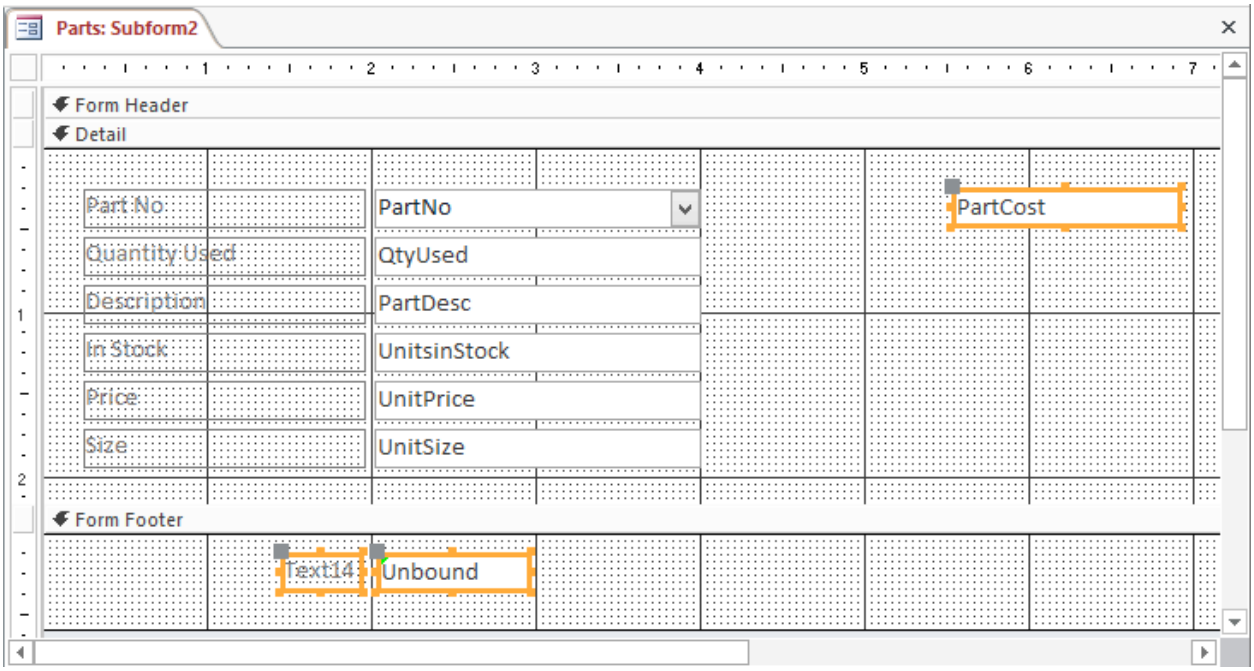


Figure 26: Textbox Added in the Subform Footer

7. Set *Caption* and *Name* Properties: In the Property Sheet of the new label, type "Total Cost" in the *Caption* property. In the Property Sheet of the new textbox, type "TotalCost" in the *Name* property.

8. Set the *Control Source* Property: In the *TotalCost* textbox, click the ellipsis (**…**) button to open the Expression Builder window. After typing `=Sum([PartCost])`, click **OK** to close the Expression Builder window. Save your work when finished.

9. Toggle to Datasheet: View the *Part Cost* column (Figure 27). If you are satisfied, close the datasheet and save the changes. Design view appears again in the view pane.

10. Expand the Subform: To fully expose the new *PartCost* column in form view, the subform must be expanded. Open the Properties window of the subform and change the *Width* property to "7.7". Close the Properties window when finished.

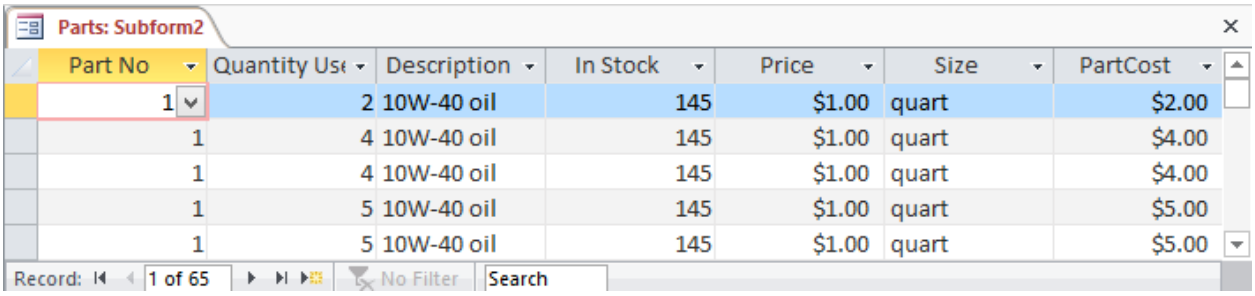


Figure 27: Datasheet of *Parts: Subform2*

5.5.2 Referencing Subform Fields

In the main form, you need to add a textbox to reference the total parts cost from the subform. The following steps guide you to completing the form.

1. Add a Textbox to the Main Form: Place a new textbox on the main form directly above the subform, as depicted in Figure 28.

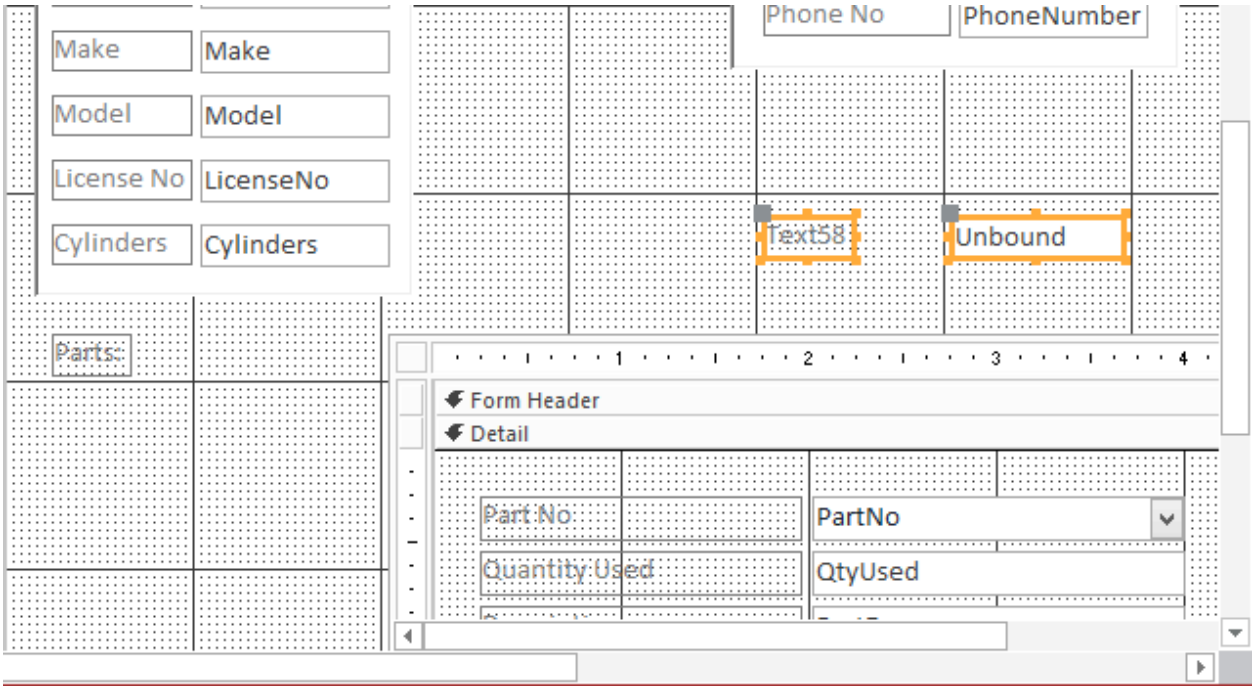


Figure 28: Adding a Textbox to the Main Form

2. Type Caption in Label: Select the new label and type inside "Total Parts Charge". Next, select the label again, click the right mouse button, select **Size**, and choose **To Fit** (Figure 29). Also in this menu select **Fill/Back Color** and choose the "Lightest Gray" color.

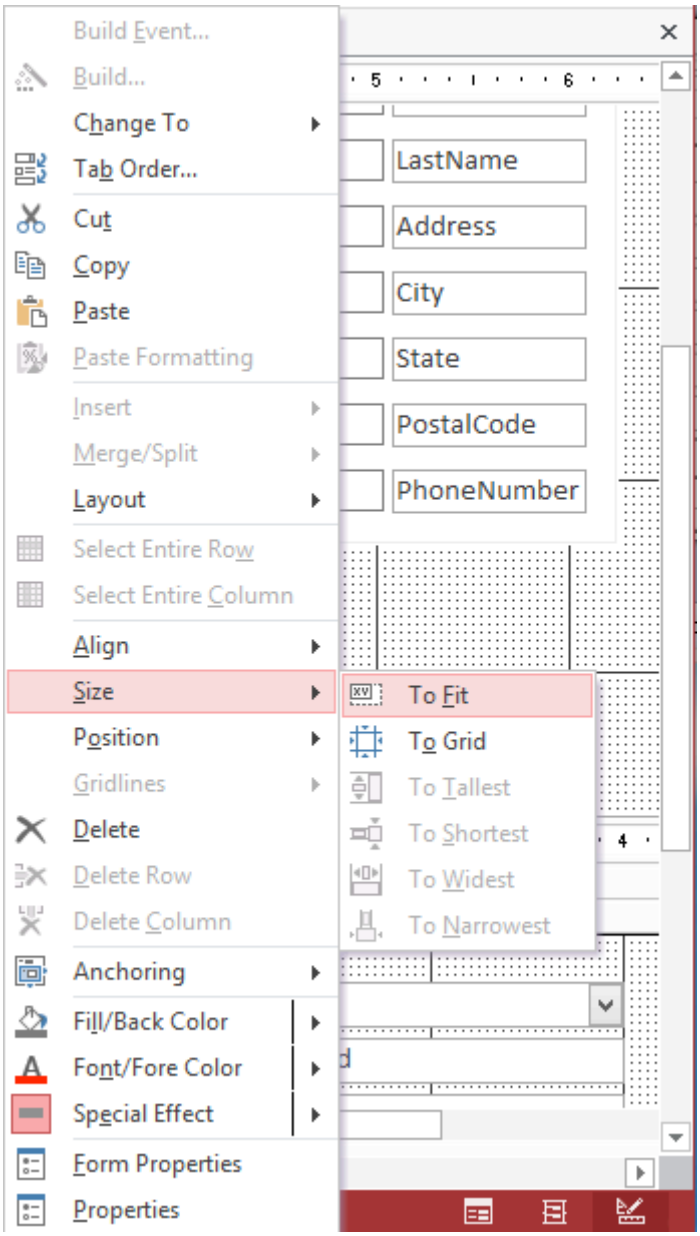


Figure 29: Drop Down Menu Showing **Size** Options

3. Set Textbox *Name* Property: Select the textbox, open its Properties window, and change the *Name* property to "TotalPartChg".

4. Set the *Control Source* Property: Click the ellipsis (**…**) button to open the Expression Builder window. After typing the expression `=[Parts:Subform2].[Form]![TotalCost]`, click **OK**.

Shortcut: You can use the expression builder to generate this name. In the Expression Builder window, select *RepairOrder2* and *Parts: Subform2* in the leftmost folder. Then double-click the *TotalCost* field in the middle folder. The value `<Expr> [Parts: Subform2].Form![TotalCost]` appears in the text window. Replace `<expr>` with the = sign and close the Expression Builder window.

5. Set Other Properties According to Table 7.

Table 7: *TotalPartsChg* Textbox Properties and Settings

Property	**Setting**
Format	"Currency"
Enabled	"No"
Locked	"Yes"
Width	"1.0"
Back Color	The "Lightest Gray"
Special Effect	"Sunken"

6. Enclose Label and Textbox in a Rectangle: Select the **Rectangle** tool from the toolbox and enclose the label and the textbox.
    1. Set *Back Color*: Right-click the rectangle, select **Back Color**, and choose the "Lightest Gray" color. Click **Position → Send to Back**.
    1. Set *Border Style* property: Double-click on the rectangle to open its Property Sheet. Change the *Border Style* property to "Transparent".
7. Toggle to Form View (Figure 30): If you are satisfied, close and save changes.

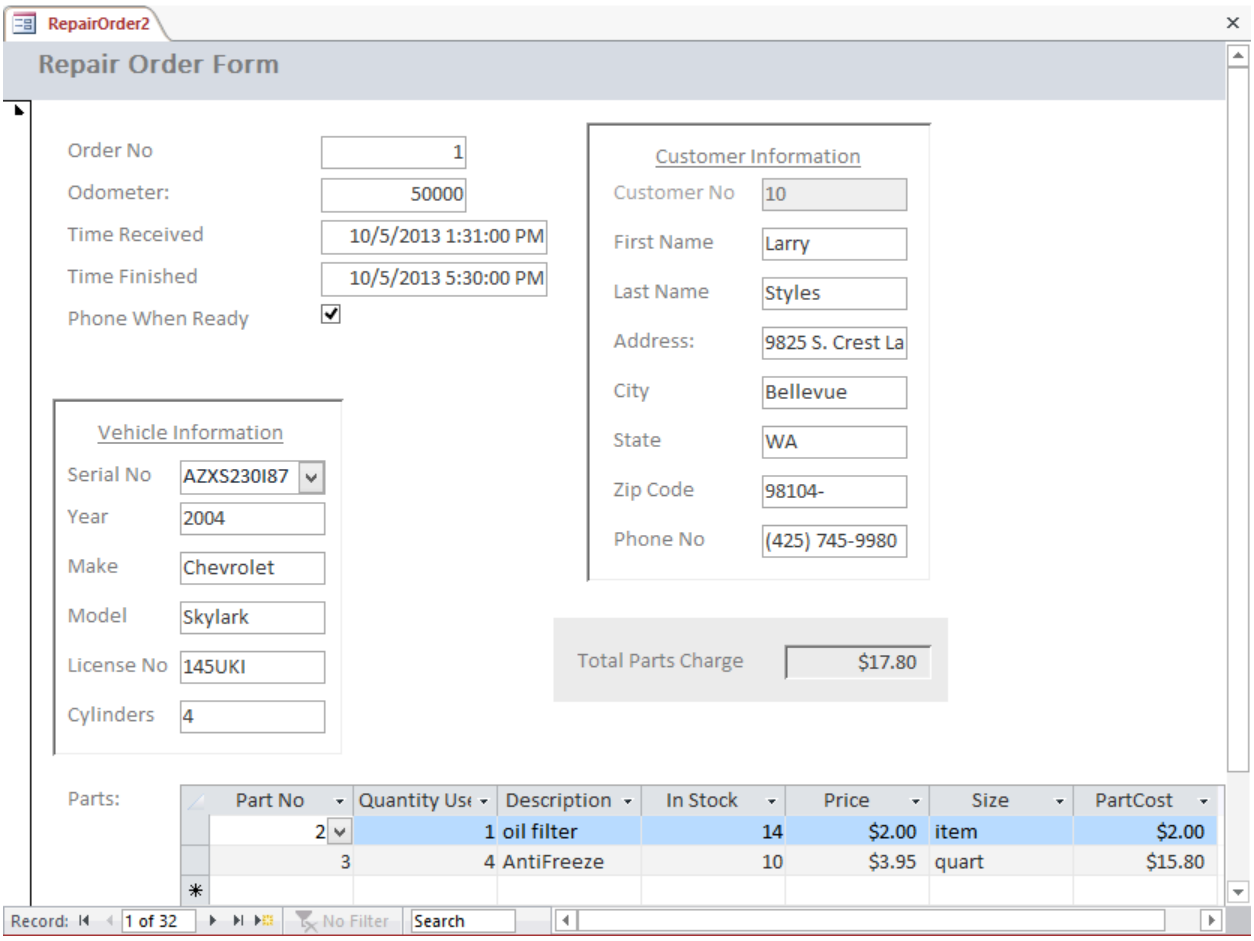


Figure 30: Final Version of *RepairOrder2*

5.6 Advanced Form Features

Before leaving the form chapters, there are four specialized but important concepts left to discuss: the Subform Wizard, the tab control, conditional formatting, and object dependencies. The Subform Wizard allows you to add a subform to a main form by dragging a subform control from the toolbox to the form. The tab control allows you to create tabbed forms within a form, to improve management of scarce screen space. Conditional formatting allows you to make specialized formats that depend on the value of a form field. Object dependencies show objects (tables, queries, forms, and reports) related to a specific object to facilitate management of changes. To learn about the first three features, you will add a subform to the *Customer* form, a tab control to the *Vehicle* form, and conditional formatting to the *Odometer* field of the *Vehicle* form.

5.6.1 Adding a Subform Using the Subform Wizard

You will be adding a subform to the *Customer* table to display the customer's vehicle information in the *Vehicle* table datasheet. To begin, have the *Customer* form open in Design view:

1. Open the Subform Wizard: On the ribbon, click **Design → Use Control Wizards** in the Conrols group to activate a control wizard that will open when the appropriate control tool is selected. Note that if it is highlighted, it is already activated. Next, click the **Design → Subform/Subreport** command

 in the ribbon. Drag it from the bottom of the phone number field down about one inch and four-grid-squares wide to provide a subform shape as shown in Figure 31. When you release the mouse, the Subform Wizard will open to the first window (Figure 32). In this window choose "Use existing Tables and Queries" and click **Next**.

2. Choose Fields for Subform: In the Tables/Queries area, choose "Table:Vehicle". From the left side (Available Fields) below, choose *CustNo*, *Year*, *Make*, and *Model* using the **>** button to move these fields to the right side (Selected Fields), as shown in Figure 33. Click **Next >** to continue.

3. Define Fields to Link Main Form and Subform: Select the first choice from the list (Figure 34) and click **Next**.

4. Name the Subform: In the area provided, type the name "Vehicle: Subform" and click **Finish**. The form will open in Design view (Figure 35).

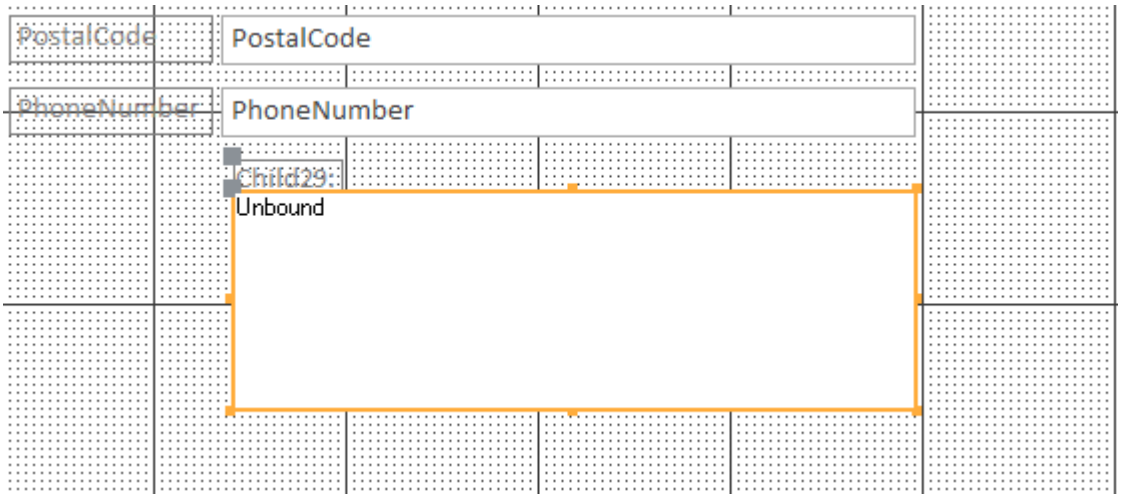


Figure 31: New Unbound Subform

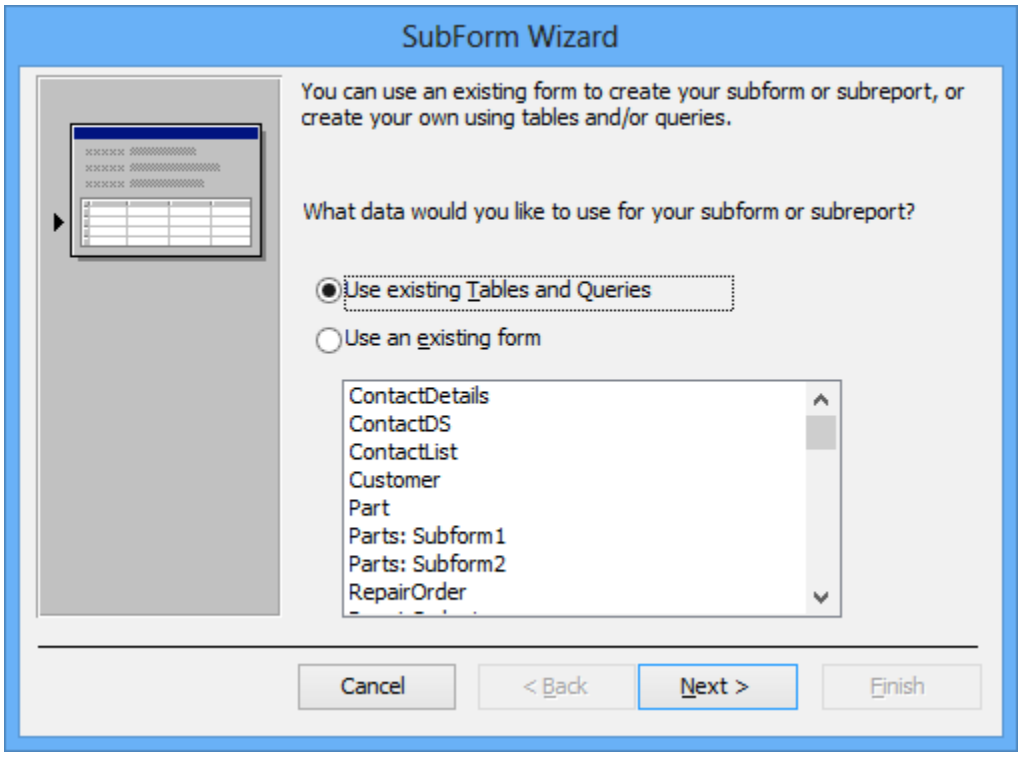


Figure 32: Initial Subform Wizard Window

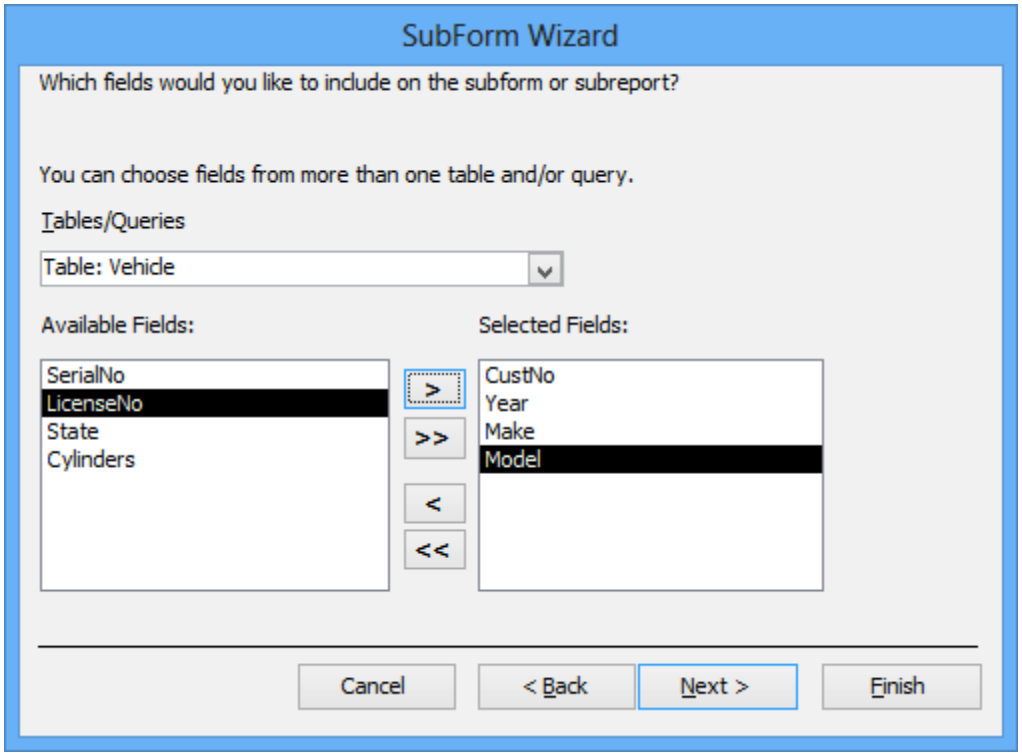


Figure 33: Choose Fields to Include on Subform

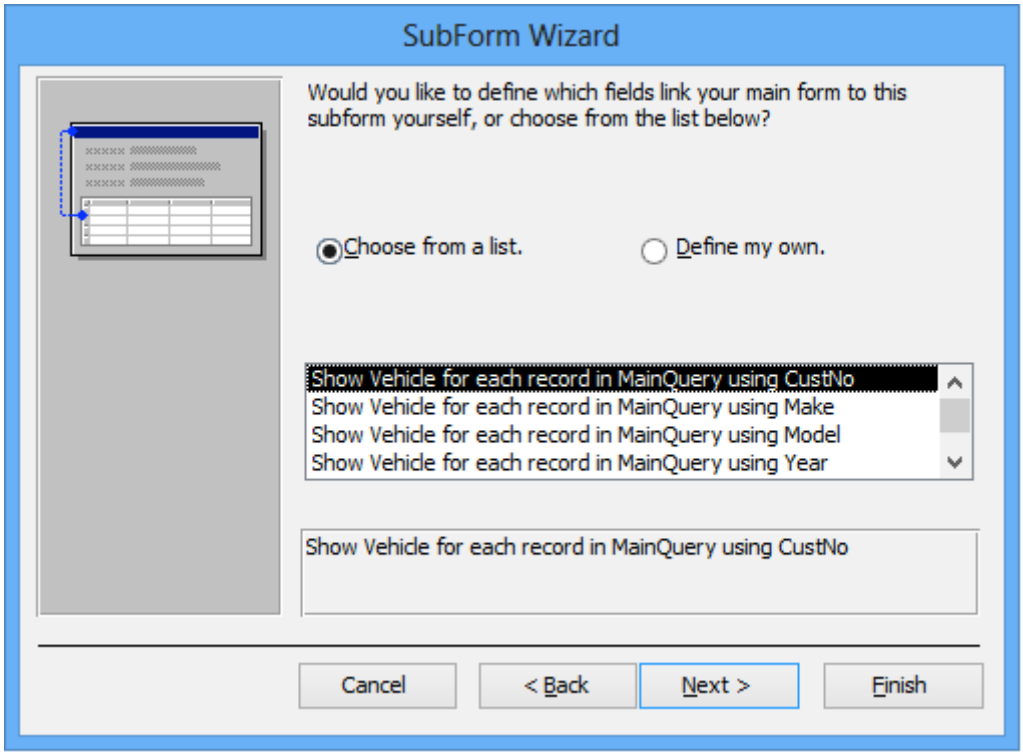


Figure 34: Define Fields to Link Main Form and Subform

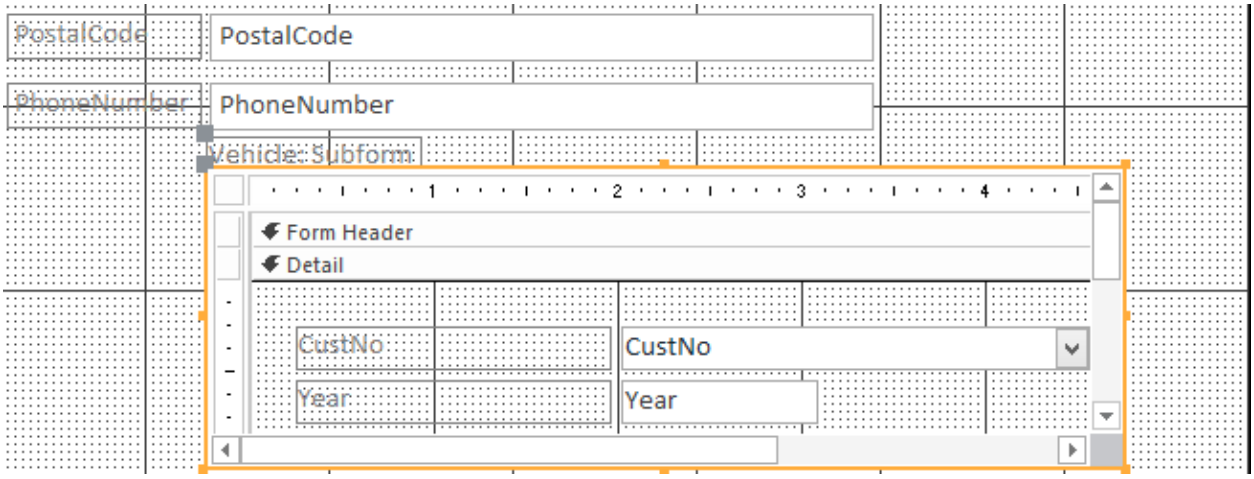


Figure 35: Design View of the *Vehicle* Subform

5. Modify the Subform: Toggle to Form view and you will modify the size of the *CustNo* column to "Best Fit". Select the *CustNo* column and right-mouse-click to open the shortcut menu (Figure 36). Select **Column Width…** and choose "Best Fit" from the dialog box. Close and answer "Yes" to save the form when prompted.

   - Modify the Subform Width: Open the Customer form in Design view. Select the subform and click on the handle on the right side of the rectangle (Figure 35). When the arrow appears, drag the subform to the left, decreasing its width approximately one grid division. Although it will be difficult to notice in design view, this action will decrease the background of the subform in form view from step 5 above. Also, change the subform label to "Vehicle Information".

6. View the Completed *Customer* Form: Toggle to Form view and your form should appear as in Figure 37. Close the form and save changes.

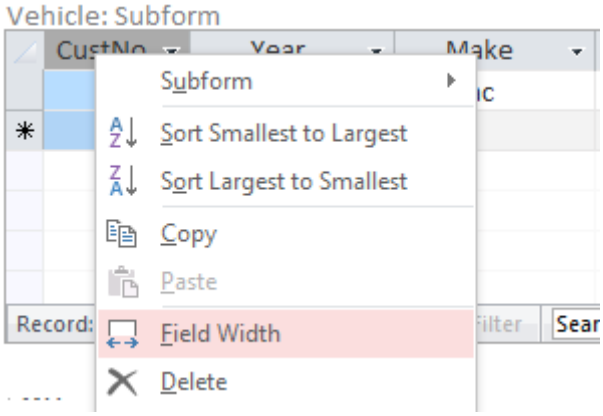


Figure 36: Selected Column and Right Mouse Shortcut Menu

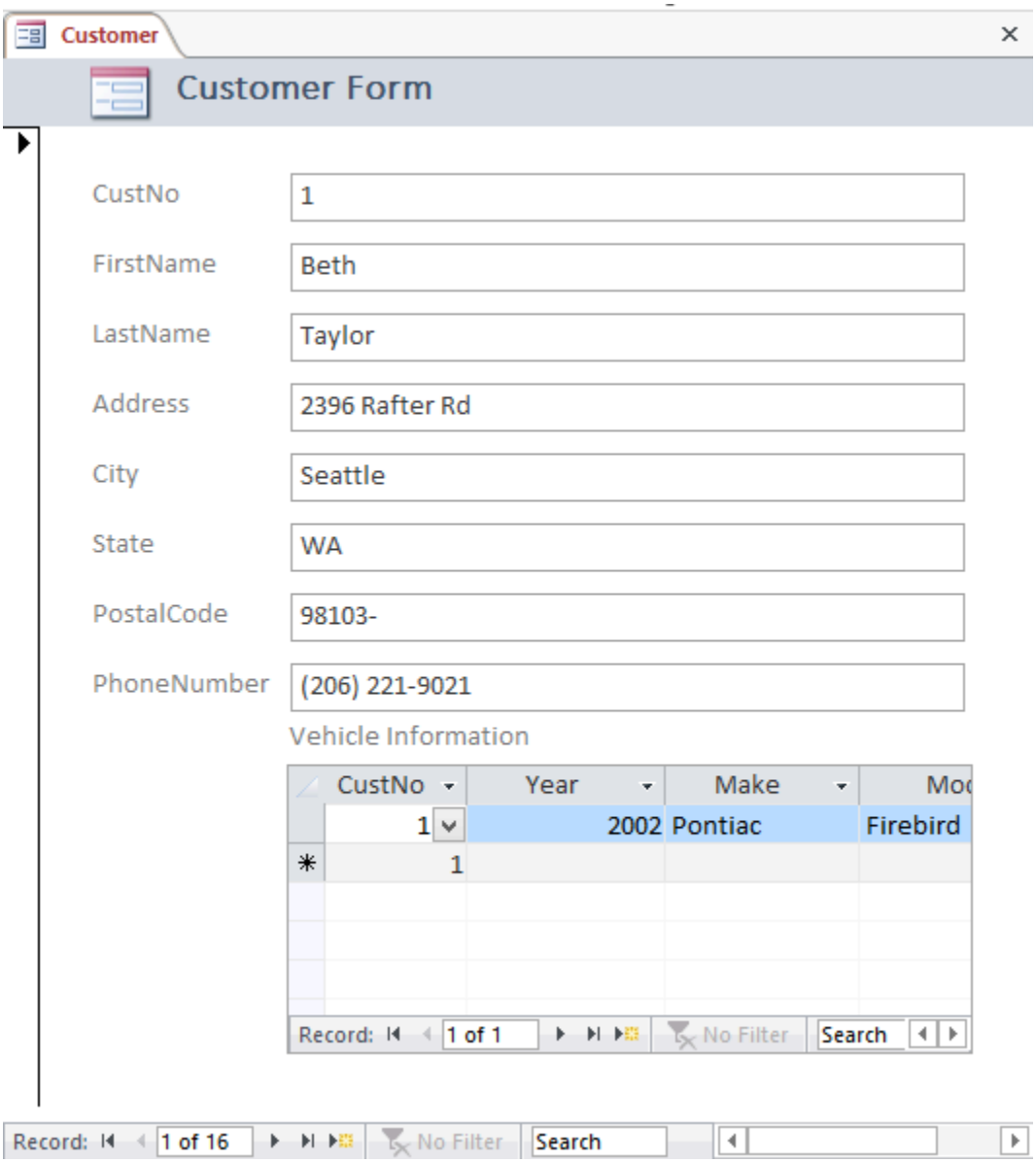


Figure 37: Form View of *Customer* Form with the *Vehicle* Subform

5.6.2 Adding a Tab Control

You will be adding a tab control to the *Vehicle* form to divide the vehicle information into two tabbed sections. To begin, open the *Vehicle* form in Design view.

1. Create a Tab Control: Select the **Tab Control** tool  from the Controls group of the **Design** ribbon and drag it to the right of the *Vehicle* form fields (Figure 38). Disregard the page numbers on the tabs since you will be modifying them later.

2. View the Tab Control Property Sheets: There are several Property Sheets for setting the tab control properties: one for each page in the tab control and then a window for the overall tab control. Therefore, you must pay close attention to the Property Sheet heading to be sure you are in the correct object. Practice selecting each Tab Control Property Sheet.

a. Name the Tab Control: To select the Property Sheet of the overall tab control, click to the right side on the second page. Change the *Name* property from its default name to "Vehicle Tabs" as shown in Figure 39.

b. Name the Two Tab Pages: Click on the first page to select its Property Sheet. Change the *Name* property to "Vehicle Information". Click on the second tab and name it "License Information". Your tab control should now appear as in Figure 40.

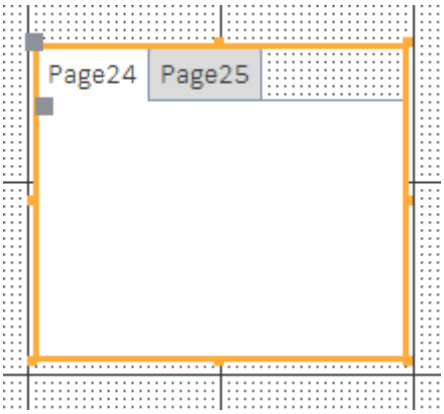


Figure 38: New Tab Control Created on the *Vehicle* Form

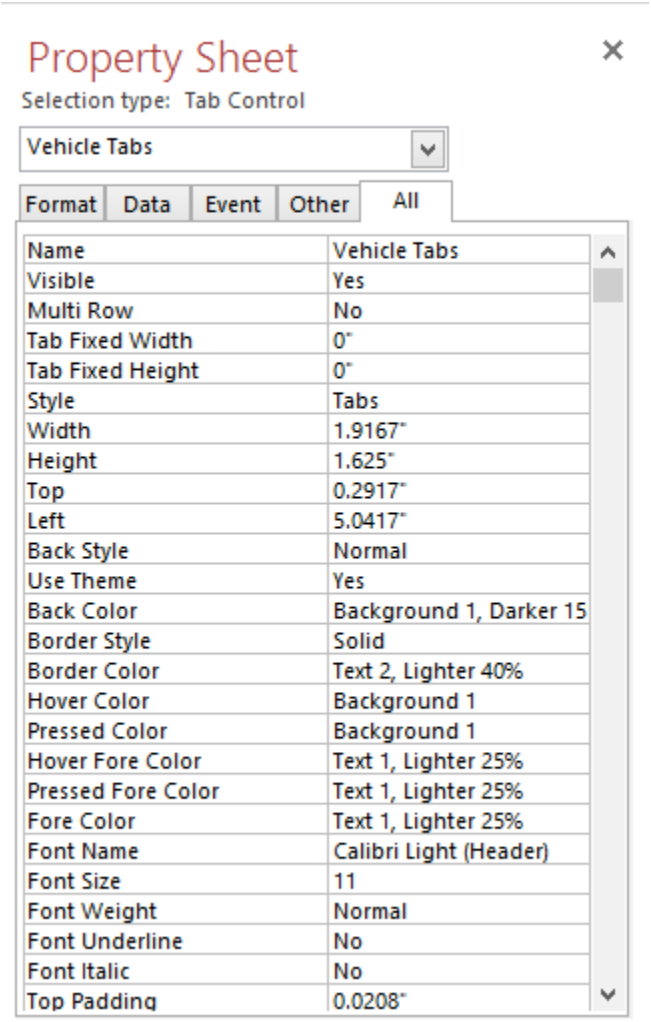


Figure 39: Tab Control Properties Window

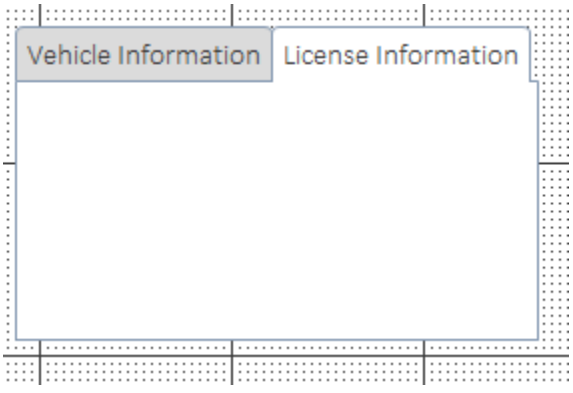


Figure 40: Tab Control after Property Changes

3. Move *Vehicle* Fields into the Tab Pages: To place existing form fields into the pages of the tab control, you must cut and paste since dragging does not always work.

    1. Move Fields into the Vehicle Information Page: Open the *Vehicle Information* tab to move fields into it. To move fields as a unit, hold down the **Shift** key and select the following fields from the *Vehicle* table (you may have to select both the field's label and textbox): *Year*, *Make*, *Model,* and

Cylinders. After they are selected, click **Home → Cut** in the Clipboard group (or right-mouse-click). Be sure the *Vehicle Information* page is selected (click on the page) and click **Home → Paste** (or right-mouse-click). Finally, drag the *Cylinders* label and textbox to align underneath the *Model* field to appear as in Figure 41.

2. Move Fields into the License Information Page: Using the same procedure as moving fields into the *Vehicle Information* page, move the *LicenseNo* and the *State* fields into the *License Information* page (Figure 42).

- Align Tab and Fields in the Form: Drag the tab control and remaining fields to appear as in Figure 43.
- Rename Fields: Rename fields to the following names: "Serial Number", "Customer Number", "License Number", and "State". When you are finished, the completed *Vehicle* form should appear as in Figures 43 and 44.

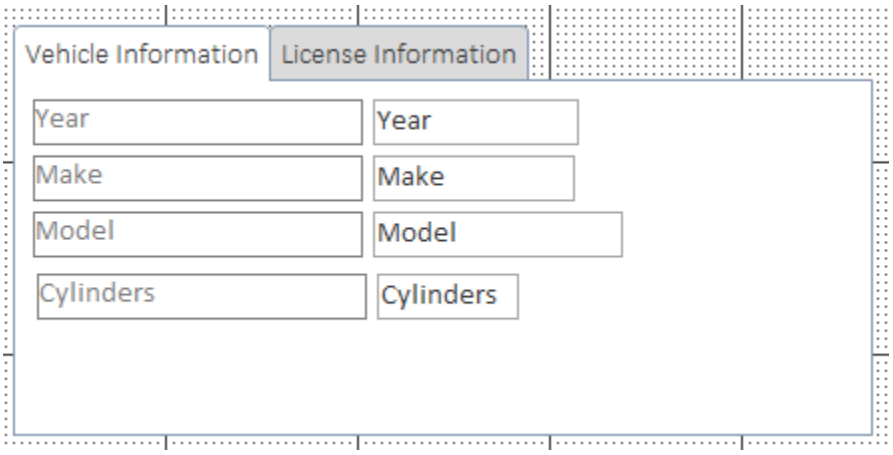


Figure 41: Fields Pasted into the Vehicle Information Tab Page

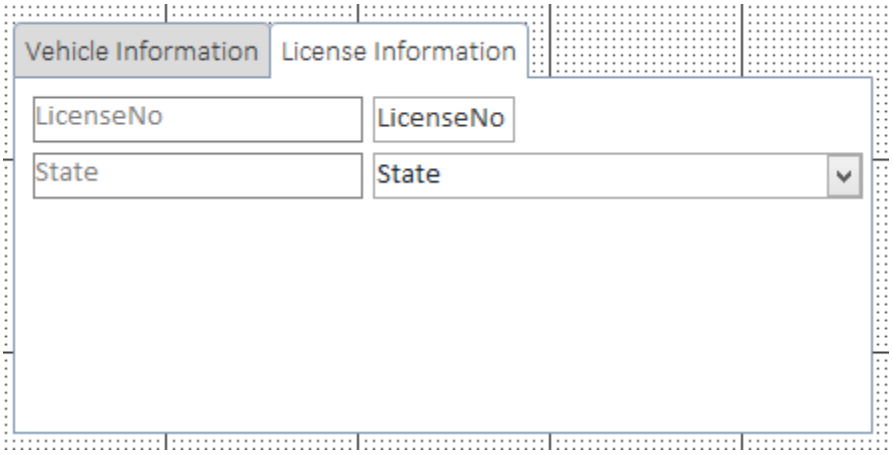


Figure 42: Fields Pasted into the License Information Page

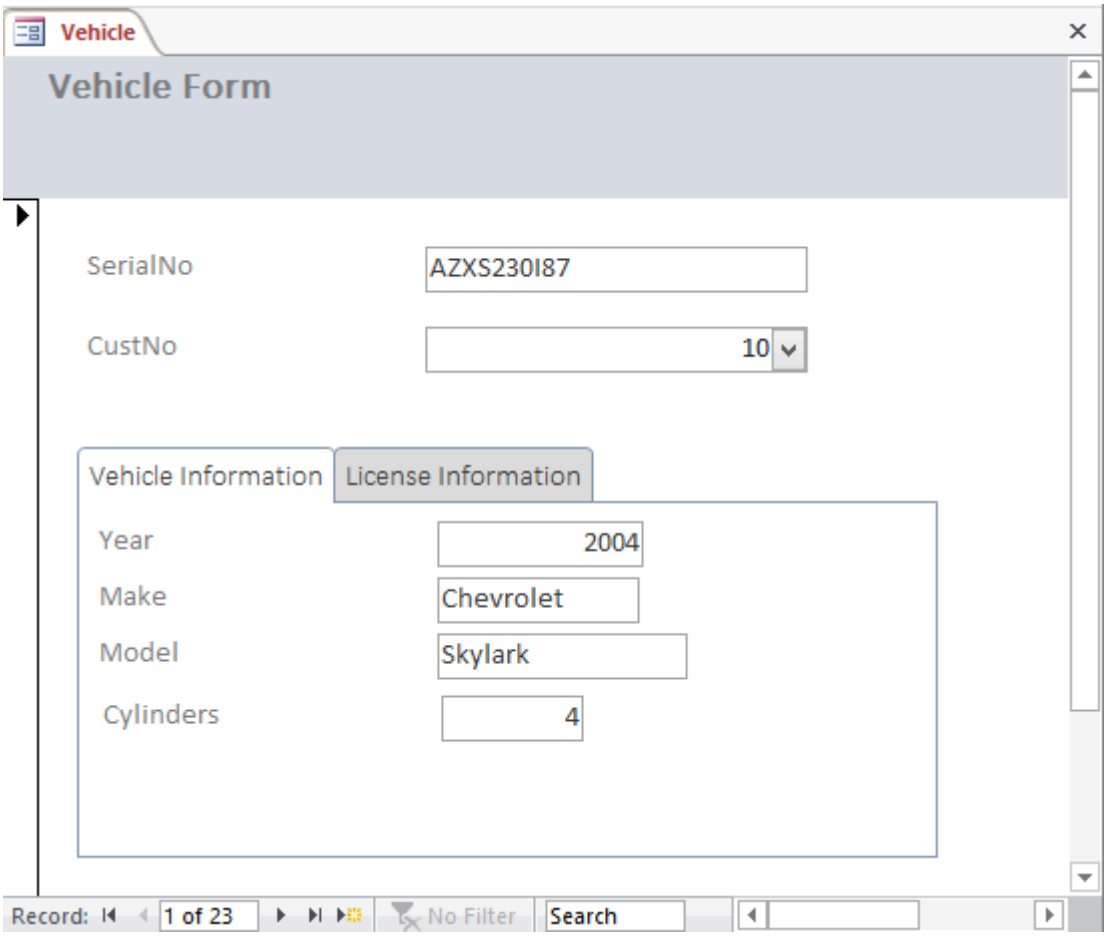


Figure 43: Completed Form Showing Vehicle Information Page

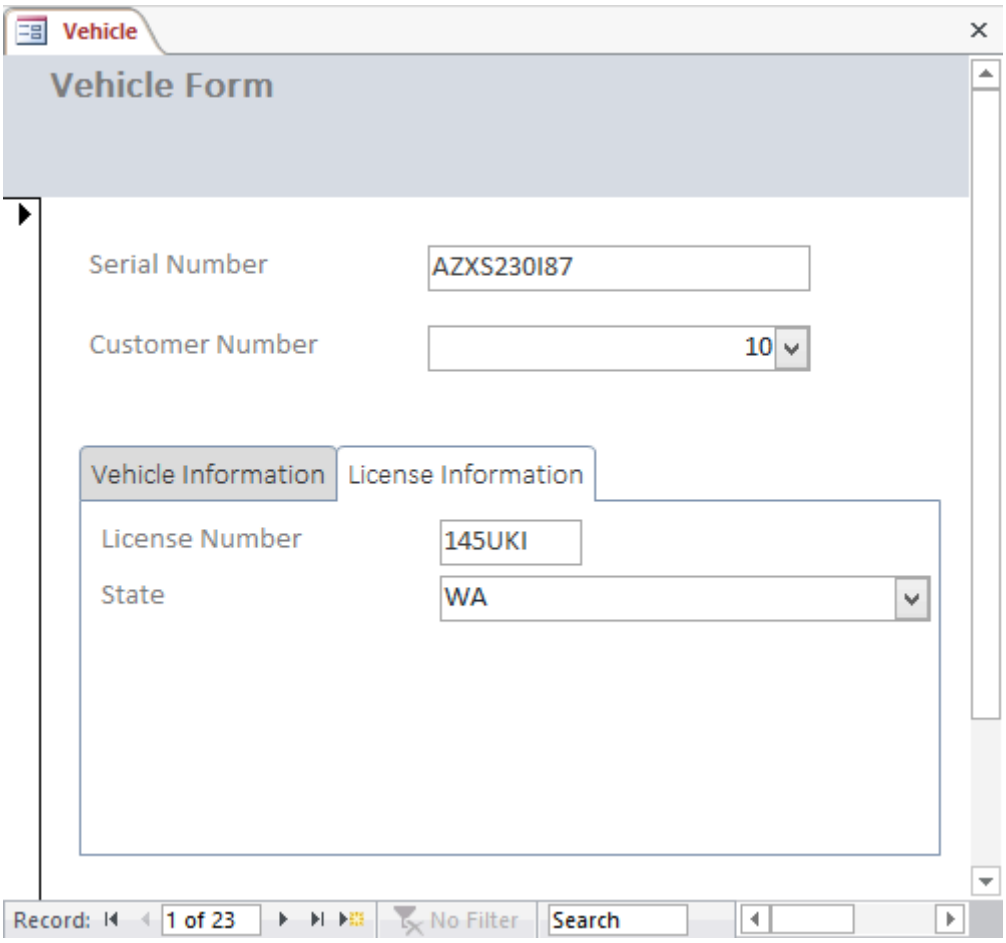


Figure 44: Completed Form Showing License Information Page

- Add a Customer Information Tab Page: To add another tab page, right-mouse-click to the right of the second tab page and choose **Insert Page** from the shortcut menu. A new page is inserted (Figure 45). Name the new page "Customer Information". You may have to drag the tab control wider to accommodate the new tab page.
- Add Fields from the *Customer* Table to the New Tab Page: To accomplish this task, you must modify the *Record Source* query so that it contains fields from both the *Vehicle* and the *Customer* tables.
  - Create the *Vehicle* Form *Record Source* Query: Open the main form's Property Sheet. In the *Record Source* property area, click the ellipsis (**…**) button to invoke the Query Builder. When it opens, you will see the Query Design window with the *Vehicle* table in the upper window. To add the *Customer* table, click **Design → Show Table** in the Query Setup group and add the *Customer* table. Close the Show Table window. Starting with the *Customer* table, drag all of the fields except for *CustNo* and all of the fields from the *Vehicle* table to the empty grid below (Figure 46). Close Query Design in the view pane. Now the *Record Source* property contains this query (Figure 47).
  - Drag Fields onto the Tab: To begin, open and select the *Customer Information* page. Next, open the Field List window (**Design → Add Existing Fields** in the Tools group) as shown in Figure 47. In the field list, hold down the **Shift** key and select (as a unit) all customer fields except *CustNo*. Drag them toward the middle of the tab page (Figure 47). If the field labels are too close to the textboxes, select and align the labels and the textboxes as you have learned previously. When you are finished, your *Vehicle* form should appear as in Figure 48.

- Modify the *State* Field in the License Information Page: Because you added the *Customer.State* field in the *Record Source* for the form, there is an ambiguity in the *Control Source* property for the *State* field in the *License Information* page. In the *Control Source* for the *State* form field, select "Vehicle.State" as the value to eliminate the ambiguity. You can close and save the changes to the *Vehicle* form after you finish.

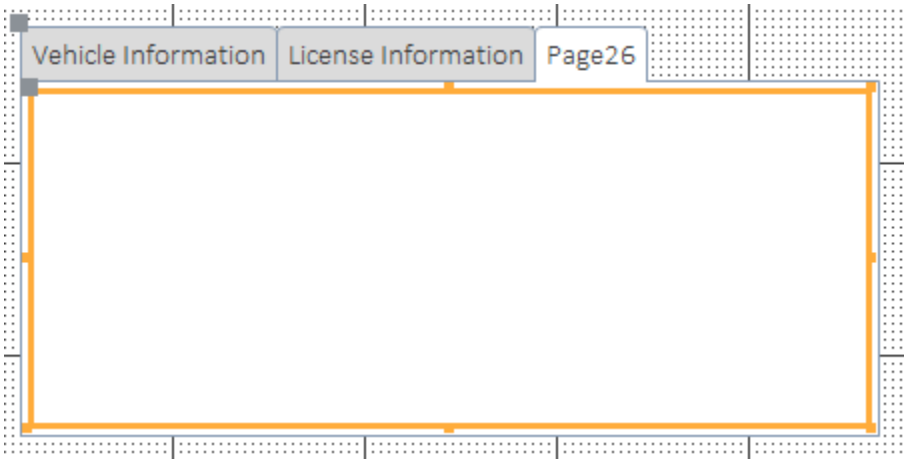


Figure 45: New Tab Page

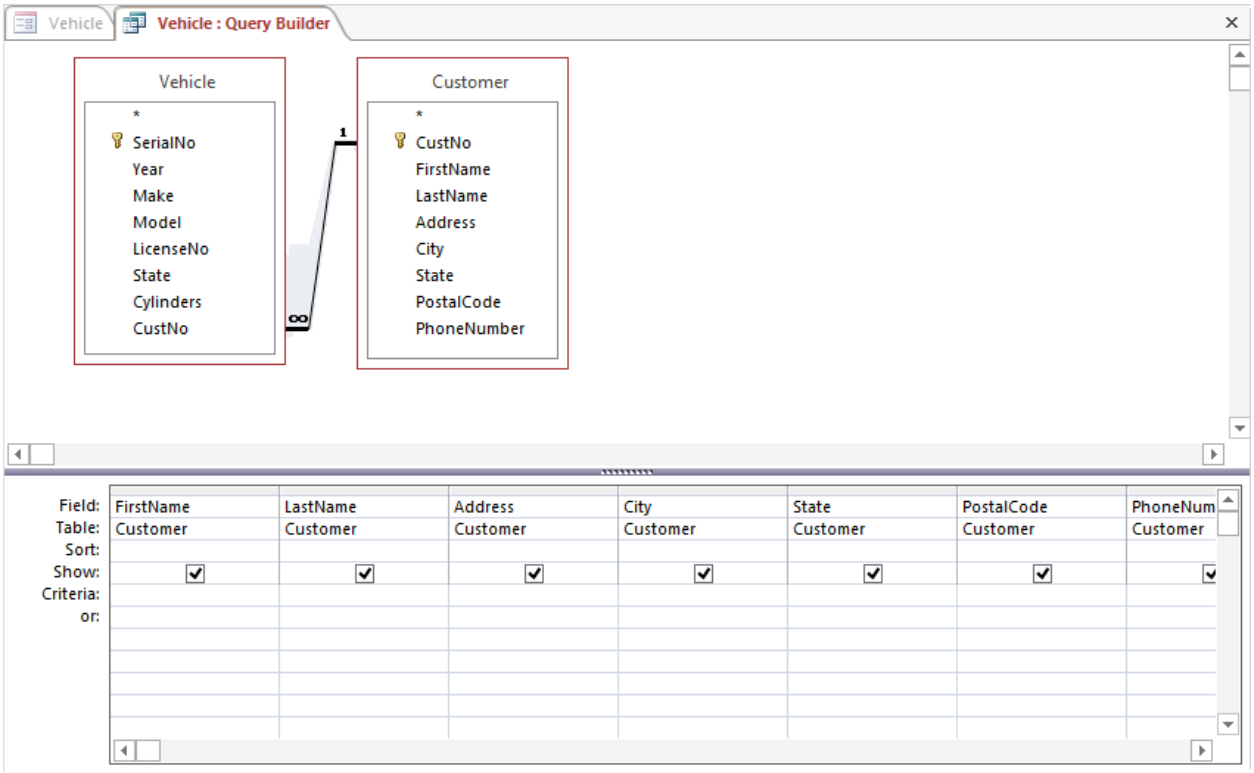


Figure 46: Query for the *Record Source* Property

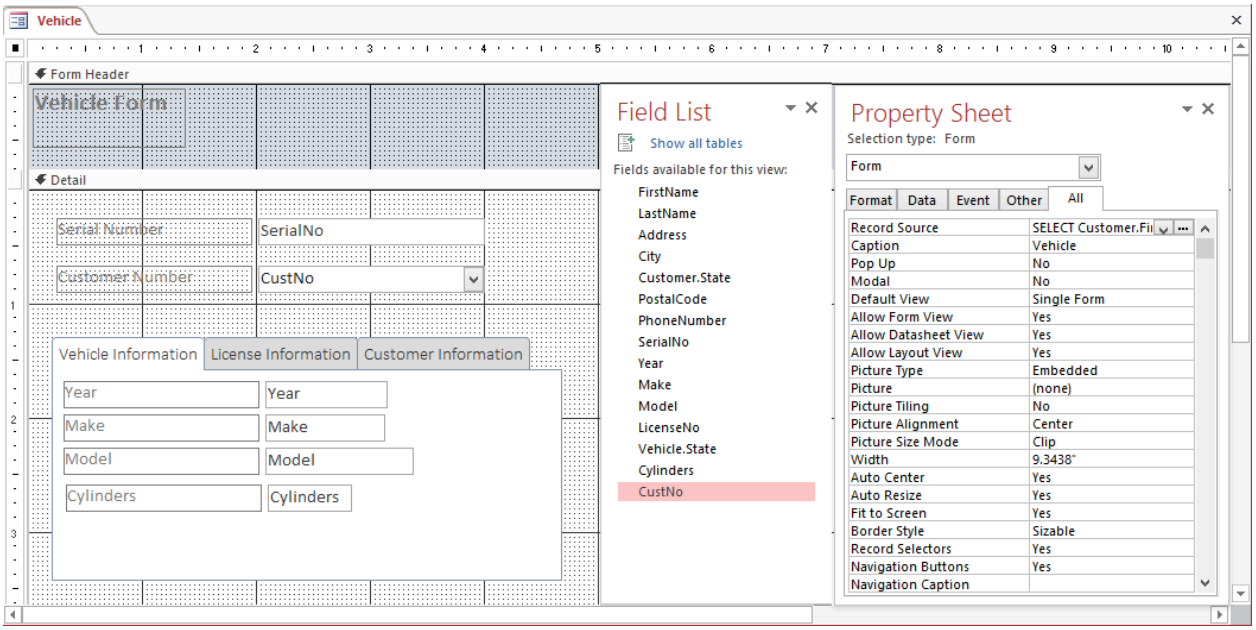


Figure 47: Fields in the *Customer Information* Tab, Form Property Sheet, and *Customer* Table Field List

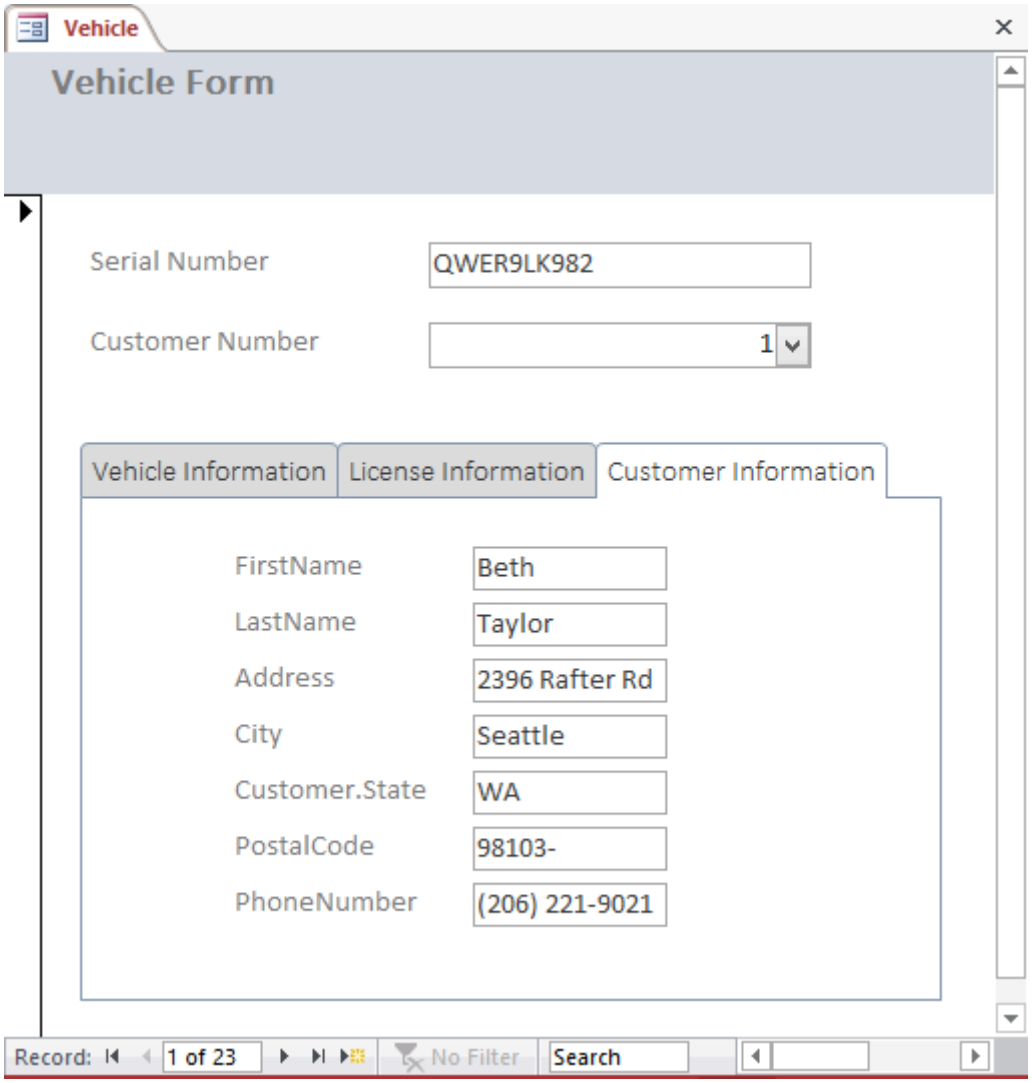


Figure 48: Completed *Vehicle* Form

5.6.3 Using Conditional Formatting

An important feature first introduced in Access 2003 for both forms and reports is called "conditional formatting." This feature, located in the Control Formatting group of the **Format** tab, allows the format of a textbox based on the result of a condition. To use this feature, you will set a conditional format for the *Odometer* field based on a high odometer reading to alert the repair person.

1. Open the *RepairOrder2* form in design view and select the *Odometer* textbox. Click **Format → Conditional Formatting** in the Control Formatting group (Figure 49) and the Conditional Formatting window opens (Figure 50). Click the **New Rule** button to create a new formatting rule (Figure 51).

2. In the bottom part of the window, make a condition as shown in Figure 52. The select the font color of red and bold formatting as shown in Figure 52. The condition in Figure 52 can be interpreted as "when the odometer reading is greater than or equal to 100,000 (miles), then highlight the value in bold font with red color text."

3. Click **OK** to see the formatting rule in the Conditional Formatting Rules Manager window (Figure 53).

4. Figure 54 shows the conditional formatting when the odometer reading is 125,000 miles.

5. In the Conditional Formatting window, you can define more than one condition by clicking the **Add >>** button on the bottom.

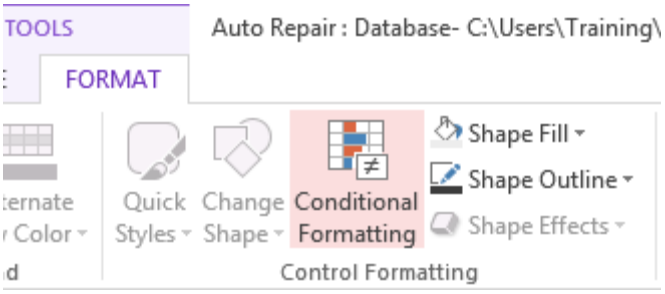


Figure 49: Conditional Button in the Font Group of the Ribbon

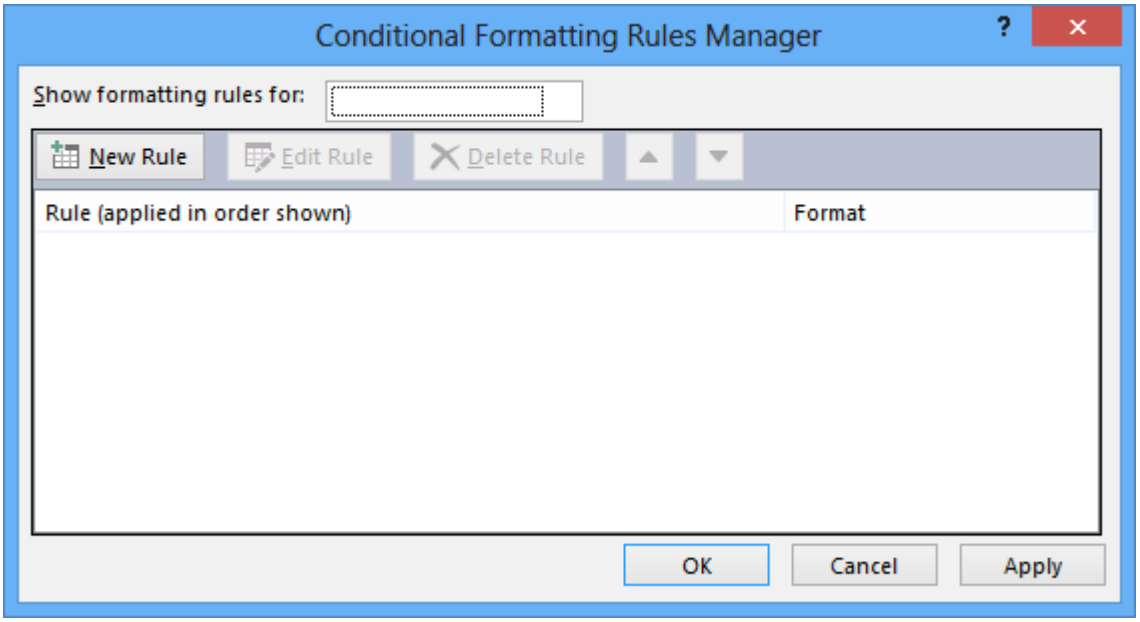


Figure 50: Conditional Formatting Window

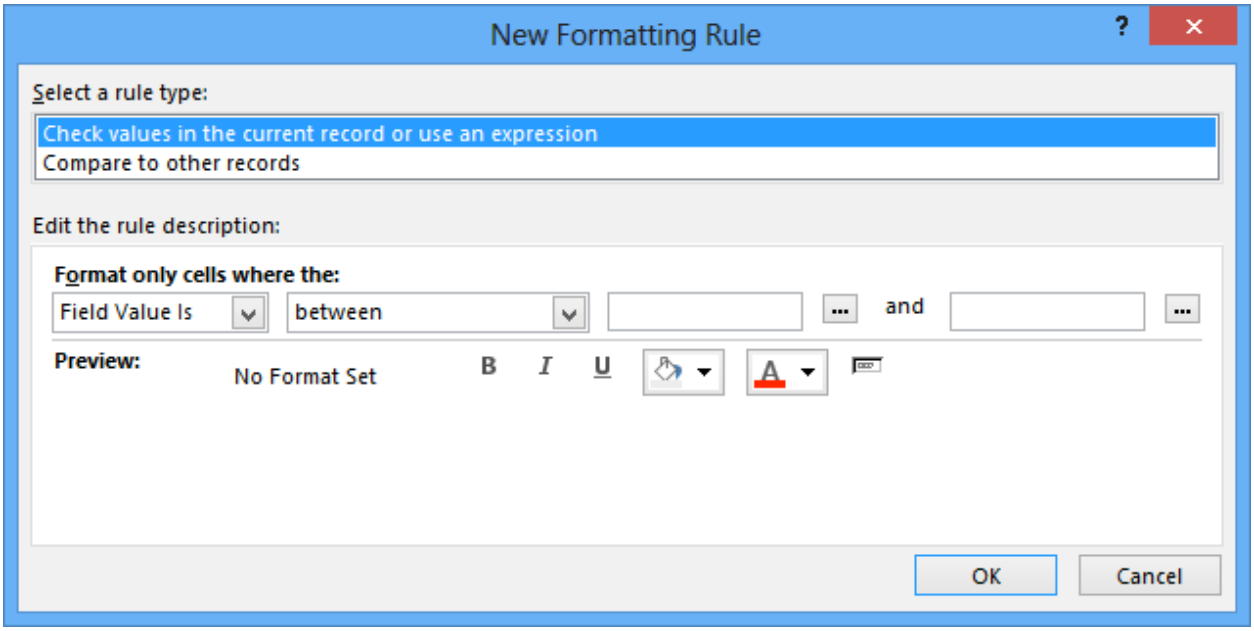


Figure 51: Empty New Formatting Rule Window

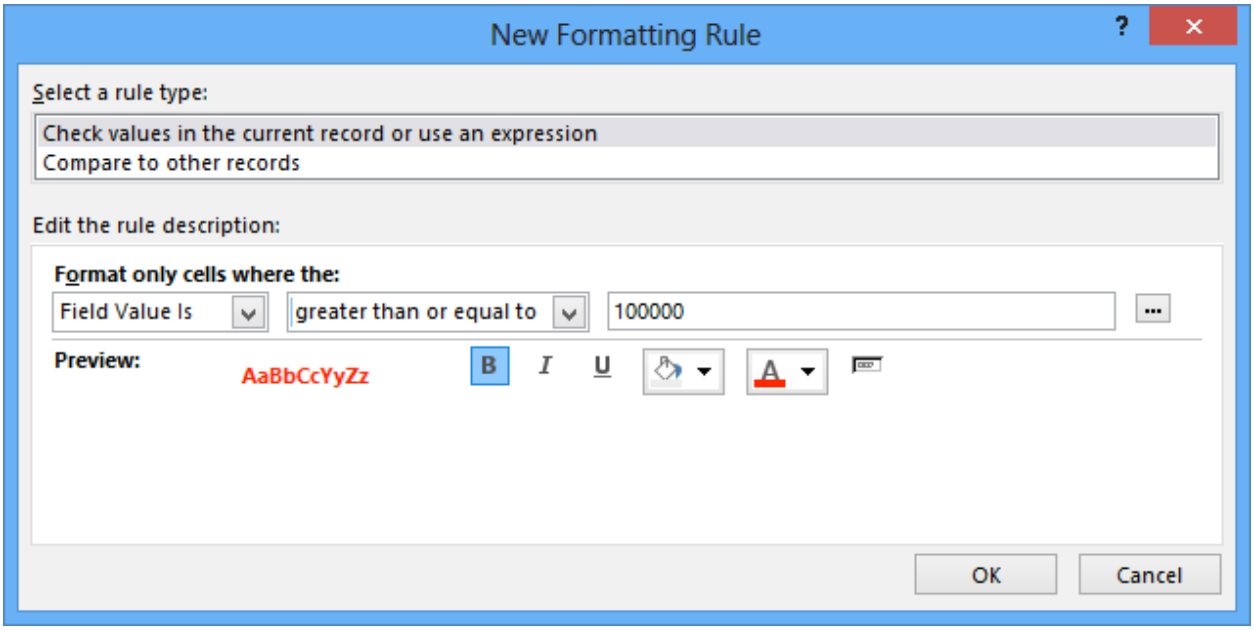


Figure 52: Setting a Condition and Formatting for a Field

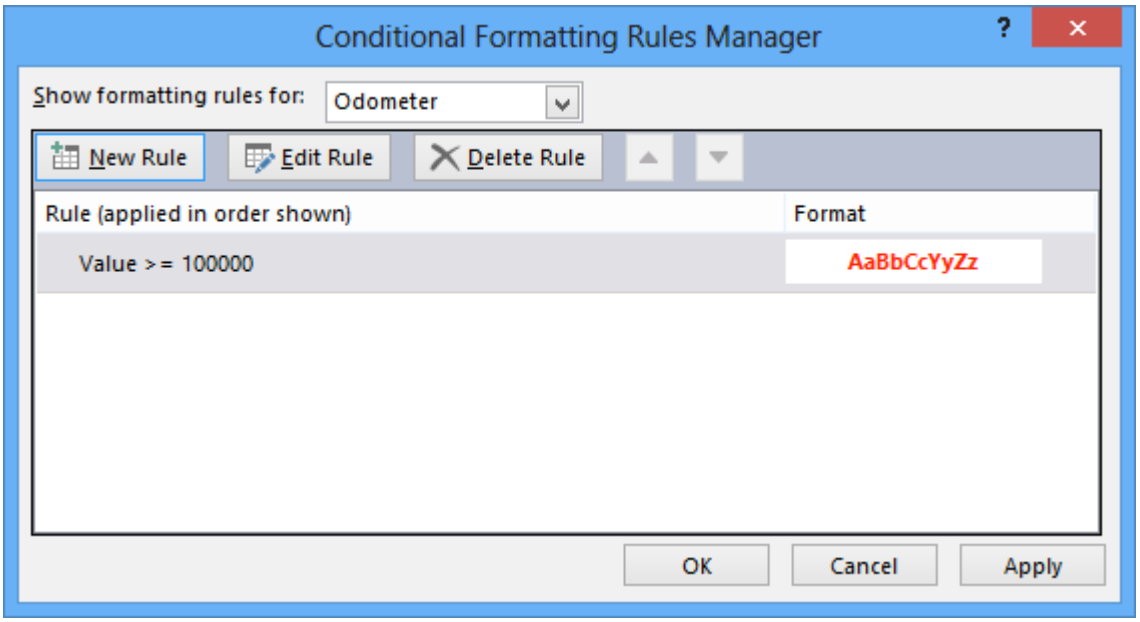


Figure 53: Formatting Rule Displayed in the Conditional Formatting Window

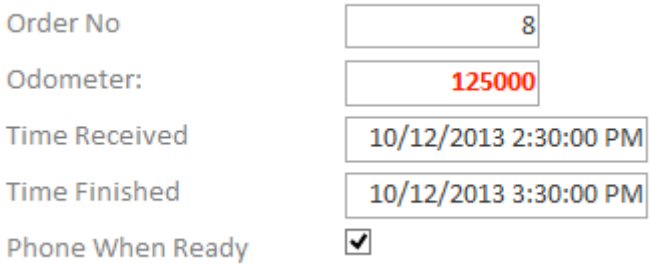


Figure 54: Bold, Red Formatting for a High-Mileage Vehicle

5.6.4 Object Dependencies

Object dependencies, a new feature first introduced in Access 2003, provide a convenient listing of related objects. For example, when working on a hierarchical form, it is convenient to know the tables and queries used in the form as well as other objects that use the form (perhaps as a subform). For a given database object (table, query, form, or report), the object dependency feature shows the objects using the given object and the objects used by the given object. For large databases, knowledge of object dependencies facilitates assessment of change impact. Note that Access does not track some specialized query objects including action queries, SQL-specific queries, and subqueries so dependency information may not be complete.

To activate the object dependency feature, you need to enable the Name AutoCorrect feature using the Options window (**File → Options**). In the Current Database section of the Options window, you should check the "Track name AutoCorrect info" as shown in Figure 55.

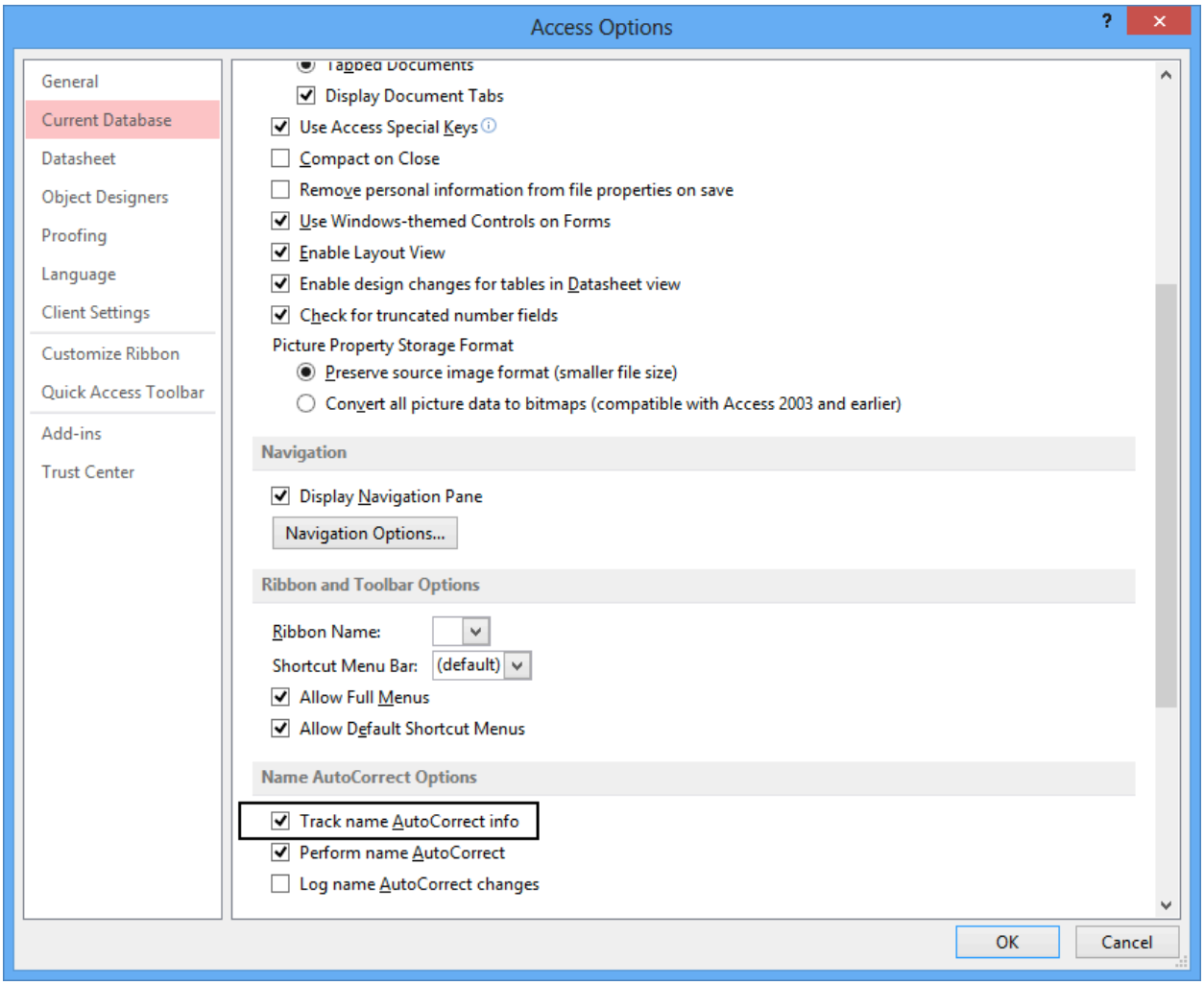


Figure 55: Using the Options Window to Enable Tracking of Object Dependencies

After enabling dependency tracking, you can view the object dependencies for tables, queries, forms, and reports. To view the object dependency of an object, select the object in the navigation pane, click **Database Tools → Object Dependencies …** in the Show/Hide group. The Object Dependencies window appears on the right. Figure 56 shows dependencies for the RepairOrder2 form. The Object Dependencies window provides options to view the objects using the given object and the objects used by the given object. The latter option is selected in Figure 56.

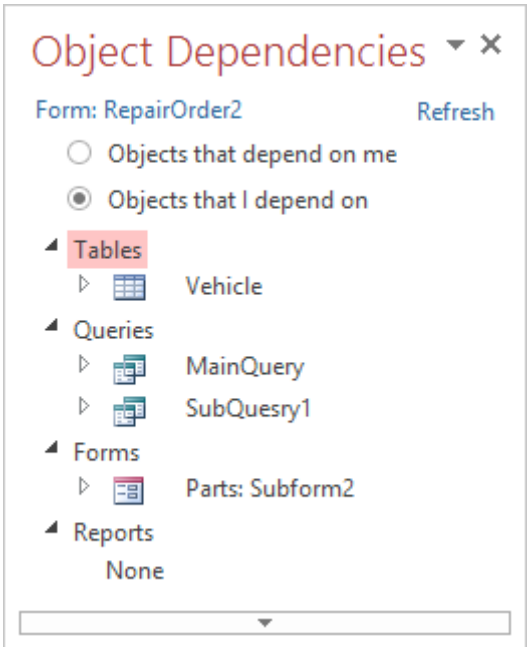


Figure 56: Object Dependencies Pane for the RepairOrder2 Form

Access performs some dependency management through field property propagation, a new feature first introduced in Access 2003. When you create a bound control in a form or report by dragging a field from the field list, Microsoft Access copies certain properties (*Format*, *DecimalPlaces*, *InputMask*, *ValidationRule*, *ValidationText*, and *DefaultValue*) from the field in the underlying table or query to the control. In previous Access versions, the designer had to manually make the changes. For example, if the *TimeRecvd* column in the *RepairOrder* table has the *Format* property set to short time, when you create a bound text box in a form, Access automatically sets the *Format* property for the text box control to the same value. Thus, it is usually preferable to set properties in the underlying column rather than in a control to ensure consistent settings.

Closing Thoughts

After completing this chapter, you should have a good understanding of hierarchical forms in Access. You have learned to create a simple hierarchical form using the Form Wizard and to extend it with multiple table queries, AutoLookup queries, combo boxes, and computations shared between a main form and a subform. You also have learned other advanced form development features including the Subform Wizard, tab controls, conditional formatting, and object dependencies. In addition to the guided instruction in the chapter body, the appendices provide valuable reference material to help you develop your own forms.

You should have an appreciation for the complexity of hierarchical forms and the powerful and easy-to-use tools provided in Access. In many database management systems, developing hierarchical forms requires detailed coding. For example, you would need to write several hundred lines to code the final repair order form (Figure 30) in Visual Basic. In Access, coding is necessary only to add capabilities such as invoking other forms.

Access provides an alternative to a hierarchical form when you want to present your data without using a subform. If you have a form with many controls, you may not have room for a subform. In this case, you can use the Form Wizard to create synchronized forms. When you click a command button on one form, it opens another form that is synchronized with the record on the first form.

Forms can be extended and customized using event-driven modules. For example, you can add a button to a form to open a related form, to requery a combo box when a user obtains focus, and to update fields in related tables. You will learn about event-driven coding in Chapter 8 using both the macro builder and Visual Basic for Applications.

Chapter Reference

The chapter reference summarizes procedures that you practiced. For wizards discussed in the chapter, the procedures highlight important parts of the wizards but do not list all of the steps.

Procedure 1: Using the Form Wizard to Create a Hierarchical Form (Section 5.1.1)

1. Choose **Create → Form Wizard** in the Forms group to open the New Form window and choose the table where indicated.
2. Click the >> or > button to move fields from the left side (Available Fields) to the right side (Selected Fields). The Selected Fields area contains fields that will appear in the main form.
3. In the same window, go up to the Tables/Queries selection box where the previous table name appears and select a different table or query for the subform. Click the >> or > button to move fields to the right side (Selected Fields). Note, the Selected Fields window will contain fields for both the main form and the subform.
4. The next window asks how you want to view the data. Select the table/query that you want to appear in the main form. The option "Form with subform(s)" should be selected.
5. Select the layout of the subform in the next window. Select "Datasheet" if it is not already selected.
6. In the last window, you must provide names for the main form and the subform. In the same window, select the second option, "Modify the form's design", since you will most likely need to make modifications to the form.

Procedure 2: Editing a Subform's Properties (Section 5.2.3)

1. You can edit the properties of a subform to change the linking fields or the form associated with the subform.
2. The Subform Properties window can be opened by selecting the subform. Then, click the right mouse button and select **Properties**. Alternatively, you can click **Design → Property Sheet** in the Tools group.
3. Use the *Width* and the *Height* properties to expand the size of the subform. Alternatively, you can expand the size by selecting the subform and dragging the double-arrow cursor.

Procedure 3: Editing a Subform in Design View (Section 5.2.3)

1. You should open a subform in design view to edit controls in the subform.
2. To open a subform in design view, make sure that the subform is not selected before opening it. If the subform displays as a white area, you can double click on it to open the subform in Design view or right click inside the subform and select **Subform in New Window**. If the subform is not displayed as a white area, it is already open in Design view.
3. To change the field names that appear in the subform's datasheet, change the *Caption* property of the label associated with each textbox.
4. To change column widths, toggle the main form to form view with the subform displayed in datasheet view. To adjust the widths of individual columns, select a column in the subform, right-click to display a popup menu, and select the **Field Width** item. You also can set a column to a standard fit (using the check box) or best fit using the button in the Column Width window.

Procedure 4: Making a Textbox Read-Only (Section 5.4.3)

1. Making a textbox read-only prevents focus from being established. Without establishing focus, a user cannot edit the value in a textbox. Textboxes that receive values through AutoLookup queries are usually read-only.
2. Double-click the textbox to open its Property Sheet. Change the *Enabled* property to "No" and the *Locked* property to "Yes".

Procedure 5: Restricting the Values in a Combo Box (Section 5.4.2)

1. You may want to restrict the values in a combo box list to make the values consistent with other values displayed on a form. Restricting the values can help a user quickly find a value to select.
2. To restrict the values, edit the *Row Source* query value. Add a condition to a field in the query to match the value displayed on a form. See Appendix B for the notation about referring to form fields in a query.

Procedure 6: Sharing Computations between Forms (Section 5.5)

106. Sometimes a main form needs to display values computed in a subform. Typically, the main form should display a summary total computed using all of the records in a subform. To display a summary total, you need to change both the subform and the main form.

1. First, you need to add a textbox in the footer section of the subform. The *Control Source* property of this textbox should perform the summary calculation. You should not perform the summary calculation in the underlying query (*Record Source*) of the subform.
2. Then, you need to add a textbox in the main form. The new textbox should reference the textbox in the footer section of the subform. See Appendix B for the notation about referring to a subform control from a control in the main form. You also can use the expression builder to generate the proper notation.

Procedure 7: Using the Subform Wizard (Section 5.6.1)

- Select the **Use Control Wizards** button on the ribbon. Next, select the **Subform** tool from the toolbox and drag it to create the subform. When you release the mouse, the Subform Wizard will open to its first window. In this window, choose "Use existing Tables and Queries" and click **Next >**.
- In the Tables/Queries area, choose the table or query for the subform fields. From the left side (Available Fields) below, choose the desired fields using the **>** button and click **Next >**.
- In the next window, you select the linking fields. If there is a 1-M relationship between tables in the main form and the subform, Access will detect the linking fields. In this case, you select the first choice, "Choose from a list", to define fields to link the main form and the subform. Otherwise, you may need to define the linking fields. Click **Next >** to continue.
- Name the subform in the area provided and click **Finish**. The form will open in design view with the main form.

Procedure 8: Adding a Tab Control (Section 5.6.2)

1. Select the **Tab Control** tool from the ribbon and drag it to the form.

- The Property Sheet of the tab control depends on the selection. Each tab page has its own Properties window and the entire tab control has its own Properties window.

  1. Name the tab control: To access the Property Sheet for the Tab Control, click to the right side of the second page. Change the value of the *Name* property from its default value.

  - Name an individual tab page: Click on the first tab to access its Property Sheet. Change the *Name* property from its default value for each tab page.

3. The default color for the tab control is gray. To set it for transparent (if desired), open the Property Sheet for the Tab Control and change the *Back Style* property to "Transparent".
4. To insert or delete a tab page, click to the right side of the last page and right-mouse-click to open the shortcut window containing choices to insert page, delete page, and so on.
5. To place main form fields into the tab control pages, you must cut and paste since dragging does not often work.

Additional Practice

The following problems provide additional practice with the extended auto repair database as well as the textbook databases.

Part 1: Hierarchical Forms for the Extended Auto Repair Database

1. Using the Form Wizard, build a new hierarchical form similar to the Repair Order Form in this chapter except that the subform contains details about labor usage, not part usage. Here are some additional details to accommodate:

   - The subform should contain the *LaborCode*, the *LabDesc*, the *Hours*, and the *HourRate* columns. In addition there should be a computed column for the labor cost (*HourRate* * *Hours*).
   - Write the subform query so that the *RepairLabor* table is updatable. Follow the rules for 1-M updatable queries presented in textbook Chapter 10 (Section 10.3).
   - The subform fields for labor description, hourly rate, and labor cost should be read-only.
   - The subform footer should contain a summary calculation for the sum of the labor cost. The main form should display the sum of the labor cost, the sales tax, and the total amount.

2. Build the same hierarchical form in (1) except that you should use design view to revise the original Repair Order Form. Save the new main and subform under new names.
3. Using design view, add another sub form to the form built in (2). The second subform should display the part charges in a similar manner to the labor usage. The second subform is independent of (not embedded in) the labor usage subform. Here are some additional details to accommodate:

   - You can use the drag-and-drop method to add the second subform. Make a copy of the main form from (1) or (2). Then, drag the *Part* subform from the Database window and drop it into the design

view of the new form. You might want to expand the length of the detail section before performing the drag-and-drop operation.

- Add a total field for the sum of the part charges and revise the tax and the amount due calculations to include part charges and labor charges.

Part 2: Hierarchical Forms for the University Database

1. Use the Form Wizard to build the Course Offering Form described in textbook Chapter 10 (Section 10.4).
2. Use the Form Wizard to build the Faculty Assignment Form described in textbook Chapter 10 (Section 10.4).
3. Use the Form Wizard to build the Registration Form described in Chapter 5 (Section 5.4).

Part 3: Hierarchical Forms for the Order Entry Database

1. Using the Form Wizard and the form design view, build the Simple Order Form described in problem (22) of textbook Chapter 10.
2. Using the Form Wizard and the form design view, build the Order Form described in problem (23) of textbook Chapter 10. Define the main form and the subform queries before using the wizard.
3. Modify the Order Form in problem (2) according to the change described in problem (24) of textbook Chapter 10.
4. Using the Form Wizard and the form design view, build the Simple Purchase Form described in problem (25) of textbook Chapter 10.
5. Using the Form Wizard and the form design view, build the Purchase Form described in problem (26) of textbook Chapter 10. Define the main form and the subform queries before using the wizard.
6. Using the Form Wizard and the form design view, build the Supplier Form described in problem (27) of textbook Chapter 10. Define the main form and the subform queries before using the wizard.

Appendix A: Common Errors in Form Design

Appendix A describes problems that students have encountered in past academic terms. You should review these situations carefully when you encounter an error. Most likely, your problem and its resolution are described here. For your convenience, these problems are organized by topic area. The topic areas are listed in order of the most common problems encountered. For simpler errors involving individual controls, the Access error checking feature should suffice as described in Section 4.4.4 of Chapter 4.

- Problems in formulating the underlying queries (*Record Source* property) for the main form and the subform.

  ⇒ The error message "Records in Table _____ would have no records on the 1 side." indicates an error in the underlying query (*Record Source*) of the form. This error often occurs when you change a value with a combo box, but it does not indicate an error with the combo box. Remember that in a 1-M query, it is critical that the foreign key field of the child table be in the query output. For example, in the *RepairOrder2* subform, the *Record Source* involves a join of *Part* (the parent table) and *PartsUsed* (the child table). The field *PartsUsed.PartNo* must be in the query result. If this field is omitted (or *Part.PartNo* is included instead), the resulting query will NOT be updatable. The error message that results is "Records in Table PartsUsed would have no records on the 1 (parent) side."

  ⇒ You must have a column that is common to the *Record Source* for the main form and the subform. In the *RepairOrder2* form, the common column is *OrdNo*. This column appears in both underlying queries. In the query for the subform, this column must appear although there will not be a field in the subform using this column. The common field is used in the *Linking Child Fields* and the *Linking Master Fields* properties.

  ⇒ The queries for the main form and the subform should share a common field but usually do not have a common table. In the *RepairOrder2* form, the common column is *OrdNo*. But there is no table that is used in both queries. The query for the main form contains *RepairOrder*, *Vehicle*, and *Customer*. The query for the subform contains the *PartsUsed* and *Part* tables. Note that if you include the *RepairOrder* table in the subform query, your hierarchical form may not work correctly.

  ⇒ If you do not have a common column, the Form Wizard may not detect that you want a hierarchical form. For example, in the query for the repair order subform, assume that you use *RepairOrder.OrdNo* instead of *PartsUsed.OrdNo*. The Form Wizard will not detect that you want a hierarchical form. This query formulation error often occurs because your subform query has a common table (*RepairOrder*) rather than a common column.

  ⇒ Make sure that all required fields from the updatable tables are included. Usually, there is one table in the main form and one table from the subform that can be changed by using the form. All required fields from these tables must be included in the query output.

  ⇒ Think carefully about outer joins. In some cases they are necessary; in other cases harmful. Usually the only time you need an outer join is to include nonmatched records from a child table.

  ⇒ A query is not updatable if it contains a `GROUP BY` clause or the `DISTINCT` keyword. Access likes to put `GROUP BY` in queries when you use Query Design. Remove the `GROUP BY` when using a query for a main or subform. `GROUP BY` destroys updatability because each row of a `GROUP BY` query can refer to several rows of the base tables. `DISTINCT` destroys updatability because each row of a view may correspond to more than one row in a base table. Note that queries with the `DISTINCTROW` keyword are usually updatable. With `DISTINCTROW`, each row in a query corresponds to one row of a base table.

⇒ Use the join operator style to express joins in the main form and the subform queries. If you use the cross product style (see textbook Chapter 3), Access will think that your query is not updatable. Because Access prefers the join operator style, it is a good idea to use Query Design rather than SQL to formulate the queries for the main form and the subform. Query Design automatically generates the join operator style.

- Errors relating to combo boxes.

  ⇒ Do not confuse queries for combo boxes with the underlying queries for the main and the subform. The combo box queries are independent of the underlying queries. Combo boxes are memory aids as they allow a user to select a value rather than needing to remember the possible values. Therefore, the purpose of a combo box query is to generate a list of values from which the user can select. In contrast, the purpose of a main form or a subform query is to supply the data displayed in the various form fields.

  ⇒ A combo box must be bound (*Control Source* property) to a field in the underlying query. Do not use an equal sign (=) before the name. The = symbol indicates an expression, not a field.

  ⇒ For AutoLookup queries, the *Control Source* should be the foreign key, not the primary key. For example, the *Control Source* for serial number should be *RepairOrder.SerialNo* (the foreign key), not *Vehicle.SerialNo* (the primary key). If your subform query only includes the foreign key, you should not have a problem with setting the *Control Source* to the primary key instead of the foreign key.

  ⇒ A combo box query is specified in the *Row Source* property. For example, the *Row Source* for the *Part No.* combo box is a query on the *Part* table. Usually combo box queries involve just a single table. Do not use the underlying query (*Record Source*) in the *Row Source* property.

  ⇒ You can specify the number of columns to display in the combo box with the *Column Count* property. Usually the first column in the query result is bound to the combo box control. Select "1" for the *Bound Column* property to bind the first column. If you set it to another value (say, "2"), no values will display in the combo box and an error message will be generated when you move to the next record.

  ⇒ The Combo Box Wizard can cause confusion regarding the *Column Widths* property. The Combo Box Wizard recommends that you do not display the bound column (usually the first column). Following this recommendation is fine except that the width of the first column is set to 0. Access does not display the value of the first column if the width is 0. To alleviate this problem, you may need to change the *Column Widths* property after using the Combo Box Wizard. Set the width of the first column to a small value such as 0.01".

- AutoLookup problems.

  ⇒ You should be aware that AutoLookup queries only work for foreign key fields. In particular, you should not expect AutoLookup queries to work for the primary key of the main form. For example, you cannot enter a new value for *OrdNo* in the main form and expect related fields to be displayed. You would need to formulate a search or implement a command button to search on order number.

  ⇒ You can tell whether AutoLookup queries apply to a form field by examining the field's *Control Source* property. If the *Control Source* property is set to a foreign key, AutoLookup applies to the form field. For example, if the *Control Source* is *Vehicle.SerialNo*, AutoLookup does not apply because it is the primary key of the *Vehicle* table. On the other hand, if the *Control Source* is *RepairOrder.SerialNo*, AutoLookup applies because it is a foreign key in the *RepairOrder* table.

- Problems relating to the appearance of your form.

⇒ You can change the amount of space on the main form by manipulating the embedded subform (stretching or shrinking it).

⇒ You can rearrange the order of fields in the subform by moving fields in the datasheet. You must invoke the embedded subform in form design view and then change to datasheet view. As a datasheet, you can rearrange the columns, change the size of the columns, and cause a horizontal scroll bar to appear. The horizontal scroll bar appears if the datasheet is wider than the subform. Note that if you do not see a horizontal scroll bar in the datasheet view of the subform, you will NOT see a horizontal scroll bar in the subform when it appears on the main form. This is true even if the datasheet view of the subform is wider than the space allocated on the main form.

⇒ Be careful about the size of the form footer on your main form. The size of the form footer is the space between the footer boundary and the end of form. If the form footer area is too large, only the footer will display. Nothing will display in the detail section of the form. The detail section should contain everything on your main form. To correct this problem, reduce the size of the footer in your main form. You can do this in design mode. Scroll down until you find the form footer divider. Then scroll to the end of the form. Drag the end-of-form boundary so that it is close to the form footer boundary.

- Calculation and naming problems.

⇒ It is generally better to perform calculations (such as `UnitPrice * QtyUsed`) in the underlying query rather than in the form itself. An example of the SQL syntax for a computed field is `UnitPrice * QtyUsed AS PartCost`. If you have a form field defined as a computation, you cannot refer to the form field in an aggregate function. Rather, you must repeat the expression in the aggregate function.

⇒ Access gives the message *#name* when it cannot recognize a field name in an expression. For example in the following expression, a *#name* error is given if the field *units* is misspelled *unit*: `= SUM(unit)`. Such a misspelling error is easy to correct. Often the *#name* errors occur when referencing fields in subforms. You can use the expression builder to check references to subform fields (see the note about form field references in Appendix B).

⇒ For main form fields that reference subform fields, refer to Rule 3 described in Appendix B.

⇒ Do not use qualified field names for the *Name* property of a bound control. For example, do not use *RepairOrder.Ordno* as the name of a textbox. If your underlying query needs a qualified name in the result, rename the column using the AS keyword (SQL) or the rename feature of Query Design.

⇒ Access gives the "#error" message when the *Name* property of a form field matches a field used in the expression of the *Control Source* property. If you are using an expression for the *Control Source* property, make sure that the control name does not match one of the fields used in the expression.

⇒ Access may give the "#error" message when a main form field refers to a field that is computed in a subform. This situation typically occurs when the main form field displays a summary value computed in a subform field (see Section 5.5). The problem is not due to the computation of the summary value in a subform field. Rather the problem is caused by a row computation performed in the subform. For example if the subform contains a field (amount) computed as qty * prodprice, this calculation should be performed in the subform query, not in the subform. The subform footer field can summarize this subform field (SUM(amount)) without a problem. The problem occurs if the subform footer field uses a field computed in another part of the subform. Because of this problem, I recommend performing row calculations in form queries, not in the form.

⇒ On some occasions Access refuses to compute aggregate functions in subform footer fields. You may have an error message like "#name" or "#error". Make sure that you have followed the naming rules described in Appendix B. If you are sure that you have followed the naming rules, here are several tactics to try:

- Try defining a new field in the subform footer and then save the subform changes. Then define a new field in the main form with a reference to the field in the subform's footer and save the form again. If the error appears in the new main form field, delete the new fields (main form and subform) and try the next tactic.
- Close the form and repair the database (**Database Tools → Compact and Repair Database**). If the error still appears, try the next tactic.
- Close the form and return to the Forms section in the Navigation pane. Copy and paste the main form. Open the new copy of the main form to see if the error still appears.

• Problems when testing your hierarchical form.

⇒ Before completing your form, make sure that it works as expected. Test the form by inserting new rows, scrolling through records, changing field values, and deleting records. Perform these operations in both the main and the subform. When inserting a new record in the main form, you must select **Home → New** in the Records group. Pay special attention to combo boxes. Make sure that you can select a new record in a combo box without an error message being issued.

⇒ When testing your main form, remember that updates to the primary key field of the main form are usually not supported. For example, if you change the order number, Access will give an error unless the *Cascade Update* property of the relationship is set to true.

⇒ If you set the *Data Entry* property of a form to "Yes", your form will begin in data entry mode. You will not be able to see existing records. To avoid confusion, you should usually set the ***Data Entry*** property to "No" for both the main and the subform.

Appendix B: Form Field References

This appendix presents rules about referring to form fields in a property or query. The rules are explained in the help documentation but assembling them can be frustrating. Here is a convenient summary to help you.

Rule 1

When referring to a main form field in a property of another main form field, simply use the field name. For example, in a new field in the main form that computes tax (8.5%) on the total part cost, the *Control Source* property should be `TotalPartChg * 0.085`. The field *TotalPartChg* is a reference to a field in the main form.

Rule 2

To refer to a form field inside a query, use the notation `[Forms]![Form Name]![Field Name]`. For example, the Criteria property for the CustNo field in the query of the Serial No. combo box is

[Forms]![RepairOrder2]![CustNo]

The Criteria value references the CustNo field on the main form. The purpose of the query is to restrict the combo box list to those vehicles with the same owner as displayed in the main form.

Rule 3

To refer to a subform field in a property of a main form field, use the notation `[Subform Control Name].[Form]![Field Name]`. For example, the *Control Source* property of the main form field *TotalPartChg* should refer to the subform field *TotalCost*. The *Control Source* property is specified as

=[Parts: Subform2].[Form]![TotalCost]

Note that *Parts: Subform2* is the value in the *Name* property of the embedded form in the main form. If you click (not double-click) on the embedded subform with the Properties window visible, you will see the *Name* property for the embedded subform. Usually the *Name* property has the same value as the subform's name in the Database window. If they are different, use the *Name* property when referencing fields in the subform. Do not use the name of the subform that appears in the Database window.

Rule 4

To refer to a main form field in a property of a subform field, use the notation `Parent![Field Name]`. For example, to refer to the main form field *RepairDate* in an expression of a subform field, use the notation `Parent![RepairDate]`.

Appendix C: Steps to Test a 1-M Query

This appendix presents the steps for testing a 1-M query for insertions to the child table and updates to the parent table. You should follow these steps before using a query as the *Record Source* for a main form or subform. Refer to textbook Chapter 10 for more details about the concepts underlying updatable queries. For the exact rules about modifications supported to 1-M queries, search the help documentation using "updating data from a query" as the keyword and then select the topic "about updating data".

Step 1

Identify the child table in the query. The child table will be the one that you will update by using the query's datasheet.

Step 2

Make sure that the last row contains an asterisk. If it does not, you probably have a `GROUP BY` or `DISTINCT` in your query. Remove these if present. When you see the asterisk in the last row, proceed to step 3.

Step 3

Try to insert a new row in the child table. First, enter the primary key of the child table. Second, enter values for the foreign keys of the child table. You need to use valid (existing) foreign key values. Notice that an AutoLookup query occurs after entering each foreign key value. Third, enter the other required fields in the child table. After entering the values, move the cursor to the new row containing an asterisk (*). If Access does not display an error message, your query is updatable.

Example

Apply the following steps to the example below. In step 1, *PartsUsed* is the child table. In step 2, an asterisk should appear in the last row of the datasheet so you can proceed to step 3. In step 3, enter an existing *OrdNo* (primary key of *RepairOrder*) and an existing *PartsUsed.PartNo* (primary key of *Part*). Notice that an AutoLookup query displays values of *PartDesc*, *UnitPrice*, *UnitSize*, and *UnitsInStock*. After entering a value for *QtyUsed*, move the cursor to the new row. You should not have any error messages. To verify that a new row was added to *PartsUsed*, open the *PartsUsed* table as a datasheet and see the new row that you just added.

Updatable Query Example:

```
SELECT OrdNo, PartsUsed.PartNo, QtyUsed, PartDesc
    UnitPrice, UnitSize, UnitsInStock
    FROM PartUsed INNER JOIN Part
ON PartUsed.PartNo = Part.PartNo
```

This query does not support insertions to the parent table (*Part*) because the primary key of the *Part* table is not in the query result. However, it does support updates to the fields of the *Part* table (*PartDesc*, *UnitPrice*, *UnitSize*, and *UnitsInStock*). To test updates to these fields, open the query as a datasheet and change one of the *Part* fields. You will see the value change in all rows with the same part number.

Appendix D: Using the Form Wizard with Two Queries

This appendix provides practice with using the Form Wizard with a main form query and a subform query. The Form Wizard is most applicable when you are starting from scratch with a main form query and a subform query. The following steps assume that you have created the main form and the subform queries described in sections 5.2 and 5.3.

1. Open the New Form Window: Select the *MainQuery* query. Choose **Create → Form Wizard** in the Forms group.
2. Select Fields to Include: Click the **>>** button to move all of the fields shown in the left side (Available Fields) to the right side (Selected Fields).
   - In the same wizard window, go to the Tables/Queries selection box where the *MainQuery* name appears and select a different query; *Subquery1*. Now, the fields of *SubQuery1* appear under Available Fields. Click the single **>** button to move the following fields to the right side (Selected Fields): *PartNo*, *QtyUsed*, *PartDesc*, *UnitsInStock*, *UnitPrice*, and *UnitSize*. Note, the Selected Fields window now contains fields from both queries. Click the **Next** button when finished.
3. Select Viewing Option: The next window asks how you want to view the data. Select "by MainQuery" because you want the *MainQuery* data to appear in the main form. The option button "Form with subform(s)" should be selected. Select it if it is not selected. Click the **Next** button to advance.
4. Select the Layout of the Subform: In the next wizard window, select "Datasheet" (it should be the default). Click the **Next >** button to advance.
5. Name the Forms: You must provide names for the main form and the subform. If not entered already, type "RepairOrder2New" for the main form and "Parts: Subform2New" for the subform. In the same window, select the second option, "Modify the form's design", and click **Finish**.
6. Customize and Make Format Changes: To customize the main form and the subform, return to section 5.2.3 above and follow the steps. When finished, further modify the main form as described in section 5.3.3 beginning with step 3.

Sometimes Access cannot make the connection between two queries. On earlier versions of Access (2003 and prior), I never encountered this problem. On newer versions of Access (2007 and later), I have seen the form wizard generate an error message at the end of step 2. If you receive an error message, make sure that your main form and subform queries are correctly designed. If your queries are designed correctly and the form wizard error still occurs, you will need to use an alternative approach. You can create a single-level form using the main form query. In design view, you can use the subform/subreport control to add a subform that is bound to the subform query. You should follow the steps in Section 5.6.1 to add a subform/subreport control.

Chapter 6: Report Lab

Learning Objectives

This chapter demonstrates features of Access 2013 reports that provide enhanced formatting compared to datasheets. At the completion of this chapter, you should have acquired the knowledge and skills to

- Understand the role of grouping and sorting in report design.
- Use the Report Wizard to create reports.
- Modify reports using the Report Design window.
- Create a report that uses a query.
- Create a parameter query and use it in a report.
- Create a crosstab query and use it in a report.

Overview

In Chapters 4 and 5, you learned how to create forms for data entry and display. The forms that you created provide a much better interface than a datasheet. When you only need to display data, you should create a report rather than a form. Reports provide a stylized presentation that can be much easier to read than a datasheet.

This chapter provides guided instruction about creating reports that utilize sorting and nesting. Nesting allows users to spot trends and relationships among a large amount of data. As in Chapter 5, you will develop several reports of increasing complexity. In developing the reports, you will learn how to define a query for a report, use the Report Wizard to create an initial report, add groups and summary calculations to a report, and use a parameter query to customize the data presented to the user. The practice in this chapter complements the conceptual material in textbook Chapter 10 about query formulation for reports.

At the end of this chapter, you will develop a crosstab query to demonstrate how Access supports decision making with multidimensional data. This practice complements the conceptual material in textbook Chapter 16.

6.1 Creating a Simple Report

A report is a stylized presentation of data appropriate to a selected audience. A report is similar to a form since they both typically use an underlying query (the *Record Source* property) and present data differently than it appears in the database tables. Because both form and report design involves presentation, many form design skills also apply to report design. Also, since a report is similar to a form, a user does not perceive any of the complexity of the underlying query, tables, and mapping between the report and the tables.

This section begins your exploration of reports. You will use the Report Wizard to develop a simple report. After creating the report, some explanation about its components is presented.

6.1.1 Using the Report Wizard

You will begin by creating a simple version of the part report that uses the *Part* and the *RepairOrder* tables. Using the Report Wizard, you will create the report in a way similar to creating a main form and a subform with the Form Wizard. You will first select fields from the *Part* table and then again from the *RepairOrder* table. The steps are as follows:

1. Open the Report Wizard: Choose **Create → Report Wizard** in the Reports group of the Ribbon. Select *PartsUsed* in response to the table or query question.

2. Select Fields to Include: Click the >> button to move all of the fields shown in the left side (Available Fields) to the right side (Selected Fields).

   - In the same wizard window, go up to the Tables/Queries selection box where the *PartsUsed* table name appears and select *RepairOrder*.
   - From the *RepairOrder* table click the single > button to move the *TimeRecvd* field to the right side (Selected Fields). Note, the Selected Fields window will contain fields from both the *PartsUsed* table and the *RepairOrder* table (Figure 1). Click the **Next >** button when finished.

3. How to View Data: Select "by PartsUsed" and click **Next >**.

4. Add Grouping Levels: Select and add *PartNo,* then *TimeRecvd* using the > button. However, *TimeRecvd* must be changed because it defaults to *TimeRecvd by Month* instead of "by Day". To correct this, perform the following:

   - Click the **Grouping Options…** button at the lower left of the window (Figure 2). Note that the Grouping Intervals column for the *TimeRecvd* field is set to "Month". Change the grouping interval to "Day" for the *TimeRecvd* field by choosing "Day" from the drop down menu. After clicking the **OK** button, the Grouping window appears as in Figure 3. Click the **Next >** button to continue.

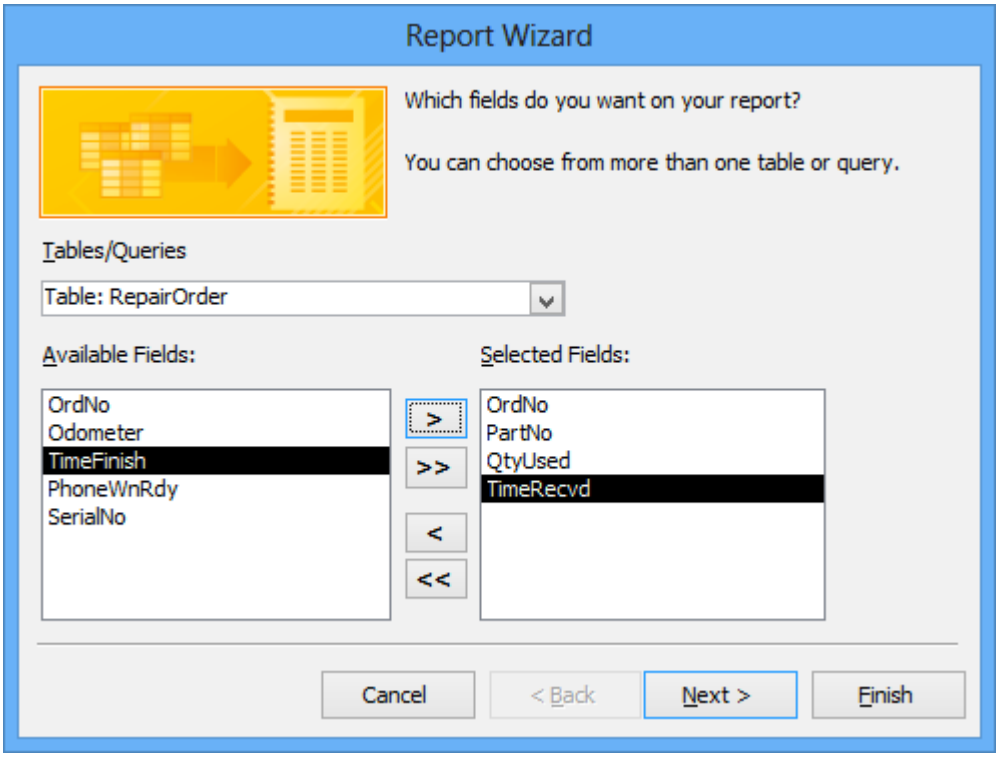


Figure 1: Fields in the Report

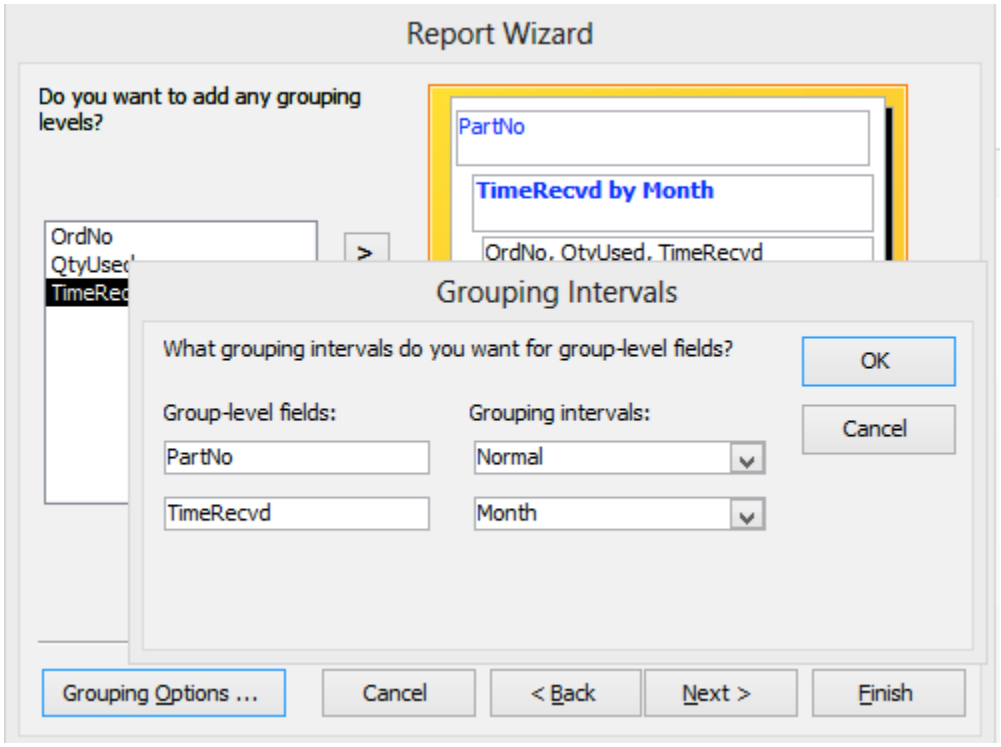


Figure 2: Grouping Window and Grouping Interval Window

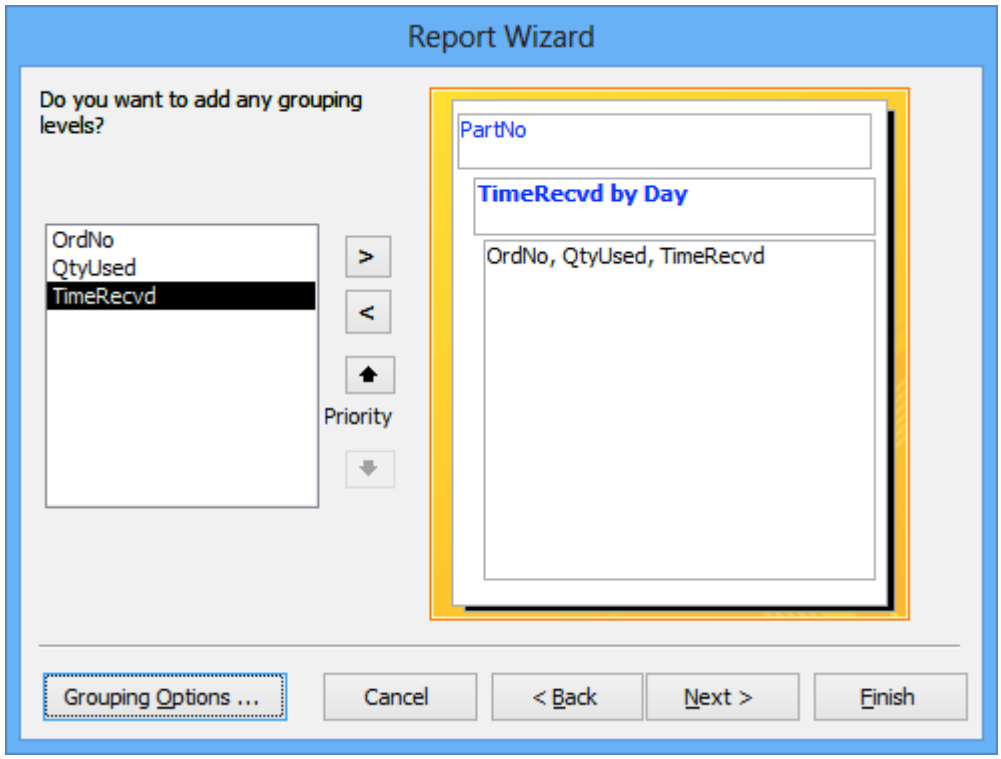


Figure 3: Corrected Grouping Window

5. Sort Order: Select *OrdNo* with an ascending sort order, and click **Next >**.

6. Report Layout: Select "Stepped" and "Portrait" from the option buttons in the top part of the window. Later you can experiment with other styles. Make sure that the check box is selected to adjust field widths. Click the **Next >** button.
7. Title the Report: Type "Part Report 1" and click **Finish**.
8. View the Report: The report is opened in the Print Preview window (Figure 4). The data are grouped by *PartNo* and *TimeRevcd by Day*. Within *TimeRecvd,* the records are sorted by *OrdNo*. Note that the Report Wizard default duplicated the *TimeRecvd* field.
9. Toggle to Design View: Close the print preview using the **Print Preview → Close Print Preview** button in the Ribbon. To toggle between the Print Preview and the Design View windows, click the **View** dropdown button on the Ribbon or use the View pane. You can also right click on mouse to switch between views.

> Technical Note: The Layout View shows a simplified preview of a report. It only provides a quick look to verify the report's formatting, not the actual data in the report. Selection criteria and joins are ignored in layout preview. Thus, Layout View is useful when your report contains a large amount of data and hence takes a long time to produce the complete print preview.

10. Close the Report: The report is saved under the name given for the report title.

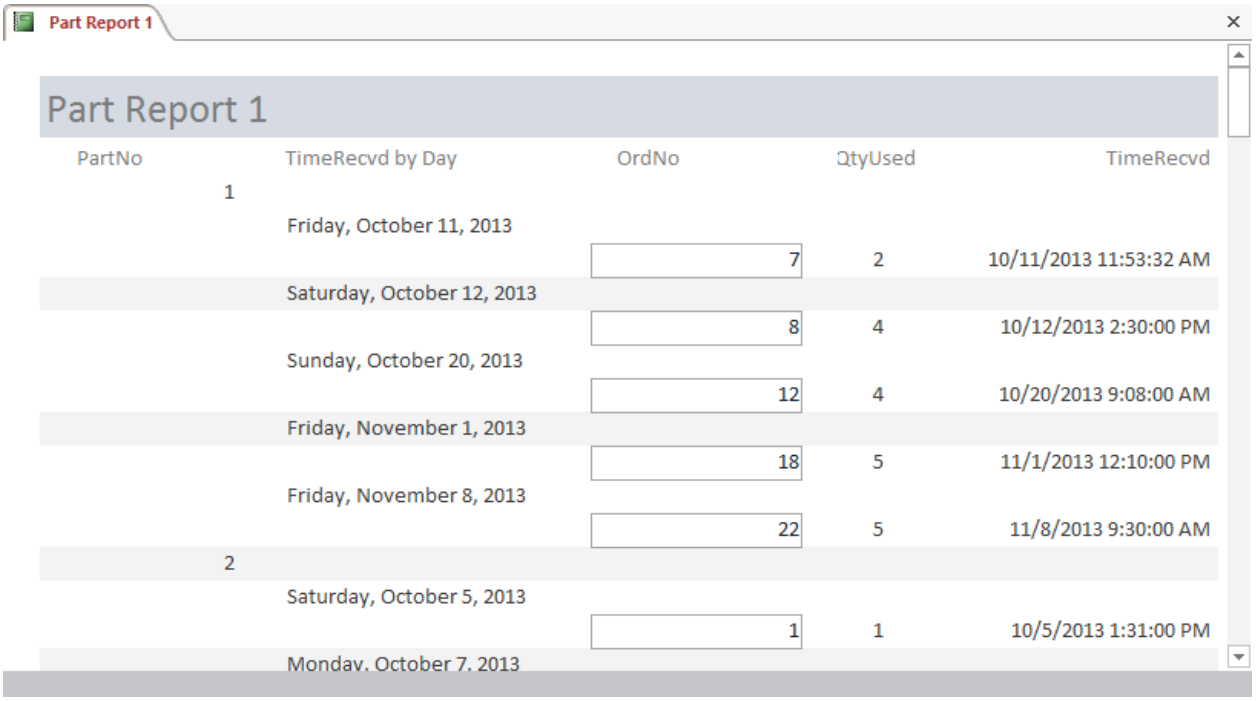


Figure 4: Print Preview of *Part Report 1*[13]

6.1.2 Understanding Report Sections

A report design is divided into a number of sections (Figure 5). The report header describes what happens just after the report begins. Typically a heading is printed to identify the report. Likewise, the page

[13] If the TimeRecvd field displays with ##### instead of values, you will need to increase the length of the field in Design view. After increasing the width of the textbox, it will display values instead of ####.

header describes what happens just after a new page begins. Typically you want to display column headings at the top of each page. For each grouping field, there is also a group header. Typically you want to display the value of the group field in the group header. The Report Wizard automatically creates a number of controls in each header, as shown in Figure 5.

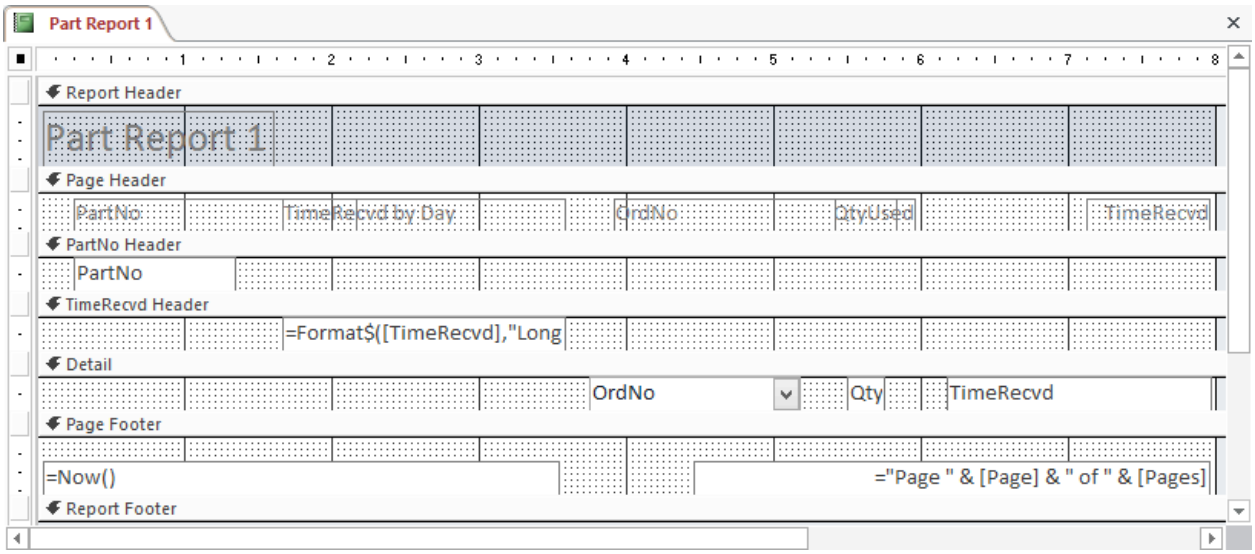


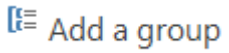
Figure 5: Design View of *Part Report 1*

The detail section contains the actual data from the record source of the report. Typically, the values of non-grouped fields are displayed in the detail section. Unlike group sections, the detail section does not contain headers and footers.

A footer summarizes a group as a total or a subtotal. Typically, some kind of summary is computed such as an average or sum. However, a group footer may be omitted if no summary data are necessary. The page and the report footers are specially designated footers for the end of a page and the end of a report, respectively. Typically, page numbers are displayed at the end of a page, while a final report summary is displayed at the end of a report.

Viewing the Sorting and the Grouping Fields

If you need to change the grouping and the sorting fields, you do not have to start over with the Report Wizard. To view the grouping and the sorting fields, click **Design → Group & Sort** in the Grouping & Totals group. In the Sorting and Grouping window (Figure 6), note the grouping fields (*PartNo* and *TimeRecvd*) and the sorting field (*OrdNo*). Even though a field may not be a "grouped" field, you can still sort on it. You can delete the sorting on *TimeRecvd* by clicking the delete button (✕) because it has no effect with grouping on *TimeRecvd*.

If you need to add a new grouping field or change the order of the grouping fields, you can use the Sorting and Grouping window. You can add a grouping field using the **Add a group** button 

and then selecting the appropriate field in the field list. Similarly you can add a sorting field using the **Add a sort** button. After making changes in the Sorting and Grouping window, the changes take effect in design view.

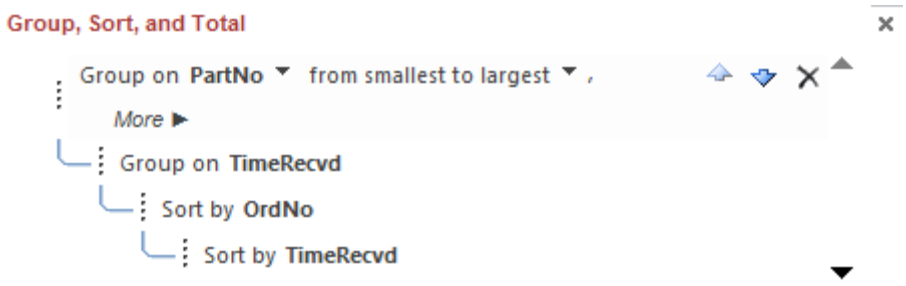


Figure 6: Sorting and Grouping Window

6.2 Using the Report Wizard with a Query

The initial version of the part report is obviously inadequate as it does not show some important data. You will revise the report to contain fields for *Part Name*, *TimeRecvd*, *Total Quantity*, *Parts Used*, and *Total Cost*. Note, the last two fields are calculated.

As a first step to improving the report, you need to change the *Record Source* property to a query. The query joins fields from the *Part,* the *RepairOrder,* and the *PartsUsed* tables as well as performs a calculation.

6.2.1 Creating the Query for the Revised Report

To begin, return to the **Query** section in the Database window. Create a new query and type the following SELECT statement into the SQL window. Save the query as *RptQuery1*.

```
SELECT Part.PartDesc, format(RepairOrder.TimeRecvd,
 "Medium Date") AS RepairDate, SUM(PartsUsed.QtyUsed)
 AS TotQty, SUM(PartsUsed.QtyUsed*UnitPrice) AS TotCost
FROM RepairOrder INNER JOIN (Part INNER JOIN PartsUsed
 ON Part.PartNo = PartsUsed.PartNo)
 ON RepairOrder.OrdNo = PartsUsed.OrdNo
GROUP BY Part.PartDesc, format(RepairOrder.TimeRecvd,
 "Medium Date");
```

As you experienced when creating *Part Report 1*, the Report Wizard can develop multiple-table reports without using a predefined query. However, there is a slight speed advantage if you define the query first, rather than selecting fields from multiple tables in the Report Wizard. In addition, you should define the query separately when it contains aggregate computations. Textbook Chapter 10 (section 10.5) presents guidelines for writing report queries including when a report query should contain aggregate computations.

6.2.2 Using the Report Wizard to Create the Revised Report

With the above query formulated, you are now ready to develop the revised report by answering the questions of the Report Wizard as shown below.

1. Open the Report Wizard: Choose **Create → Report Wizard** in the Reports group and select *RptQuery1* in response to the table question.
2. Select Fields **to** Include: Click the **>>** button to move all of the fields shown in the left side (Available Fields) to the right side (Selected Fields), Click the **Next >** button to continue.

3. Add Grouping Levels: Select and add *PartDesc* using the **>** button. Your Report Wizard window should appear as in Figure 7. Click the **Next >** button to continue.

4. Sort Order and Summary Information for Detail Records: Select *RepairDate* with an ascending sort order as shown in Figure 8.

   - In the same Report Wizard window, click the **Summary Options…** button to set summary values to be calculated. Check the *Sum* column for both *TotQty* and *TotCost* as shown in the Summary Options window in Figure 8. Also click the **Detail and Summary** option button to display detail lines and group totals.

   - Click **OK** to close the Summary Options window and click **Next >**.

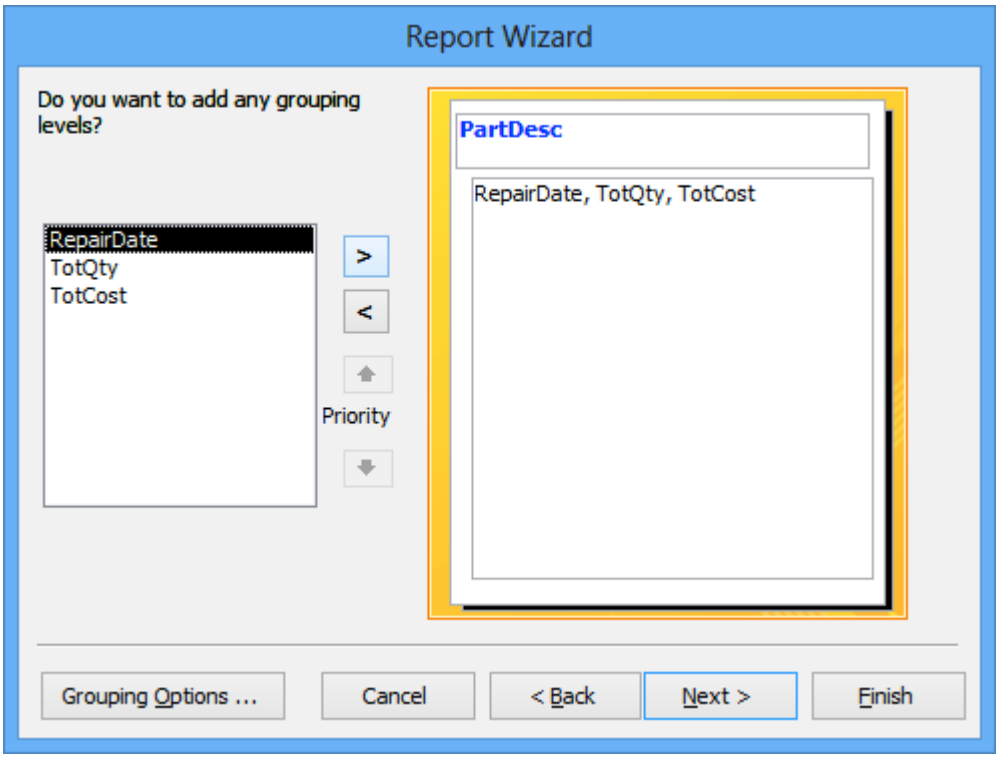


Figure 7: Report Wizard Grouping Window

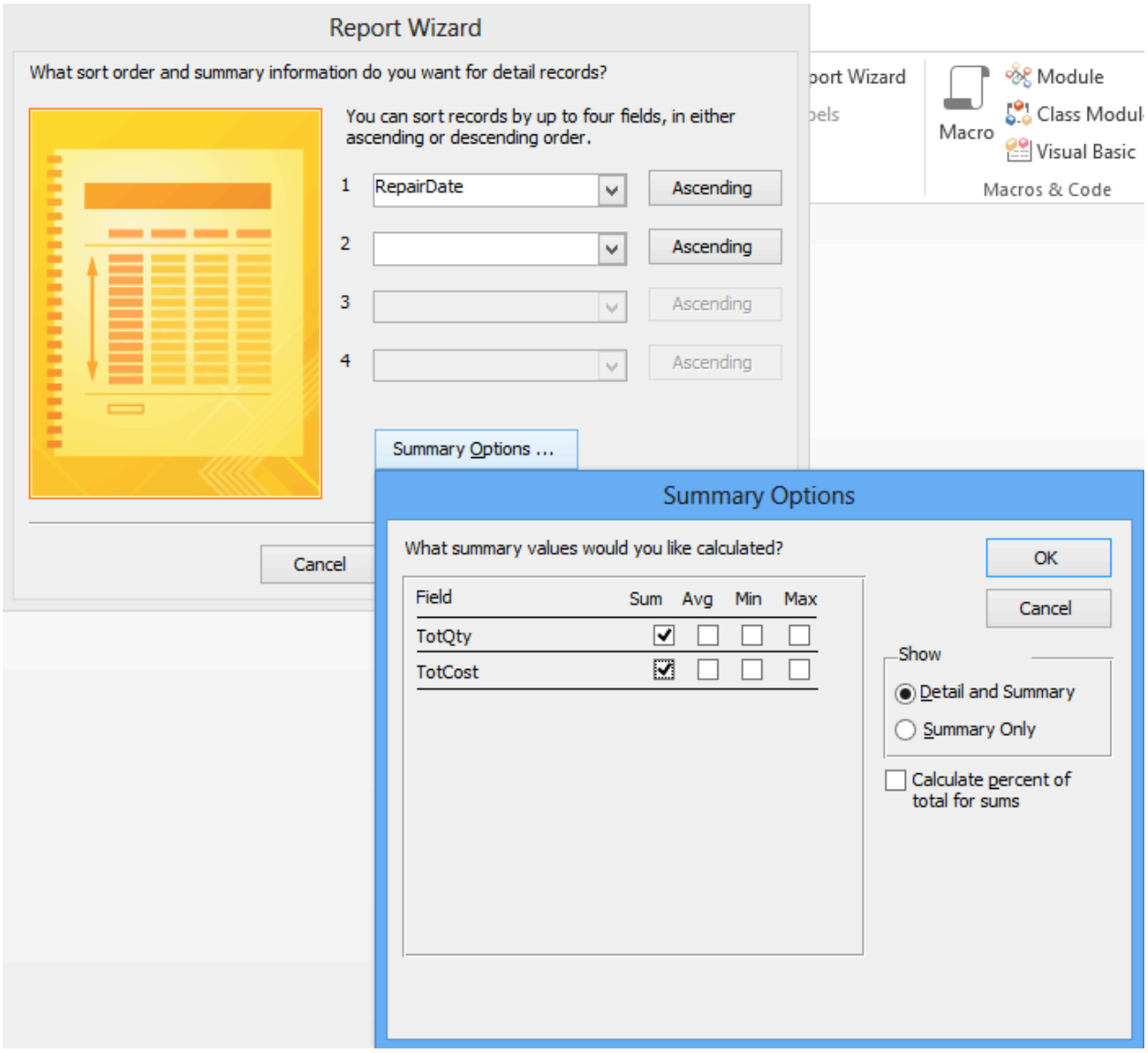


Figure 8: Sort Order Window with Overlapping Summary Options Window

5. Report Layout: Select "Stepped" and "Portrait" from the option buttons in the top part of the window. Make sure that the check box is selected to adjust field widths. Click the **Next >** button.
6. Title the Report: Type "Part Report2" and click **Finish**.
7. View the Report: The report is opened in print preview (Figure 9). The data are grouped by *PartDesc* and sorted by *Repair Date*. Close the report after viewing it.

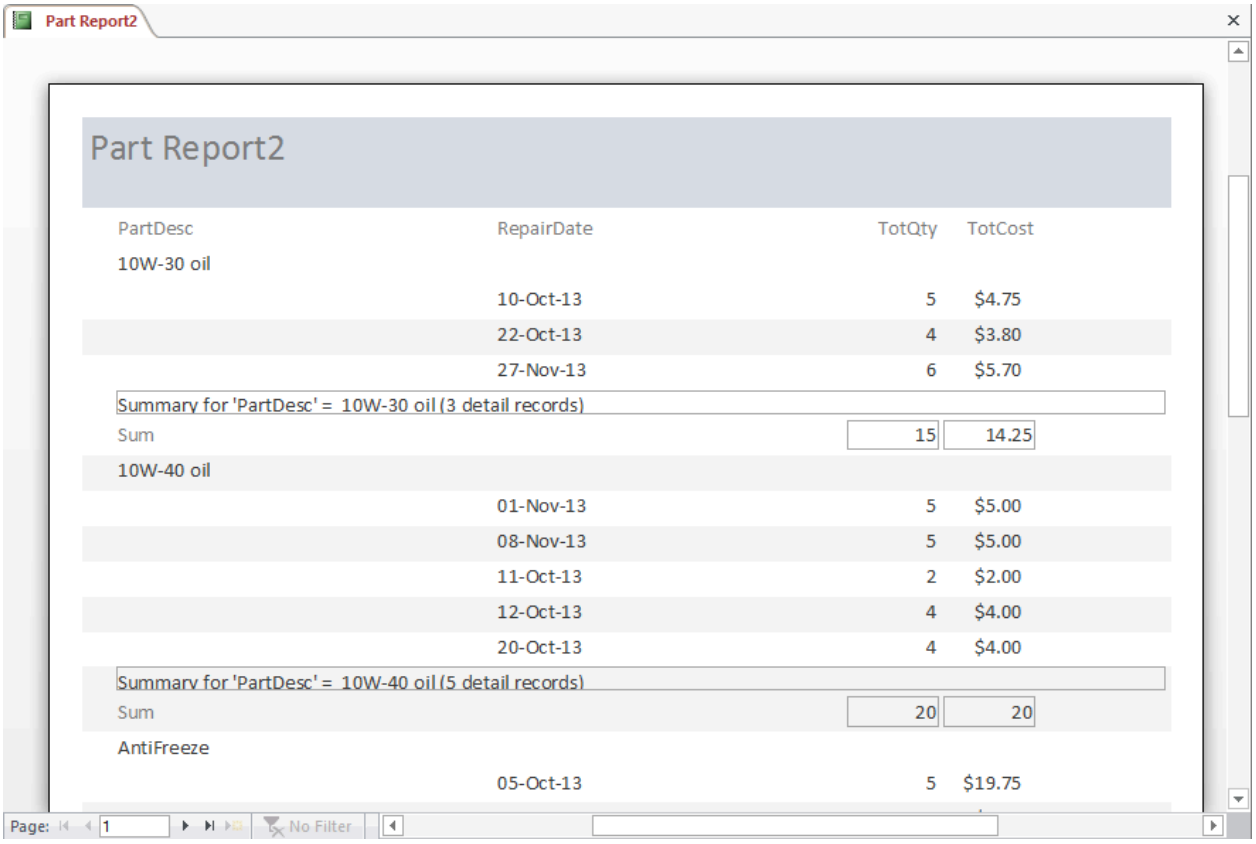


Figure 9: Print Preview of *Part Report2*

6.3 Modifying a Report

The third version of the part report is still inadequate. It needs another group for the month of a repair and additional summaries for the average, the minimum, and the maximum quantities and costs. In addition, the report needs many format changes to improve its appearance. This section provides you experience with the report design view and various format properties. You will begin by using the Report Wizard and *RptQuery1*.

6.3.1 Creating the Revised Report with the Report Wizard

To begin, return to the **Query** section in the navigation pane. Create a new query and type the following SELECT statement into the SQL window. Save the query as *RptQuery2*.

```
SELECT Part.PartDesc, RepairOrder.TimeRecvd AS
 RepairDate, SUM(PartsUsed.QtyUsed) AS TotQty,
 SUM(PartsUsed.QtyUsed*UnitPrice) AS TotCost
FROM RepairOrder INNER JOIN
 (Part INNER JOIN PartsUsed
 ON Part.PartNo = PartsUsed.PartNo)
 ON RepairOrder.OrdNo = PartsUsed.OrdNo
GROUP BY Part.PartDesc, RepairOrder.TimeRecvd
```

With the above query formulated, you may develop the next report using the Report Wizard by following the steps below:

1. Open the Report Wizard: Choose **Create → Report Wizard** and select *RptQuery2* in response to the table question.
2. Select Fields to Include: Click the >> button to move all of the fields shown in the left side (Available Fields) to the right side (Selected Fields). Click the **Next >** button to continue.
3. Add Grouping Levels: Select and add *PartDesc* and then *RepairDate* using the > button. Note that *RepairDate* defaults to *RepairDate by Month* as shown in Figure 10. Click the **Next >** button to continue.
4. Sort Order and Summary Information for Detail Records: Select *RepairDate* with an ascending sort order.
5. Change Summary Options: In the same Report Wizard window, click the **Summary Options…** button to set summary values to calculate. Check *Sum*, *Avg*, *Min*, and *Max* (all columns) for both *TotQty* and *TotCost*. Also check the **Detail and Summary** option button. Click **OK** to close the Summary Options window. Click the **Next >** button to continue.
6. Report Layout: Select "Stepped" and "Portrait" from the option buttons in the top part of the window. Do not select the check box to adjust field widths as shown in Figure 11. Click the **Next >** button.
7. Title the Report: Type "Part Report3" then click **Finish**.
8. View the Report in Print Preview (Figure 12). Note that the report does not fit on the page. Toggle to design view to make format changes (Figure 13).

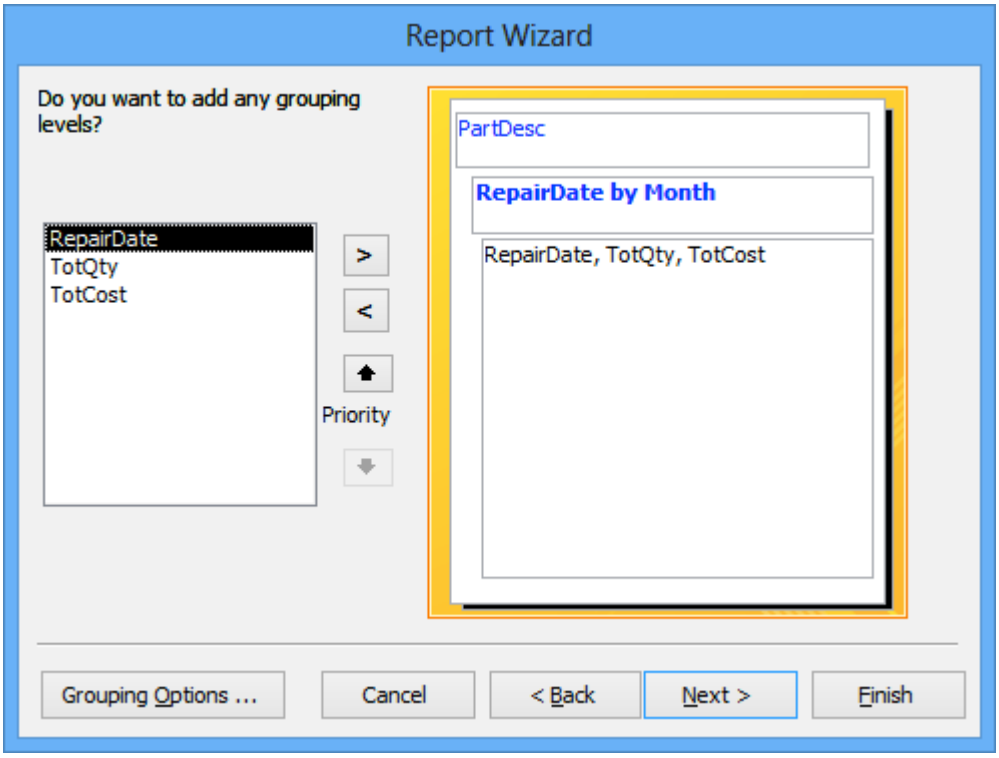


Figure 10: Report Wizard Grouping Window

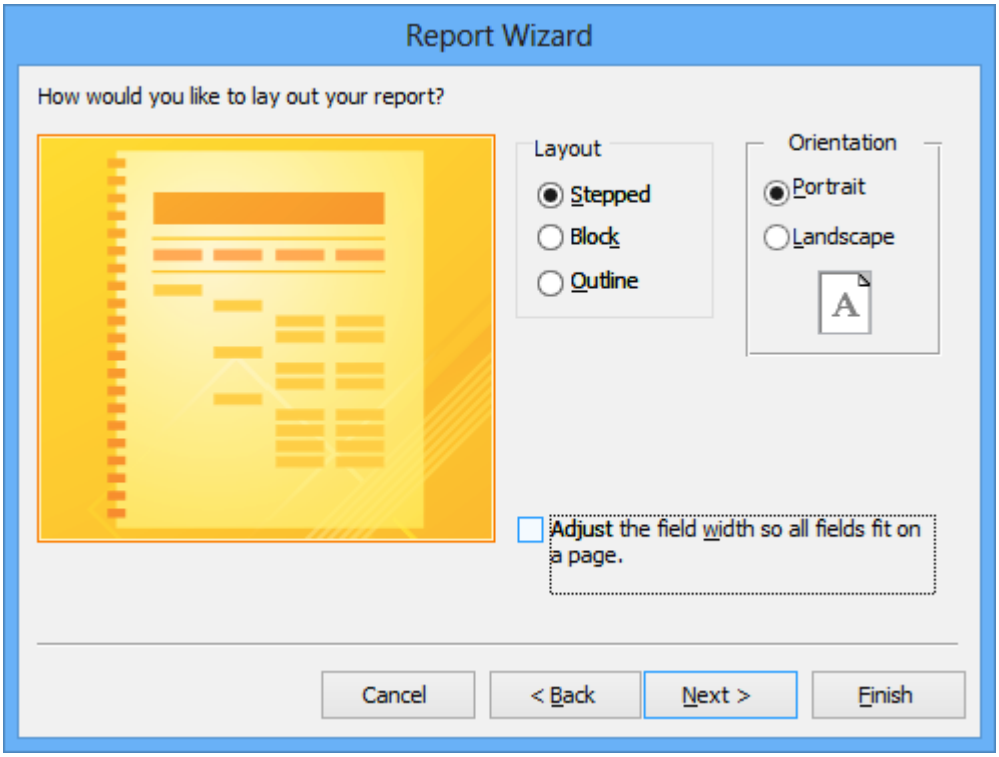


Figure 11: Uncheck “Adjust the field width so all fields fit on a page”

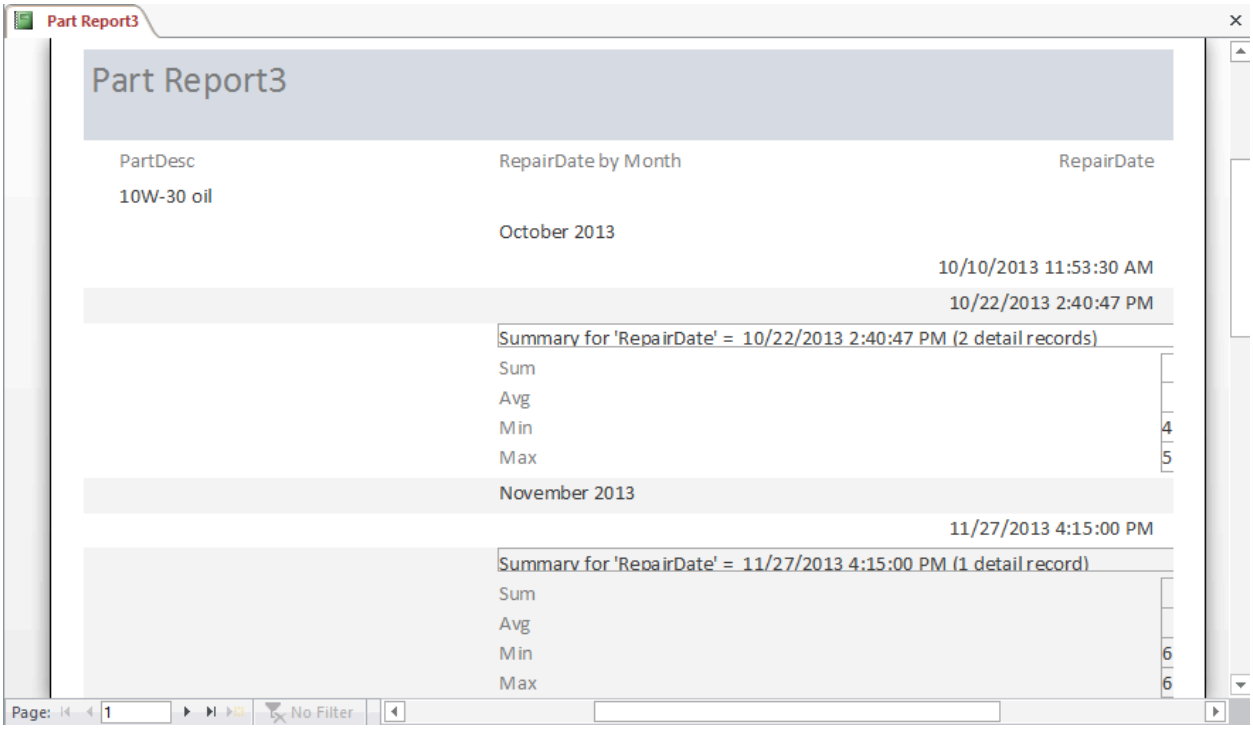


Figure 12: Print Preview of *Part Report3*

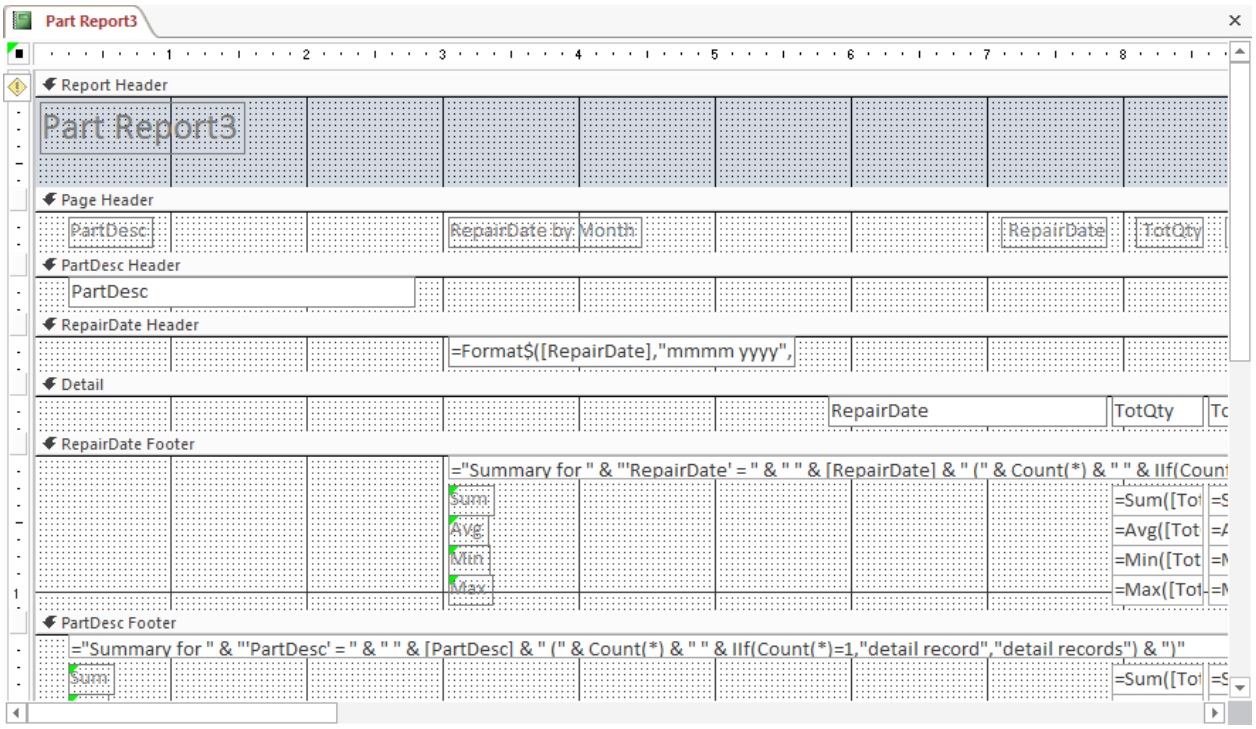


Figure 13: *Part Report3* in Design View before Changes

6.3.2 Guidelines for Making Format Changes

After completing the Report Wizard for *Part Report3*, you will find that the report (Figure 12) needs modifications to the alignment and visibility of controls. In addition, captions and some computations need adjustments. Because these modifications are rather tedious, some guidelines can help to improve your efficiency. This section presents guidelines for manipulating controls as a unit and avoiding layout problems.

- Fomat and Move Controls as a Unit: To group controls as a unit for the purpose of formatting or moving, you can either (a) drag a grouping rectangle around them, (b) click on each control while holding down the **Shift** key, or (c) click on each control while holding down the **Shift** key and click **Arrange → Size/Space → Group (**  **)**. Either way, the controls will be grouped as a unit. Choose **Arrange → Size/Space → Ungroup (**  **)** to ungroup individual controls.
  - To Format the Group Unit: You can change fonts and use bold, italics, or underline with the toolbar. Your changes will affect all of the controls at the same time. You can use the commands in the **Arrange** tab in the Report Design Tools dynamic part on the Ribbon (Figures 14) to align, size, or space controls. The **Align** and **Size/Space** menus are also available by clicking the right mouse button. To obtain a precise fit, you can use the **Arrange → Size/Space → To Fit** command. The control will automatically adjust to the size of its data if you use this formatting feature.
  - To Move Controls as a Unit: While the unit is still selected, drag the unit to the desired location.
- Set Control Properties as a Unit: To change the same property for each control in a unit (except individual properties such as *Caption*, *Name*, and *Control Source*), click the right mouse button (while the unit is selected) and select **Properties** to open the Property Sheet. Alternatively, you can click **Design | Tools → Property Sheet**. The Property Sheet will be titled "Multiple selection". When you make a property change, it will affect all of the controls in the unit.
- Change the Caption: To change an individual property such as *Name* or *Caption*, you do not need to

select each control and individually open its Properties window. A faster way is to click on the first control and open its Property Sheet. After changing its individual properties, do not close the window. While the Property Sheet is still open, select the next control and the Property Sheet changes accordingly.

- Aligning Controls Individually: To fine-tune alignment, you can move controls individually the same as when designing forms (see Chapters 4 and 5). While a textbox or label is selected, point at it until you have the index finger cursor (instead of the small hand) and hold the mouse down while moving the control.
- Copy Label and Textbox Format with the Format Painter: Select the label or textbox containing the format that you want to copy. After selecting the source control, click the **Format Painter** button  in the Font group of the **Format** tab. The cursor then changes to a paintbrush. To apply the formatting, click on the target control (label or textbox) to change its formatting.
- Toggle to Print Preview after Each Adjustment: To see the effects of your changes, toggle to print preview and then toggle back to design view.
- Inspect Visibility and Alignment: Before finishing, you should review each page in print preview to ensure that all the controls are visible and aligned correctly.
- Eliminate Extra Pages: If your report contains extra pages, you may have a conflict between the report layout and the right-hand margin. To eliminate extra pages, adjust the *Widths* property (no longer than 8" for portrait layout) and make sure that all controls fit entirely within the right-hand margin.

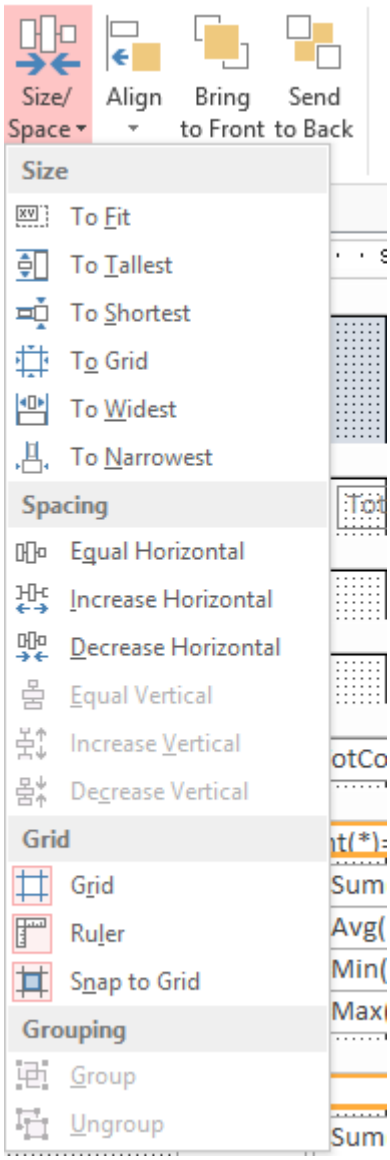


Figure 14: Size/Shape Items in the Arrange Tab of the Ribbon

6.3.3 Making Format Changes

You are now ready to use the suggestions of the previous section about formatting to revise *Part Report3* to improve its appearance. Use Figures 15 and 16 as a guide to the appearance of *Part Report3* after your modifications are finished. Table 1 summarizes the changes by report section. Note that formatting a report can be a tedious process. These changes may take you considerable time to complete.

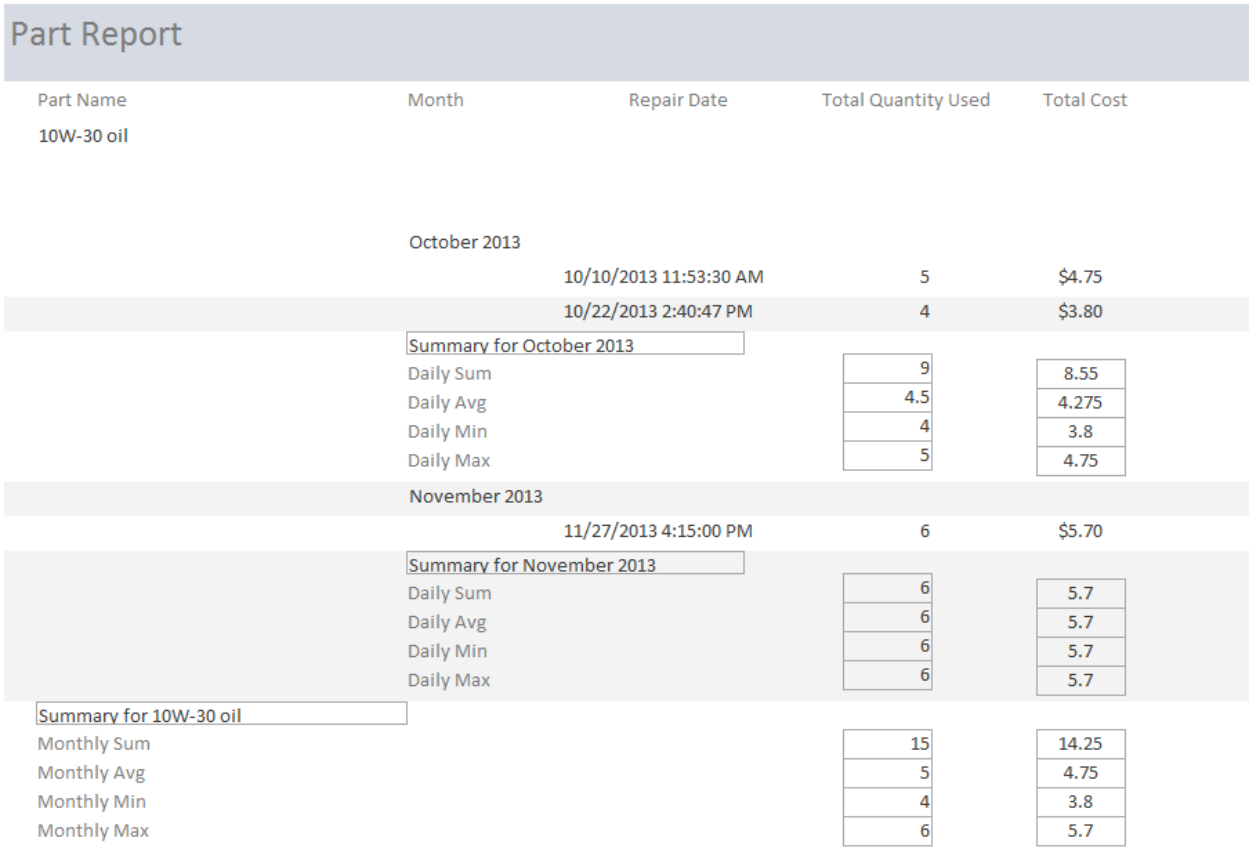


Figure 15: Print Preview of Modified *Part Report3* Showing Report Sections

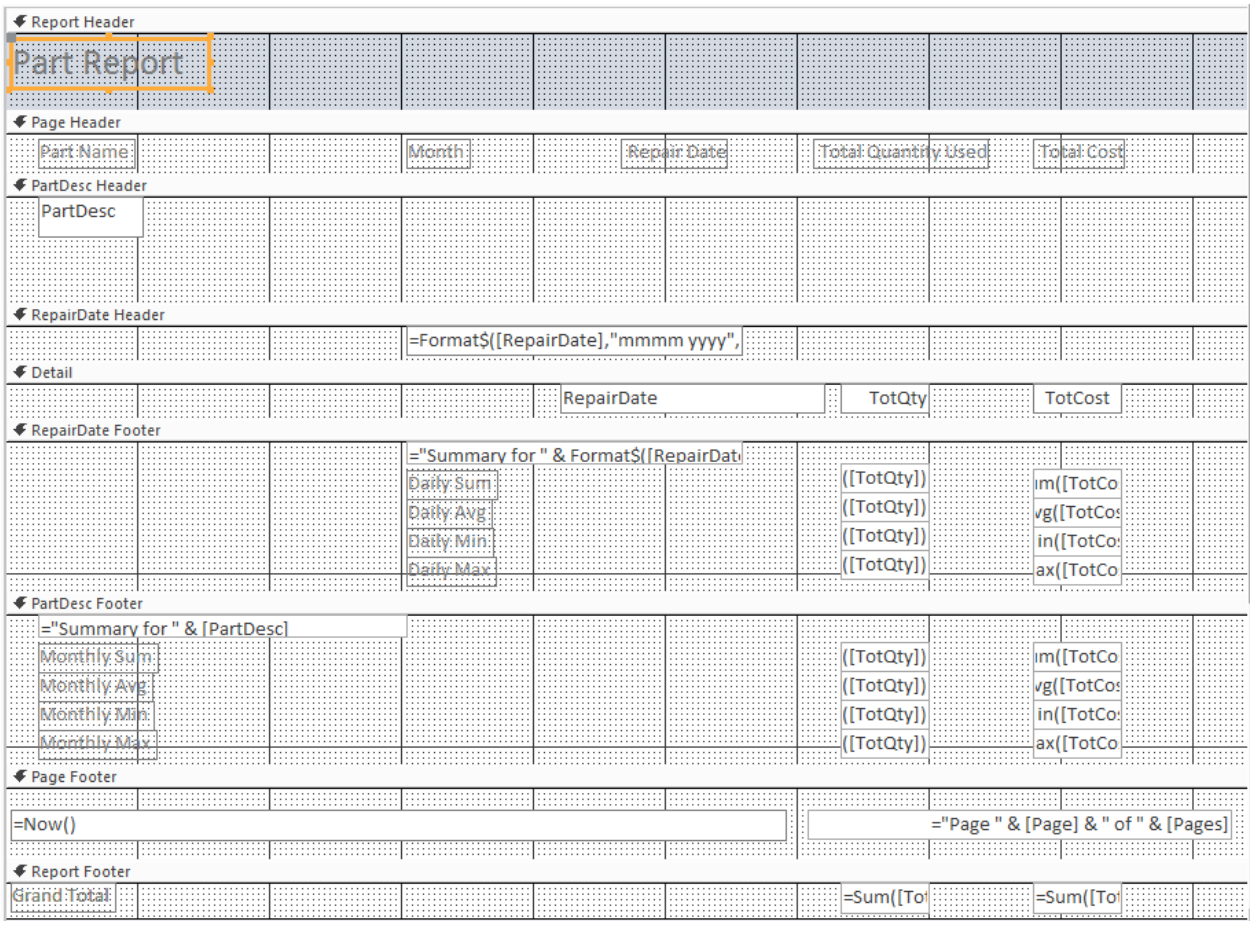
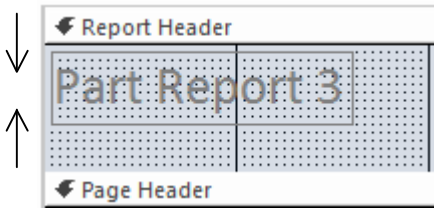


Figure 16: Design View of Modified *Part Report3* Showing Most Report Sections

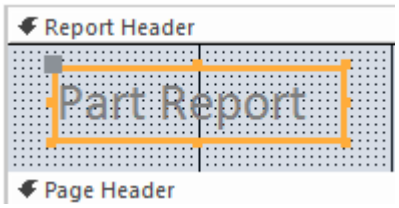
Table 1: Summary of Report Modifications

Report Section	**Changes**
Report Header	Modify label caption to read "Part Report" and decrease header height (Figure 17).
Page Header	Change label captions, align labels across header, adjust label sizes "To Fit", and move labels to fit in 8 inch page width (Figure 18).
PartDesc Header	Increase header height and align the textbox (Figure 19).
RepairDate Header	Align "Repair Month" under its label and decrease textbox width; insert line (Figure 20).
Detail Section	Align textboxes under their labels (Figure 21), left justify (*TextAlign* property to "left") the *RepairDate* textbox, set *Format* property of *TotCost* textbox to "Currency"..
RepairDate Footer	Modify summary textbox expression in *Control Source* property (use the same expression as the repair date header); change captions, font, size, and alignment of labels; change alignment, font, *Text Align* and *Format*

	properties of textboxes (Figure 22).
PartDesc Footer	Modify summary textbox expression in *Control Source* property; change captions, font, size, and alignment of labels; change alignment, font, *Text Align* and *Format* properties of textboxes (Figure 23).
Page Footer	Move and reduce page number textbox to the left so it fits in 8 inch margin.
Report Footer	Textboxes: set to bold, change the *Text Align* and the *Format* properties (Figure 24), and align textboxes under footer textboxes.



Report header before modifications



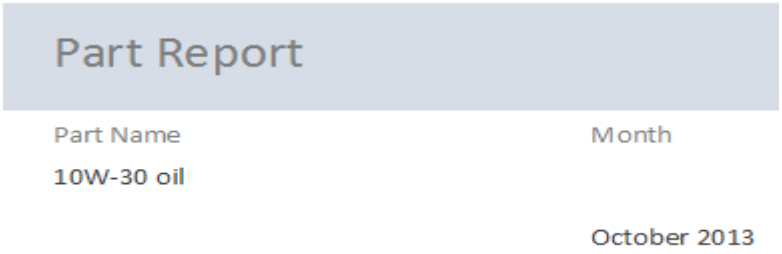
After label change and decreasing height

Figure 17: Report Header Modifications

Page header before modifications

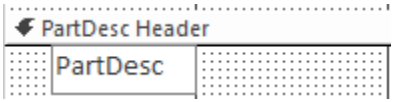
Page header after aligning labels

Figure 18: Page Header Modifications



Print preview of *PartDesc* header before modifications

Design view of *PartDesc* header before modifications



Design view of *PartDesc* header after increasing header height

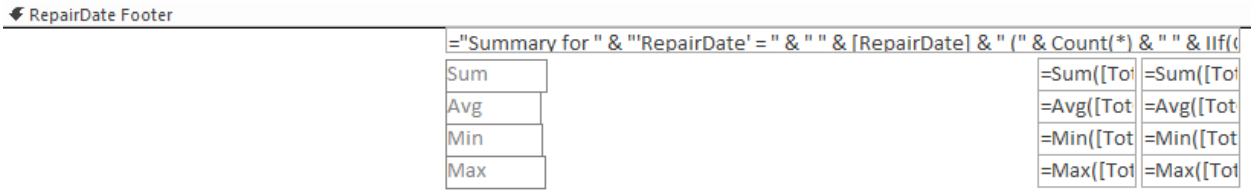
Figure 19: *PartDesc* Header Modifications

Design view of *RepairDate* header after alignment

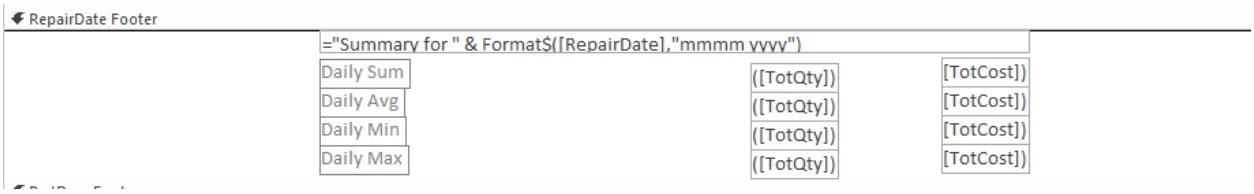
Design View of *RepairDate* header after reducing textbox size

Figure 20: *RepairDate* Header Modification

Figure 21: Design View (without gridlines) of Detail Section after Aligning Textboxes

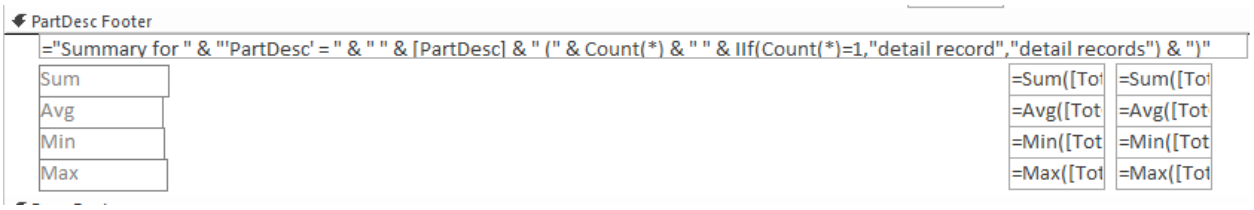


RepairDate footer before modifications



RepairDate footer (without gridlines) after modifications

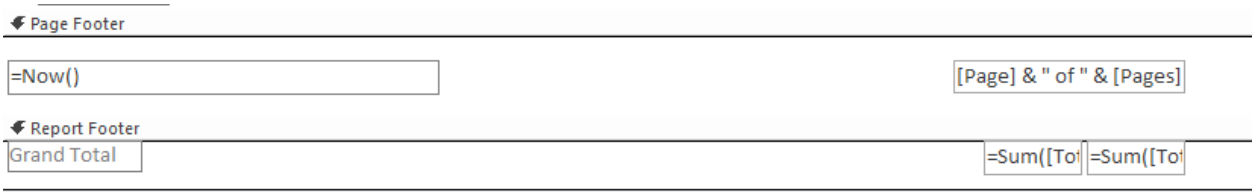
Figure 22: *RepairDate* Footer



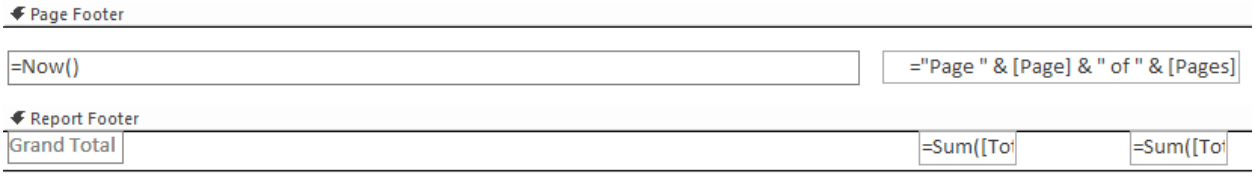
Design view of *PartDesc* footer before modifications

Design view (without gridlines) of *PartDesc* footer after modifications

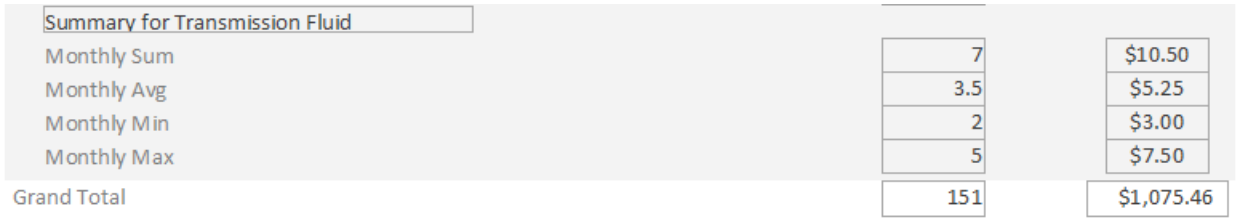
Figure 23: *PartDesc* Footer Modification



Design view of report footer before modifications



Design view (without gridlines) of report footer after modifications



Print preview of report footer after modifications

Figure 24: Report Footer Modification

6.4 Additional Kinds of Queries and Reports

You will study two kinds of queries, parameter queries and crosstab queries, which can be used in reports. The report created with a parameter query will prompt a user for data before the report is displayed. The crosstab query displays results in a spreadsheet format to provide a simple multidimensional data representation. To provide perspective, you can compare the crosstab result with the grouping query result with which you are already familiar.

6.4.1 Using a Parameter Query in a Report

Parameter queries allow a selection criterion value to be determined at the time a query is executed. Often a user may want a report executed against a slightly different set of data. Instead of having to write slightly different queries each time, Access supports a criterion value entered as a parameter. When the query executes, the user is prompted for the parameter value.

Modify Query

To begin, return to the **Queries** section in the Navigation pane. Copy *RptQuery2* and paste it as *RptQuery3*. In SQL view, modify the SELECT statement by adding a WHERE clause. The underlined parts are new from *RptQuery2*.

```
SELECT Part.PartDesc, RepairOrder.TimeRecvd AS
 RepairDate, SUM(PartsUsed.QtyUsed) AS TotQty,
 SUM(PartsUsed.QtyUsed*UnitPrice) AS TotCost
FROM RepairOrder INNER JOIN
 (Part INNER JOIN PartsUsed
 ON Part.PartNo = PartsUsed.PartNo)
 ON RepairOrder.OrdNo = PartsUsed.OrdNo
WHERE Month(TimeRecvd) = [Enter Repair Month: ]
  AND Year(TimeRecvd) = [Enter Repair Year: ]
GROUP BY Part.PartDesc, RepairOrder.TimeRecvd
```

Revise *Part Report3*

When the report executes, the user will be prompted for the parameter value. The parameter value will be used to filter the data in the query and to display in the report heading. Develop the revised report by following the steps below:

1. Copy and Paste *Part Report3*: Begin by copying *Part Report3* and pasting it as *Part Report4* in the **Reports** section of the Navigation pane. Open the new report in design view.
2. Change Report *Record Source* Property: Open the Property Sheet for the Report and change the *Record Source* property to the new query *RptQuery3*. In addition, change the *Caption* property to "Part Report 4".
3. Change Report Heading: The report heading must be modified to accommodate the parameter query.
   - Delete Report Heading Label: Delete the existing label ("Part Report") in the report heading and replace it with a text box from the toolbox. Delete the adjoining label but keep the textbox.
   - Set the *Control Source* Property: Open the Property Sheet for the new textbox. Click in the *Control Source* property area and then the ellipsis (**…**) button to open the Expression Builder window. Type in the following expression:

```
="Auto Repair Shop Part Report for " & Format(DateValue([Enter Repair Month: ] & "/1/" & [Enter Repair Year: ]),"mmmm") & " " & [Enter Repair Year: ]
```

This expression uses the parameter values entered by the user. The *Format* function takes a date and gives the month of the date as a result. The *DateValue* function converts a string into the internal date representation. The *Concatenation* function (&) combines two string values into a longer string value.

- Set Other Properties: Set the *Width* property to "4", the *Font Size* property to "12", and the *Font Weight* property to "Bold" (Figure 25). Click **OK**.

Figure 25: Report Header after Modification

4. Delete RepairDate Header: Click **Design → Group & Sort** and there will be four entries as shown in Figure 26. Select the second and third entries, "RepairDate", and delete them. Click **Yes** when the warning dialog appears (Figure 27). The Sorting and Grouping window will appear as in Figure 28. Close the window.

5. Page Header Changes: Delete the *Month* label and align the remaining labels evenly across the page header (Figure 29).

6. Align Detail Texboxes: Select the textboxes and align them under the appropriate label in the page header section (Figure 29).

7. *PartDesc* Footer: Select each column of four textboxes as a unit and move them accordingly under the appropriate detail textboxes (Figure 29).

8. Toggle to Print Preview: This will give you a chance to try out your parameters query. Note, when you are prompted for the month, use the month number, not the word, and use the four digit year, such as 2013, rather than just 13. When the report appears, check that the items in the report are positioned correctly (Figure 30). If they are not positioned well, return to design view and make adjustments. Otherwise, close the report.

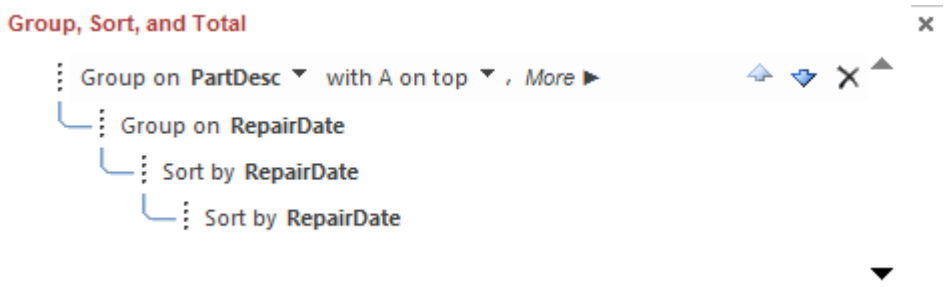


Figure 26: Sorting and Grouping Window before Deletions

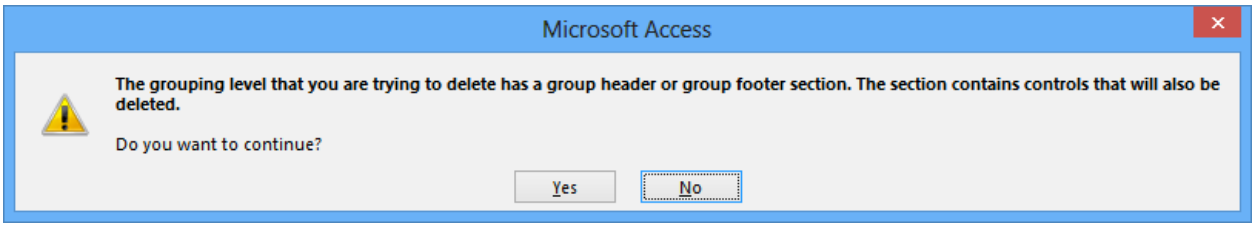


Figure 27: Group Deletion Warning Dialog

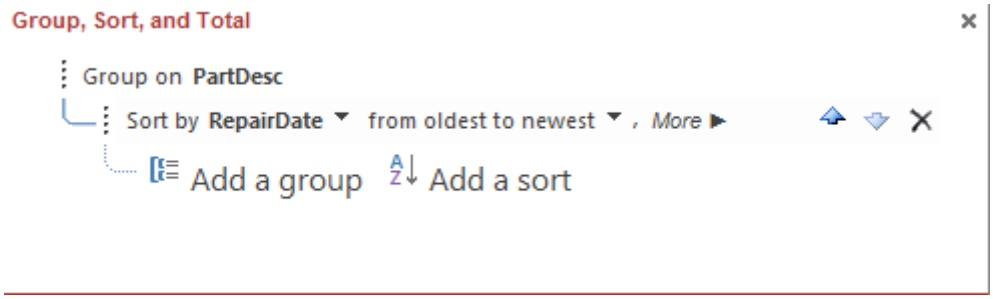


Figure 28: Sorting and Grouping Window after Deletions

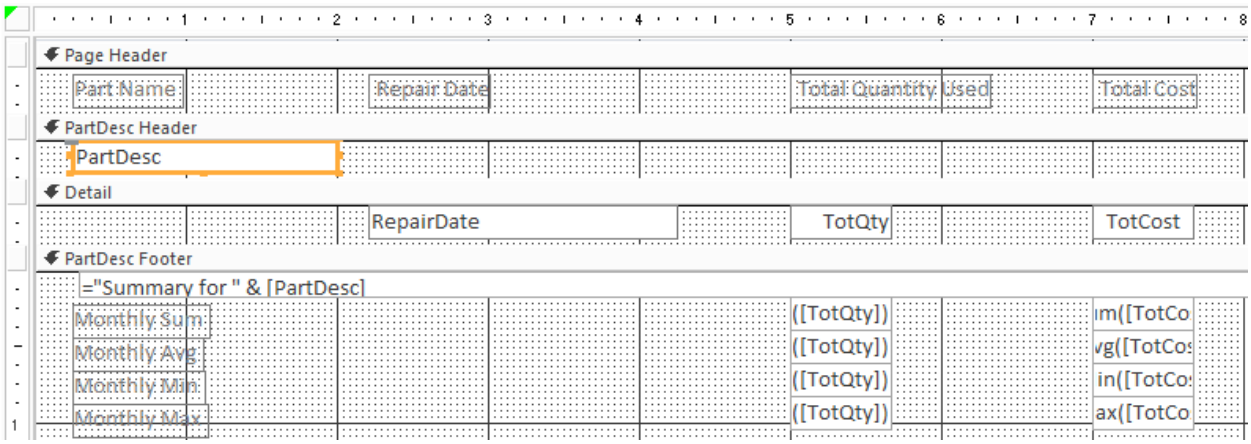


Figure 29: Design View (without gridlines) of *Part Report 4* after Modifications

Auto Repair Shop Part Report for October 2013

Part Name	Repair Date	Total Quantity Used	Total Cost
10W-30 oil			
	10/10/2013 11:53:30 AM	5	$4.75
	10/22/2013 2:40:47 PM	4	$3.80
Summary for 10W-30 oil			
Monthly Sum		9	8.55
Monthly Avg		4.5	4.275
Monthly Min		4	3.8
Monthly Max		5	4.75
10W-40 oil			
	10/11/2013 11:53:32 AM	2	$2.00
	10/12/2013 2:30:00 PM	4	$4.00
	10/20/2013 9:08:00 AM	4	$4.00
Summary for 10W-40 oil			
Monthly Sum		10	10
Monthly Avg		3.333333	3.3333
Monthly Min		2	2
Monthly Max		4	4

Figure 30: Print Preview of *Part Report4* after Modifications

6.4.2 Creating a Crosstab Query

The other kind of query that you will investigate is the crosstab query that provides a spreadsheet appearance. In a crosstab query, the data are presented as a two-dimensional array with one field's values on the rows, another field's values on the columns, and a third field's values in the cells. Crosstab queries are especially useful for summarizing financial and other kinds of data. The need to summarize multidimensional financial data is so important that a specialty known as "online analytical processing" (or OLAP for short) has developed. Textbook Chapter 16 describes online analytic processing as part of data warehouses.

You can create a crosstab query with just one table by using the Crosstab Query Wizard. But to create a crosstab query with more than one table, you must create a separate query first. You will create a crosstab query that summarizes part descriptions and the make of vehicles. Thus, you will be using fields from three tables, the *Part*, the *PartsUsed*, and the *Vehicle* tables. Begin by creating a preliminary query that will be used for the crosstab query.

1. Create a New Query: Choose **Create → Query** in the Other group.
2. Select Tables: From the Show Table window, add the following tables: *Part, PartsUsed*, *RepairOrder*, and *Vehicle*. Close the window.
3. Select Fields: Drag the following fields down to the query grid (Figure 31):
   - *Part*: *PartNo*, *PartDesc*, *UnitPrice*
   - *PartsUsed*: *QtyUsed*
   - *RepairOrder*: No fields (this is only a linking table)
   - *Vehicle*: *Make*, *Model*
4. Add a Calculated Field: Click in the last empty column of the query grid. To open the Expression Builder window, click the right mouse button and select **Build…**. Type the following expression into the Expression Builder window and click **OK** when finished.

Amount: [QtyUsed]*[UnitPrice]

5. Execute the Query: When you execute the query, it should appear as in Figure 32. When you are finished, close and save changes as "RptQuery4" when prompted.

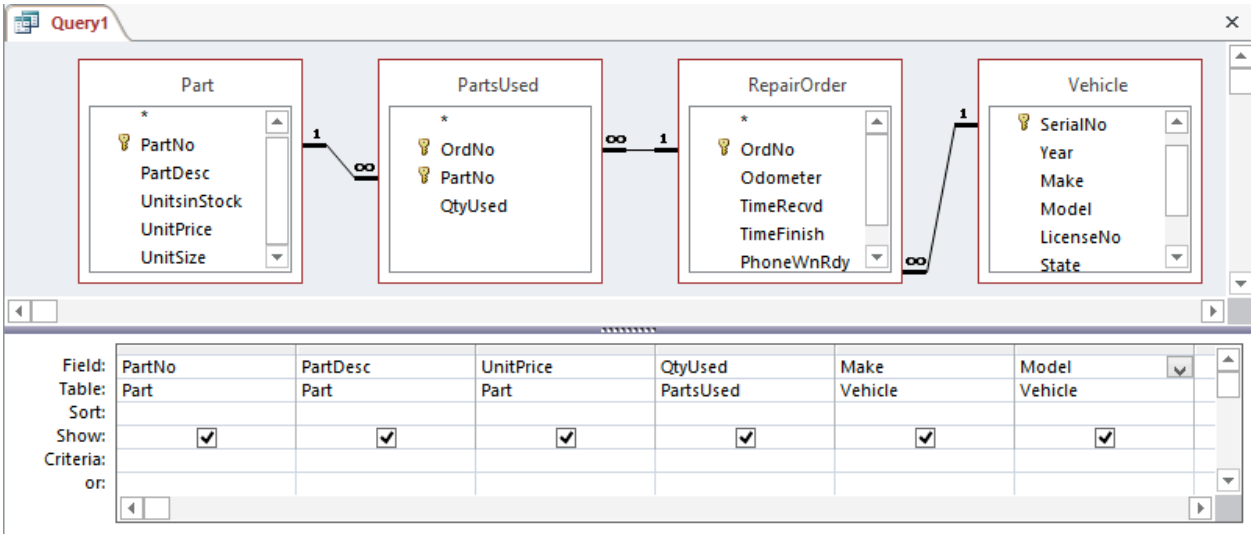
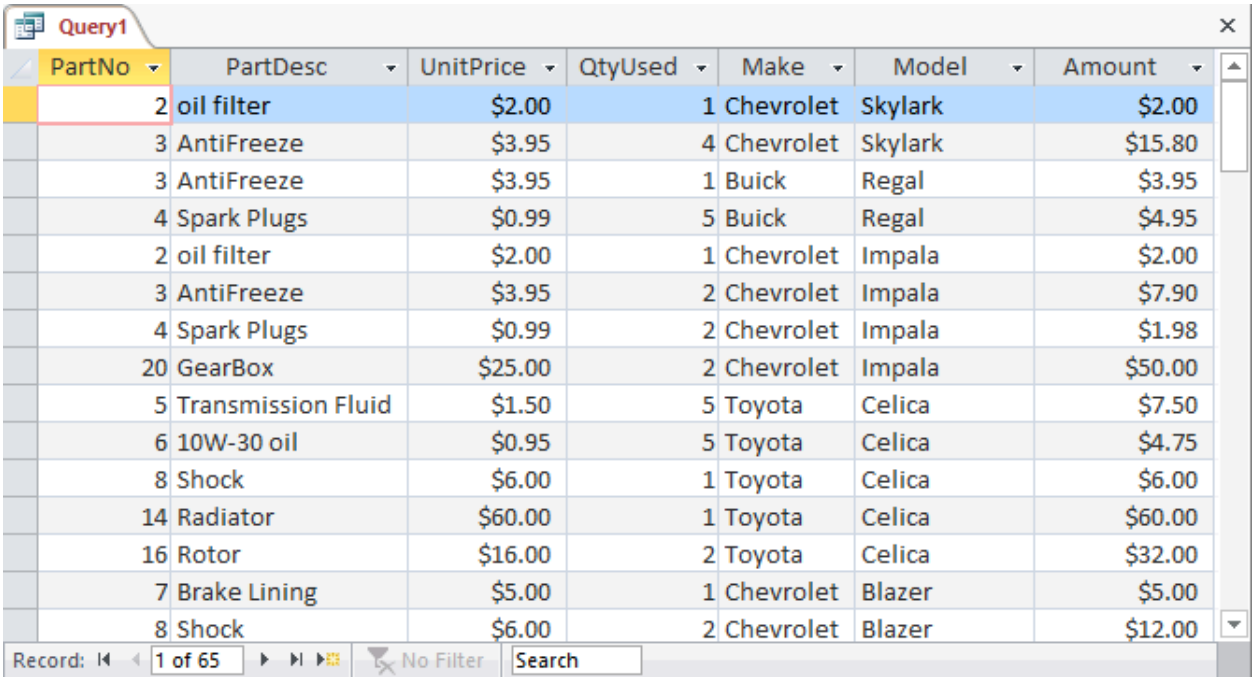


Figure 31: Design View of *RptQuery4*



Query1

PartNo	PartDesc	UnitPrice	QtyUsed	Make	Model	Amount
2	oil filter	$2.00	1	Chevrolet	Skylark	$2.00
3	AntiFreeze	$3.95	4	Chevrolet	Skylark	$15.80
3	AntiFreeze	$3.95	1	Buick	Regal	$3.95
4	Spark Plugs	$0.99	5	Buick	Regal	$4.95
2	oil filter	$2.00	1	Chevrolet	Impala	$2.00
3	AntiFreeze	$3.95	2	Chevrolet	Impala	$7.90
4	Spark Plugs	$0.99	2	Chevrolet	Impala	$1.98
20	GearBox	$25.00	2	Chevrolet	Impala	$50.00
5	Transmission Fluid	$1.50	5	Toyota	Celica	$7.50
6	10W-30 oil	$0.95	5	Toyota	Celica	$4.75
8	Shock	$6.00	1	Toyota	Celica	$6.00
14	Radiator	$60.00	1	Toyota	Celica	$60.00
16	Rotor	$16.00	2	Toyota	Celica	$32.00
7	Brake Lining	$5.00	1	Chevrolet	Blazer	$5.00
8	Shock	$6.00	2	Chevrolet	Blazer	$12.00

Record: 1 of 65 No Filter Search

Figure 32: Datasheet of *RptQuery4*

Create Crosstab Report Query

You will create a crosstab query using the *RptQuery4* that you just created. Follow the instructions below:

1. Open Crosstab Query Wizard: Choose **Create → Query Wizard** in the Other group. In the New Query dialog box, select the Crosstab Query Wizard item. Click the **Ok** button.

2. Select Table or Query: In the first wizard window, select the "Queries" option and choose *RptQuery4* from the list (Figure 33). Click the **Next >** button.

3. Select Fields for Row Headings: In the next window, select *PartDesc* as the field for the row heading (Figure 34). Click the **Next >** button.

4. Select Fields for Column Headings: In the next window, select *Make* as the field for the column heading (Figure 35). Click the **Next >** button.

5. Add Calculated Column: In the next window, modify the expression under the *Make* column. Select "Amount" from the Fields list column (left side) and select "Sum" from the Functions list column (right side). Your expression should appear as "Sum(Amount)" as in Figure 36. Click the **Next** button.

6. Name the Query: In the final window, name the query "Crosstab" and click **Finish**. The query will open in datasheet view (Figure 37). If you are interested, open the query design to see how a crosstab query is specified. When you are finished viewing, close the window.

7. Create Crosstab Report: Select *Crosstab* query from the navigation pane. Choose **Create → Report** in the Reports group. The report will open in Layout preview, but toggle to Design view to make minor modifications. Change the title from "Crosstab" to "Parts Expense Report". Expand the report to the right and align the labels and the textboxes evenly across the report (refer to section 6.3.2 above). When you are finished the report should appear as in Figure 38. Close and save the report as "Parts Expense Report".

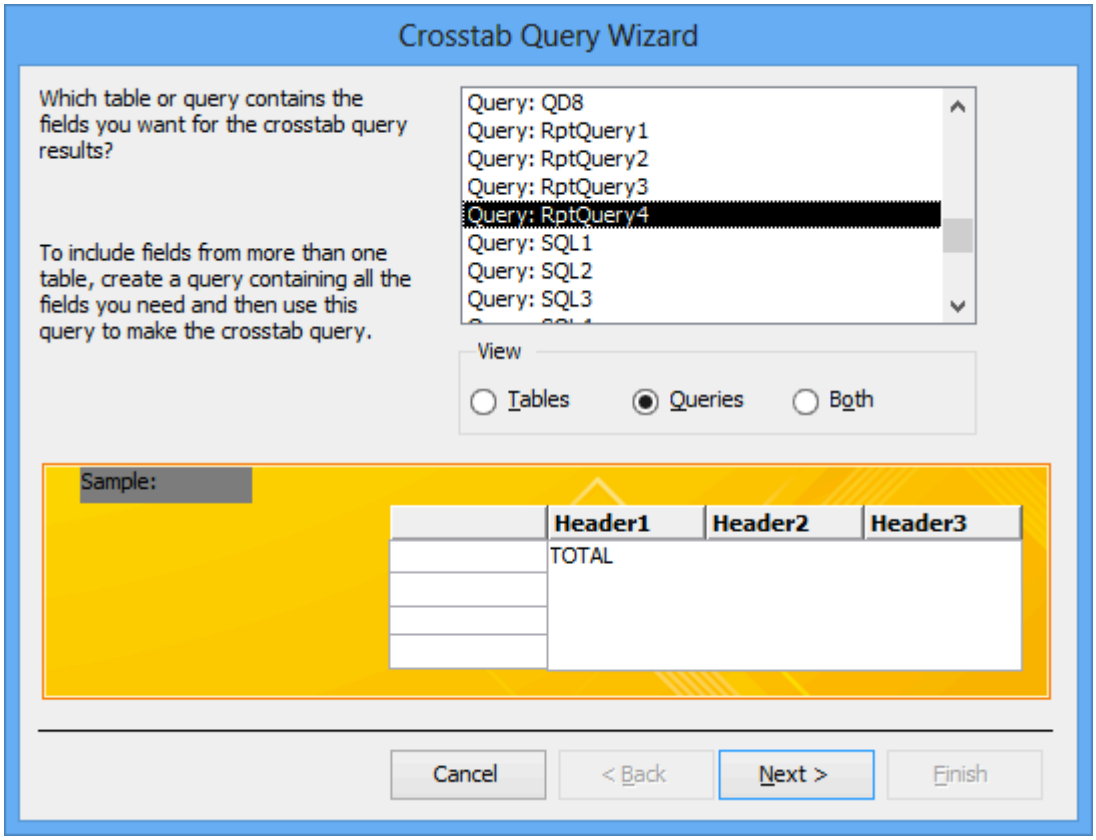


Figure 33: Crosstab Wizard Query List

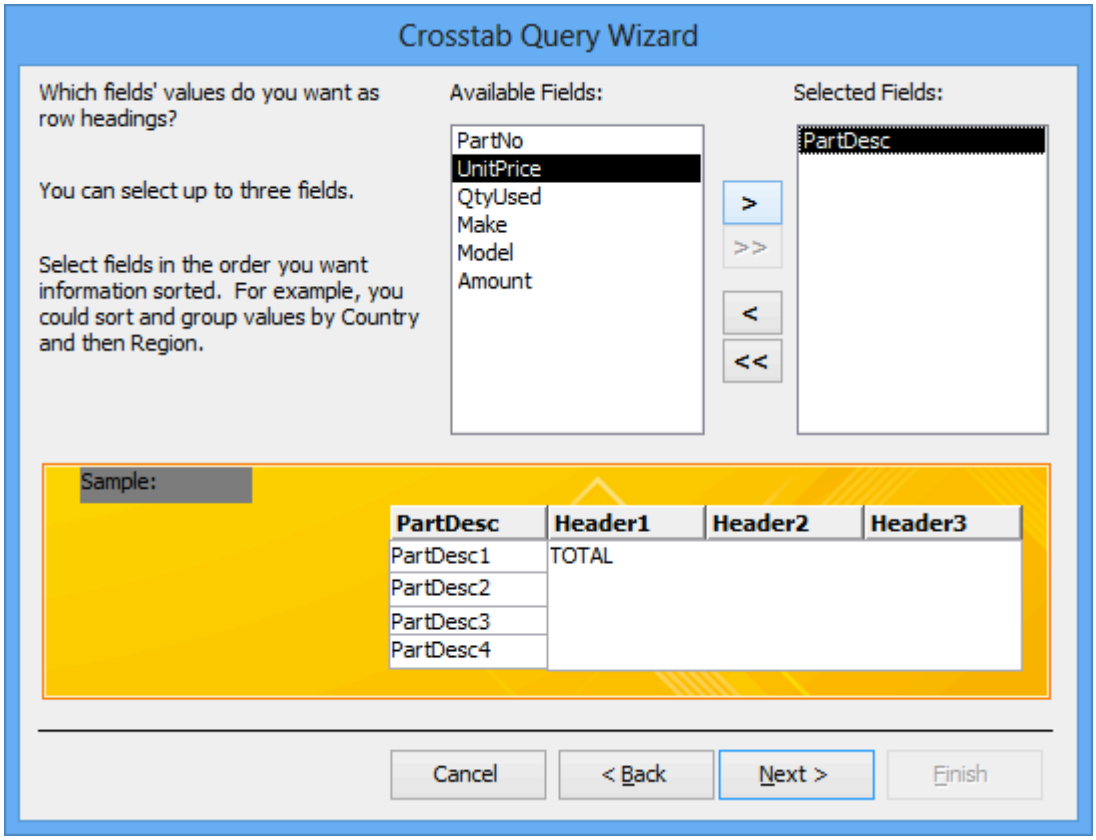


Figure 34: Select *PartDesc* as the Row Heading

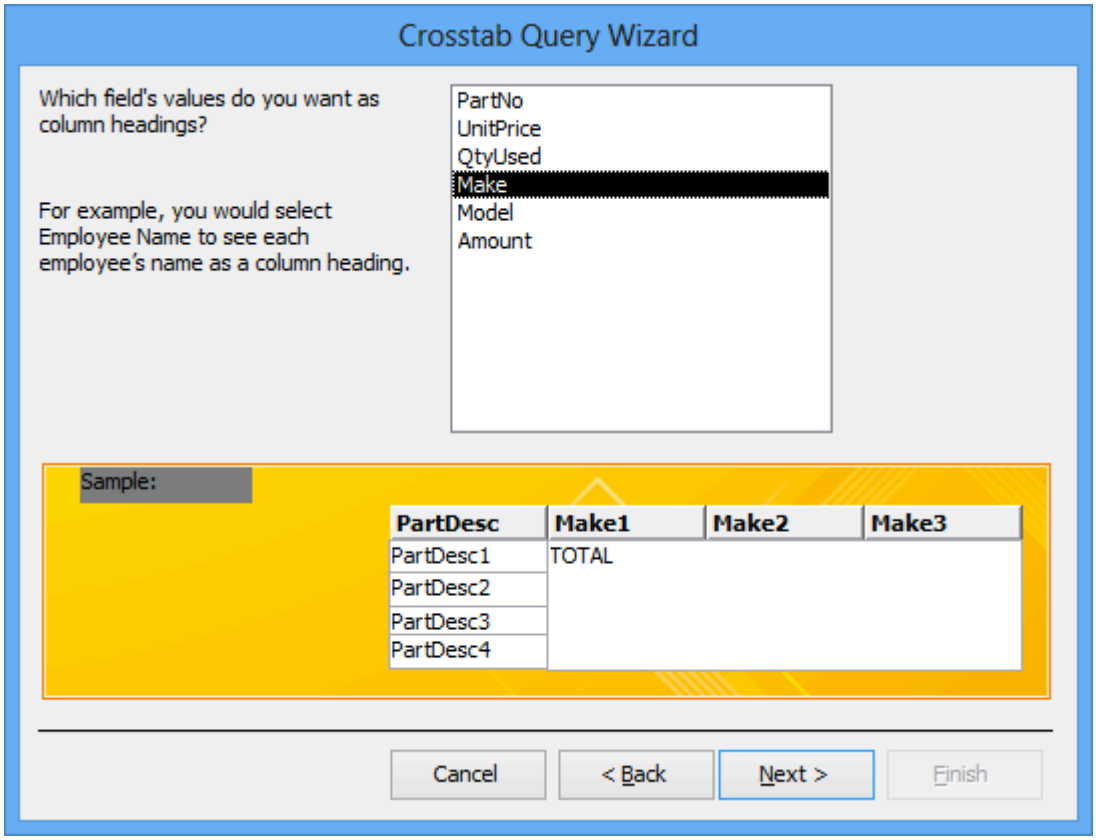


Figure 35: Select *Make* as the Column Heading

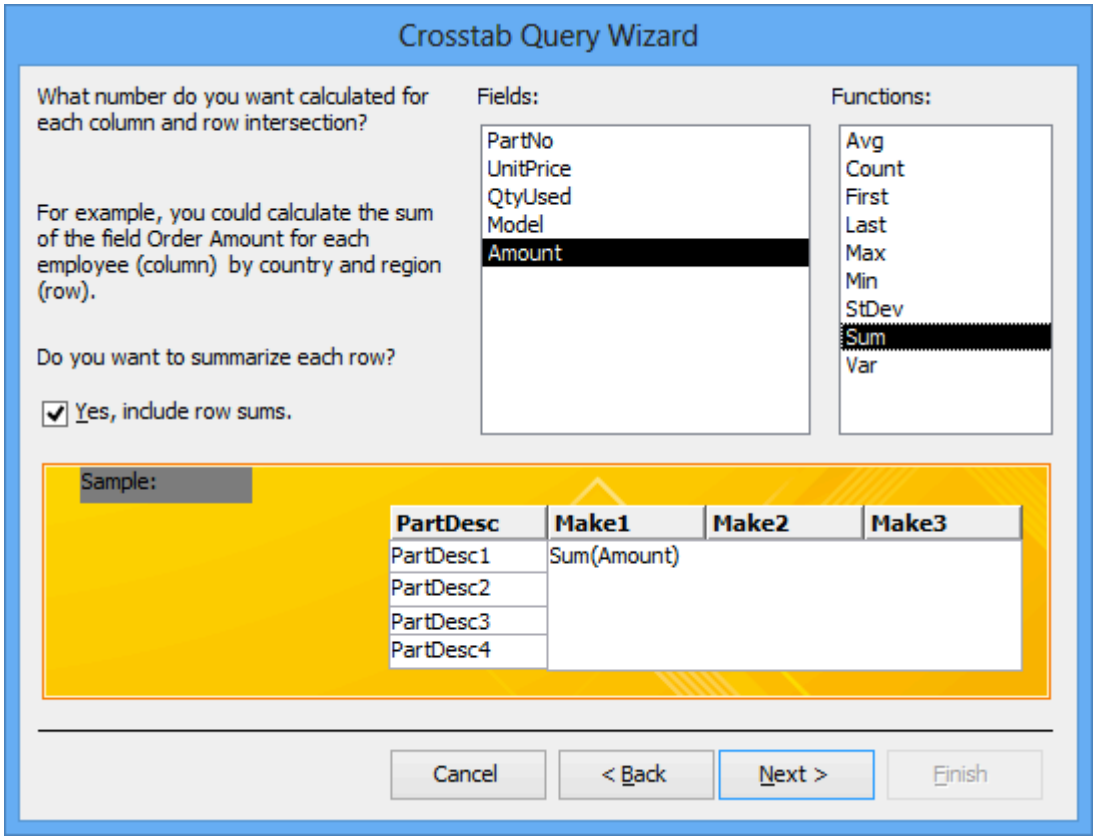


Figure 36: Cell Expression

Crosstab

PartDesc	Total Of Am	BMW	Buick	Chevrolet	Ford	Honda	Lincoln	Pontiac	Toyota
10W-30 oil	$14.25			$5.70				$3.80	$4.7
10W-40 oil	$20.00				$9.00	$5.00			$6.0
AntiFreeze	$79.00		$3.95	$31.60		$7.90	$7.90	$23.70	$3.9
Battery	$110.00						$55.00		$55.0
Brake Lining	$80.00		$20.00	$25.00	$10.00		$20.00	$5.00	
GearBox	$125.00	$50.00		$75.00					
Headlight	$5.00		$5.00						
Muffler	$45.00			$15.00	$15.00				$15.0
oil filter	$10.00			$6.00	$2.00				$2.0
Radiator	$240.00				$60.00			$60.00	$120.0
Radiator Hose	$3.00		$3.00						
Rotor	$96.00	$48.00	$16.00						$32.0
Shock	$48.00			$36.00				$6.00	$6.0
Spark Plugs	$28.71	$1.98	$10.89	$3.96	$3.96	$5.94	$0.99		$0.9
Tail Light	$3.00								$3.0
Tail Pipe	$20.00								$20.0
Tire	$138.00		$138.00						
Transmission Fluid	$10.50			$3.00					$7.5

Record: 1 of 18 No Filter Search

Figure 37: Datasheet of Crosstab Query

Parts Expense Report

Sunday, November 10, 2013
9:21:23 PM

PartDesc	Total Of Amount	BMW	Buick	Chevrolet	Ford	Honda	Lincoln	Pontiac	Toyota
10W-30 oil	$14.25			$5.70				$3.80	$4.75
10W-40 oil	$20.00				$9.00	$5.00			$6.00
AntiFreeze	$79.00		$3.95	$31.60		$7.90	$7.90	$23.70	$3.95
Battery	$110.00						$55.00		$55.00
Brake Lining	$80.00		$20.00	$25.00	$10.00		$20.00	$5.00	
GearBox	$125.00	$50.00		$75.00					
Headlight	$5.00		$5.00						
Muffler	$45.00			$15.00	$15.00				$15.00
oil filter	$10.00			$6.00	$2.00				$2.00
Radiator	$240.00				$60.00			$60.00	$120.00
Radiator Hose	$3.00		$3.00						
Rotor	$96.00	$48.00	$16.00						$32.00
Shock	$48.00			$36.00				$6.00	$6.00
Spark Plugs	$28.71	$1.98	$10.89	$3.96	$3.96	$5.94	$0.99		$0.99
Tail Light	$3.00								$3.00
Tail Pipe	$20.00								$20.00
Tire	$138.00		$138.00						
Transmission Fluid	$10.50			$3.00					$7.50
									276.19

Figure 38: Parts Expense Report Using the Crosstab Query

Grouping Query

Now let's compare the crosstab query with a grouping query. Create a grouping query in SQL view using the *Vehicle*, the *RepairOrder*, the *Part*, and the *PartsUsed* tables as shown below:

```
SELECT Part.PartDesc, Vehicle.Make, sum
 ([QtyUsed]*[UnitPrice]) AS SumAmount
FROM Vehicle INNER JOIN (RepairOrder INNER JOIN
 (Part INNER JOIN PartsUsed
 ON Part.PartNo = PartsUsed.PartNo)
 ON RepairOrder.OrdNo = PartsUsed.OrdNo)
 ON Vehicle.SerialNo = RepairOrder.SerialNo
GROUP BY Part.PartDesc, Vehicle.Make
```

Execute the query and compare its output (Figure 39) to the crosstab query (Figure 37). Note that the crosstab query shows all combinations of *PartDesc* and *Make* while the grouping query shows only those combinations that appear as rows in *PartDesc*. Also note the two-dimensional appearance of the crosstab query. Close the query and save it as "Group" when prompted.

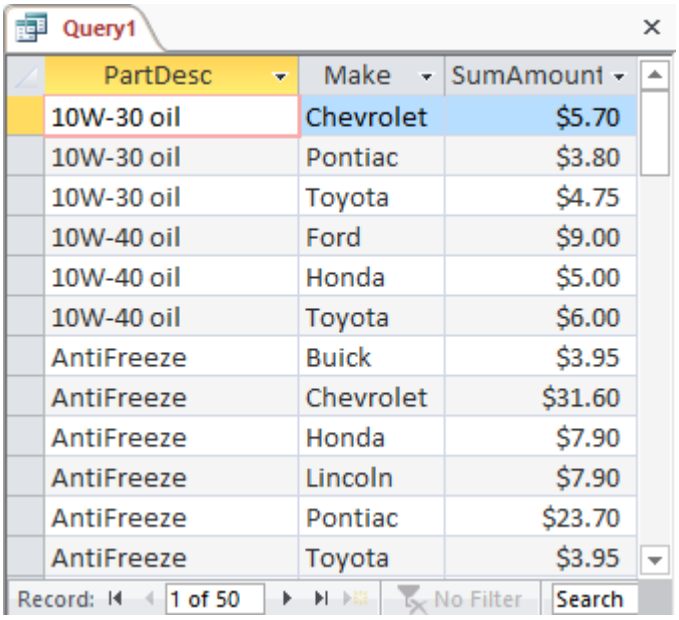


Figure 39: Datasheet Result of Grouping Query

Closing Thoughts

After completing this chapter, you should have a good understanding of reports in Access. You have learned how to create a grouping report using the Report Wizard and how to extend it with multiple-table queries and computed fields. You also have learned how to formulate parameter queries and crosstab queries as well integrate these types of queries into a report.

There are a number of extensions to reports that have not been covered. In reports, like forms, you can embed other Access objects. You can create and display objects such as pictures and charts. You can incorporate subreports to group independently of a parent report. Another useful capability is to create mailing labels and form letters.

Chapter Reference

The chapter reference summarizes procedures that you practiced. For wizards discussed in the chapter, the procedures highlight important parts of the wizards but do not list all of the steps.

Procedure 1: Using the Report Wizard (Sections 6.1.1 and 6.2.2)

107. Choose **Create → Report Wizard** in the Reports group. Select the table or query for the report.
108. Click the >> or > button to move fields shown in the left side (Available Fields) to the right side (Selected Fields).
109. If you want the report also to contain fields from another table or report, then in the same wizard window, go up to the Tables/Queries selection box where the previous table name appears and select another table or query. You then can move additional fields into the Selected Fields area.
110. In the next window, select the table in which to view the report data.
111. To add grouping levels to a report, select the desired fields using the > button. If the

grouping defaults need to be changed, click the **Grouping Options** button at the lower left of the window to change the grouping intervals.

112. If the detail lines should be sorted, select the desired sort order. You also may click the **Summary Options…** button to set summary values for calculations. This opens a window allowing you to choose *Sum*, *Avg*, *Min*, and *Max* for any report columns. Select the **Detail and Summary** option button if you want the detail lines to appear in the report. Select the **Calculate Percentage Total** option button to add percentage calculations.

113. Select the report layout and style. The default choices usually work well. Make sure that the check box is selected to adjust field widths.

114. Type a name for the report and then click the **Finish** button. You may want to open the report in design view to fine-tune the design.

Procedure 2: Toggle between Report Views (Sections 6.1.1)

115. To toggle between the Print Preview and the Design View windows, click the **View** button on the toolbar.

116. The Layout Preview window shows a simplified view of a report to verify a report's layout, not the actual data in the report. Use layout view instead of print preview when your report retrieves a large amount of data.

117. You also can right click on mouse to switch among report views.

Procedure 3: Editing Report Controls (Sections 6.3.2)

118. Select a group of controls: To group controls as a unit for the purpose of formatting or moving, you can either (a) drag a grouping rectangle around them, (b) click on each control while holding down the **Shift** key, or (c) click on each control while holding down the **Shift** key and then click **Arrange → Size/Space → Group** (choose **Ungroup** to separate into individual controls again).

119. To format the group unit: You can change fonts and use bold, italics, and underlining with the toolbar. Your changes will affect all of the selected controls. You also can click the right mouse button to align or size all of the controls at the same time or use the **Arrange** tools under the **Report Design Tools** dynamic part of the Ribbon. To obtain a precise fit for a label's caption or a textbox's data, you can use the **Size/Space → To Fit** feature. The control will automatically adjust to the size of its data if you use this formatting feature.

120. To move controls as a unit: While the unit is still selected, move the mouse until the hand cursor appears. While holding the mouse down, drag the unit to the desired location.

121. Set control properties as a unit: To change the same property for each control in a unit (except individual properties such as *Caption*, *Name*, and *Control Source*), click the right mouse button (while the unit is selected) and select **Properties** to open the Property Sheet. Alternatively, you can click **Design → Property Sheet** in the Tools group. The Property Sheet will be titled "Multiple selection". When you make a property change, it will affect all of the controls in the unit.

122. Change the caption: To change an individual property such as *Name* or *Caption*, you do not need to select each control and individually open its Property Sheet. A faster way is to click on the first control and open its Property Sheet. After changing its individual properties, do not close the window. While the Property Sheet is still open, select the next control and the Property Sheet changes accordingly.

Additional Practice

The following problems provide additional practice with the extended auto repair database as well as the textbook databases.

Part 1: Reports for the Extended Auto Repair Database

1. Create a report that summarizes the labor usage on repair orders. The report is similar to the parts summary reports except that the labor usage is summarized, not part usage. Refer to Figure AP1 as a layout guide for your report. Here are some additional points to help you in creating the report:

   - The query for the report should retrieve the labor usage in the range 10/1/2010 to 11/30/2010. Follow the rules given in textbook Chapter 10 (section 10.5) about how to formulate queries for reports. Note that each detail line in the report involves summary data (averages and counts).
   - The *AvgDev* column in the report uses a computed field: `AVG((RepairLabor.Hours - Labor.StdTime) / Labor.StdTime)`. It is recommended that you perform this calculation in your query rather than in the report.
   - The report should be grouped on labor description and sorted on the date that the repair order was received within each labor description. The fields next to a date summarize the labor usage for that date and labor description.

- The summary for labor description should be the number of repairs and the average of the number of hours and deviation per repair. In addition, the percentage total for the number of repairs is displayed.

Labor Usage Report for 10/1/2013 to 11/30/2013

Labor Description	Repair Date	Number of Repairs	AvgHrs	AvgDev
Bleed Brakes				
	05-Oct-13	1	1.25	25.00%
	18-Oct-13	1	0.5	-50.00%
	22-Oct-13	1	0.9	-10.00%
Summary for Bleed Brakes				
Sum		3		
Avg			0.883333	-11.67%
Percent		11.11%		

Figure AP1: Print Preview of the Labor Usage Report

2. Modify the report in (1) so that the beginning and ending dates are given as parameters rather than hard coded into the report. Change the report's heading to display the parameters in the heading.

Part 2: Reports for the University Database

1. Implement the Faculty Schedule Report as described in section 10.5 of textbook Chapter 10.
2. Implement the Faculty Workload Report as described in section 10.5 of textbook Chapter 10.

Part 3: Reports for the Order Entry Database

1. Implement the Order Detail Report as described in problem (28) of textbook Chapter 10.
2. Implement the Order Summary Report as described in problem (29) of textbook Chapter 10.
3. Implement the revised Order Summary Report as described in problem (30) of textbook Chapter 10.

Appendix A: Tips for Report Design

Here are some tips that may help you in building reports. Most of the difficulties occur in formulating the query. See textbook Chapter 10 (section 10.5) for guidelines about formulating queries for reports. Tips specific to Access are discussed below.

- Formatting controls is rather tedious in report design. As described in Section 6.3.2, you can use the **Align** and the **Size/Space** items under the **Arrange** tab to help. You can select all controls that you want to format together and then use the **Arrange** tab. You also can use the *Text Align* property to specify how data are aligned inside a text box. Right alignment is useful for numeric data. The "General" value for the *Text Align* property works well. It aligns numbers to the right and strings to the left. See the help documentation for more information.
- If your report contains extra pages, you may have a conflict between the report layout and the right-hand margin. To eliminate extra pages, adjust the right-hand margin (it should not longer than 8" for portrait layout). Make sure that all controls fit entirely within the right-hand margin.
- You will find it easier to compute percentages in a report than in a query. In the Report Wizard, you can select percentages for each group footer. Examine the expressions in these group footers to see how percentages are calculated.
- If you write a query with date constants, you need to surround the date constants with pound signs (#). For example, `OrdDate = #10/1/2013#` is a condition to check whether the charge date is October 1, 2013. When using a parameter, the parameter value can be entered with or without the pound signs. For example, `OrdDdate = [Enter Date:]` will cause a parameter dialog for entering a date. You can type the date with or without pound signs.
- You can write an expression in either the *Control Source* property or in the SELECT clause to take one action when there is no value for a field and another action when there is a value. This kind of action can be useful for optional foreign keys. You need to find a function that works like a conditional statement and another function that can tell whether there is a value for the foreign key. If you use these functions in the *Control Source* property, the expression builder is a big help. Unfortunately the expression builder cannot be used to help define an expression in SQL.
- The *Format* property for textboxes has special syntax to deal with null values and zero-length strings. See the help documentation for details.
- The *Format$* function can be used to display dates in a specific format (such as the medium date format). See the help documentation for details. Use the *Format$* function for expressions in textboxes when a date should be displayed along with other text.

Chapter 7: Pivot Tables and Specialized Access Forms

Learning Objectives

This chapter covers features that allow you to use the more specialized objects of Excel and Access 2013. At the completion of this chapter, you should have acquired the knowledge and skills to:

- Understand the relationship between pivot tables and data cubes
- Understand the process to design a pivot table in Excel 2013
- Import an Access query and create a pivot table in Excel 2013 and
- Use filters to customize pivot tables in Excel 2013

- View a pivot table as a pivot chart in Excel 2013
- Create a split form and understand when it is appropriate in Access 2013
- Create a multiple items form and understand when it is appropriate in Access 2013

Overview

In other lab chapters you learned how to create the objects of a database application system. You created tables, queries, forms, and reports. This chapter shows you how to build and customize a few specialized objects. Pivot tables and charts are no longer supported by Access 2013. Instead, Excel 2013 can be used to create pivot tables and charts. You will create a pivot table to demonstrate how Excel supports decision making with multidimensional data. Pivot tables provide a more flexible interface for multidimensional data than crosstab queries presented in Chapter 6. The second part of the chapter provides practice with creating two specialized kind of forms, split forms and multiple item forms. These form types are easy to create and have some important but specialized uses so you should become familiar with them.

7.1 Pivot Tables

Data warehouse processing requires a multidimensional data representation that is convenient for business analysts. Data cubes, a widely accepted representation of multidimensional data, support operations for business intelligence processing by analysts. In a data cube, cells contain numeric data called measures while rows and columns contain dimensions to organize the cells. Textbook Chapters 16 and 17 provide detailed conceptual material about data cubes and other aspects of data warehouses.

In Excel, pivot tables and pivot charts provide a convenient and flexible interface for manipulating data cubes. Pivot tables display data in rows and columns and allow convenient rearrangement of the row and column headings. Pivot charts display numerical data graphically to provide insights to business analysts. Pivot tables support dynamic manipulation of the row and column headings as compared to the static representation of crosstab queries presented in Chapter 6. Pivot tables were first available in Access 2000, but Excel 2000 was needed to use them. Since Access 2002 until Access 2010, Excel was not needed to use pivot tables. Pivot charts were new in Access 2002. However, pivot tables and charts were discontinued in Access 2013. Pivot tables and charts can be created using Excel 2013 instead.

This section provides practice with pivot tables and pivot charts. Because the terminology of pivot tables is somewhat different than the terminology of data cubes in textbook Chapter 16, the first section presents background on pivot tables and pivot charts. The second section provides practice with creating pivot tables and viewing a pivot table as a pivot chart. The third section extends the pivot table presented in the second section and discusses additional pivot table features.

7.1.1 Background about Pivot Tables

To explain the Excel terminology, Figure 1 presents a pivot table for parts used data along with annotations to depict the terminology. Table 1 provides explanations for the annotated parts of Figure 1 along with some additional terms.

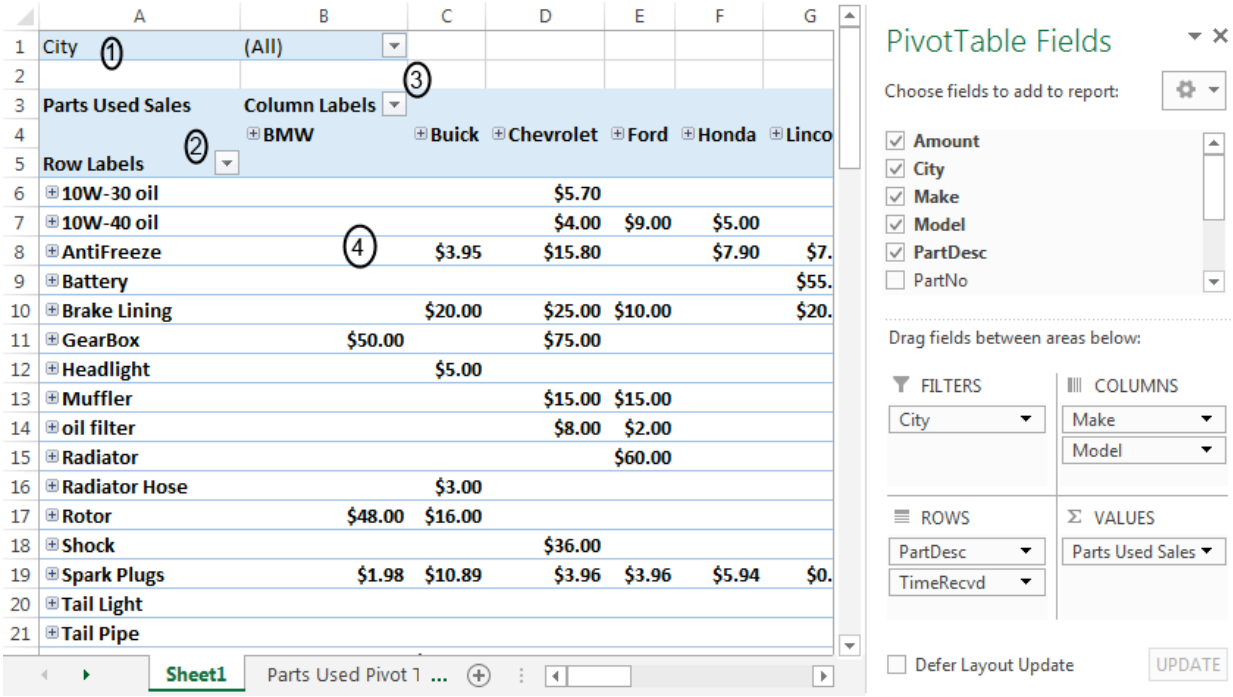


Figure 1: Example Pivot Table

Table 1: Excel Terminology for Pivot Tables

Term	**Meaning**
Filters Field (1)	Restricts data that appears in the pivot table. *City* is a filter field. The user can choose to display pivot table data for selected cities by choosing a value for the *City* filter field.
Rows Field (2)	A field appearing in the row area of a pivot table. *PartDesc* and *TimeRecvd* are row fields.
Columns Field (3)	A field appearing in the column area of a pivot table. *Make* and *Model* are column fields.
Values Field (4)	A field appearing in the cell area of a pivot table. The numeric values in the cells are the sum of the parts used for the combination of the row and the column values. For example, $50 is the sales for gear boxes repaired in BMWs.
Field drop-down arrow	The arrow at the right side of each field. The user can select values to display or hide by clicking the arrow.
Item	A data value within a field. Expanding a field displays its items, while collapsing a field hides its items. In Figure 1, *PartDesc* and *Make* are expanded, while *TimeRecvd* and *Model* are collapsed.

Beyond understanding the Excel terminology, you should understand the correspondence between data cubes as presented in textbook Chapter 16 and pivot tables. Table 2 summarizes the correspondence between the terminologies. If you are unfamiliar with the data cube terminology, you should review textbook Section 16.2.

Table 2: Terminology Correspodence between Data Cubes and Excel Pivot Tables

Data Cube Term	**Excel Pivot Table Term**
Dimension	Row field, Column field
Measure, Measure value	Value field, Value item
Dimension hierarchy	Custom groups of a row field or column field. Date/Time fields have predefined groups. Dimension hierarchies can also be implemented using adjacent row or column fields.
Member	Row item, Column item

Before creating a pivot table, you should carefully plan its components and formulate a query (in Access) to retrieve the data contained in the pivot table (in Excel). In date cube terminology, you should determine the dimensions, dimension hierarchies, and measures. For each measure, you should determine whether it is a basic or a derived measure. For each derived measure, you should determine whether it is a summary value (computed from groups of rows) or a row value (computed from individual rows). Pivot tables containing row data are more flexible than pivot tables with only summary data although row data consumes more space than summary data. After determining the components of a pivot table, you should formulate the query to derive the data used in the pivot table. The query should contain all fields of the pivot table although derived measures can be calculated by either the pivot table or by the query. Beyond these general guidelines, here are some specific points to consider when designing pivot tables:

- With pivot tables, an important way to represent dimension hierarchies is using adjacent row or column fields. Typically, geographic dimension hierarchies such as country/region/state/city can be represented as adjacent fields.
- If the pivot table only contains summary measures, the underlying query can contain a GROUP BY clause and aggregate functions to compute the summary measures. However, defining the underlying query to retrieve row data instead of summary data provides more flexibility for modifications to the pivot table.
- If the pivot table contains at least one row measure, the underlying query should not have a GROUP BY clause. In addition, summary measures are computed by the pivot table, not by the query.

7.1.2 Creating Pivot Tables

Before discussing the steps to create pivot tables, the design of the pivot table example is presented. The pivot table should contain part descriptions and the makes and models of vehicles as dimensions. For measures, the amount paid for the repair along with the sum of the amount paid are the measures. For comparison purposes with the crosstab query in Chapter 6, we use *RptQuery4* (shown in Figures 2 and 3) as the underlying query for the pivot table. The next section extends this simple pivot table design with additional fields and other pivot table features.

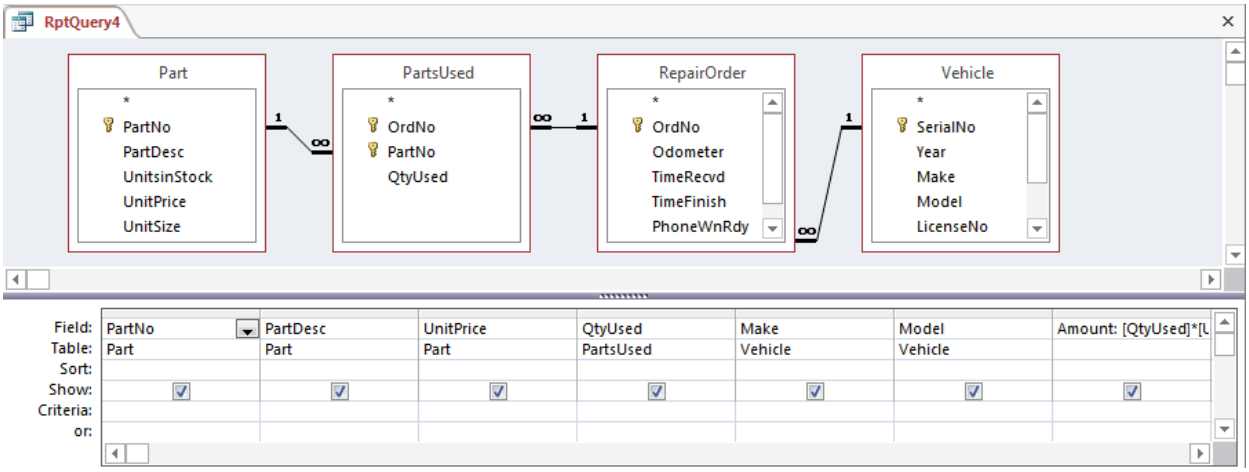


Figure 2: Design View of *RptQuery4*

RptQuery4

PartNo	PartDesc	UnitPrice	QtyUsed	Make	Model	Amount
2	oil filter	$2.00	2	Chevrolet	Skylark	$4.00
3	AntiFreeze	$3.95	1	Buick	Regal	$3.95
4	Spark Plugs	$0.99	5	Buick	Regal	$4.95
2	oil filter	$2.00	1	Chevrolet	Impala	$2.00
3	AntiFreeze	$3.95	2	Chevrolet	Impala	$7.90
4	Spark Plugs	$0.99	2	Chevrolet	Impala	$1.98
20	GearBox	$25.00	2	Chevrolet	Impala	$50.00
5	Transmission Fluid	$1.50	5	Toyota	Celica	$7.50
6	10W-30 oil	$0.95	5	Toyota	Celica	$4.75
8	Shock	$6.00	1	Toyota	Celica	$6.00
14	Radiator	$60.00	1	Toyota	Celica	$60.00
16	Rotor	$16.00	2	Toyota	Celica	$32.00
7	Brake Lining	$5.00	1	Chevrolet	Blazer	$5.00

Record: 1 of 65 No Filter Search

Figure 3: Datasheet of *RptQuery4*

The following steps demonstrate how to create a pivot table in Excel 2013. After creating the pivot table, you will be instructed to modify it and manipulate its data.

1. Import Access Query to Excel: Make sure that the Access database is not running before you start. Open a new excel file. Choose **DATA → Get External Data → From Access** (Figure 4). Navigate to where the database is located, select the database and click **Open**. A dialogue will appear showing the list of all tables, queries, reports and forms in the selected database. Select RptQuery4 and click **Ok** (Figure 5). Then, another dialogue will appear asking about how to view the imported query. Select PivotTable Report and put it in a new worksheet (Figure 6). If the database is open, the data link properties window will appear as in Figure 7. When you click on the test connection, an error message will appear as in Figure 8. Close the database and try again.

2. View Field List: A new sheet opens with a Field List window containing the fields necessary for laying out your pivot table (Figure 9).

3. Place Fields on the Diagram: To place the fields onto the diagram, you may drag them from the Field List window at the top into the designated areas at the bottom (Figure 10). Until you are familiar with the design window's labeled areas, it is suggested that you use the area choices as follows: place the

PartDesc field in the rows area (Figure 11), the *Make* and *Model* fields in the columns area (Figure 12), and the *Amount* field in the values area (Figure 13).

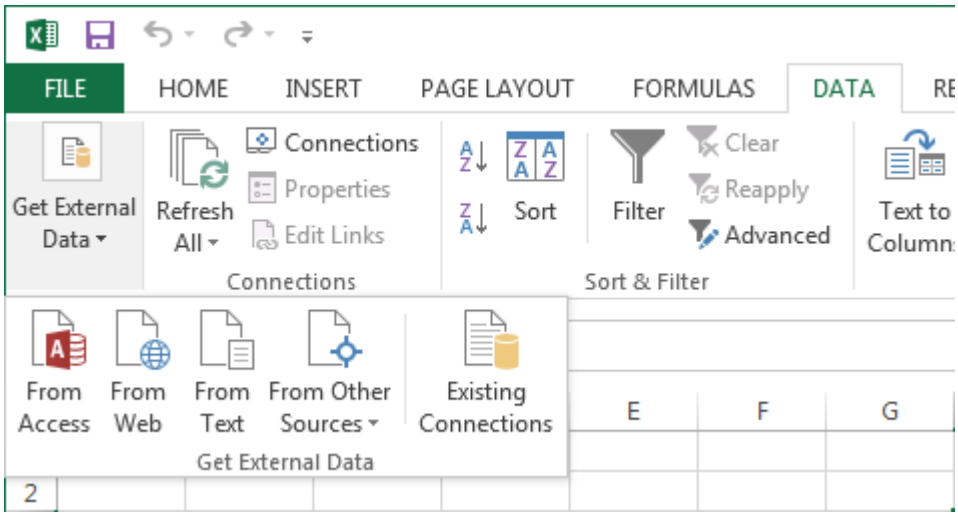


Figure 4: Import from Access Database

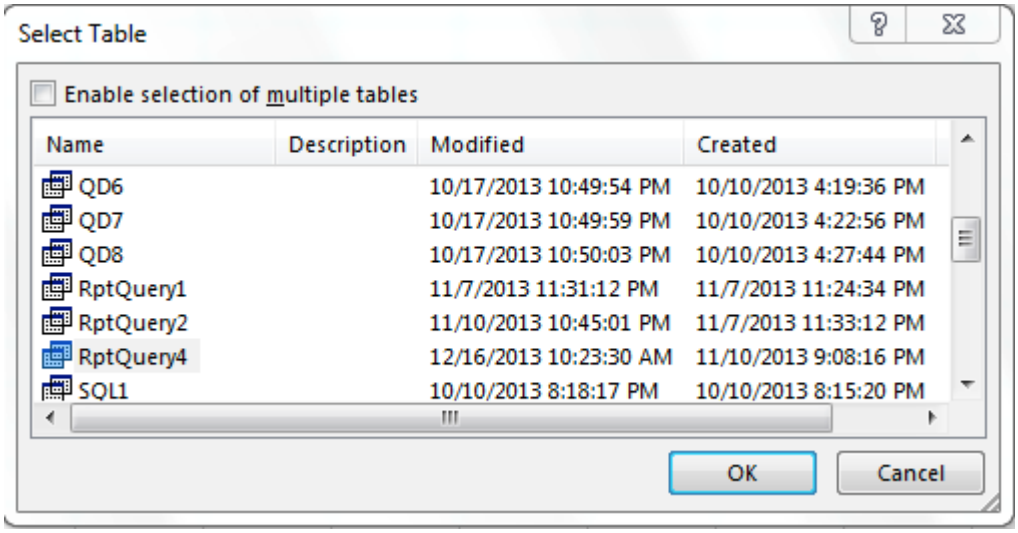


Figure 5: Table Selection Dialogue

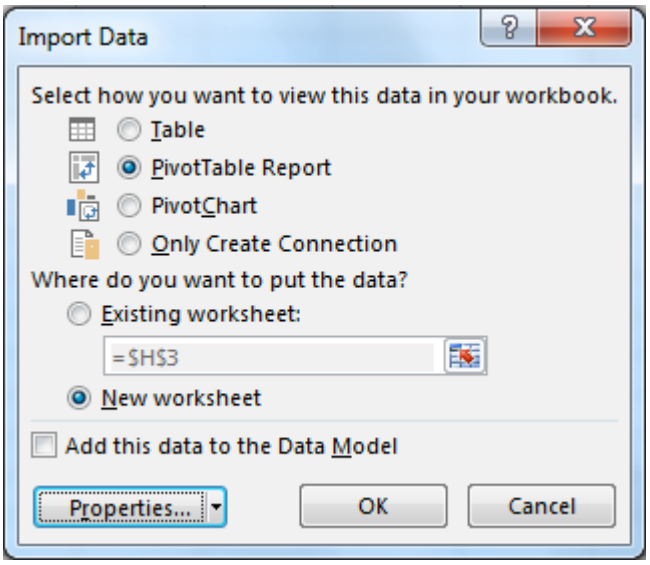


Figure 6: Import Data Dialogue

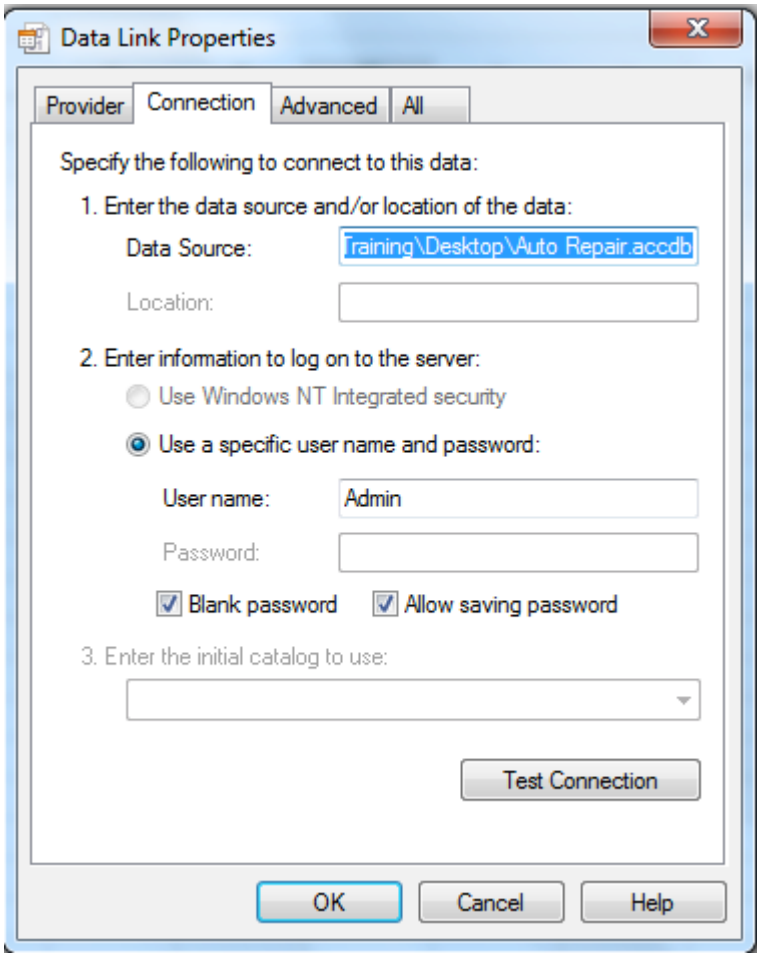


Figure 7: Data Link Properties

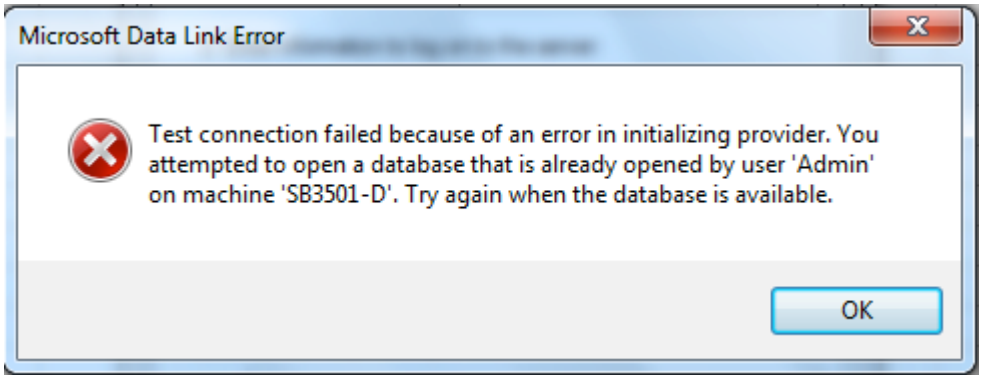


Figure 8: Data Link Error

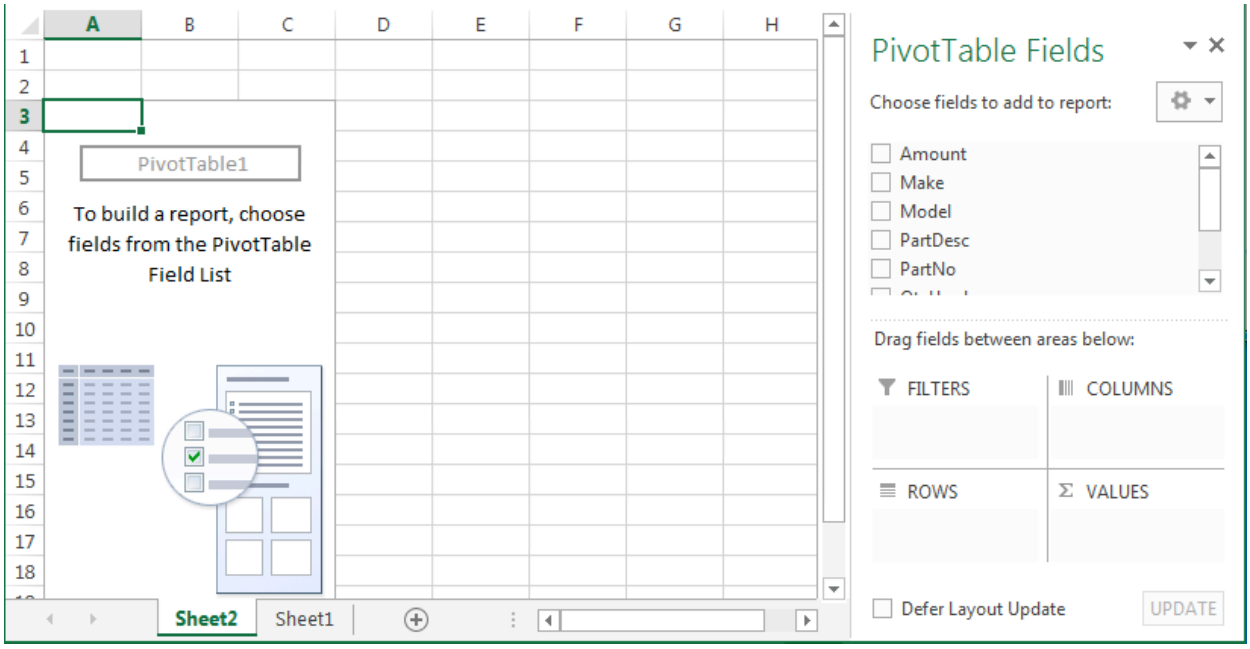


Figure 9: Datasheet with PivotTable Fields

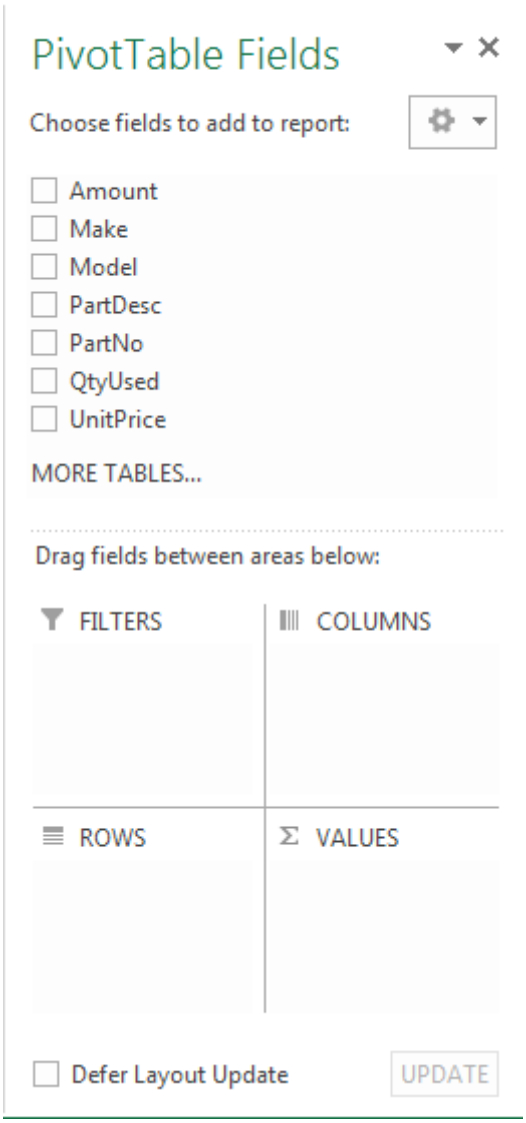


Figure 10: PivotTable Fields

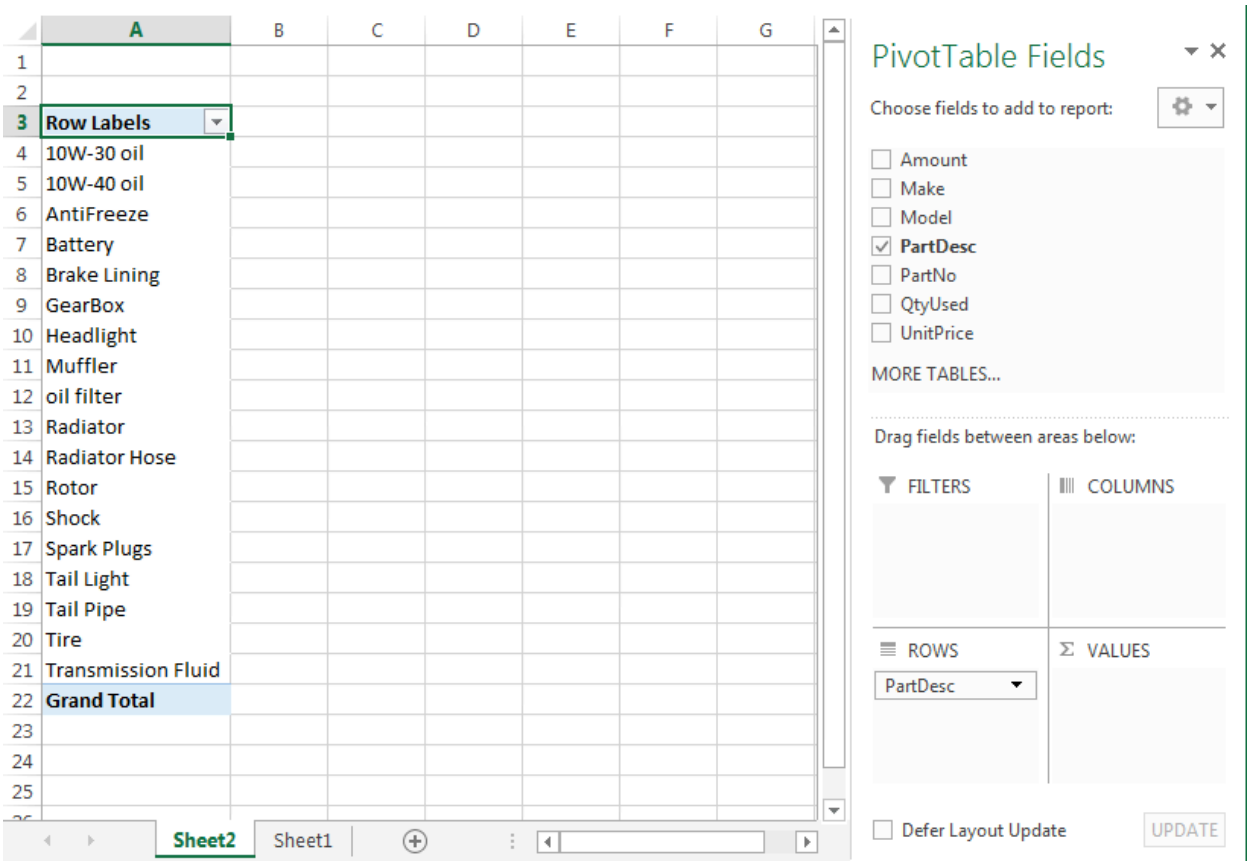


Figure 11: PartDesc Field after Adding it in the Row Area

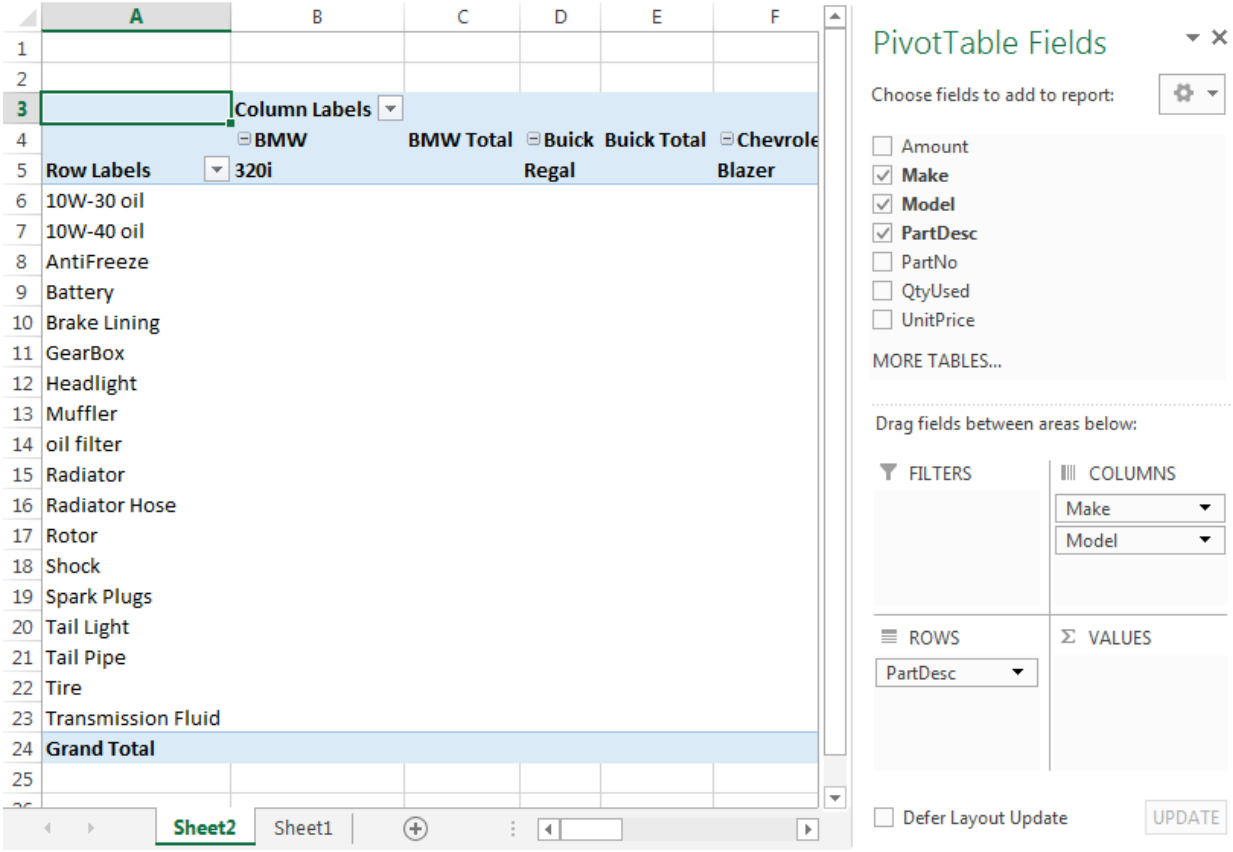
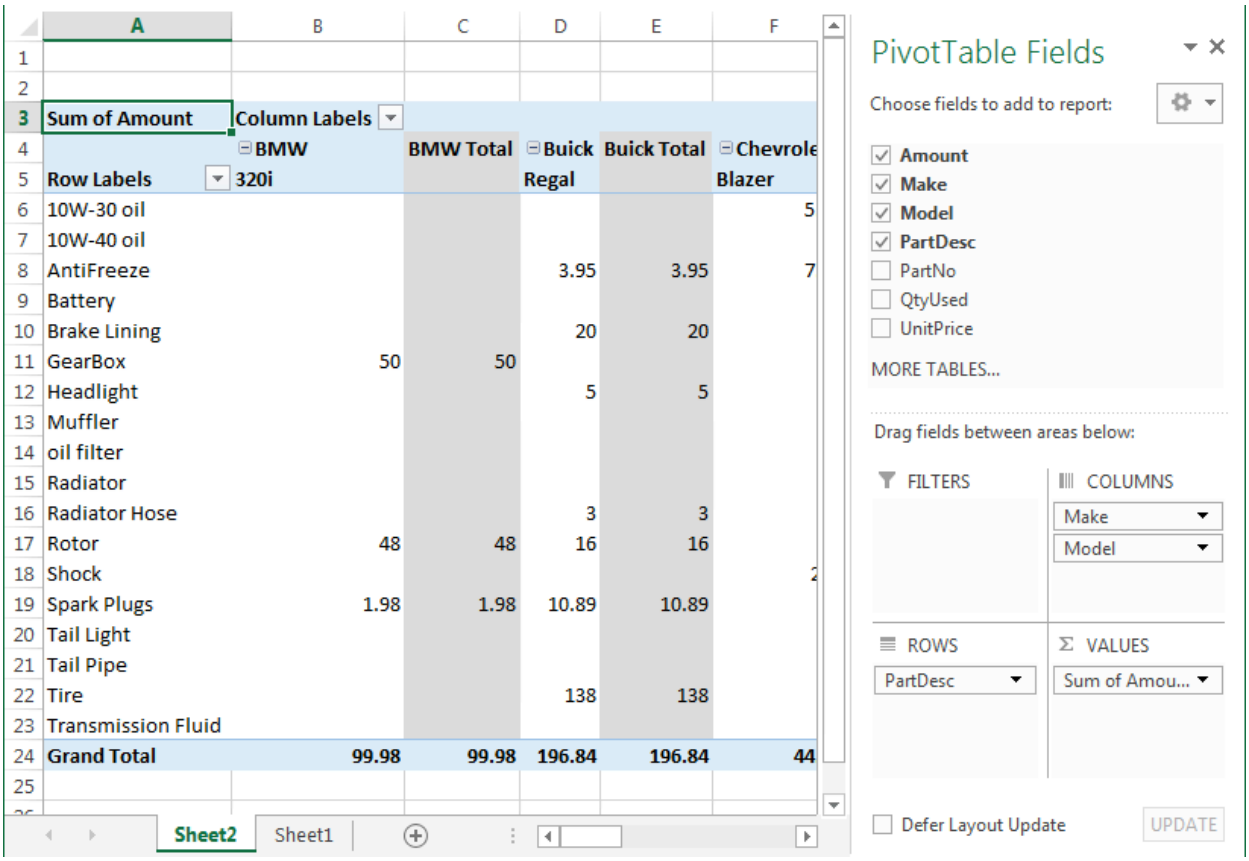


Figure 12: Make and Model Fields after Adding them in the Column Area



Sum of Amount	Column Labels				
	BMW	BMW Total	Buick	Buick Total	Chevrole
Row Labels	320i		Regal		Blazer
10W-30 oil					5
10W-40 oil					
AntiFreeze			3.95	3.95	7
Battery					
Brake Lining			20	20	
GearBox	50	50			
Headlight			5	5	
Muffler					
oil filter					
Radiator					
Radiator Hose			3	3	
Rotor	48	48	16	16	
Shock					2
Spark Plugs	1.98	1.98	10.89	10.89	
Tail Light					
Tail Pipe					
Tire			138	138	
Transmission Fluid					
Grand Total	99.98	99.98	196.84	196.84	44

Figure 13: Amount Field after Adding it in the Value Area

4. Format *Sum of Amount* field: To change the properties for the Sum of Amount field, click on the field title ("Sum of Amount") in the values area. Then, choose the **Value Field Settings** option (Figure 14). A new window opens as in Figure 15. In the **Custom Name** area, change the name to "Parts Used Sales" (Figure 16). In the **Number Format**, change the Numbers to "Currency" (Figure 17) if there is another format specified. Click Ok when finished. In the diagram, the *Sum of Amount* field has changed to *Parts Used Sales* (Figure 18).

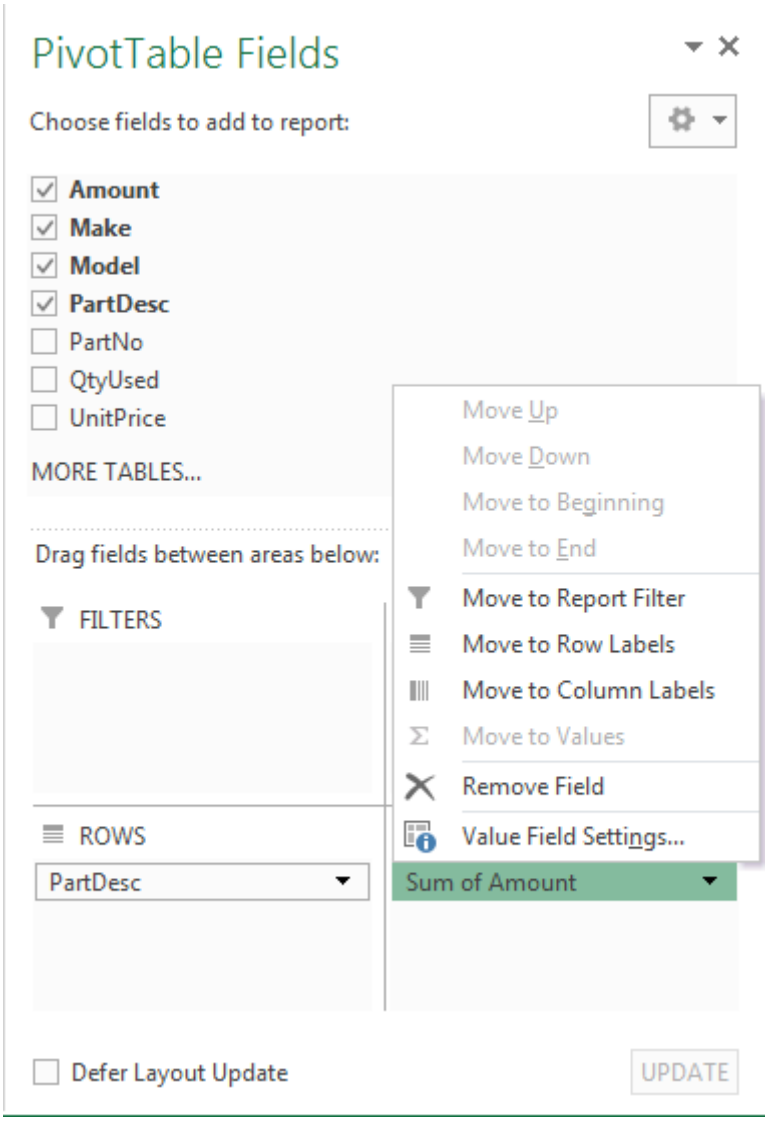


Figure 14: Value Field Settings Option

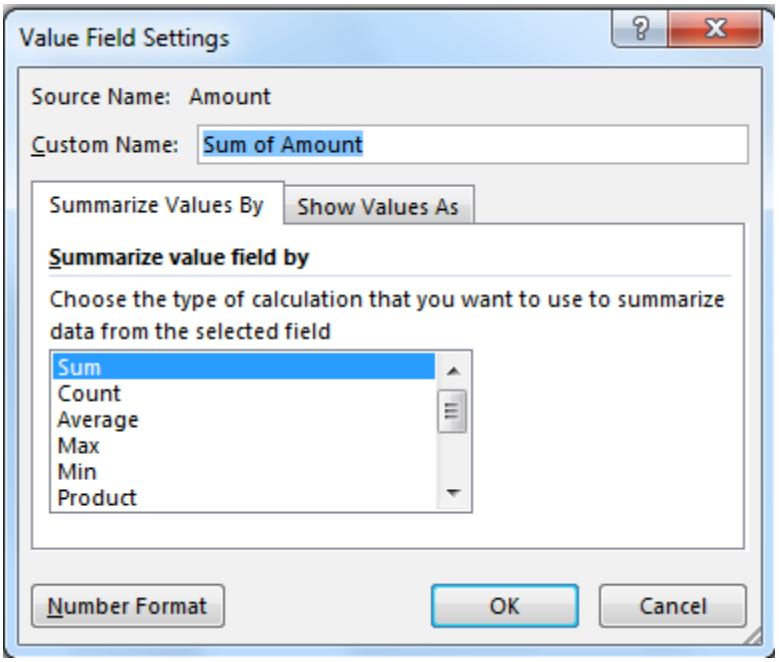


Figure 15: Value Field Settings Window

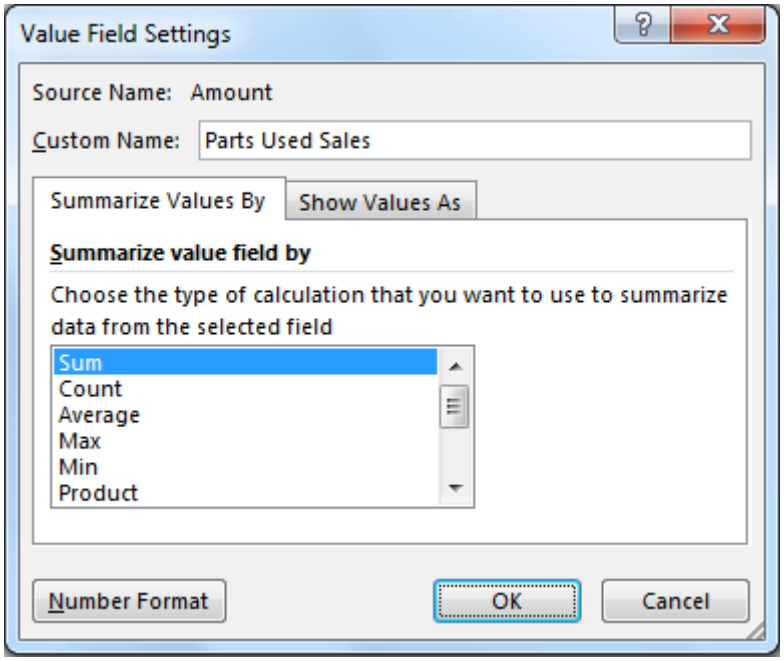


Figure 16: Changed Title Field Name

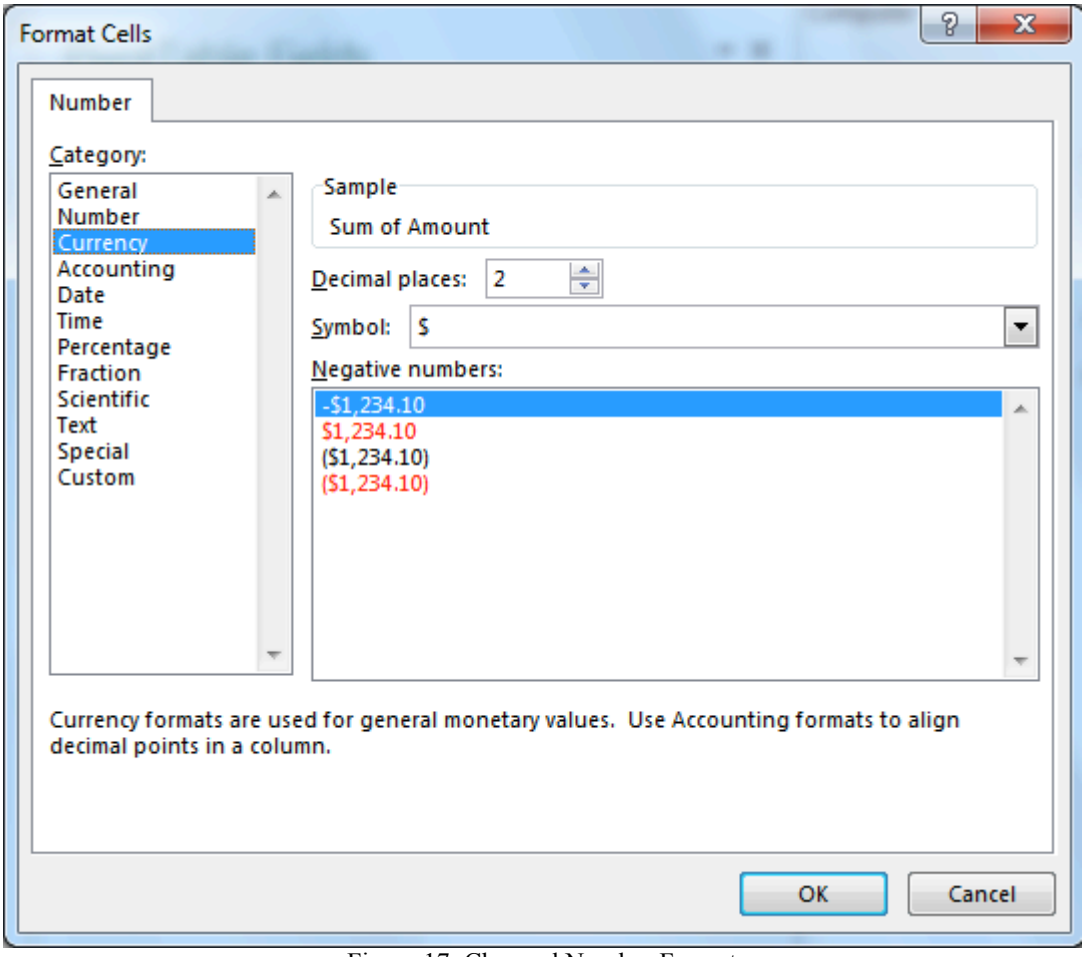
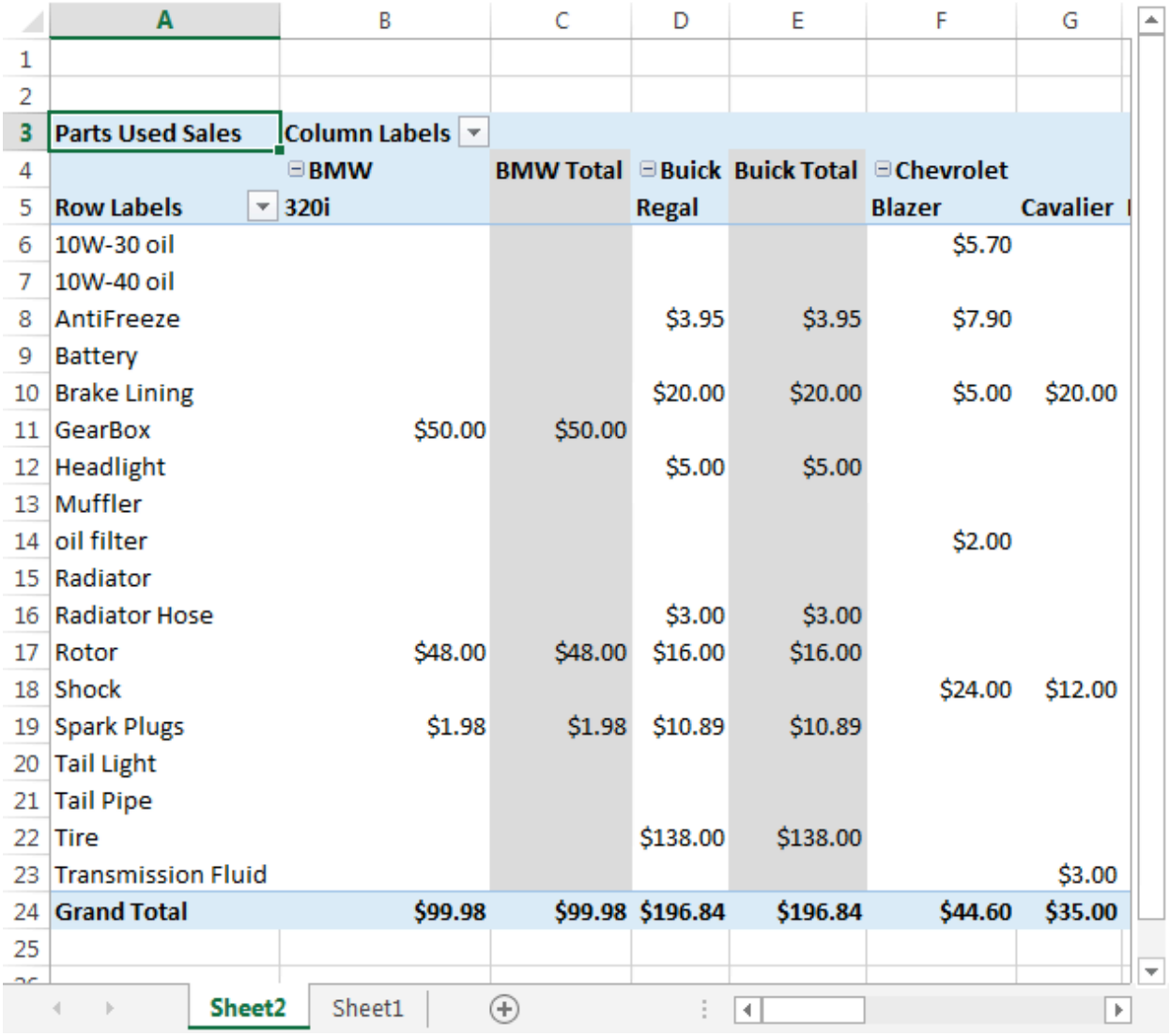


Figure 17: Changed Number Format

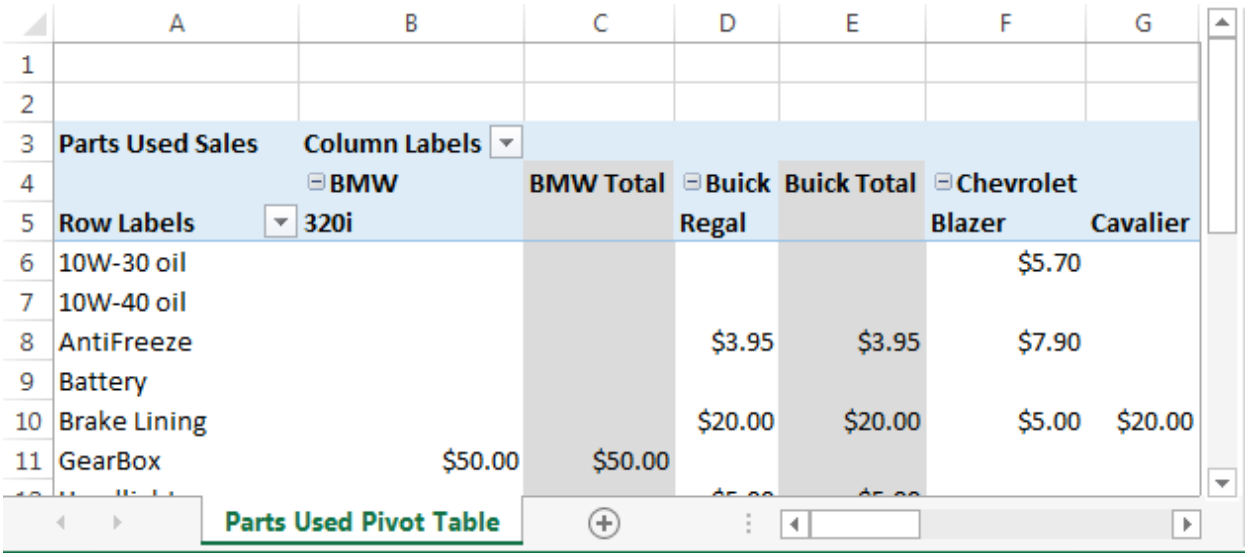


Parts Used Sales	Column Labels					
	BMW	BMW Total	Buick	Buick Total	Chevrolet	
Row Labels	320i		Regal		Blazer	Cavalier
10W-30 oil					$5.70	
10W-40 oil						
AntiFreeze			$3.95	$3.95	$7.90	
Battery						
Brake Lining			$20.00	$20.00	$5.00	$20.00
GearBox	$50.00	$50.00				
Headlight			$5.00	$5.00		
Muffler						
oil filter					$2.00	
Radiator						
Radiator Hose			$3.00	$3.00		
Rotor	$48.00	$48.00	$16.00	$16.00		
Shock					$24.00	$12.00
Spark Plugs	$1.98	$1.98	$10.89	$10.89		
Tail Light						
Tail Pipe						
Tire			$138.00	$138.00		
Transmission Fluid						$3.00
Grand Total	**$99.98**	**$99.98**	**$196.84**	**$196.84**	**$44.60**	**$35.00**

Figure 18: Title Changed in the Pivot Table Datasheet

5. Modify the Pivot Table Sheet: To modify the pivot table properties, right click anywhere in the pivot table and select **PivotTable Options** from the menu. Change the name to "Parts Used Pivot Table". To rename the sheet, right click on ("Sheet2") and select **Rename** from the option list. Rename the sheet to "Parts Used Pivot Table" and click Enter (Figure 19). Also, right click on ("Sheet1") and select Delete. Save the Excel file as "PartsUsedPivotTable".

6. View the Pivot Table as a Pivot Chart: To view the pivot table as a pivot chart, you need to choose **Insert → PivotChart** from the Charts section in the Ribbon (Figure 20). A new window opens with chart options (Figure 21). Select **Column** and click Ok. The pivot chart will show up as in Figure 22. Save the changes when you are finished. For more details about pivot charts, you are encouraged to study the Excel help documentation.



	A	B	C	D	E	F	G
1							
2							
3	Parts Used Sales	Column Labels					
4		BMW	BMW Total	Buick	Buick Total	Chevrolet	
5	Row Labels	320i		Regal		Blazer	Cavalier
6	10W-30 oil					$5.70	
7	10W-40 oil						
8	AntiFreeze			$3.95	$3.95	$7.90	
9	Battery						
10	Brake Lining			$20.00	$20.00	$5.00	$20.00
11	GearBox	$50.00	$50.00				

Figure 19: Pivot Table Sheet

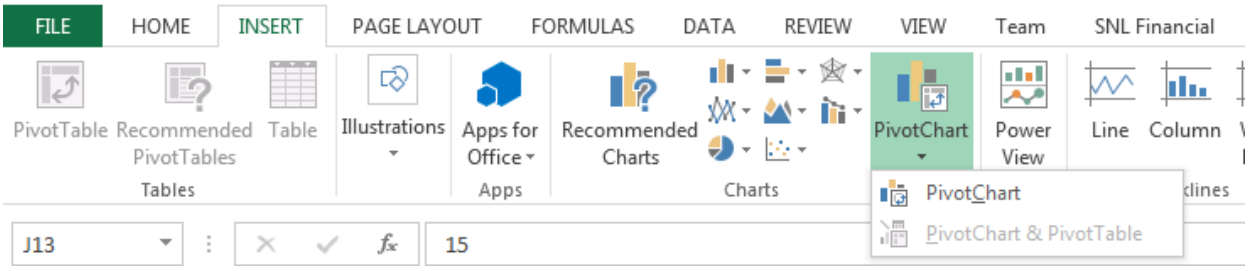


Figure 20: PivotChart Option

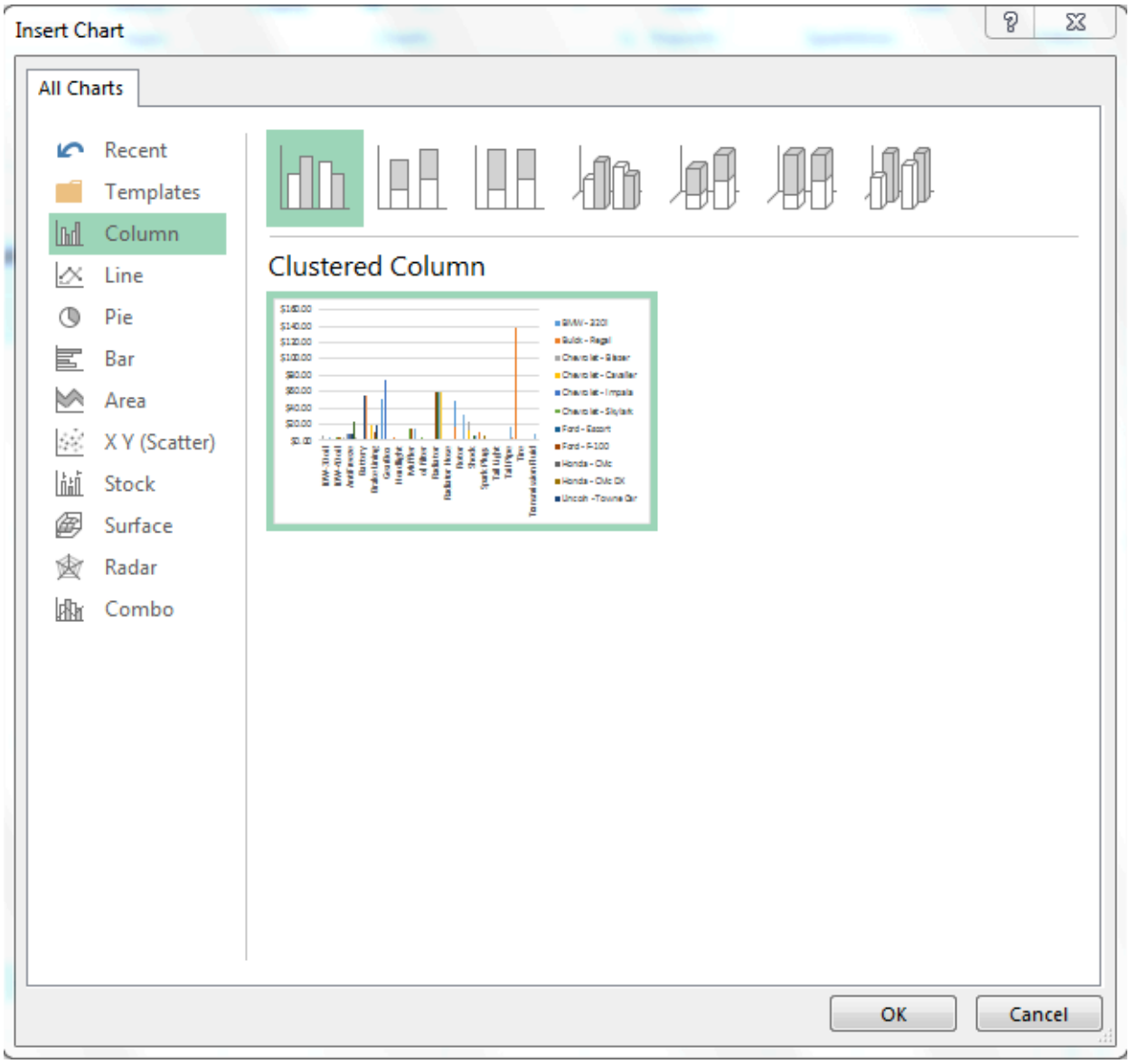


Figure 21: Pivot Chart Insert Window

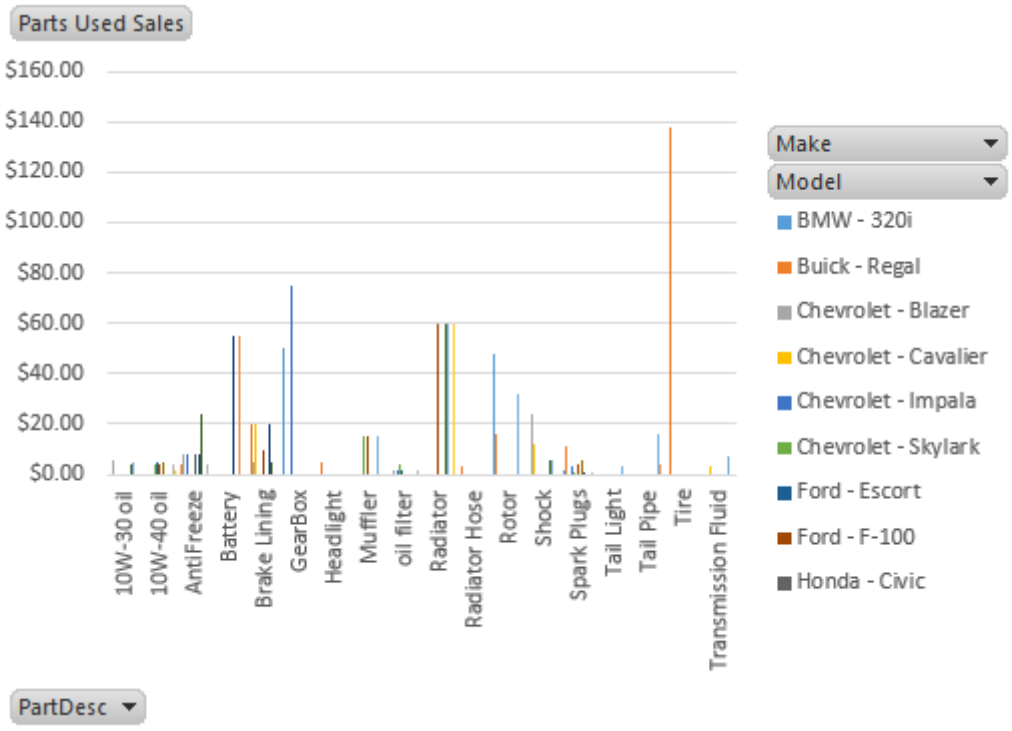


Figure 22: Pivot Chart View

7.1.3 Extending the Pivot Table

The pivot table in the previous section is somewhat limited because it lacks a time dimension, an important element of most data cubes. In addition, the pivot table lacks a filter field to restrict the data displayed. In this section, we add these elements to the previous pivot table and then demonstrate some additional features of pivot tables.

To extend the pivot table of the previous section, we first extend the underlying query and then create another pivot table with the additional fields. Follow these steps to create the new query and the pivot table.

1. Modify the Underlying Query in Access: Add the *Customer* table to the Query Design window as shown in Figure 23. Then, add the *RepairOrder.TimeRecvd* field and the *Customer.City* field to the query result. Save the query as "RptQuery4Revised" when you finish with the changes. Close the Access database.

2. Define a New Pivot Table: Using the same Excel sheet, follow step 1 from Section 7.2.1 to import a query from Access to Excel with *RptQuery4Revised* as the underlying query.

3. Create Pivot Table Fields: Drag *PartDesc* and then *TimeRecvd* from the field list to the row area of the pivot table. Then, drag *Make* and *Model* from the field list to the column area. Finally, drag *City* to the filter area. Figure 24 shows the resulting pivot table.

4. Create the Detail Fields: Drag *Amount* from the field list to the values area. Figure 25 shows the resulting pivot table.

5. Rename the Calculated Field: Change the name of the calculated field from "Sum of amount" to "Parts Used Sales" by using the **Custom Name field** of the Value Field Settings option.). In the **Number Format**, change the Number to "Currency". Figure 26 shows the resulting pivot table.

6. Compact the Pivot Table: Remove the members of the *TimeRecvd* and *Make* fields using the **Collapse** menu item. To remove the *TimeRecvd* items, right-click in any *PartDesc* field heading and select **Expand/Collapse → Collapse Entire Field** from the menu (Figure 27). Similarly, to remove the *Model* items, right-click in any *Make* field heading and select **Expand/Collapse → Collapse Entire Field** from the menu. Figure 28 shows the compressed pivot table after collapsing both fields.

7. Filter the Pivot Table: To see the effect of filtering, use the drop-down arrow of the *City* field. Select **"Seattle"** from the drop down list. Figure 29 shows the pivot table after filtering for Seattle customers.

8. Save the Pivot Table: Change the name of the pivot table to "Part Used Pivot Table 2" using PivotTable Options. Then rename the pivot table sheet as "Part Used Pivot Table 2". Close the excel file when you are finished.

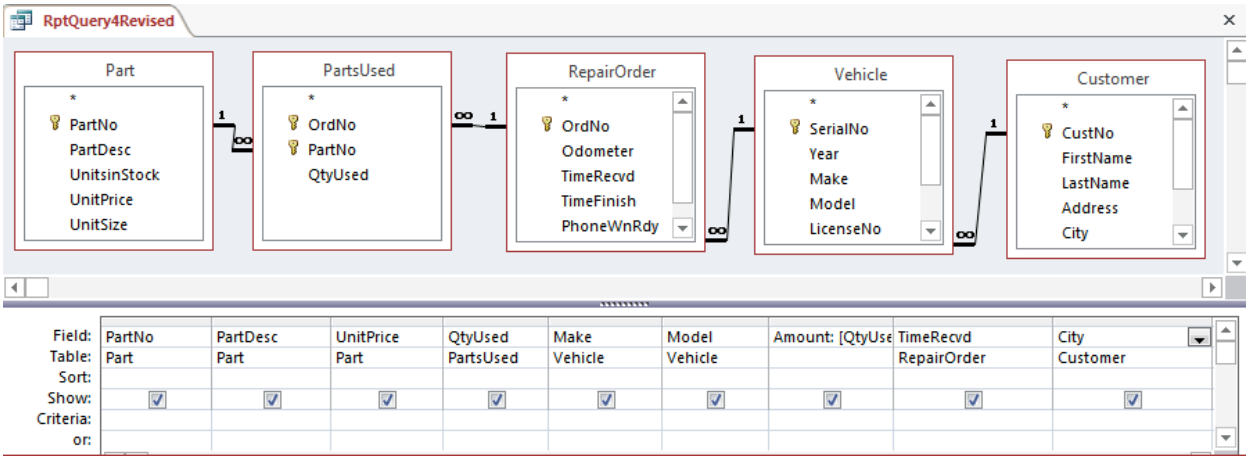


Figure 23: Revised Query for the Modified Pivot Table

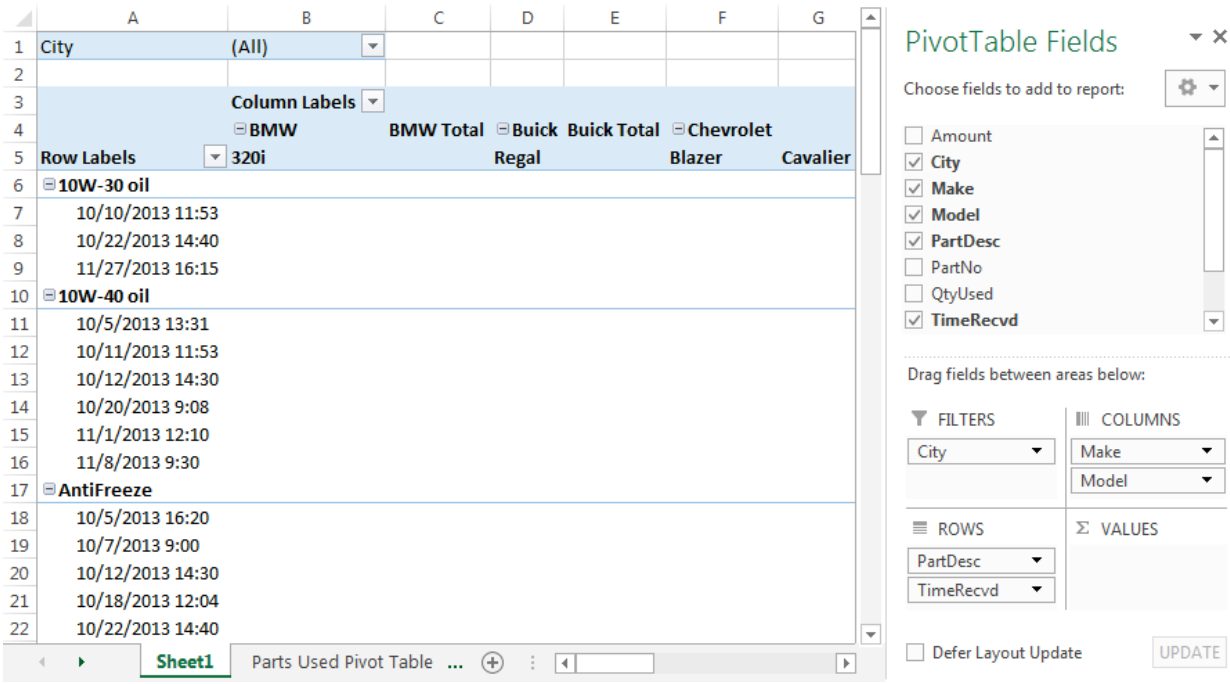
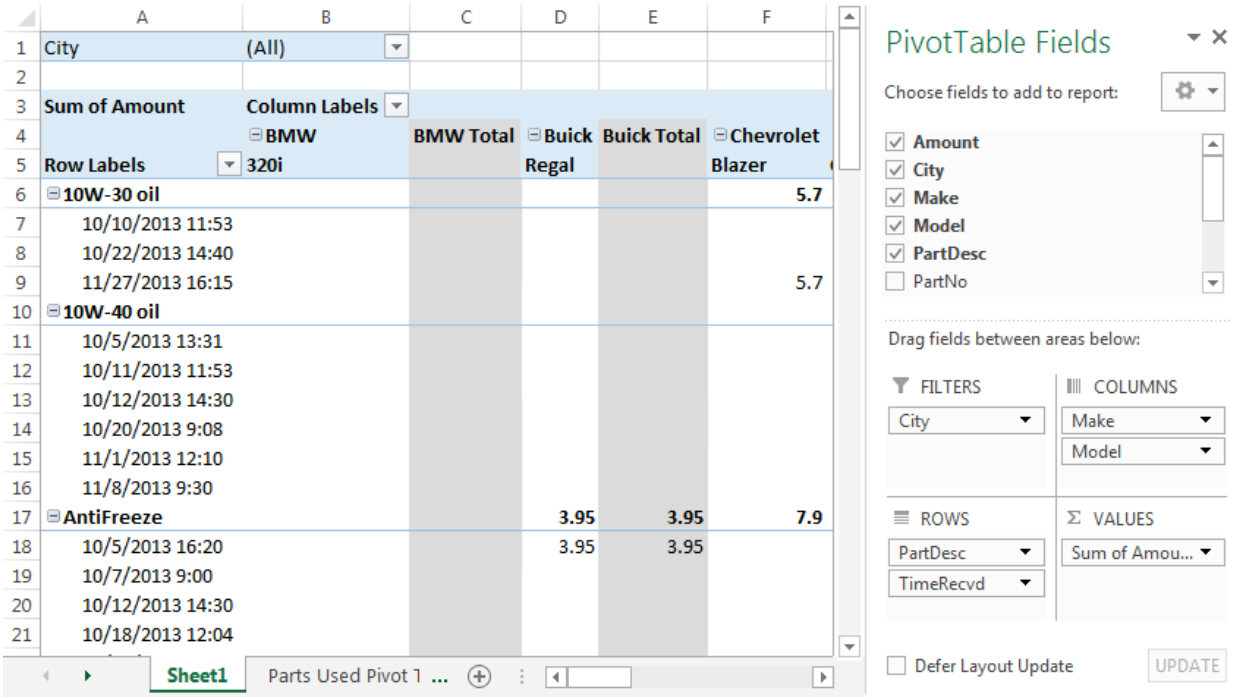


Figure 24: Pivot Table after Dragging Fields to the Rows, Coulmiuns and Filters Areas



	A	B	C	D	E	F
1	City	(All)				
2						
3	Sum of Amount	Column Labels				
4		BMW	BMW Total	Buick	Buick Total	Chevrolet
5	Row Labels	320i		Regal		Blazer
6	10W-30 oil					5.7
7	10/10/2013 11:53					
8	10/22/2013 14:40					
9	11/27/2013 16:15					5.7
10	10W-40 oil					
11	10/5/2013 13:31					
12	10/11/2013 11:53					
13	10/12/2013 14:30					
14	10/20/2013 9:08					
15	11/1/2013 12:10					
16	11/8/2013 9:30					
17	AntiFreeze			3.95	3.95	7.9
18	10/5/2013 16:20			3.95	3.95	
19	10/7/2013 9:00					
20	10/12/2013 14:30					
21	10/18/2013 12:04					

Sheet1 | Parts Used Pivot 1 ...

PivotTable Fields

Choose fields to add to report:

- Amount
- City
- Make
- Model
- PartDesc
- PartNo

Drag fields between areas below:

FILTERS: City

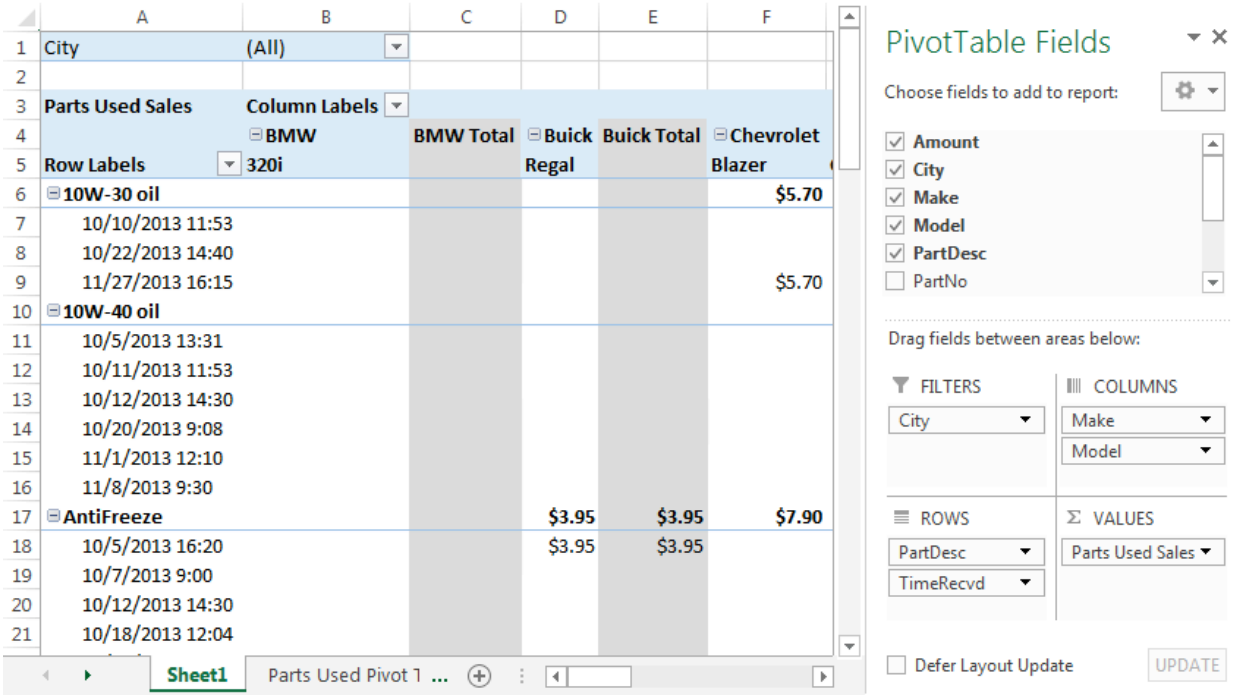
COLUMNS: Make, Model

ROWS: PartDesc, TimeRecvd

VALUES: Sum of Amou...

Defer Layout Update UPDATE

Figure 25: Pivot Table after Dragging Amount to the Values Area



	A	B	C	D	E	F
1	City	(All)				
2						
3	Parts Used Sales	Column Labels				
4		BMW	BMW Total	Buick	Buick Total	Chevrolet
5	Row Labels	320i		Regal		Blazer
6	10W-30 oil					$5.70
7	10/10/2013 11:53					
8	10/22/2013 14:40					
9	11/27/2013 16:15					$5.70
10	10W-40 oil					
11	10/5/2013 13:31					
12	10/11/2013 11:53					
13	10/12/2013 14:30					
14	10/20/2013 9:08					
15	11/1/2013 12:10					
16	11/8/2013 9:30					
17	AntiFreeze			$3.95	$3.95	$7.90
18	10/5/2013 16:20			$3.95	$3.95	
19	10/7/2013 9:00					
20	10/12/2013 14:30					
21	10/18/2013 12:04					

Sheet1 | Parts Used Pivot 1 ...

PivotTable Fields

Choose fields to add to report:

- Amount
- City
- Make
- Model
- PartDesc
- PartNo

Drag fields between areas below:

FILTERS: City

COLUMNS: Make, Model

ROWS: PartDesc, TimeRecvd

VALUES: Parts Used Sales

Defer Layout Update UPDATE

Figure 26: Pivot Table after Changing the Name and Number Format

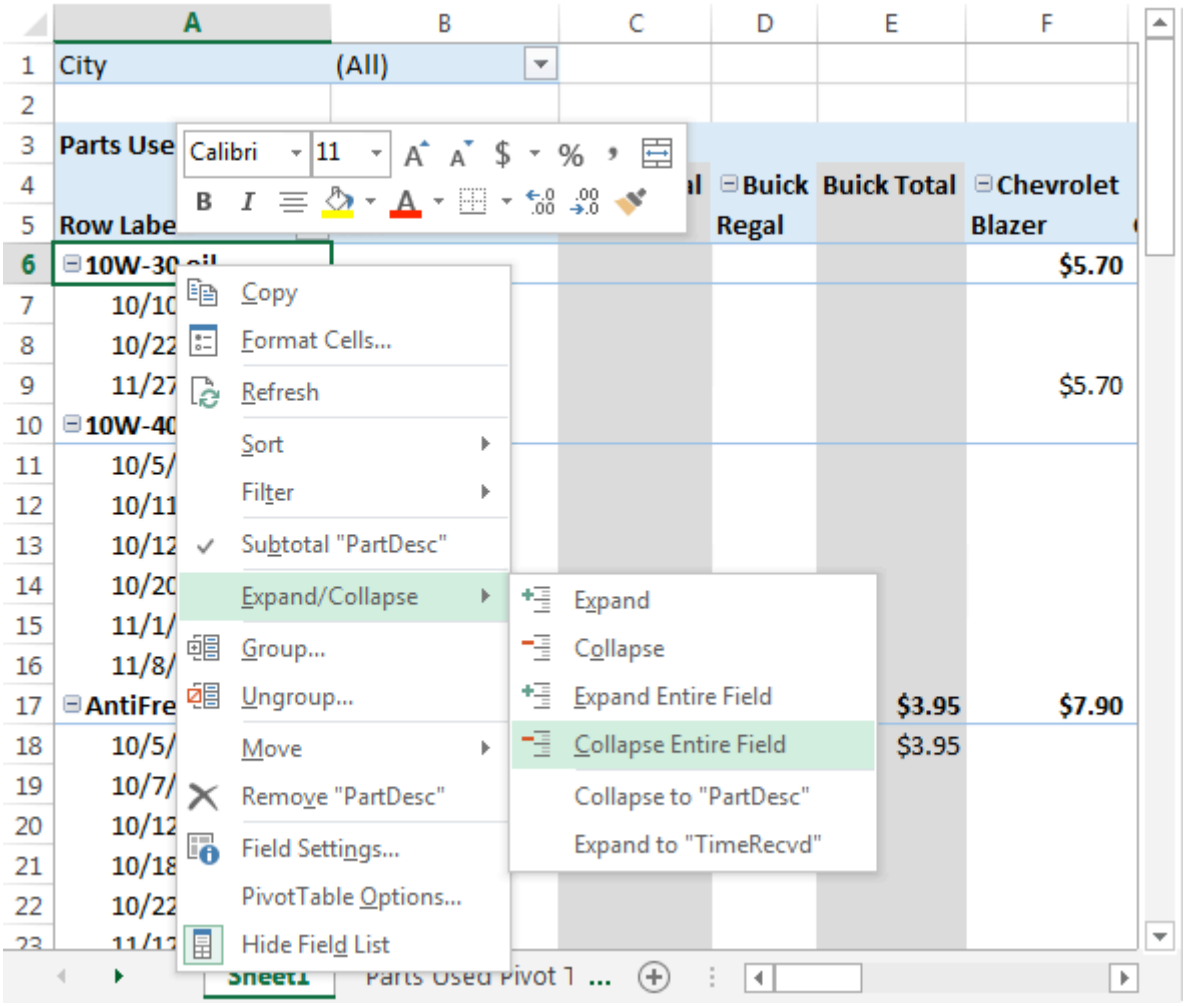


Figure 27: Collapse Option

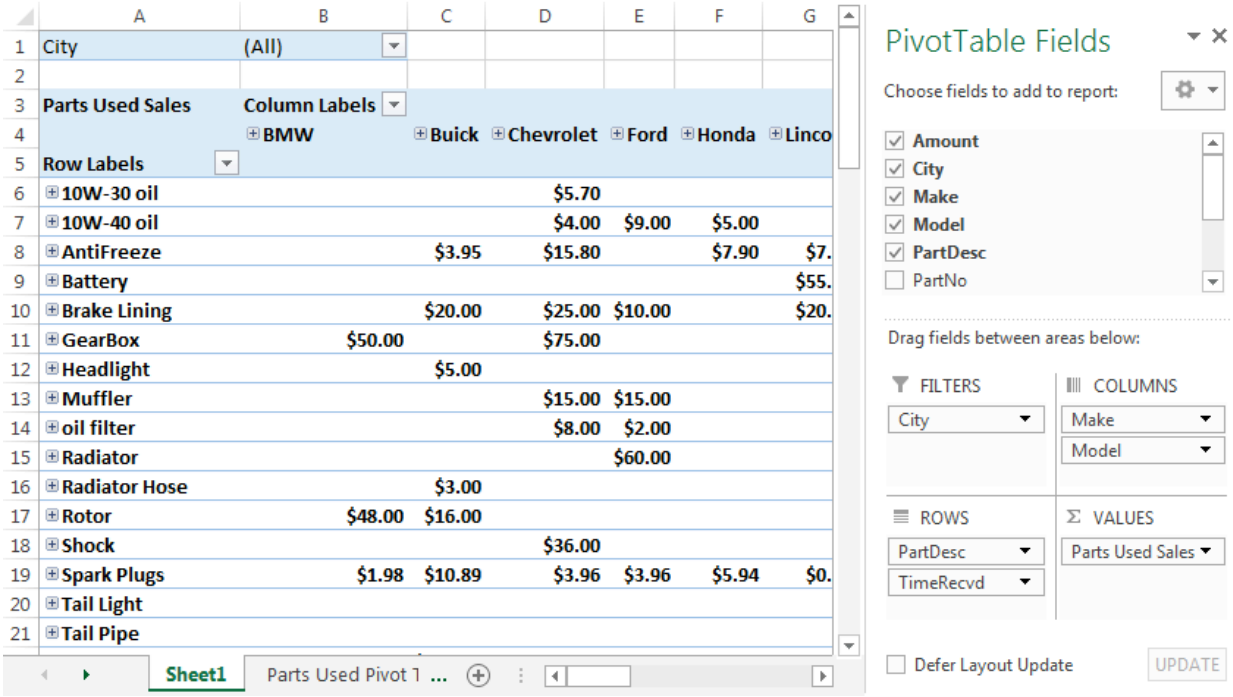


Figure 28: Pivot Table after Collapsing Fields

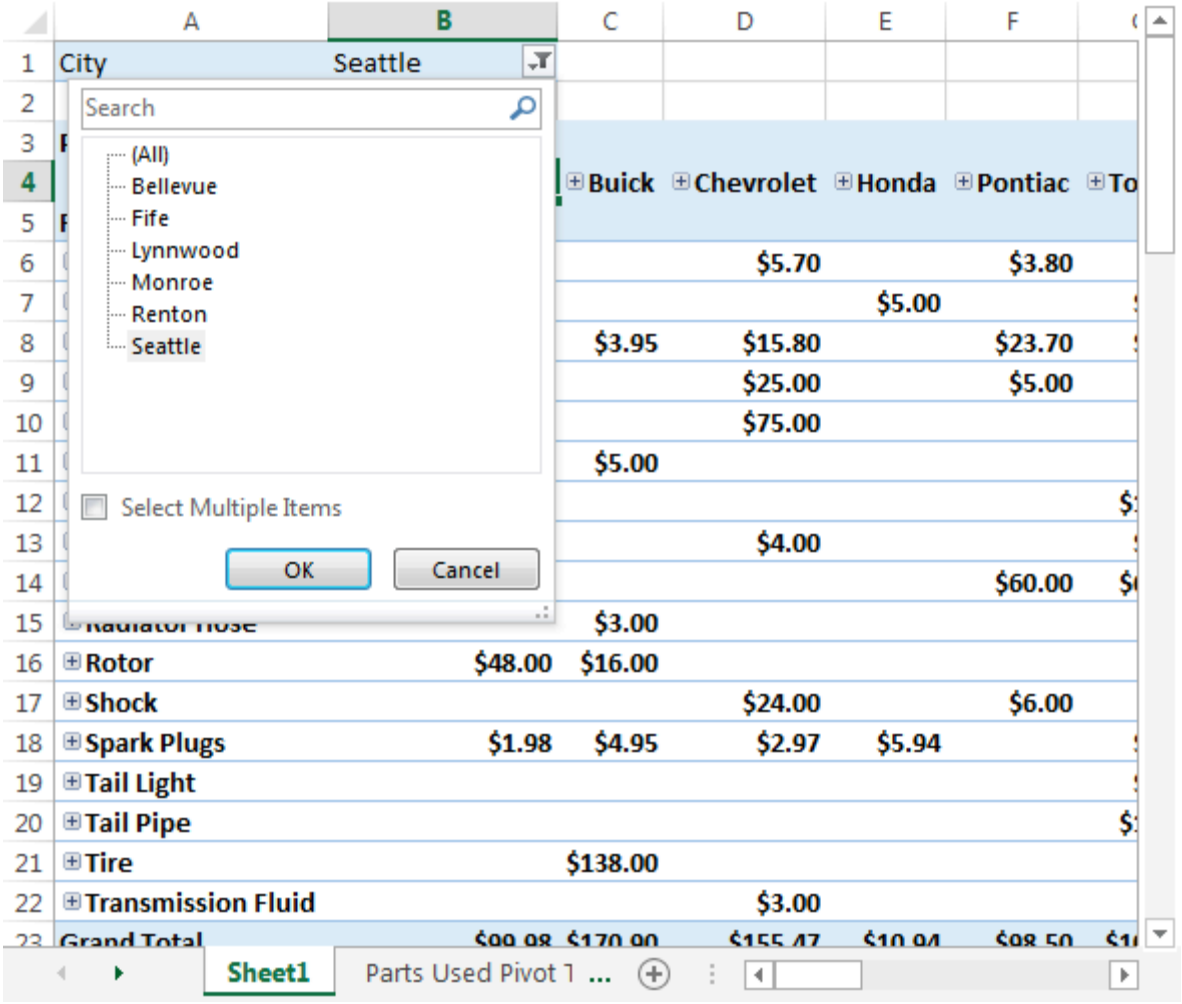


Figure 29: Pivot Table after Filtering on Seattle

Excel also provides the ability to group row and column fields. For date/time fields, Excel provides predefined grouping levels using the **Group** option. For the *TimeRecvd* field, you can group on predefined intervals such as year, quarter, and month. Grouping to a finer or coarser level is comparable to drilling down or rolling up as described in textbook Chapter 16. You can determine groups for fields without predefined groups by selecting items and using the **Group** menu item.

7.2 Specialized Access Forms

This section covers two specialized Access forms, the split form and the multiple items form. These forms are easy to create and useful in specialized situations so you should become familiar with them.

7.2.1 Split Form

A split form, a new feature in Microsoft Office Access 2007, provides two views of data at the same time. You can view all records in a table in Datasheet view and the selected record in Form view. A split form ensures that both views are connected to the same data source and synchronized with each other. For example, if you select a field in one part of a split form, the same field is selected in the other part of the form. Split forms allow you to add, edit, or delete data from either part subject to the normal form constraints (allow properties and record source updatability).

A split form seems ideal for a situation in which a small group of rows should be reviewed at the same time. For example, a split form may be useful to review interesting subsets of vehicles such as vehicles of the same model, make, and year.

This section provides practice with creating a split form and modifying an existing form into a split form. Both tasks can be accomplished easily as demonstrated by the instructions in this section.

New Split Form

1. Select Table: In the Navigation pane, select the **Tables** menu. Then select the *Vehicle* table.
2. Create the Split Form: While the *Vehicle* table is selected in the Navigation pane, select **Create → More Forms → Split Form** in the Forms group of the Ribbon. The split form appears in the view pane in Layout view (Figure 30).
3. Navigate Records: Navigate records in Datasheet view and see the current record change in Layout view. Access ensures that the actions on each form are synchronized. Figure 31 shows the second record in Layout view with the second record selected in Datasheet view.
4. Navigate Fields: Select different fields to see that the same field is selected in the other view. In Figure 33, the *Model* field is selected in both views. In Form view, the field in the single form area is not highlighted as shown in Figure 33. Note, however, that the records in the single form and datasheet parts are still synchronized.
5. Save the Form: Close the form and save it as “Vehicle-SplitForm”.

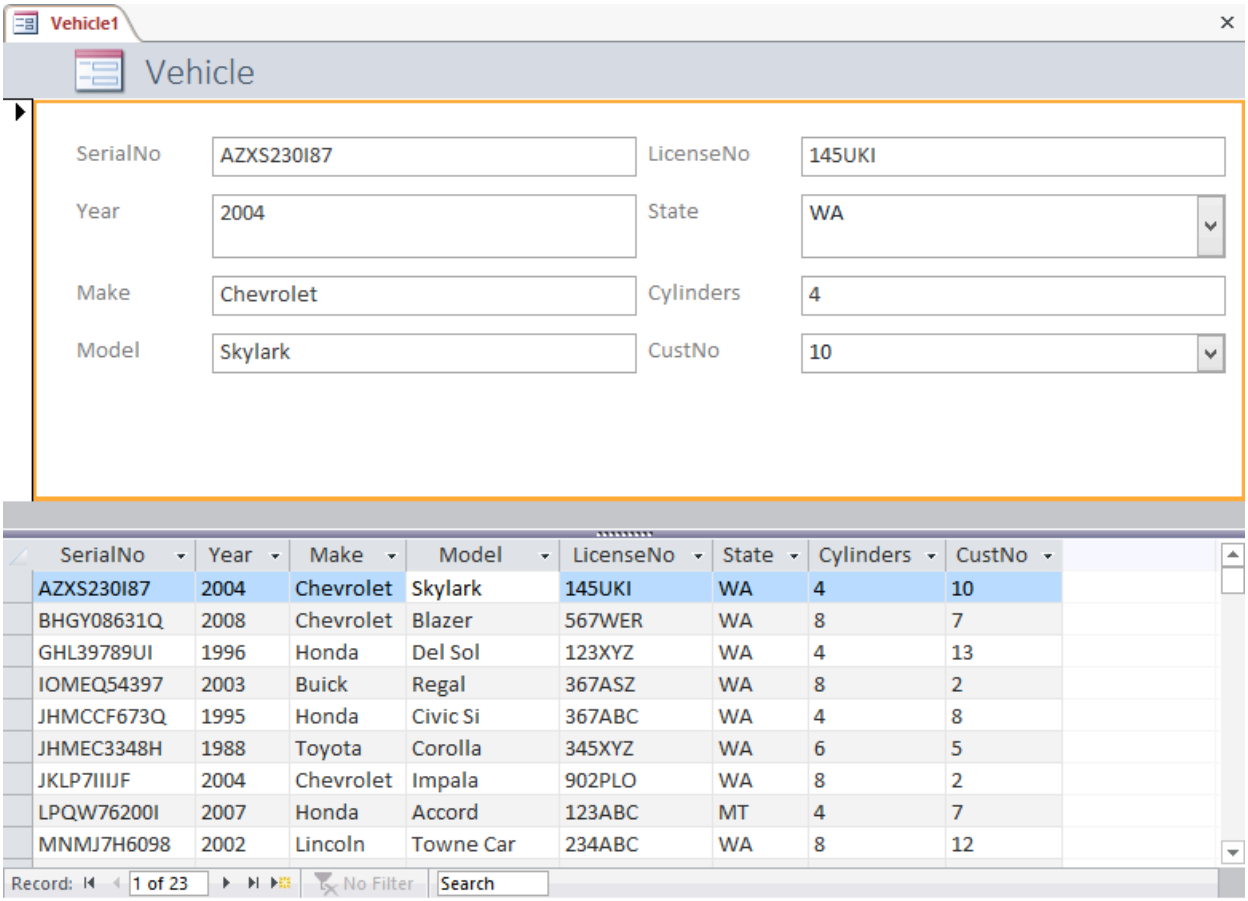


Figure 30: Split Form in Layout View for the Vehicle Table

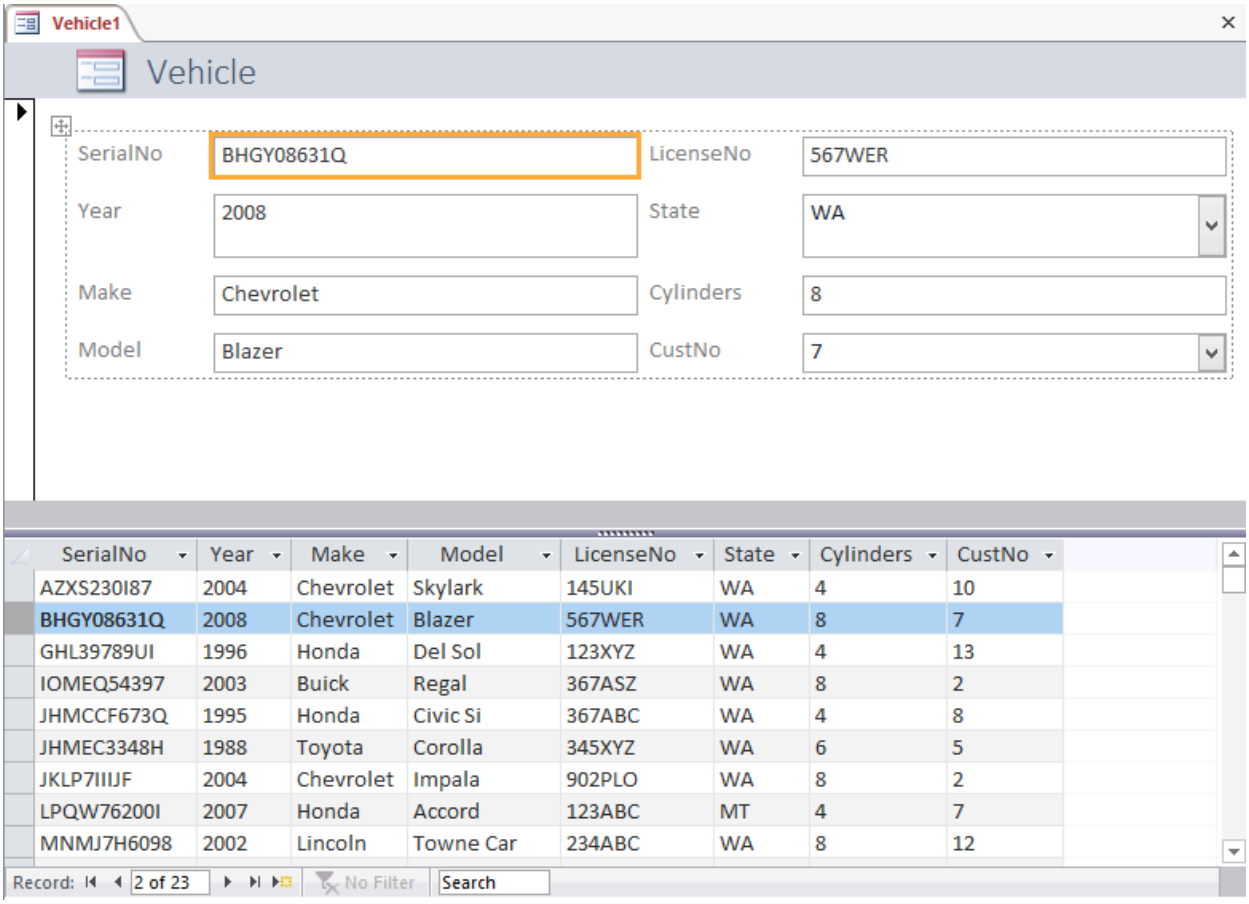


Figure 31: Split Form in Layout View Showing the Second Record Selected in Both Views

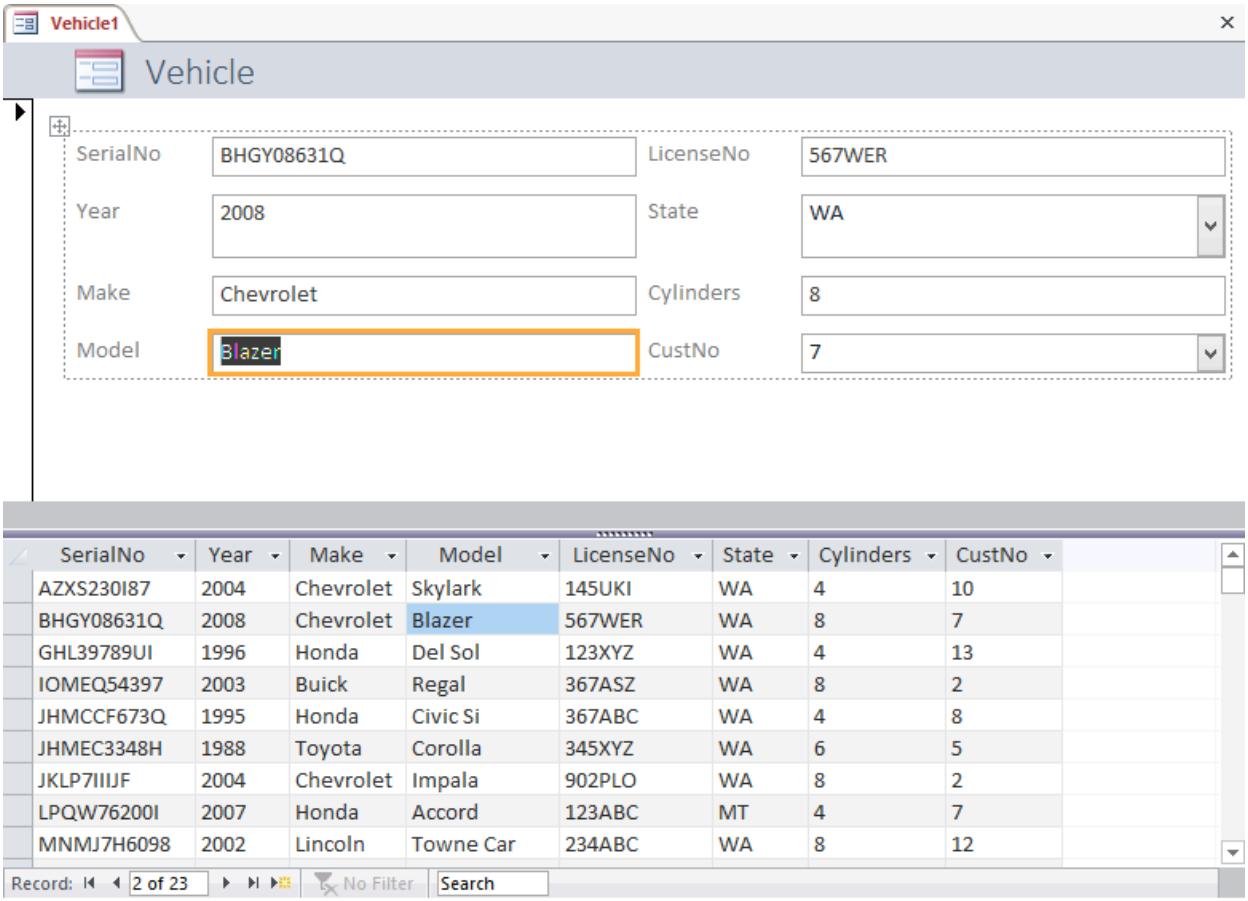


Figure 32: Split Form in Layout View Showing the Model Field Selected in Both Views

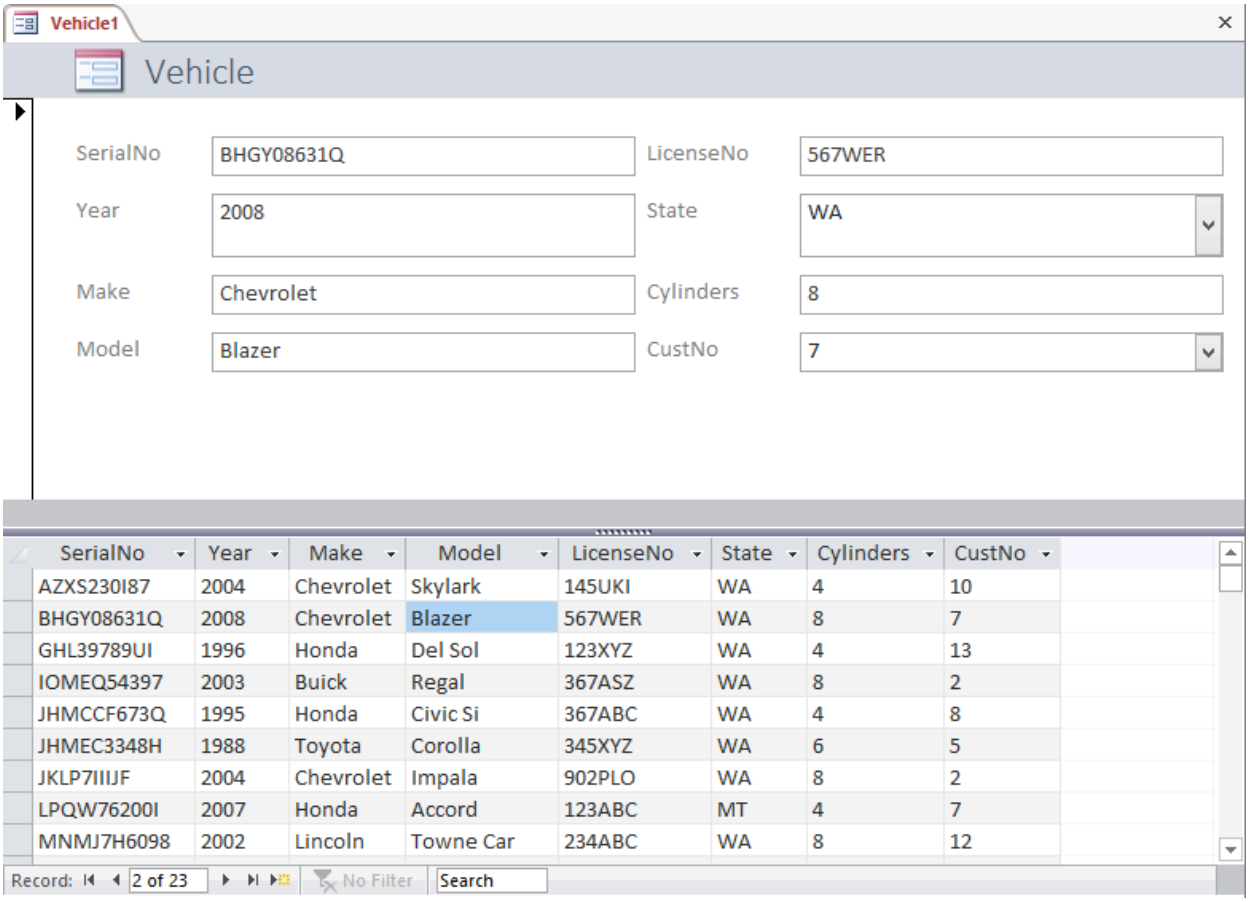


Figure 33: Split Form in Form View with the Model Field Selected in Datasheet View

Modify an Existing Form to a Split Form

1. Create New Form: In the Navigation pane, select the **Tables** item. Then select the *Customer* table.
2. Create a Form: While the *Customer* table is selected in the Navigation pane, select **Create → Form** in the Forms group of the Ribbon. The form appears in the view area in Layout view (Figure 34).
3. Open Property Sheet: Open the new Customer form in Design view. Open its Property Sheet by clicking **Design → Property Sheet**. Alternatively, you can type **F4** to see the property sheet after selecting the form.
4. Select Form: In the Property Sheet, select Form in the Selection Type combo box as shown in Figure 35.
5. Change the Default View Property: On the **Format** tab of the property sheet, select "Split Form" in the *Default View* property as shown in Figure 36.
6. View the Form: Switch to Form View to see the split form view of the new Customer form (Figure 37).
7. Save the Form: Close the form and save it as "Customer-SplitForm".

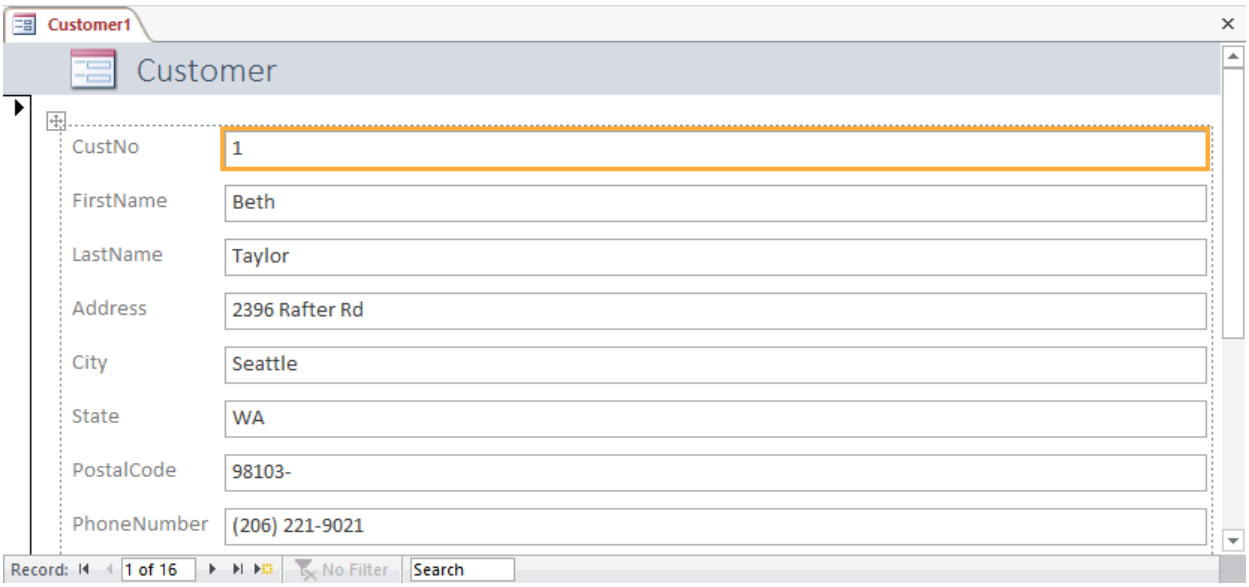


Figure 34: Layout View for the Customer Form

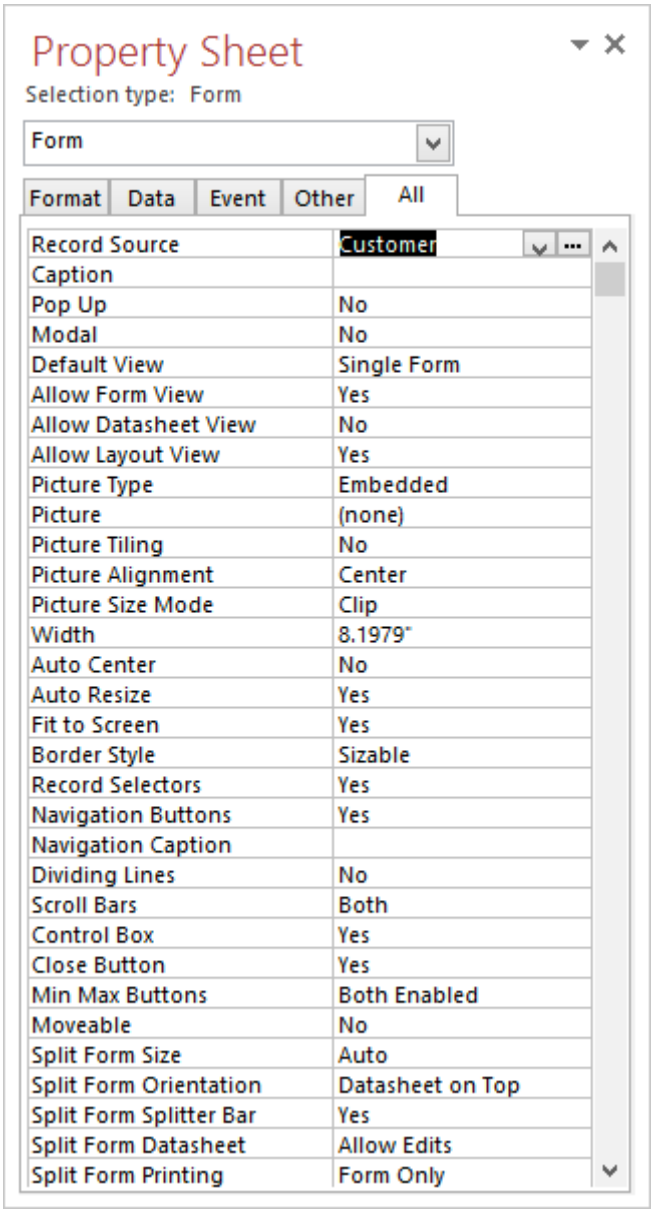


Figure 35: Property Sheet Showing the Form Properties

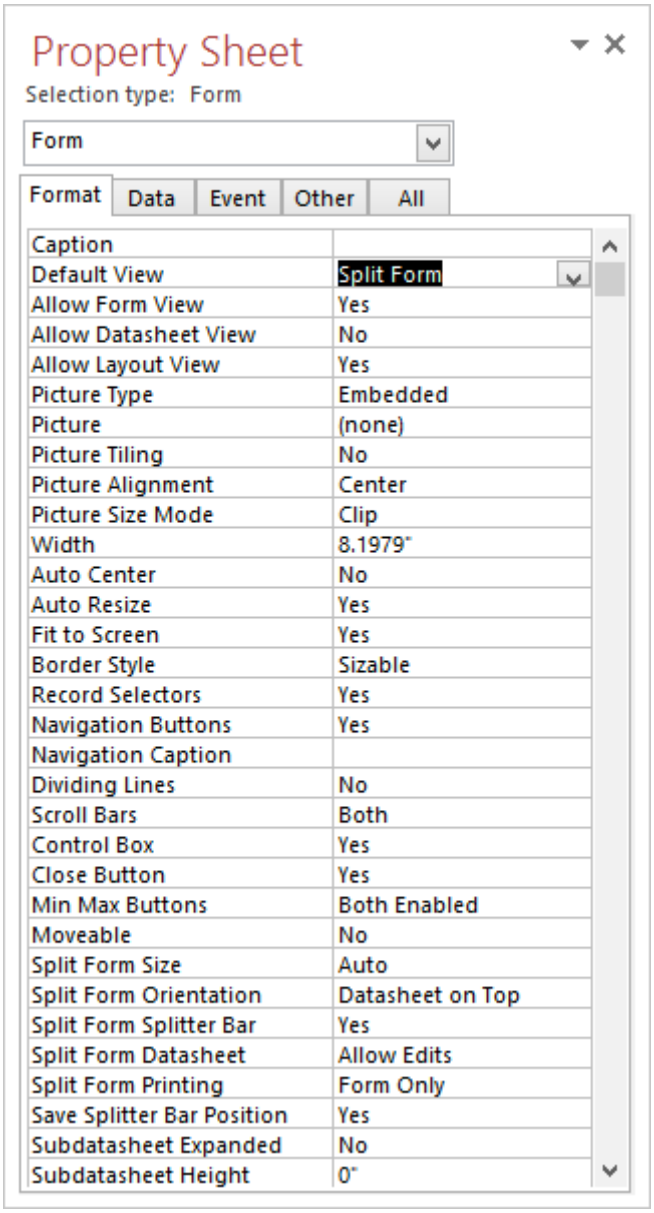


Figure 36: Property Sheet Showing “Split Form” Chosen for the Default View Property

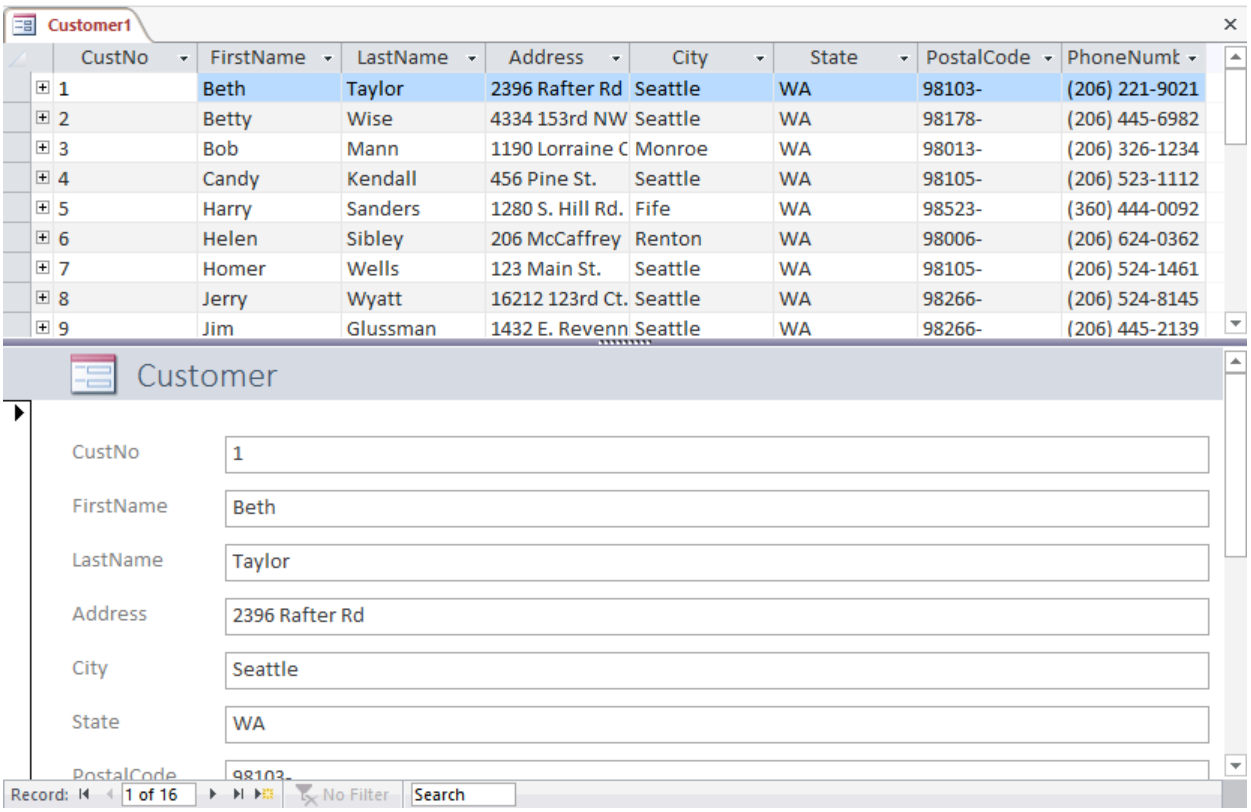


Figure 37: Split Form in Form View for the Customer Table

7.2.2 Multiple Items Form

A multiple items form[14] displays data from more than record at the same time. When first created, a multiple items form resembles a datasheet. A multiple items form is appropriate for power users who want a stylized datasheet for data entry. The process of creating a multiple items form is similar to creating a split form as presented in the remainder of this section.

1. Select Table: In the Navigation pane, select the **Tables** item. Then select the *Vehicle* table.
2. Create the Split Form: While the *Vehicle* table is selected in the Navigation pane, select **Create → More Forms → Multiple Items** in the Forms group of the Ribbon. The multiple items form appears in the view pane in Layout view (Figure 38).
3. Change to Form View: Click **Home → View → Form View** to see the same form in Form view (Figure 39). Form view appears similar to Layout view except for the highlighted fields in Layout view.
4. Change to Design View: Click **Home → View → Design View** to see the same form in Design view (Figure 40). The heading area shows the labels for the datasheet headings and the detail area shows the textboxes for the datasheet rows.
5. Review Property Sheet: In Design view, click **Design → Property Sheet** to open the Property Sheet for the multiple items form. Note that the *Default View* property is set to "Continuous Forms" as shown in Figure 41.
6. Save the Form: Close the form and save it as "Vehicle-MultipleItemsForm".

[14] A multiple items form is sometimes called a continuous form.

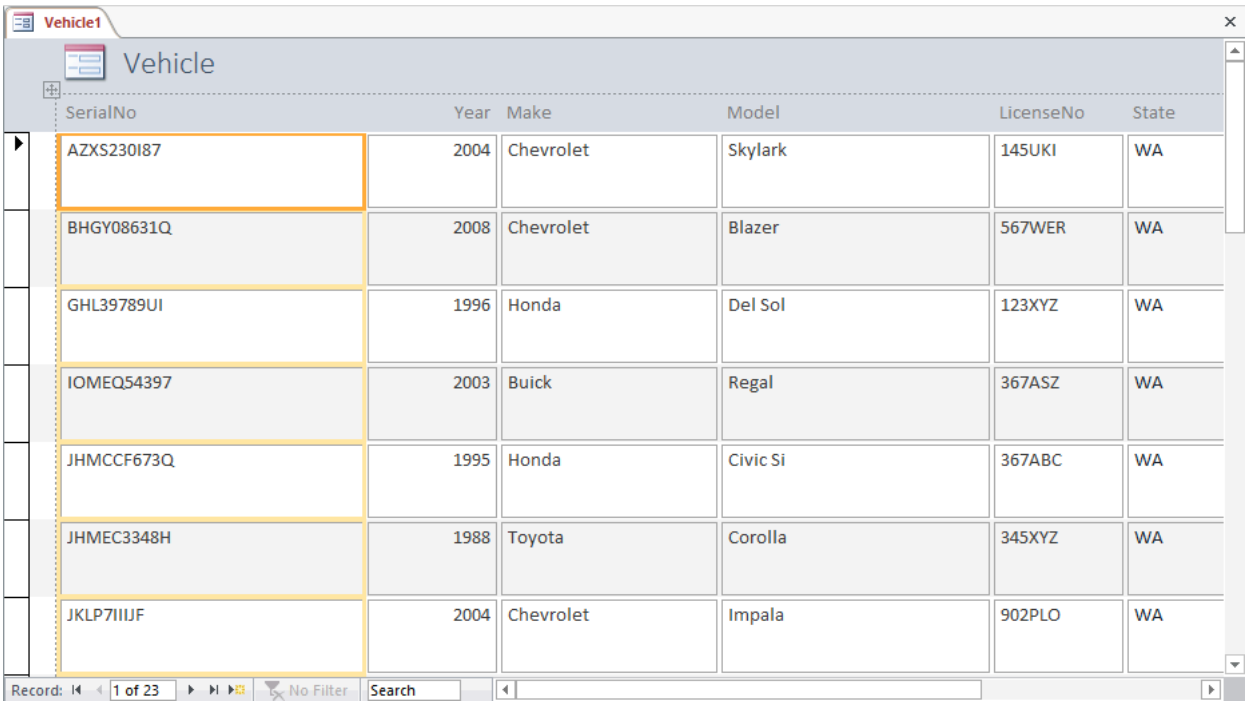


Figure 38: Multiple Items Form in Layout View for the Vehicle Table

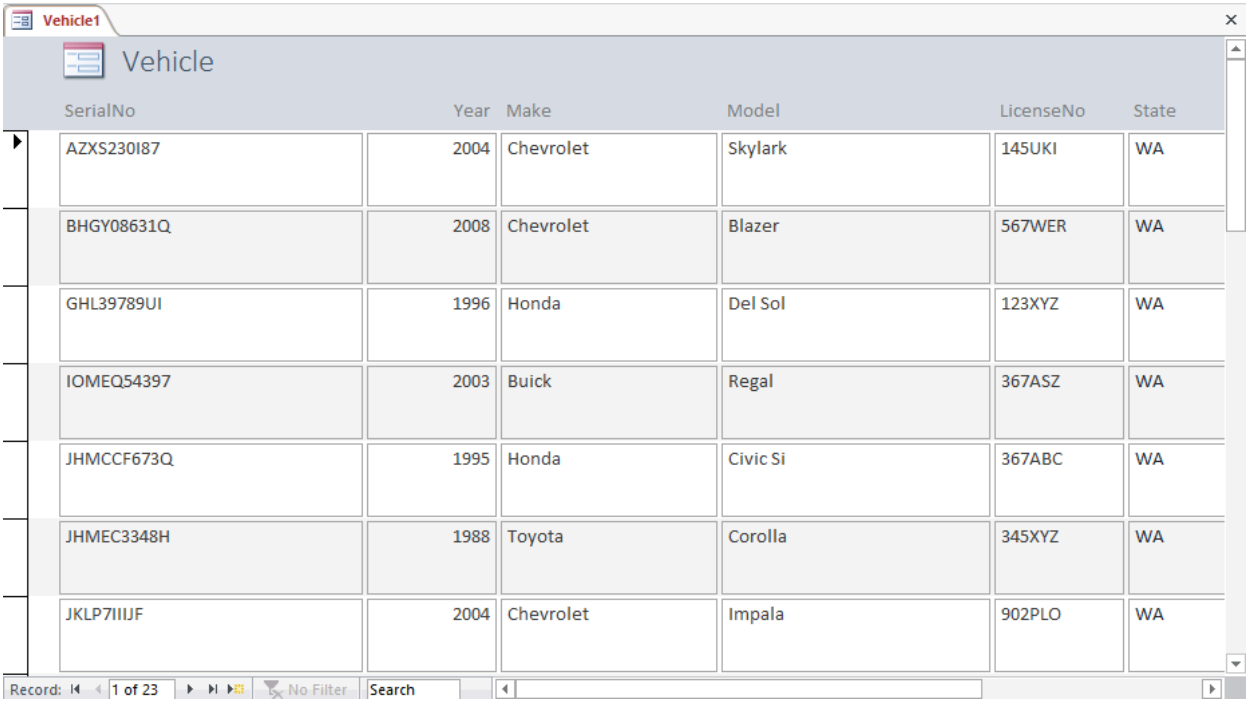


Figure 39: Multiple Items Form in Form View for the Vehicle Table

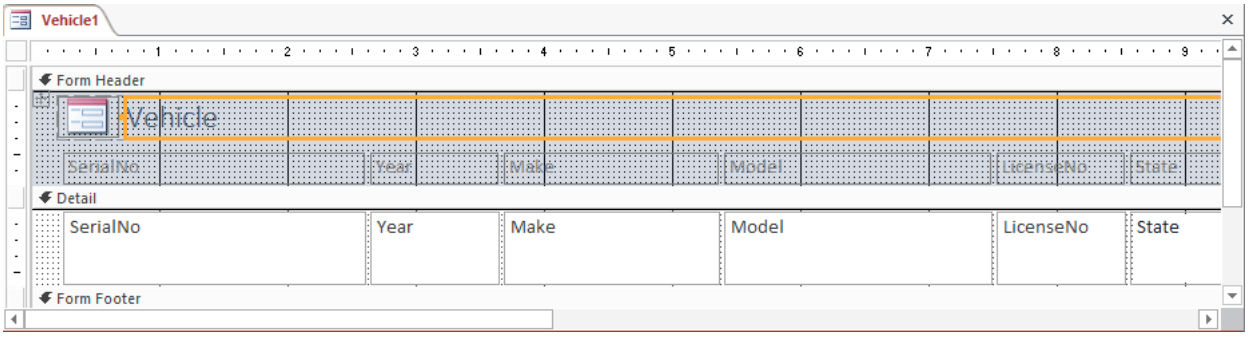


Figure 40: Multiple Items Form in Design View for the Vehicle Table

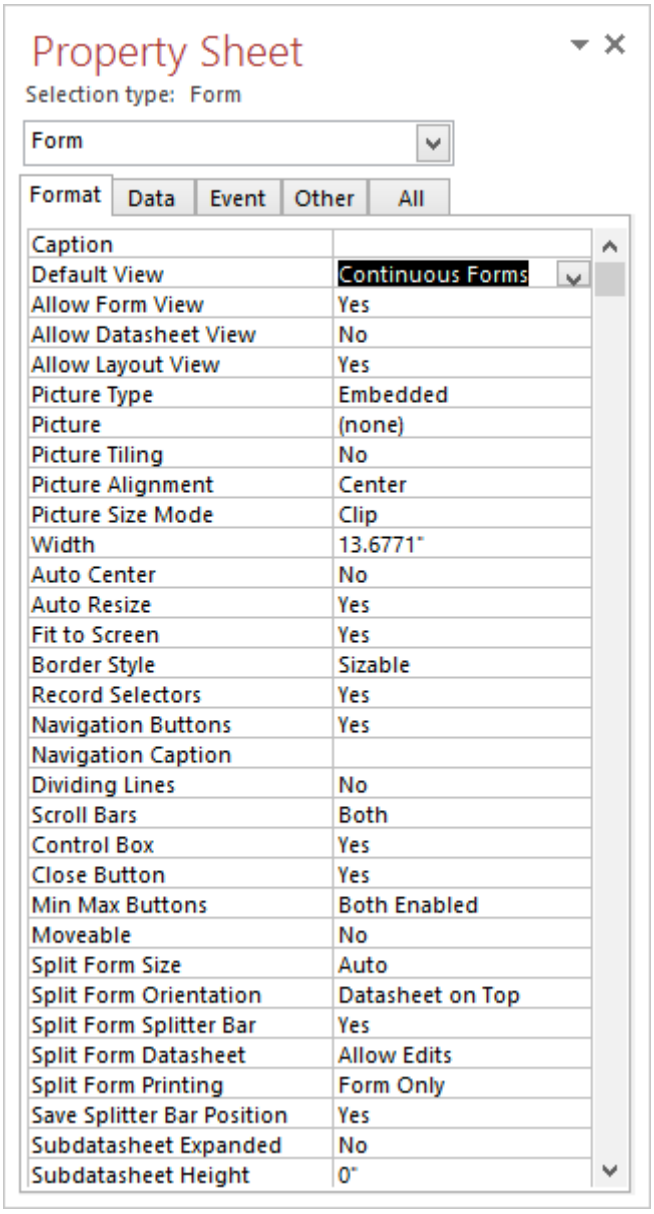


Figure 41: Property Sheet for the Vehicle Multiple Items Form

Closing Thoughts

This chapter has covered specialized database objects, pivot tables, split forms, and multiple item forms. Pivot tables provide a convenient interface for manipulating multidimensional data, a common format for business analysis. You learned about the vocabulary for pivot tables and steps in designing a pivot table. Since Access 2013 does not support pivot tables, you gained practice to import queries from Access 2013 into Excel 2013 and create and customize pivot tables in Excel 2013. The practical material presented in Section 7.1 complements the conceptual material in textbook Chapter 16. Split forms and multiple item forms have important but specialized uses. They are simple to create and modify when needed.

This chapter concludes the study of individual database objects for application development. Chapter 8 integrates the material from other chapters in the coverage of navigation aids and customization of database objects using macros.

Chapter Reference

The chapter reference section summarizes procedures that you practiced. For wizards discussed in the chapter, the procedures highlight important parts of the wizards but do not list all of the steps.

Procedure 1: Creating a Pivot Table (Section 7.1.2)

1. Before creating a pivot table, you should determine the dimensions and measures of the data cube that it supports.
2. The underlying query for the pivot table should include all fields in the pivot table except for fields that are computed in the pivot table.
3. Use "Get External Data" utility to import queries from Access database.
4. Use the PivotTable option to choose the row fields, column fields, filter fields, and value fields from the fields in the underlying query.
5. Use Value Field Settings to change the properties of individual fields in the pivot table.
6. Use pivot table options to change the pivot table properties such as the *name* property.

Procedure 2: Create a Split Form (Sections 7.2.1)

1. Choose the object (table or updatable query) in the Navigation pane for which you want to create the split form.
2. Click **Create → More Forms → Split Form** in the Forms group of the Ribbon to create a split form for the object selected in the Navigation pane.
3. The form opens in Layout view. In Layout view, selecting a field highlights it in Datasheet view.
4. You can switch views (Design, Layout, and Form) using the **Home → View** drop down list.

Procedure 3: Modify a Form to a Split Form (Sections 7.2.1)

1. Choose the form in the Navigation pane for which you want to modify into a split form.
2. While the form is selected in the Navigation pane, open the form in Design view by right clicking on the form and selecting **Design View**.
3. After the form opens in Design view, open its Property Sheet by clicking **Design → Property Sheet**.
4. In the **Format** tab of the property sheet, select "Split Form" in the *Default View* property.

Procedure 4: Create a Multiple Items Form (Section 7.2.2)

1. Choose the object (table or updatable query) in the Navigation pane for which you want to create the split form.
2. Click **Create → More Forms → Multiple Items** in the Forms group of the Ribbon to create a multiple items form for the object selected in the Navigation pane.
3. The form opens in Layout view.
4. You can switch views (Design, Layout, and Form) using the **Home → View** drop down list.

Additional Practice

The following problems provide additional practice with the extended auto repair database as well as the textbook databases.

Pivot Tables for the Extended Auto Repair Database

1. Create a pivot table to support decision making about the labor costs of repairs. Your pivot table should support a data cube with dimensions of make, model, labor code, and starting repair date and measures for the actual labor cost, the standard labor cost, and the labor cost difference (actual labor cost minus standard labor cost). You need to write the underlying query in Access 2013 and then create the pivot table using Excel 2013.
2. Create a pivot table to support decision making about part expense for vehicles. Your pivot table should support a data cube with dimensions of make, model, cylinders, starting repair date and measures for the total part expense and the number of repairs. You need to write the underlying query in Access 2013 and then create the pivot table in Excel 2013. The pivot table should also have vehicle state as a filtering field.

Chapter 8: Navigation Structure and Macro Lab

Learning Objectives

This chapter covers Access 2013 features that allow you to define a navigation structure to connect application objects and specify macros for customization of application objects and enforcement of business rules. At the completion of this chapter, you should have acquired the knowledge and skills to

- Create a switchboard form to organize applications
- Create a navigation form to organize applications for web deployment
- Create command buttons on forms using the Command Button Wizard
- Create application event macros for business rules associated with applications
- Create data macros for business rules associated with tables and columns

Overview

In other lab chapters you learned how to create the objects of a database application system. You created tables, queries, forms and reports. This chapter demonstrates development of a navigation structure for using database objects, customizing application objects with macros, and developing data macros to enforce business rules.

This chapter demonstrates a number of tools to support convenient access to database objects, customization of application objects, and enforcement of business rules. You will create two types of forms to organize application objects: the older style switchboard form and the navigation form, a new feature started in Access 2010 especially well-suited for deploying database applications on a website. You also will gain experience using command buttons to provide direct navigation among forms. Macros allow you to customize the way a database object responds to user actions. In addition, macros reduce user time by consolidating the steps for routine actions. You will create a simple macro as well as more complex macros that use conditions. You will gain experience with another feature of Access 2013, data macros for business rules associated with tables and columns.

Previous versions of the Access Labs (2007 and before) presented modules that provide more flexibility than macros although module coding is more difficult than macro specification. To write a module, you need to use Visual Basic for Applications (VBA), the language for modules. With major extensions to macro specification and the inclusion of data macros, the need to code modules has been sharply reduced. Modules are primarily required for detailed calculations that cannot be performed in macros. For example in a pension database, calculation of the present value of a single premium annuity providing lifetime income cannot be performed in macros. Since the need to write modules has been sharply reduced, the Access 2013 Lab book no longer covers modules. The module coverage has been replaced with extensive coverage of data macros.

8.1 Navigation among Database Objects

When a database contains many forms and reports, uses need a navigation aid to organize the choices. When Access starts, the navigation form initially opens to allow convenient usage of database objects. The navigation form allows a user to click a command button or tab that opens a specified form or report. To return to the navigation form, the user closes the form or report. Otherwise, the user clicks another command button to open another form or report. This section demonstrates the navigation features of Access using the older style switchboard form and the contemporary style navigation forms especially suited for web deployment of a database.

8.1.1 Using the Switchboard Manager

A switchboard form is simply a blank form that contains command buttons with identifying labels. A user clicks a button and the appropriate form or report opens. A switchboard form may be created in design view with a title added to the form header and command buttons added on the form using the toolbox. However, a faster way to build a switchboard form is to use the Switchboard Manager. Similar to a wizard, the Switchboard Manager prompts you with a sequence of dialog windows.

When you create a form with the Switchboard Manager, Access creates the *Switchboard Items* table and the *Switchboard* form. The *Switchboard Items* table contains details about the command buttons on the form such as the action to take when a button is clicked. The *Switchboard* form contains the actual command buttons but not the action details. Access uses the *Switchboard Items* table to operate the *Switchboard* form.

To change a switchboard form, you can use form design to change the appearance of the form and the buttons. To change the action details, you should use the Switchboard Manager to edit an existing switchboard form. Do not use form design to change action details. Keep in mind that the Switchboard Manager has certain limitations. For example, a switchboard form can contain at most eight command buttons. To add more, you need to make another switchboard form. You would probably want to rearrange the command buttons between the original switchboard form and a new switchboard form. In addition, you would want to have a button on the original switchboard form that opens the new switchboard form. The instructions below will help you make a switchboard menu for the auto repair database.

After opening the database from Chapter 7, you should follow these steps to build a switchboard form.

1. Open the Access Options Window: In Access 2010, the switchboard manager was removed from the Ribbon because navigation style forms are the preferred navigation aids for web deployed databases. Thus, you need to add the Switchboard Manager to the Ribbon before you can use it. Click **File → Options** to display the Access Options window (Figure 1). Select the **Customize Ribbon** button. In the "Choose commands from:" drop-down list, choose "Commands Not in the Ribbon". Scroll down the list to find the Switchboard Manager item.

2. Add the Switchboard Manager to the Database Tools tab: Before you can add the Switchboard item, you need to create a new group in the Ribbon. Select Tools under Database Tools in the right pane. Then click the **New Group** button. Rename the group as "Switchboard Manager Group" using the **Rename** button. The new group appears as in Figure 2. Select the Switchboard item in the left pane and click the **Add >>** button to place it in the Database Tools tab as shown in Figure 3. Click the **OK** button to close the Access Options window.

3. Open the Switchboard Manager Window: Click **Database Tools → Switchboard Manager** in the new Switchboard Manager group of the Ribbon (Figure 4). Note that the switchboard manager icon in the Ribbon may be different depending on your choice of icon in step 2. A dialog box appears stating that there is no valid switchboard in this database and asks if you would like to create one. Click **Yes**.

4. Open the Edit Switchboard Window: In the Switchboard Manager window (Figure 5), click **Edit…** to open the Edit Switchboard window (Figure 6).

5. Enter Menu Name: In the Edit Switchboard window, type "Auto Repair Shop Database Menu" in the Switchboard Name area (Figure 7). Click the **New** button to open the Edit Switchboard Item window (Figure 8).

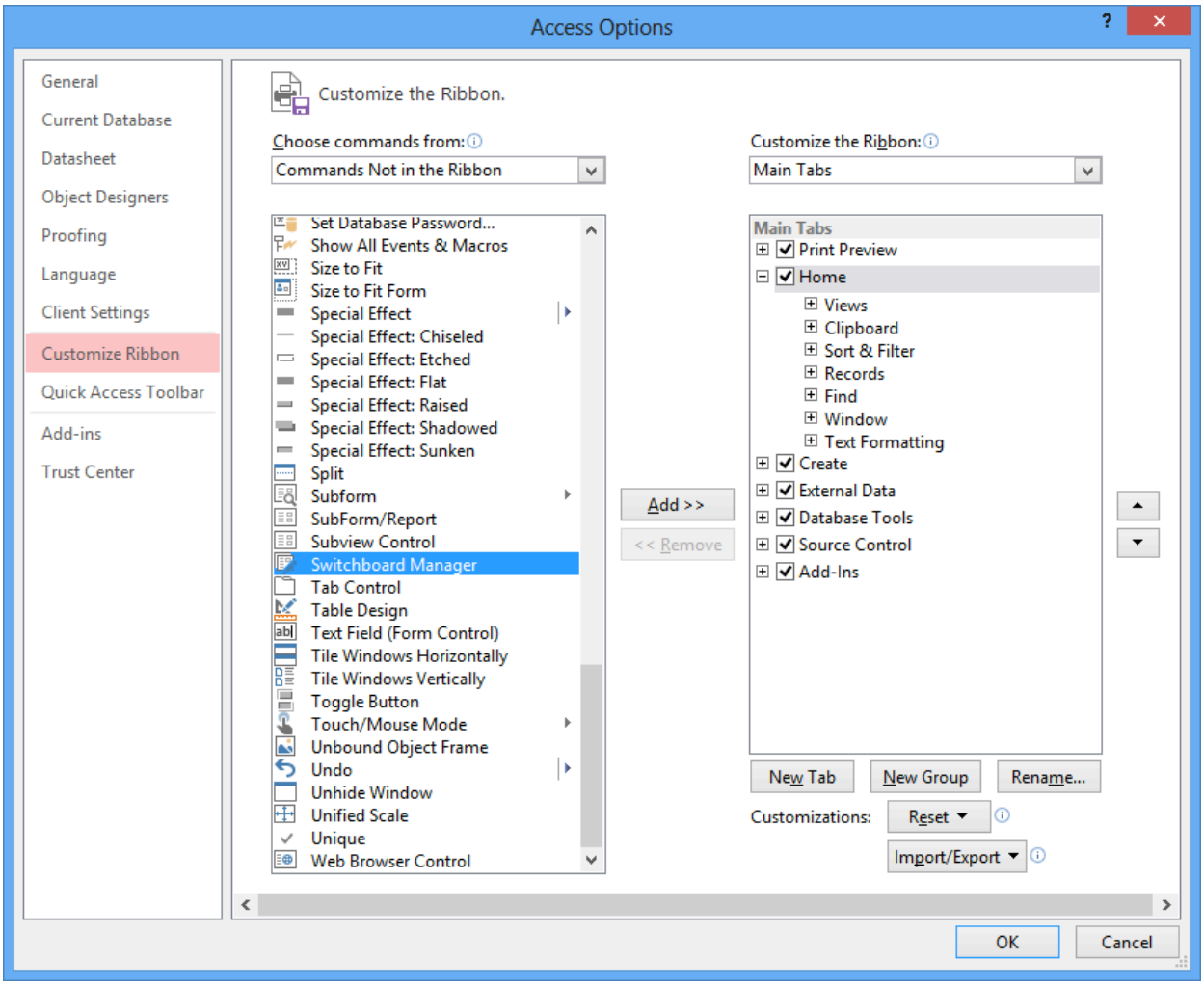


Figure 1: Access Options Window Showing the Switchboard Manager

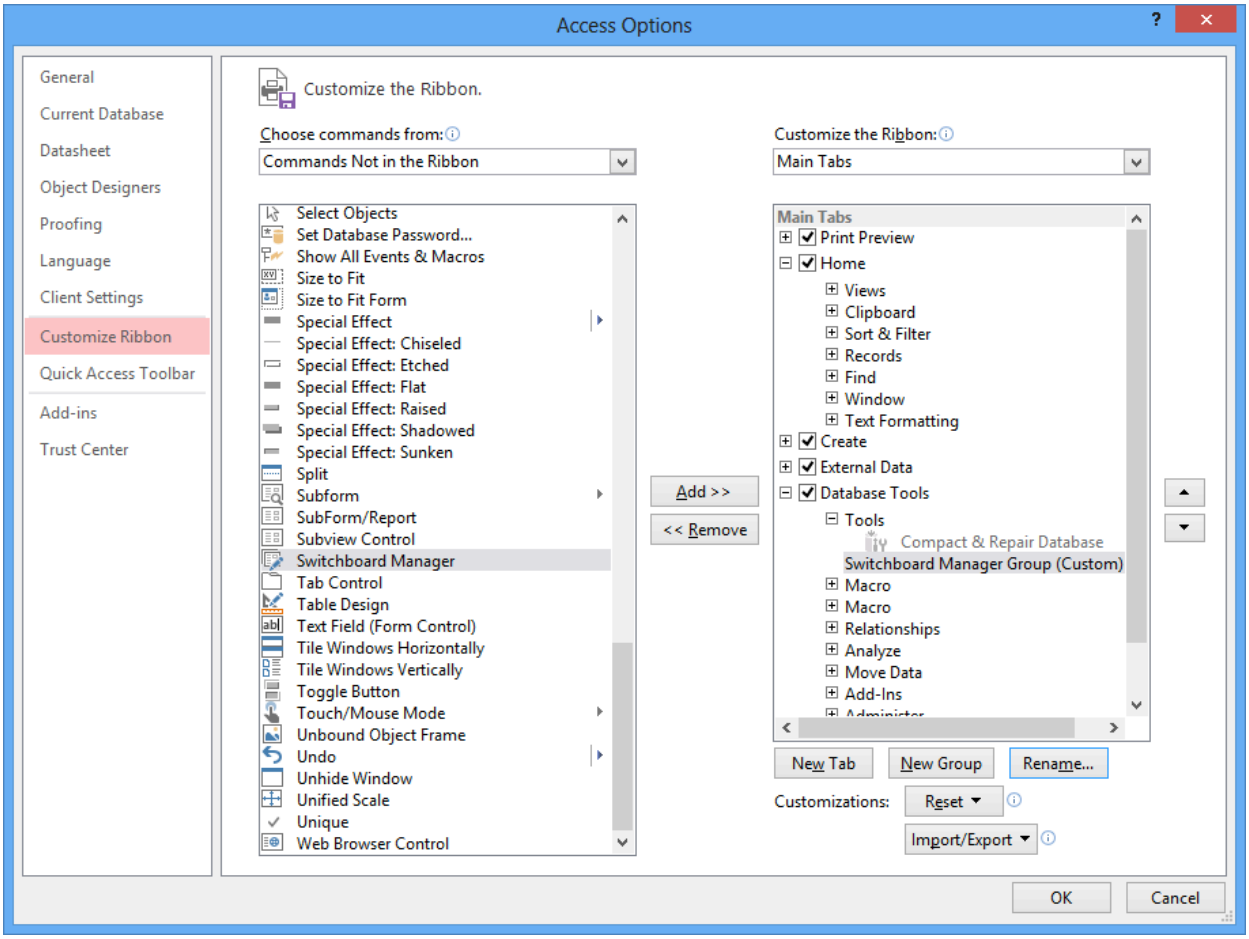


Figure 2: Switchboard Manager Group added to the Database Tools

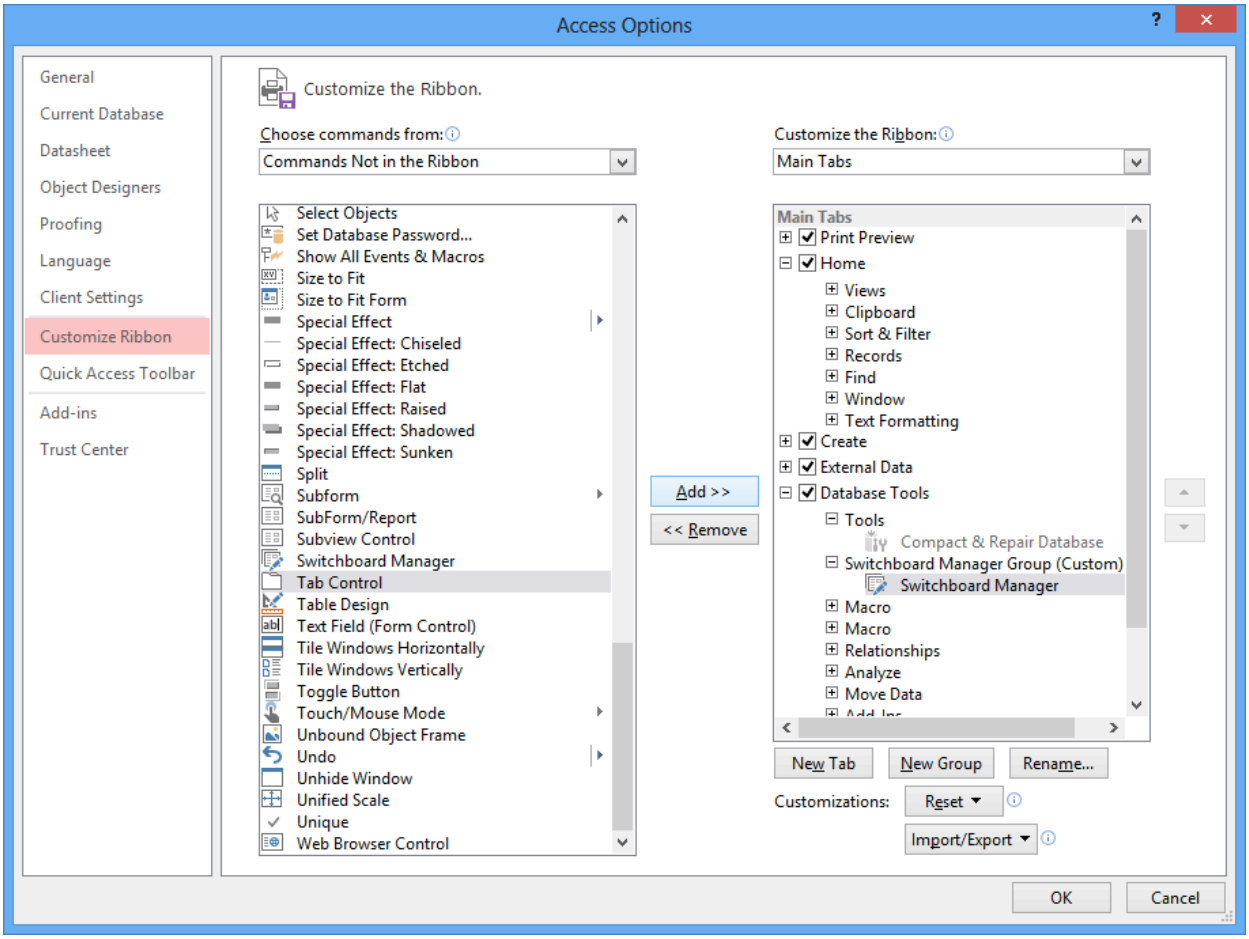


Figure 3: Switchboard Manager Item added to the Switchboard Manager Group

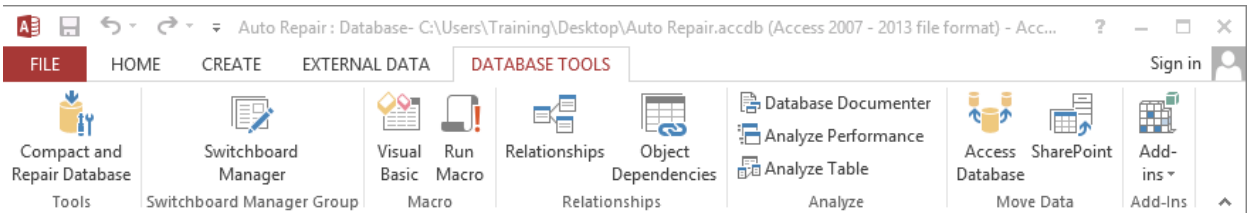


Figure 4: Switchboard Manager in the Database Tools Tab of the Ribbon

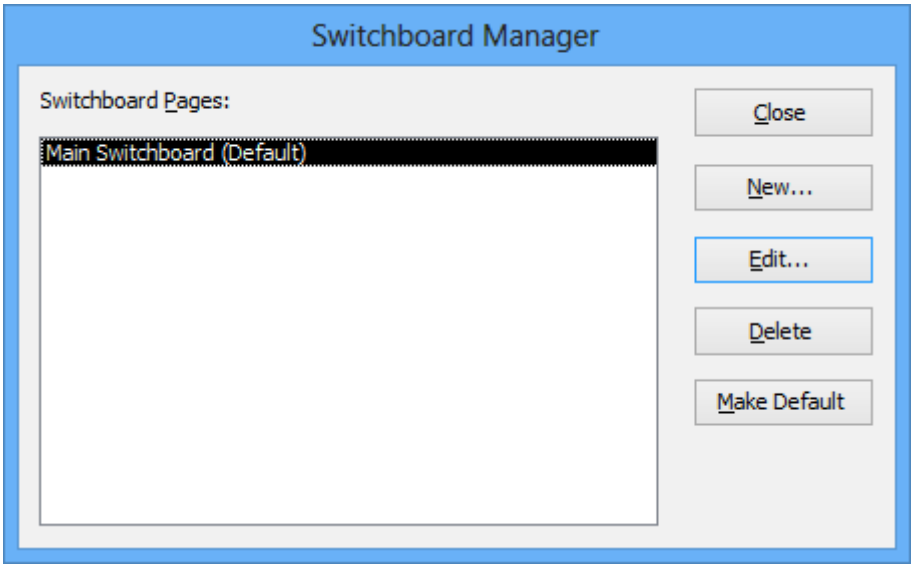


Figure 5: Switchboard Manager Window

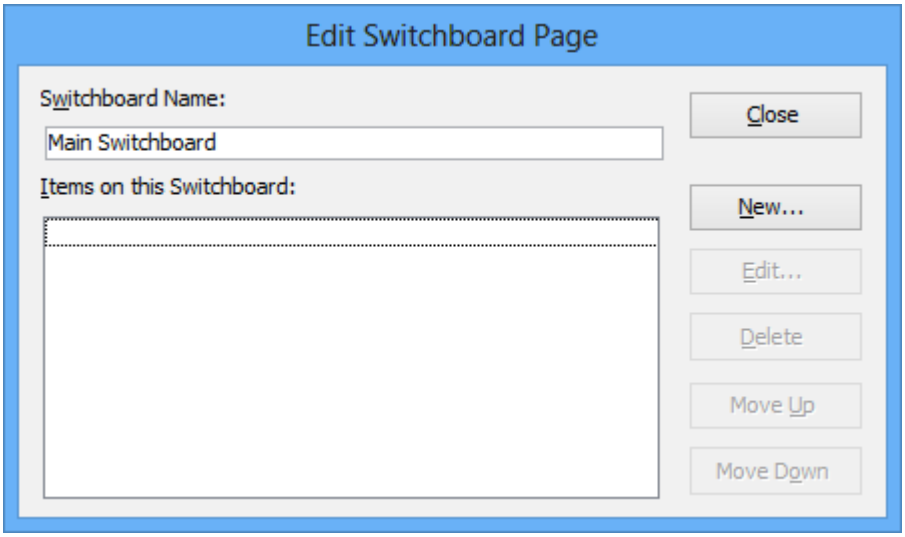


Figure 6: Edit Switchboard Window

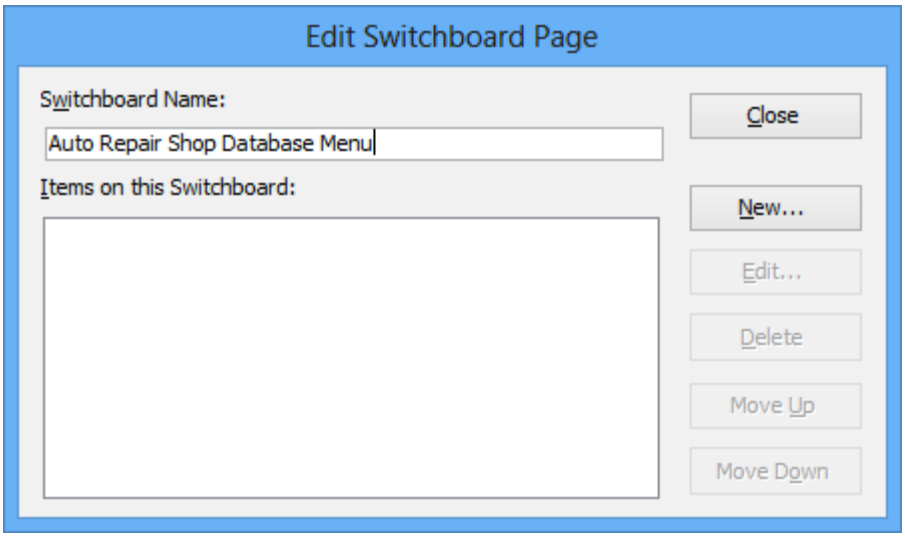


Figure 7: Switchboard Name

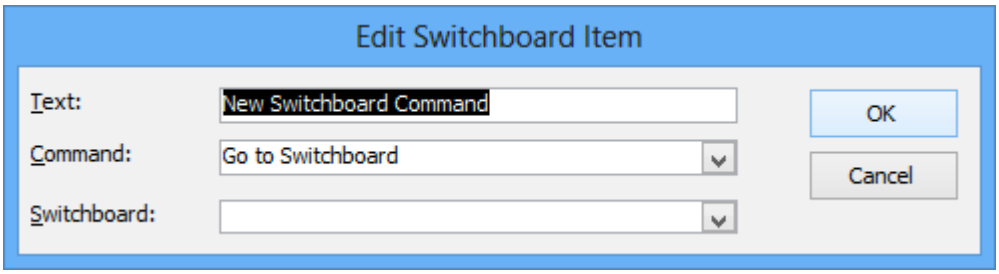


Figure 8: Default Edit Switchboard Item Window

6. Add Switchboard Items: Using the Edit Switchboard Item window, you should add command buttons that open the forms and reports.

- In the text area, type the caption for the first command button, "Enter Customer".
- In the command area, click the arrow and select "Open Form in Edit Mode". Note that the last area in the window just changed from "Switchboard" to "Form".
- In the form area, click the arrow and select the "Customer" form (Figure 9).
- Click **OK** to return to the Edit Switchboard window. This first item is now shown in the switchboard items list (Figure 10).

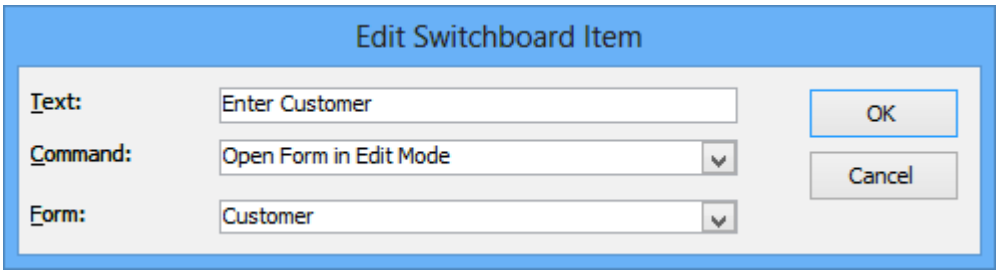


Figure 9: Modified Edit Switchboard Item Window

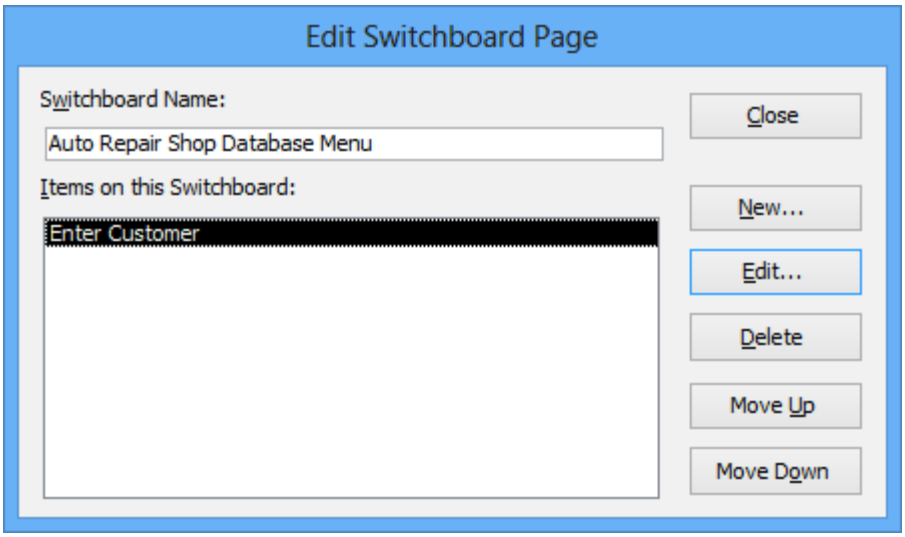


Figure 10: Item Added to the Switchboard

7. Add Remaining Command Buttons: In the Edit Switchboard window, click **New** to open the Edit Switchboard Item window again. Repeat step 4 until all of the items from Table 1 are added to the switchboard.

8. Edit or Delete: If you need to edit or delete an item in the Edit Switchboard window, click the item in the list and then click **Edit** or **Delete**. If you want to rearrange items, click the item and then click **Move Up** or **Move Down**. If you have already returned to the Navigation pane, then select **Database Tools → Switchboard Manager**. When the Switchboard window opens, click **Edit…** to open the Edit Switchboard window and follow the above instructions.

9. Close Switchboard: When you are finished, close the Edit Switchboard window and the Switchboard Manager window.

10. Modify Form in Design View: In the Forms section of the Navigation pane, select the *Switchboard* form. Right click on the item and select **Design View**. Select the label in the form heading areas and open its Properties window. Change the *Caption* property to "Auto Repair Shop". If you do not care for the default color (green) background, simply select both parts (there are two parts) and delete them.

11. View the Menu: Toggle to form view. Your menu should appear as in Figure 11. Close the menu and save the changes.

Table 1: Command Buttons in the Switchboard Form

Text	Command	Form or Report
Repair Orders	Open Form in Edit Mode	*RepairOrder2* (Form)
Edit Parts List	Open Form in Edit Mode	*Part* (Form)
Enter Vehicle	Open Form in Edit Mode	*Vehicle* (Form)
Complete Part Report	Open Report	*Part Report3* (Report)
Monthly Part Report	Open Report	*Part Report4* (Report)
Parts Expense Report	Open Report	*Parts Expense Report* (Report)

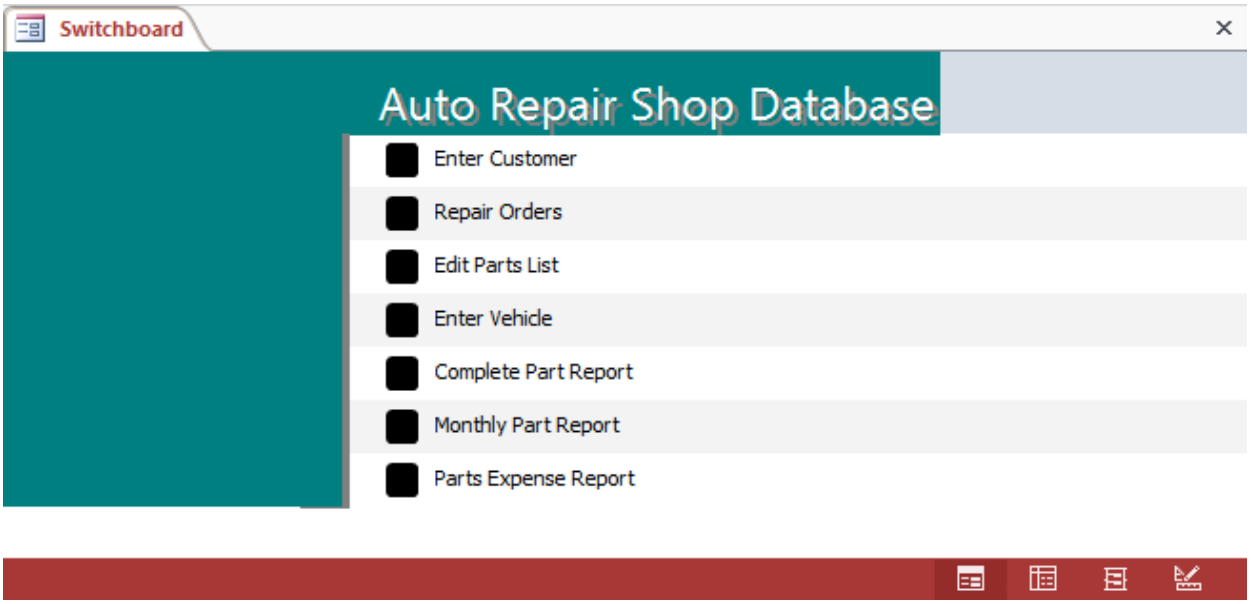


Figure 11: Auto Repair Shop Menu

8.1.2 Displaying the Switchboard Form at Startup

After designing a switchboard, you want to make it appear when your database opens. In addition, you want to hide the Navigation pane to prevent users from viewing it. After hiding the Navigation pane, you can make it reappear by clicking **F11**. Follow the steps below to display the *Switchboard* form when the database opens and hide the Database window by default:

1. Open the Access Options Window for Current Database: Click **File → Options** to open the Access Options window. In the Access Options window, click the **Current Database** command.

2. Modify the Current Database Window: In the *Display Form* field (Figure 12), select "Switchboard". Uncheck the *Display Navigation Pane* box to clear it (Figure 13). You do not want the Navigation pane to appear. Click **OK** to close the window. Note that this change becomes effective the next time you open the database.

    - To use the Navigation pane for making modifications to database objects, type **F11**. This action restores the Navigation pane to the left of the Switchboard.

The Access Options window supports other customization choices for database applications. You can control the full menus, shortcut menus, toolbars, and shortcut keys available when a database opens. For more details about the options available in the window, use the search term “Access Options” in the help documentation.

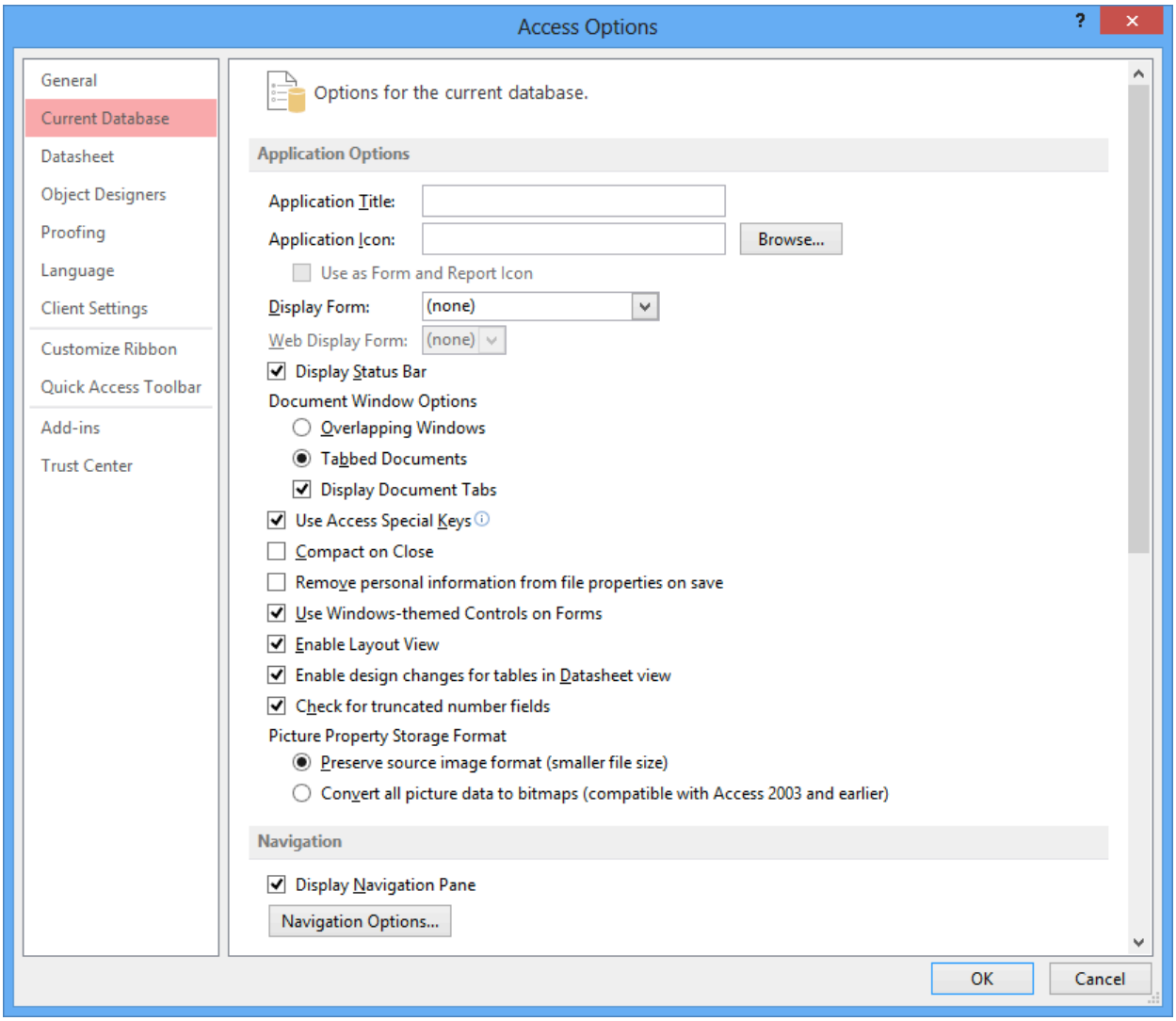


Figure 12: Current Database Window

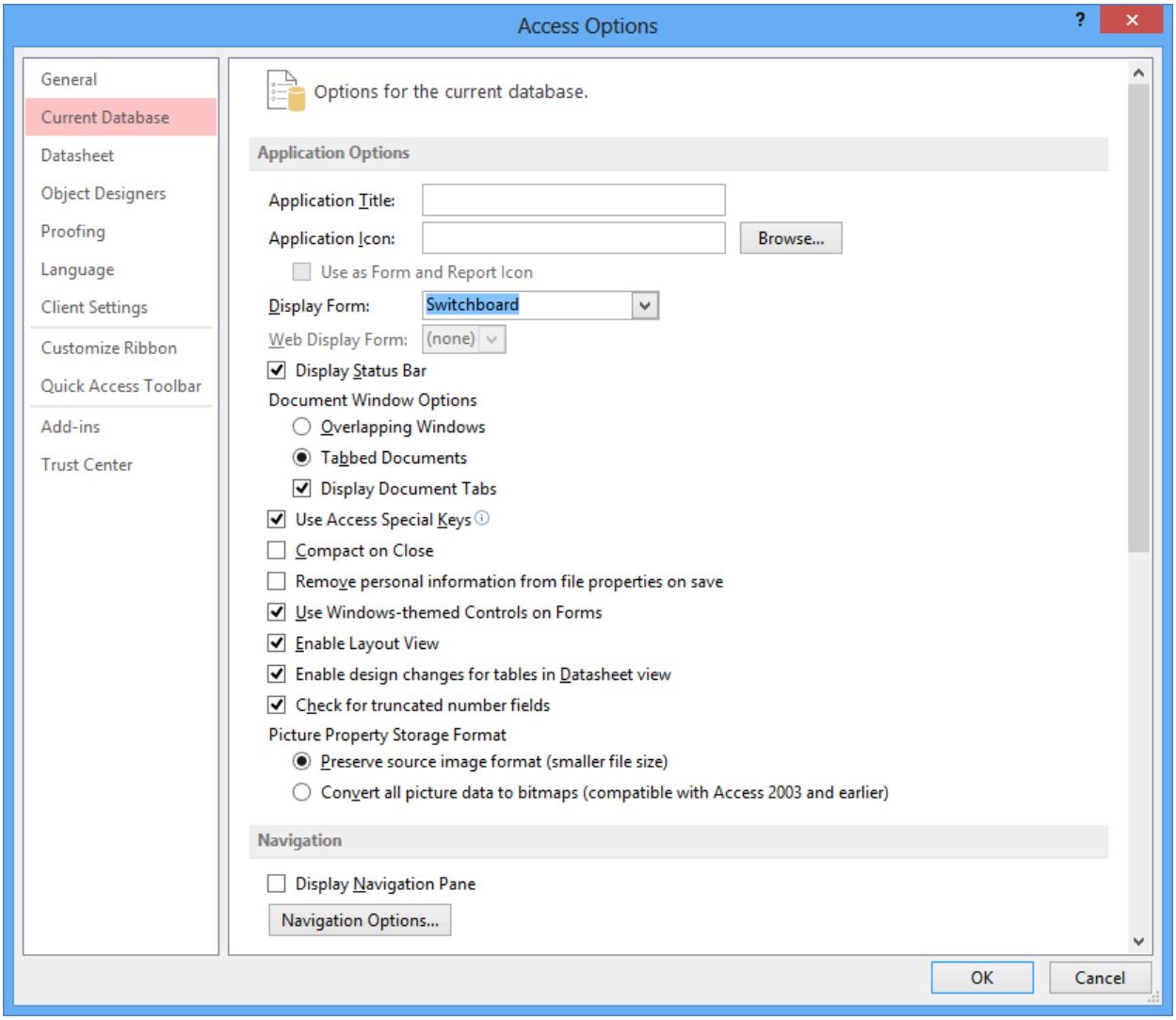


Figure 13: Current Database Window after Modifications

8.1.3 Creating a Navigation Form

Since a switchboard form cannot be used for deployment of Access databases on a Sharepoint server, the switchboard form is no longer the preferred navigation aid for Access databases. Access 2013 supports navigation forms with tabs for a more contemporary Web interface.

This section demonstrates a simple navigation form for the Auto Repair Database. Navigation forms are easy to create and customize. Access provides a collection of prebuilt navigation templates so you should not need to create your own templates. You will create a navigation form containing the same objects as the switchboard form.

1. View Navigation Form Templates: Click **Create → Navigation** in the Forms group to see the prebuilt navigation form templates (Figure 14). Access provides navigation forms with horizontal tabs, vertical tabs (left or right), multiple-level horizontal tabs, and mixed horizontal and vertical tabs.

2. Create a Horizontal Navigation Form: Select **Create → Navigation → Horizontal Tabs** to create an empty navigation form with horizontal tabs. Close the Field List window because it will not be used to demonstrate navigation form creation. Figure 15 shows the empty navigation form in layout view. The top part of a horizontal navigation form contains tabs and the bottom part contains the object pane. Each tab contains a separate object pane.

3. Use the Customer Form in the Object Pane of the First Tab: Dragging and dropping objects from the Navigation pane is a simple way to specify the object pane of a tab. With the navigation form in layout view, drag the *Customer* form from the object pane and drop it on the tab. The *Customer* form then displays in layout view (Figure 16). Note that a second tab has been automatically added.

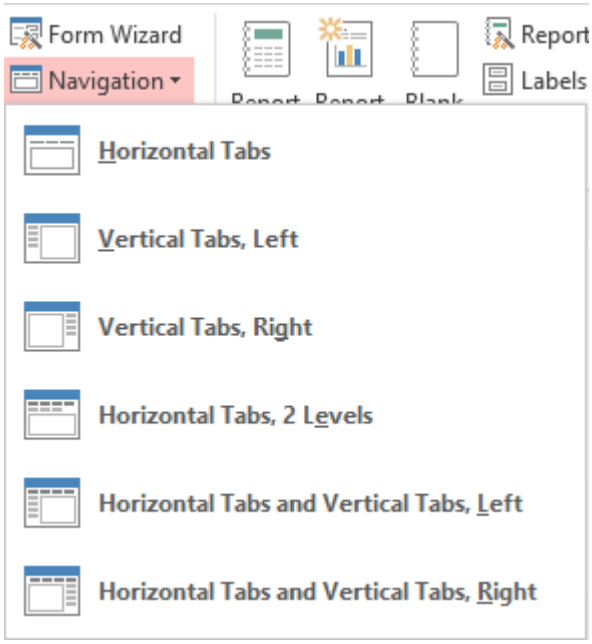


Figure 14: Navigation Templates in the **Create → Navigation** menu

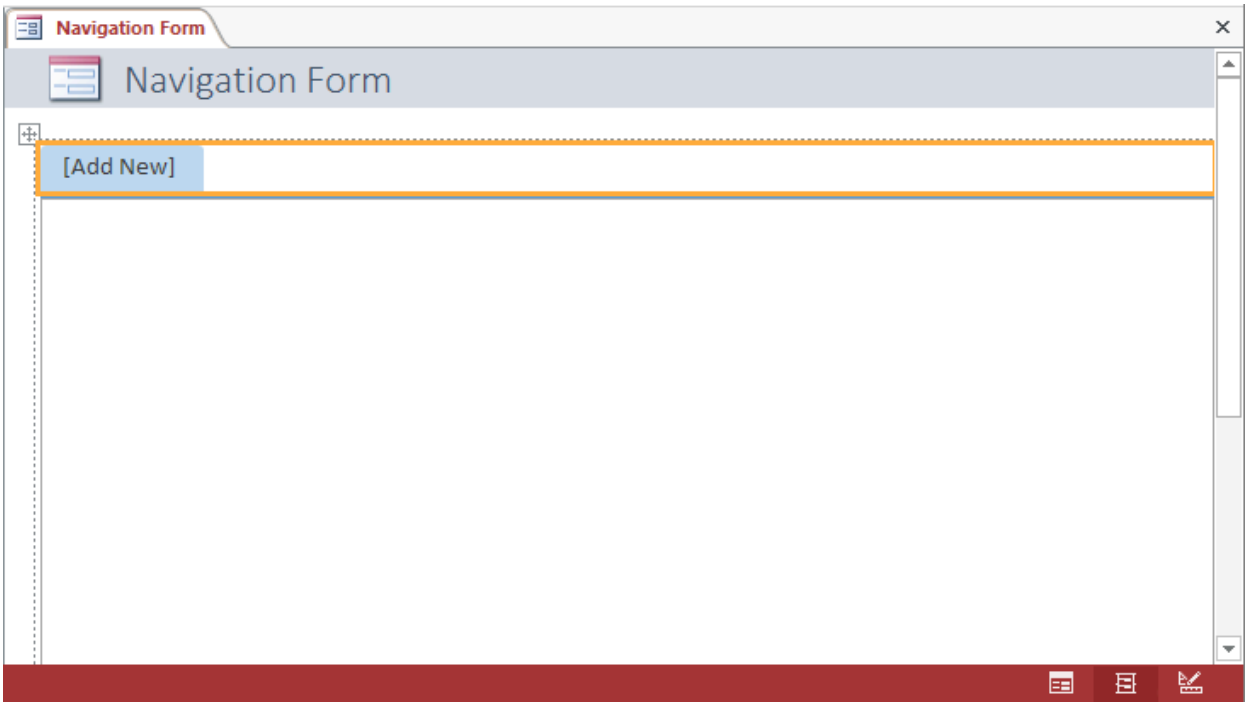


Figure 15: Empty Navigation Form in Layout View with Horizontal Tabs

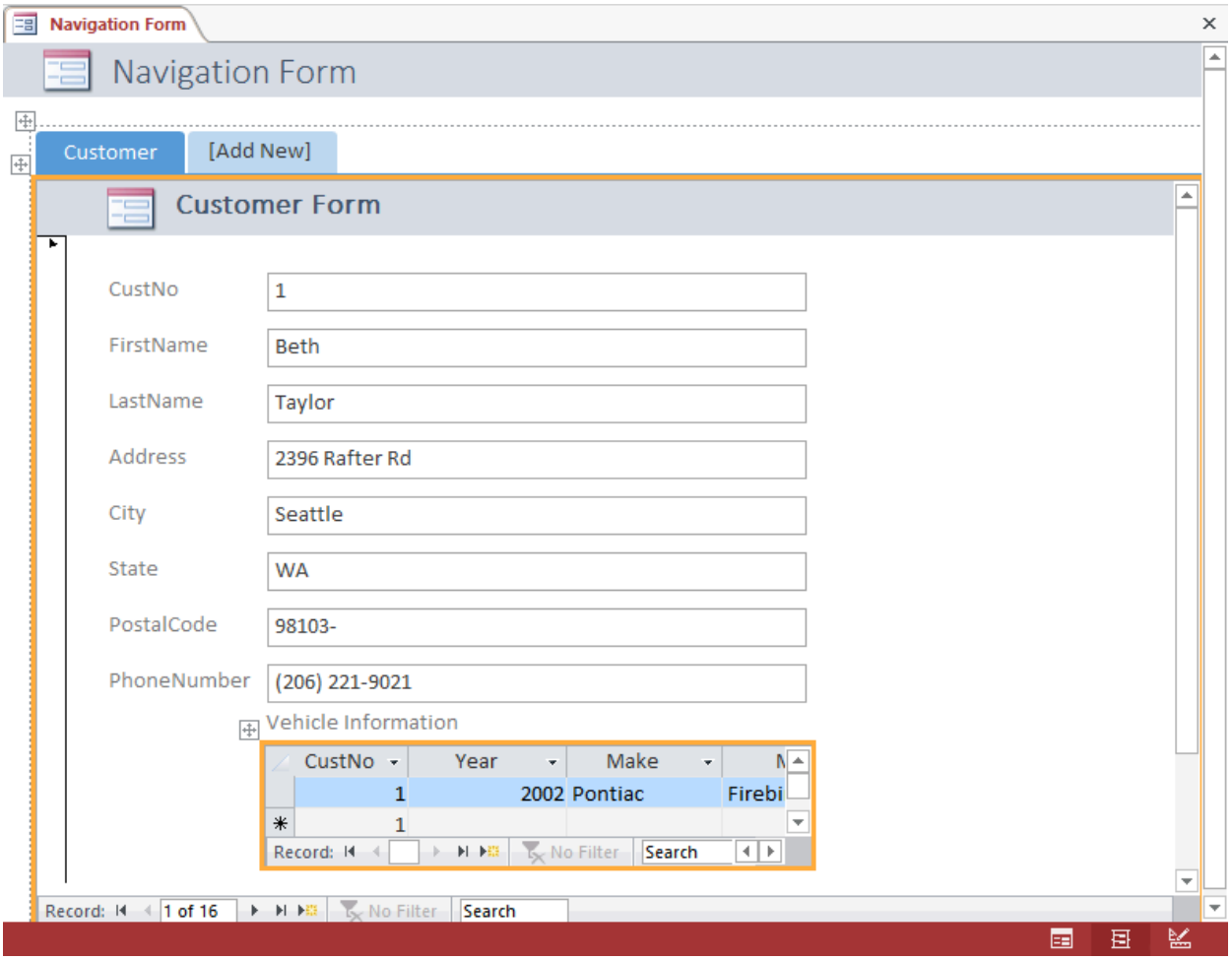


Figure 16: Layout View with Customer Form in the First Tab

4. Add the *RepairOrder2* Form: Drag and drop the *RepairOrder2* form from the object pane to the second tab. Rename the second tab by clicking inside the tab until a cursor appears. Rename the tab as "Repair Order Form". Changing the tab name layout view is identical to changing the caption property of the tab. Figure 17 shows the revised navigation form. You may need to stretch the length of the tab in layout view to make the entire tab name visible.

5. Complete the Navigation Form: Add the objects in Table 2 to the navigation pane. Each object should be dragged from the object pane to a tab in the navigation form using layout view. Change the caption of each tab as shown in Table 2. Note that you may need to expand the length of some tabs so that the entire caption fits.

Table 2: Remaining Objects to Add to the Tabs of the Navigation Form

Form or Report	**Caption**
Part (Form)	Parts List
Vehicle (Form)	Vehicle Form
Part Report3 (Report)	Complete Part Report
Part Report4 (Report)	Monthly Part Report
Parts Expense Report (Report)	Parts Expense Report

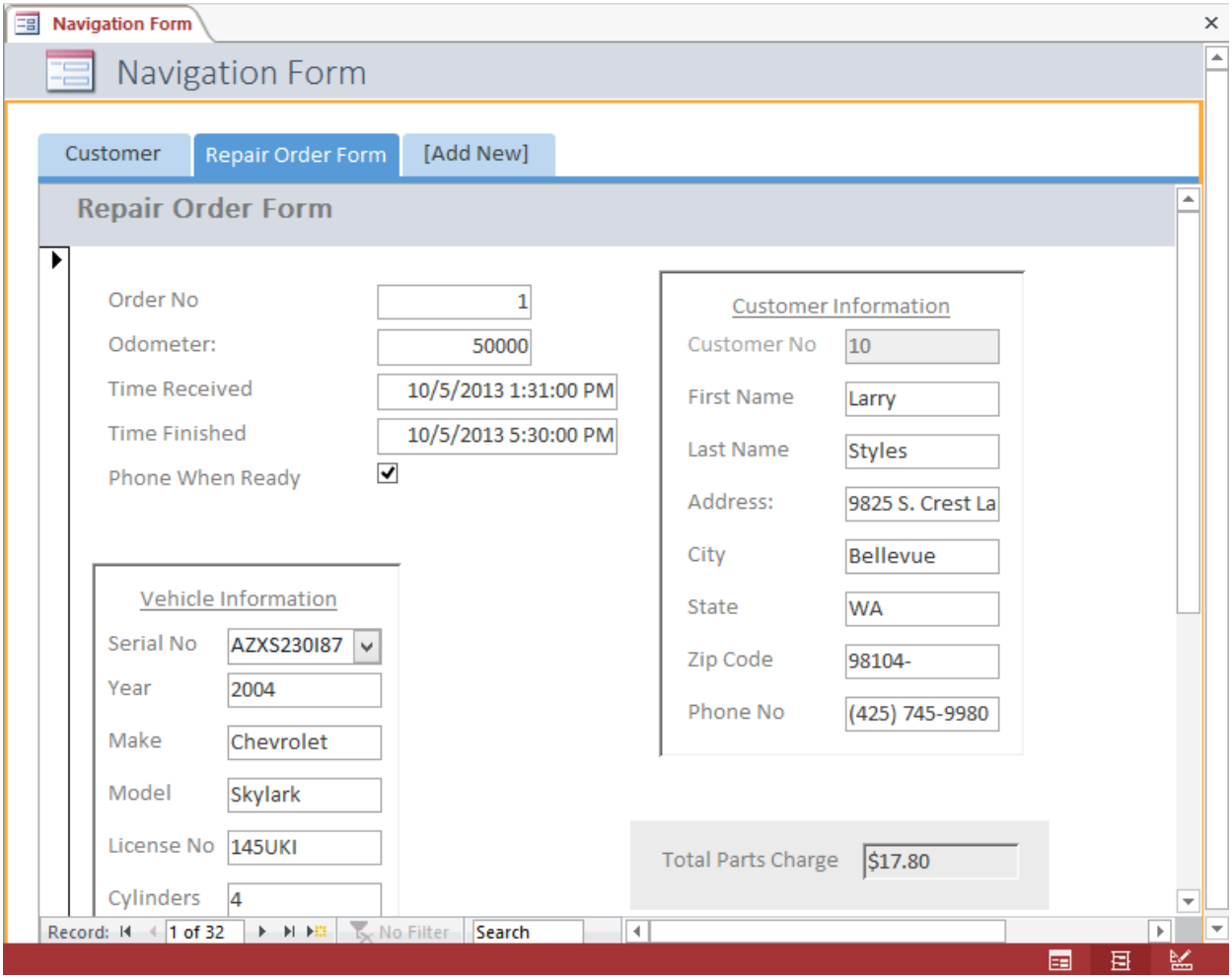


Figure 17: Layout View with the RepairOrder2 Form in the Second Tab

6. Change the Form Caption Name: Change the caption of the form to "Auto Repair Navigation Form" using the Property Sheet.

7. Change the Form Heading: Change the label contents in the heading of the navigation form to "Auto Repair Database".

8. Save the Form: Save the form as "Auto Repair Navigation Form".

9. View Completed Navigation Pane: Switch to form view to see the completed navigation form (Figure 18) with an object added to the object pane of each tab.

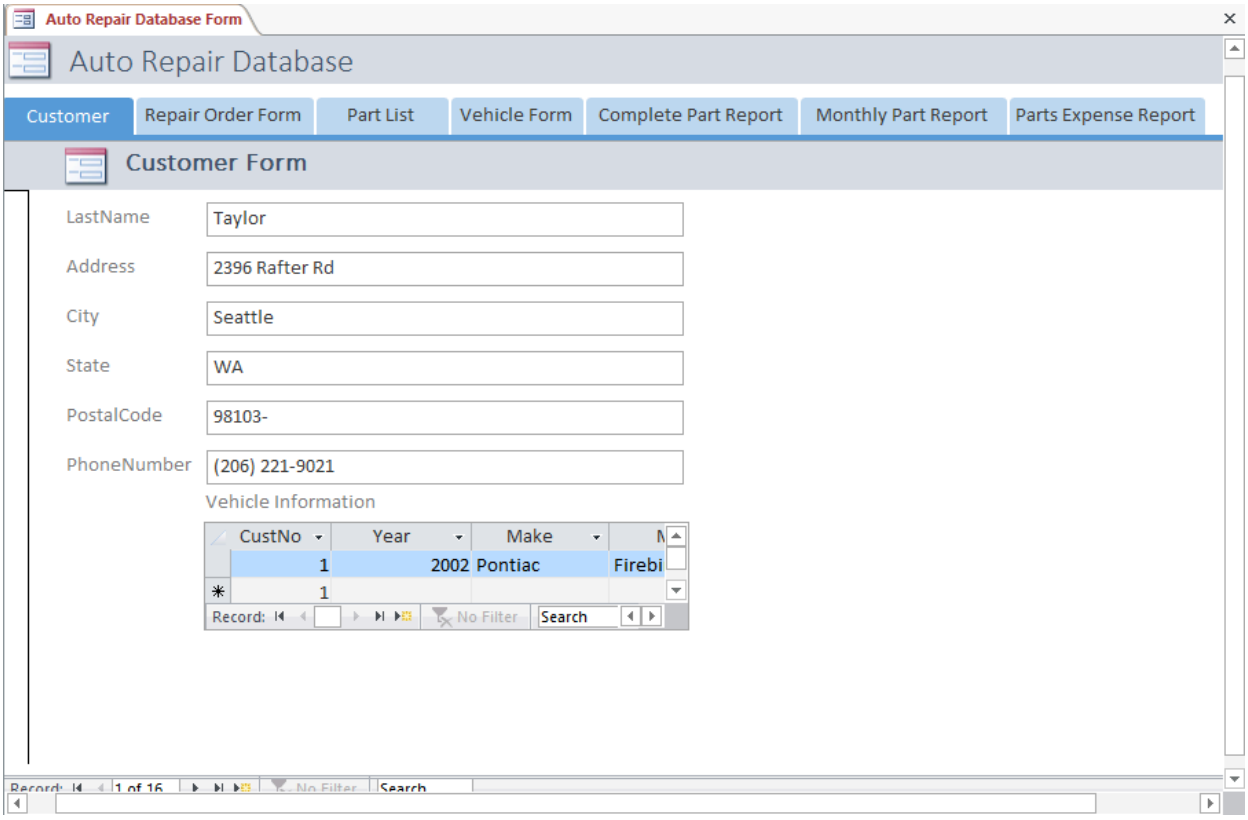


Figure 18: Completed Navigation Form in Form View

Navigation forms can be used at startup just like switchboard forms. To make the *Auto Repair Navigation Form* as the startup form, follow the instructions in Section 8.1.2. If you want to deploy your database to a Sharepoint server, you should select the navigation form in the Web Display Form list in Figure 12.

You can customize the appearance of the tabs using the Control Formatting group in the **Format** tab of the Ribbon. You can change the style (**Format → Quick Styles**), shape (**Format → Change Shape**), fill (**Format → Shape Fill**), outline (**Format → Shape Outline**), and effects (**Format → Shape Effects**).

Despite the effort by Microsoft to create a navigation style consistent with web navigation, navigation forms have some limitations for web deployment. Navigation tabs lack the pull-down style common on typical web navigation. Navigation tabs do not have dynamic display common for commercial websites. It may be possible to add these capabilities with some event coding, but the coding would require some effort.

8.1.4 Adding Command Buttons to Forms

A switchboard or navigation form provides navigation when a database opens. To provide navigation when using a particular form or report, you can add command buttons. Access provides the Command Button Wizard to help you create buttons that initiate actions when clicked. The wizard can generate code to perform a variety of actions such as opening database objects (forms and reports) and navigating among records. You will use the Command Button Wizard to generate code to open forms and reports. The following instructions guide you to adding command buttons to several forms that you previously created.

8. Open the Customer Form: Open the *Customer* form in Design view.

9. Add the First Command Button: In the Controls group of the **Design** tab, click the **Control Wizards** tool to activate a Control Wizard that will open when the appropriate control tool is selected. Next, click on the **Command Button** tool in the Controls group. A small box icon appears with the crossbar cursor. Place a command button directly under the subform as shown in Figure 19. After releasing the mouse button, the Command Button Wizard appears automatically.

10. Select Command Button Action: The first wizard window (Figure 20) asks you to choose a category and an action. In the *Categories* column select "Form Operations" and in the *Actions* column select "Open Form" and click **Next**.

11. Select Form to Be Opened: The next window asks you to identify the form to open (Figure 21). Select "Vehicle" and click **Next**.

12. Show Form Records: The third wizard window asks if a specific record is to be shown when the form is open or if all records will be available. Select the second choice (Figure 22), "Open the form and show all the records", and click **Next**.

13. Command Button Format: You are asked if you want text or a picture to appear on the button. Select "Text" and type "Vehicle Form" (Figure 23). Click the **Next** button to continue.

14. Name the Command Button: This final wizard window (Figure 24) asks you to name the command button. Type "OpenVehicle" and click **Finish**. A new command button appears on the *Customer* form (Figure 25). If the text in the button appears too light, change the *Font Color* property for the button to "Text Dark".

15. Try the Command Button: Toggle to form view and click the button to see if the *Vehicle* form opens. When you close the *Vehicle* form, you are returned to the *Customer* form. Toggle to Design view.

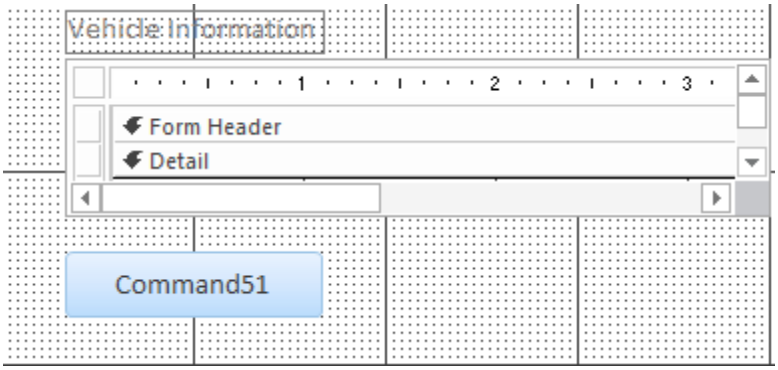


Figure 19: Initial Layout of the Command Button

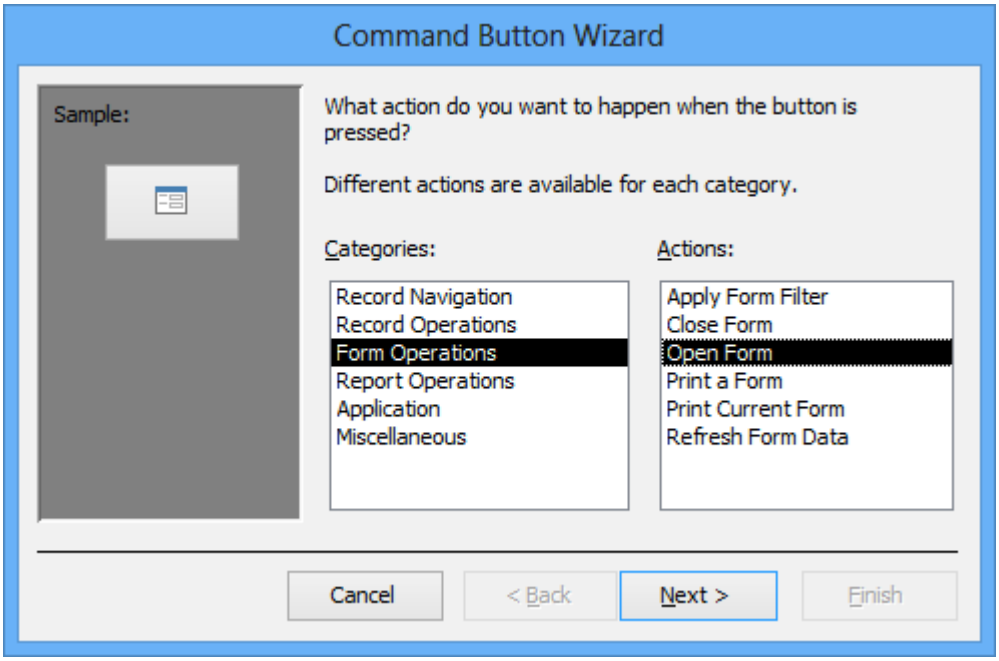


Figure 20: First Window of the Command Button Wizard

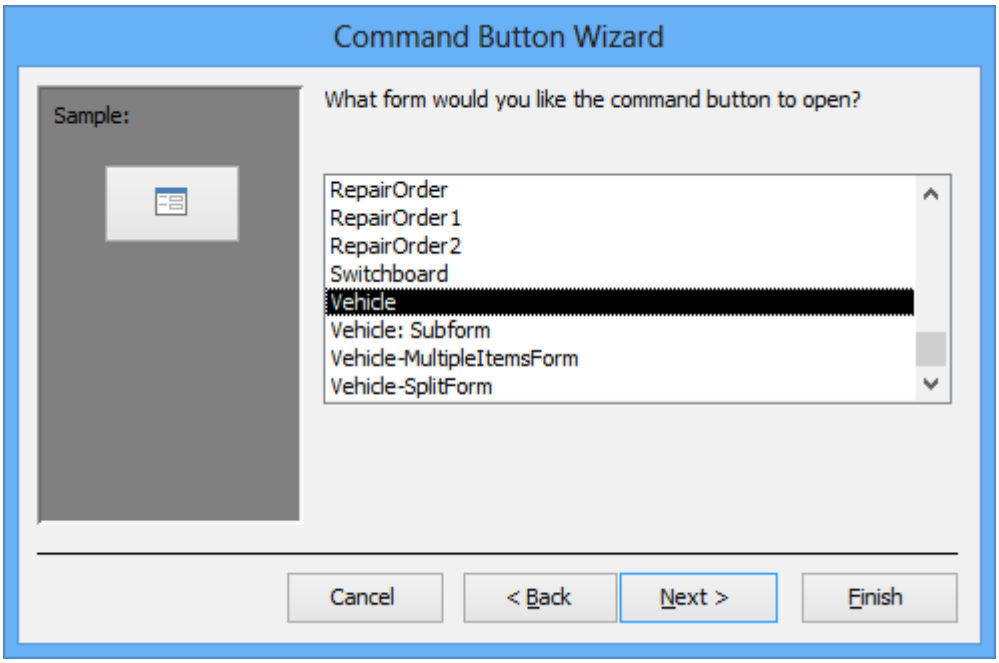


Figure 21: Second Window of the Command Button Wizard

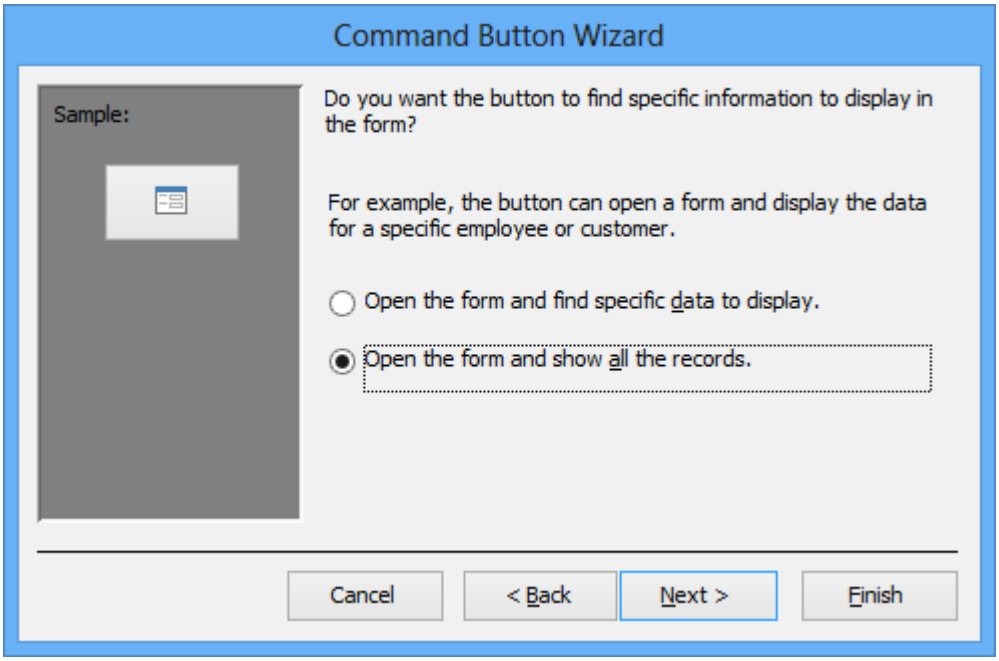


Figure 22: Third Window of the Command Button Wizard

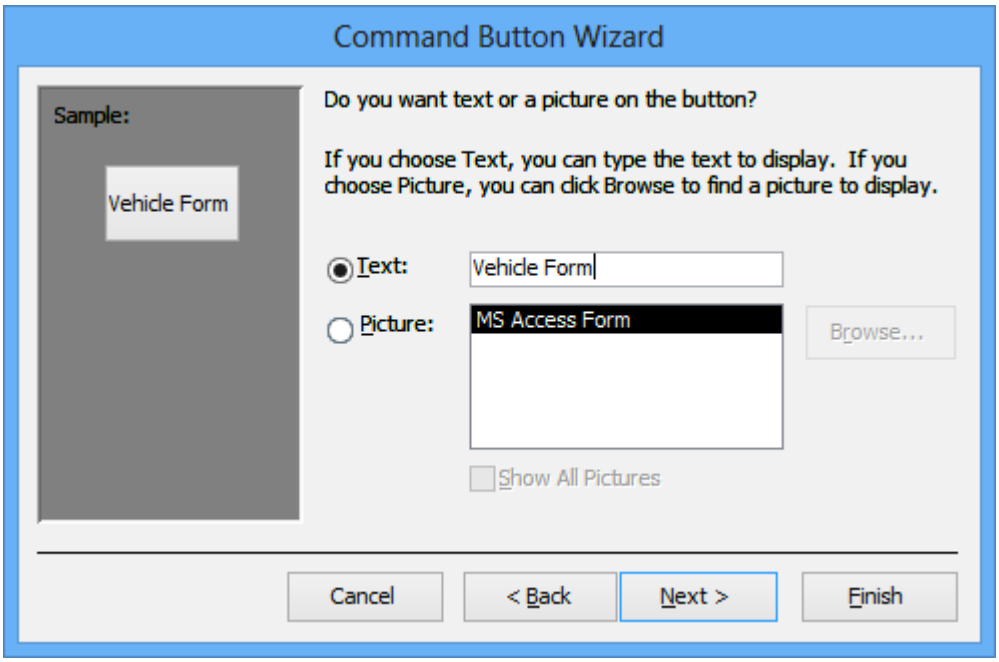


Figure 23: Command Button Text Entry

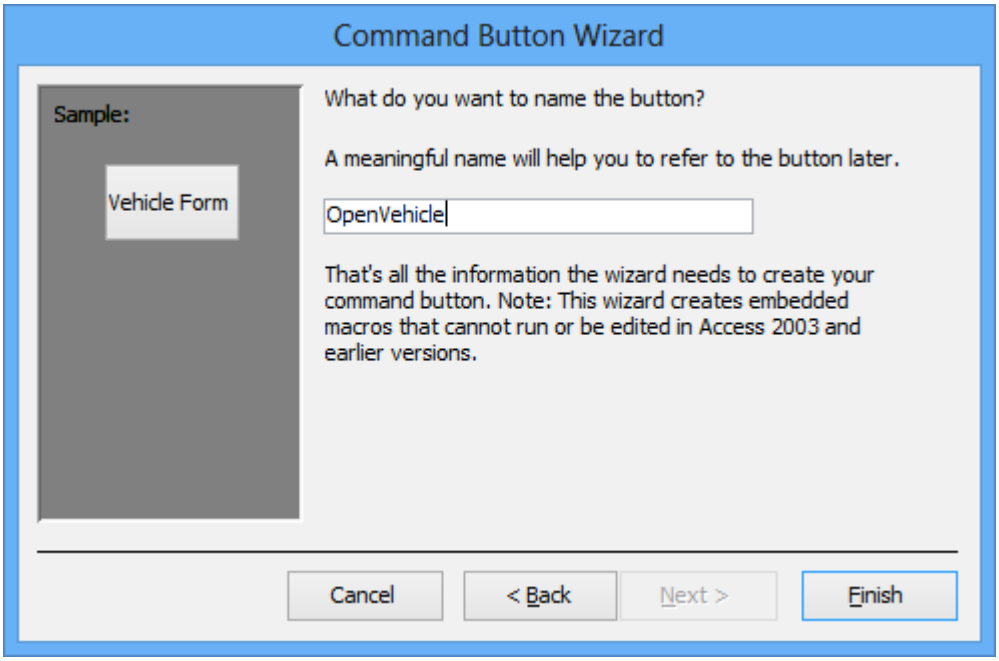


Figure 24: Final Window of the Command Button Wizard

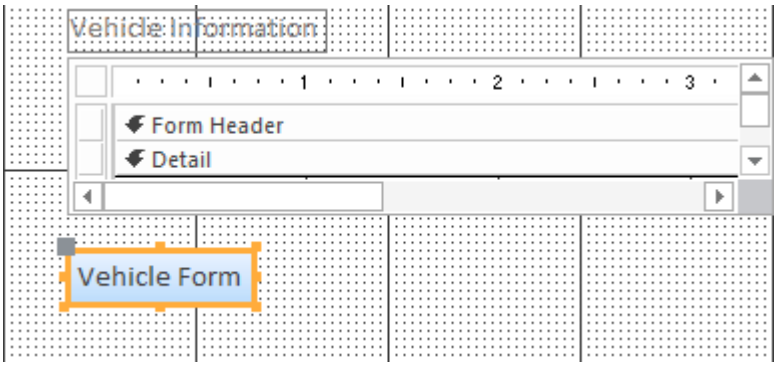


Figure 25: Customer Form with a New Command Button

16. Add the Second Command Button: Follow steps 2–8 above and place a new button next to the *Vehicle Form*. Make the button open the *RepairOrder2* form. The caption on the button should be "Repair Orders" and the name should be "OpenRepairOrd". If the button is not the same size as the *Vehicle Form* button, adjust it after you finish with the wizard.

17. Add the **Return to Menu** Command Button: Place a third button next to the *Repair Orders* button. Make the button open the *Switchboard* form (choose "Form Operations" and "Open Form" in the first wizard window). The caption on the button should be "Return to Menu" and the name should be "OpenMenu". When you are finished, toggle to form view (Figure 26) and try the buttons. When finished, close the form and save the changes.

18. Add Command Buttons to Other Forms: Refer to Table 3 to place command buttons on the forms by following steps 2–8 above. Note, for a command button to open a report, choose "Report Operations" and "Preview Report" in the wizard windows. When you are finished with each form, toggle to form view and try the buttons (Figures 27 to 29). When finished, close each form and save the changes.

Table 3: Command Buttons to Add to Forms

Form	**Object to Open**	**Button Caption**	**Button Name**
Vehicle	*Customer*	Customer Form	OpenCustomer
Vehicle	*RepairOrder2*	Repair Orders	OpenRepairOrd
Vehicle	*Switchboard*	Return to Menu	OpenMenu
RepairOrder2	*Customer*	Customer Form	OpenCustomer
RepairOrder2	*Vehicle*	Vehicle Form	OpenVehicle
RepairOrder2	*Part Report3*	Complete Part Report	OpenPartRept
RepairOrder2	*Part Report4*	Monthly Part Report	OpenMoRept
RepairOrder2	*Switchboard*	Return to Menu	OpenMenu
Part	*Part Report3*	Complete Part Report	OpenPartRept
Part	*Part Report4*	Monthly Part Report	OpenMoRept
Part	*Switchboard*	Return to Menu	OpenMenu

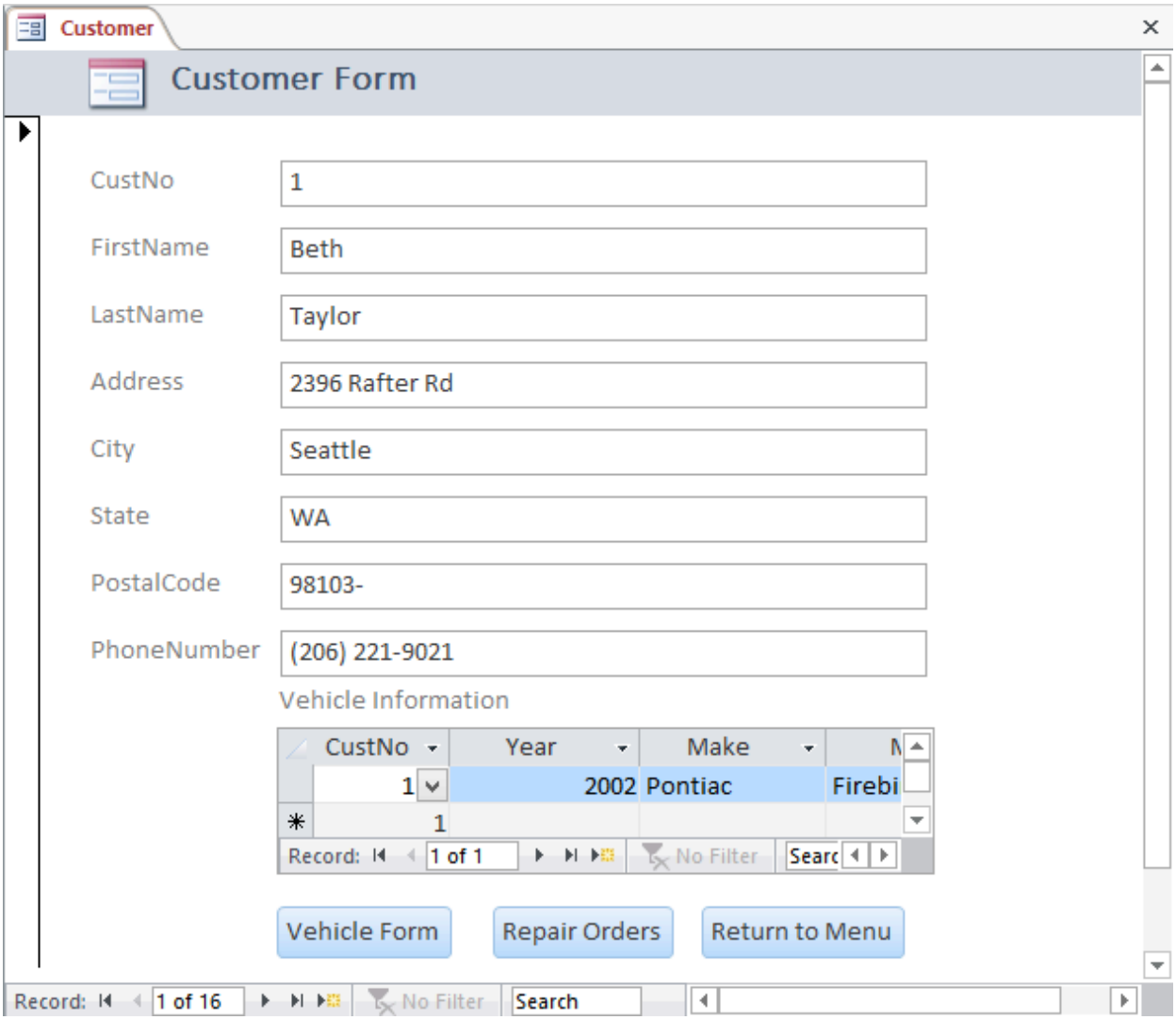


Figure 26: Revised *Customer* Form

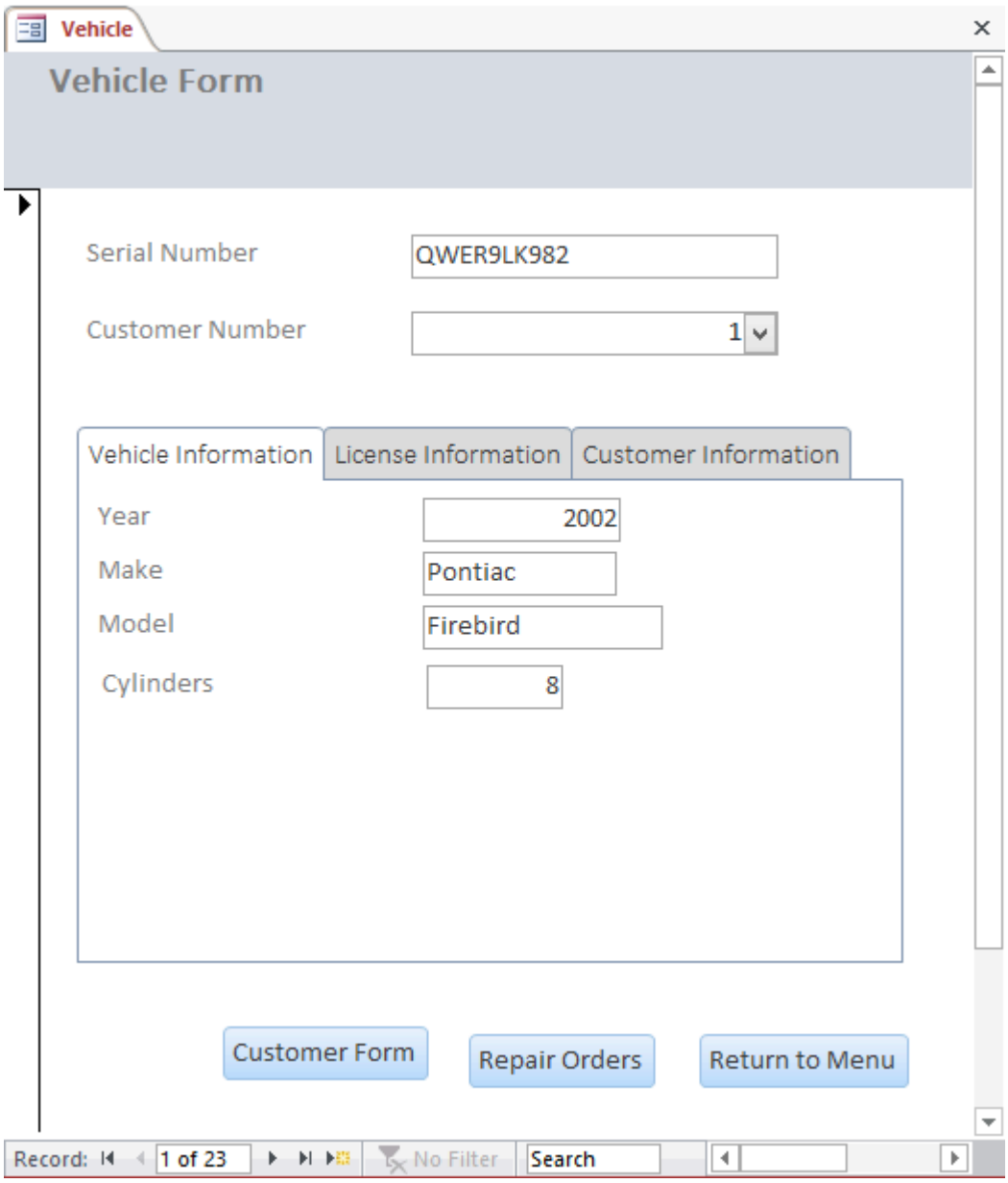


Figure 27: Revised *Vehicle* Form

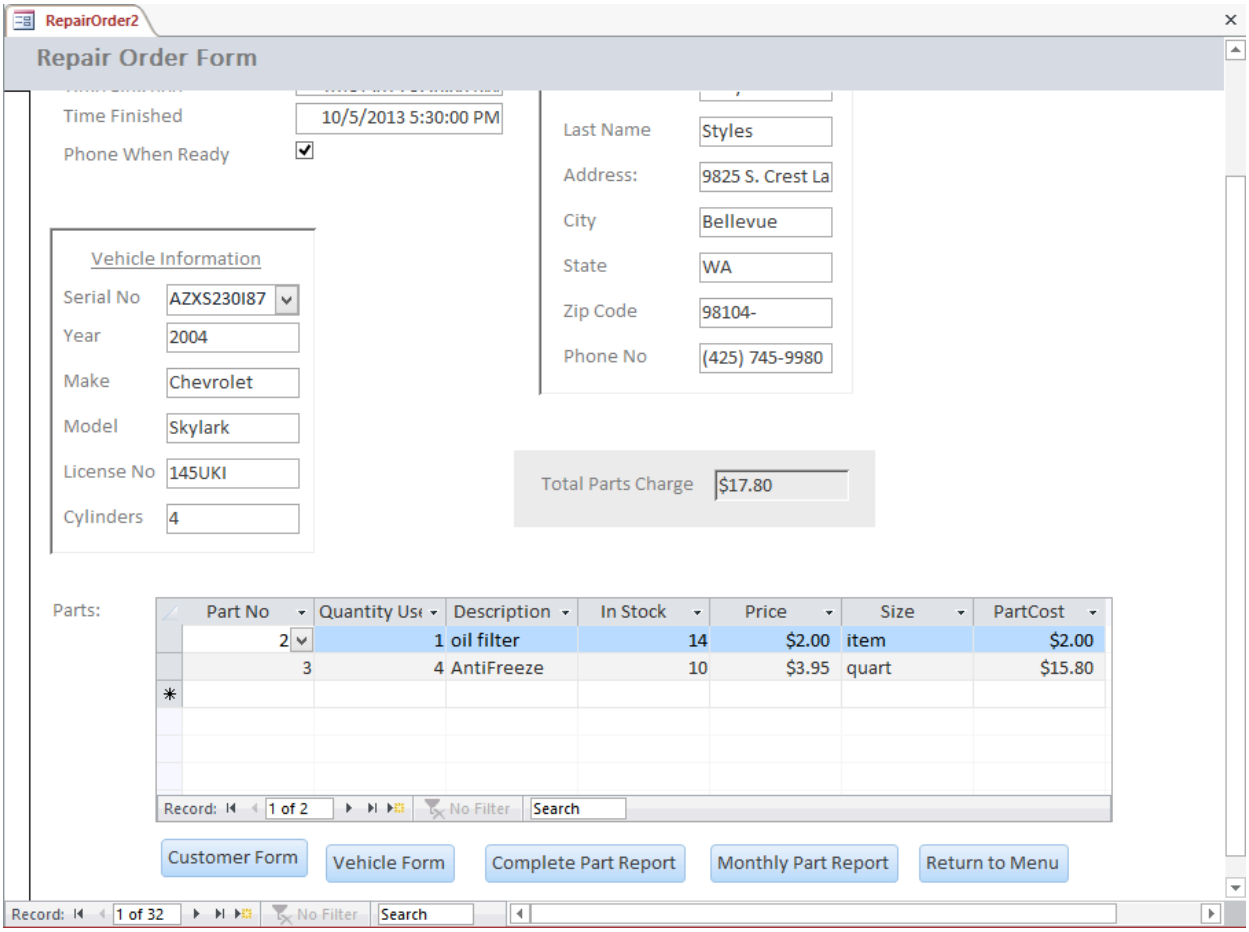


Figure 28: Revised *RepairOrder2* Form

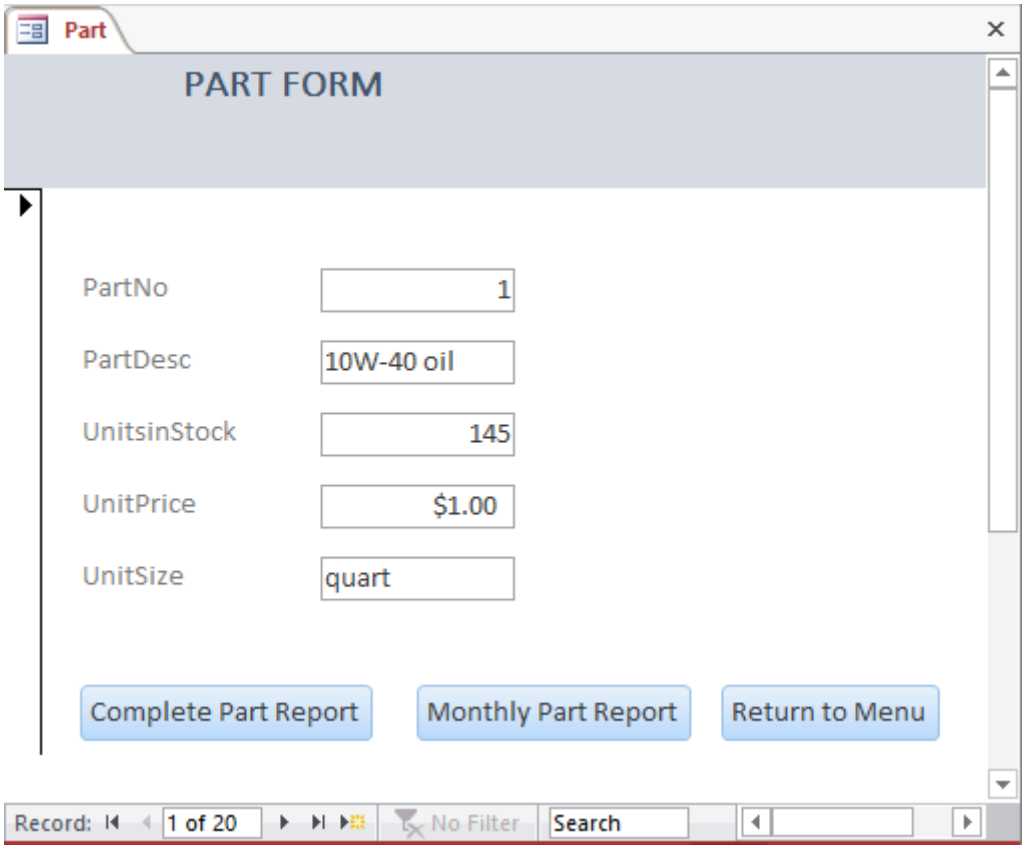


Figure 29: Revised *Part* Form

8.2 Working with Macros

Access supports two tools, macros and modules, to automate repetitive tasks and customize the way that forms and reports react to user actions. Macros are the older and simpler tool for automation. Although previous versions of Access discouraged macro usage, macros became prominent in Access 2007 due to security concerns. Macros are trusted because they are prevented from performing certain, potentially unsafe, operations. With enhancements to the macro development environment, extensions in macro specification, and support of business rules through data macros, macros have become the preferred tool for customization of database applications.

Access 2007 and 2010 provided a number of new capabilities for macros as summarized in Table 4. The Access 2007 enhancements were targeted towards security, error handling, and variable scope so that macros could replace procedures. The Access 2010 enhancements are targeted towards productivity through an improved macro designer and coding extensions. Data macros provide a capability similar to SQL triggers (see textbook Chapter 11) to extend the scope of macro events to tables.

Table 4: New Capabilities of Macros in Access 2007 and 2010

Feature	Brief description
Global temporary variable usage (2007)	Create and use global temporary variables in macro actions
Embedded macro (2007)	Stored in a property and a part of the object to which it belongs; Easily modified because each embedded macro is independent; Automatically prevented from performing certain, potentially unsafe, operations
Error handling macro actions (2007)	Handle errors more gracefully in macros
Macro designer improvements (2010)	Action catalog, Intellisense for expressions, keyboard shortcuts, copy/paste, reuse of existing macros
Coding extensions (2010)	More readable code logic and flow, nested if statements, new actions
Data macros (2010)	Event coding for table events similar to SQL triggers

This section guides you to formulate several macros of increasing complexity. Section 8.2.1 presents background on macros along with simple macros with parameters. Section 8.2.2 presents macros with conditions to refine when macros execute. Section 8.2.3 provides examples of data macros, a substantial new feature of Access 2010 and Access 2013.

8.2.1 Creating Simple Macros

A macro is a name given to a sequence of one or more actions that perform a common task. You formulate a macro by specifying its event, its actions, and the parameters or arguments for each action. Names for macros connected to events are implicit with the name derived from the object and event. To more easily manage macros, you can store them in a group. Since you will develop only a few macros in this chapter, the macro group feature is not used.

An event triggers or causes a macro to execute. You indirectly have used the *Click* event for common buttons through the Command Button Wizard. The wizard generated macro code that executes when a user clicks on a button. Determining the proper event for a control is not always so easy. Events apply to every kind of control including the form itself. Sometimes the same event applies to multiple controls. To determine the proper event for a macro, you may need to carefully study the meaning of events for available kinds of controls. The help documentation provides a detailed explanation of each event including when the event occurs and what kinds of actions are typically used with the event.

Overview of Macro Tools

As an introduction to the macro development environment, let's begin with an empty macro window. To start a new macro, choose **Create → Macro** in the Macros & Code group of the Ribbon. The starting point for a macro usually is to select the action in the drop-down list (Figure 30) appearing in the object creation pane. The Action Catalog pane (Figure 31) appears to the right of the macro object pane. To support macro development, the dynamic Macro Tools Design tab (Figure 32) provides tools, action expansion and collapsing, and action catalog manipulation. When you are finished viewing the empty Macro pane and tools, close the window.

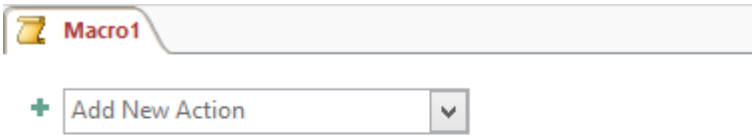


Figure 30: Add New Action List in the Empty Macro Pane

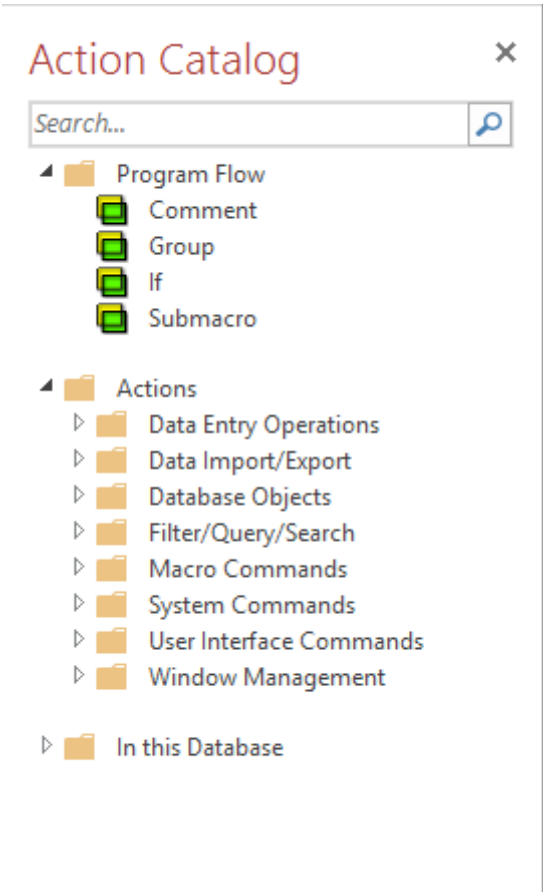


Figure 31: Action Catalog Pane

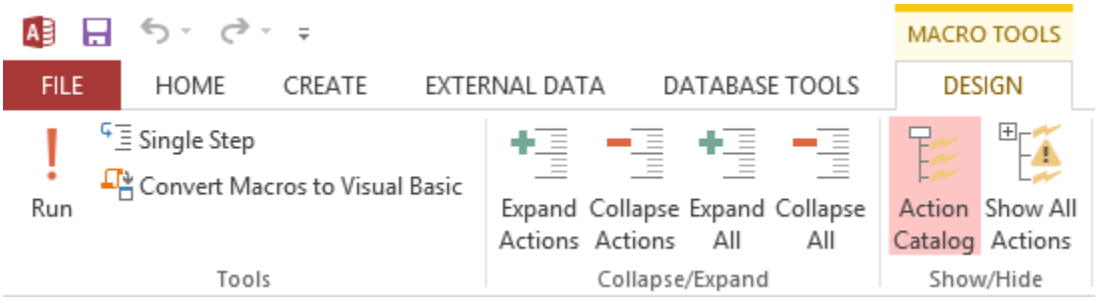


Figure 32: Dynamic Macro Tools Design Tab in the Ribbon

Formulating Your First Macro

The first macro refreshes the *Serial No*. combo box in the *RepairOrder2* form. Remember that this combo box uses a query that references a field on the main form. In Chapter 5, you had to manually requery the combo box by using the **Home → Refresh All → Refresh** item in the Records group of the Ribbon. This macro relieves a user from manually refreshing for each record in the main form.

In this macro, you will use the *On Current* event. The *On Current* event occurs when a new record obtains "focus" (becomes current) or when a form is refreshed. Making a record current means that the record has the ability to receive user input. The *On Current* event allows the *Serial No*. combo box to be automatically requeried when the customer number value changes on the main form. Access executes the macro just before the new record is displayed. Recall that the query for the *Serial No*. combo box uses the customer number value on the main form to restrict the vehicles displayed. Because the customer number field is read-only on the form, the only way to change the customer number value is to advance to a new record.

The following steps instruct you how to formulate a macro to requery the *Serial No*. combo box on the *RepairOrder2* form:

1. Open the *RepairOrder2* Form and Its Properties Window: On the Navigation pane, select the *RepairOrder2* form and open it in Design view. Open the Form section of the Property Sheet and click the **Event** tab.

2. Open the Macro Builder: Click in the *On Current* event property and click again on the ellipsis (**…**) button to open the Choose Builder window. Select **Macro Builder** and click **OK**.

3. Create the Macro: In the *Add New Action* list, click the arrow and scroll down to select "Requery". In the *Control Name* field below, type "SerialNo" as shown in Figure 33. Close the window and save the macro when prompted. Access does not allow you to provide a name for the macro so it is saved under the default name.

4. Try the Macro: Toggle to form view to see the effects of the macro. Click the *Serial No*. combo box to see the list of vehicles for the customer on the main form. To see the effect of the macro, advance to a record with a different customer number. Click the *Serial No*. combo box to see that the list reflects the vehicles owned by the current customer. Close the *RepairOrder2* form and save it when prompted.

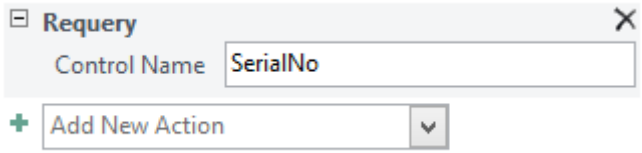


Figure 33: Control Name for Requery Action

You also can also use a macro with the *RunCommand* action and the "Refresh" argument instead of the *Requery* action and "SerialNo" argument. For refreshing combo box queries in subform fields, the *Requery* action causes a run-time error. The *RunCommand* action works well for subform fields, however.

Formulating a Macro with a More Complex Action

Many macros use more complex actions than the *Requery* action. More complex actions such as the *OpenForm* action involve a number of parameters. The *OpenForm* action uses parameters that specify the form to open, the view of the form when opened, a filter name, a condition to restrict the form's records, the data mode of the form, and the window mode. To facilitate macro development, Access provides default values for many parameters. You are encouraged to understand the meaning of each parameter to ensure that default values are appropriate.

The next macro involves a command button on the *RepairOrder2* form. You will replace the current *Click* macro in the **Customer Form** command button with a macro. The macro for this button opens the *Customer* form when clicked by the user. To customize the action of this button, you will revise the macro to retrieve a specific customer's record when the *Customer* form opens, rather than retrieving all customer

records. However, if there is no record for a customer (a null value), such as for a new customer, then all records will still be displayed. Follow the instructions below to formulate this macro:

1. Open the *RepairOrder2* Form: Select the *RepairOrder2* form and open it in design view. Double-click the button with the caption "Customer Form" to open its Property Sheet. Click the **Event** tab to see the *On Click* property.

2. Open a Macro Tab: Because you had previously defined an event macro using the command button wizard, you need to revise it rather than create a new macro. To revise the event macro, click on the ellipsis (**…**) button for the *OnClick* event. A new tab opens as shown in Figure 34.

3. Revise Macro: Click on text of the macro to open it. After clicking on the text, you should see the expanded Macro details (Figure 35). Revise the parameters of the expanded macro as listed in the following bulleted list.

    - Where Condition: Click inside this area and then click the Magic Wand( ) icon to open the Expression Builder window. Type the following condition in the Expression Builder window, then click **OK**:

[Forms]![RepairOrder2]![CustNo]=[CustNo] Or [Forms]![RepairOrder2]![CustNo] Is Null

    - Data Mode: Select "Edit" from the list.
    - Close Tab: When you are finished with the revisions, the revised macro should appear as shown in Figure 36. Close the tab and save the changes when prompted.

4. Try the Macro: Toggle to form view. When you click the **Customer Form** command button, the record with the current customer number should appear when the *Customer* form opens. Return to the *RepairOrder2* form to test the other condition with a null customer number. Place the *RepairOrder2* form in data entry mode using **Home → New** in the Records group. Click the command button again to open the *Customer* form with all records available. Close and save the *RepairOrder2* form when you are finished.

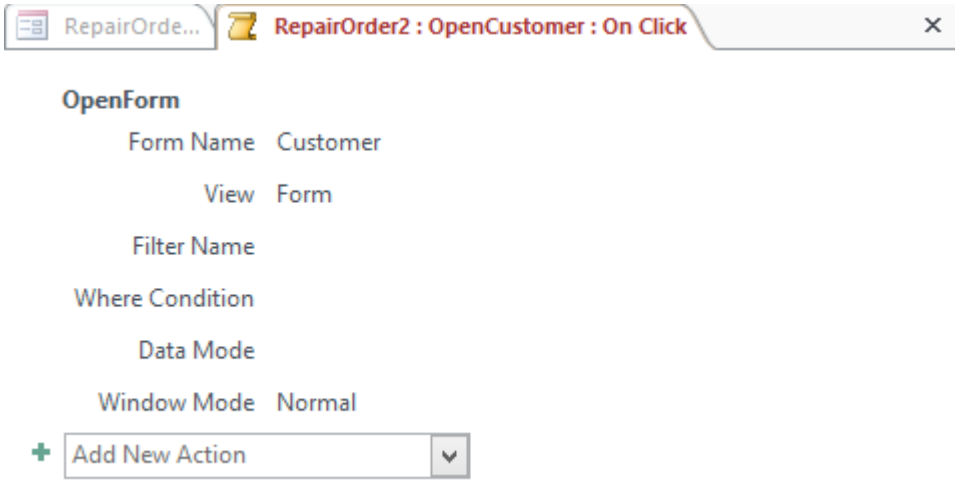


Figure 34: Event Macro for the Click Event

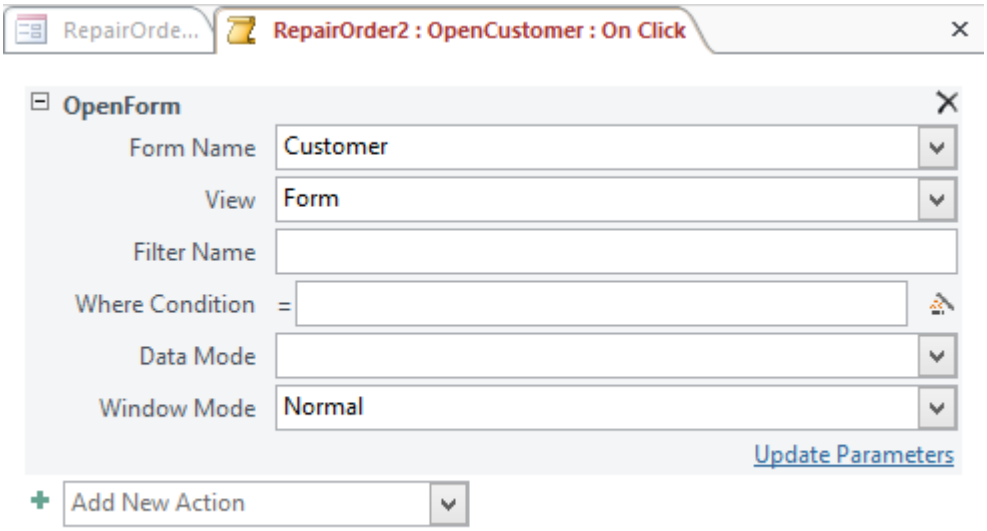


Figure 35: Expanded Event Macro for the Click Event

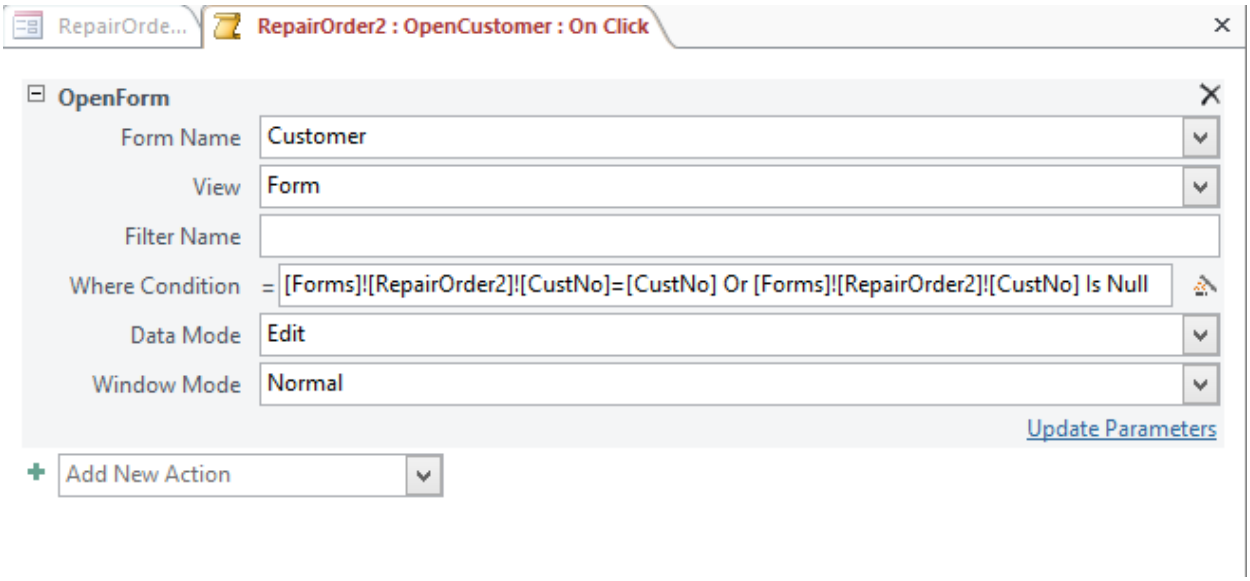


Figure 36: Expanded Event Macro for the Click Event

8.2.2 Adding Conditional Logic to Macros

Sometimes you want to restrict a macro's execution beyond the occurrence of an event. In previous versions of Access, you could add conditions to refine when a macro executes. Access 2013 has eliminated conditions from the macro specification. Instead, Access 2013 supports the *If* action to provide more flexibility for adding various conditional logic to macros. This section demonstrates the *If* action to restrict the logic in a macro. This section also demonstrates a second macro that works with the first macro to update a related column.

The new macros involve the subform *Parts: Subform2* of the *RepairOrder2* form. Recall that this subform displays the parts used on a repair order (Figure 37). This subform can change records in the *PartsUsed* table. However, changing the *QtyUsed* field of a record in the *PartsUsed* table should cause a change in the *UnitsInStock* column of the *Part* table. Currently, the subform does not update the *In Stock* form field. You need to write macros to update the *In Stock* field if there is sufficient stock to meet the requested demand.

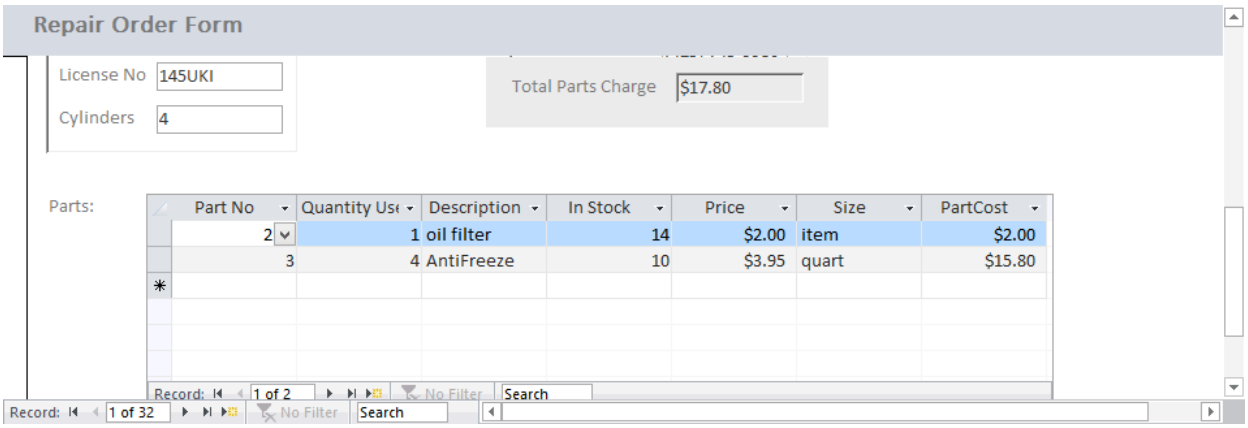


Figure 37: *Parts: Subform2* Embedded in *RepairOrder2*

The macros involve the *Before Update* and *After Update* events of the *Quantity Used* field in the subform. Because the *Before Update* event occurs just before updating a control with changed data, it supports complex validation rules. This event is not triggered if the data in a control are not changed. The *After Update* event occurs just after the update has been processed. This event supports changes to related fields. These macros work together so that if the validation rule in the *Before Update* event is violated, the update action in the *After Update* event does not execute.

The condition in the *Before Update* macro is true if there is not sufficient stock to meet the requested demand. If the number of additional units required is greater than the quantity in stock, then display a message box to alert the user and cancel the update. The number of additional units is computed as the new *Quantity Used* textbox value minus the old *Quantity Used* value. You will use the *OldValue* property to reference the old value. Simply using the field name references the new value. The *CancelEvent* action removes the effect of a change.

The *After Update* event does not occur if the *CancelEvent* action has been executed in the *Before Update* macro. Thus, the *After Update* event requires no condition. The first action of the *After Update* event updates the *In Stock* textbox with the change in quantity used. The change is computed as the current *In Stock* value minus the number of additional parts required. The number of additional parts required is the same as described in the previous paragraph.

The *After Update* macro contains one additional action to save the subform record using the *RunCommand* action and the "SaveRecord" parameter value. Saving the record is necessary to update the *OldValue* property. If the *OldValue* property is not updated to reflect the update to the *Quantity Used* textbox, subsequent changes before saving the record will not update the *In Stock* textbox correctly.

After you have understood the motivation and the description of these macros, you are ready to formulate them. The instructions to formulate the macros are given in three parts. First, you will create the *Before Update* macro. Next, you will create the *After Update* macro. Finally, you will modify the subform to work properly with the macros.

Before Update Macro

1. Open the Properties Window: On the navigation pane, select the *Parts: Subform2* and open it in Design view. Select the *QtyUsed* textbox and open its Property Sheet window to the **Event** tab.
2. Open the Macro Builder: Click in the *Before Update* event property and click again on the ellipsis (**…**) button to open the Choose Builder window. Select **Macro Builder** and click **OK**. A new tab should open containing a combo box for adding a new action.
3. Add *If* Action: The actions should execute if a condition is true. In Access versions before Access 2010, it was necessary to use a condition column in a macro step. In Access 2013, you need to us an *If* action instead. You can add an *If* action by dragging and dropping the *If* action from the Program Flow group of the Action catalog. Alternatively, you can select the *If* action from the action list.

4. Enter Condition: In the *Conditional expression* textbox, click the expression builder (**Magic Wand** icon). Type the following text in the Expression Builder window and then click **OK**:

   [QtyUsed]-NZ([QtyUsed].[OldValue])>[UnitsInStock]

   The *NZ* function converts a null value into 0. Null values make the result of the condition unpredictable. The *OldValue* property is null for new records. Figure 38 shows the If action with the conditional expression specified.

5. Document the Condition: Drag and drop a comment action from the Program Flow group in the Action Catalog to the *If* action in the Macro pane. Enter text in the *Comment* column to document the condition. Figure 39 shows the *If* action with the comment added.

6. Add a Message Box Action to the If Action: Select "MessageBox" in the action list of the *If* action. Do not add the action below the End If. In the area below, set parameter values as follow:

   - Message: Type "Quantity Used change exceeds units in stock".
   - Beep: "Yes".
   - Type: Click inside this area, click the arrow, scroll down, and select "Warning!".
   - Title: Type "Quantity Used Change".
   - Figure 40 shows the completed *MessageBox* action with parameters.

7. Add *CancelEvent* Action: In Add New Action combo box below the MessageBox action (but still inside the If action), select "CancelEvent". Figure 41 shows the completed macro. Save the macro when finished and return to the *QtyUsed* Properties window.

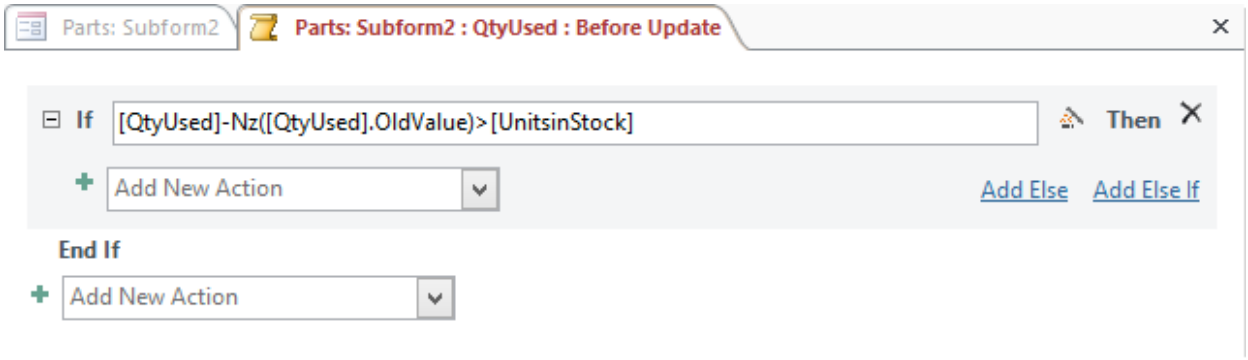


Figure 38: Conditional Expression Specified for the If Action

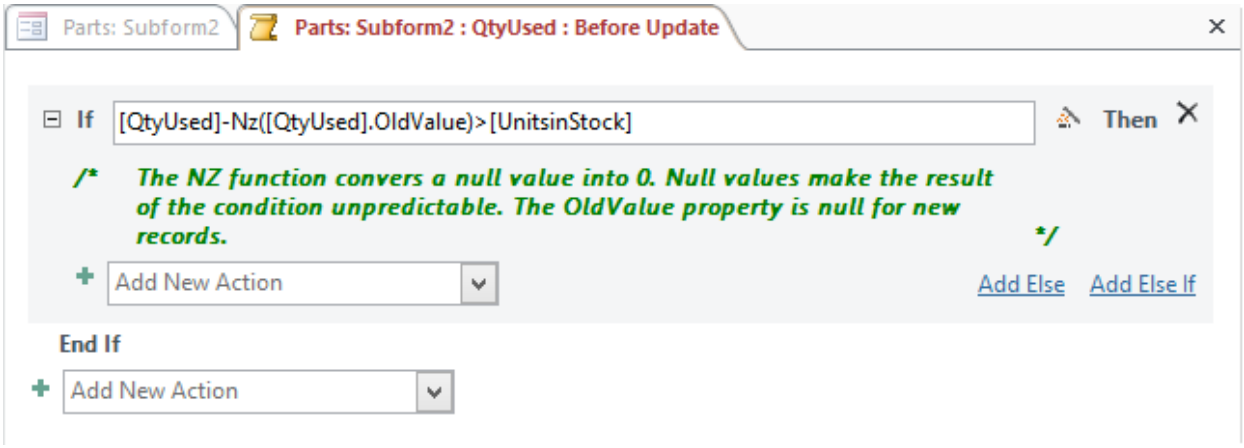


Figure 39: Comment Specified for the If Action

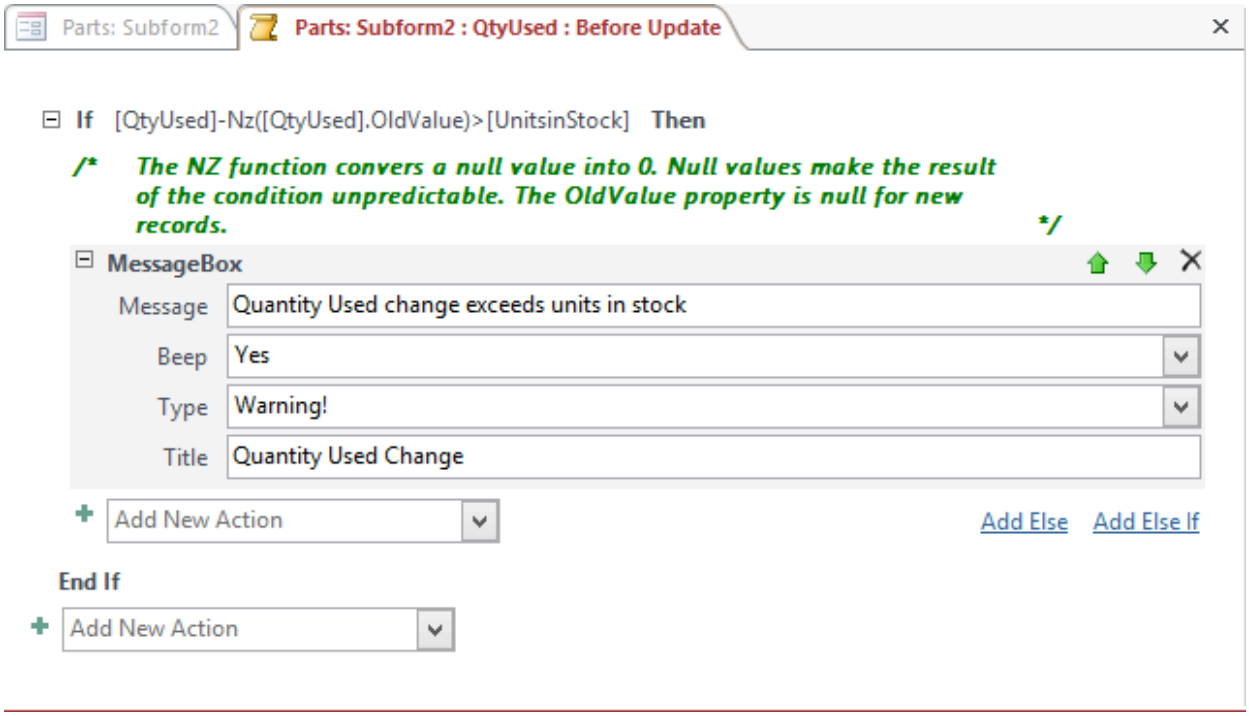


Figure 40: *MessageBox* Action Specified for the If Action

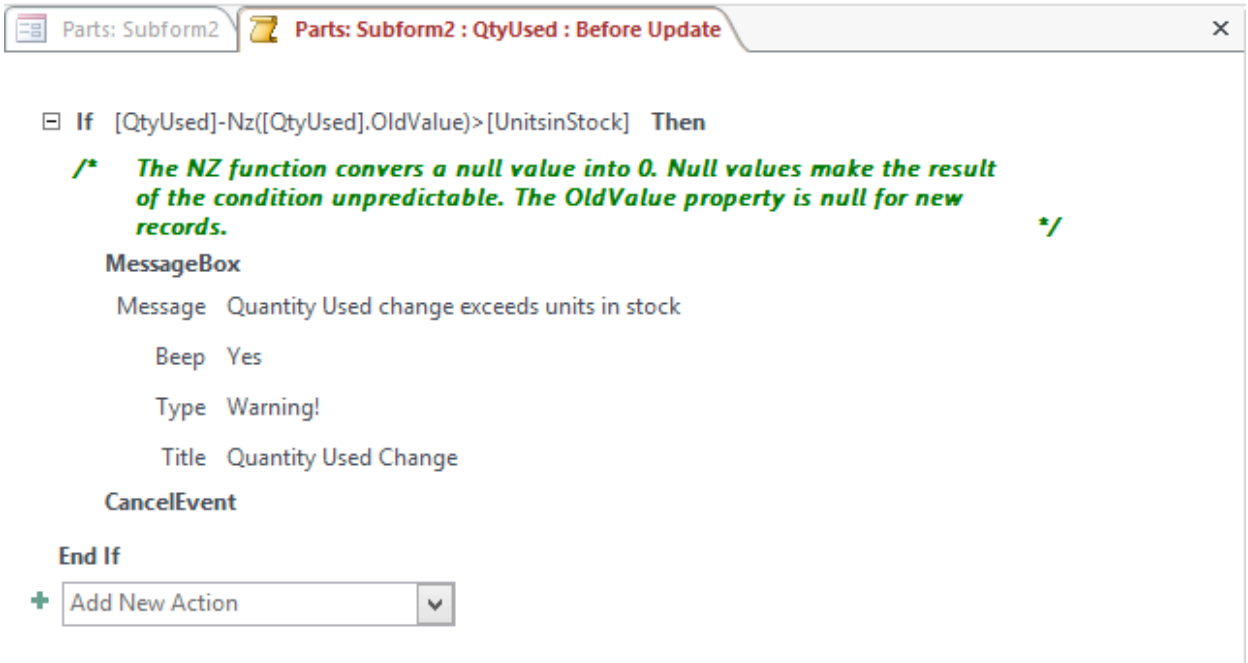


Figure 41: *CancelEvent* Action Specified for the If Action

After Update Macro

1. Open the Macro Builder: Click in the *After Update* event property and click again on the ellipsis (**…**) button to open the Choose Builder window. Select **Macro Builder** and click **OK**.

2. Set the *SetValue* Action: Because the *SetValue* action is not in the Action Catalog, you need to show all actions (**Design → Show All Actions**) so that it appears in the action list. After showing all actions, you should select "SetValue" in the action combo box. Note that *SetValue* is considered as an unsafe action by Access.

3. Set Argument Values: In the area below, set argument values as indicated in the list below. For your reference, see Figure 42.

   - Item: Type the value "[UnitsInStock]". Alternatively, you can select the expression builder button and type the value in the Expression Builder window and then click **OK**.
   - Expression: Click the expression builder button to open the Expression Builder window, type the following expression, and then click **OK**.

     [UnitsInStock]-([QtyUsed]-NZ([QtyUsed].[OldValue]))

4. Add *RunCommand* Action and Argument Values: Add a new action with "RunMenuCommand" as the action. In the argument area, choose "SaveRecord" from the command list. Figure 43 shows the resulting macro.

5. Enter Comments: Drag and drop *Comment* actions from the Program Flow group in the Action Catalog to the Macro object pane. Figure 44 shows the macro object pane after adding comments after both actions. When finished, save the macro and return to the *QtyUsed* Properties window.

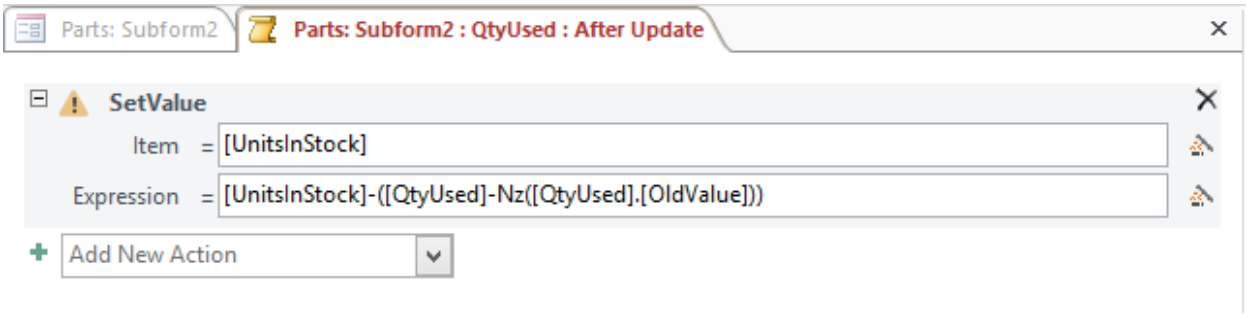


Figure 42: Macro with *SetValue* Action and Arguments

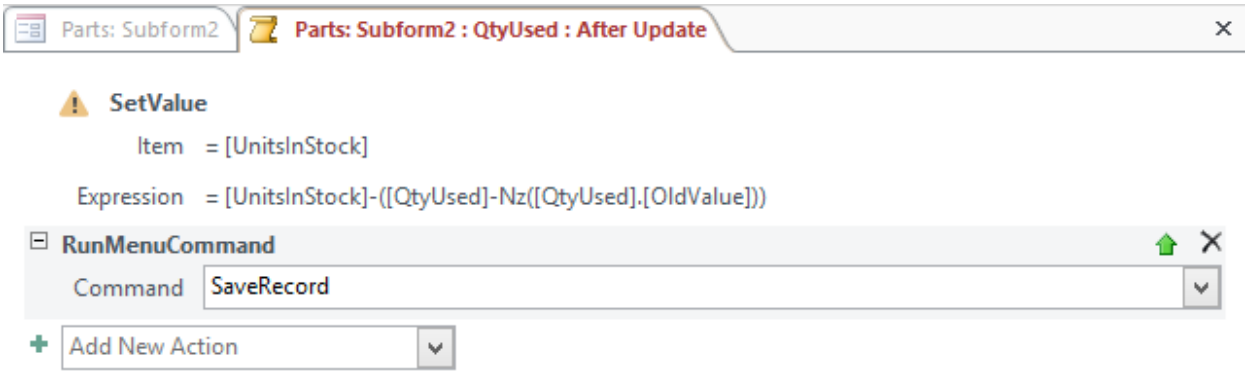


Figure 43: Macro with *RunMenuCommand* Action and Arguments

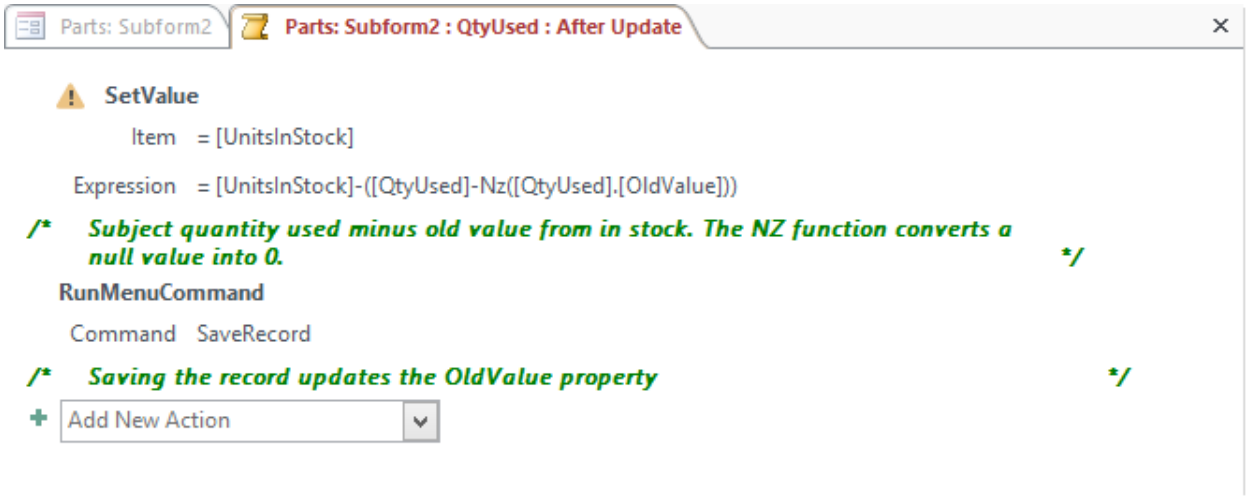


Figure 44: Macro with Comments Added for Each Action

Make Subform Changes and Execute Macros

1. Change *UnitsInStock* Properties: Select the *UnitsInStock* textbox and open its Properties window. Change the *Enabled* property to "No" and the *Locked* property to "Yes". These properties prevent a user from changing the value. Only the *After Update* macro can change the value.

2. Execute the Macros: Open the main form of the *RepairOrder2* form in form view. In the subform, enter a new value for a part in the *Quantity Used* column. A corresponding change should occur in the *In Stock* column. If you enter a value in which the additional parts required exceed the *In Stock* value, a message box should appear as shown in Figure 45. Close the form and save the changes when prompted.

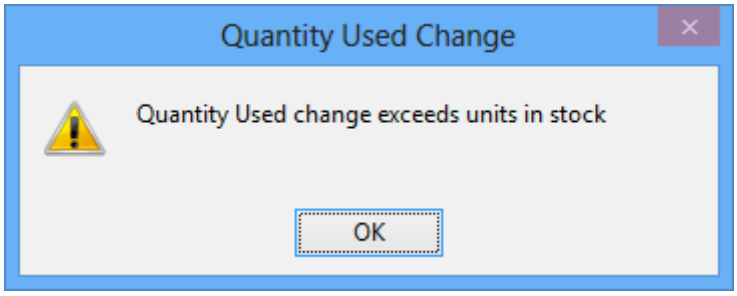


Figure 45: Message Box when Quantity Used Exceeds Units In Stock

The macros demonstrated in section 8.2.2 should be defined as data macros if the rules are independent of the Repair Order form. The next section demonstrates data macros to implement business rules that are independent of application objects.

8.2.3 Creating Data Macros

Data macros, a new feature in Access 2010, support business rules associated with tables. The macros defined in the earlier subsections are associated with operations on forms, not with operations on tables. Macros for forms support form customization, an important aspect of user interface design. In contrast, data macros support business rules independent of application objects. Data macros apply to database manipulations in all application objects so there can be economies of scale in defining data macros rather than application macros.

Data macros support a subset of functionality provided by SQL triggers described in textbook Chapter 11. The data macro feature in Access 2013 is limited to SQL row triggers, the most widely used types of triggers. Statement trigger capability is not available for data macros. Data macros are less precise than

triggers as data macros fire for operations on an entire table, not operations on specific columns of a table. Microsoft Access does not follow the SQL trigger syntax so data macros are not portable across database products. In addition, Access 2013 does not support macro names for database events although named data macros can be defined without a database event. Still, despite some limits in functionality and non-portability, data macros are a major enhancement for Access databases.

This section presents data macros for common business rules using the events *Before Change*, *After Update*, *After Delete*, and *After Insert*. The first example demonstrates a *Before Change* macro to implement a hard constraint to prohibit large price increases. The second example demonstrates an *After Change* macro to implement a soft constraint that logs price increases that are somewhat large but not prohibited. Both examples depict transition constraints involving a comparison of a column value before and after an update operation. The third example involves a *Before Change* macro for data standardization to ensure that state codes are stored in upper case format. The fourth example depicts an *After Delete* macro for update propagation. The fifth example involves an *After Insert* macro for a complex soft constraint to ensure that the odometer values have not decreased across repairs of the same vehicle.

Before Change Data Macro for a Hard Constraint

The first data macro fires before changing a *Part* row ensuring that the *UnitPrice* has not increased by more than 10 percent. This macro implements a hard constraint causing the update to fail.

1. Open the *Part* Table: Since the data macro is associated with the *Part* table, you need to open the *Part* table in Design view.

2. Select the *Before Change* Event: Select **Design → Create Data Macros → Before Change** in the Field, Record, & Table Events group of the dynamic Table Tools tab group (Figure 46). Before selecting the event, you can hold the mouse over the **Before Change** item to reveal its event tip. After selecting the item, a new tab is created for the *Before Change* event (Figure 47) with an empty combo box for specifying a new action.

3. Add an *If* Action: Double click on the *If* action in the Program Flow group of the Action catalog to add an *If* action to the macro object area. Alternatively, you can select the *If* action from the action list in the macro object pane. Figure 48 shows the macro object pane with the empty *If* action.

4. Specify Conditional Expression: Invoke the Expression Builder () to specify the conditional expression part of the *If* action. You can select the *Updated* function in the built-in functions and the *UnitPrice* column from the Expression Categories list for the *Part* table. The *Updated* function restricts the condition to update operations for the specified column. The *Old* function retrieves the value of the column prior to the update. Figure 49 shows the Expression Builder window with the text below.

   Updated("UnitPrice") AND [UnitPrice] > 1.1 * [Old].[UnitPrice]

5. Specify the *RaiseError* Action for the True Part of the *If* Action: You can select the *RaiseError* action in the action list or in the Data Actions group of the Action Catalog. Specify the error number and error description parameters as specified in Figure 50. The parameter values have no meaning to Access so you can choose different values if you wish.

6. Save the Macro: Before saving the macro, you should make sure that it matches Figure 51. Save and close the Macro object pane to return to the object pane with the *Part* table in design view. Save the *Part* table in Design view after the macro object pane closes.

7. Test the *Before Change* Macro: Switch to datasheet view and increase the *UnitPrice* column value in a row by more than 10 percent. You should see an error message shown in Figure 52. If you want to modify the macro, you should switch to design view and select **Design → Create Data Macros → Before Change**.

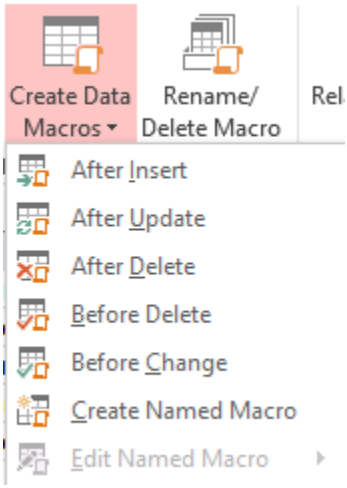


Figure 46: List of Events for Creating a Data Macro

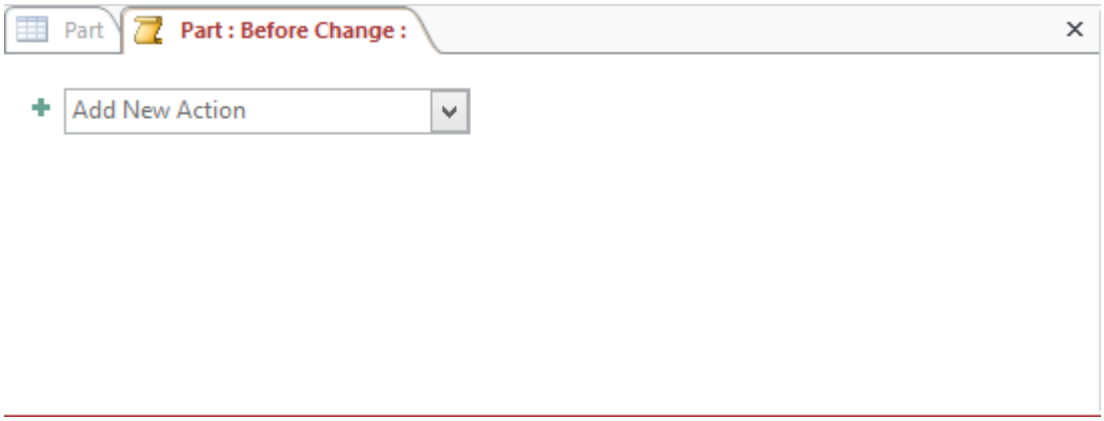


Figure 47: Empty Tab for the Before Change Event

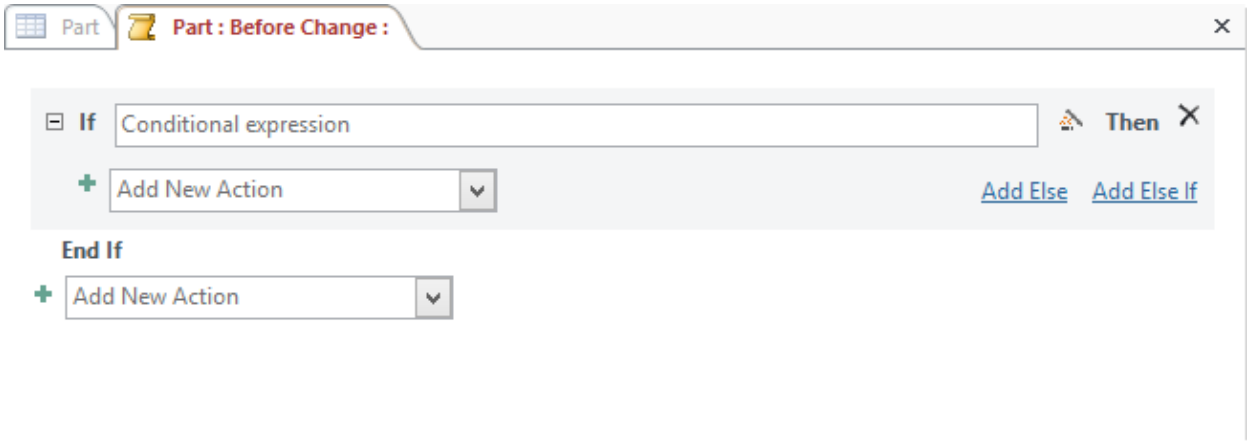


Figure 48: If Action Added to Macro Object Pane

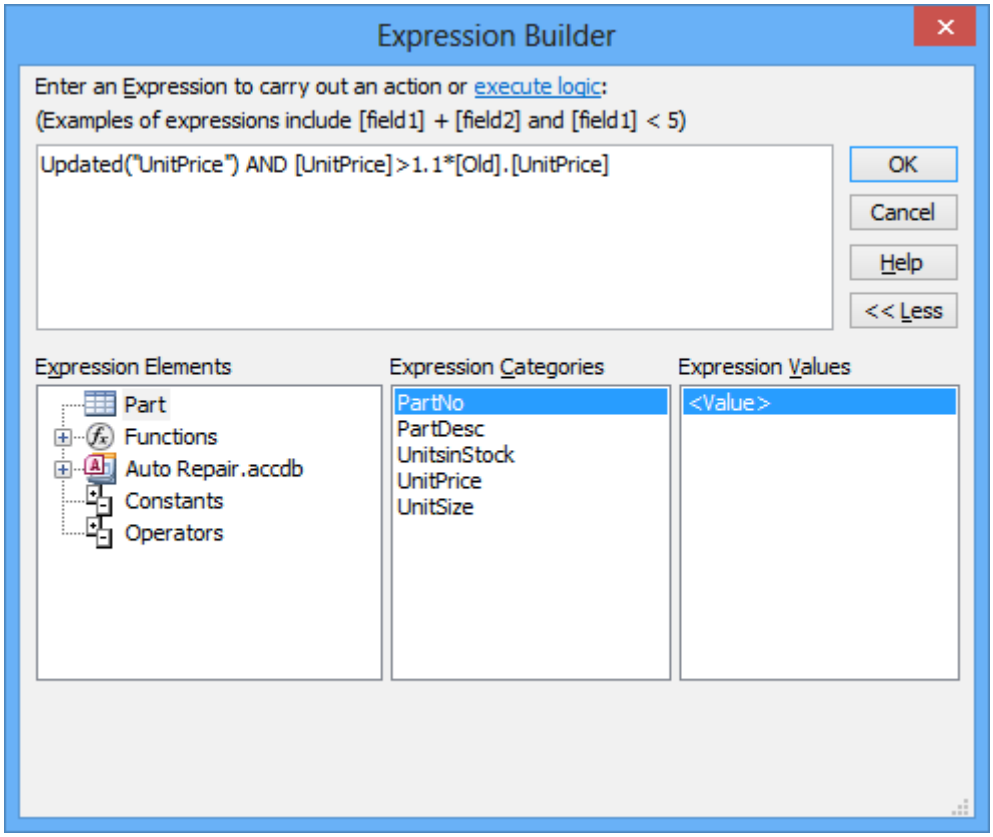


Figure 49: Expression Builder Window with Conditional Expression

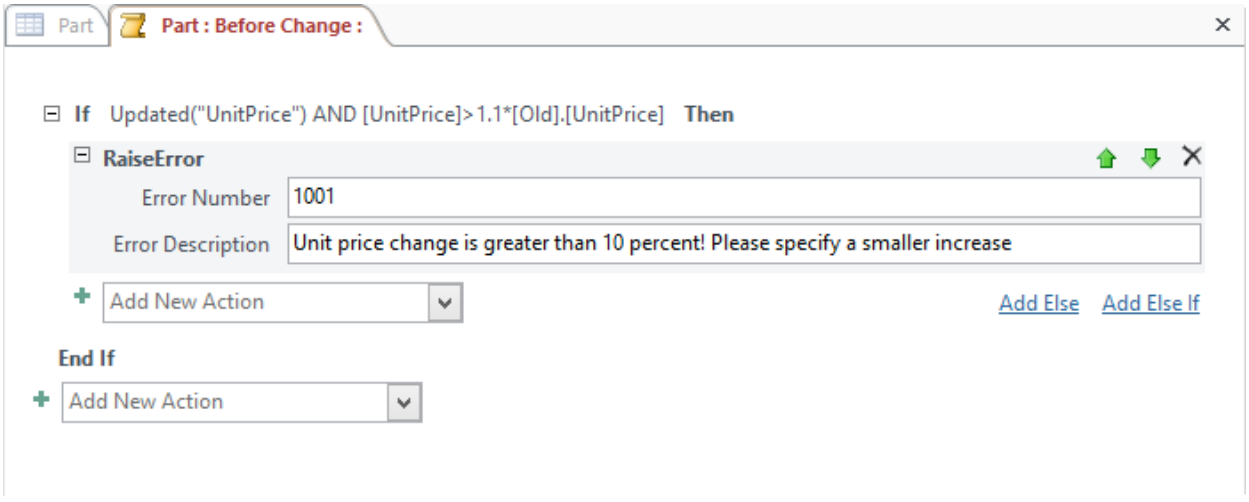


Figure 50: *RaiseError* Parameter Values Specified

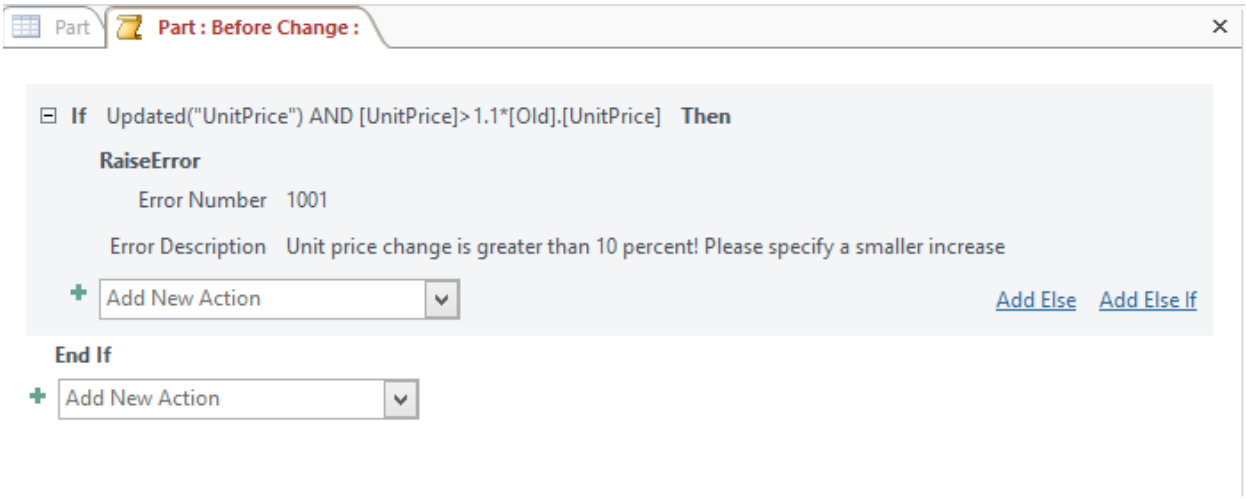


Figure 51: Final *Before Change* Data Macro

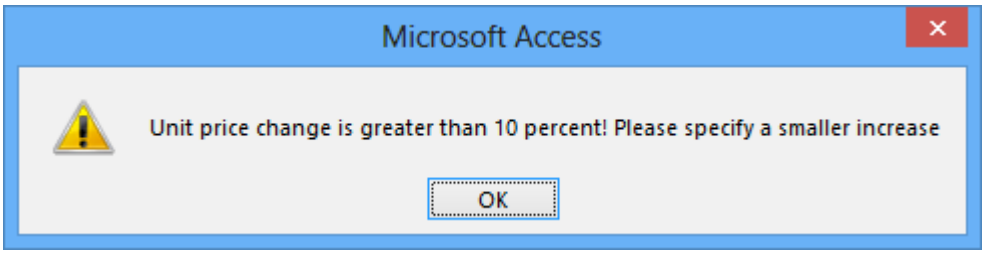


Figure 52: Error Message *Before Change* Macro Violation

The *Before Change* data macro depicted in the first example implements a hard constraint. Price increases larger than 10 percent are rejected. An alternative approach is a soft constraint in which the update is not rejected. Instead a row can be inserted into an exception table and a message sent to a data specialist to ensure that the price increase was allowable. A soft constraint cannot be implemented as a *Before Change* macro because a *Before Change* macro cannot insert a row in another table. An *After Update* macro is necessary to manipulate rows in other tables. The *Before Change* macro can update values in the same row such as to standardize a text value by using upper case.

After Change Data Macro for a Soft Constraint

The second macro depicts a soft constraint for changes to the *UnitPrice* column of the *Part* table. Moderate price increases (between 5 and 10 percent) are logged so that an administrator can be alerted. For simplicity, the *After Update* macro uses the *LogEvent* action to insert a row into the Access system log table rather than insert a row into a custom log table.

1. Select the *After Update* Event: After opening the *Part* table in Design view, Select **Design → Create Data Macros → After Update** in the Field, Record, & Table Events group of the dynamic Table Tools tab group. Before selecting the event, you can hold the mouse over the **After Update** item to reveal its event tip. A new tab is created for the *After Update* event with an empty combo box for specifying a new action.

2. Select the *If* Action: Add an *If* action to the macro pane by selecting the *If* action from the action list or the Program Flow group of the Action Catalog.

3. Specify the Conditional Expression Parameter: The conditional expression is similar to the conditional expression for the first macro. The conditional expression uses the *Updated* function and the *Old* function as shown in Table 5. The *AND* operator connects different parts of the expression.

4. Select the *LogEvent* Action: Add a *LogEvent* action nested inside the *If* action from either the action list or the Data Actions group in the Action catalog. Before adding the action, you should hold the mouse over the *LogEvent* action in the Action Catalog to see an action tip.

5. Specify the *Description* Parameter: The only parameter for the *LogEvent* action is the *Description* parameter. Table 5 shows the value as a string expression containing a concatenation of string constants and row data converted to text using the *Str$* function. The message includes the old and new *UnitPrice* values and the *PartNo* value. Make sure that your expression contains the = symbol at the beginning of the value.

6. View the Final *After Update* Macro: Compare your macro to Figure 53 to ensure that your macro has been correctly specified. Make sure that the *LogEvent* action is nested inside the *If* action as shown in Figure 53.

7. Test the *After Update* Macro: After saving the macro and *Part* table (in Design view), you should test the macro. Update the *UnitPrice* column in an existing *Part* row with a larger value (5 to 10 percent larger) than the previous value. To see the log row, you need to change the options in the Navigation pane. Right click in the Navigation pane and select **Navigation Options …** to display the Navigation Options window. Check the Show System Options check box as shown in Figure 54. Open the *USysApplicationLog* table in datasheet view. You should see a row in the *USysApplicationLog* table with the message in the *Description* column.

Table 5: Parameter Values for Actions in the *After Update* Macro

Action	**Parameter**	**Value**
If	*Conditional expression*	Updated("UnitPrice") AND [UnitPrice] BETWEEN 1.05 * [Old].[UnitPrice] AND 1.1 * [Old].[UnitPrice]
LogEvemt	*Description*	="Old Unit Price: " & Str$([Old].[UnitPrice]) & " New Unit Price: " & Str$([UnitPrice]) & " for PartNo: " & Str$([PartNo]) & ". Problem: moderate price increase"

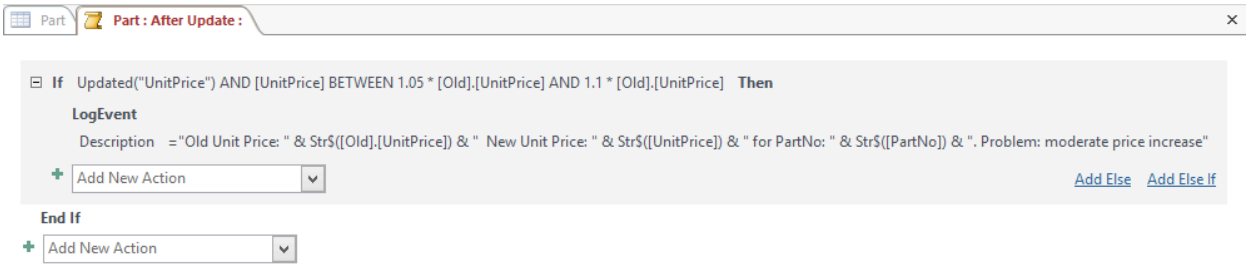


Figure 53: Final *After Update* Data Macro

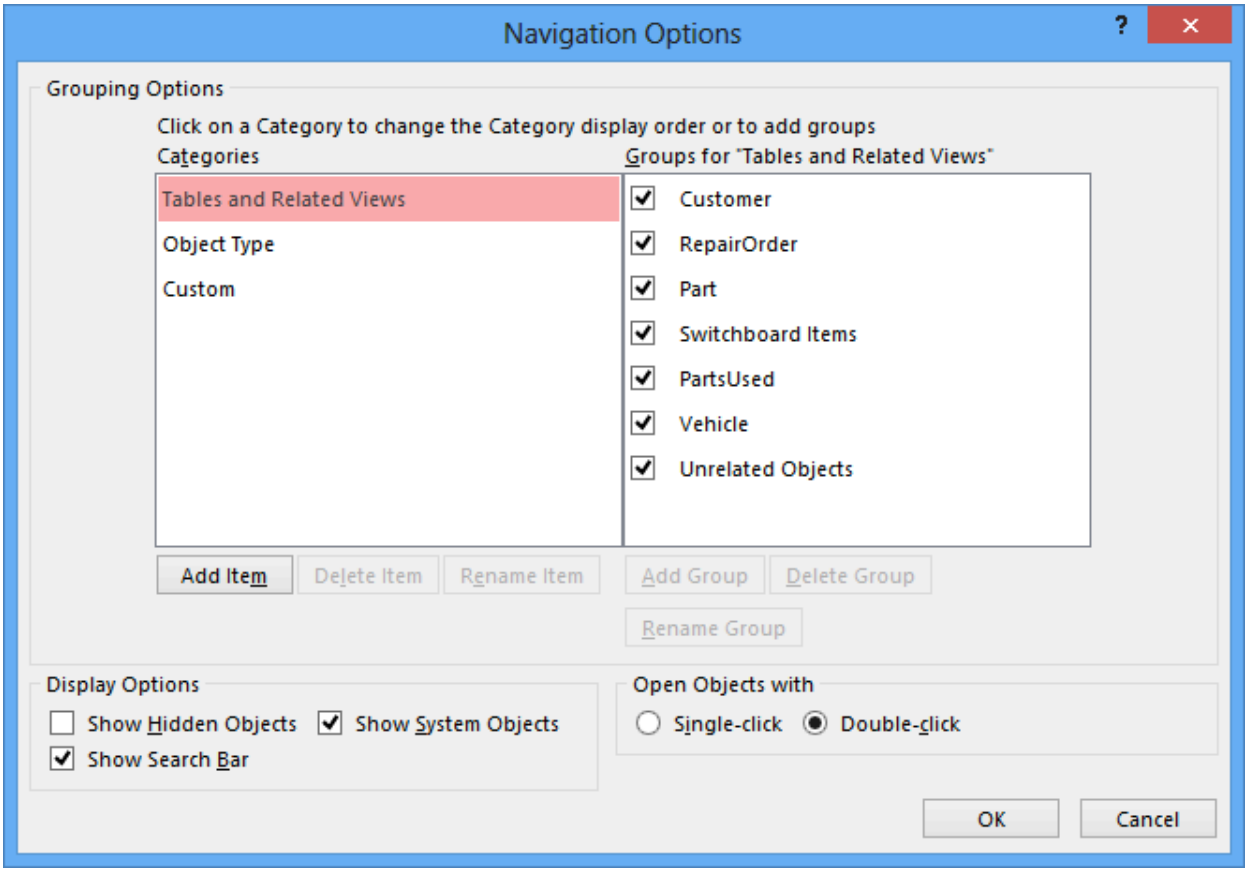


Figure 54: Navigation Options Window with Show System Objects Checked

Before Change Data Macro for Data Standardization

The third macro standardizes the *State* column of the *Customer* table by converting the value to upper case. More complex standardizations are possible such as converting a state name into a two character abbreviation but the principles are the same for more complex transformations. The macro uses the *Before Change* event because only a column value in the same row is changed.

1. Select the *Before Change* Event: After opening the *Customer* table in Design view, select **Design → Create Data Macros → Before Change** in the Field, Record, & Table Events group of the dynamic Table Tools tab group. A new tab is created for the *Before Change* event with an empty combo box for specifying a new action.

2. Select the *If* Action: Add an *If* action to the macro pane by selecting the *If* action from the action list or the Program Flow group of the Action Catalog.

3. Specify the Conditional Expression Parameter: The conditional expression uses the *Updated* function as shown in Table 6.

4. Specify the *SetField* Action and Parameter Values: Add the *SetField* action inside the *If* action. Specify the argument values as shown in Table 6. The *Value* parameter value uses the *UCase$* function that converts a string to upper case.

5. View the Final *Before Change* Macro: Compare your macro to Figure 55 to ensure that your macro has been correctly specified. Make sure that the *SetField* action is nested inside the *If* action as shown in Figure 55.

6. Test the *Before Change* Macro: After saving the macro and *Customer* table (in design view), you should test the macro. Update the *State* column in an existing *Customer* row with a different value (such as "WA" to "co"). After moving to the next row, the lower case value is changed to upper case ("CO"). However, if the *State* value is changed from "WA" to "wa", the macro does not fire and the value remains as "wa". Macro firing is not case sensitive because string comparisons are not case sensitive in Access. Thus, the macro does not fire for case changes alone.

Table 6: Parameter Values for Actions in the *Before Change* Macro

Action	**Parameter**	**Value**
If	*Conditional expression*	Updated("State")
SetField	*Name*	State
SetField	*Value*	UCase$([State])

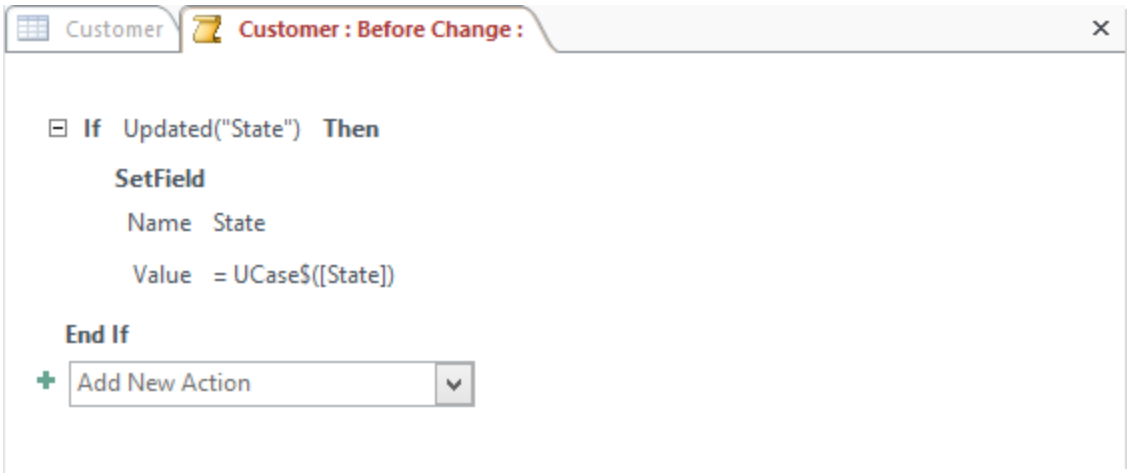


Figure 55: Final *Before Change* Data Macro

After Delete Data Macro for Update Propagation

This data macro fires after deletions of *PartsUsed* rows, incrementing the *InStock* column of the related *Part* row. The value of the *InStock* column in the related *Part* row is incremented by the old value of the *QtyUsed* in the deleted *PartsUsed* row.

1. Open the *PartsUsed* Table: Since the data macro is associated with the *PartsUsed* table, you need to open it in Design view.

2. Select the *After Delete* Event: Select **Design → Create Data Macros → After Delete** in the Field, Record, & Table Events group of the dynamic Table Tools tab group. Before selecting the event, you can hold the mouse over the item to reveal its tip. A new tab is created for the *After Delete* event with an empty combo box for specifying a new action.

3. Add *LookupRecord* Action: The *LookupRecord* action supports retrieval of row in a related table. You will use the *LookupRecord* action to retrieve the related *Part* row. You need to set the parameter values as indicated in Table 7. The *Alias* parameter allows you to refer to the *Part* row in the next action.

4. Add the *EditRecord* Action: Add an *EditRecord* action inside the *LookupRecord* action. Specify the *Alias* parameter value as indicated in Table 7.

5. Add the *SetField* Action: Add a *SetField* action inside the *EditRecord* action. Specify the parameter values as indicated in Table 7. The *SetField* and *EditRecord* actions are typically used together because the *SetField* action indicates the column to change for the *EditField* action.

6. View the Final *After Delete* Macro: Compare your macro to Figure 56 to ensure that your macro has been correctly specified. Make sure that the *EditRecord* and *SetField* actions are nested the same as shown in Figure 56.

7. Test the *After Delete* Macro: After saving the macro and *PartsUsed* table (in Design view), you should test the macro. Insert a new row in the *PartsUsed* table with a *QtyUsed* value of 1. Update the corresponding *Part* row by decrementing the *UnitsInStock* column by 1. After deleting the new *PartsUsed* row, you should see the *UnitsInStock* value in the corresponding *Part* row return to its original value.

Table 7: Parameter Values for Actions in the *After Delete* Macro

Action	**Parameter**	**Value**
LookupRecord	*Where Condition*	[P1].[PartNo]=[Old].[PartNo]
LookupRecord	*Alias*	P1
EditRecord	*Alias*	P1
SetField	*Name*	UnitsInStock
SetField	*Value*	[P1].[UnitsInStock]+[Old].[QtyUsed]

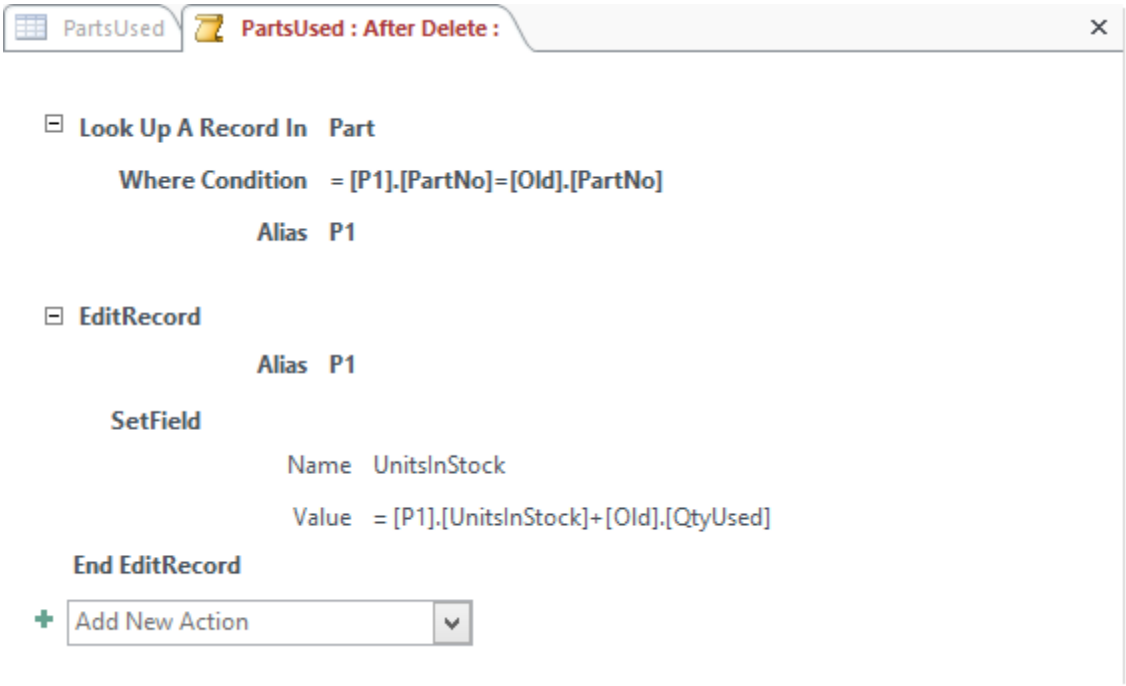


Figure 56: Final *After Delete* Data Macro

After Insert Data Macro for a Complex Integrity Constraint

This data macro fires after insertions of *RepairOrder* rows, checking a complex constraint involving the *Odometer* column. The constraint is soft as rows are inserted into the system log table for violations rather than stopping the insert operation. The macro ensures that the *Odometer* value in the new row is greater than or equal to the value in the other repairs for the same vehicle. It uses the *ForEachRecord* action to find *RepairOrder* rows with the same vehicle serial number but larger odometer values than the odometer value in the inserted row. Rows are inserted in the system log table for each row with a larger odometer value.

1. Open the *PartsUsed* Table: Since the data macro is associated with the *RepairOrder* table, you need to open it in Design view.
2. Select the *After Insert* Event: Select **Design → Create Data Macros → After Insert** in the Field, Record, & Table Events group of the dynamic Table Tools tab group. Before selecting the event, you can hold the mouse over the item to reveal its tip. A new tab is created for the *After Delete* event with an empty combo box for specifying a new action.
3. Enter Actions and Parameter Values: Follow Table 8 to enter the action and parameter values. Note that you should have two *SetValue* actions at the beginning followed by one *ForEachRecord* action and then a *LogEvent* action nested inside the *ForEachRecord* action.
4. Review your Macro: Compare your macro to the completed macro in Figure 57. Since the full value of the *Description* parameter will not fit in the window, you should compare your *Description* parameter value to the Expression Builder window (Figure 58).
5. Test your Macro: After saving your macro and the *RepairOrder* table, you should test the macro. Insert a new row in the *RepairOrder* table with an odometer value less than the value for another repair order on the same vehicle. You should see a new row in the *USysApplicationLog* table.

Table 8: Parameter Values for Actions in the *After Insert* Macro

Action	**Parameter**	**Value**
SetLocalVar	*Name*	CurrOdometer
SetLocalVar	*Expression*	[Odometer]
SetLocalVar	*Name*	CurrSerialNo
SetLocalVar	*Expression*	[SerialNo]
ForEachRecord	*In*	RepairOrder
ForEachRecord	*Where condition*	[CurrOdometer]<[Odometer] And [CurrSerialNo]=[SerialNo]
ForEachRecord	*Alias*	RP1
LogEvent	*Description*	="Odometer has been reset. Current Odometer: " & Str$([CurrOdometer]) & "; Previous Odometer: " & Str$([RP1].[Odometer]) & " for SerialNo: " & [CurrSerialNo] & " and OrdNo: " & Str$([RP1].[OrdNo])

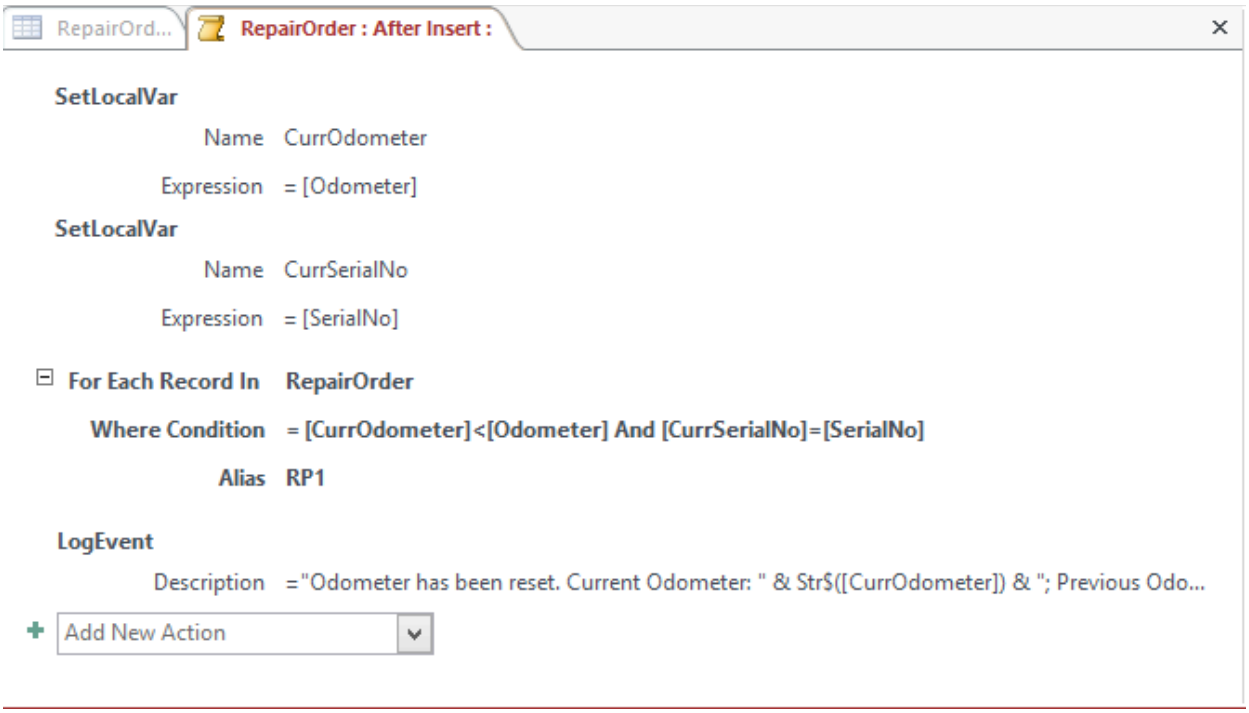


Figure 57: Final *After Insert* Data Macro

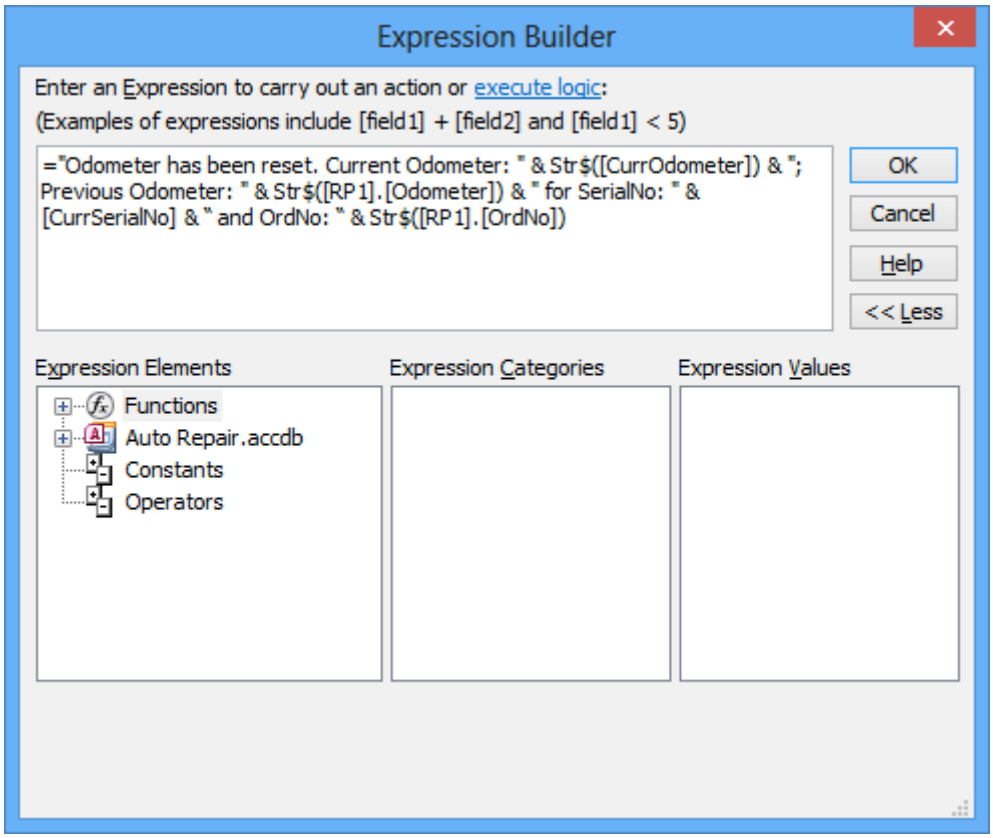


Figure 58: Expression Builder Window with the *Description* Parameter Value

Closing Thoughts

This chapter has covered features that allow you to create a navigation structure for your database. You have gained practice with two tools for creating a navigation structure. The older switchboard form provides a convenient menu for navigation. Because the switchboard form cannot be deployed on a website, you also gained experience with navigation forms, a new feature of Access 2013 providing a tabbed interface for web deployment on a Microsoft Sharepoint server. For direct navigation, you created command buttons to link forms and reports.

To customize the behavior of application objects, you learned about macros. Access 2010 provided a substantially revised environment for developing macros as well as new capabilities for macro actions. Access 2013 builds on this new environment for developing macros. You defined simple macros and also a more complex set of macros containing a variety of actions.

Data macros were a major enhancement in Access 2010 with continued support in Access 2013. Data macros support specification of business rules that are independent of application objects. Data macros provide functionality similar to SQL row triggers described in textbook Chapter 11. You gained experience with specifying data macros for transition constraints, data standardization, update propagation, and complex integrity constraints involving rows of the same table. These macros involved the events *Before Change*, *After Insert*, *After Delete*, and *After Update*.

After completing this chapter, you should have a good understanding of all database objects in Access except modules. You have seen that Access is a powerful desktop DBMS. You created a moderate-size database with powerful forms and reports. Most application development work was accomplished with no coding. The ease of developing powerful database application systems is a strong feature of Access. Many other DBMSs require more coding to build such powerful applications. If you outgrow the limitations of Access databases, your applications can connect to an enterprise DBMS such as Microsoft SQL Server.

To complete your database and Access background, you are encouraged to develop additional database application systems. In developing other systems, you will cement the database concepts and practice emphasized in the textbook and the lab manual. You also are encouraged to explore beyond this chapter, especially to gain experience with deploying a database application on a Microsoft Sharepoint server. With the solid background of database concepts and the practice provided by the textbook and the lab manual along with additional study, you will become a highly skilled Access developer and database professional.

Chapter Reference

The chapter reference section summarizes procedures that you practiced. For wizards discussed in the chapter, the procedures highlight important parts of the wizards but do not list all of the steps.

Procedure 1: Adding the Switchboard Manager to the Ribbon (Section 8.1.1)

123. Click **File → Open** to display the Access Options window. Select the Customize Ribbon. In the "Choose commands from:" list, choose "Commands Not in the Ribbon". Scroll down the list to find the Switchboard Manager.

124. You need to create a new group in the Ribbon. Select Tools under Database Tools in the right pane. Then click the **New Group** button. Rename the group as Switchboard Manager Group using the **Rename** button.

125. Select the Switchboard item in the left pane and click the **Add >>** button to place it in the Database Tools tab.

126. Click the **OK** button to close the Access Options window.

Procedure 2: Creating a Switchboard Form (Section 8.1.1)

1. Open the Switchboard Manager window by clicking on **Database Tools → Switchboard Manager** in the new Switchboard Manager group. A dialog box may appear stating that there is no valid switchboard in this database and asks if you would like to create one. Click **Yes**.
2. In the Switchboard Manager window, click **Edit…** to open the Edit Switchboard window. Type a name for the switchboard form. Click **New** to open the Edit Switchboard Item window.
3. Using the Edit Switchboard Item window, assign form, report, or other commands as items. For each item, you need to identify a caption, the action performed (usually opening a form or report), and the database object. Access only supports a maximum of eight items. If you need more items, create a multiple-level switchboard in which an item in the main switchboard opens another switchboard.
4. If you need to edit or delete an item, while in the Edit Switchboard window, click the item in the list and then click **Edit** or **Delete**.
5. If you want to rearrange items, select an item, and then click **Move Up** or **Move Down**.
6. If already returned to the Database window, then click **Database Tools → Switchboard Manager**. When the window opens, click **Edit** to open the Edit Switchboard window and follow the above instructions.
7. To make modifications to the appearance of the switchboard form, open it in Design view and make changes to fonts, background color, and so on.

Procedure 3: Displaying a Switchboard Form at Startup (Section 8.1.2)

1. To open the Access Options window, click **File → Options → Current Database**.
2. In the Display Form/Page area, click the arrow and select "Switchboard". Under that area, unclick the Display Navigation pane box to clear it. You do not want the Database window to appear. Click **OK** to close the Startup window. Note: This change will go into effect the next time you open the database.
3. To access the Navigation pane to make modifications to forms or reports, type **F11**. This action returns the Navigation pane to the front.

Procedure 4: Creating a Navigation Form (Section 8.1.3)

1. Click **Create → Navigation** to see the list of navigation template forms.
2. After choosing a navigation template, the navigation form opens in Layout view. You can drag and drop objects from the Navigation pane.
3. You can customize the style, shape, fill, outline, and effects of the tabs in a navigation form using the Control Formatting group in the **Format** tab of the Ribbon.
4. Navigation forms can be specified as a startup form as specified in Procedure 3. In addition, you can specify the navigation form as the startup form when deploying the database to a Microsoft Sharepoint server.

Procedure 5: Formulating a Macro in a Form (Sections 8.2.1 and 8.2.2)

1. Open the form in Design view.
2. Determine the control and the event that will contain the macro. You may need to carefully study events for more than one kind of control. Open the Property Sheet for the control and click the **Event** tab.
3. Click in the desired event property and click again on the ellipsis (**…**) button to open the Choose Builder window. Select **Macro Builder** and click **OK**.
4. In the *Add New Action* list, click the arrow and scroll down to select an action. Specify the parameter values for the action.
5. You may continue to add actions as needed. Actions can be selected from the action list or the Action Catalog pane. Each action has a list of parameters that you should provide. For expressions, you can invoke the Expression Builder.
6. Save the macro and close the window.
7. Toggle to form view and test the macro for all conditions.

Note: Macros support the *If* action for conditional logic. The *If* action involves a conditional expression and action parameter. You can add optional *Else* and *Else If* actions for more complex conditional logic.

Procedure 6: Formulating a Data Macro (Sections 8.2.3)

1. Open the target table in Design view.
2. Select the event from the event list (**Design → Create Data Macros**). Note that the *Before Change* and *Before Delete* events cannot alter data in other tables.
3. Specify actions and parameter values as specified for application macros in Procedure 5. Pay careful attention to nesting of actions.
4. Save the macro and close the window. Save changes to the table in Design view.
5. Test the macro by causing the macro to fire through operations on the target table.

Additional Practice

The following problems provide additional practice with the extended auto repair database as well as the textbook databases.

Part 1: User Interface for the Extended Auto Repair Database

3. Revise the *Switchboard* form to have multiple levels as follows:
   - Add a switchboard page called "Forms Menu". Add items to this switchboard page to open the *Customer*, *Vehicle*, *RepairOrder2*, *Part*, *Labor* (the form created in problem (3) of Chapter 4), and *RepairLabor* (the form created in problem (1) of Chapter 5) forms.
   - Add a switchboard page called "Reports Menu". Add items to this switchboard page to open the *Complete Part Report*, the *Monthly Part Report*, the *Part Expense Report*, and the *Labor Usage Report* (the report created in problem (2) of Chapter 6).
   - Revise the "Auto Repair Shop Database Menu". Remove the previous items on the menu and add new items to open the "Forms Menu" page and the "Reports Menu" page.
4. Add command buttons to the *RepairLabor* form as follows:
   - Use the Command Button Wizard to add a button to open the *Customer* form on a record containing the current customer number.
   - Without using the Command Button Wizard, add another button. Write a *Click* macro to open the *Vehicle* form with the current serial number.
   - Using the Command Button Wizard, add a button to open the *Labor Usage Report* created in problem (2) of Chapter 6.
   - Use the Command Button Wizard to add a button that opens the *Switchboard* form.
5. Add command buttons to the *Vehicle* and the *Customer* forms to open the *RepairLabor* form.

Part 2: User Interface for the University Database in Textbook Chapter 10

1. Create a multiple-level switchboard form as you did in problem (1) for the extended auto repair database.
2. Create command buttons to navigate among the forms and the switchboard.
3. Write a macro(s) to decrease the number of seats remaining in a course offering when a student registers for a course offering. You will need to add a field to the *Offering* table for the seats remaining and modify the registration form.
4. Write a data macro to increment the seats remaining when a student drops a course offering.
5. Write a macro to refresh the query for the *OfferNo* combo box in the subform. You should use the *RunCommand* action with the *Refresh* command in the macro. Search for an event that only executes the macro one time even if there are multiple rows in the subform.

Part 3: User Interface for the Order Entry Database in Textbook Chapter 10

1. Create a multiple-level switchboard form as you did in Part 1, problem (1) for the extended auto repair database.
2. Create command buttons to navigate among the forms, the reports, and the switchboard.
3. Create a macro in the Order Form. The macro should copy the billing data (customer name (first and last), street, city, state, and zip) into the corresponding shipping fields (order name, street, city, state, and zip). The replication should occur after a user enters a value for the customer number. The macro

saves data entry time as the billing (in the *Customer* table) and the shipment (in the *OrderTbl* table) data are identical in most orders.

4. Create a macro in the Purchase Form to refresh the combo box query for the product number in the subform. Refresh the query on an event that will occur only one time for the subform, not one time for each subform record.

Glossary of Terms

Access Options Window: provides a variety of options to customize various parts of Access. The Access Options Window provides options in the categories of general, current database, object designers, proofing, language, client settings, customize ribbon, quick access toolbar, add-ins, and trust center. The Access Options Window is available through the **File→Options** command.

After Delete: a data macro event that fires after a row has been deleted from a table. You can use the expression *Old.FieldName* to check the value of a field before the deletion. See also Data Macro.

After Insert: a data macro event that fires after a row has been added to a table. See also Data Macro.

After Update: a data macro event that fires after an existing row has been updated. You can use the expression *Updated("FieldName")* function to check if a specific field has been changed. You can use the expression *Old.FieldName* to check the value of a field before the update. See also Data Macro.

Align Commands: used to align form and report fields. Select the fields to align (use the **Shift** key) and choose the kind of alignment from the buttons in the Control Alignment group of the **Arrange** tab in the Ribbon. You can align fields to the left, right, top, bottom, or grid.

Allow Additions Property: specifies whether records may be added to the database. If the property is set to "Yes", records may be added. If it is set to "No", records cannot be added.

Allow Deletions Property: specifies whether existing records can be deleted. If the property is set to "Yes", the user may delete records. If it is set to "No", the user cannot delete records.

Allow Edits Property: specifies whether a user is allowed to edit existing records. If the property is set to "Yes", the user may edit records. If it is set to "No", the user cannot edit existing records.

Allow Filters Property: specifies whether records in a form can be filtered. Setting the *Allow Filters* property to "No" does not have any effect on the *Filter* and the *Filter On* properties.

Allow Properties: a form has a number of actions available to a user allowing them to add data, edit data, delete data, and view existing data. A form may be created to do any or all of these actions. The allow properties are set in a form's Properties window.

Allow View Properties: the allow view properties (*Allow Form View*, *Allow Datasheet View*, *Allow PivotTable View*, and *Allow PivotChart View*) control the views permitted for a form. For most forms, pivottable and pivotchart views are not appropriate. For must subforms, datasheet view is appropriate, but form view is not.

Allow Zero Length: a property used for text fields when the value does not exist. A zero-length string has no characters.

Application Parts: An application part consists of a table along with associated application objects such as forms and reports. Thus, an application part is a subset of a database template. You can create an application part using the **Create→Application Parts** item in the Templates group. See also Database Template.

Append Query: inserts records into a table. Note that an Append query cannot be viewed in Design view after it is created in SQL view.

AutoLookup Query: provides values for read-only form fields if these fields are bound to tables participating in a 1-M relationship. An AutoLookup query retrieves data from a parent table when a foreign key value is entered from the corresponding child table. You do not need to write AutoLookup queries. Access automatically creates and executes AutoLookup queries when a user enters a foreign key value.

AutoNumber: a data type that guarantees unique values for fields using this data type.

Before Change: a data macro event that fires before an existing row is inserted or updated. You can use the *IsInsert* property to determine if the row has been updated or inserted. Typically, this event is used to perform validation or change the value of a field in the same table. See also Data Macro.

Before Delete: a data macro event that fires before a row has been deletes from a table. Typically, this event is used to ensure that the deletion is valid. See also Data Macro.

Bound Control: most controls that Access places on a form are "bound" to fields from a table. Binding a control means identifying a source of data for the control.

Chart Wizard: this tool creates a form containing a chart.

Column Field: a field appearing in the column area of a pivot table.

Combined Primary Key: a primary key consisting of more than one field. In table design view, when two rows of a table are assigned as primary keys, Access considers these two fields as one combined primary key.

Combo Box: a memory aid that allows a user to select a value from a list of possible values. A combo box has its own set of properties.

Command Button: clicking on this control initiates an action such as opening a related form.

Command Button Wizard: a wizard to help you create buttons that initiate actions when clicked. The wizard can generate code to perform a variety of actions such as opening database objects (forms and reports) and navigating among records.

Control: allows you to manipulate data or initiate actions. Common controls are textboxes for text and numeric data, check boxes for true/false data, and combo boxes for selecting a choice from a list. Each kind of control has its own set of properties.

Control Source Property: binds or associates a control to a database field.

Crosstab Query: data are presented as a two-dimensional array with one field's values on the rows, another field's values on the columns, and a third field's values in the cells. Crosstab queries are especially useful for summarizing financial and other numeric data. The results of a crosstab query display in a spreadsheet format. You can create a crosstab query with just one table by using the Crosstab Query Wizard. However, to create a crosstab query with more than one table, you must create a separate query first.

Data Definition Query: SQL statements that create tables, views, users, and groups. You can use the SQL Window to create a data definition query. Data definition queries are alternatives to the Access tools for creating tables, stored queries, users, and groups.

Data Entry Property: specifies whether a form initially allows data entry. If set to "Yes", the form will open showing a blank record. If the property is set to "No", the form will open showing existing records.

Data Macro: allow specification of business rules associated with tables. Data macros support a subset of functionality provided by SQL triggers described in textbook Chapter 11. The data macro feature in Access 2010 is limited to SQL row triggers, the most widely used types of triggers.

Data Type: determines the kinds of data accepted and the operations that can be performed. See Appendix A at the end of Chapter 1 for a list of the Access data types.

Database Objects: an Access database may be viewed as a group of five objects consisting of tables, forms, queries, reports, and macros. Modules are another type of object although they cannot be viewed in the Navigation pane.

Database Template: a template database contains a complete application including tables, forms, reports, queries, macros, and relationships. Each database template can be hosted on a Microsoft Sharepoint Server providing the ability to use the databases through Web pages. See also Application Parts.

Datasheet: a way to display a table in which the column names appear in the first row and the body in the other rows. Datasheet is a Microsoft Access term.

Datasheet View: a view that shows data for a table or query. When you right click a table or query in the Navigation pane, Access shows the object in Datasheet view. You can also select Datasheet view in the **View** drop down list available in the **Home** or **Design** tab.

Default Value: allows you to specify a value when the user does not provide one.

Default Value Property: preset values determined by the developer for fields in tables and forms.

Delete Query: removes selected rows from the database. Delete queries cannot be viewed in Design view after being created in SQL view.

Design View: an Access view for creating objects.

Detail Field: a field appearing in the cell area of a pivot table.

Enabled Property: allows a control to receive the focus when the user types a value into the control.

Enforce Referential Integrity: a property that can be specified for each relationship shown in the Relationships window. Normally, you should select this property for each relationship. Refer to sections 2.4 and 2.5 of textbook Chapter 2 to refresh the concepts of referential integrity.

Expression Builder: provides a window of structured choices to define complex expressions (combinations of operators, functions, fields, and constants). The expression builder can be used to create property values for most objects. See also Intellisense.

Field Properties: allow you to specify how users can interact with your database. Field properties depend on the data type selected. See Appendix A in Chapter 1 for a cross referencing of field properties and data types.

Field Property Propagation: when creating a bound control in a form, report, or data access page by dragging a field from the field list, Microsoft Access copies certain properties (*Format*, *DecimalPlaces*, *InputMask*, *ValidationRule*, *ValidationText*, and *DefaultValue*) from the field in the underlying table or query to the control.

Field Size Property: defines the maximum size of a field.

Filter: filtering records allows only certain records to be displayed based on criteria specified in form fields. You may choose one or more filter criteria from different fields or from within the same field. You can filter by form, by selection, or by excluding certain selections.

Filter Field: a field appearing in the filter area of a pivot table. Filter fields restrict data that appears in a pivot table.

Filter Property: displays a subset of records that meet specific conditions. For security reasons, a filter may be used to restrict records that contain private information. If the property is set to "Yes", records can be filtered. If the property is set to "No", records cannot be filtered and the filter options on the **Records** menu are disabled.

Find Tool: a fast and convenient way to search and locate a specific record. It can be used regardless of the record displayed on a form or datasheet. Click the **Binocular** button in the Find group in the Ribbon to open the Find window.

Foreign Key: a column or combination of columns whose values are required to match those of a primary key from another table. A foreign key must have the same data type as its associated primary key.

Form: provides convenient data entry especially as compared to tediously entering data directly into the rows and the columns of a datasheet.

Form Command: gives you three style choices for designing your form: Columnar, Tabular, and Datasheet. You can also use this command to design a pivot table and pivot chart.

Form Design Window: an interface that allows you to create a form from scratch and toggle between Design view and Form view. You also can add controls and modify existing controls.

Form Properties: allow you to customize the appearance of a form and change the way that controls react in response to a user's actions. Forms have properties at the form level, the control level, and the section level (e.g., the detail section).

Form View: a view that shows data for a form. When you right click a form in the Navigation pane, Access shows the form in Form view. You can also select Form view in the **View** drop down list available in the **Home** or **Design** tab.

Form Wizard: a sequence of windows that guide you to create a form. You also can use the Form Wizard to create simple forms or complex, hierarchical forms.

Format Property: controls the way data are displayed in a datasheet, form, or report. Format choices depend on the data type. For example, there are a variety of numeric formats such as "Currency", "Percent", and "Scientific".

Get External Data Wizard: allows you to load data from a variety of data sources including other Access databases, Excel spreadsheets, Sharepoint Lists, text files, or XML files into an Access database. The wizard is customized for the type of data source used.

Group: a feature of the Ribbon that arranges commands to improve the organization of commands. See also Ribbon.

Hierarchical Form: allows data entry consisting of a fixed part (main form) and a variable part (subform) on a single form. One record is shown in the main form and multiple, related records are shown in the subform. Beginning in Access 2007, hierarchical forms have been known as "one-to-many forms."

Index: a secondary file structure that provides an alternative path to a table. Indexes typically contain only key values, not the other fields in a logical record. In Access, the *Index* property indicates that a field should have an associated index.

Input Mask: specifies a data entry pattern such as for dates, times, social security numbers, and phone numbers. Access uses input masks to inform users of the pattern of data entry and to ensure that the data entered conforms to the pattern.

Input Mask Wizard: guides you to specify several parts of an input mask.

Intellisense: a Microsoft coding technology used in the Expression Builder in Access 2011. Intellisense supports coding of complex expressions using a variety of completion aids. Intellisense displays a list of object members matching the initial characters typed in a name, provides a list of parameters for functions, matches braces in expressions, and provides quick tips for each item in a list of drop down values. See also Expression Builder.

Join Properties Window: a window that can be opened when tables are joined in Query Design. To see this window, double-click on the join line. The Join Properties window provides choices for join, left outer join, and right outer join.

Join Query: a query created to combine two or more tables with the join operator. When the query executes, a datasheet lists data from both tables that were joined together.

Label Control: automatically attached to identify a textbox. By default, the label contains the field name (since the textbox is bound to the table field) unless another name was entered in the *Caption* field property of the database field. A label also can be created using the toolbox.

Layout View: an intermediate view incorporating some aspects of Design view and Form view or Print Preview view. Layout view is a more visually-oriented view than Design view, but not all tasks can be performed in Layout view. While viewing a form or report in Layout view, each control displays real data. Thus, Layout view is useful for setting the size of controls, or performing many other tasks that affect the appearance of a form or report.

Locked Property: gives a control the ability to lock out (or prevent) the user from changing a value.

Lookup Wizard: provides the ability to create a list of choices for a field. The list of choices comes from either another field or a fixed list of values. This wizard allows you to enter the values in the list or select a field from another table with the allowable values.

Macro: a name given to a sequence of one or more actions that perform a common task. You formulate a macro by specifying its actions and the parameters or arguments for each action. To more easily manage macros, you can store them in a group. A macro is faster to develop but slower to execute than a module.

Macros became prominent in Access 2007 due to security concerns. Macros are trusted because they are prevented from performing certain, potentially unsafe, operations. Access 2010 substantially extends macros with an enhanced integrated development environment and new specification capabilities. See also Data Macro.

Main Form: contains the fixed part of a hierarchical form.

Memo Data Type: for long text such as a field containing notes or comments. Memo fields can be displayed in Datasheet view but cannot be primary key fields. Access 2007 supports rich text format for Memo fields. See also Attachment Data Type.

Module: consists of program statements and procedures written in the Visual Basic for Applications (VBA) language. A module provides more control than a macro but modules are more difficult to manage for security. Modules are usually more efficient than macros because they can be compiled rather than interpreted as macros are.

Multiple Items Form: displays data from more than record at the same time. When first created, a multiple items form resembles a datasheet. A multiple items form is appropriate for power users who want a stylized datasheet for data entry. See also Split Form.

Navigation Form: a navigation aid using tabs for Web style navigation. Navigation forms are new in Access 2010. Since the older style switchboard form cannot be used for Web deployed databases, navigation forms are the preferred style for web deployed databases.

Navigation Pane: database objects are conveniently grouped together in the Navigation pane. The Navigation pane allows you to navigate easily among the different kinds of objects. You can also organize objects by table.

Nesting: a report style that conveniently allows users to spot trends and relationships among a large amount of data.

Object Dependencies: provide a convenient listing of related objects. For a given database object (table, query, form, or report), the object dependency feature shows the objects using the given object and the objects used by the given object. The Object Dependency command is available in the Show/Hide group in the **Database Tools** tab of the Ribbon. To activate the object dependency feature, you need to enable the Name AutoCorrect options using the Access Options window (**Office→Access Options→Current Database**).

Outer Join Query: a query that uses an operator of relational algebra that combines two tables. In an outer join, the matching and nonmatching rows are retained in the result. Thus, the outer join operator is needed to include nonmatching records when joining tables. Access only supports the one-sided outer join operator that retains the nonmatching rows from one of the tables.

Parameter Query: a query that opens a dialog prompting the user for specified data. Parameter queries allow a selection criterion value to be determined at the time a query is executed. Often a user may want a report executed against a slightly different set of data.

PivotChart View: a view that displays a database object as a pivot chart. Pivot charts provide a graphical display of multidimensional data instead of the text display in pivot charts. PivotChart view was a new view in Access 2002.

PivotTable Wizard: creates a form containing a pivot table. A pivot table allows you to manipulate multidimensional data. To view a pivot table, you should use PivotTable View.

Primary Key: a field or combination of fields with unique values. Primary keys ensure entity integrity because each record can be identified through its primary key value.

Print Preview View: shows how a report will appear on screen or printed.

Property Sheet: contains a list of properties specifically for a section or a control along with Access default settings.

Query: a nonprocedural request for data to satisfy decision-making requirements. Access performs searches in the database using the specifications in a query. Query formulation involves translating a problem into a language (such as a SELECT statement) understood by a database system.

Query Design: a visual tool that allows you to create queries without writing much code. In this query design, you can formulate queries by selecting tables and fields, joining tables, and specifying conditions to restrict query results.

Query Wizard: a sequence of windows that guide you to create a query by allowing you to choose tables and other queries along with setting certain criteria and conditions.

Quick Access Toolbar: supports fast access to common tools. The Quick Access toolbar can be customized by using the drop down list to the right of the toolbar.

Record Source Property: binds or associates a form or report record to a database table or query.

Referential Integrity: an integrity constraint involving the primary key in one table with the related foreign key of another table. Only two kinds of values can be stored in a foreign key: (1) a value matching the primary key value in some row of its associated primary key table or (2) a null value.

Relational Database: a system that uses the Relational Data Model to manage collections of data.

Relationships Window: a window where Access allows you to view relationships between tables in the database.

Report: a stylized presentation of data appropriate to a selected audience. Reports can utilize nesting so that a user can follow data relationships easier than in a flat datasheet. Reports can utilize data from many tables.

Report View: a view that shows data for a report. When you right click a report in the Navigation pane, Access shows the report in Report view. You can also select Report view in the **View** drop down list available in the **Home** or **Design** tab.

Report Wizard: a sequence of windows that guide you to create a customized report by selecting specific tables and queries that contain the fields to appear in the report.

Requery Property: updates data in a control or object by rerunning a specified query.

Required Property: determines whether null values are permitted. A null value in a field indicates that the value exists but is unknown at this time. Setting the *Required* property to "No" allows null values.

Ribbon: a functional grouping of buttons and drop-down lists that are relevant to particular tasks. Ribbons are an essential part of every program in Microsoft Office 2007. A ribbon contains tabs (similar to menus) and groups within each tab. Dynamic areas of a ribbon are added when certain objects are selected in a particular view. For example, when a report is opened in Design view, several tabs appear in the Report Design Tools area of the Ribbon.

Row Field: a field appearing in the row area of a pivot table.

Sample Field: allows you to create a new field by choosing from a list of existing fields. You can choose sample fields for basic types, numbers, date and time, yes/no, address, and calculated fields. A sample field can be added when a table is in Datasheet view and a new column is selected. You select **Fields→More Fields** in the in the Add & Delete group while the cursor is in the Click to Add area of the datasheet. See also Table Template.

Show Table Dialog Box: a dialog box that appears listing the tables available for forming relationships.

Sort Tool: to help see patterns in data and find records. Access allows sorting in ascending (smallest to largest) and descending (largest to smallest) orders on one or more form fields.

Split Form: a new feature in Microsoft Office Access 2007. A split form provides two views of data at the same time. You can view all records in a table in Datasheet view and the selected record in Form view. A split form ensures that both views are connected to the same data source and synchronized with each other.

A split form seems ideal for a situation in which a small group of rows should be reviewed at the same time. See also Multiple Items Form.

SQL Server Compatible Syntax: a choice in the **Object Designers** tab of the Access Options window. The choice can be made inthe "SQL Server Compatible Syntax (ANSI 92)" area. You can set the syntax for this database or default for new databases. This setting primarily affects the syntax of SQL data definition statements.

SQL View: a view where you write SQL statements.

Subdatasheet: an embedded or smaller datasheet inside a parent datasheet. Subdatasheets allow convenient navigation to related records in the child side of a 1-M relationship. A subdatasheet may be created for a table or query.

Subform: an embedded or smaller form inside a main form as part of a hierarchical form. It displays a datasheet when opened. The subform is normally opened as part of a hierarchical form, not by itself. A subform has its own set of form properties for customizing.

Switchboard Manager: a way to build a menu for the user when the database first opens. Access calls this menu a "switchboard form" and the Switchboard Manager is used to create it. Similar to a wizard, the Switchboard Manager prompts you with a series of dialog windows.

Tab Order: the order that controls will be accessed on a form as the **Tab** key is pressed by the user.

Table: a named, two-dimensional arrangement of data. A table consists of a heading part and a body part.

Table Design: an interface that allows creation of a table by typing field names, selecting data types, and setting field properties.

Table Properties: allow you to specify a few properties of a table. You can set row level tracking, replication, and attribute visibility.

Table Property Sheet: allow you to specify how a user interacts with a table. You can specify a sub datasheet for a parent table to display related child rows in a datasheet.

Table Template: allows you to create a new table that resembles an existing table. Access provides a list of tables from which to choose including contacts, assets, tasks, issues, and events. When it is finished, you may customize the table further to your requirements. See also Field Template.

Table Validation Rule: integrity constraint that can involve more than one field. You specify a table validation rule in the Table Property Sheet.

Textbox Control: a control for entering text by a user.

Union Query: combines the results of two or more queries using the union operator. Two tables are union compatible if they have the same number of columns and each corresponding column has a compatible data type. A union query cannot be viewed in Design view after it is created in SQL view.

Update Query: changes the values of individual fields of selected records. An update query can be viewed in design view after it is created in SQL view.

Validation Rule: a property that allows you to specify simple rules that restrict values entered by a user. A validation rule is applied differently depending on where it is set, such as for a table, a table field, or a form control. A field validation rule can involve only one field.

Views Allowed Property: sets the views allowed for a form. The possible views are a single record as a form ("Form"), multiple records as a table ("Datasheet"), or multiple records as a form ("Continuous Forms").

View Toolbar: appears in the bottom right of the Access window. The View Toolbar allows convenient switching between views of an object. The views appearing in the View Toolbar are relevant to the object manipulated in the view pane.

Wizard: a sequence of windows that guide you to create an object or property value. The result of a wizard is an object or property value that you can customize further.