APPENDIX 9.A: USAGE OF MULTIPLE STATEMENTS IN MICROSOFT ACCESS

In Microsoft Access, you can use multiple SELECT statements instead of nested queries in the FROM clause. Using multiple statements can provide simpler formulation in some cases than using nested queries in the FROM clause. For example, instead of using DISTINCT inside COUNT as in Example 9.29, you can use a stored query with the DISTINCT keyword following the SELECT keyword. In Example 9A.1, the first stored query (Temp9A-1) finds the unique combinations of faculty name and course number. Note the use of the DISTINCT keyword to eliminate duplicates. The second stored query (Temp9A-2) finds the unique course numbers in the *Offering* table. The final query combines the two stored queries. Note that you can use stored queries similar to the way tables are used. Simply use the stored query name in the FROM clause.

Example 9A.1

Using Stored Queries Instead of Nested Queries in the FROM Clause

List the name of faculty who teach in at least one section of all fall 2016 information systems courses. The result is identical to that in Example 9.29.

Temp9A-1:

```
SELECT DISTINCT Faculty.FacNo, FacFirstName,
         FacLastName, CourseNo
 FROM Faculty, Offering
 WHERE Faculty.FacNo = Offering.FacNo
   AND OffTerm = 'FALL' AND OffYear = 2016
   AND CourseNo LIKE 'IS*'
```

Temp9A-2:

```
SELECT DISTINCT CourseNo
 FROM Offering
 WHERE OffTerm = 'FALL' AND OffYear = 2016
   AND CourseNo LIKE 'IS*'

SELECT FacNo, FacFirstName, FacLastName
 from [Temp9A-1]
 GROUP by FacNo, FacFirstName, FacLastName
 HAVING COUNT(*) = ( SELECT COUNT(*) FROM [Temp9A-2] )
```

APPENDIX 9.B: SQL:2016 SYNTAX SUMMARY

This appendix summarizes the SQL:2016 syntax for nested SELECT statements (subqueries) and outer join operations presented in Chapter 9. For the syntax of other variations of the nested SELECT and outer join operations not presented in Chapter 9, consult an SQL reference book. Nested SELECT statements can be used in the FROM clause and the WHERE clause of the SELECT, UPDATE, and DELETE statements. The conventions used in the syntax notation are identical to those used at the end of Chapter 3.

Expanded Syntax for Nested Queries in the FROM Clause

```
<Table-Specification>:
  { <Simple-Table>   | -- defined in Chapter 4
    <Join-Operation> | -- defined in Chapter 4
    <Simple-Select> [ [ AS ] AliasName ] }
  -- <Simple-Select> is defined in Chapter 4
```

Expanded Syntax for Row Conditions

```
<Row-Condition>:
{ <Simple-Condition> |   -- defined in Chapter 4
  <Compound-Condition> | -- defined in Chapter 4
  <Exists-Condition> |
  <Element-Condition> }

<Exists-Condition>: [ NOT ] EXISTS <Simple-Select>

<Simple-Select>: -- defined in Chapter 4

<Element-Condition>:
 <Scalar-Expression> <Element-Operator>( <Simple-Select> )

<Element-Operator>:
 { = | < | > | >= | <= | <> | [ NOT ] IN }

<Scalar-Expression>: -- defined in Chapter 4
```

Expanded Syntax for Group Conditions

```
<Simple-Group-Condition>: -- Last choice is new
{ <Column-Expression> ComparisonOperator
    <Column-Experssion> |
  <Column-Expression> [ NOT ] IN ( Constant* ) |
  <Column-Expression> BETWEEN <Column-Expression>
     AND <Column-Expression> |
  <Column-Expression> IS [NOT] NULL |
  ColumnName [ NOT ] LIKE StringPattern |
  <Exists-Condition> |
  <Column-Expression> <Element-Operator> <Simple-Select> }

<Column-Expression>: -- defined in Chapter 4
```

Expanded Syntax for Outer Join Operations

```
<Join-Operation>:
 { <Simple-Table> <Join-Operator> <Simple-Table>
   ON <Join-Condition> |
  { <Simple-Table> | <Join-Operation> } <Join-Operator>
   { <Simple-Table> | <Join-Operation> }
   ON <Join-Condition> |
  ( <Join-Operation> ) }

<Join-Operator>:
{ [ INNER ] JOIN     |
  LEFT [ OUTER ] JOIN |
  RIGHT [ OUTER ] JOIN |
  FULL [ OUTER ] JOIN }
```

Expanded Syntax for Recursive Common Table Expressions

```
WITH CTEName ( ColumnName* )
AS
-- Anchor member (AM) referencing the hierarchical table.
   ( <Simple-Select> – Using expanded <Table-Specification> above
UNION ALL
-- Recursive member (RM) referencing CTEName.
  <Simple-Select> ) – Using expanded <Table-Specification> above
-- Statement using CTEName
<Select-Statement>; – Using expanded <Table-Specification> above
```

APPENDIX 9.C: ORACLE 8I NOTATION FOR OUTER JOINS

Until the Oracle 9i release, Oracle used a proprietary extension for one-sided outer joins. To express a one-sided outer join in Oracle 8i SQL, you need to use the notation (+) as part of a join condition in the WHERE clause. You place the (+) notation just after the join column of the null table, that is, the table with null values in the result. In contrast, the SQL:2016 LEFT and RIGHT keywords are placed after the table in which nonmatching rows are preserved in the result. The Oracle 8i formulations of Examples 9.1, 9.2, 9.4, 9.5, and 9.6 demonstrate the (+) notation.

Example 9.1 (Oracle 8i)

One-Sided Outer Join with Outer Join Symbol on the Right Side of a Join Condition

The (+) notation is placed after the *Faculty.FacNo* column in the join condition because *Faculty* is the null table in the result.

```
SELECT OfferNo, CourseNo, Offering.FacNo, Faculty.FacNo,
       FacFirstName, FacLastName
 FROM Faculty, Offering
 WHERE Offering.FacNo = Faculty.FacNo (+)
   AND CourseNo LIKE 'IS%'
```

Example 9.2 (Oracle 8i)

One-Sided Outer Join with Outer Join Symbol on the Left Side of a Join Condition

The (+) notation is placed after the *Faculty.FacNo* column in the join condition because *Faculty* is the null table in the result.

```
SELECT OfferNo, CourseNo, Offering.FacNo, Faculty.FacNo,
       FacFirstName, FacLastName
 FROM Faculty, Offering
 WHERE Faculty.FacNo (+) = Offering.FacNo
   AND CourseNo LIKE 'IS%'
```

Example 9.4 (Oracle 8i)

Full Outer Join Using a Union of Two One-Sided Outer Joins

Combine the *Faculty* and *Student* tables using a full outer join. List the Social Security number, the name (first and last), the salary (faculty only), and the GPA (students only) in the result.

```
SELECT FacNo, FacFirstName, FacLastName, FacSalary,
       StdNo, StdFirstName, StdLastName, StdGPA
 FROM Faculty, Student
 WHERE Student.StdNo = Faculty.FacNo (+)
            UNION
SELECT FacNo, FacFirstName, FacLastName, FacSalary,
    StdNo, StdFirstName, StdLastName, StdGPA
 FROM Faculty, Student
 WHERE Student.StdNo (+) = Faculty.FacNo
```

Example 9.5 (Oracle 8i)

Mixing a One-Sided Outer Join and an Inner Join

Combine columns from the *Faculty*, *Offering*, and *Course* tables for IS courses offered in 2017. Include a row in the result even if there is not an assigned instructor.

```
SELECT OfferNo, Offering.CourseNo, OffTerm, CrsDesc,
       Faculty.FacNo, FacFirstName, FacLastName
 FROM Faculty, Offering, Course
 WHERE Offering.FacNo = Faculty.FacNo (+)
   AND Course.CourseNo = Offering.CourseNo
   AND Course.CourseNo LIKE 'IS%' AND OffYear = 2017
```

Example 9.6 (Oracle 8i)

Mixing a One-Sided Outer Join and Two Inner Joins

List the rows of the *Offering* table where there is at least one student enrolled, in addition to the requirements of Example 9.6. Remove duplicate rows when there is more than one student enrolled in an offering.

```
SELECT DISTINCT Offering.OfferNo, Offering.CourseNo,
    OffTerm, CrsDesc, Faculty.FacNo, FacFirstName,
    FacLastName
 FROM Faculty, Offering, Course, Enrollment
 WHERE Offering.FacNo = Faculty.FacNo (+)
   AND Course.CourseNo = Offering.CourseNo
   AND Offering.OfferNo = Enrollment.OfferNo
   AND Course.CourseNo LIKE 'IS%' AND OffYear = 2017
```

It should be noted that the proprietary extension of Oracle is inferior to the SQL:2016 notation. The proprietary extension does not allow specification of the order of performing outer joins. This limitation can be problematic on difficult problems involving more than one outer join. Thus, you should use the SQL:2016 outer join syntax although later Oracle versions (9i and beyond) still support the proprietary extension using the (+) symbol.