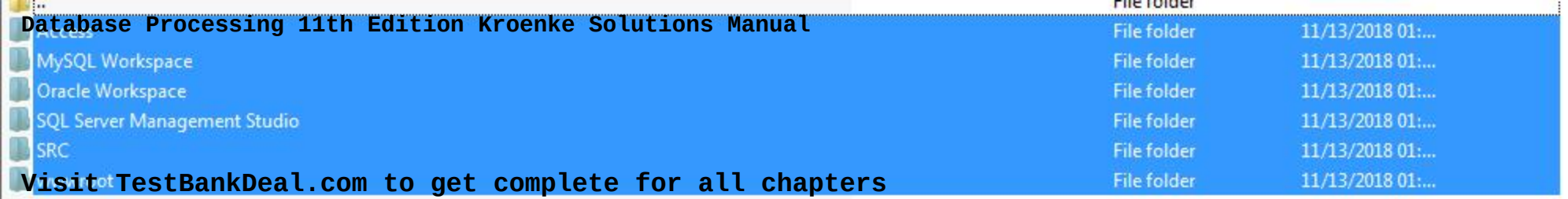


INSTRUCTOR'S MANUAL

TO ACCOMPANY

DAVID M. KROENKE AND DAVID J. AUER

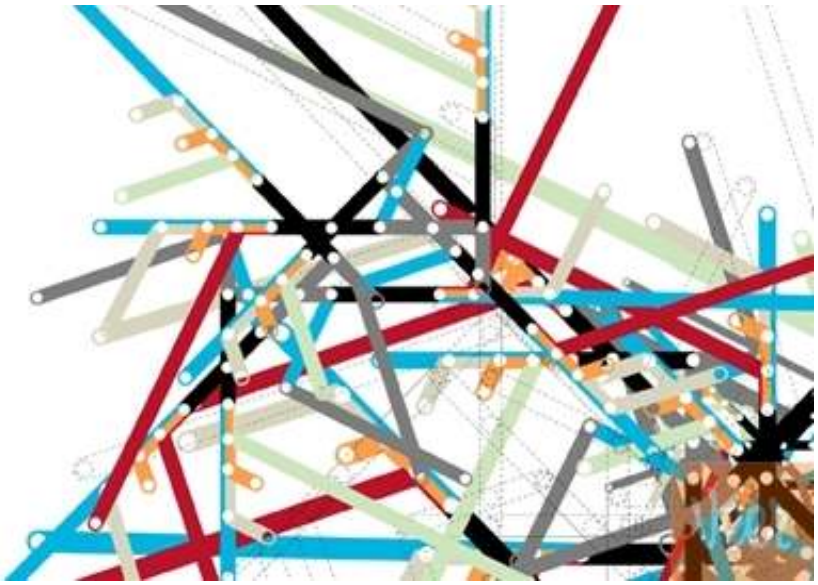
Database Processing

Fundamentals, Design, and Implementation

(11th Edition)

CHAPTER TWO

INTRODUCTION TO STRUCTURED QUERY LANGUAGE



Prepared By

David J. Auer

Western Washington University

CHAPTER OBJECTIVES

- To understand the use of extracted data sets.
- To understand the use of ad-hoc queries.
- To understand the history and significance of Structured Query Language (SQL).
- To understand the basic SQL SELECT/FROM/WHERE framework as the basis for database queries.
- To be able to write queries in SQL to retrieve data from a single table.
- To be able to write queries in SQL to use the SQL SELECT, FROM, WHERE, ORDER BY, GROUP BY, and HAVING clauses.
- To be able to write queries in SQL to use SQL DISTINCT, AND, OR, NOT, BETWEEN, LIKE, and IN keywords.
- To be able to use the SQL built-in functions of SUM, COUNT, MIN, MAX, and AVG with and without the use of a GROUP BY clause.
- To be able to write queries in SQL to retrieve data from a single table but restricting the data based upon data in another table (subquery).
- To be able to write queries in SQL to retrieve data from multiple tables using an SQL JOIN.

CHAPTER ERRATA

- Page 81 - There is no Review Question numbered 2.26. Review Question 2.26 should be inserted as:

  Write an SQL statement to display the SKU, SKU_Description, and Warehouse on products having QuantityOnHand equal to 0. Sort the results in descending order by Warehouse.

- Page 81 - There is an error in Review Question numbered 2.27. Review Question 2.27 should read as follows:

  Write an SQL statement to display the SKU, SKU_Description, and Warehouse on products having QuantityOnHand equal to 0. Sort the results in descending order by Warehouse and in ascending order by SKU.

- Page 81 - Review Question 2.37 should refer to "**SKU_Description**" instead of "Description" to read:

  "Write an SQL statement to show SKU and SKU_Description for all products having a description that includes the word 'Foot'."

- Page 83 - There are two Project Questions numbered 2.57, and Questions 2.59 and 2.60 are identical. DELETE the current Project Question 2.60, and renumber the second Project Question 2.57 as 2.58, renumber the current 2.58 as 2.59, and renumber the current Project Question 2.59 as 2.60.

- Page 83 - Project Questions 2.57. The name of the ASSIGNMENT table is misspelled in the second line of the question. The question should read:

  Figure 2-28 shows the column characteristics for the WPC ASSIGNMENT table. Using the column characteristics, create the ASSIGNMENT table in the WPC.accdb database

- Page 85 - The first sentence in the introduction to the NDX Project Questions should read:

  The following questions refer to the NDX table of data as described starting on page 67.

- Page 86 - Figure 2-31 is miscaptioned – it should read:

  "Column Characteristics for the **ORDER** Table.

- Page 86 - Figure 2-31 shows the column characteristics for the CustomerNumber column in the wrong order - this row should appear second, between the column characteristics rows for InvoiceNumber and DateIn.

- Page 86 - Figure 2-31 shows the wrong Column Name for **DateIn**. In the figure, it appears as **DataIn**.

- Page 86 - Figure 2-31 shows the wrong Column Name for **DateOut**. In the figure, it appears as **DataOut**.

- Page 87 - Figure 2-32 is miscaptioned – it should read:

  "Column Characteristics for the **ORDER_ITEM** Table"

- Page 87 - Figure 2-34 is miscaptioned – it should read:

  "Sample Data for the **ORDER** Table"

- Page 87 - Figure 2-34, the TotalAmount for InvoiceNumber 2009003 is incorrrect – it should read:

  "$49.00"

- Page 88 - In Figure 2-33, the email address for CustomerID 7 [Besty Miller] is incorrect – it should read:

  "Betsy.Miller@**elsewhere**.com"

- Page 88 - Figure 2-35 is miscaptioned – it should read:

  "Sample Data for the **ORDER_ITEM** Table"

- Page 91 - Figure 2-40 is miscaptioned – it should read:

  "Sample Data for the **SHIPMENT_ITEM** Table"

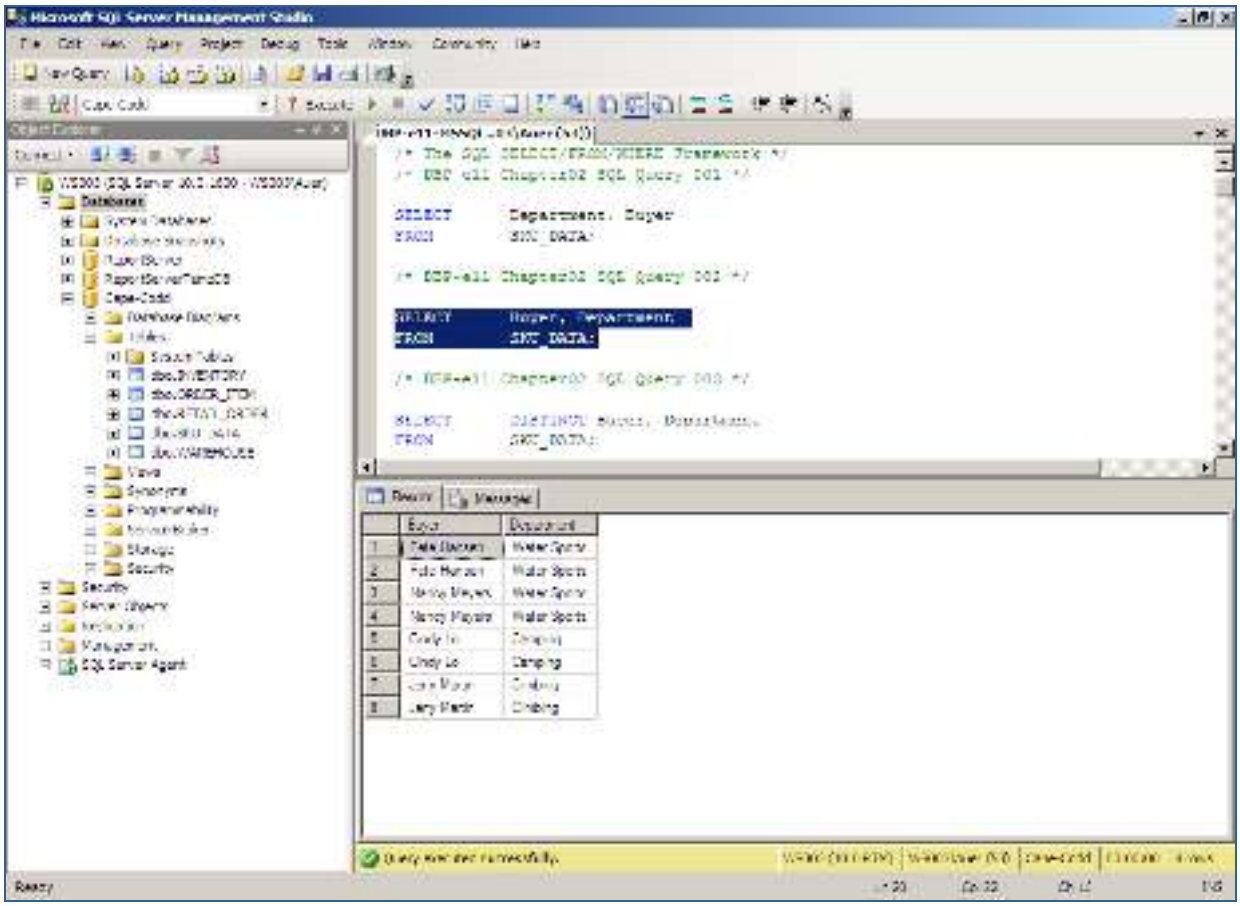
- Page 91 - Figure 2-41 is miscaptioned – it should read:

  "Sample Data for the **ITEM** Table"

- Pages 89 – 91- In Morgan Importing Project Questions B, C, D, E, F, M, N, O, P, and Q, any reference to the column name "Shipper" is a reference to the actual column name "**ShipperName**".

TEACHING SUGGESTIONS

- Database files to illustrate the examples in the chapter and solution database files for your use are available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).
- The best way for students to understand SQL is by using it. Have your students work through the Review Questions, Project Questions, and the Marcia's Dry Cleaning and Morgan Importing Project Questions in an actual database. Student databases in MS Access with basic tables, relationships and data are available in the Instructor's Resource Center and Student Resources on the text's Web site (www.pearsonhighered.com/kroenke).
- The SQL processors in the various DBMSs are very fussy about character sets used for SQL statements. They want to see plain ASCII text, not fancy fonts. This is particularly true of the single quotation used to designate character strings, but I've also had the minus sign have problems. If your students are having problems getting a "properly structured SQL statement" to run, look closely for this type of problem.
- There is a useful teaching technique developed will allow you to demonstrate the SQL queries in the text using MS SQL Server if you have it available.
    - Create a new SQL Server database named Cape-Codd.
    - Use the SQL statements in the *.sql text file DBPe11-MSSQL-Cape-Codd-Create-Tables.sql to create the RETAIL_ORDER, ORDER_ITEM and SKU_DATA tables [the WAREHOUSE and INVENTORY tables, used in the Review Questions, are also created].
    - Use the SQL statements *.sql text file DBPe11-MSSQL-Cape-Dodd-Insert-Data.sql to populate the RETAIL_ORDER, ORDER_ITEM and SKU_DATA tables [the WAREHOUSE and INVENTORY tables, used in the Review Questions, are also populated].
    - Open the Microsoft SQL Server Management Studio and select the Cape-Codd database.
    - In the Microsoft SQL Server Management Studio, open the *.sql text file *DBPe11-MSSQL-Cape-Codd-Query-Set-CH02.sql*. This file contains all the queries shown in the Chapter Two text.
    - Highlight the query you want to run and Execute Query button to display the results of the query. An example of this is shown in the following screenshot on the next page.
    - All of the *.sql text files needed to do this are available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).



- Microsoft Access 2007 does not support all SQL-92 (and newer) constructs. While this chapter still considers Access as the DBMS most likely to be used by students at this point in the course, there are some Review Questions and Project Questions that use the ORDER BY clause with aliased computed columns that will not run in Access (see Review Questions 2.42 – 2.44 and Project Questions 2.63.e – 2.63.g). The correct solutions for these questions were obtained using Microsoft SQL Server 2008. The Access results without the ORDER BY clause are also shown, so you can assign these problems without the ORDER BY part of the questions.
- Microsoft Access 2007 does not support SQL wildcards (see Review Questions 2.36 – 2.38). The correct solutions for these questions were obtained using Microsoft SQL Server 2008.
- For those students that are used to procedural languages, they may have some initial difficulty with a language the does set processing like SQL. These students are accustomed to processing rows (records) rather than sets. It is time well spent to make sure they understand that SQL processes tables at a time, not rows at a time.

- Students have some trouble understanding the GROUP BY clause. If you can explain it in terms of traditional control break logic (sort rows on a key then process the rows until the value of the key changes) they will have less trouble. This also explains why the GROUP BY clause will present the rows sorted even though you do not use an ORDER BY clause.
- At this point, students familiar with Microsoft Access will wonder why they are learning SQL. They have made queries in Access using Access's version of Query-By-Example (QBE), and therefore never had to understand the SQL. In many cases, they will not know that Microsoft Access generates SQL code when you create a query in design view. It is worth letting them know this is done and even showing them the SQL created for and underlying an Access query.
- It is also important for students to understand that, in many cases, the Query-By-Example forms such as Microsoft Access' design view can be very inefficient. Also, the QBE forms are not available from within an application program such as Java or C so SQL must be written.
- It has been our experience that a review of a Cartesian Product from an algebra class is time well spent. Show students what will happen if a WHERE statement is left off of a join. The following example will work. Assume you create four tables with five columns each and 100 rows each. How many columns and rows will be displayed by the statement:

  ```
  SELECT * FROM TABLE1, TABLE2, TABLE3, TABLE4;
  ```

  The result is 20 columns (not bad) but 100,000,000 rows (100 * 100 = 10,000, 10,000 * 100 = 1,00,000, 1,000,000 * 100 = 100,000,000). This happens because the JOIN is not qualified. If they understand Cartesian products then they will understand how to fix a JOIN where the results are much too large.
- Note that in the Marcia's Dry Cleaning project, there is a table named ORDER. This presents the students with an interesting complication, because ORDER is an SQL reserved word (part of ORDER BY). Therefore, when the table name ORDER is used as part of a query, it may need to be ("must be" in Access 2007) enclosed in delimiters as [ORDER] if the query is going to run correctly. The topic of reserved words and delimiters is discussed in more detail in Chapters 6 and 7. However, now is a good time to introduce it to your students. If you do not want your students to have to deal with this situation at this time, rename the ORDER table as CUSTOMER_ORDER in the Marcia's Dry Cleaning project sets.

ANSWERS TO REVIEW QUESTIONS

2.1 What is a business intelligence (BI) system?

A business intelligence (BI) system, is a system used to support management decisions by producing information for assessment, analysis, planning and control.

2.2 What is an ad-hoc query?

An ad-hoc query is a query created by the user as needed, rather than a query programmed into an application.

2.3 What does SQL stand for, and what is SQL?

SQL stands for *Structured Query Language*. SQL is the universal query language for relational DBMS products.

2.4 What does SKU stand for, and what is an SKU?

SKU stands for stock keeping unit. An SKU is a an identifier used to label and distinguish each item sold by a business.

2.5 Summarize how data were altered and filtered in creating the Cape Codd data extraction.

Data from the Cape Codd operational retail sales database was used to create a retail sales extraction database with three tables: RETAIL_ORDER, ORDER_ITEM and SKU_DATA.

The **RETAIL_ORDER** table uses only a few of the columns in the operational database. The structure of the table is:

RETAIL_ORDER (OrderNumber, StoreNumber, StoreZip, OrderMonth, OrderYear, OrderTotal)

For this table, the original column OrderDate (in the data format MM/DD/YYYY [04/26/2005]) was converted into the columns OrderMonth (in a Character(12) format so that each month is spelled out [April]) and OrderYear (in an Integer format with each year appearing as a four-digit year [2005]).

We also note that the OrderTotal column includes tax, shipping and other charges that do not appear in the data extract. Thus, it does not equal the sum of the related ExtendedPrice column in the ORDER_ITEM table discussed below.

The **ORDER_ITEM** table uses an extract of the items purchased for each order. The structure of the table is:

ORDER_ITEM (OrderNumber, SKU, Quantity, Price, ExtendedPrice)

For this table, there is one row for each SKU associated with a given OrderNumber, representing one row for each type of item purchased in a specific order.

The **SKU_DATA** table uses an extract of the item identifying and describing data in the complete operational table. The structure of the table is:

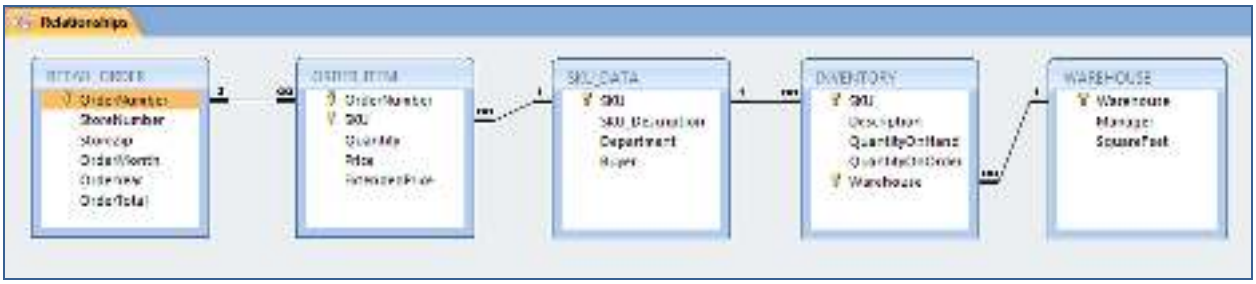
SKU_DATA (SKU, SKU_Description, Department, Buyer)

For this table, there is one row to describe each SKU, representing one particular item that is sold by Cape Codd.

2.6 Explain, in general terms, the relationships of the RETAIL_ORDER, ORDER_ITEM, and SKU_DATA tables.

In general, each sale in RETAIL_ORDER relates to one or more rows in ORDER_ITEM that detail the items sold in the specific order. Each row in ORDER_ITEM is associated with a specific SKU in the SKU_DATA table. Thus one SKU may be associated once with each specific order number, but may also be associated with many different order numbers (as long as it appears only once in each order).

Using the Microsoft Access Relationship window, the relationships (including the additional relationships with the INVENTORY and WAREHOUSE tables described after Review Question 2.15) look like this:



In traditional database terms (which will be discussed Chapter 6) OrderNumber and SKU in ORDER_ITEM are foreign keys that provide the links to the RETAIL_ORDER and SKU_DATA tables respectively. Using an underline to show primary keys and italics to show foreign keys, the tables and their relationships are shown as:

RETAIL_ORDER (OrderNumber, StoreNumber, StoreZip, OrderMonth, OrderYear, OrderTotal)

ORDER_ITEM (*OrderNumber*, *SKU*, Quantity, Price, ExtendedPrice)

SKU_DATA (SKU, SKU_Description, Department, Buyer)

2.7 Summarize the background of SQL.

SQL was developed by IBM in the late 1970s, and in 1992 it was endorsed as a national standard by the American National Standards Institute (ANSI). That version is called SQL-92. There is a later version called SQL3 that has some object-oriented concepts, but SQL3 has not received much commercial attention.

2.8 What is SQL-92? How does it relate to the SQL statements in this chapter?

SQL-92 is the version of SQL endorsed as a national standard by the American National Standards Institute (ANSI) in 1992. It is the version of SQL supported by most commonly used database management systems. The SQL statements in the chapter are based on SQL-92.

2.9 What features have been added to SQL in versions subsequent to the SQL-92?

Versions of SQL subsequent to SQL-92 have extended features or added new features to SQL, the most important of which, for our purposes, is support for Extensible Markup Language (XML).

2.10 Why is SQL described as a data sublanguage?

A data sublanguage consists only of language statements for defining and processing a database. To obtain a full programming language, SQL statements must be embedded in scripting languages such as VBScript or in programming languages such as Java or C#.

2.11 What does DML stand for? What are DML statements?

DML stands for *data manipulation language*. DML statements are used for querying and modifying data.

2.12 What does DDL stand for? What are DDL statements?

DDL stands for *data definition language*. DDL statements are used for creating tables, relationships and other database querying and modifying data.

2.13 What is the SQL SELECT/FROM/WHERE framework?

The SQL SELECT/FROM/WHERE framework is the basis for queries in SQL. In this framework:

- The SQL SELECT clause specifies which columns are to be listed in the query results.
- The SQL FROM clause specifies which tables are to be used in the query.
- The SQL WHERE clause specifies which rows are to be listed in the query results.

2.14 Explain how Access uses SQL.

Access uses SQL, but generally hides the SQL from the user. For example, Access automatically generates SQL and sends it to the Access Jet DBMS every time you run a query, process a form or create a report. To go beyond elementary database processing, you need to know how to use SQL in Access.

2.15 Explain how enterprise-class DBMS products use SQL.

Enterprise-class DBMS products, which include Microsoft SQL Server, Oracle Corporation's Oracle, IBM's DB2 and MySQL's MySQL, require you to know and use SQL. All data manipulation is expressed in SQL in these products.

The Cape Codd Outdoor Sports sale extraction database has been modified to include two additional tables, the INVENTORY table and the WAREHOUSE table. The table schemas for these tables, together with the SKU table, are as follows:

SKU_DATA (SKU, SKU_Description, Department, Buyer)

INVENTORY (SKU, Warehouse, SKU_Description, QuantityOnHand, QuantityOnOrder)

WAREHOUSE (Warehouse, Manager, Squarefeet)

The column characteristics for the WAREHOUSE table are shown in Figure 2-22, and the column characteristics for the INVENTORY table are shown in Figure 2-23. The data for the WAREHOUSE table are shown in Figure 2-24, and the data for the INVENTORY table is shown in Figure 2-25.

Column Name	Type	Key	Required	Remarks
Warehouse	Text (30)	Primary Key	Yes	
Manager	Text (25)	No	No	
SquareFeet	Integer	No	No	

Figure 2-22 - Column Characteristics for the WAREHOUSE Table

Column Name	Type	Key	Required	Remarks
SKU	Integer	Primary Key, Foreign Key	Yes	Surrogate Key
Warehouse	Text (30)	Primary Key, Foreign Key	Yes	
SKU_Description	Text (35)	No	Yes	
QuantityOnHand	Integer	No	No	
QuantityOnOrder	Integer	No	No	

Figure 2-23 - Column Characteristics for the INVENTORY Table

Warehouse	Manager	SquareFeet
Atlanta	Jones	125,000
Chicago	Smith	100,000
New Jersey	Evans	150,000
Seattle	Rogers	130,000

Figure 2-24 - Cape Codd Outdoor Sports WAREHOUSE Data

[Figure 2-25 is on the following page]

If at all possible, you should run your SQL solutions to the following questions against an actual database. A Microsoft Access database named Cape-Codd.accdb is available on our Web site (www.pearsonhighered.com/kroenke) that contains all the tables and data for the Cape Codd Outdoor Sports sales data extract database. Also available on our Web site are SQL scripts for creating and populating the tables for the Cape Codd database in SQL Server, Oracle, and MySQL.

NOTE: All answers below show the correct SQL statement, as well as SQL statements modified for Microsoft Access 2007 when needed. All results were obtained by running the SQL statements in Microsoft Access 2007, and the corresponding screen shots of the results are shown below. As explained in the text, some queries cannot be run in Microsoft Access 2007, and for those queries the correct result was obtained using Microsoft SQL Server 2008. The SQL statements shown should run with little, if any, modification needed for Oracle Database 11g and MySQL 5.1.

SKU	Warehouse	SKU_Description	QuantityOnHand	QuantityOnOrder
100100	Atlanta	Std. Scuba Tank, Yellow	250	0
100100	Chicago	Std. Scuba Tank, Yellow	100	50
100100	New Jersey	Std. Scuba Tank, Yellow	100	0
100100	Seattle	Std. Scuba Tank, Yellow	200	0
100200	Atlanta	Std. Scuba Tank, Magenta	200	30
100200	Chicago	Std. Scuba Tank, Magenta	75	75
100200	New Jersey	Std. Scuba Tank, Magenta	100	100
100200	Seattle	Std. Scuba Tank, Magenta	250	0
101100	Atlanta	Dive Mask, Small Clear	0	500
101100	Chicago	Dive Mask, Small Clear	0	500
101100	New Jersey	Dive Mask, Small Clear	300	200
101100	Seattle	Dive Mask, Small Clear	450	0
101200	Atlanta	Dive Mask, Med Clear	100	500
101200	Chicago	Dive Mask, Med Clear	50	500
101200	New Jersey	Dive Mask, Med Clear	475	0
101200	Seattle	Dive Mask, Med Clear	250	250
201000	Atlanta	Half-Dome Tent	2	100
201000	Chicago	Half-Dome Tent	10	250
201000	New Jersey	Half-Dome Tent	250	0
201000	Seattle	Half-Dome Tent	0	250
202000	Atlanta	Half-Dome Tent Footprint	10	250
202000	Chicago	Half-Dome Tent Footprint	1	250
202000	New Jersey	Half-Dome Tent Footprint	100	0
202000	Seattle	Half-Dome Tent Footprint	0	200
301000	Atlanta	Light Fly Climbing Harness	300	250
301000	Chicago	Light Fly Climbing Harness	250	250
301000	New Jersey	Light Fly Climbing Harness	0	250
301000	Seattle	Light Fly Climbing Harness	0	250
302000	Atlanta	Locking Carabiner	1000	0
302000	Chicago	Locking Carabiner	1250	0
302000	New Jersey	Locking Carabiner	500	500
302000	Seattle	Locking Carabiner	0	1000

Figure 2-24 - Cape Codd Outdoor Sports INVENTORY Data

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

NOTE: If your students are using a DBMS other than Microsoft ACCESS, and need to create the INVENTORY and WAREHOUSE tables, use the SQL code shown here to create and populate the tables.

SQL code to create the tables is shown below. This code is also contained in the *.sql text file *DBP-e11-MSSQL-Cape-Codd-Create-Tables.sql* available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
CREATE TABLE WAREHOUSE (
      Warehouse         Char (30)    NOT NULL,
      Manager           Char (30)    NOT NULL,
      SquareFeet        Integer      NOT NULL,
      CONSTRAINT  WAREHOUSE_PK PRIMARY KEY (Warehouse)
      );

CREATE TABLE INVENTORY (
      SKU               Integer      NOT NULL,
      Warehouse         Char (30)    NOT NULL,
      Description       Char (35)    NOT NULL,
      QuantityOnHand    Integer      NOT NULL,
      QuantityOnOrder   Integer      NULL,
      CONSTRAINT INVENTORY_PK PRIMARY KEY (SKU, Warehouse),
      CONSTRAINT SKU_INV_Relationship Foreign Key (SKU)
            REFERENCES SKU_DATA (SKU),
      CONSTRAINT Warehouse_Relationship Foreign Key (Warehouse)
            REFERENCES WAREHOUSE (Warehouse)
      );
```

SQL code to insert the data into the tables is shown below. This code is also contained in the *.sql text file *DBP-e11-MSSQL-Cape-Codd-Insert-Data.sql* available in the the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
INSERT INTO WAREHOUSE VALUES (
   'Atlanta', 'Jones', 125000);
INSERT INTO WAREHOUSE VALUES (
   'Chicago', 'Smith', 100000);
INSERT INTO WAREHOUSE VALUES (
   'New Jersey', 'Evans', 150000);
INSERT INTO WAREHOUSE VALUES (
   'Seattle', 'Rogers', 130000);

INSERT INTO INVENTORY VALUES (
   100100, 'Atlanta', 'Std. Scuba Tank, Yellow', 250, 0);
INSERT INTO INVENTORY VALUES (
   100100, 'Chicago', 'Std. Scuba Tank, Yellow', 100, 50);
INSERT INTO INVENTORY VALUES (
   100100, 'New Jersey', 'Std. Scuba Tank, Yellow', 100, 0);
INSERT INTO INVENTORY VALUES (
   100100, 'Seattle', 'Std. Scuba Tank, Yellow', 200, 0);
```

```
INSERT INTO INVENTORY VALUES (
   100200, 'Atlanta', 'Std. Scuba Tank, Magenta', 200, 30);
INSERT INTO INVENTORY VALUES (
   100200, 'Chicago', 'Std. Scuba Tank, Magenta', 75, 75);
INSERT INTO INVENTORY VALUES (
   100200, 'New Jersey', 'Std. Scuba Tank, Magenta', 100, 100);
INSERT INTO INVENTORY VALUES (
   100200, 'Seattle', 'Std. Scuba Tank, Magenta', 250, 0);
INSERT INTO INVENTORY VALUES (
   101100, 'Atlanta', 'Dive Mask, Small Clear', 0, 500);
INSERT INTO INVENTORY VALUES (
   101100, 'Chicago', 'Dive Mask, Small Clear', 0, 500);
INSERT INTO INVENTORY VALUES (
   101100, 'New Jersey', 'Dive Mask, Small Clear', 300, 200);
INSERT INTO INVENTORY VALUES (
   101100, 'Seattle', 'Dive Mask, Small Clear', 450, 0);
INSERT INTO INVENTORY VALUES (
   101200, 'Atlanta', 'Dive Mask, Med Clear', 100, 500);
INSERT INTO INVENTORY VALUES (
   101200, 'Chicago', 'Dive Mask, Med Clear', 50, 500);
INSERT INTO INVENTORY VALUES (
   101200, 'New Jersey', 'Dive Mask, Med Clear', 475, 0);
INSERT INTO INVENTORY VALUES (
   101200, 'Seattle', 'Dive Mask, Med Clear', 250, 250);
INSERT INTO INVENTORY VALUES (
   201000, 'Atlanta', 'Half-dome Tent', 2, 100);
INSERT INTO INVENTORY VALUES (
   201000, 'Chicago', 'Half-dome Tent', 10, 250);
INSERT INTO INVENTORY VALUES (
   201000, 'New Jersey', 'Half-dome Tent', 250, 0);
INSERT INTO INVENTORY VALUES (
   201000, 'Seattle', 'Half-dome Tent', 0, 250);
INSERT INTO INVENTORY VALUES (
   202000, 'Atlanta', 'Half-dome Tent Footprint', 10, 250);
INSERT INTO INVENTORY VALUES (
   202000, 'Chicago', 'Half-dome Tent Footprint', 1, 250);
INSERT INTO INVENTORY VALUES (
   202000, 'New Jersey', 'Half-dome Tent Footprint', 100, 0);
INSERT INTO INVENTORY VALUES (
   202000, 'Seattle', 'Half-dome Tent Footprint', 0, 200);
INSERT INTO INVENTORY VALUES (
   301000, 'Atlanta', 'Light Fly Climbing Harness', 300, 250);
INSERT INTO INVENTORY VALUES (
   301000, 'Chicago', 'Light Fly Climbing Harness', 250, 250);
INSERT INTO INVENTORY VALUES (
   301000, 'New Jersey', 'Light Fly Climbing Harness', 0, 250);
INSERT INTO INVENTORY VALUES (
   301000, 'Seattle', 'Light Fly Climbing Harness', 0, 250);
INSERT INTO INVENTORY VALUES (
   302000, 'Atlanta', 'Locking carabiner', 1000, 0);
INSERT INTO INVENTORY VALUES (
   302000, 'Chicago', 'Locking carabiner', 1250, 0);
INSERT INTO INVENTORY VALUES (
   302000, 'New Jersey', 'Locking carabiner', 500, 500);
INSERT INTO INVENTORY VALUES (
   302000, 'Seattle', 'Locking carabiner', 0, 1000);
```

2.16 There is an intentional flaw in the design of the INVENTORY table used in these exercises. This flaw was purposely included in the INVENTORY tables so that you can answer some of the following questions using only that table. Compare the SKU and INVENTORY tables, and determine what design flaw is included in INVENTORY. Specifically, why did we include it?

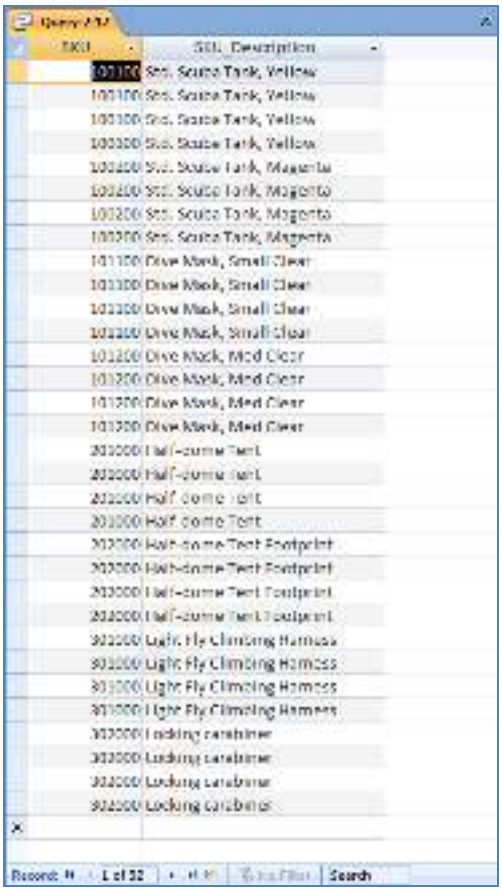
The flaw is the inclusion of the SKU_Description attribute in the INVENTORY table. This attribute duplicates the SKU_Description attribute and data in the SKU_DATA table, where the attribute rightfully belongs. By duplicating SKU_Description in the INVENTORY table, we can ask you to list the SKU and its associated description in a single table query against the INVENTORY table. Otherwise, a two table query would be required. If these tables were in a production database, we would eliminate the INVENTORY.SKU_Description column.

Use only the INVENTORY table to answer Review Questions 2.17 through 2.46:

2.17 Write an SQL statement to display SKU and SKU_Description.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

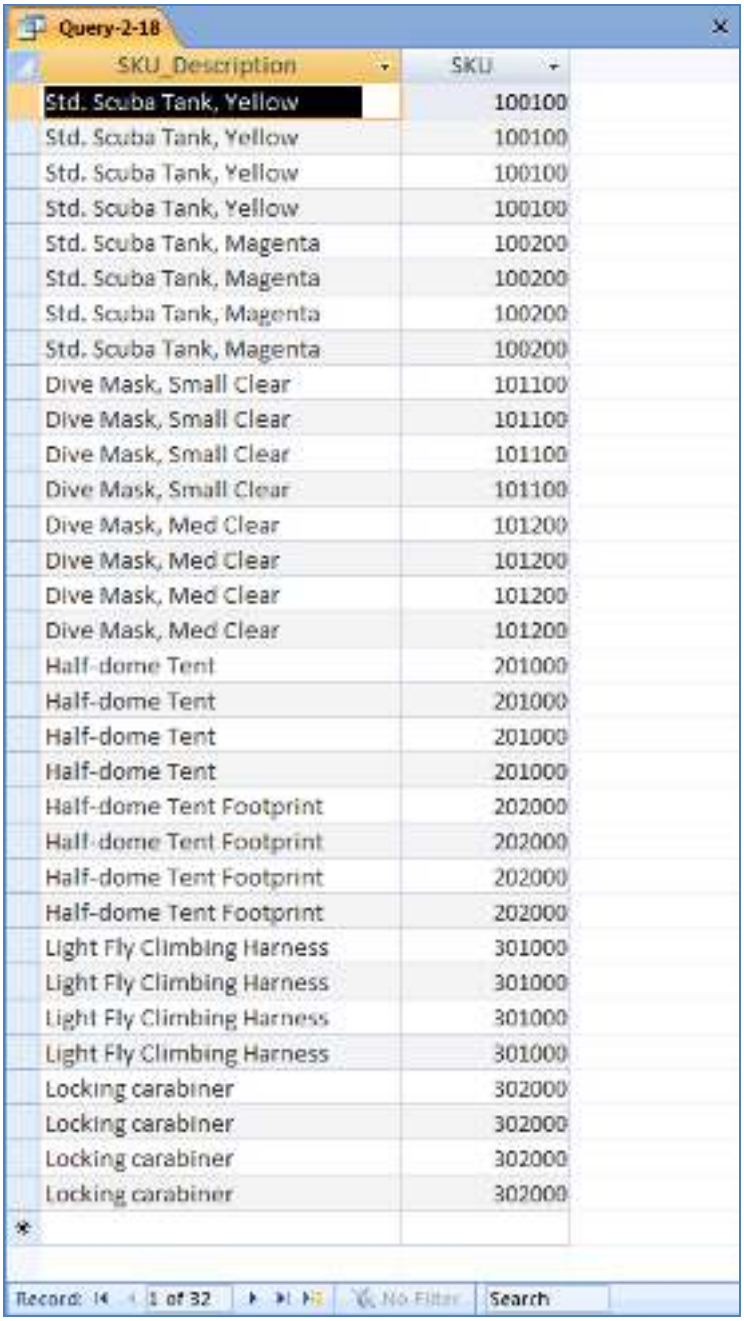
```
SELECT   SKU, SKU_Description
FROM     INVENTORY;
```



2.18 Write an SQL statement to display SKU_Description and SKU.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   SKU_Description, SKU
FROM     INVENTORY;
```



Query-2-18

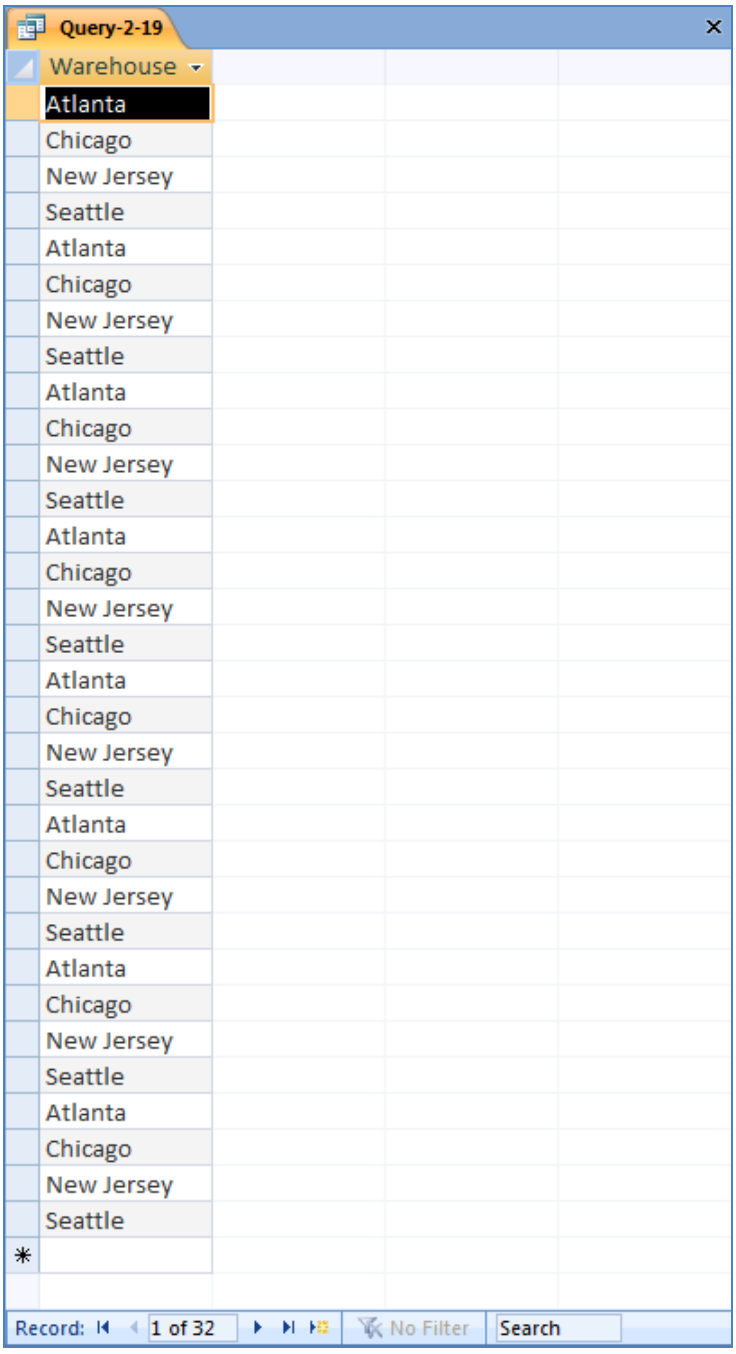
SKU_Description	SKU
Std. Scuba Tank, Yellow	100100
Std. Scuba Tank, Yellow	100100
Std. Scuba Tank, Yellow	100100
Std. Scuba Tank, Yellow	100100
Std. Scuba Tank, Magenta	100200
Std. Scuba Tank, Magenta	100200
Std. Scuba Tank, Magenta	100200
Std. Scuba Tank, Magenta	100200
Dive Mask, Small Clear	101100
Dive Mask, Small Clear	101100
Dive Mask, Small Clear	101100
Dive Mask, Small Clear	101100
Dive Mask, Med Clear	101200
Dive Mask, Med Clear	101200
Dive Mask, Med Clear	101200
Dive Mask, Med Clear	101200
Half-dome Tent	201000
Half-dome Tent	201000
Half-dome Tent	201000
Half-dome Tent	201000
Half-dome Tent Footprint	202000
Half-dome Tent Footprint	202000
Half-dome Tent Footprint	202000
Half-dome Tent Footprint	202000
Light Fly Climbing Harness	301000
Light Fly Climbing Harness	301000
Light Fly Climbing Harness	301000
Light Fly Climbing Harness	301000
Locking carabiner	302000
Locking carabiner	302000
Locking carabiner	302000
Locking carabiner	302000

Record: 1 of 32 No Filter Search

2.19 Write an SQL statement to display Warehouse.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

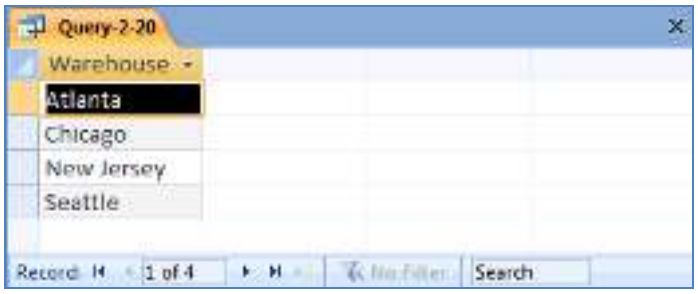
```
SELECT    Warehouse
FROM      INVENTORY;
```



2.20 Write an SQL statement to display Warehouse with no duplications.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

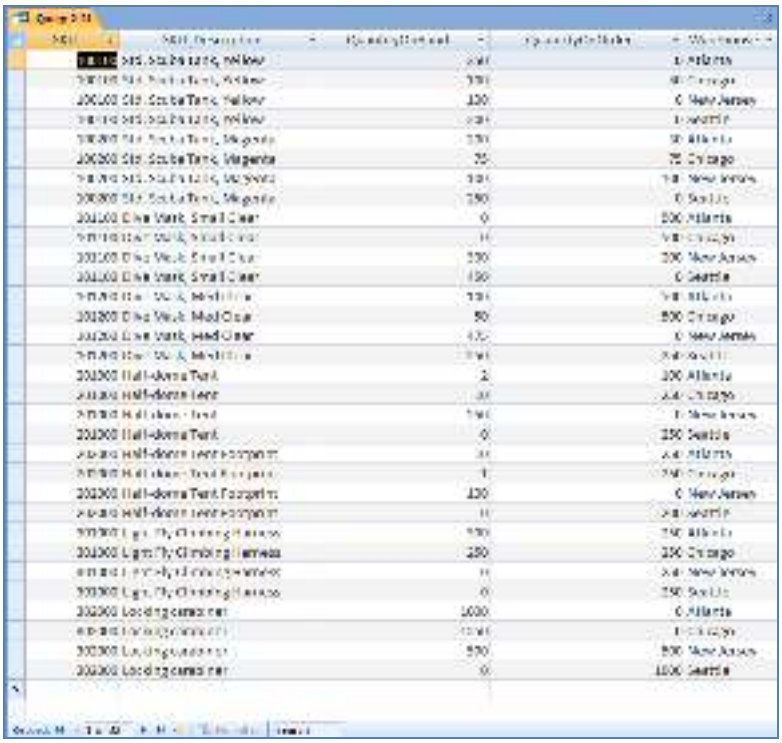
```
SELECT    DISTINCT Warehouse
FROM      INVENTORY;
```



*2.21 Write an SQL statement to display all of the columns without using *.*

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    SKU, SKU_Description, QuantityOnHand, QuantityOnOrder,
Warehouse
FROM      INVENTORY;
```



2.22 *Write an SQL statement to display all of the columns using *.*

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    *
FROM      INVENTORY;
```

Query 2 22

SKU	Warehouse	SKU_Description	QuantityOnHand	QuantityOnOrder
100100	Atlanta	Std. Scuba Tank, Yellow	250	0
100100	Chicago	Std. Scuba Tank, Yellow	100	50
100100	New Jersey	Std. Scuba Tank, Yellow	100	0
100100	Seattle	Std. Scuba Tank, Yellow	200	0
100200	Atlanta	Std. Scuba Tank, Magenta	200	50
100200	Chicago	Std. Scuba Tank, Magenta	75	75
100200	New Jersey	Std. Scuba Tank, Magenta	100	100
100200	Seattle	Std. Scuba Tank, Magenta	250	0
101100	Atlanta	Dive Mask, Small Clear	0	500
101100	Chicago	Dive Mask, Small Clear	0	500
101100	New Jersey	Dive Mask, Small Clear	300	200
101100	Seattle	Dive Mask, Small Clear	450	0
101200	Atlanta	Dive Mask, Med Clear	100	500
101200	Chicago	Dive Mask, Med Clear	50	500
101200	New Jersey	Dive Mask, Med Clear	475	0
101200	Seattle	Dive Mask, Med Clear	250	250
201000	Atlanta	Half-dome Tent	2	100
201000	Chicago	Half-dome Tent	10	250
201000	New Jersey	Half-dome Tent	250	0
201000	Seattle	Half-dome Tent	0	250
202000	Atlanta	Half-dome Tent Footprint	10	250
202000	Chicago	Half-dome Tent Footprint	1	250
202000	New Jersey	Half-dome Tent Footprint	100	0
202000	Seattle	Half-dome Tent Footprint	0	200
301000	Atlanta	Light Fly Climbing Harness	300	250
301000	Chicago	Light Fly Climbing Harness	250	250
301000	New Jersey	Light Fly Climbing Harness	0	250
301000	Seattle	Light Fly Climbing Harness	0	250
302000	Atlanta	Locking carabiner	1000	0
302000	Chicago	Locking carabiner	1250	0
302000	New Jersey	Locking carabiner	500	500
302000	Seattle	Locking carabiner	0	1000

2.23 Write an SQL statement to display all data on products having a QuantityOnHand greater than 0.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    *
FROM      INVENTORY
WHERE     QuantityOnHand >0;
```

Query 2.23

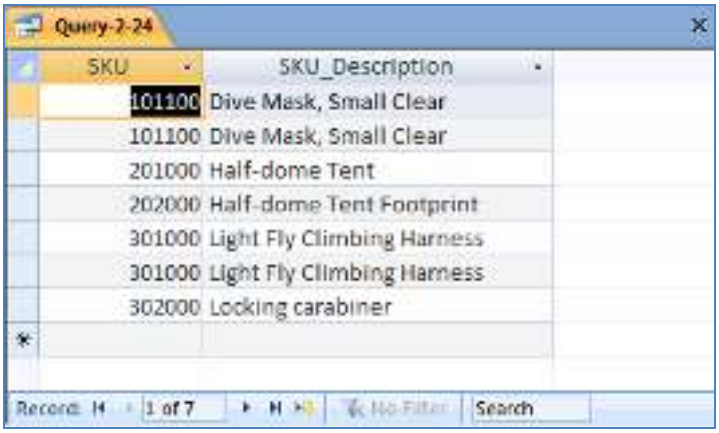
SKU	Warehouse	SKU_Description	QuantityOnHand	QuantityOnOrder
100100	Atlanta	Std. Scuba Tank, Yellow	250	0
100100	Chicago	Std. Scuba Tank, Yellow	100	50
100100	New Jersey	Std. Scuba Tank, Yellow	100	0
100100	Seattle	Std. Scuba Tank, Yellow	200	0
100200	Atlanta	Std. Scuba Tank, Magenta	200	50
100200	Chicago	Std. Scuba Tank, Magenta	75	75
100200	New Jersey	Std. Scuba Tank, Magenta	100	100
100200	Seattle	Std. Scuba Tank, Magenta	250	0
101100	New Jersey	Dive Mask, Small Clear	300	200
101100	Seattle	Dive Mask, Small Clear	450	0
101200	Atlanta	Dive Mask, Med Clear	100	500
101200	Chicago	Dive Mask, Med Clear	50	500
101200	New Jersey	Dive Mask, Med Clear	475	0
101200	Seattle	Dive Mask, Med Clear	250	250
201000	Atlanta	Half-dome Tent	2	100
201000	Chicago	Half-dome Tent	10	250
201000	New Jersey	Half-dome Tent	250	0
202000	Atlanta	Half-dome Tent Footprint	10	250
202000	Chicago	Half-dome Tent Footprint	1	250
202000	New Jersey	Half-dome Tent Footprint	100	0
301000	Atlanta	Light Fly Climbing Harness	300	250
301000	Chicago	Light Fly Climbing Harness	250	250
302000	Atlanta	Locking carabiner	1000	0
302000	Chicago	Locking carabiner	1250	0
302000	New Jersey	Locking carabiner	500	500

Record: 1 of 25 Search

2.24 Write an SQL statement to display the SKU and Description on products having QuantityOnHand equal to 0.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

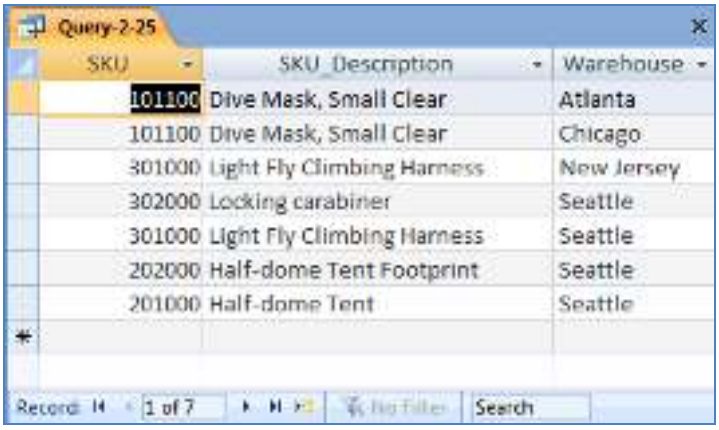
```
SELECT    SKU, SKU_Description
FROM      INVENTORY
WHERE     QuantityOnHand =0;
```



2.25 Write an SQL statement to display the SKU, SKU_Description, and Warehouse on products having QuantityOnHand equal to 0. Sort the results in ascending order by Warehouse.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

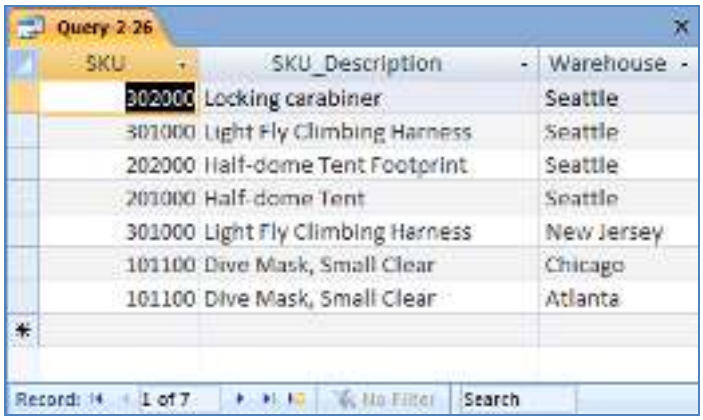
```
SELECT    SKU, SKU_Description, Warehouse
FROM      INVENTORY
WHERE     QuantityOnHand =0
ORDER BY Warehouse;
```



2.26 Write an SQL statement to display the SKU, SKU_Description, and Warehouse on products having QuantityOnHand equal to 0. Sort the results in descending order by Warehouse.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

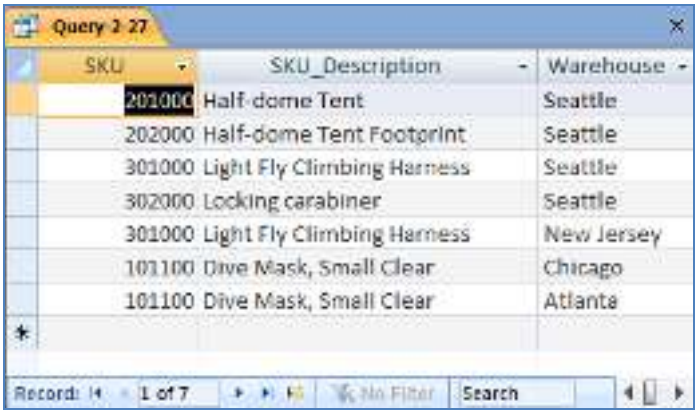
```
SELECT   SKU, SKU_Description, Warehouse
FROM     INVENTORY
WHERE    QuantityOnHand =0
ORDER BY Warehouse DESC;
```



*2.27 Write an SQL statement to display the SKU, SKU_Description, and Warehouse on products having QuantityOnHand equal to 0. Sort the results in descending order by Warehouse and ascending order of **SKU**.*

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

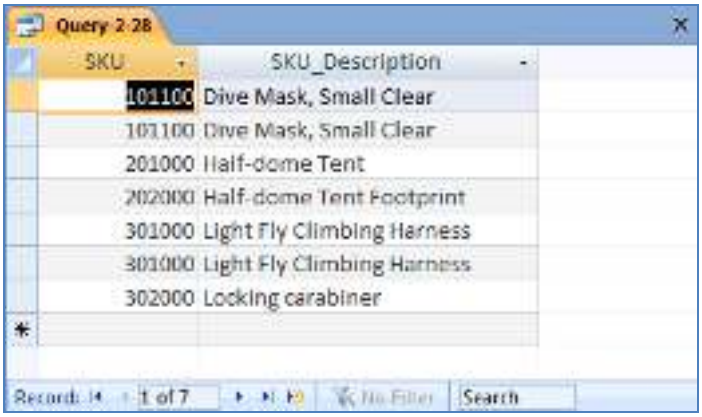
```
SELECT   SKU, SKU_Description, Warehouse
FROM     INVENTORY
WHERE    QuantityOnHand =0
ORDER BY Warehouse DESC, SKU;
```



2.28 Write an SQL statement to display SKU and SKU_Description for all products that have a QuantityOnHand equal to 0 and a QuantityOnOrder greater than 0.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   SKU, SKU_Description
FROM     INVENTORY
WHERE    QuantityOnHand =0
   AND   QuantityOnOrder > 0;
```



2.29 Write an SQL statement to display SKU and SKU_Description for all products that have a QuantityOnHand equal to 0 or QuantityOnOrder equal to 0.

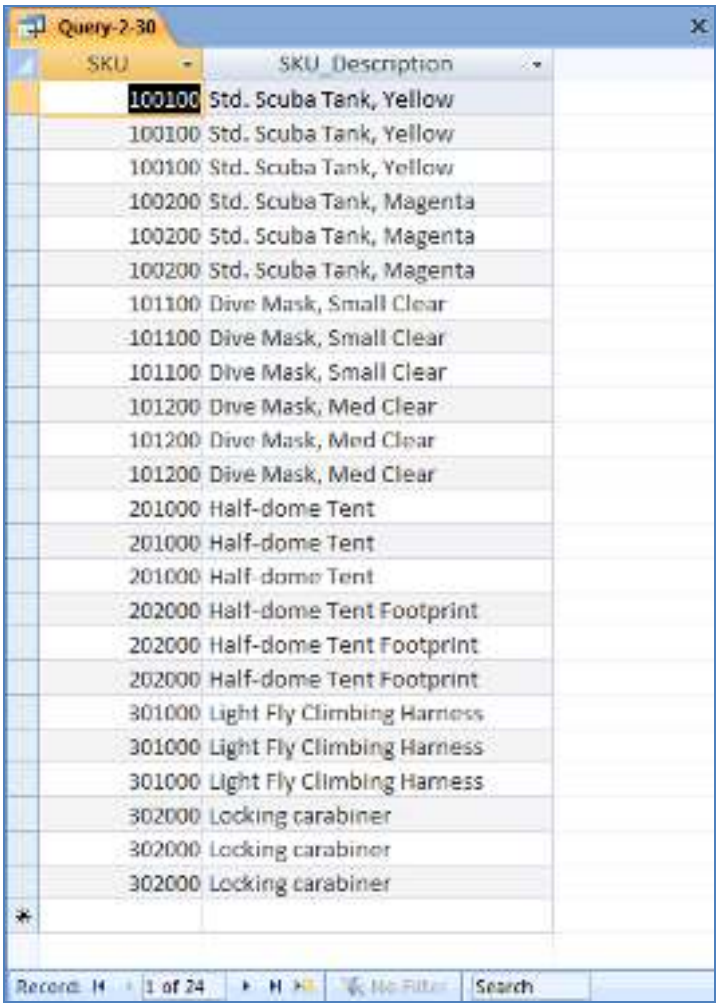
Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   SKU, SKU_Description
FROM     INVENTORY
WHERE    QuantityOnHand =0
   OR    QuantityOnOrder = 0;
```

2.30 Write an SQL statement to display the SKU and SKU_Description of all items stored in the Seattle, Chicago, or New Jersey warehouse. Do not use the IN keyword.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   SKU, SKU_Description
FROM     INVENTORY
WHERE    Warehouse = 'Seattle'
   OR    Warehouse = 'Chicago'
   OR    Warehouse = 'New Jersey';
```



2.31 Write an SQL statement to display the SKU and SKU_Description of all items stored in the Seattle, Chicago, or New Jersey warehouse. Use the IN keyword.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

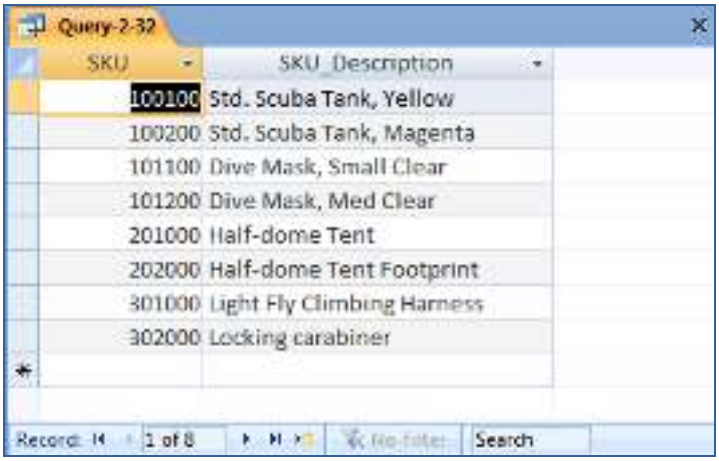
```
SELECT    SKU, SKU_Description
FROM      INVENTORY
WHERE     Warehouse IN ('Seattle', 'Chicago', 'New Jersey');
```

2.32 Write an SQL statement to display the SKU and Description of all items not stored in the Seattle, Chicago, or New Jersey warehouse. Do not use the NOT IN keyword.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

NOTE: The symbol for "not equal to" is < >. Since we want the SKU and Description for warehouses that are not Seattle or Chicago or New Jersey as a set, we must ask for warehouses that are not in the group (Seattle **and** Chicago **and** New Jersey). This means we use AND in the WHERE clause – if we used OR in the WHERE clause, we would end up with ALL warehouses being in the query output. This happens because each OR eliminates only one warehouse, but that warehouse still qualifies for inclusion in the other OR statements. To demonstrate this, substitute OR for each AND in the SQL statement below.

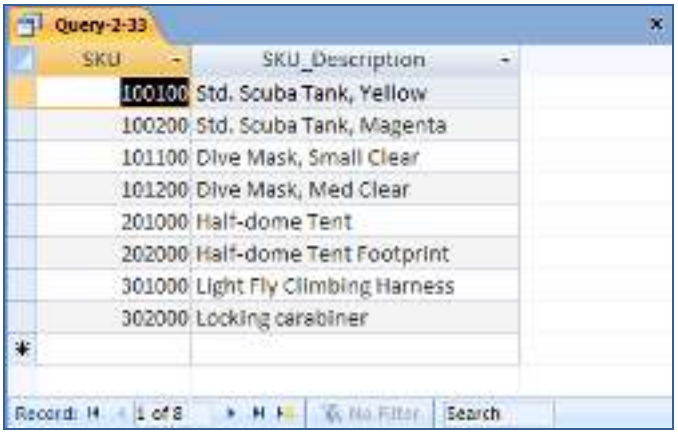
```
SELECT   SKU, SKU_Description
FROM     INVENTORY
WHERE    Warehouse <> 'Seattle'
   AND   Warehouse <> 'Chicago'
   AND   Warehouse <> 'New Jersey';
```



2.33 Write an SQL statement to display the SKU and Description of all items not stored in the Seattle, Chicago, or New Jersey warehouse. Use the NOT IN keyword.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   SKU, SKU_Description
FROM     INVENTORY
WHERE    Warehouse NOT IN ('Seattle', 'Chicago', 'New Jersey');
```

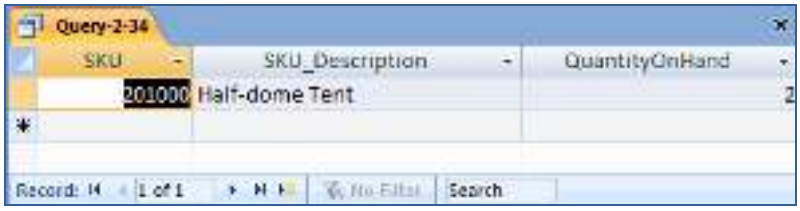


2.34 Write an SQL statement to display the SKU, SKU_Description, and QuantityOnHand for all products having a QuantityOnHand greater than 1 and less than 10. Do not use the BETWEEN keyword.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

Since we can't use the BETWEEN keyword, we'll have to use a set of OR clauses:

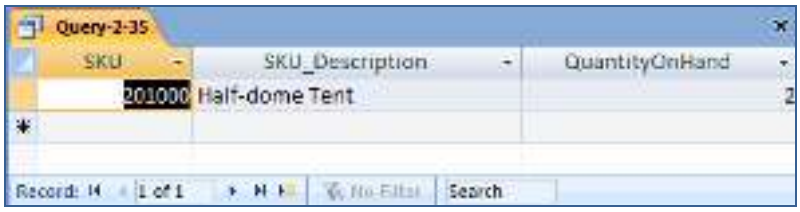
```
SELECT   SKU, SKU_Description, QuantityOnHand
FROM     INVENTORY
WHERE    QuantityOnHand = 2
   OR    QuantityOnHand = 3
   OR    QuantityOnHand = 4
   OR    QuantityOnHand = 5
   OR    QuantityOnHand = 6
   OR    QuantityOnHand = 7
   OR    QuantityOnHand = 8
   OR    QuantityOnHand = 9;
```



2.35 Write an SQL statement to display the SKU, SKU_Description, and QuantityOnHand for all products having a QuantityOnHand greater than 1 and less than 10. Use the BETWEEN keyword.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    SKU, SKU_Description, QuantityOnHand
FROM      INVENTORY
WHERE     QuantityOnHand BETWEEN 2 AND 9;
```



2.36 Write an SQL statement to show SKU and SKU_Description for all products having an SKU_Description starting with "Half-dome".

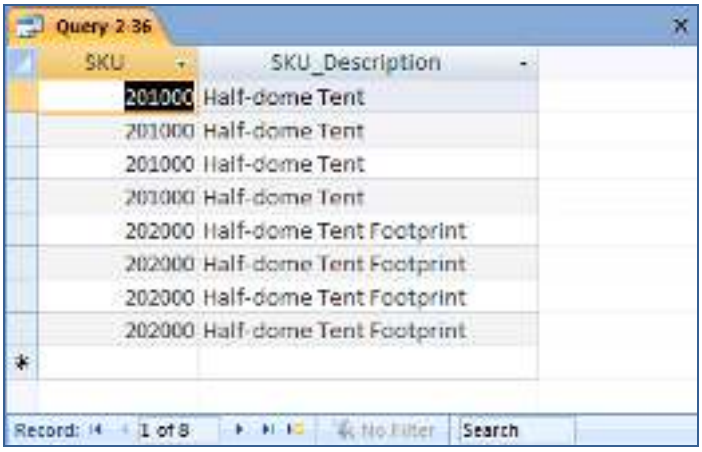
Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

The correct SQL statement, which uses the wildcard % for multiple characters, is:

```
SELECT    SKU, SKU_Description
FROM      INVENTORY
WHERE     SKU_Description LIKE 'Half-dome%';
```

However, Microsoft Access uses the wildcard *, resulting in the following SQL statement:

```
SELECT    SKU, SKU_Description
FROM      INVENTORY
WHERE     SKU_Description LIKE 'Half-dome*';
```



2.37 Write an SQL statement to show SKU and SKU_Description for all products having Description that includes the word "Foot".

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

The correct SQL statement, which uses the wildcard % for multiple characters, is:

```
SELECT   SKU, SKU_Description
FROM     INVENTORY
WHERE    SKU_Description LIKE '%Foot%';
```

However, Microsoft Access uses the wildcard *, which give the following SQL statement:

```
SELECT   SKU, SKU_Description
FROM     INVENTORY
WHERE    SKU_Description LIKE '*Foot*';
```

2.38 Write an SQL statement to show SKU and Warehouse for all products having a 'w' in the third position from the left in Warehouse.

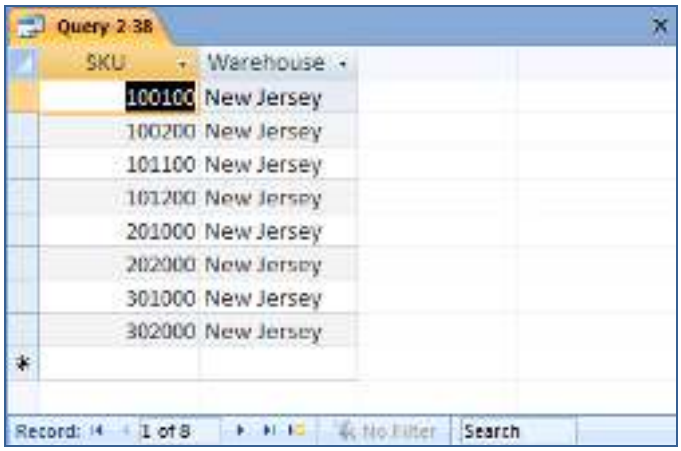
Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

The correct SQL statement, which uses the wildcards % (multiple characters) and _ (a single character), is:

```
SELECT   SKU, Warehouse
FROM     INVENTORY
WHERE    Warehouse LIKE '__w%';
```

However, Microsoft Access uses the wildcards * (multiple characters) and ? (a single character), which give the following SQL statement:

```
SELECT   SKU, Warehouse
FROM     INVENTORY
WHERE    Warehouse LIKE '??w*';
```



2.39 Write an SQL statement that uses all of the built-in functions on the QuantityOnHand column. Include meaningful column names in the result.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   COUNT (QuantityOnHand) AS Number_Of_Records,
         SUM (QuantityOnHand) AS Total_Number_Of_Items_On_Hand,
         AVG (QuantityOnHand) AS Ave_Number_Of_Items_On_Hand,
         MAX (QuantityOnHand) AS Max_Number_Of_Items_On_Hand,
         MIN (QuantityOnHand) AS Min_Number_Of_Items_On_Hand
FROM     INVENTORY;
```

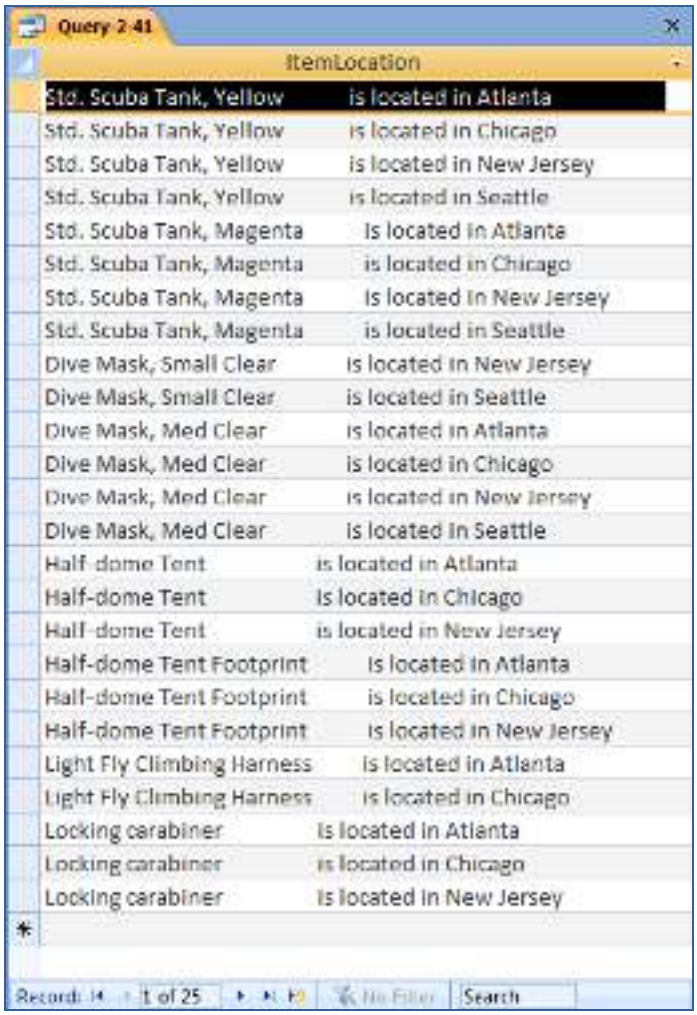
2.40 Explain the difference between the SQL buit-in functions COUNT and SUM.

COUNT counts the number of rows or records in a table, while SUM adds up the data values in the specified column.

2.41 Write an SQL statement to produce a single column called ItemLocation that combines the SKU_Description, the phrase "is located in", and Warehouse for all products that have a QuantityOnHand greater than 0. Do not be concerned with removing leading or trailing blanks.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   SKU_Description+' is located in '+Warehouse AS ItemLocation
FROM     INVENTORY
WHERE    QuantityOnHand > 0;
```



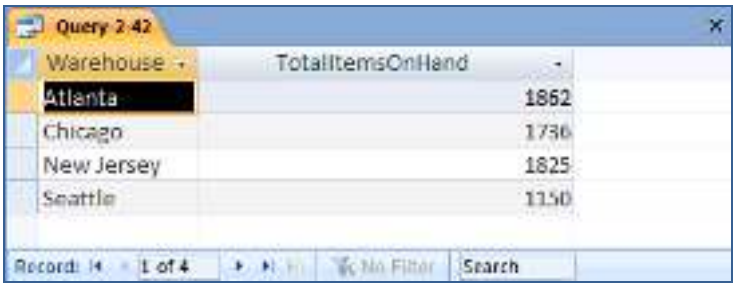
2.42 Write an SQL statement to display the Warehouse and a count of QuantityOnHand, grouped by Warehouse. Name the count TotalItemsOnHand and display the results in descending order of TotalItemsOnHand.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

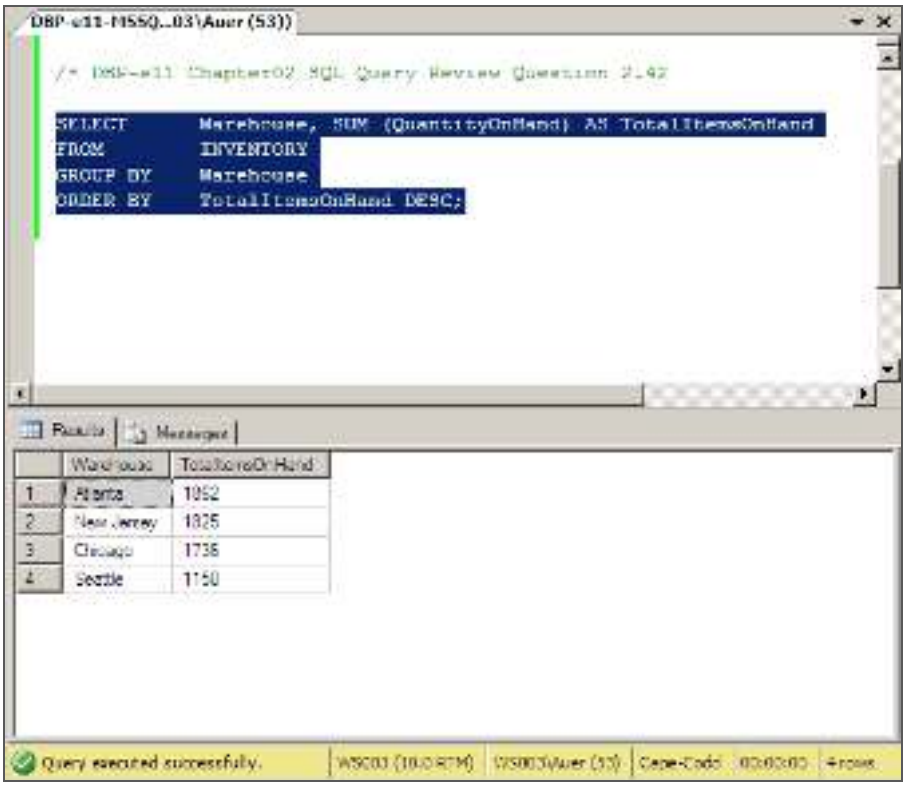
Note that "a count of" actually means the "sum" in this context. The correct SQL Statement is:

```
SELECT   Warehouse, SUM (QuantityOnHand) AS TotalItemsOnHand
FROM     INVENTORY
GROUP BY Warehouse
ORDER BY TotalItemsOnHand DESC;
```

Unfortunately, Microsoft Access cannot process the ORDER BY clause because it contains an aliased computed result. The Microsoft Access result **without** the ORDER BY clause is:



The correct results, obtained from SQL Server 2008, are:



2.43 Write an SQL statement to display the Warehouse and a count of QuantityOnHand, grouped by Warehouse. Omit all items that have a count greater than 2. Name the count TotalItemsOnHand and display the results in descending order of TotalItemsOnHand.

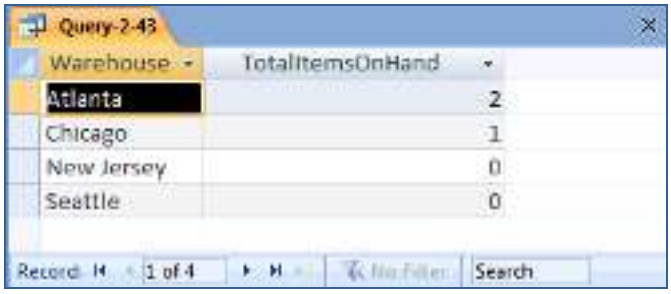
Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that "a count of" actually means the "sum" in this context. The correct SQL Statement is:

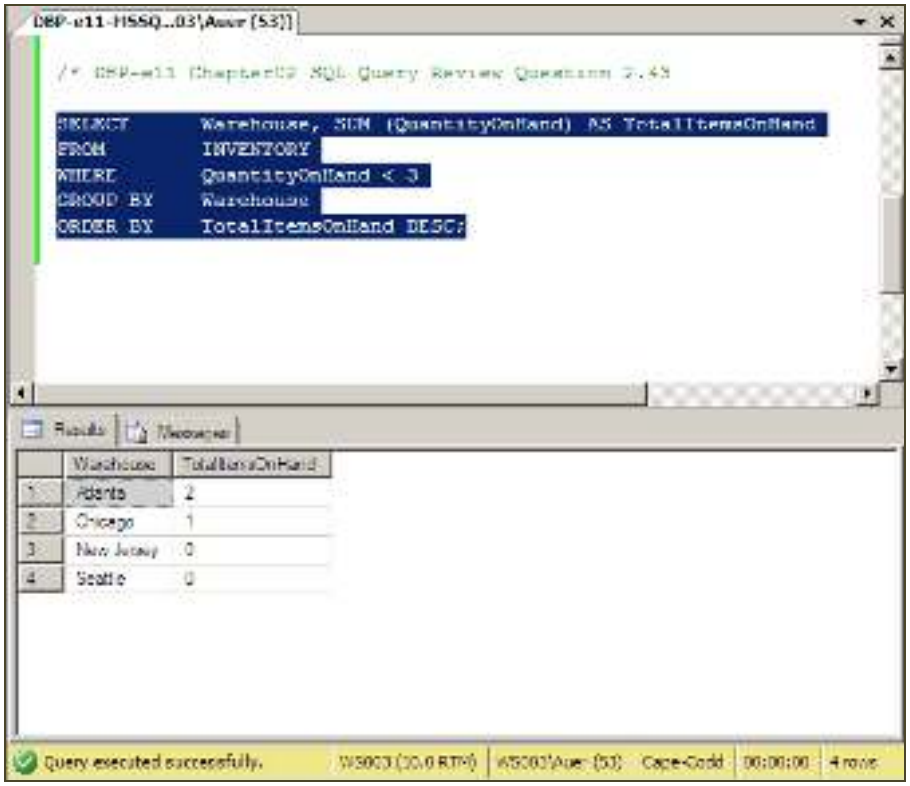
```
SELECT   Warehouse, SUM (QuantityOnHand) AS TotalItemsOnHand
FROM     INVENTORY
WHERE    QuantityOnHand < 3
GROUP BY Warehouse
```

```
ORDER BY TotalItemsOnHand DESC;
```

Unfortunately, Microsoft Access cannot process the ORDER BY clause because it contains an aliased computed result. The Microsoft Access result **without** the ORDER BY clause is:



The correct results, obtained from SQL Server 2008, are:



2.44 *Write an SQL statement to display the Warehouse and a count of QuantityOnHand, grouped by Warehouse. Omit all items that have a count greater than 2. Show only groups having fewer than 2 item counts. Name the count TotalItemsOnHand and display the results in descending order of TotalItemsOnHand.*

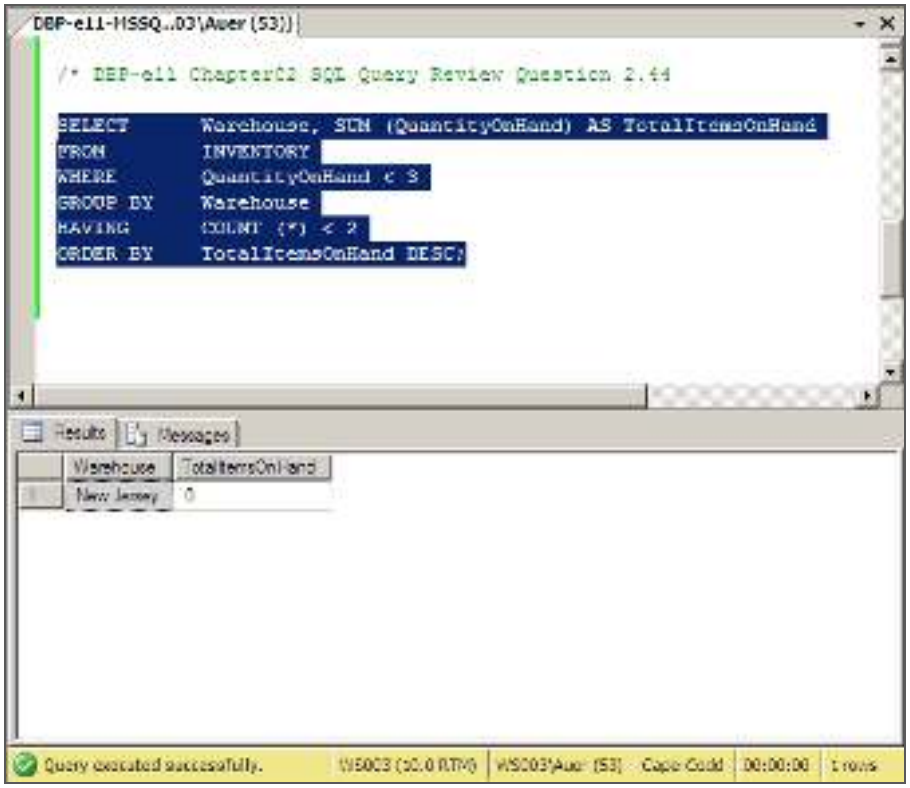
Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that "a count of" actually means the "sum" in this context, but that "fewer than 2 item counts" means "a number of records (rows or individual items) fewer than 2". The correct SQL Statement is:

```
SELECT   Warehouse, SUM (QuantityOnHand) AS TotalItemsOnHand
FROM     INVENTORY
WHERE    QuantityOnHand < 3
GROUP BY Warehouse
HAVING   COUNT (*) < 2
ORDER BY TotalItemsOnHand DESC;
```

Unfortunately, Microsoft Access cannot process the ORDER BY clause because it contains an aliased computed result. The Microsoft Access result **without** the ORDER BY clause is:

The correct results, obtained from SQL Server 2008, are:



2.45 In your answer to Review Question 2.44, was the WHERE or HAVING applied first? Why?

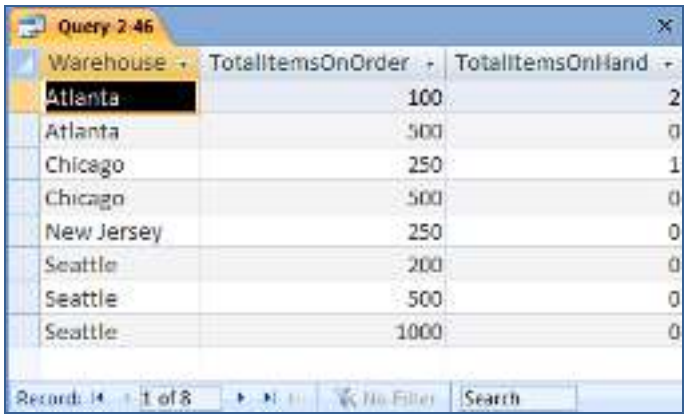
The WHERE clause is always applied before the HAVING clause. Otherwise there would be ambiguity in the SQL statement and the results would differ according to which clause was applied first.

2.46 Write an SQL statement to display the Warehouse, the sum of QuantityOnOrder and the sum of QuantityOnHand, grouped by Warehouse and QuantityOnOrder. Omit all items that have a count greater than 2. Name the count TotalItemsOnHand and display the results in descending order of TotalItemsOnHand.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that "a count of" actually means the "sum" in this context. The correct SQL Statement is:

```
SELECT   Warehouse, SUM (QuantityOnOrder) AS TotalItemsOnOrder, SUM
         (QuantityOnHand) AS TotalItemsOnHand
FROM     INVENTORY
WHERE    QuantityOnHand < 3
GROUP BY Warehouse, QuantityOnOrder;
```

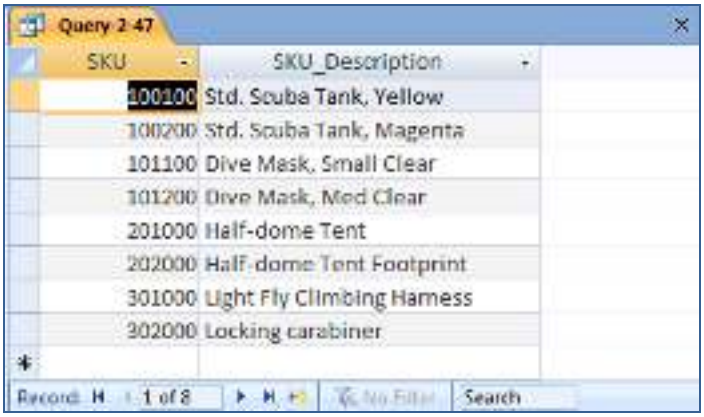


Use both the INVENTORY and WAREHOUSE table to answer Review Questions 2.47 through 2.53:

2.47 Write an SQL statement to show the SKU and SKU_Description of all items stored in a warehouse managed by "Smith". Use a subquery.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

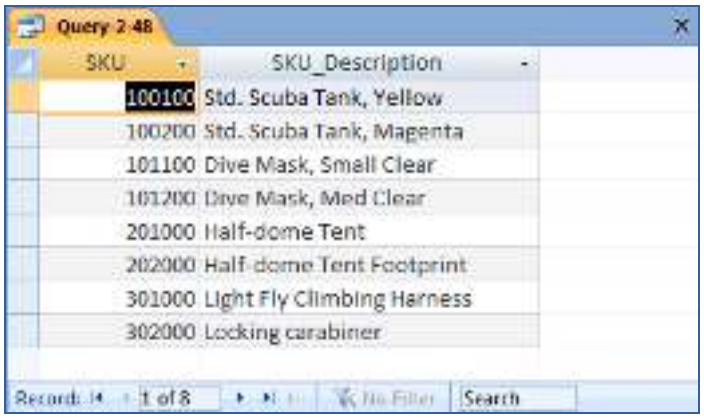
```
SELECT    SKU, SKU_Description
FROM      INVENTORY
WHERE     Warehouse IN
          (SELECT  Warehouse
          FROM     WAREHOUSE
          WHERE    Manager = 'Smith');
```



2.48 Write an SQL statement to show the SKU and SKU_Description of all items stored in a warehouse managed by "Smith". Use a join.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

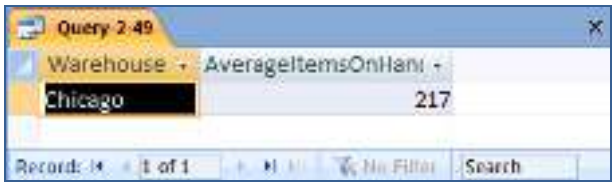
```
SELECT    SKU, SKU_Description
FROM      INVENTORY, WAREHOUSE
WHERE     INVENTORY.Warehouse = WAREHOUSE.Warehouse
   AND    Manager = 'Smith';

ALTERNATELY:

SELECT    INVENTORY.SKU, INVENTORY.SKU_Description
FROM      INVENTORY, WAREHOUSE
WHERE     INVENTORY.Warehouse = WAREHOUSE.Warehouse
   AND    WAREHOUSE.Manager = 'Smith';
```



2.49 Write an SQL statement to show the Warehouse and average QuantityOnHand of all items stored in a warehouse managed by "Smith". Use a subquery.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   Warehouse, AVG(QuantityOnHand) AS AverageItemsOnHand
FROM     INVENTORY
WHERE    Warehouse IN
         (SELECT  Warehouse
         FROM     WAREHOUSE
         WHERE    Manager = 'Smith')
GROUP BY Warehouse;
```

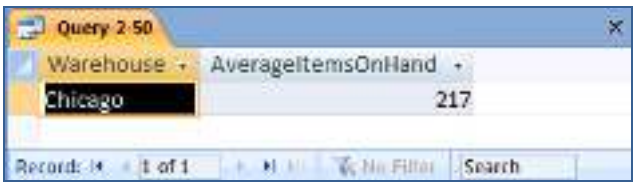


2.50 Write an SQL statement to show the Warehouse and average QuantityOnHand of all items stored in a warehouse managed by "Smith". Use a join.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   INVENTORY.Warehouse,
         AVG(QuantityOnHand) AS AverageItemsOnHand
FROM     INVENTORY, WAREHOUSE
WHERE    INVENTORY.Warehouse = WAREHOUSE.Warehouse
   AND   Manager = 'Smith'
GROUP BY INVENTORY.Warehouse;
```

Note the use of the complete references to **INVENTORY.Warehouse** – the query will NOT work without them.



2.51 Write an SQL statement to show the Warehouse, Manager, and QuantityOnHand of all items stored in a warehouse managed by "Smith". Use a join.

Solutions to Project Questions 2.16 – 2.53 are contained in the Microsoft Access database *DBPe11-IM-Ch02-Cape-Codd.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

There is some ambiguity in the question. If we want the QuantityOnHand for each individual item, we would use:

```
SELECT   INVENTORY.Warehouse, Manager, QuantityOnHand
FROM     INVENTORY, WAREHOUSE
WHERE    INVENTORY.Warehouse =WAREHOUSE.Warehouse
   AND   Manager = 'Smith';
```

Query 2-51 A

Warehouse	Manager	QuantityOnHand
Chicago	Smith	100
Chicago	Smith	75
Chicago	Smith	0
Chicago	Smith	50
Chicago	Smith	10
Chicago	Smith	1
Chicago	Smith	250
Chicago	Smith	1250

Record: 1 of 8 No Filter Search

We should add an additional column to identify each item in this query. On the other hand, if we want the total QuantityOnHand for the entire warehouse, we would use:

```
SELECT   INVENTORY.Warehouse, Manager,
         SUM (QuantityOnHand) AS TotalItemsOnHand
FROM     INVENTORY, WAREHOUSE
WHERE    INVENTORY.Warehouse =WAREHOUSE.Warehouse
   AND   Manager = 'Smith'
GROUP BY INVENTORY.Warehouse, WAREHOUSE.Manager;
```

Query 2-51 B

Warehouse	Manager	TotalItemsOnHand
Chicago	Smith	1736

Record: 1 of 1 No Filter Search

In each case, note the use of the complete references to **INVENTORY.Warehouse** – the query will NOT work without them.

2.52 Explain why you cannot use a subquery in your answer to question 2.51.

In a query that contains a subquery, only data from fields in the table used in the top-level query can be included in the SELECT statement. If data from fields from other tables is also needed, a join must be used. In question 2.51 we needed to display WAREHOUSE.Manager but INVENTORY would have been the table in the top-level query. Therefore, we had to use a join.

2.53 Explain how subqueries and joins differ.

(1) In a query that contains a subquery, only data from fields in the table used in the top-level query can be included in the SELECT statement. If data from fields from other tables are also needed, a join must be used. See the answer to question 2.46.

(2) The subqueries in this chapter are **non-correlated subqueries**, which have an equivalent join structure. In Chapter 8, **correlated subqueries** will be discussed, and correlated subqueries do not have an equivalent join structure – you must use subqueries.

ANSWERS TO PROJECT QUESTIONS

For this set of project questions, we will continue creating a Microsoft Access database for the Wedgewood Pacific Corporation (WPC). Founded in 1957 in Seattle, Washington, WPC has grown into an internationally recognized organization. The company is located in two buildings. One building houses the Administration, Accounting, Finance, and Human Resources departments, and the second houses the Production, Marketing, and Information Systems departments. The company database contains data about company employees, departments, company projects, company assets such as computer equipment, and other aspects of company operations. In the following project questions, we have already created the WPC.accdb database with the following two tables:

DEPARTMENT (DepartmentName, BudgetCode, OfficeNumber, Phone)

EMPLOYEE (EmployeeNumber, FirstName, LastName, Department, Phone, Email)

Now we will add in the following two tables:

PROJECT (ProjectID, Name, Department, MaxHours, StartDate, EndDate)

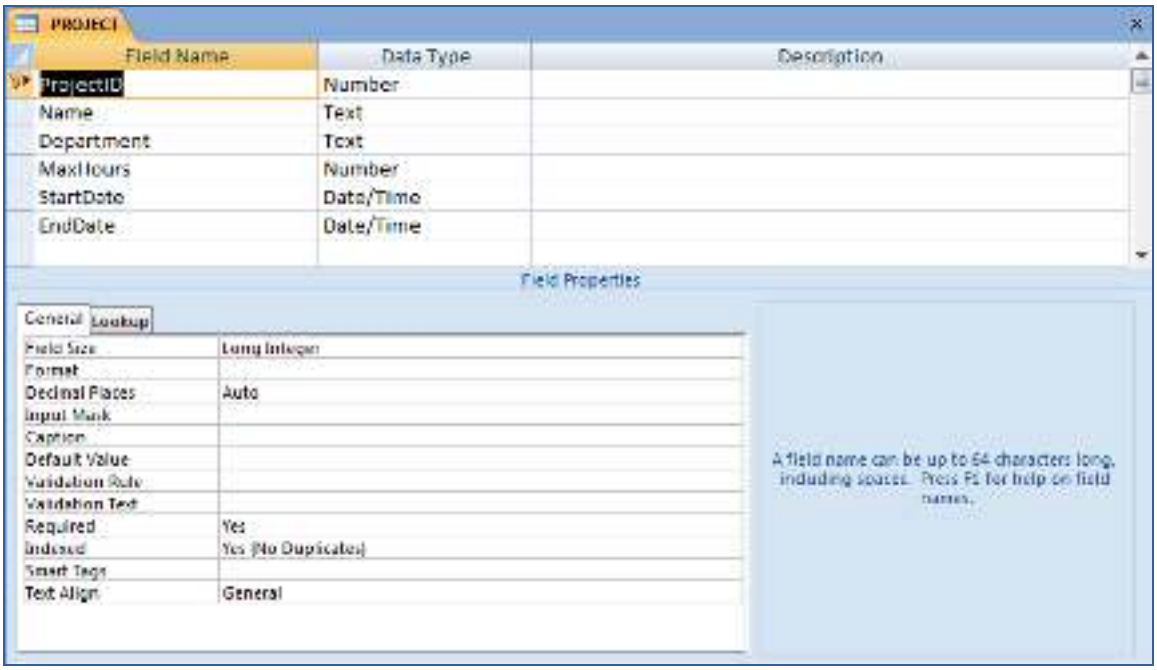
ASSIGNMENT (ProjectID, EmployeeNumber, HoursWorked)

2.54 *Figure 2-26 shows the column characteristics for the WPC PROJECT table. Using the column characteristics, create the PROJECT table in the WPC.accdb database.*

Solutions to Project Questions 2.54 – 2.62 are contained in the Microsoft Access database *DBPe11-IM-Ch02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

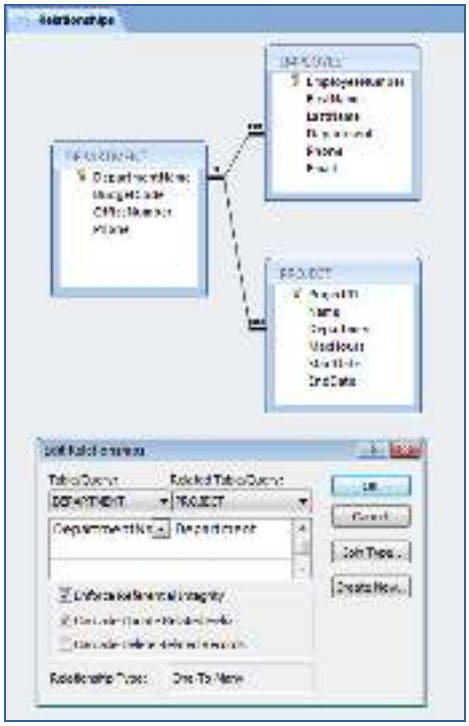
Column Name	Type	Key	Required	Remarks
ProjectID	Number	Primary Key	Yes	Long Integer
Name	Text (50)	No	Yes	
Department	Text (35)	Foreign Key	Yes	
MaxHours	Number	No	Yes	Double
StartDate	Date/Time	No	No	
EndDate	Date/Time	No	No	

Figure 2-26 - Column Characteristics for the PROJECT Table



2.55 Create the relationship and referential integrity constraint between PROJECT and DEPARTMENT. Enable enforcing of referential integrity and cascading of data updates, but do not enable cascading of data from deleted records.

Solutions to Project Questions 2.54 – 2.62 are contained in the Microsoft Access database *DBPe11-IM-Ch02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

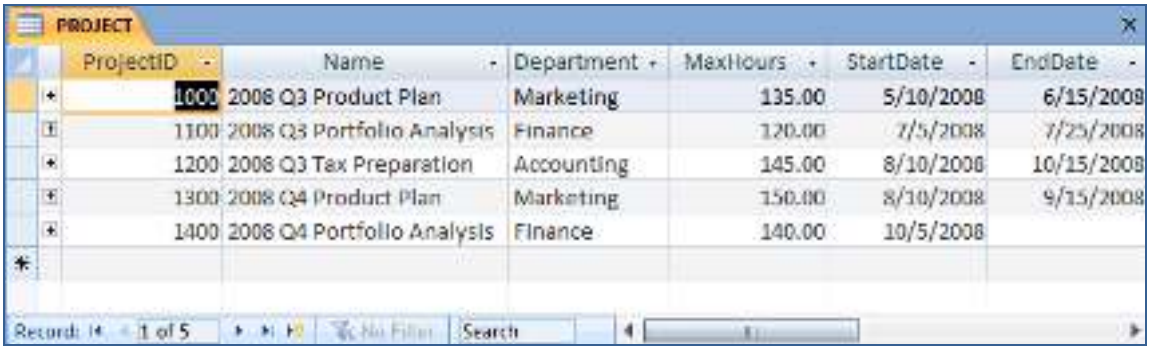


2.56 Figure 2-27 shows the data for the WPC PROJECT table. Using the Datasheet view, enter the data shown in Figure 2-27 into your PROJECT table.

Solutions to Project Questions 2.54 – 2.62 are contained in the Microsoft Access database *DBPe11-IM-Ch02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

ProjectID	Name	Department	MaxHours	StartDate	EndDate
1000	2008 Q3 Product Plan	Marketing	135	05/10/08	06/15/08
1100	2008 Q3 Portfolio Analysis	Finance	120	07/05/08	07/25/08
1200	2008 Q3 Tax Preparation	Accounting	145	08/10/08	10/25/08
1300	2008 Q4 Product Plan	Marketing	150	08/10/08	09/15/08
1400	2008 Q4 Portfolio Analysis	Finance	140	10/05/08	

Figure 2-27 - Sample Data for the PROJECT Table

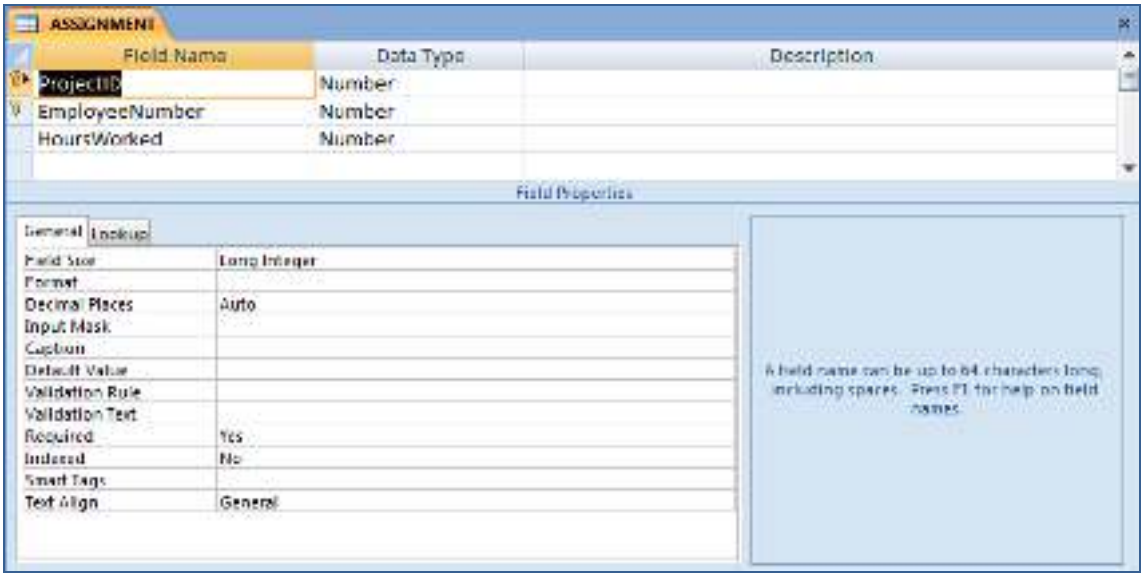


2.57 Figure 2-28 shows the column characteristics for the WPC ASSIGNMENT table. Using the column characteristics, create the ASSIGNMENT table in the WPC.accdb database.

Solutions to Project Questions 2.54 – 2.62 are contained in the Microsoft Access database *DBPe11-IM-Ch02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

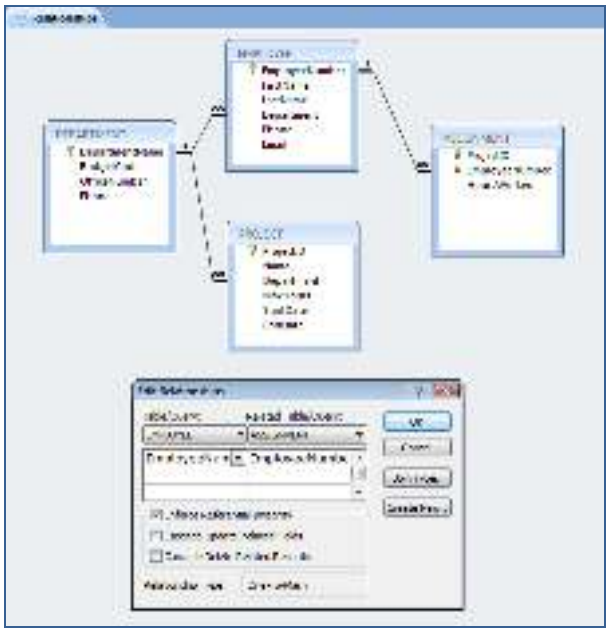
Column Name	Type	Key	Required	Remarks
ProjectID	Number	Primary Key, Foreign Key	Yes	Long Integer
EmployeeNumber	Number	Primary Key, Foreign Key	Yes	Long Integer
HoursWorked	Number	No	No	Double

Figure 2-28 - Column Characteristics for the ASSIGNMENT Table

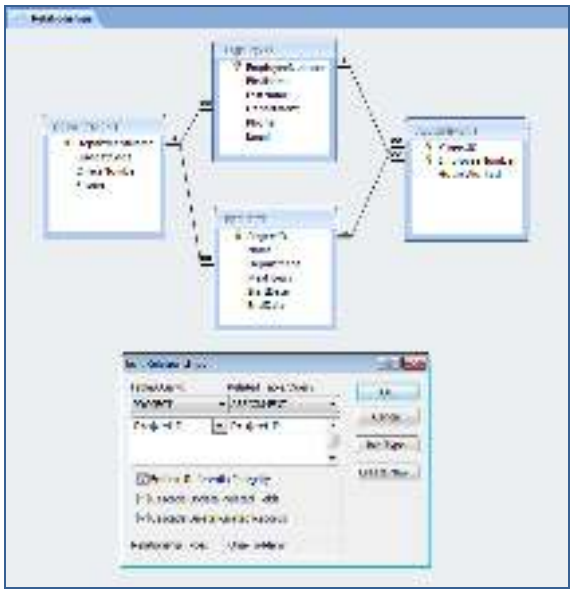


2.58 Create the relationship and referential integrity constraint between ASSIGNMENT and EMPLOYEE. Enable enforcing of referential integrity, but do not enable either cascading updates or the cascading of data from deleted records.

Solutions to Project Questions 2.54 – 2.62 are contained in the Microsoft Access database *DBPe11-IM-Ch02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).



2.59 Create the relationship and referential integrity constraint between ASSIGNMENT and PROJECT. Enable enforcing of referential integrity and cascading of deletes, but do not enable cascading updates.



2.60 Figure 2-29 shows the data for the WPC ASSIGNMENT table. Using the Datasheet view, enter the data shown in Figure 2-29 into your ASSIGNMENT table.

Solutions to Project Questions 2.54 – 2.62 are contained in the Microsoft Access database *DBPe11-IM-Ch02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

ProjectID	EmployeeNumber	HoursWorked
1000	1	30.0
1000	8	75.0
1000	10	55.0
1100	4	40.0
1100	6	45.0
1100	1	25.0
1200	2	20.0
1200	4	45.0
1200	5	40.0
1300	1	35.0
1300	8	80.0
1300	10	50.0
1400	4	15.0
1400	5	10.0
1400	6	27.5

Figure 2-29 - Sample Data for the PROJECT Table

ASSIGNMENT

ProjectID	EmployeeNumber	HoursWorked
1000	1	30.00
1000	8	75.00
1000	10	55.00
1100	4	40.00
1100	6	45.00
1100	1	25.00
1200	2	20.00
1200	4	45.00
1200	5	40.00
1300	1	35.00
1300	8	80.00
1300	10	50.00
1400	4	15.00
1400	5	10.00
1400	6	27.50

Record: 1 of 15 No Filter Search

2.61 Using Access SQL, create and run queries to answer the following questions. Save each query using the query name format SQL-Query-02-##, where the ## sign is replaced by the letter designator of the question. For example, the first query will be saved as SQL-Query-02-A.Write SQL queries to produce the following results:

Solutions to Project Questions 2.54 – 2.62 are contained in the Microsoft Access database *DBPe11-IM-Ch02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

A. What projects are in the PROJECT table? Show all information for each project.

```
/*****   Question A - SQL-Query-02-A   ************************/

SELECT * FROM PROJECT;
```

SQL-Query-02-A

ProjectID	Name	Department	MaxHours	StartDate	EndDate
1000	2008 Q3 Product Plan	Marketing	135.00	5/10/2008	6/15/2008
1100	2008 Q3 Portfolio Analysis	Finance	120.00	7/5/2008	7/25/2008
1200	2008 Q3 Tax Preparation	Accounting	145.00	8/10/2008	10/15/2008
1300	2008 Q4 Product Plan	Marketing	150.00	8/10/2008	9/15/2008
1400	2008 Q4 Portfolio Analysis	Finance	140.00	10/5/2008	

Record: 1 of 5 No Filter Search

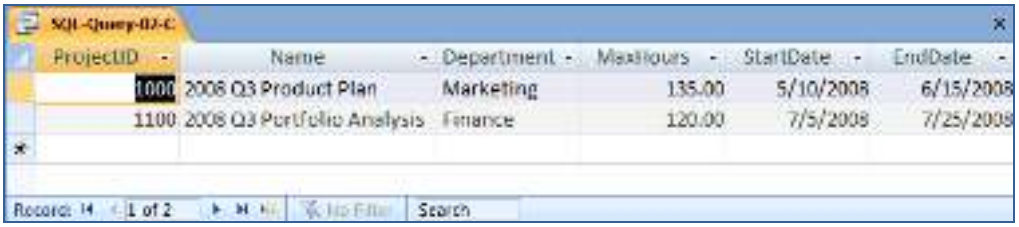
B. *What are the ProjectID, Name, StartDate, and EndDate values of projects in the PROJECT table?*

```
/*****    Question B - SQL-Query-02-B    **********************/

SELECT     ProjectID, Name, StartDate, EndDate
FROM       PROJECT;
```

ProjectID	Name	StartDate	EndDate
1000	2008 Q3 Product Plan	5/10/2008	6/15/2008
1100	2008 Q3 Portfolio Analysis	7/5/2008	7/25/2008
1200	2008 Q3 Tax Preparation	8/10/2008	10/15/2008
1300	2008 Q4 Product Plan	8/10/2008	9/15/2008
1400	2008 Q4 Portfolio Analysis	10/5/2008	

C. *What projects in the PROJECT table started before August 1, 2008? Show all the information for each project.*

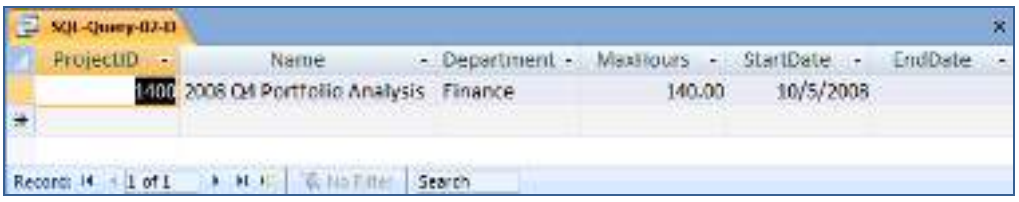
```
/*****    Question C - SQL-Query-02-C    **********************/

SELECT     *
FROM       PROJECT
WHERE      StartDate < #01-AUG-08#;
```



ProjectID	Name	Department	MaxHours	StartDate	EndDate
1000	2008 Q3 Product Plan	Marketing	135.00	5/10/2008	6/15/2008
1100	2008 Q3 Portfolio Analysis	Finance	120.00	7/5/2008	7/25/2008

D. *What projects in the PROJECT table have not been completed? Show all the information for each project.*

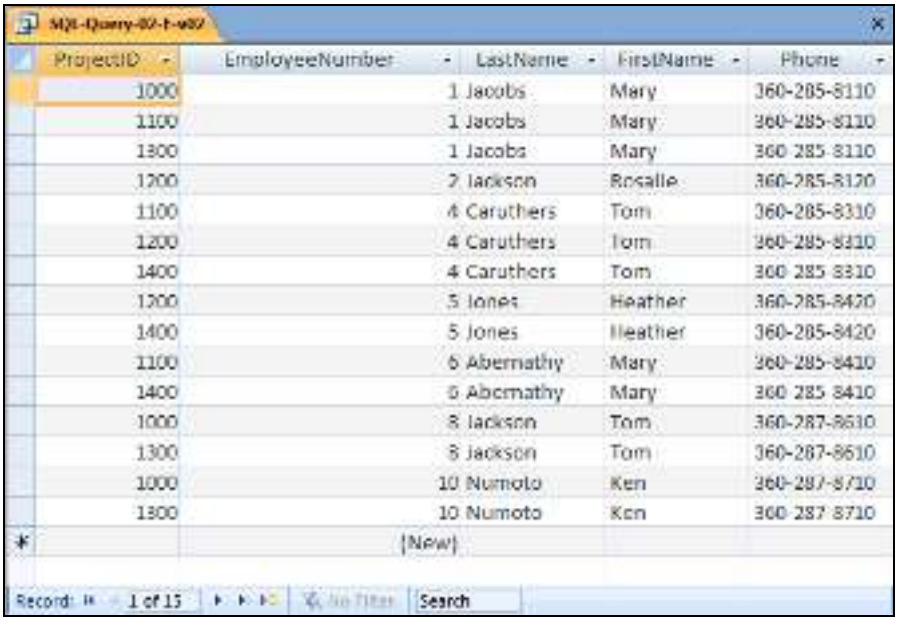
```
/*****    Question D - SQL-Query-02-D    **********************/

SELECT     *
FROM       PROJECT
WHERE      EndDate IS NULL;
```



ProjectID	Name	Department	MaxHours	StartDate	EndDate
1400	2008 Q4 Portfolio Analysis	Finance	140.00	10/5/2008	

E. Who are the employees assigned to each project? Show ProjectID, Employee-Number, LastName, FirstName, and Phone.

```
/*****   Question E - SQL-Query-02-E   ***********************/

SELECT    ProjectID, E.EmployeeNumber, LastName, FirstName, Phone
FROM      ASSIGNMENT AS A INNER JOIN EMPLOYEE AS E
          ON A.EmployeeNumber=E.EmployeeNumber;
```

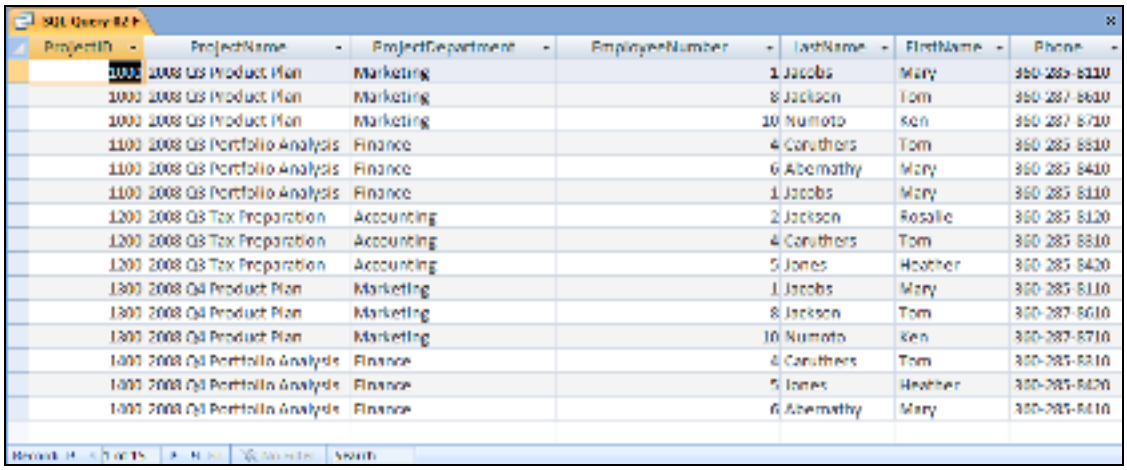


ProjectID	EmployeeNumber	LastName	FirstName	Phone
1000	1	Jacobs	Mary	360-285-8110
1100	1	Jacobs	Mary	360-285-8110
1300	1	Jacobs	Mary	360 285 8110
1200	2	Jackson	Rosalie	360-285-8120
1100	4	Caruthers	Tom	360-285-8310
1200	4	Caruthers	Tom	360-285-8310
1400	4	Caruthers	Tom	360 285 8310
1200	5	Jones	Heather	360-285-8420
1400	5	Jones	Heather	360-285-8420
1100	6	Abernathy	Mary	360-285-8410
1400	6	Abernathy	Mary	360 285 8410
1000	8	Jackson	Tom	360-287-8610
1300	8	Jackson	Tom	360-287-8610
1000	10	Numoto	Ken	360-287-8710
1300	10	Numoto	Ken	360 287 8710
*	(New)			

Record: 1 of 15 Search

F. Who are the employees assigned to each project? Show the ProjectID, Name, and Department. Show EmployeeNumber, LastName, FirstName, and Phone.

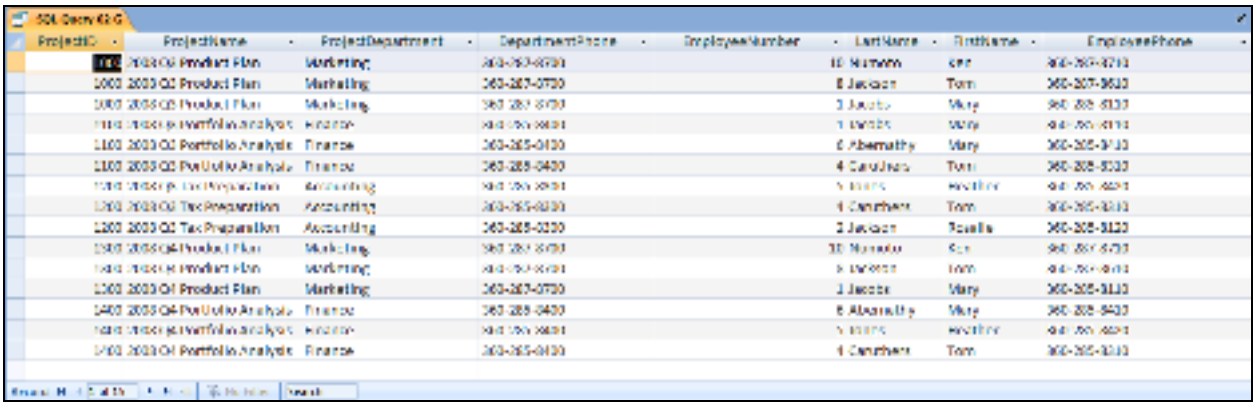
```
/*****   Question F - SQL-Query-02-F   ***********************/

SELECT    P.ProjectID, Name, P.Department,
          E.EmployeeNumber, LastName, FirstName, Phone
FROM      (ASSIGNMENT AS A INNER JOIN EMPLOYEE AS E
             ON A.EmployeeNumber=E.EmployeeNumber)
          INNER JOIN PROJECT AS P
             ON A.ProjectID=P.ProjectID;
```



ProjectID	ProjectName	ProjectDepartment	EmployeeNumber	LastName	FirstName	Phone
1000	2008 Q3 Product Plan	Marketing	1	Jacobs	Mary	360-285-8110
1000	2008 Q3 Product Plan	Marketing	8	Jackson	Tom	360-287-8610
1000	2008 Q3 Product Plan	Marketing	10	Numoto	Ken	360-287-8710
1100	2008 Q3 Portfolio Analysis	Finance	4	Caruthers	Tom	360-285-8310
1100	2008 Q3 Portfolio Analysis	Finance	6	Abernathy	Mary	360-285-8410
1100	2008 Q3 Portfolio Analysis	Finance	1	Jacobs	Mary	360-285-8110
1200	2008 Q3 Tax Preparation	Accounting	2	Jackson	Rosalie	360-285-8120
1200	2008 Q3 Tax Preparation	Accounting	4	Caruthers	Tom	360-285-8310
1200	2008 Q3 Tax Preparation	Accounting	5	Jones	Heather	360-285-8420
1300	2008 Q4 Product Plan	Marketing	1	Jacobs	Mary	360-285-8110
1300	2008 Q4 Product Plan	Marketing	8	Jackson	Tom	360-287-8610
1300	2008 Q4 Product Plan	Marketing	10	Numoto	Ken	360-287-8710
1400	2008 Q4 Portfolio Analysis	Finance	4	Caruthers	Tom	360-285-8310
1400	2008 Q4 Portfolio Analysis	Finance	5	Jones	Heather	360-285-8420
1400	2008 Q4 Portfolio Analysis	Finance	6	Abernathy	Mary	360-285-8410

G. Who are the employees assigned to each project? Show ProjectID, Name, Department, and Department Phone. Show EmployeeNumber, LastName, FirstName, and Employee Phone. Sort by ProjectID in ascending order.

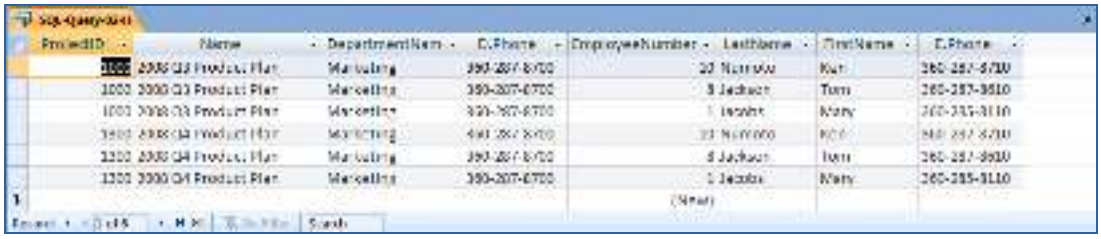
```
/*****    Question G - SQL-Query-02-G    ************************/

SELECT     P.ProjectID, Name, D.DepartmentName, D.Phone,
           E.EmployeeNumber, LastName, FirstName, E.Phone
FROM       ((ASSIGNMENT AS A INNER JOIN EMPLOYEE AS E
              ON A.EmployeeNumber=E.EmployeeNumber)
           INNER JOIN PROJECT AS P
              ON A.ProjectID=P.ProjectID)
           INNER JOIN DEPARTMENT AS D
              ON P.Department=D.DepartmentName
ORDER BY   P.ProjectID;
```



H. Who are the employees assigned to projects run by the marketing department? Show ProjectID, Name, Department, and Department Phone. Show EmployeeNumber, LastName, FirstName, and Employee Phone. Sort by ProjectID in ascending order.

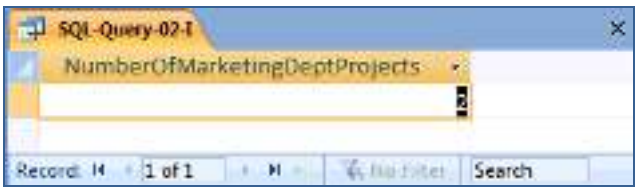
```
/*****    Question H - SQL-Query-02-H    ************************/

SELECT     P.ProjectID, Name, D.DepartmentName, D.Phone,
           E.EmployeeNumber, LastName, FirstName, E.Phone
FROM       ((ASSIGNMENT AS A INNER JOIN EMPLOYEE AS E
              ON A.EmployeeNumber=E.EmployeeNumber)
```

```
            INNER JOIN PROJECT AS P
               ON A.ProjectID=P.ProjectID)
            INNER JOIN DEPARTMENT AS D
               ON P.Department=D.DepartmentName
WHERE       DepartmentName='Marketing'
ORDER BY    P.ProjectID;
```



I. How many projects are being run by the marketing department? Be sure to assign an appropriate column name to the computed results.

```
/*****   Question I - SQL-Query-02-I   ************************/

SELECT     COUNT(*) AS NumberOfMarketingDeptProjects
FROM       PROJECT
WHERE      Department='Marketing';
```

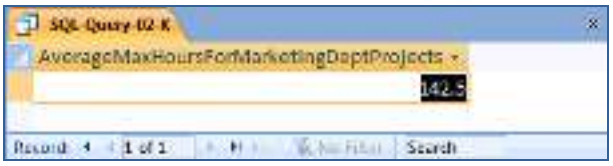


J. What is the total MaxHours of projects being run by the marketing department? Be sure to assign an appropriate column name to the computed results.

```
/*****   Question J - SQL-Query-02-J   ************************/

SELECT     SUM(MaxHours) AS TotalMaxHoursForMarketingDeptProjects
FROM       PROJECT
WHERE      Department='Marketing';
```

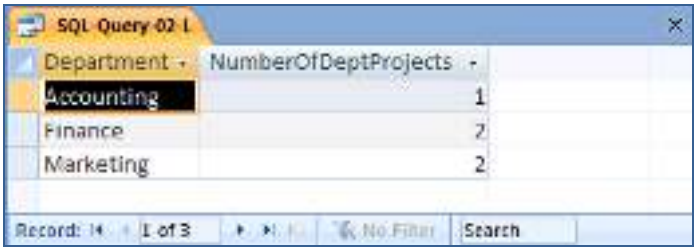
K. What is the average MaxHours of projects being run by the marketing department? Be sure to assign an appropriate column name to the computed results.

```
/*****   Question K - SQL-Query-02-K   ************************/

SELECT     AVG(MaxHours) AS AverageMaxHoursForMarketingDeptProjects
FROM       PROJECT
WHERE      Department='Marketing';
```



L. How many projects are being run by each department? Be sure to display each DepartmentName and to assign an appropriate column name to the computed results.

```
/*****   Question L - SQL-Query-02-L   ***********************/

SELECT     Department, COUNT(*) AS NumberOfDeptProjects
FROM       PROJECT
GROUP BY   Department;
```

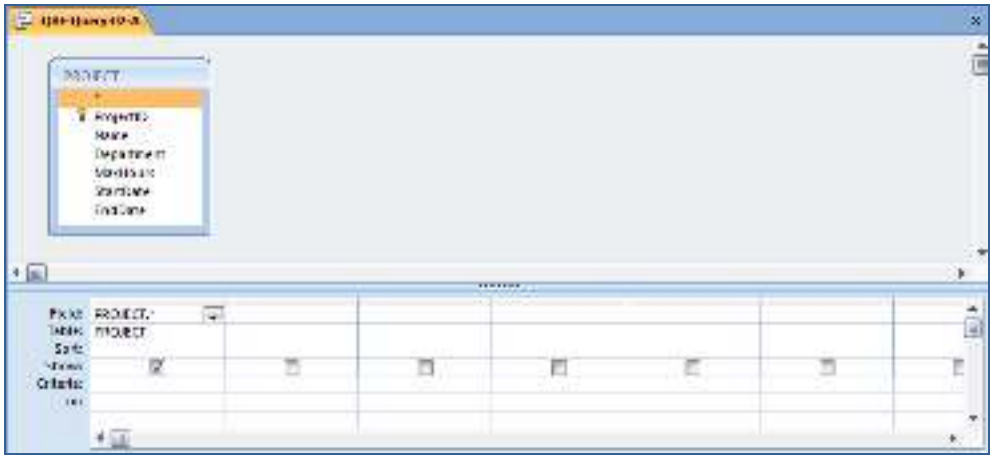


2.62 Using Access QBE, create and run new queries to answer the questions in exercise 2.61. Save each query using the query name format QBE-Query-02-##, where the ## sign is replaced by the letter designator of the question. For example, the first query will be saved as QBE-Query-02-A.

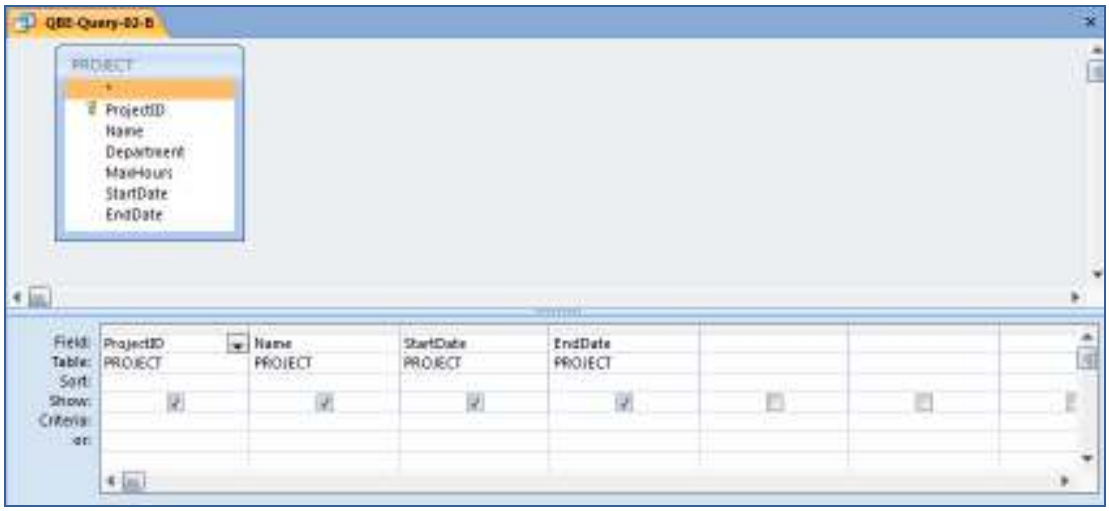
Solutions to Project Questions 2.54 – 2.62 are contained in the Microsoft Access database *DBPe11-IM-Ch02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

The results of each query will be identical to the corresponding SQL query in the previous Project Question. Here we will show the QBE design of the query.

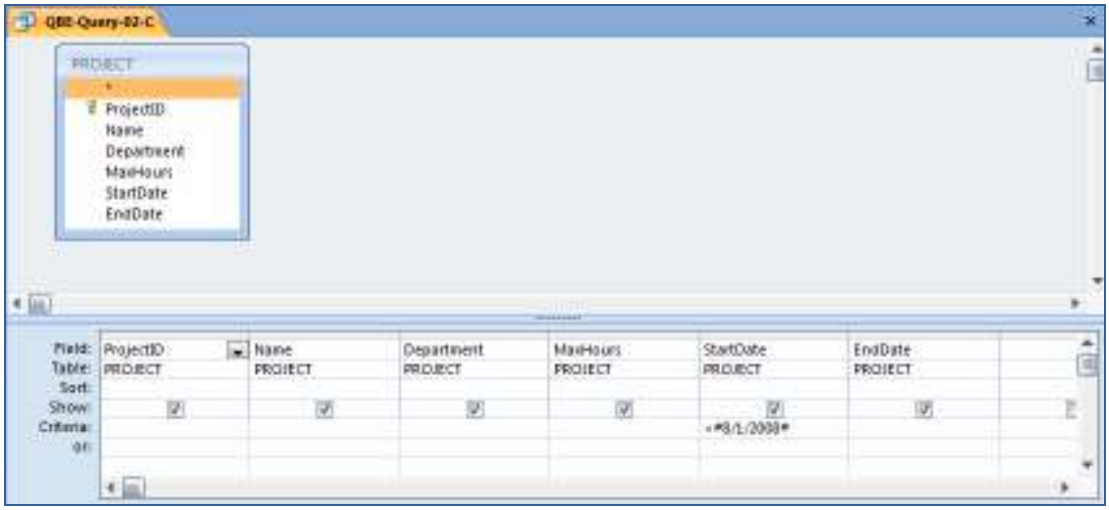
A. What projects are in the PROJECT table? Show all information for each project.



B. *What are the ProjectID, Name, StartDate, and EndDate values of projects in the PROJECT table?*

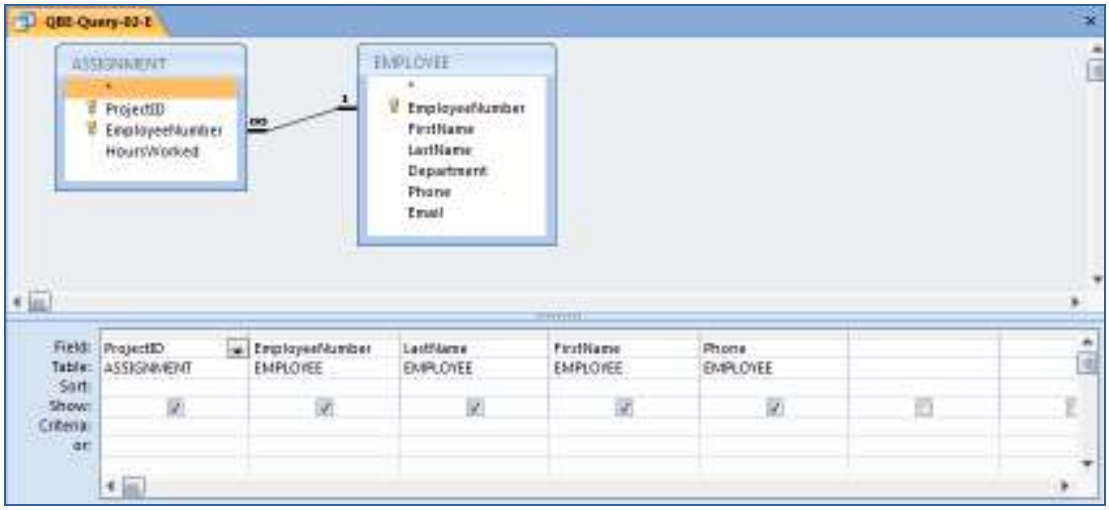


C. *What projects in the PROJECT table started before August 1, 2008? Show all the information for each project.*

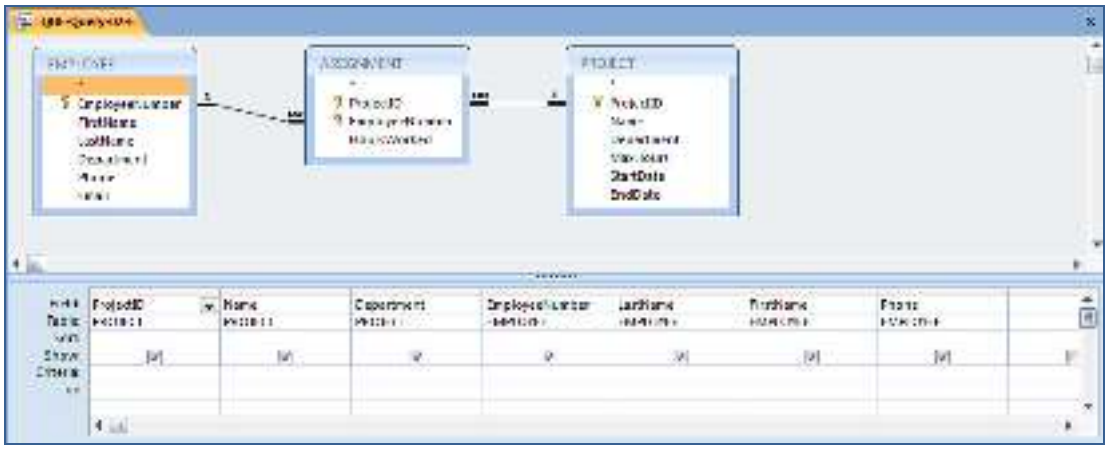


D. *What projects in the PROJECT table have not been completed? Show all the information for each project.*

E. *Who are the employees assigned to each project? Show ProjectID, Employee-Number, LastName, FirstName, and Phone.*

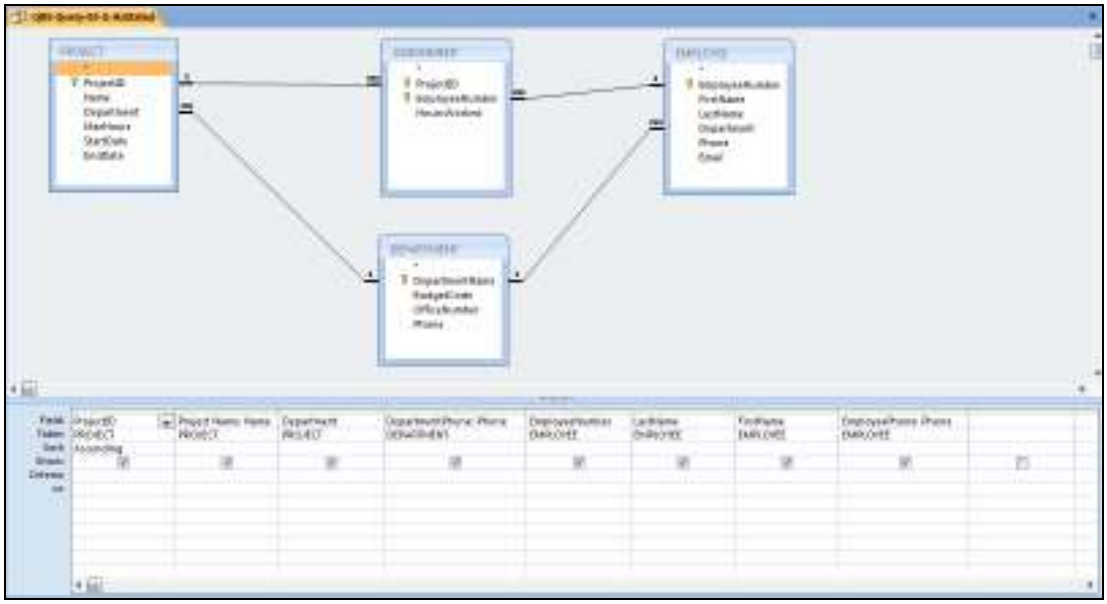


F. *Who are the employees assigned to each project? Show the ProjectID, Name, and Department. Show EmployeeNumber, LastName, FirstName, and Phone.*

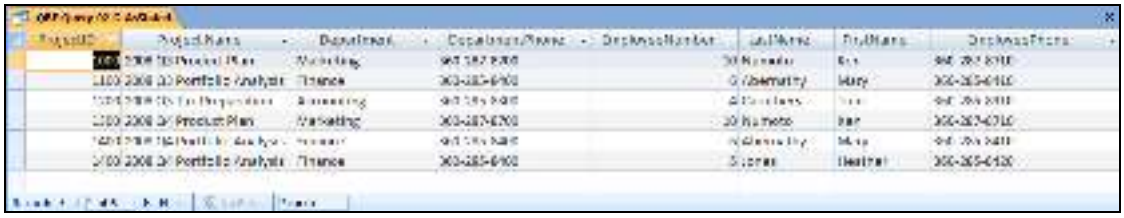


G. *Who are the employees assigned to each project? Show ProjectID, Name, Department, and Department Phone. Show EmployeeNumber, LastName, FirstName, and Employee Phone. Sort by ProjectID in ascending order.*

The QBE query shows the solution to the question as stated, but it will not run correctly due to how Access interprets JOIN...ON commands.

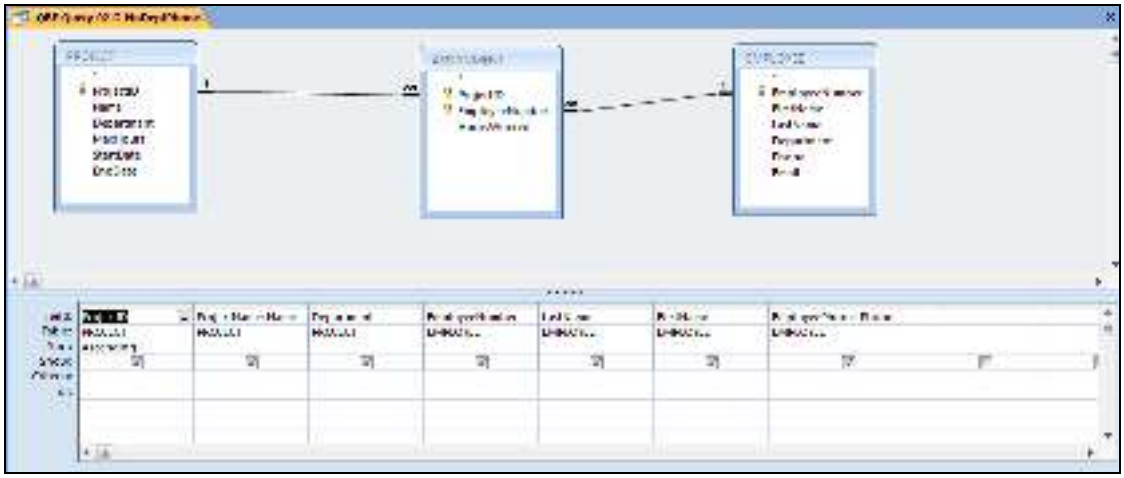


The results are:



Compare these results with those shown for 2.61 G above, and you will see the difference.

There are two work arounds. First, create the query *without* Department Phone. This is the only column needed from the DEPARTMENT table, which can thus be eliminated from the query. The QBE query is:

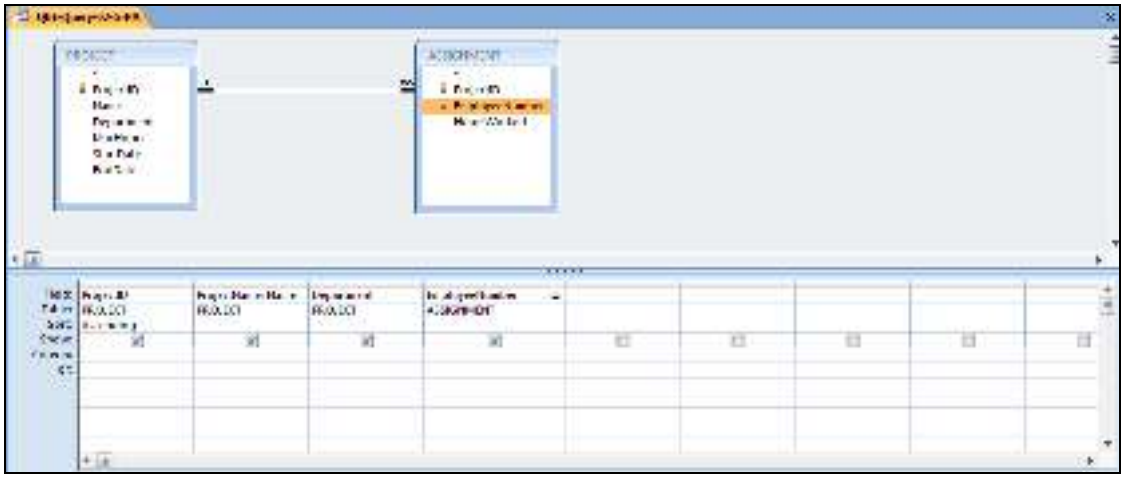


The results will be correct, but without the DepartmentPhone column.

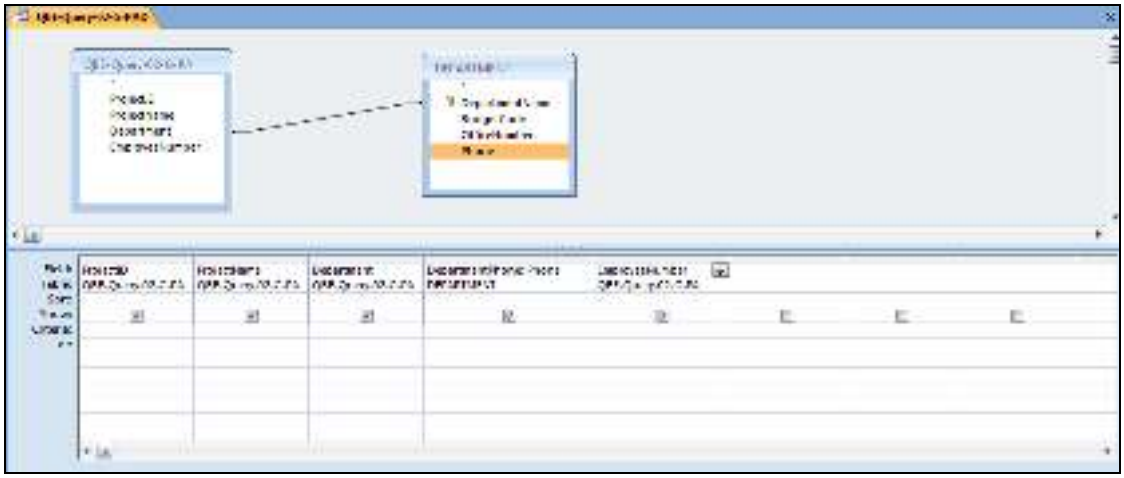
Alternatively, as devised by Professor John Schauf of Edgewood College, Madison, WI, you can illustrate building a set of queries, where each one uses the previous query and adds one additional table. This is possible because Access allows saved queries to be used as the equivalent of a table in a query. By adding in one table at a time, you can control the JOIN...ON statement sequence, and obtain the correct answer.

The steps below show how to create the needed sequence of QBE queries:

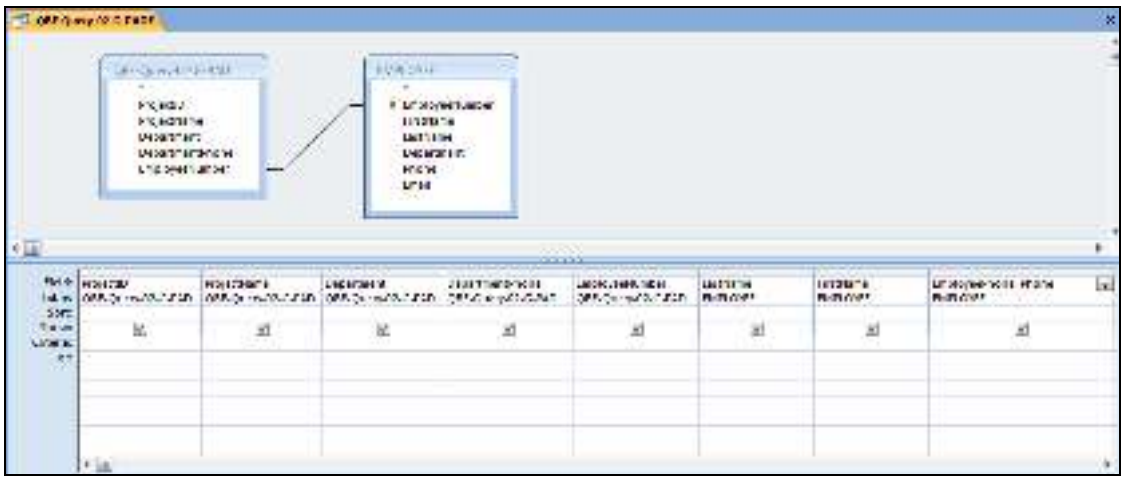
(1) Create a query that joins PROJECT and ASSIGMENT, and name it QBE-Query-02-G-PA. Note that you must include ASSIGNMENT.EmployeeNumber in this query:



(2) Create a query that joins QBE-Query-02-G-PA and DEPARTMENT, and name it QBE-Query-02-G-PAD - Note that you will have to manually link the DEPARTMENT primary key to the foreign key in QBE-Query-02-G-PA:



(3) Create a query that joins QBE-Query-02-G-PAD and EMPLOYEE, and name it QBE-Query-02-G-PADE - Note that you will have to manually link the DEPARTMENT primary key to the foreign key in QBE-Query-02-G-PAD:



The query results are now correct:

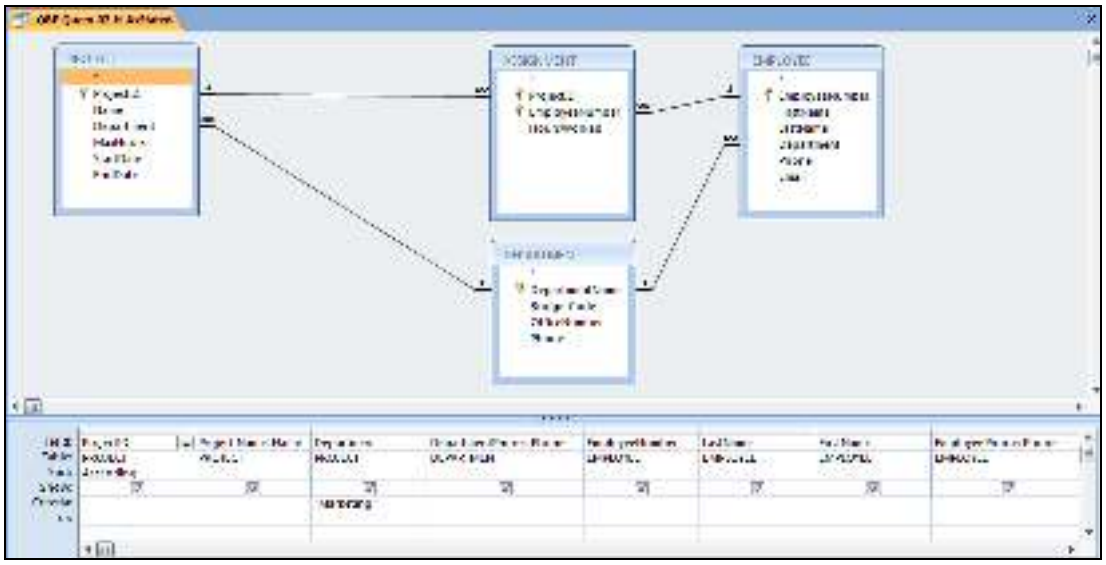
QBE-Query-02-G-PADE

ProjectID	ProjectName	Department	Department	EmployeeNumber	LastName	FirstName	EmployeePhone
1000	2008 Q3 Product Plan	Marketing	360-287-8700	1	Jacobs	Mary	360-285-8110
1000	2008 Q3 Product Plan	Marketing	360-287-8700	8	Jackson	Tom	360-287-8610
1000	2008 Q3 Product Plan	Marketing	360-287-8700	10	Numoto	Ken	360-287-8710
1100	2008 Q3 Portfolio Analysis	Finance	360-285-8400	4	Caruthers	Tom	360-285-8310
1100	2008 Q3 Portfolio Analysis	Finance	360-285-8400	6	Abernathy	Mary	360-285-8410
1100	2008 Q3 Portfolio Analysis	Finance	360-285-8400	1	Jacobs	Mary	360-285-8110
1200	2008 Q3 Tax Preparation	Accounting	360-285-8300	2	Jackson	Rosalie	360-285-8120
1200	2008 Q3 Tax Preparation	Accounting	360-285-8300	4	Caruthers	Tom	360-285-8310
1200	2008 Q3 Tax Preparation	Accounting	360-285-8300	5	Jones	Heather	360-285-8420
1300	2008 Q4 Product Plan	Marketing	360-287-8700	1	Jacobs	Mary	360-285-8110
1300	2008 Q4 Product Plan	Marketing	360-287-8700	8	Jackson	Tom	360-287-8610
1300	2008 Q4 Product Plan	Marketing	360-287-8700	10	Numoto	Ken	360-287-8710
1400	2008 Q4 Portfolio Analysis	Finance	360-285-8400	4	Caruthers	Tom	360-285-8310
1400	2008 Q4 Portfolio Analysis	Finance	360-285-8400	5	Jones	Heather	360-285-8420
1400	2008 Q4 Portfolio Analysis	Finance	360-285-8400	6	Abernathy	Mary	360-285-8410

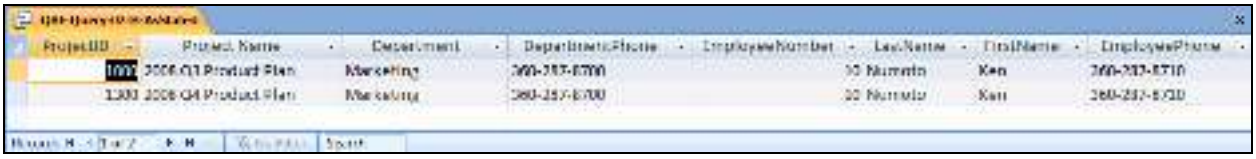
Record: 1 of 15 Search

H. Who are the employees assigned to projects run by the marketing department? Show ProjectID, Name, Department, and Department Phone. Show EmployeeNumber, LastName, FirstName, and Employee Phone. Sort by ProjectID in ascending order.

This question is identical to question G except for the restriction to marketing department projects. The QBE query shows the solution to the question as stated, but it will not run correctly due to how Access interprets JOIN...ON commands.



The results are:



Compare these results with those shown for Project Question 2.61 H above, and you will see the difference.

The problem we are encountering here is the same as described above in 2.26 G. Again, there are two work arounds. First, create the query *without* Department Phone. This is the only column needed from the DEPARTMENT table, which can thus be eliminated from the query.

The results will be correct, but without the DepartmentPhone column.

Alternatively, as devised by Professor John Schauf of Edgewood College, Madison, WI, you can illustrate building a set of queries, where each one uses the previous query and adds one additional table. This is possible because Access allows saved queries to be used as the equivalent of a table in a query. By adding in one table at a time, you can control the JOIN...ON statement sequence, and obtain the correct answer.

The steps below show how to create the needed sequence of QBE queries, which is indentical to the sequence used for Project Question 2.62 G above - see the screen shots for that solution:

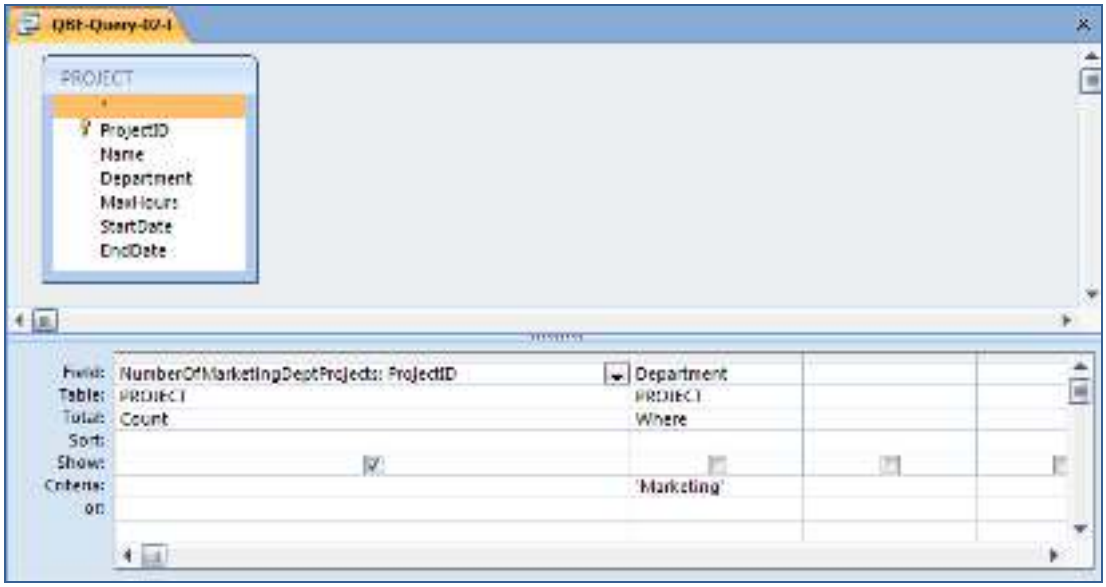
(1) Create a query that joins PROJECT and ASSIGMENT, and name it QBE-Query-0H-G-PA. Note that you must include ASSIGNMENT.EmployeeNumber in this query:

(2) Create a query that joins QBE-Query-02-H-PA and DEPARTMENT, and name it QBE-Query-02-H-PAD - Note that you will have to manually link the DEPARTMENT primary key to the foreign key in QBE-Query-02-H-PA:

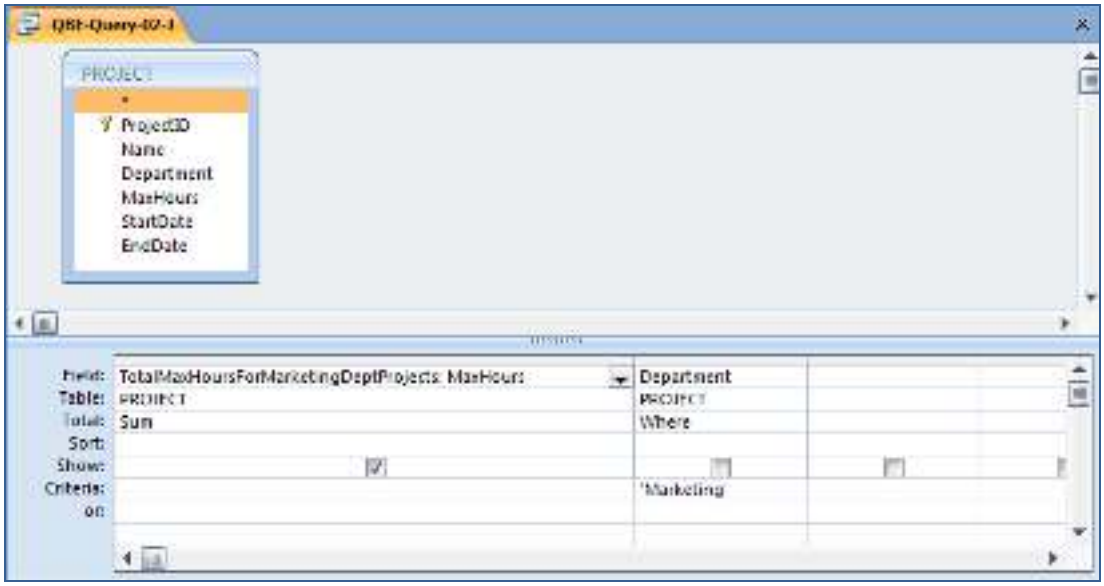
(3) Create a query that joins QBE-Query-02-H-PAD and EMPLOYEE, and name it QBE-Query-02-H-PADE - Note that you will have to manually link the DEPARTMENT primary key to the foreign key in QBE-Query-02-H-PAD:

The query results are now correct, and may be found in DBPe11-IM-Ch02-WPC.accdb file.

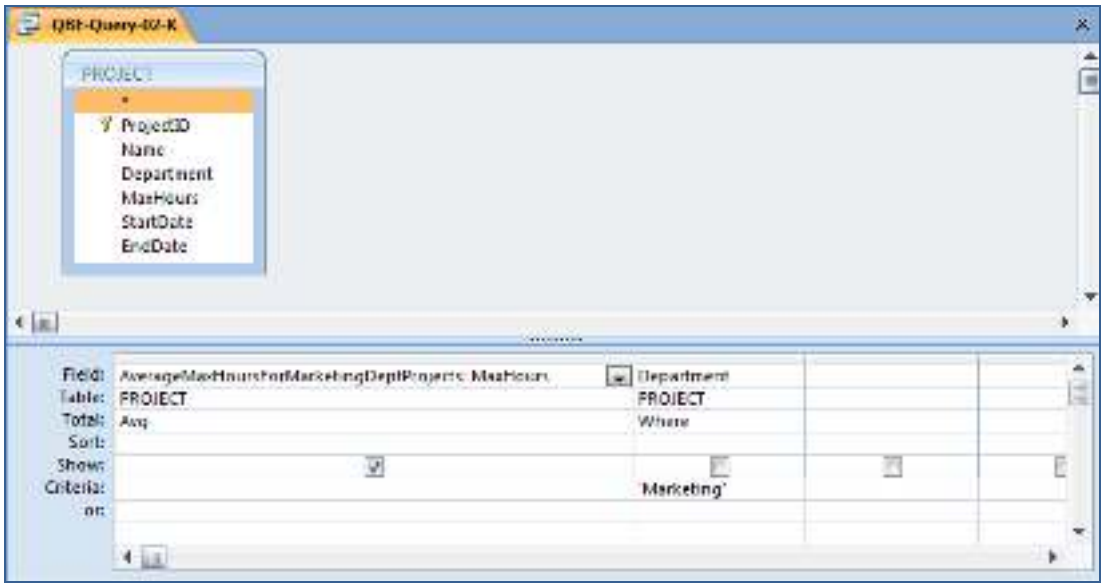
I. *How many projects are being run by the marketing department? Be sure to assign an appropriate column name to the computed results.*



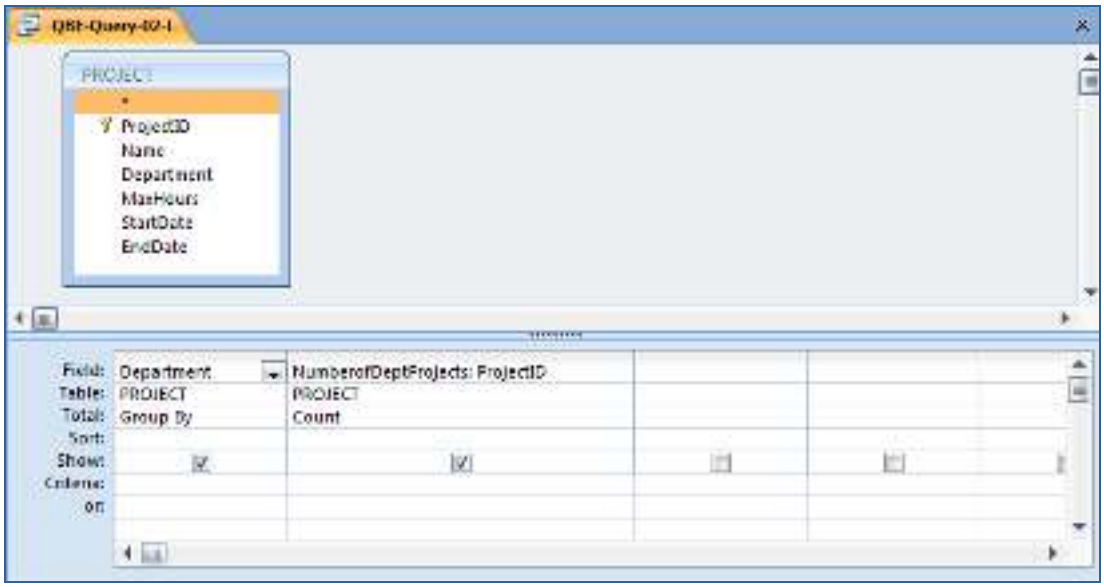
J. *What is the total MaxHours of projects being run by the marketing department? Be sure to assign an appropriate column name to the computed results.*



K. *What is the average MaxHours of projects being run by the marketing department? Be sure to assign an appropriate column name to the computed results.*



L. *How many projects are being run by each department? Be sure to display each DepartmentName and to assign an appropriate column name to the computed results.*



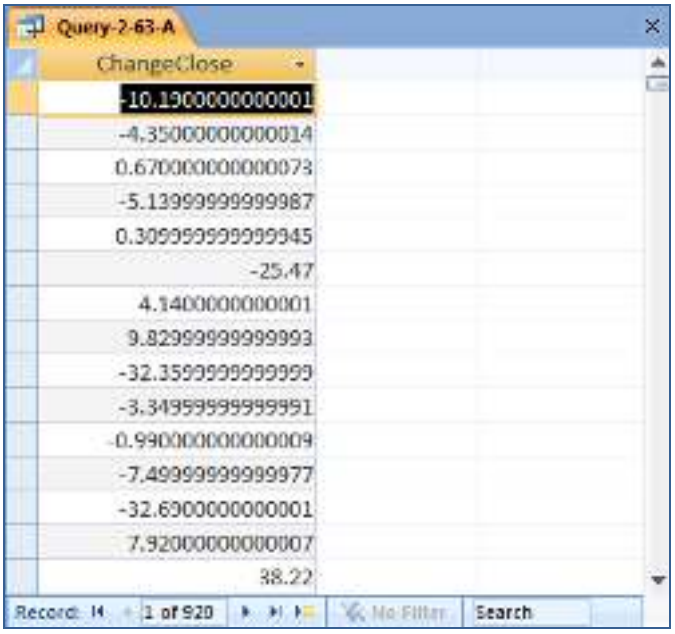
The following questions refer to the NDX table of data as described starting on page 67. You can obtain a copy of this data in the Access database, DBPe11-NDX.accdb located on this text's Web site at www.pearsonhighered.com/kroenke.

2.63 Write SQL queries to produce the following results:

A. The ChangeClose on Fridays.

Solutions to Project Questions 2.63.A – 2.63.H are contained in the Microsoft Access database *DBPe11-IM-NDX.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    ChangeClose
FROM      NDX
WHERE     TDayOfWeeK = 'Friday';
```



B. The minimum, maximum, and average ChangeClose on Fridays.

Solutions to Project Questions 2.63.A – 2.63.H are contained in the Microsoft Access database *DBPe11-IM-NDX.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

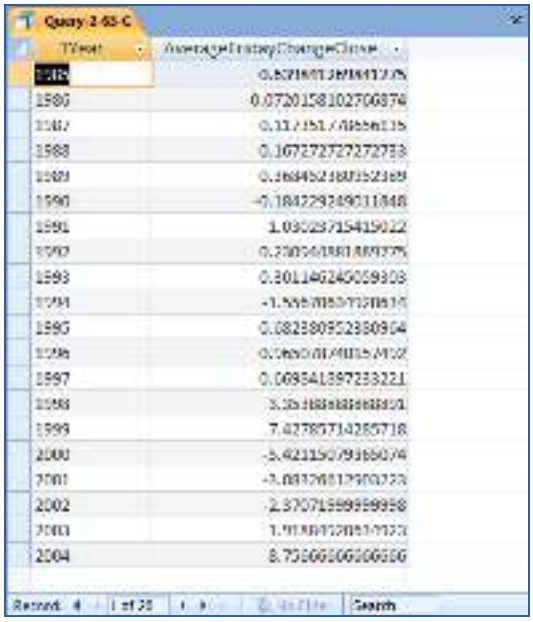
```
SELECT   MIN (ChangeClose) AS MinFridayChangeClose,
         MAX (ChangeClose) AS MaxFridayChangeClose,
         AVG (ChangeClose) AS AverageFridayChangeClose
FROM     NDX
WHERE    TDayOfWeeK = 'Friday';
```

C. The average ChangeClose grouped by TYear. Show TYear.

Since TYear is being displayed, it makes sense to sort the results by TYear although this is not explicitly stated in the question.

Solutions to Project Questions 2.63.A – 2.63.H are contained in the Microsoft Access database *DBPe11-IM-NDX.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   TYear, AVG (ChangeClose) AS AverageChangeClose
FROM     NDX
GROUP BY TYear
ORDER BY TYear;
```



D. The average ChangeClose grouped by TYear and TMonth. Show TYear and TMonth.

Since TYear and TMonth are being displayed, it makes sense to sort the results by TYear and TMonth although this is not explicitly stated in the question.

Solutions to Project Questions 2.63.A – 2.63.H are contained in the Microsoft Access database *DBPe11-IM-NDX.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   TYear, TMonth,
         AVG (ChangeClose) AS AverageChangeClose
FROM     NDX
GROUP BY TYear, TMonth
ORDER BY TYear, TMonth;
```

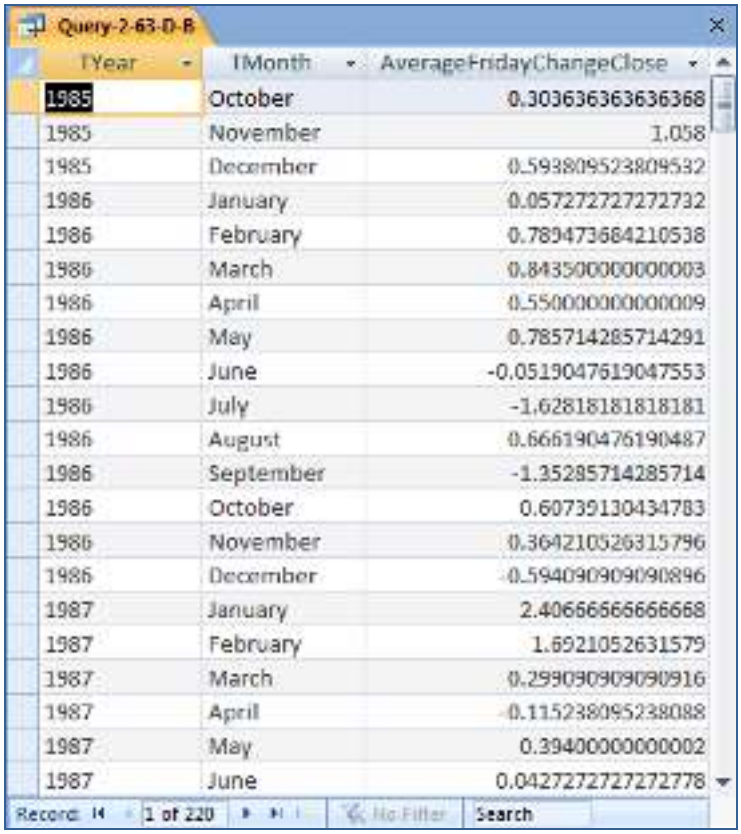
Query-2-63-D-A

TYear	TMonth	AverageFridayChangeClose
1985	December	0.593809523809532
1985	November	1.058
1985	October	0.303636363636368
1986	April	0.550000000000009
1986	August	0.666190476190487
1986	December	-0.594090909090896
1986	February	0.7894736842105 38
1986	January	0.057272727272732
1986	July	-1.628181818181 81
1986	June	-0.0519047619047553
1986	March	0.843500000000003
1986	May	0.785714285714291
1986	November	0.364210526315796
1986	October	0.60739130434783
1986	September	-1.35285714285714
1987	April	-0.115238095238088
1987	August	1.25952380952383
1987	December	1.73863636363637
1987	February	1.6921052631579
1987	January	2.40666666666668
1987	July	0.646363636363638

Record: 1 of 220 No Filter Search

Unfortunately, the table NDX does not contain a numeric value of the month, so in order to sort the months correctly, we need a TMonthNumber which has a column containing a representative number for each month (January = 1, February = 2, etc.). In the DBPe11-NDX.accdb and DBPe11-IM-Ch02-NDX.accdb databases, this column is included in a table named NDX_FULL.

```
SELECT   TYear, TMonth,
         AVG (ChangeClose) AS AverageFridayChangeClose
FROM     NDX_Full
GROUP BY TYear, TMonth, TMonthNumber
ORDER BY TYear, TMonthNumber;
```



Query-2-63-D-B

TYear	TMonth	AverageFridayChangeClose
1985	October	0.303636363636368
1985	November	1.058
1985	December	0.593809523809532
1986	January	0.057272727272732
1986	February	0.789473684210538
1986	March	0.843500000000003
1986	April	0.550000000000009
1986	May	0.785714285714291
1986	June	-0.0519047619047553
1986	July	-1.62818181818181
1986	August	0.666190476190487
1986	September	-1.35285714285714
1986	October	0.60739130434783
1986	November	0.364210526315796
1986	December	-0.594090909090896
1987	January	2.40666666666668
1987	February	1.6921052631579
1987	March	0.299090909090916
1987	April	-0.115238095238088
1987	May	0.39400000000002
1987	June	0.0427272727272778

Record: 1 of 220 No Filter Search

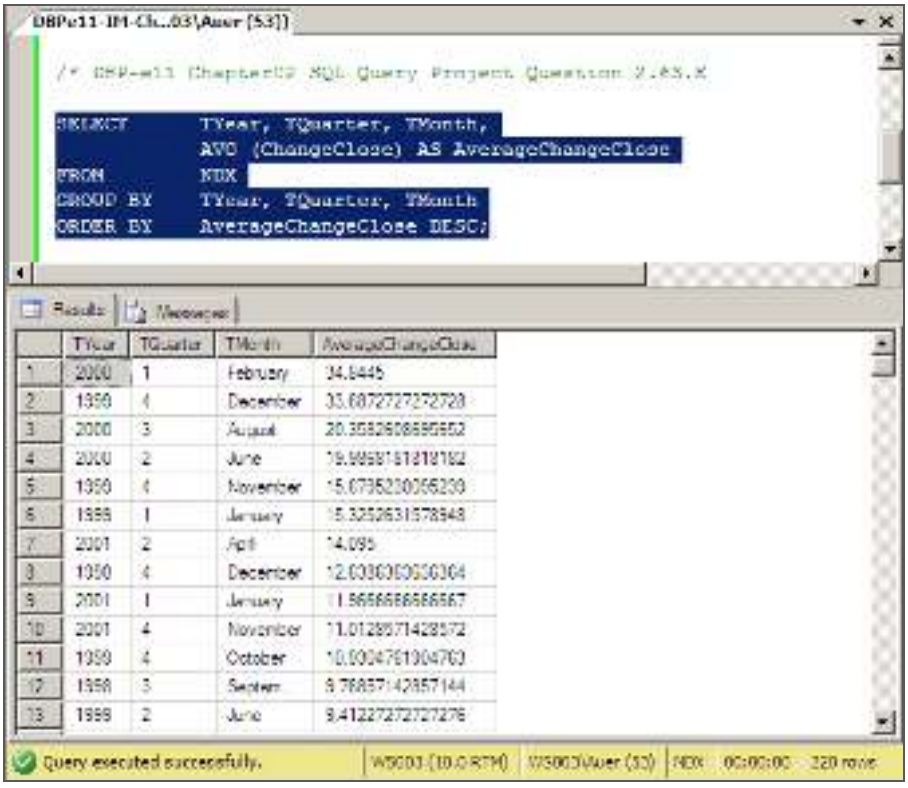
E. The average ChangeClose grouped by TYear, TQuarter, TMonth shown in descending order of the average (you will have to give a name to the average in order to sort by it). Show TYear, TQuarter, and TMonth. Note that months appear in alphabetical and not calendar order. Explain what you need to do to obtain months in calendar order.

Solutions to Project Questions 2.63.A – 2.63.H are contained in the Microsoft Access database *DBPe11-IM-NDX.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   TYear, TQuarter, TMonth,
         AVG (ChangeClose) AS AverageChangeClose
FROM     NDX
GROUP BY TYear, TQuarter, TMonth
ORDER BY AverageChangeClose DESC;
```

Unfortunately, as discussed above, Microsoft Access cannot process the ORDER BY clause correctly when an SQL built-in function is used.

The correct result, obtained from SQL Server 2008, is:



In order to obtain the months in calendar order, we would have to use a numerical value for each month (1, 2, 3, …, 12) and sort by those values.

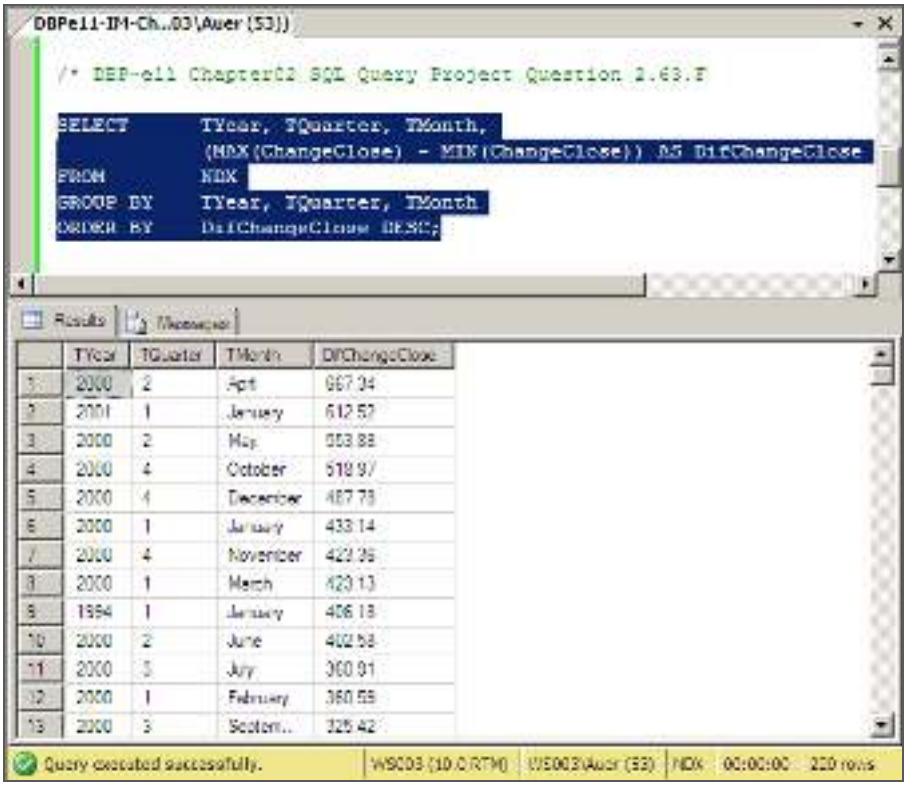
F. The difference between the maximum ChangeClose and the minimum ChangeClose grouped by TYear, TQuarter, TMonth shown in descending order of the difference (you will have to give a name to the difference in order to sort by it). Show TYear, TQuarter, and TMonth.

Solutions to Project Questions 2.63.A – 2.63.H are contained in the Microsoft Access database *DBPe11-IM-NDX.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    TYear, TQuarter, TMonth,
          (MAX (ChangeClose) - MIN(ChangeClose)) AS DifChangeClose
FROM      NDX
GROUP BY TYear, TQuarter, TMonth
ORDER BY DifChangeClose DESC;
```

Unfortunately, as discussed above, Microsoft Access cannot process the ORDER BY clause correctly because it contains an aliased computed result .

The correct result, obtained from SQL Server 2008, is:



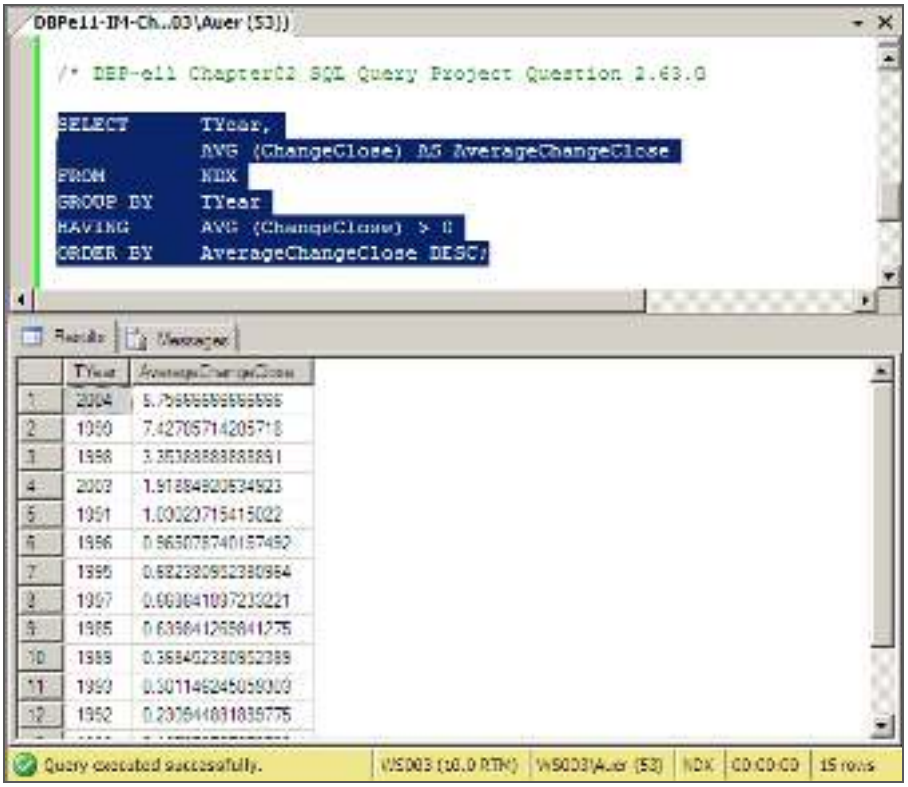
G. *The average ChangeClose grouped by TYear shown in descending order of the average (you will have to give a name to the average in order to sort by it). Show only groups for which the average is positive.*

Solutions to Project Questions 2.63.A – 2.63.H are contained in the Microsoft Access database *DBPe11-IM-NDX.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    TYear,
          AVG (ChangeClose) AS AverageChangeClose
FROM      NDX
GROUP BY  TYear
HAVING    AVG (ChangeClose) > 0
ORDER BY  AverageChangeClose DESC;
```

Unfortunately, as discussed abve, Microsoft Access cannot process the ORDER BY clause correctly because it contains an aliased computed result.

The correct result, obtained from SQL Server 2008, is:



H. Display a single field with the date in the form: day/monthy/year. Do not be concerned with trailing blanks.

Solutions to Project Questions 2.63.A – 2.63.H are contained in the Microsoft Access database *DBPe11-IM-NDX.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

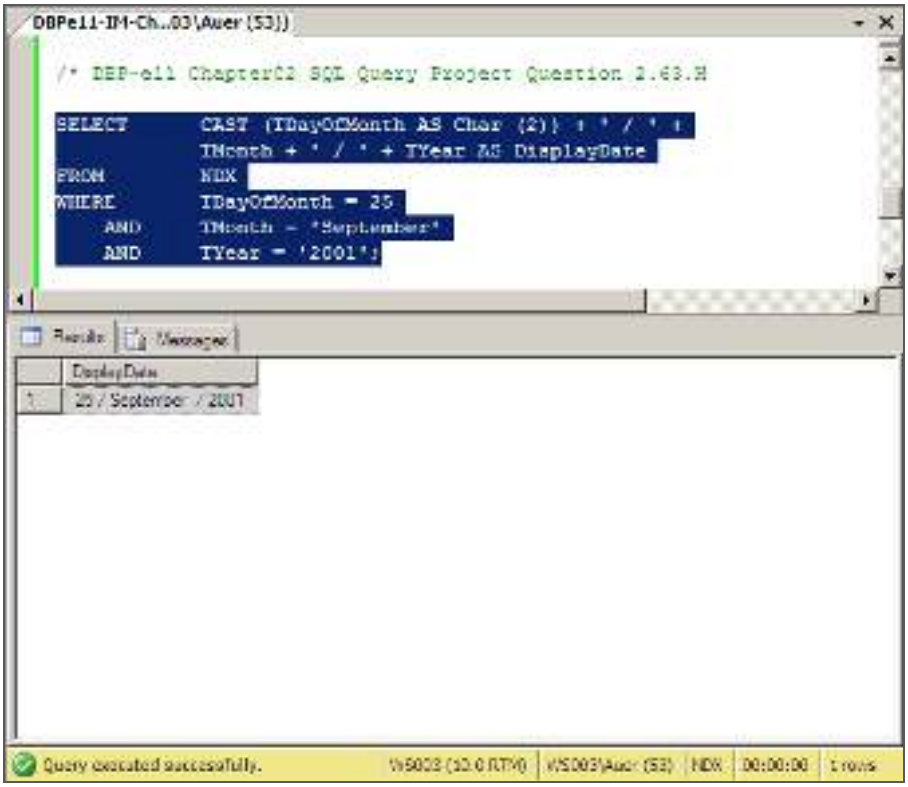
The solution to this question requires the student to use the DBMS help function or other references to figure out a conversion function to convert the numerical day of the month to a character string that can be combined with other data already in character format.

The table NDX does not have a numeric value for month, so the names of the months will appear in the solution. If we want the numeric value of the month, we could use the NDX_Full table, which has a numeric value. We would need to use the data type conversion on this field as well.

The SQL Statement using SQL Server 2008 character string functions is:

```
SELECT     CAST (TDayOfMonth AS Char (2)) + ' / ' +
           TMonth + ' / ' + TYear AS DisplayDate
FROM       NDX
WHERE      TDayOfMonth = 25
   AND     TMonth = 'September'
   AND     TYear = '2001';
```

The SQL Server 2008 result is:



The SQL Statement using Microsoft Access 2007 character string functions is:

```
SELECT CStr(TDayOfMonth) +' / '+
       TMonth +' / '+TYear AS DisplayDate
FROM   NDX
WHERE  NDX.TDayOfMonth =25
   AND NDX.TMonth ='September'
   AND NDX.TYear ='2001';
```

The Microsoft Access 2007 result is:

2.64 It is possible that volume (the number of shares traded) has some correlation with the direction of the stock market. Use the SQL you have learned in this chapter to investigate that possibility. Develop at least five different SQL statements in your investigation.

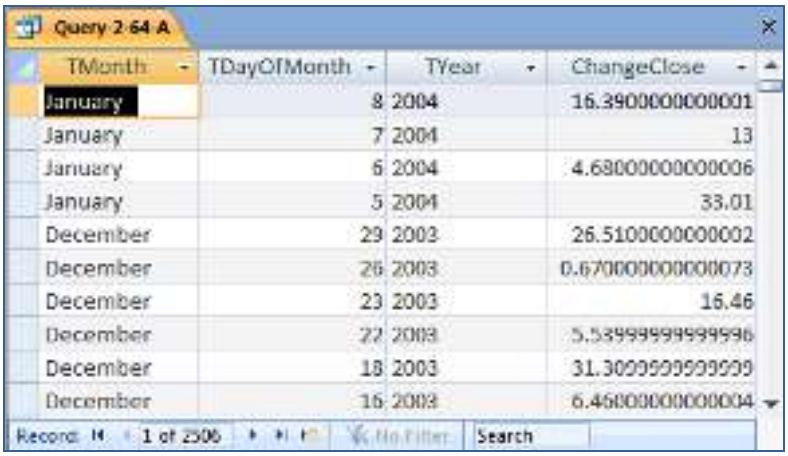
If volume is correlated with the direction of the stock market, this means that there should be either:

(1) POSITIVE CORRELEATION: Higher volume when the market closes higher, or

(2) NEGATIVE CORRELATION: Higher volume when the market closes lower.

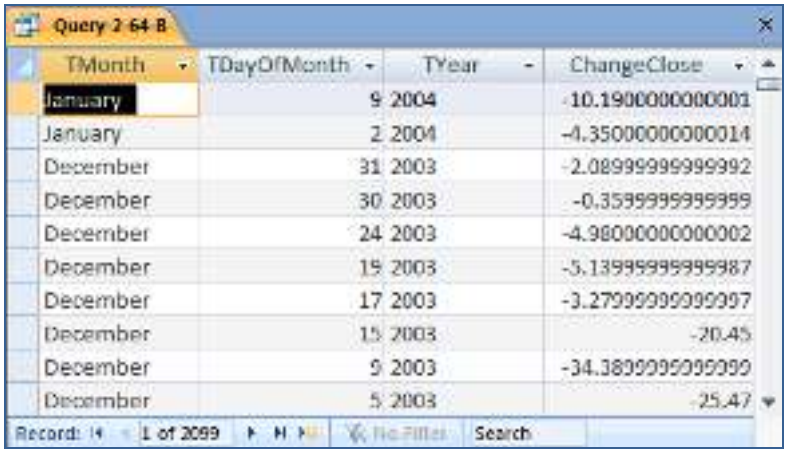
When does the market close higher? When NDX.ChangeClose is positive.

```
SELECT     TMonth, TDayOfMonth, TYear, ChangeClose
FROM       NDX
WHERE      ChangeClose > 0;
```



When does the market close lower? When NDX.ChangeClose is negative.

```
SELECT    TMonth, TDayOfMonth, TYear, ChangeClose
FROM      NDX
WHERE     ChangeClose < 0;
```



Now, what are the average positive and negative changes?

```
SELECT    AVG (ChangeClose) AS AvgPositiveChange
FROM      NDX
WHERE     ChangeClose > 0;
```


```
SELECT    AVG (ChangeClose) AS AvgNegativeChange
FROM      NDX
WHERE     ChangeClose < 0;
```

Now, what are the average volumes associated with the positive and negative changes?

```
SELECT    AVG (ChangeClose) AS AvgPositiveChange,
          AVG (Volume) AS AvgVolumeOnPositiveChange
FROM      NDX
WHERE     ChangeClose > 0;
```


```
SELECT     AVG (ChangeClose) AS AvgNegativeChange,
           AVG (Volume) AS AvgVolumeOnNegativeChange
FROM       NDX
WHERE      ChangeClose < 0;
```

So, when there is a positive, or upward, change in the market we have an average volume of 641417.1117318 shares traded, and when we have a negative, or downward, change in the market we have an average volume of 6742500.66698428 shares. These numbers do not look significantly different, we will conclude that there is no correlation between the direction of the market movement and the volume of shares traded (if we wanted to be more formal, we could use a statistical procedure and do a hypothesis test as to whether or not there is really a statistically significant difference between these two numbers).

ANSWERS TO MARCIA'S DRY CLEANING PROJECT QUESTIONS

Marcia's Dry Cleaning is an upscale dry cleaners in a well-to-do suburban neighborhood. Marcia makes her business stand out from the competition by providing superior customer service. She wants to keep track of each of her customers and their orders. Ultimately, she wants to notify them that their clothes are ready via email. To provide this service, she has developed an initial database with several tables. Three of those tables are the following:

CUSTOMER (CustomerID, FirstName, LastName, Phone, Email)

ORDER (InvoiceNumber, *CustomerNumber*, DateIn, DateOut, TotalAmt)

ORDER_ITEM (*InvoiceNumber*, ItemNumber, Item, Quantity, UnitPrice)

In the database schema above, the primary keys are underlined and the foreign keys are shown in italics.

The database is named MDC. The column characteristics for the tables are shown in Figures 2-30, 2-31, and 2-32 [on the next page]. The data for these tables are shown in Figures 2-33, 2-34, and 2-35 [on the second and third following pages].

We recommend that you create an Access 2007 database named MDC-Ch02.accdb using the database characteristics and data above, and then use this database to test your solutions to the questions in this section.

Column Name	Type	Key	Required	Remarks
CustomerID	Number	Primary Key	Yes	Long Integer
FirstName	Text (25)	No	Yes	
LastName	Text (25)	No	Yes	
Phone	Text (12)	No	No	
Email	Text (100)	No	No	

Figure 2-30 - Column Characteristics for the CUSTOMER Table

Column Name	Type	Key	Required	Remarks
InvoiceNumber	Number	Primary Key	Yes	Long Integer
DataIn	Date/Time	No	Yes	
DataOut	Date/Time	No	No	
TotalAmount	Currency	No	No	Two Decimal Places
CustomerNumber	Number	Foreign Key	Yes	Long Integer

Figure 2-31 - Column Characteristics for the ORDER Table

Column Name	Type	Key	Required	Remarks
InvoiceNumber	Number	Primary Key, Foreign Key	Yes	Long Integer
ItemNumber	Number	Primary Key	Yes	Long Integer
Item	Text (50)	No	Yes	
Quantity	Number	No	Yes	Long Integer
UnitPrice	Currency	No	Yes	Two Decimal Places

Figure 2-32 - Column Characteristics for the ORDER_ITEM Table

CustomerID	FirstName	LastName	Phone	Email
1	Nikki	Kaccaton	723-543-1233	NKaccaton@somewhere.com
2	Brenda	Catnazaro	723-543-2344	BCatnazaro@somewhere.com
3	Bruce	LeCat	723-543-3455	BLeCat@somewhere.com
4	Betsy	Miller	723-654-3211	BMiller@somewhere.com
5	George	Miller	723-654-4322	GMiller@somewhere.com
6	Kathy	Miller	723-514-9877	KMiller@somewhere.com
7	Betsy	Miller	723-514-8766	BMiller@somewhere.com

This Email should be BMiller@ elsewhere.com

Figure 2-33 - Sample Data for the CUSTOMER table

InvoiceNumber	DateIn	DateOut	TotalAmount	CustomerID
2009001	04-Oct-09	06-Oct-09	$158.50	1
2009002	04-Oct-09	06-Oct-09	$25.00	2
2009003	06-Oct-09	08-Oct-09	$55.00	1
2009004	06-Oct-09	08-Oct-09	$17.50	4
2009005	07-Oct-09	11-Oct-09	$12.00	6
2009006	11-Oct-09	13-Oct-09	$152.50	3
2009007	11-Oct-09	13-Oct-09	$7.00	3
2009008	12-Oct-09	14-Oct-09	$140.50	7
2009009	12-Oct-09	14-Oct-09	$27.00	5

This number should be $49.00

Figure 2-35 - Sample Data for the ORDER table

InvoiceNumber	ItemNumber	Item	Quantity	UnitPrice
2009001	1	Blouse	2	$3.50
2009001	2	Dress Shirt	5	$2.50
2009001	3	Formal Gown	2	$10.00
2009001	4	Slacks-Mens	10	$5.00
2009001	5	Slacks-Womens	10	$6.00
2009001	6	Suit-Mens	1	$9.00
2009002	1	Dress Shirt	10	$2.50
2009003	1	Slacks-Mens	5	$5.00
2009003	2	Slacks-Womens	4	$6.00
2009004	1	Dress Shirt	7	$2.50
2009005	1	Blouse	2	$3.50
2009005	2	Dress Shirt	2	$2.50
2009006	1	Blouse	5	$3.50
2009006	2	Dress Shirt	10	$2.50
2009006	3	Slacks-Mens	10	$5.00
2009006	4	Slacks-Womens	10	$6.00
2009007	1	Blouse	2	$3.50
2009008	1	Blouse	3	$3.50
2009008	2	Dress Shirt	12	$2.50
2009008	3	Slacks-Mens	8	$5.00
2009008	4	Slacks-Womens	10	$6.00
2009009	1	Suit-Mens	3	$9.00

Figure 2-35 - Sample Data for the ORDER_ITEM table

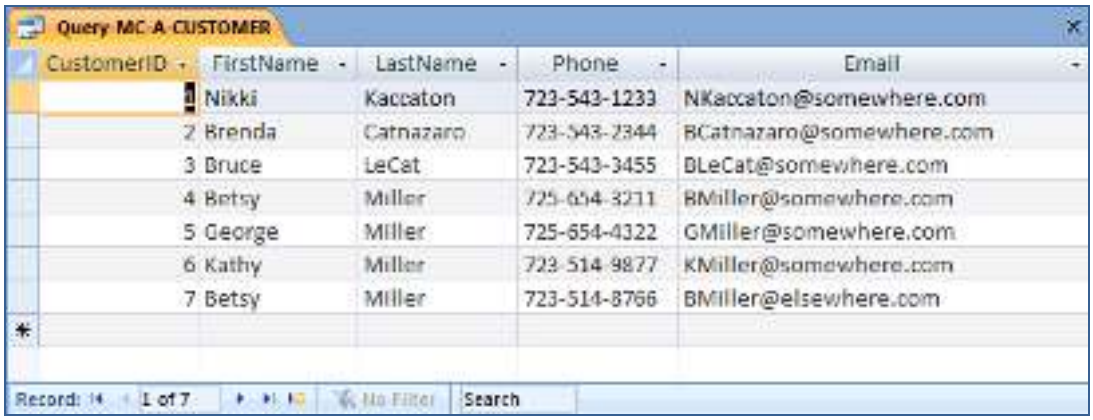
Write SQL statements and show the results based on the MDC data for each of the following:

A. Show all data in each of the tables.

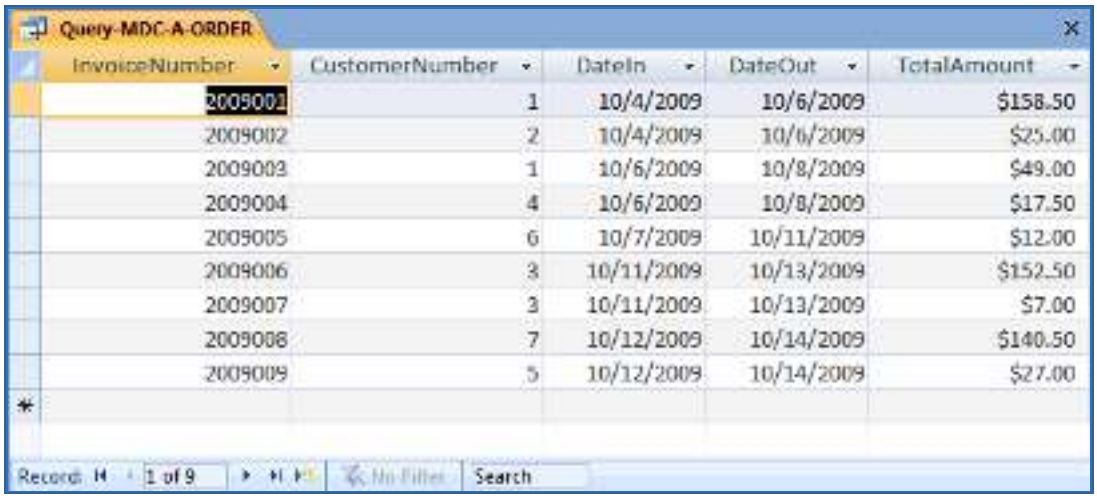
Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    *
FROM      CUSTOMER;
```

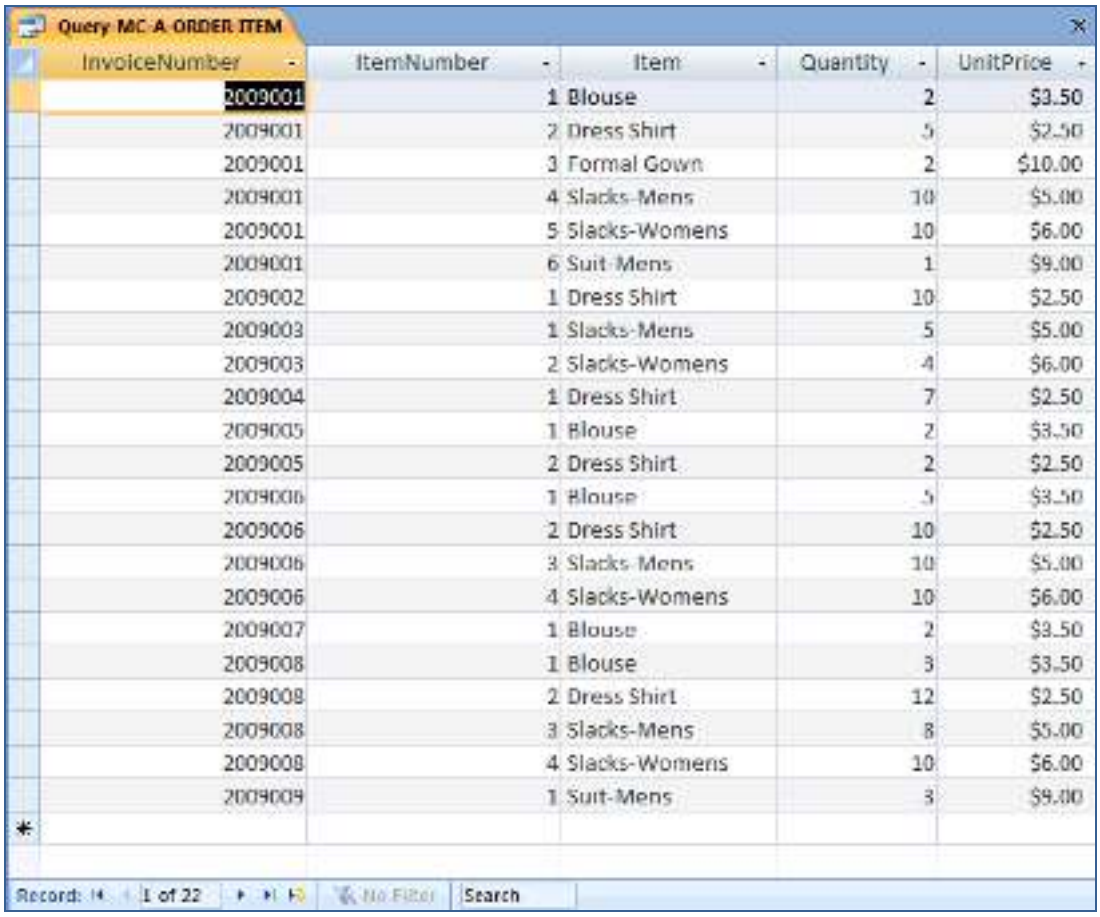
Note the two customers both named Betsy Miller.




```
SELECT    *
FROM      ORDER;
```



```
SELECT    *
FROM      ORDER_ITEM;
```



Query MC-A ORDER ITEM

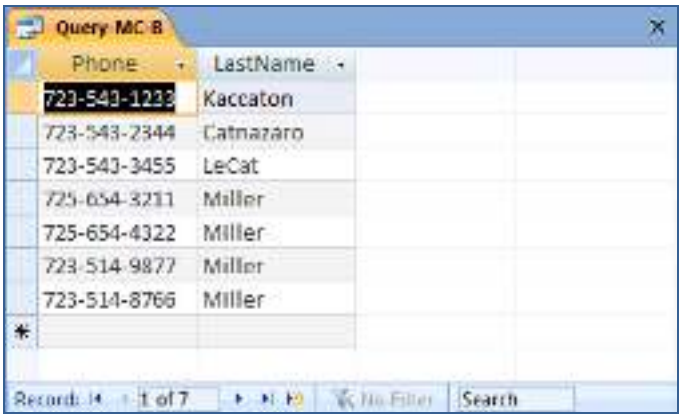
InvoiceNumber	ItemNumber	Item	Quantity	UnitPrice
2009001	1	Blouse	2	$3.50
2009001	2	Dress Shirt	5	$2.50
2009001	3	Formal Gown	2	$10.00
2009001	4	Slacks-Mens	10	$5.00
2009001	5	Slacks-Womens	10	$6.00
2009001	6	Suit-Mens	1	$9.00
2009002	1	Dress Shirt	10	$2.50
2009003	1	Slacks-Mens	5	$5.00
2009003	2	Slacks-Womens	4	$6.00
2009004	1	Dress Shirt	7	$2.50
2009005	1	Blouse	2	$3.50
2009005	2	Dress Shirt	2	$2.50
2009006	1	Blouse	5	$3.50
2009006	2	Dress Shirt	10	$2.50
2009006	3	Slacks-Mens	10	$5.00
2009006	4	Slacks-Womens	10	$6.00
2009007	1	Blouse	2	$3.50
2009008	1	Blouse	3	$3.50
2009008	2	Dress Shirt	12	$2.50
2009008	3	Slacks-Mens	8	$5.00
2009008	4	Slacks-Womens	10	$6.00
2009009	1	Suit-Mens	3	$9.00

Record: 1 of 22 No Filter Search

B. List the Phone and LastName of all customers.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

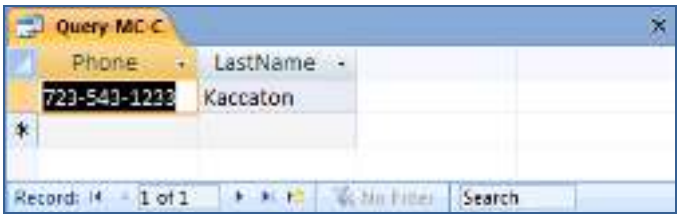
```
SELECT    Phone, LastName
FROM      CUSTOMER;
```



C. List the Phone and LastName for all customers with a FirstName of "Nikki".

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    Phone, LastName
FROM      CUSTOMER
WHERE     FirstName = 'Nikki';
```

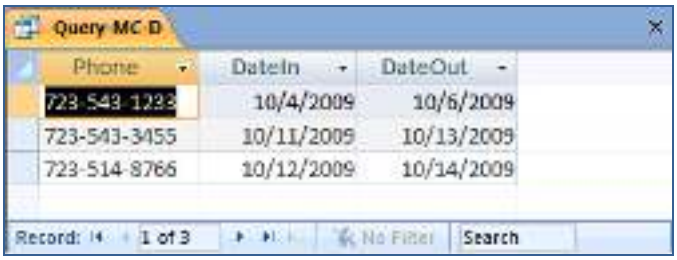


D. List the Phone, DateIn, and DateOut of all orders in excess of 100.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that since ORDER is an SQL reserved word, it must be enclosed in delimiters (square brackets [ ]).

```
SELECT   Phone, DateIn, DateOut
FROM     CUSTOMER, [ORDER]
WHERE    TotalAmount >100
   AND   CUSTOMER.CustomerID = ORDER.CustomerNumber;
```



E. List the Phone and FirstName of all customers whose first name starts with 'B'.

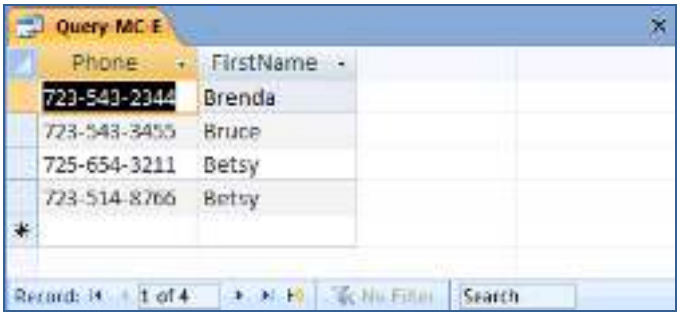
Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

The correct SQL-92 statement, which uses the wildcard %, is:

```
SELECT   Phone, FirstName
FROM     CUSTOMER
WHERE    FirstName LIKE 'B%';
```

However, MS Access uses the wildcard *, which gives the following SQL statement:

```
SELECT   Phone, FirstName
FROM     CUSTOMER
WHERE    FirstName LIKE 'B*';
```



F. List the Phone and FirstName of all customers whose last name includes the characters, 'cat'.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

The correct SQL-92 statement, which uses the wildcard %, is:

```
SELECT   Phone, FirstName
FROM     CUSTOMER
WHERE    LastName LIKE '%cat%';
```

However, MS Access uses the wildcard *, which give the following SQL statement:

```
SELECT   Phone, FirstName
FROM     CUSTOMER
WHERE    LastName LIKE '*cat*';
```

G. List the Phone, FirstName, and LastName for all customers whose second and third characters of phone number is 23.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that since the phone numbers in this database include the area code, we are really finding phone numbers with '23' as the second and third numbers in the area code. We could, off course, write statements to find '23' in the prefix or in the 4-digit sequence portion of the phone number.

The correct SQL-92 statement, which uses the wildcards % and _, is:

```
SELECT   Phone, FirstName, LastName
FROM     CUSTOMER
WHERE    Phone LIKE '_23%';
```

However, MS Access uses the wildcards * and ?, which give the following SQL statement:

```
SELECT    Phone, FirstName, LastName
FROM      CUSTOMER
WHERE     Phone LIKE '?23*';
```

H. Determine the maximum and minimum TotalAmounts.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that since ORDER is an SQL reserved word, it must be enclosed in delimiters (square brackets [ ]).

```
SELECT    MAX (TotalAmt) AS MaxTotalAmount,
          MIN (TotalAmt) AS MinTotalAmount
FROM      [ORDER];
```

I. Determine the average TotalAmount.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that since ORDER is an SQL reserved word, it must be enclosed in delimiters (square brackets [ ]).

```
SELECT    AVG (TotalAmt) AS AvgTotalAmount
FROM      [ORDER];
```

J. Count the number of customers.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    Count (*)AS NumberOfCustomers
FROM      CUSTOMER;
```

K. Group customers by LastName and then by FirstName.

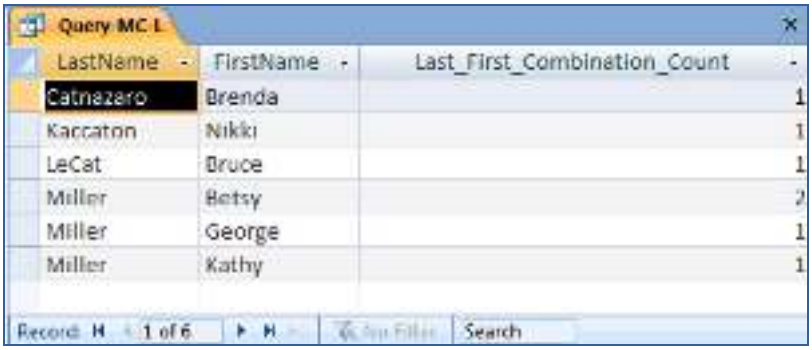
Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   LastName, FirstName
FROM     CUSTOMER
GROUP BY LastName, FirstName;
```

L. Count the number of customers having each combination of LastName and FirstName.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   LastName, FirstName,
         COUNT (*) AS Last_First_Combination_Count
FROM     CUSTOMER
GROUP BY LastName, FirstName;
```

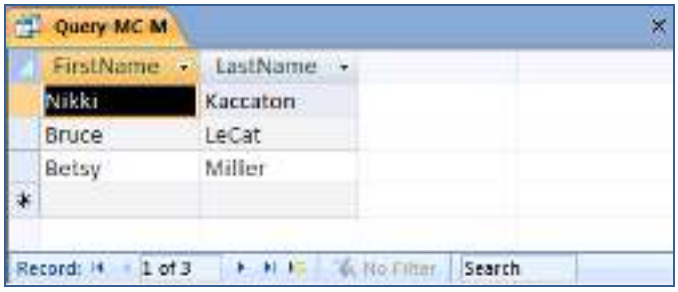


M. *Show the FirstName and LastName of all customers who have had an order with TotalAmount greater than 100. Use a subquery. Present the results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that since ORDER is an SQL reserved word, it must be enclosed in delimiters (square brackets [ ]).

```
SELECT   FirstName, LastName
FROM     CUSTOMER
WHERE    CustomerID IN
         (SELECT CustomerNumber
         FROM [ORDER]
         WHERE TotalAmount > 100)
ORDER BY LastName, FirstName DESC;
```

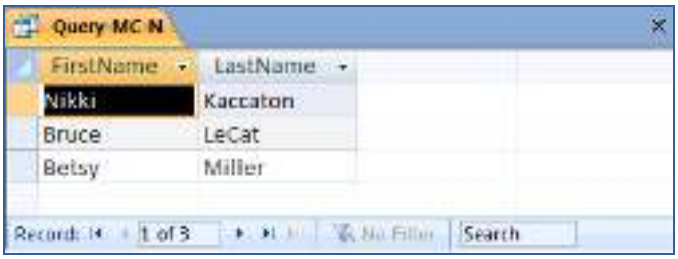


N. *Show the FirstName and LastName of all customers who have had an order with TotalAmount greater than 100. Use a join. Present the results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that since ORDER is an SQL reserved word, it must be enclosed in delimiters (square brackets [ ]).

```
SELECT   FirstName, LastName
FROM     CUSTOMER, [ORDER]
WHERE    CUSTOMER.CustomerID = [ORDER].CustomerNumber
   AND   TotalAmount > 100
ORDER BY LastName, FirstName DESC;
```

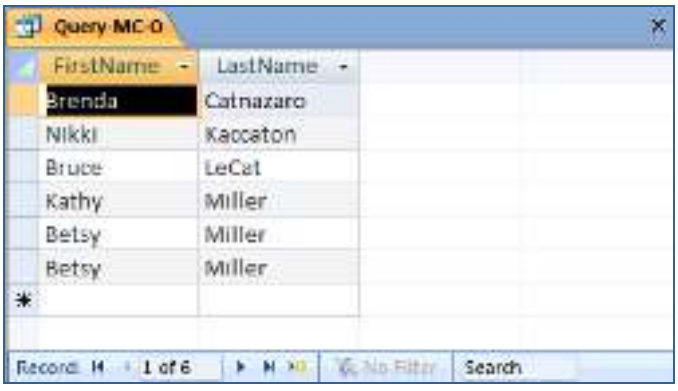


O. *Show the FirstName and LastName of all customers who have had an order with an Item named "Dress Shirt". Use a subquery. Present the results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that since ORDER is an SQL reserved word, it must be enclosed in delimiters (square brackets [ ]).

```
SELECT   FirstName, LastName
FROM     CUSTOMER
WHERE    CustomerID IN
         (SELECT  CustomerNumber
         FROM     [ORDER]
         WHERE    InvoiceNumber IN
                  (SELECT InvoiceNumber
                  FROM ORDER_ITEM
                  WHERE Item = 'Dress Shirt'))
ORDER BY LastName, FirstName DESC;
```

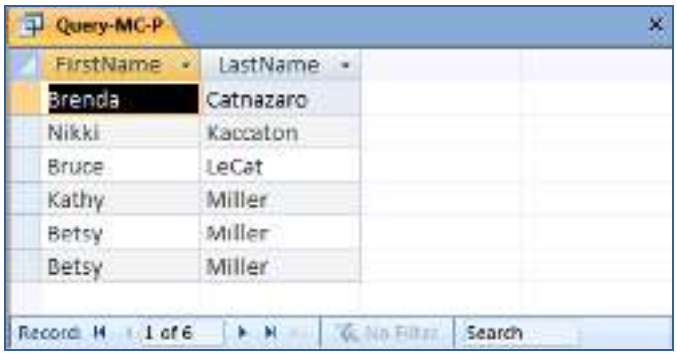


P. *Show the FirstName and LastName of all customers who have had an order with an Item named "Dress Shirt". Use a join. Present the results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that since ORDER is an SQL reserved word, it must be enclosed in delimiters (square brackets [ ]).

```
SELECT   FirstName, LastName
FROM     CUSTOMER, [ORDER], ORDER_ITEM
WHERE    CUSTOMER.CustomerID = [ORDER].CustomerNumber
   AND   [ORDER].InvoiceNumber = ORDER_ITEM.InvoiceNumber
   AND   ORDER_ITEM.Item = 'Dress Shirt'
ORDER BY LastName, FirstName DESC;
```



Q. *Show the FirstName, LastName and TotalAmount of all customers who have had an order with an Item named "Dress Shirt". Use a join with a subquery. Present results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBPe11-IM-Ch01-MDC.accdb* which is available on the Instructor's Resource CD-ROM and the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Since we want to display data in fields from two tables, these tables must be combined with a join. Data in a table without displayed fields can still be brought into the query with a subquery. Therefore, we will join CUSTOMER and ORDER, while using a subquery with ORDER_ITEM.

Note that since ORDER is an SQL reserved word, it must be enclosed in delimiters (square brackets [ ]).

```
SELECT   FirstName, LastName, TotalAmount
FROM     CUSTOMER, [ORDER]
WHERE    CUSTOMER.CustomerID = [ORDER].CustomerNumber
   AND   [ORDER].InvoiceNumber IN
                  (SELECT InvoiceNumber
                   FROM ORDER_ITEM
                   WHERE Item = 'Dress Shirt')
ORDER BY LastName, FirstName DESC;
```

ANSWERS TO MORGAN IMPORTING PROJECT QUESTIONS

Morgan Importing purchases antiques and home furnishings in Asia and ships those items to a warehouse facility in Los Angeles. Mr. Morgan uses a database to keep a list of items purchased, shipments and shipment items. His database includes the following tables:

SHIPMENT (ShipmentID, ShipperName, ShipperInvoiceNumber, DepartureDate, ArrivalDate, InsuredValue)

SHIPMENT_ITEM (ShipmentID, ShipmentItemID, ItemID, Value)

ITEM (ItemID, Description, PurchaseDate, Store, City, Quantity, LocalCurrencyAmt, ExchangeRate)

In the database schema above, the primary keys are underlined and the foreign keys are shown in italics.

The database is named MI. The column characteristics for the tables are shown in Figures 2-36, 2-37, and 2-38. The data for these tables are shown in Figures 2-39, 2-40, and 2-41.

We recommend that you create an Access 2007 database named MI-Ch02.accdb using the database characteristics and data above, and then use this database to test your solutions to the questions in this section.

Column Name	Type	Key	Required	Remarks
ShipmentID	Number	Primary Key	Yes	Long Integer
ShipperName	Text (35)	No	Yes	
ShipperInvoiceNumber	Number	No	Yes	Long Integer
DepartureDate	Date/Time	No	No	
ArrivalDate	Date/Time	No	No	
InsuredValue	Currency	No	No	Two Decimal Places

Figure 2-36 - Column Characteristics for the SHIPMENT Table

Column Name	Type	Key	Required	Remarks
ShipmentID	Number	Primary Key, Foreign Key	Yes	Long Integer
ShipmentItemID	Number	Primary Key	Yes	Long Integer
ItemID	Number	Foreign Key	Yes	Long Integer
Quantity	Number	No	Yes	Long Integer
Value	Currency	No	Yes	Two Decimal Places

Figure 2-37 - Column Characteristics for the SHIPMENT_ITEM Table

Column Name	Type	Key	Required	Remarks
ItemID	Number	Primary Key	Yes	Long Integer
Description	Text (255)	No	Yes	Long Integer
PurchaseDate	Date/Time	No	Yes	
Store	Text (50)	No	Yes	
City	Text (35)	No	Yes	
Quantity	Number	No	Yes	Long Integer
LocalCurrencyAmt	Number	No	Yes	Decimal, 18 Auto
ExchangeRate	Number	No	Yes	Decimal, 12 Auto

Figure 2-38 - Column Characteristics for the ITEM Table

ShipmentID	ShipperName	ShipperInvoiceNumber	DepartureDate	ArrivalDate	InsuredValue
1	ABC Trans-Oceanic	2009651	10-Dec-08	15-Mar-09	$15,000.00
2	ABC Trans-Oceanic	2009012	10-Jan-09	20-Mar-09	$12,000.00
3	Worldwide	49100300	05-May-09	17-Jun-09	$27,500.00
4	International	399400	02-Jun-09	17-Jul-09	$7,500.00
5	Worldwide	84699443	10-Jul-09	28-Jul-09	$25,000.00
6	International	486955	05-Aug-09	11-Sep-09	$18,000.00

Figure 2-39 - Sample Date for the SHIPMENT Table

ShipmentID	ShipmentItemID	ItemID	Quantity	Value
4	1	4	40	$1,200.00
4	2	3	6	$9,500.00
4	3	2	75	$4,500.00

Figure 2-39 - Sample Date for the SHIPMENT_ITEM Table

ItemID	Description	PurchaseDate	Store	City	Quantity	LocalCurrencyAmt	ExchangeRate
1	QE Dining Set	07-Apr-09	Eastern Treasures	Manila	2	403405	0.01774
2	Willow Serving Dishes	15-Jul-09	Jade Antiques	Singapore	75	102	0.5903
3	Large Bureau	17-Jul-09	Eastern Sales	Singapore	8	2000	0.5903
4	Brass Lamps	20-Jul-09	Jade Antiques	Singapore	40	50	0.5903

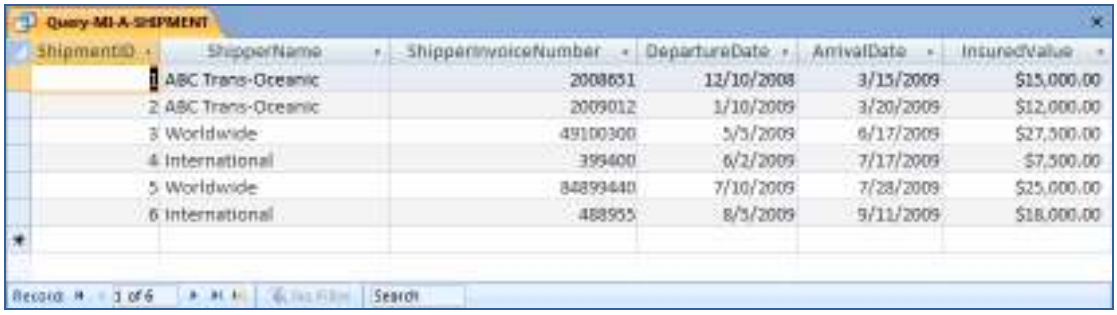
Figure 2-39 - Sample Date for the ITEM Table

Write SQL statements and show the results based on the MDC data for each of the following:

A. Show all data in each of the tables.

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    *
FROM      SHIPMENT;
```



ShipmentID	ShipperName	ShipperInvoiceNumber	DepartureDate	ArrivalDate	InsuredValue
1	ABC Trans-Oceanic	2008651	12/10/2008	3/15/2009	$15,000.00
2	ABC Trans-Oceanic	2009012	1/10/2009	3/20/2009	$12,000.00
3	Worldwide	49100300	5/5/2009	6/17/2009	$27,500.00
4	International	399400	6/2/2009	7/17/2009	$7,500.00
5	Worldwide	84899440	7/10/2009	7/28/2009	$25,000.00
6	International	488955	8/3/2009	9/11/2009	$18,000.00

```
SELECT    *
FROM      SHIPMENT_ITEM;
```

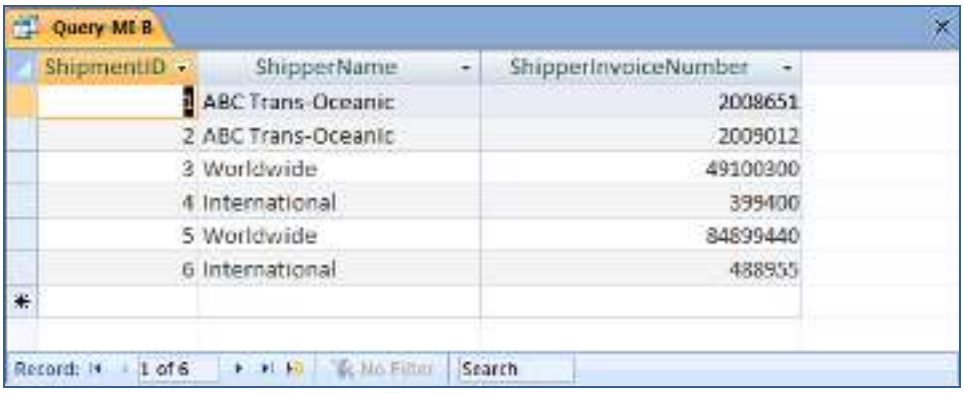
ShipmentID	ShipmentItemID	ItemID	Quantity	Value
1	1	4	40	$1,200.00
4	2	3	8	$9,500.00
4	3	2	75	$4,500.00

```
SELECT    *
FROM      ITEM_PURCHASE;
```

B. List the ShipmentID, ShipperName, and ShipperInvoiceNumber of all shipments.

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

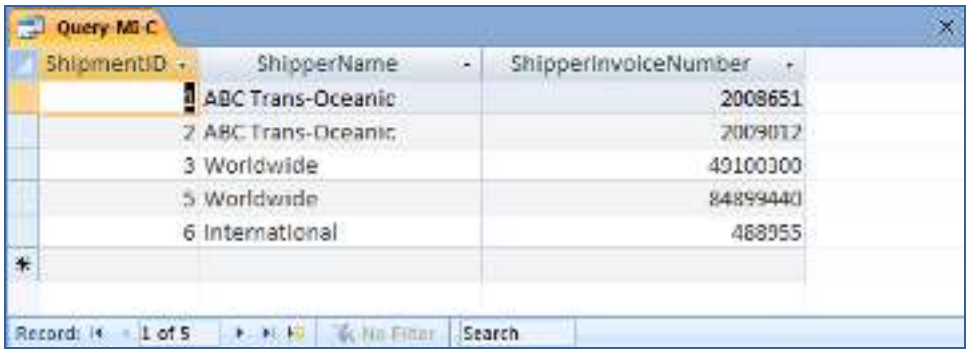
```
SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber
FROM     SHIPMENT;
```



C. List the ShipmentID, ShipperName, and ShipperInvoiceNumber for all shipments with an insured value greater than 10000.

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber
FROM     SHIPMENT
WHERE    InsuredValue > 10000;
```



D. List the ShipmentID, ShipperName, and ShipperInvoiceNumber of all shippers whose name starts with "AB".

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

The correct SQL-92 statement, which uses the wildcard %, is:

```
SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber
FROM     SHIPMENT
WHERE    Shipper LIKE 'AB%';
```

However, MS Access uses the wildcard *, which give the following SQL statement:

```
SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber
FROM     SHIPMENT
WHERE    Shipper LIKE 'AB*';
```

E. Assume DepartureDate and ArrivalDate are in the format MM/DD/YY. List the ShipmentID, ShipperName, and ShipperInvoiceNumber and ArrivalDate of all shipments that departed in December.

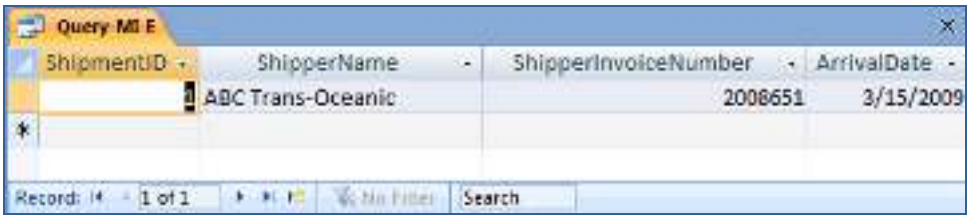
Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

The correct SQL-92 statement, which uses the wildcard %, is:

```
SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber, ArrivalDate
FROM     SHIPMENT
WHERE    DepartureDate LIKE '12%';
```

However, MS Access uses the wildcard *, which gives the following SQL statement:

```
SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber, ArrivalDate
FROM     SHIPMENT
WHERE    DepartureDate LIKE '12*';
```



F. *Assume DepartureDate and ArrivalDate are in the format MM/DD/YY. List the ShipmentID, ShipperName, and ShipperInvoiceNumber and ArrivalDate of all shipments that departed on the 10th of any month.*

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

The correct SQL-92 statement, which uses the wildcards % and _, is:

```
SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber, ArrivalDate
FROM     SHIPMENT
WHERE    DepartureDate LIKE '___10%';
```

However, MS Access uses the wildcards * and ?, which give the following SQL statement:

```
SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber, ArrivalDate
FROM     SHIPMENT
WHERE    DepartureDate LIKE '???10*';
```

Further, MS Access does NOT show the leading zero in MM, so we must add a compound WHERE clause to get months without the leading zeros:

```
SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber, ArrivalDate
FROM     SHIPMENT
WHERE    DepartureDate LIKE '???10*'
   OR    DepartureDate LIKE '??10*';
```

G. Determine the maximum and minimum InsuredValue.

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   MAX (InsuredValue) AS MaxInsuredValue,
         MIN (InsuredValue) AS MinInsuredValue,
FROM     SHIPMENT;
```

H. Determine the average InsuredValue.

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   AVG (InsuredValue) AS AvgInsuredValue
FROM     SHIPMENT;
```

I. Count the number of shipments.

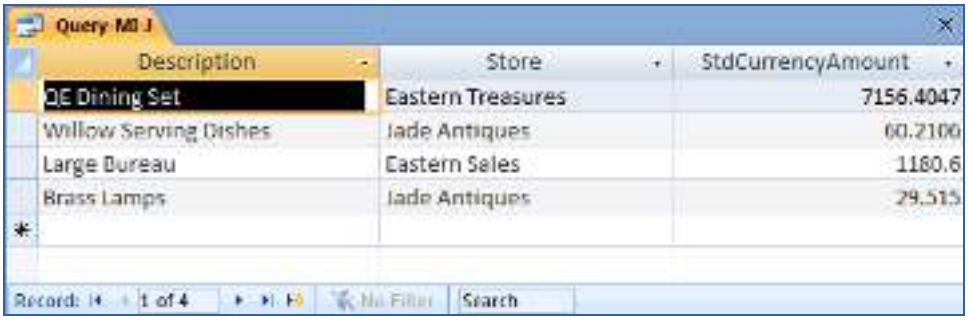
Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   COUNT (*) AS NumberOfShipments
FROM     SHIPMENT;
```

J. Show ItemID, Description, Store, and a calculated column named StdCurrencyAmount that is equal to LocalCurrencyAmt times the ExchangeRate for all rows of ITEM_PURCHASE.

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   Item, Store,
         LocalCurrencyAmt * ExchangeRate AS StdCurrencyAmount
FROM     ITEM_PURCHASE;
```



K. Group item purchases by City and Store.

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   City, Store
FROM     ITEM_PURCHASE
GROUP BY City, Store;
```

L. Count the number of purchases having each combination of City and Store.

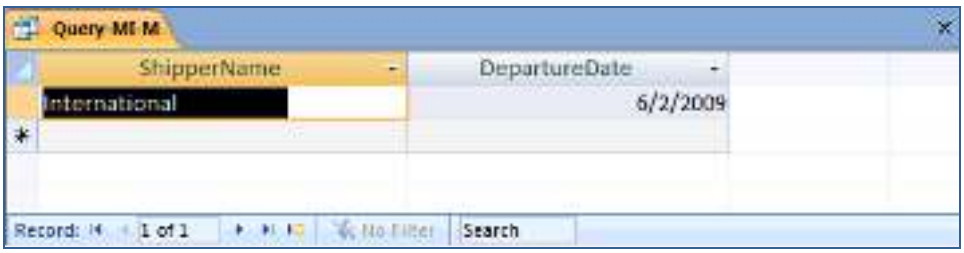
Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   City, Store
         COUNT (*) AS City_Store_Combination_Count
FROM     ITEM_PURCHASE
GROUP BY City, Store;
```

M. Show the ShipperName and DepartureDate of all shipments that have an item with a value of 1000 or more. Use a subquery. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   ShipperName, DepartureDate
FROM     SHIPMENT
WHERE    ShipmentID IN
         (SELECT  ShipmentID
          FROM    SHIPMENT_ITEM
          WHERE   Value = 1000
             OR   Value > 1000)
ORDER BY ShipperName, DepartureDate DESC;
```

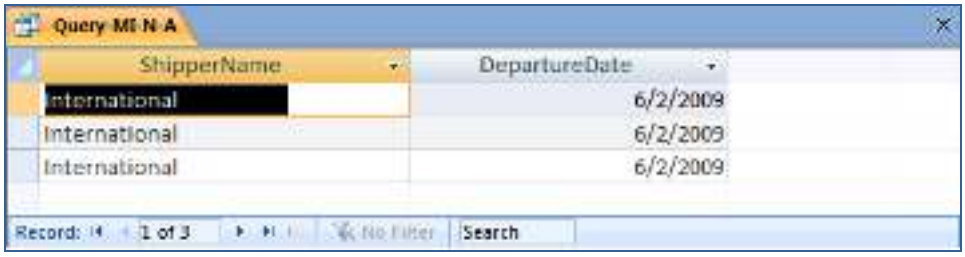


N. *Show the ShipperName and DepartureDate of all shipments that have an item with a value of 1000 or more. Use a join. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.*

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

This question is a little more complicated then it appears. Note how the following three queries determine that there is actually only one shipment that meets the criteria.

```
SELECT   ShipperName, DepartureDate
FROM     SHIPMENT, SHIPMENT_ITEM
WHERE    SHIPMENT.ShipmentID = SHIPMENT_ITEM.ShipmentID
   AND   (Value = 1000 OR Value > 1000)
ORDER BY ShipperName, DepartureDate DESC;
```

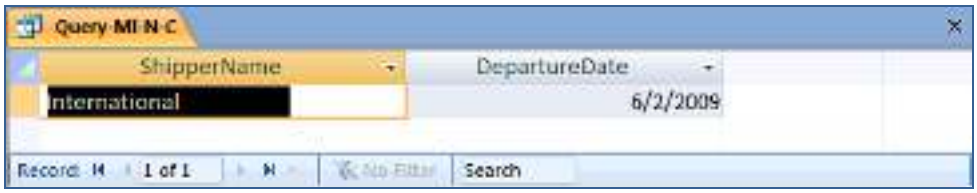


We'll add some more details to confirm that fact that there is actually only one shipment. Note that we can use the *greater than or equal to* operator >= to simplify the WHERE clause:

```
SELECT   SHIPMENT.ShipperInvoiceNumber, ShipmentItemID, Description,
         ShipperName, DepartureDate
FROM     SHIPMENT, SHIPMENT_ITEM, ITEM_PURCHASE
WHERE    SHIPMENT.ShipmentID = SHIPMENT_ITEM.ShipmentID
   AND   SHIPMENT_ITEM.ItemID = ITEM_PURCHASE.ItemID
   AND   Value >= 1000
ORDER BY ShipperName, DepartureDate DESC;
```

We'll now add the UNIQUE keyword to get the proper result:

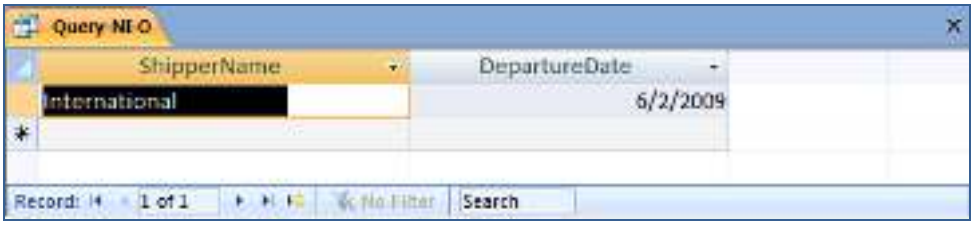
```
SELECT   DISTINCT ShipperName, DepartureDate
FROM     SHIPMENT, SHIPMENT_ITEM
WHERE    SHIPMENT.ShipmentID = SHIPMENT_ITEM.ShipmentID
   AND   Value >= 1000
ORDER BY ShipperName, DepartureDate DESC;
```



O. *Show the ShipperName and DepartureDate of all shipments that have an item that was purchased in Singapore. Use a subquery. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.*

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   ShipperName, DepartureDate
FROM     SHIPMENT
WHERE    ShipmentID IN
         (SELECT  ShipmentID
          FROM    SHIPMENT_ITEM
          WHERE   ItemID IN
                  (SELECT  ItemID
                   FROM    ITEM_PURCHASE
                   WHERE      City = 'Singapore'))
ORDER BY ShipperName, DepartureDate DESC;
```

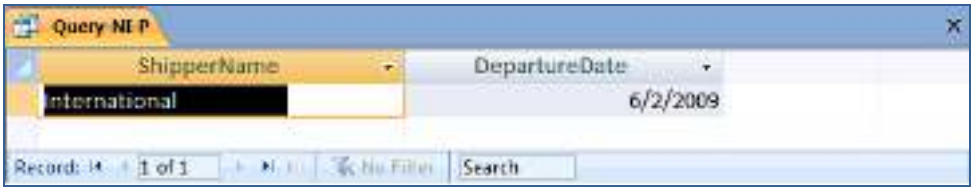


P. *Show the ShipperName and DepartureDate of all shipments that have an item that was purchased in Singapore. Use a join. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.*

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

As in question N, we will have to use a DISTINCT keyword to get the appropriate answer.

```
SELECT   DISTINCT ShipperName, DepartureDate
FROM     SHIPMENT, SHIPMENT_ITEM, ITEM_PURCHASE
WHERE    SHIPMENT.ShipmentID = SHIPMENT_ITEM.ShipmentID
   AND   SHIPMENT_ITEM.ItemID  = ITEM_PURCHASE.ItemID
   AND   City = 'Singapore'
ORDER BY ShipperName, DepartureDate DESC;
```



Q. *Show the ShipperName, DepartureDate of shipment, and Value for items that were purchased in Singapore. Use a combination of a join and a subquery. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.*

Solutions to Moran Importing questions are contained in the Microsoft Access database *DBPe11-IM-Ch02-MI.accdb* which is available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   ShipperName, DepartureDate, Value
FROM     SHIPMENT, SHIPMENT_ITEM
WHERE    SHIPMENT.ShipmentID = SHIPMENT_ITEM.ShipmentID
   AND   ItemID IN
         (SELECT  ItemID
          FROM    ITEM_PURCHASE
          WHERE   City = 'Singapore')
ORDER BY ShipperName, DepartureDate DESC;
```