INSTRUCTOR'S MANUAL

TO ACCOMPANY

David M. Kroenke and David J. Auer

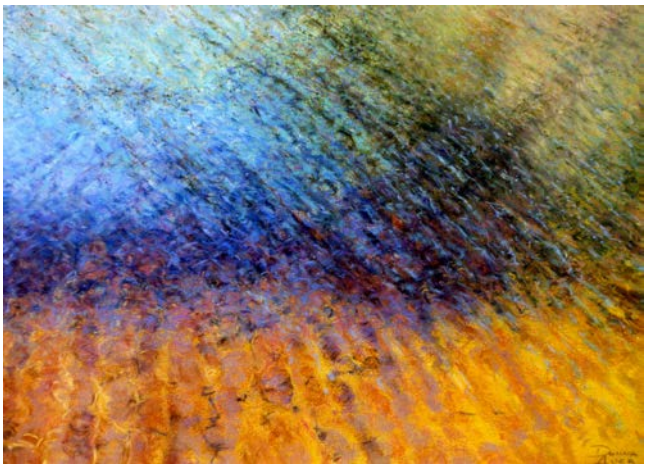
Database Processing

Fundamentals, Design, and Implementation

14th Edition

Chapter 2

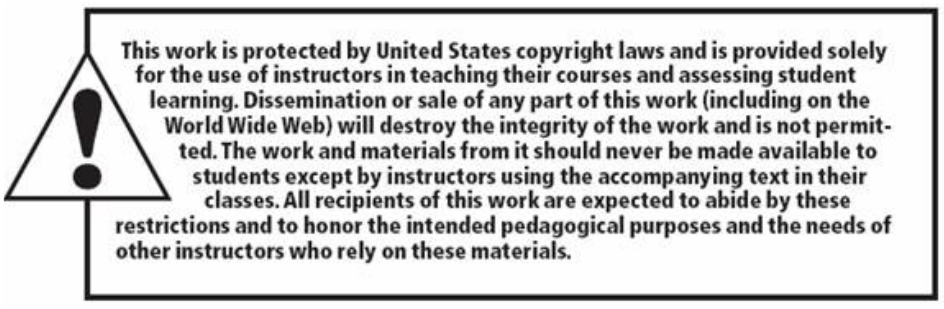
Introduction to Structured Query Language



Prepared By

Scott L. Vandenberg

Siena College



Instructor's Manual to accompany:

Database Processing: Fundamental, Design, and Implementation (14th Edition)

David M. Kroenke and David J. Auer

CHAPTER OBJECTIVES

- To understand the use of extracted data sets in business intelligence (BI) systems
- To understand the use of ad-hoc queries in business intelligence (BI) systems
- To understand the history and significance of Structured Query Language (SQL)
- To understand the SQL SELECT/FROM/WHERE framework as the basis for database queries
- To create SQL queries to retrieve data from a single table
- To create SQL queries that use the SQL SELECT, FROM, WHERE, ORDER BY, GROUP BY, and HAVING clauses
- To create SQL queries that use the SQL DISTINCT, TOP, and TOP PERCENT keywords
- To create SQL queries that use the SQL comparison operators including BETWEEN, LIKE, IN, and IS NULL
- To create SQL queries that use the SQL logical operators including AND, OR, and NOT
- To create SQL queries that use the SQL built-in aggregate functions of SUM, COUNT, MIN, MAX, and AVG with and without the SQL GROUP BY clause
- To create SQL queries that retrieve data from a single table while restricting the data based upon data in another table (subquery)
- To create SQL queries that retrieve data from multiple tables using the SQL join and JOIN ON operations
- To create SQL queries that retrieve data from multiple tables using the SQL OUTER JOIN operation
- To create SQL queries that retrieve data from multiple tables using SQL set operators UNION, INTERSECT, and EXCEPT

IMPORTANT TEACHING NOTES – READ THIS FIRST!

1. **Chapter 2 – Introduction to Structured Query Language** is intended to be taught in conjunction with the version of online Chapter 10# available at http://www.pearsonhighered.com/kroenke/ that corresponds to the DBMS that you are using in your class.
   a. If you are using **Microsoft SQL Server 2014** as your DBMS, you should use **Online Chapter 10A – Managing Databases with Microsoft SQL Server 2014**, and cover **pages 10A-1 through 10A-23** to help your students get set up for the SQL work in Chapter 2.
   b. If you are using **Oracle Database 12c** or **Oracle Database XE** as your DBMS, you should use **Online Chapter 10B – Managing Databases**

with Oracle Database, and cover **pages 10B-1 through 10BA-23** to help your students get set up for the SQL work in Chapter 2.

c. If you are using **MySQL 5.6** as your DBMS, you should use **Online Chapter 10C – Managing Databases with MySQL 5.6**, and cover **pages 10C-1 through 10C-28** to help your students get set up for the SQL work in Chapter 2.

d. These pages cover how to build a database from existing *.sql scripts, and the ***.sql scripts for the Cape Codd database** used in Chapter 2 are included in the **student data files** available at http://www.pearsonhighered.com/kroenke/.

ERRATA

- Page 70 – [27-JUL-15 – Corrected in the Instructor's Manual for Chapter 2] – Query labelled 18 on this page should be 22. On line 4:

```
/* *** SQL-Query-CH02-22 *** */
```

- Page 114 – [27-JUL-15 – Corrected in the Instructor's Manual for Chapter 2] – Review Question 2.59, last two words are redundant and should be removed:

2.59 Write an SQL statement to display the SKU, SKU_Description, and Department of all SKUs that appear in *both* the Cape Codd 2013 catalog (only in the printed catalog itself) *and* the Cape Codd 2014 catalog (only in the printed catalog itself).

- Page 83 – [27-JUL-15 – Corrected in the Instructor's Manual for Chapter 2] – Figure 2.27, bottom blue box, "Water Spots" should be:

Water Sports

- Page 132 – [27-JUL-15 – Corrected in the Instructor's Manual for Chapter 2] – Case Question MI.J, LocalCurrencyAmountt is misspelled:

J. Show ItemID, Description, Store, and a calculated column named USCurrencyAmount that is equal to LocalCurrencyAmount multiplied by the ExchangeRate for all rows of ITEM.

- Page 104 – [27-JUL-15 – Corrected in the Instructor's Manual for Chapter 2] – Microsoft Access also does not support the INTERSECT operation. Sentence before Query 77, parenthesized comment should read:

(note that MySQL and Microsoft Access do not support this operator)

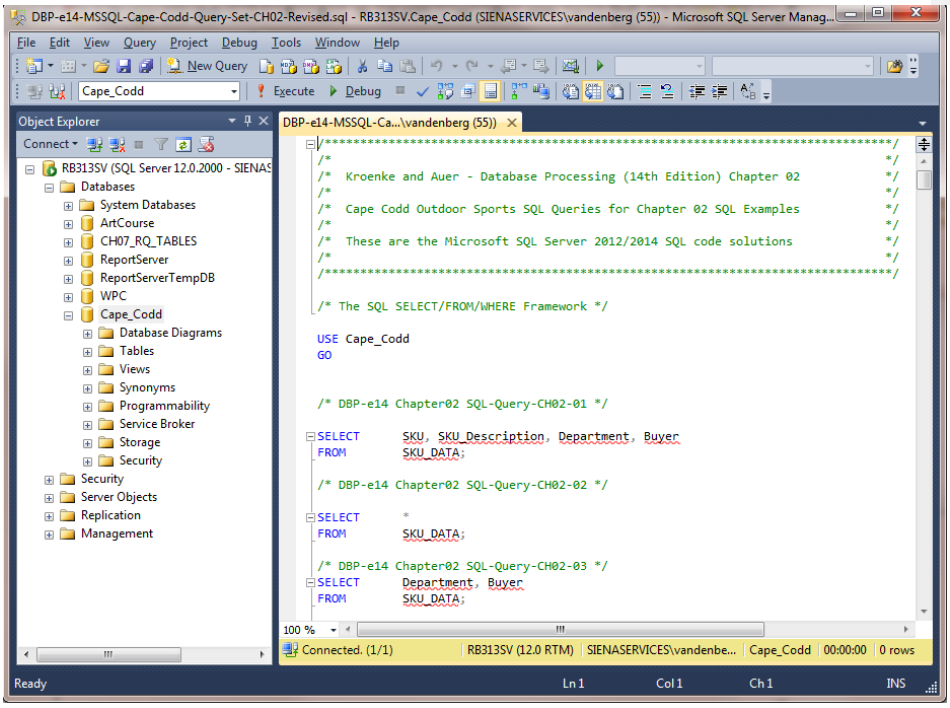
- Page 105 – [27-JUL-15 – Corrected in the Instructor's Manual for Chapter 2] – Microsoft Access also does not support the EXCEPT operation. Sentence before Query 78, parenthesized comment should read:

(note that Oracle Database calls this the SQL MINUS operator, and MySQL and Microsoft Access do not support this operation)

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, Queen Anne Curiosity Shop, or Morgan Importing Project Questions in an actual database. Students can create databases in Microsoft Access with basic tables, relationships, and data from the material in the book. SQL scripts for Microsoft SQL Server, Oracle Database, and MySQL versions of Cape Codd, MDC, QACS, and MI are available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke). An Access version of WPC is also available there.
- Microsoft Access database files for Cape Codd, together with SQL scripts for Microsoft SQL Server, Oracle Database, and MySQL versions of Cape Codd, MDC, QACS, and MI are available for student use in the 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 we've also had problems with the minus sign. If your students are having problems getting a "properly structured SQL statement" to run, look closely for this type of problem. It occurs most frequently when copying/pasting a query from a word processor into a query window.
- There is a useful teaching technique which will allow you to demonstrate the SQL queries in the text using Microsoft SQL Server if you have it available.
    - Open the Microsoft SQL Server Management Studio, and create a new SQL Server database named Cape-Codd.
    - In the Microsoft SQL Server Management Studio, use the SQL statements in the *.sql text file *DBP-e14-MSSQL-Cape-Codd-Create-Tables.sql* to create the RETAIL_ORDER, ORDER_ITEM, and SKU_DATA tables [other tables are also created].
    - In the Microsoft SQL Server Management Studio, use the SQL statements in the *.sql text file *DBP-e14-MSSQL-Cape-Codd-Insert-Data.sql* to populate the RETAIL_ORDER, ORDER_ITEM, and SKU_DATA tables [other tables are also populated].
    - In the Microsoft SQL Server Management Studio, open the *.sql text file *DBP-e14-MSSQL-Cape-Codd-Query-Set-CH02.sql*. This file contains all the queries shown in the Chapter 2 text.
    - Highlight the query you want to run and click the Execute Query button to display the results of the query. An example of this is shown in the following screenshot.

- 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 2013 does not support all SQL-92 (and newer) constructs. While this chapter still considers Microsoft 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 2.36 – 2.38). The correct solutions for these questions were obtained using Microsoft SQL Server 2014. The Microsoft Access results achieving the ORDER BY without using the alias are also shown, so you can assign these problems with or without the ORDER BY part of the questions.
- Microsoft Access 2013 does not support SQL wildcard characters (see Review Questions 2.31 – 2.33), although it does have equivalent wildcard characters as described in the chapter. The correct solutions for these questions were obtained using Microsoft SQL Server 2014, and solutions are shown for Access as well.
- For those students who are used to procedural languages, they may have some initial difficulty with a language that 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 may 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 likely 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 Microsoft Access using Microsoft 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 a Microsoft Access query.
- It is also important for students to understand that, in many cases, the Query-By-Example forms such as Microsoft Access's design view can be very inefficient. Also, the QBE forms are not available from within an application program such as Java or C++ or PHP, and 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 = 1,000,000, 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, where in some previous editions we have used tables named ORDER and ORDER_ITEM, we have changed these table names to INVOICE and INVOICE_ITEM. We did this 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 2013) 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 7 and 8. However, now is a good time to introduce it to your students.
- Note that Microsoft Access SQL requires the INNER JOIN syntax instead of the standard SQL syntax JOIN used by Microsoft SQL Server, Oracle Database, and MySQL. Also note that Oracle prohibits the "AS" keyword when aliasing table names using the JOIN syntax. See solutions to Review Question 51.
- Students will frequently try to UNION OR INTERSECT tables that are not compatible (have different schemas). It is useful to illustrate a few examples of how/why this doesn't work (e.g. try UNIONing RETAIL_ORDER and

ORDER_ITEM to answer the English query “Give me all orders and their items” to distinguish this from a join).

- String comparisons using LIKE (and other operators) may or may not be case-sensitive, depending on the DBMS used and on the default settings set up by the DBA; see solutions to Case Question MDC-F for more details and suggestions.
- Screen shot solutions to all the queries in this chapter come from Microsoft Access. Note that some of them are from Access 2010 and some from Access 2013: the differences for the purposes of this chapter are entirely cosmetic (font and other colors).

ANSWERS TO REVIEW QUESTIONS

2.1 What is an online transaction processing (OLTP) system? What is a business intelligence (BI) system? What is a data warehouse?

An OLTP system is typically one in which a database is used to store information about daily operational aspects of a business or other enterprise, such as sales, deposits, orders, customers, etc. A business intelligence (BI) system is a system used to support management decisions by producing information for assessment, analysis, planning and control. BI systems typically use data from a data warehouse, which is a database typically combining information from operational databases, other relevant internal data, and separately-purchased external data.

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? 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 were 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 [04/26/2013]) 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 [2013]).

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. What is the relationship of these tables to the CATALOG_SKU_2014 and CATALOG_SKU_2015 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). The two CATALOG tables are not formally related to any of the other tables.

Using the Microsoft Access Relationship window, the relationships are shown in Figure 2-4 and look like this:

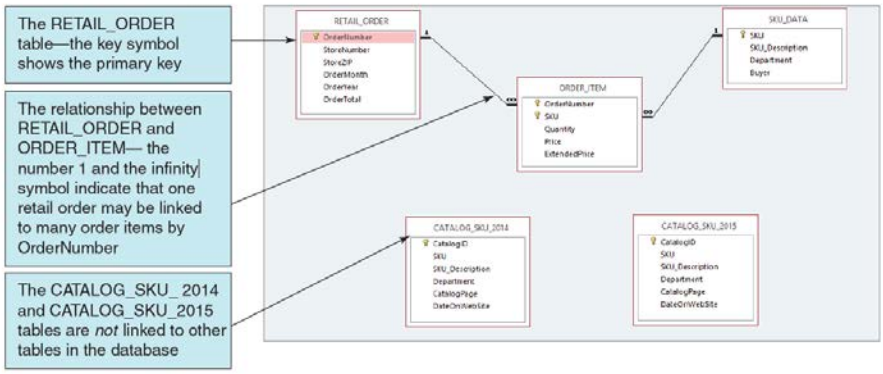


Figure 2-4 – The Cape Codd Database

In traditional database terms (which will be discussed in Chapter 3) 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 relational database management systems. The SQL statements in this chapter are based on SQL-92 and the SQL standards that followed and modified it.

2.9 What features have been added to SQL in versions subsequent to 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.

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 Microsoft Access uses SQL.

Microsoft Access uses SQL, but generally hides the SQL from the user. For example, Microsoft Access automatically generates SQL and sends it to Microsoft Access's internal Access Database Engine (ADE, which is a variant of the Microsoft Jet engine) 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 Microsoft Access. Queries in Access are by default created using the GUI QBE interface, then translated into SQL for processing. One can also create SQL queries directly in Access, bypassing QBE if desired.

2.15 Explain how enterprise-class DBMS products use SQL.

Enterprise-class DBMS products, which include Microsoft SQL Server, Oracle Corporation's Oracle Database and MySQL, and IBM's DB2, 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 three additional tables: the INVENTORY table, the WAREHOUSE table, and the CATALOG_SKU_2013 table. The table schemas for these tables, RETAIL_ORDER, ORDER_ITEM, SKU_DATA, CATALOG_SKU_2014, and CATALOG_SKU_2015 tables, are as follows:

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

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

SKU_DATA (SKU, SKU_Description, Department, Buyer)

WAREHOUSE (WarehouseID, WarehouseCity, WarehouseState, Manager, Squarefeet)

INVENTORY (*WarehouseID*, *SKU*, SKU_Description, QuantityOnHand, QuantityOnOrder)

CATALOG_SKU_2013 (CatalogID, SKU, SKU_Description, CatalogPage, DateOnWebSite)

CATALOG_SKU_2014 (CatalogID, SKU, SKU_Description, CatalogPage, DateOnWebSite)

CATALOG_SKU_2015 (CatalogID, SKU, SKU_Description, CatalogPage, DateOnWebSite)

The eight tables in the revised Cape Codd database schema are shown in Figure 2-34. The column characteristics for the WAREHOUSE table are shown in Figure 2-35, the column characteristics for the INVENTORY table are shown in Figure 2-36, and the column characteristics for the CATALOG_SKU_2013 table are shown in Figure 2-37. The data for the WAREHOUSE table are shown in Figure 2-38, the data for the INVENTORY table are shown in Figure 2-39, and the data for the CATALOG_SKU_2013 table are shown in Figure 2-40.

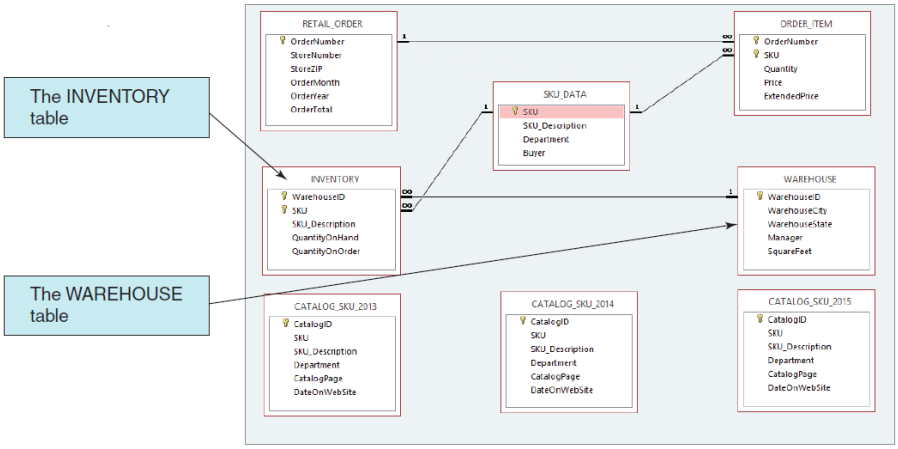


Figure 2-34 – The Cape Codd Database with the WAREHOUSE, INVENTORY, and CATALOG_SKU_2013 tables

WAREHOUSE

Column Name	Type	Key	Required	Remarks
WarehouseID	Integer	Primary Key	Yes	Surrogate Key
WarehouseCity	Character (30)	No	Yes	
WarehouseState	Character (2)	No	Yes	
Manager	Character (35)	No	No	
SquareFeet	Integer	No	No	

Figure 2-35 - Column Characteristics for the WAREHOUSE Table

INVENTORY

Column Name	Type	Key	Required	Remarks
WarehouseID	Integer	Primary Key, Foreign Key	Yes	REF: WAREHOUSE
SKU	Integer	Primary Key, Foreign Key	Yes	REF: SKU_DATA
SKU_Description	Character (35)	No	Yes	
QuantityOnHand	Integer	No	No	
QuantityOnOrder	Integer	No	No	

Figure 2-36 - Column Characteristics for the INVENTORY Table

CATALOG_SKU_2013

Column Name	Type	Key	Required	Remarks
CatalogID	Integer	Primary Key	Yes	Surrogate Key
SKU	Integer	No	Yes	
SKU_Description	Character (35)	No	Yes	
Department	Character (30)	No	Yes	
CatalogPage	Integer	No	No	
DateOnWebPage	Date	No	No	

Figure 2-37 - Column Characteristics for the CATALOG_SKU_2013 Table

WarehouseID	WarehouseCity	WarehouseState	Manager	SquareFeet
100	Atlanta	GA	Dave Jones	125,000
200	Chicago	IL	Lucille Smith	100,000
300	Bangor	ME	Bart Evans	150,000
400	Seattle	WA	Dale Rogers	130,000
500	San Francisco	CA	Grace Jefferson	200,000

Figure 2-38 - Cape Codd Database WAREHOUSE Table Data

WarehouseID	SKU	SKU_Description	QuantityOnHand	QuantityOnOrder
100	100100	Std. Scuba Tank, Yellow	250	0
200	100100	Std. Scuba Tank, Yellow	100	50
300	100100	Std. Scuba Tank, Yellow	100	0
400	100100	Std. Scuba Tank, Yellow	200	0
100	100200	Std. Scuba Tank, Magenta	200	30
200	100200	Std. Scuba Tank, Magenta	75	75
300	100200	Std. Scuba Tank, Magenta	100	100
400	100200	Std. Scuba Tank, Magenta	250	0
100	101100	Dive Mask, Small Clear	0	500
200	101100	Dive Mask, Small Clear	0	500
300	101100	Dive Mask, Small Clear	300	200
400	101100	Dive Mask, Small Clear	450	0
100	101200	Dive Mask, Med Clear	100	500
200	101200	Dive Mask, Med Clear	50	500
300	101200	Dive Mask, Med Clear	475	0
400	101200	Dive Mask, Med Clear	250	250
100	201000	Half-Dome Tent	2	100
200	201000	Half-Dome Tent	10	250
300	201000	Half-Dome Tent	250	0
400	201000	Half-Dome Tent	0	250
100	202000	Half-Dome Tent Vestibule	10	250
200	202000	Half-Dome Tent Vestibule	1	250
300	202000	Half-Dome Tent Vestibule	100	0
400	202000	Half-Dome Tent Vestibule	0	200
100	301000	Light Fly Climbing Harness	300	250
200	301000	Light Fly Climbing Harness	250	250
300	301000	Light Fly Climbing Harness	0	250
400	301000	Light Fly Climbing Harness	0	250
100	302000	Locking Carabiner, Oval	1000	0
200	302000	Locking Carabiner, Oval	1250	0
300	302000	Locking Carabiner, Oval	500	500
400	302000	Locking Carabiner, Oval	0	1000

Figure 2-39 - Cape Codd Database INVENTORY Table Data

CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
20130001	100100	Std. Scuba Tank, Yellow	Water Sports	23	2013-01-01
20130002	100500	Std. Scuba Tank, Light Green	Water Sports	NULL	2013-07-01
20130003	100600	Std. Scuba Tank, Dark Green	Water Sports	NULL	2013-07-01
20130004	101100	Dive Mask, Small Clear	Water Sports	24	2013-01-01
20130005	101200	Dive Mask, Med Clear	Water Sports	24	2013-01-01
20130006	201000	Half-dome Tent	Camping	45	2013-01-01
20130007	202000	Half-dome Tent Vestibule	Camping	47	2013-01-01
20130008	301000	Light Fly Climbing Harness	Climbing	76	2013-01-01
20130009	302000	Locking Carabiner, Oval	Climbing	78	2013-01-01

Figure 2-40 - Cape Codd Database CATALOG_SKU_2013 Table Data

You will need to create and setup a database named Cape_Codd for use with the Cape Codd review questions. You may have already created this database as suggested in Chapter 2 and used it to run the SQL queries discussed in the chapter. If you haven't, you need to do so now.

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 Microsoft SQL Server, Oracle Database, and MySQL.

If you are using the Microsoft Access 2013 Cape_Codd.accdb database, simply copy it to an appropriate location in your Documents folder. Otherwise, you will need to use the discussion and instructions necessary for setting up the Cape_Codd database in the DBMS product you are using:

- *For Microsoft SQL Server 2014, see online Chapter 10A.*
- *For Oracle Database 12c or Oracle Express Edition 11g Release 2, see online Chapter 10B.*
- *For MySQL 5.6 Community Server, see online Chapter 10C.*

Once you have setup your Cape_Codd database, create an SQL script named Cape-Codd-CH02-RQ.sql, and use it to record and store SQL statements that answer each of the following questions (if the question requires a written answer, use an SQL comment to record your answer):

NOTE: All answers below show the correct SQL statement, as well as SQL statements modified for Microsoft Access 2013 when needed. Whenever possible, all results were obtained by

running the SQL statements in Microsoft Access 2013, and the corresponding screen shots of the results are shown below. As explained in the text, some queries cannot be run in Microsoft Access 2013, and for those queries the correct result was obtained using Microsoft SQL Server 2014. The SQL statements shown should run with little, if any, modification needed for Oracle Database 12c, Oracle Database Express Edition 11g R2, and MySQL 5.6.

Solutions to Review Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke). Solutions in SQL Server, Oracle, and MySQL are also available at the same site.

If your students are using a DBMS other than Microsoft Access, the SQL code to create and populate the Cape Codd database is available in the *.sql script files for SQL Server 2014, Oracle Database 12c/Express Edition 11gR2, and MySQL 5.6 in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

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 2.39:

2.17 Write an SQL statement to display SKU and SKU_Description.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

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

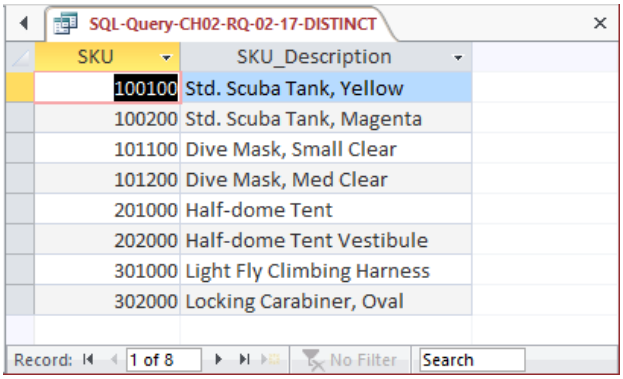
SQL-Query-CH02-RQ-02-17

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

Record: 1 of 32 No Filter Search

The question does not ask for unique SKU and SKU_Description data, but could be obtained by using:

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



2.18 Write an SQL statement to display SKU_Description and SKU.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

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

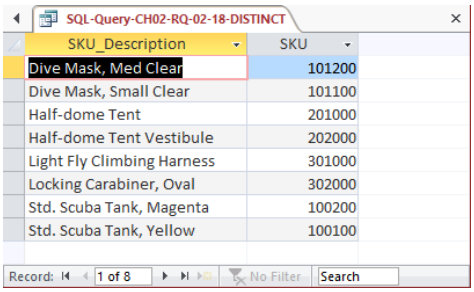
SQL-Query-CH02-RQ-02-18

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

Record: 1 of 32 No Filter Search

The question does not ask for unique SKU and SKU_Description data, but could be obtained by using:

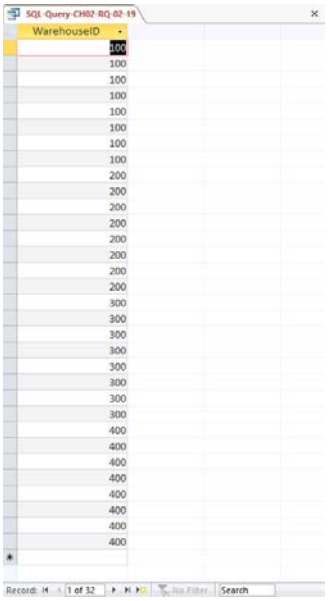
```
SELECT   UNIQUE SKU_Description, SKU
FROM     INVENTORY;
```



2.19 Write an SQL statement to display WarehouseID.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

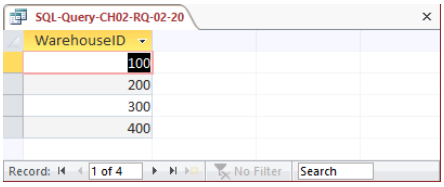
```
SELECT   WarehouseID
FROM     INVENTORY;
```



2.20 Write an SQL statement to display unique WarehouseIDs.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   DISTINCT WarehouseID
FROM     INVENTORY;
```



2.21 Write an SQL statement to display all of the columns without using the SQL asterisk (*) wildcard character.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

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

SQL-Query-CH02-RQ-02-21

WarehouseID	SKU	SKU_Description	QuantityOnHand	QuantityOnOrder
100	100100	Std. Scuba Tank, Yellow	250	0
100	100200	Std. Scuba Tank, Magenta	200	30
100	101100	Dive Mask, Small Clear	0	500
100	101200	Dive Mask, Med Clear	100	500
100	201000	Half-dome Tent	2	100
100	202000	Half-dome Tent Vestibule	10	250
100	301000	Light Fly Climbing Harness	300	250
100	302000	Locking Carabiner, Oval	1000	0
200	100100	Std. Scuba Tank, Yellow	100	50
200	100200	Std. Scuba Tank, Magenta	75	75
200	101100	Dive Mask, Small Clear	0	500
200	101200	Dive Mask, Med Clear	50	500
200	201000	Half-dome Tent	10	250
200	202000	Half-dome Tent Vestibule	1	250
200	301000	Light Fly Climbing Harness	250	250
200	302000	Locking Carabiner, Oval	1250	0
300	100100	Std. Scuba Tank, Yellow	100	0
300	100200	Std. Scuba Tank, Magenta	100	100
300	101100	Dive Mask, Small Clear	300	200
300	101200	Dive Mask, Med Clear	475	0
300	201000	Half-dome Tent	250	0
300	202000	Half-dome Tent Vestibule	100	0
300	301000	Light Fly Climbing Harness	0	250
300	302000	Locking Carabiner, Oval	500	500
400	100100	Std. Scuba Tank, Yellow	200	0
400	100200	Std. Scuba Tank, Magenta	250	0
400	101100	Dive Mask, Small Clear	450	0
400	101200	Dive Mask, Med Clear	250	250
400	201000	Half-dome Tent	0	250
400	202000	Half-dome Tent Vestibule	0	200
400	301000	Light Fly Climbing Harness	0	250
400	302000	Locking Carabiner, Oval	0	1000

Record: 1 of 32 No Filter Search

2.22 Write an SQL statement to display all of the columns using the SQL asterisk (*) wildcard character.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   *
FROM     INVENTORY;
```

SQL-Query-CH02-RQ-02-22

WarehouseID	SKU	SKU_Description	QuantityOnHand	QuantityOnOrder
100	100100	Std. Scuba Tank, Yellow	250	0
100	100200	Std. Scuba Tank, Magenta	200	30
100	101100	Dive Mask, Small Clear	0	500
100	101200	Dive Mask, Med Clear	100	500
100	201000	Half-dome Tent	2	100
100	202000	Half-dome Tent Vestibule	10	250
100	301000	Light Fly Climbing Harness	300	250
100	302000	Locking Carabiner, Oval	1000	0
200	100100	Std. Scuba Tank, Yellow	100	50
200	100200	Std. Scuba Tank, Magenta	75	75
200	101100	Dive Mask, Small Clear	0	500
200	101200	Dive Mask, Med Clear	50	500
200	201000	Half-dome Tent	10	250
200	202000	Half-dome Tent Vestibule	1	250
200	301000	Light Fly Climbing Harness	250	250
200	302000	Locking Carabiner, Oval	1250	0
300	100100	Std. Scuba Tank, Yellow	100	0
300	100200	Std. Scuba Tank, Magenta	100	100
300	101100	Dive Mask, Small Clear	300	200
300	101200	Dive Mask, Med Clear	475	0
300	201000	Half-dome Tent	250	0
300	202000	Half-dome Tent Vestibule	100	0
300	301000	Light Fly Climbing Harness	0	250
300	302000	Locking Carabiner, Oval	500	500
400	100100	Std. Scuba Tank, Yellow	200	0
400	100200	Std. Scuba Tank, Magenta	250	0
400	101100	Dive Mask, Small Clear	450	0
400	101200	Dive Mask, Med Clear	250	250
400	201000	Half-dome Tent	0	250
400	202000	Half-dome Tent Vestibule	0	200
400	301000	Light Fly Climbing Harness	0	250
400	302000	Locking Carabiner, Oval	0	1000

Record: 1 of 32 No Filter Search

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

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

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

SQL-Query-CH02-RQ-02-23

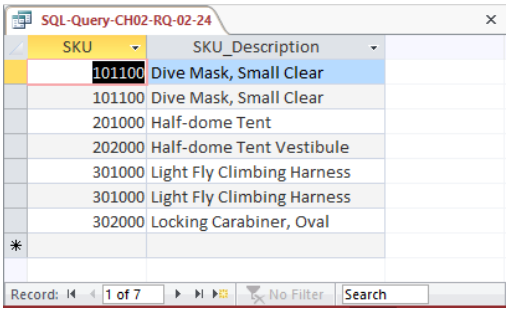
WarehouseID	SKU	SKU_Description	QuantityOnHand	QuantityOnOrder
100	100100	Std. Scuba Tank, Yellow	250	0
200	100100	Std. Scuba Tank, Yellow	100	50
300	100100	Std. Scuba Tank, Yellow	100	0
400	100100	Std. Scuba Tank, Yellow	200	0
100	100200	Std. Scuba Tank, Magenta	200	30
200	100200	Std. Scuba Tank, Magenta	75	75
300	100200	Std. Scuba Tank, Magenta	100	100
400	100200	Std. Scuba Tank, Magenta	250	0
300	101100	Dive Mask, Small Clear	300	200
400	101100	Dive Mask, Small Clear	450	0
100	101200	Dive Mask, Med Clear	100	500
200	101200	Dive Mask, Med Clear	50	500
300	101200	Dive Mask, Med Clear	475	0
400	101200	Dive Mask, Med Clear	250	250
100	201000	Half-dome Tent	2	100
200	201000	Half-dome Tent	10	250
300	201000	Half-dome Tent	250	0
100	202000	Half-dome Tent Vestibule	10	250
200	202000	Half-dome Tent Vestibule	1	250
300	202000	Half-dome Tent Vestibule	100	0
100	301000	Light Fly Climbing Harness	300	250
200	301000	Light Fly Climbing Harness	250	250
100	302000	Locking Carabiner, Oval	1000	0
200	302000	Locking Carabiner, Oval	1250	0
300	302000	Locking Carabiner, Oval	500	500

Record: 1 of 25 No Filter Search

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

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

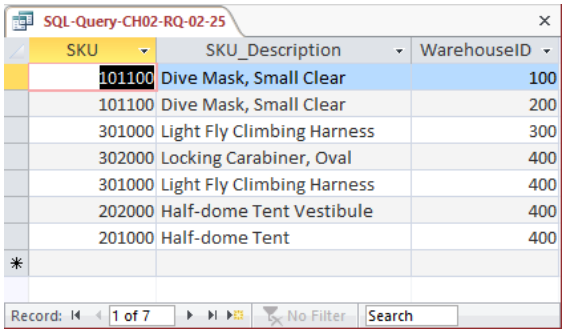
```
SELECT   SKU, SKU_Description
FROM     INVENTORY
WHERE    QuantityOnHand =0;
```



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

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

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



2.26 Write an SQL statement to display the SKU, SKU_Description, and WarehouseID for products having QuantityOnHand greater than 0. Sort the results in descending order by WarehouseID and ascending order by SKU.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT      SKU, SKU_Description, WarehouseID
FROM        INVENTORY
WHERE       QuantityOnHand > 0
ORDER BY    WarehouseID DESC, SKU;
```

SQL-Query-CH02-RQ-02-26

SKU	SKU_Description	WarehouseID
100100	Std. Scuba Tank, Yellow	400
100200	Std. Scuba Tank, Magenta	400
101100	Dive Mask, Small Clear	400
101200	Dive Mask, Med Clear	400
100100	Std. Scuba Tank, Yellow	300
100200	Std. Scuba Tank, Magenta	300
101100	Dive Mask, Small Clear	300
101200	Dive Mask, Med Clear	300
201000	Half-dome Tent	300
202000	Half-dome Tent Vestibule	300
302000	Locking Carabiner, Oval	300
100100	Std. Scuba Tank, Yellow	200
100200	Std. Scuba Tank, Magenta	200
101200	Dive Mask, Med Clear	200
201000	Half-dome Tent	200
202000	Half-dome Tent Vestibule	200
301000	Light Fly Climbing Harness	200
302000	Locking Carabiner, Oval	200
100100	Std. Scuba Tank, Yellow	100
100200	Std. Scuba Tank, Magenta	100
101200	Dive Mask, Med Clear	100
201000	Half-dome Tent	100
202000	Half-dome Tent Vestibule	100
301000	Light Fly Climbing Harness	100
302000	Locking Carabiner, Oval	100

Record: 1 of 25 No Filter Search

2.27 Write an SQL statement to display SKU, SKU_Description, and WarehouseID for all products that have a QuantityOnHand equal to 0 and a QuantityOnOrder greater than 0. Sort the results in descending order by WarehouseID and in ascending order by SKU.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT      SKU, SKU_Description, WarehouseID
FROM        INVENTORY
WHERE       QuantityOnHand = 0
   AND      QuantityOnOrder > 0
ORDER BY    WarehouseID DESC, SKU;
```

SQL-Query-CH02-RQ-02-27

SKU	SKU_Description	WarehouseID
201000	Half-dome Tent	400
202000	Half-dome Tent Vestibule	400
301000	Light Fly Climbing Harness	400
302000	Locking Carabiner, Oval	400
301000	Light Fly Climbing Harness	300
101100	Dive Mask, Small Clear	200
101100	Dive Mask, Small Clear	100

Record: 1 of 7 No Filter Search

2.28 Write an SQL statement to display SKU, SKU_Description, and WarehouseID for all products that have a QuantityOnHand equal to 0 or a QuantityOnOrder equal to 0. Sort the results in descending order by WarehouseID and in ascending order by SKU.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT      SKU, SKU_Description, WarehouseID
FROM        INVENTORY
WHERE       QuantityOnHand = 0
   OR       QuantityOnOrder = 0
ORDER BY    WarehouseID DESC, SKU;
```

SQL-Query-CH02-RQ-02-28

SKU	SKU_Description	WarehouseID
100100	Std. Scuba Tank, Yellow	400
100200	Std. Scuba Tank, Magenta	400
101100	Dive Mask, Small Clear	400
201000	Half-dome Tent	400
202000	Half-dome Tent Vestibule	400
301000	Light Fly Climbing Harness	400
302000	Locking Carabiner, Oval	400
100100	Std. Scuba Tank, Yellow	300
101200	Dive Mask, Med Clear	300
201000	Half-dome Tent	300
202000	Half-dome Tent Vestibule	300
301000	Light Fly Climbing Harness	300
101100	Dive Mask, Small Clear	200
302000	Locking Carabiner, Oval	200
100100	Std. Scuba Tank, Yellow	100
101100	Dive Mask, Small Clear	100
302000	Locking Carabiner, Oval	100

Record: 1 of 17 No Filter Search

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

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT      SKU, SKU_Description, WarehouseID, QuantityOnHand
FROM        INVENTORY
WHERE       QuantityOnHand > 1
   AND      QuantityOnhand < 10;
```

SQL-Query-CH02-RQ-02-29

SKU	SKU_Description	WarehouseID	QuantityOnHand
201000	Half-dome Tent	100	2

Record: 1 of 1 No Filter Search

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

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

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

SQL-Query-CH02-RQ-02-30

SKU	SKU_Description	WarehouseID	QuantityOnHand
201000	Half-dome Tent	100	2

Record: 1 of 1 No Filter Search

2.31 Write an SQL statement to show a unique SKU and SKU_Description for all products having an SKU description starting with 'Half-dome'.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that, as discussed in Chapter 2, Microsoft Access 2013 uses wildcard characters that differ from the SQL standard.

For Microsoft SQL Server, Oracle Database, and MySQL:

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

For Microsoft Access:

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

SQL-Query-CH02-RQ-02-31

SKU	SKU_Description
201000	Half-dome Tent
202000	Half-dome Tent Vestibule

Record: 1 of 2 No Filter Search

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

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

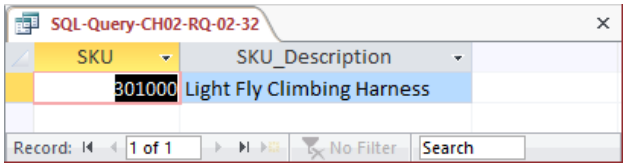
Note that, as discussed in Chapter 2, Microsoft Access 2013 uses wildcard characters that differ from the SQL standard.

For Microsoft SQL Server, Oracle Database, and MySQL:

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

For Microsoft Access:

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



2.33 Write an SQL statement to show a unique SKU and SKU_Description for all products having a 'd' in the third position from the left in SKU_Description.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

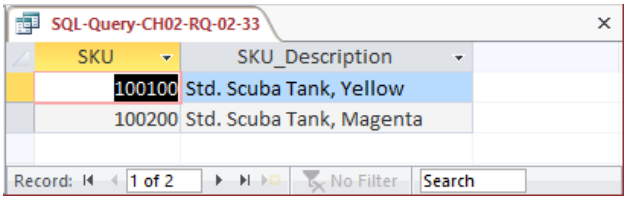
Note that, as discussed in Chapter 2, Microsoft Access 2013 uses wildcard characters that differ from the SQL standard.

For Microsoft SQL Server, Oracle Database, and MySQL:

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

For Microsoft Access:

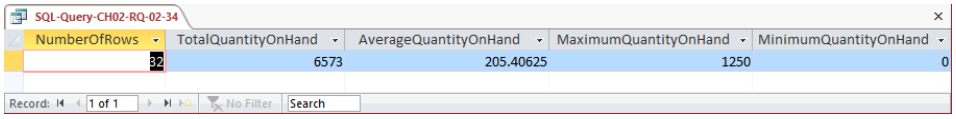
```
SELECT   DISTINCT SKU, SKU_Description
FROM     INVENTORY
WHERE    SKU_Description LIKE '??d*';
```



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

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    COUNT(QuantityOnHand) AS NumberOfRows,
          SUM(QuantityOnHand) AS TotalQuantityOnHand,
          AVG(QuantityOnHand) AS AverageQuantityOnHand,
          MAX(QuantityOnHand) AS MaximumQuantityOnHand,
          MIN(QuantityOnHand) AS MinimumQuantityOnHand
FROM      INVENTORY;
```



2.35 Explain the difference between the SQL built-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.

2.36 Write an SQL statement to display the WarehouseID and the sum of QuantityOnHand, grouped by WarehouseID. Name the sum TotalItemsOnHand and display the results in descending order of TotalItemsOnHand.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

For Microsoft SQL Server, Oracle Database, and MySQL:

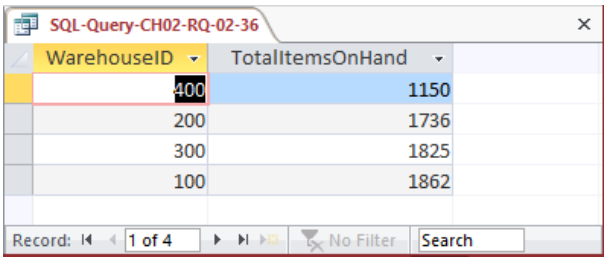
```
SELECT      WarehouseID, SUM(QuantityOnHand) AS TotalItemsOnHand
FROM        INVENTORY
GROUP BY    WarehouseID
ORDER BY    TotalItemsOnHand DESC;
```

For Microsoft Access:

Unfortunately, Microsoft Access cannot process the ORDER BY clause because it contains an aliased computed result. To correct this, we use an SQL statement with the un-aliased computation:

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

The results, presented below in Access, are identical in all 4 DBMSs:



2.37 Write an SQL statement to display the WarehouseID and the sum of QuantityOnHand, grouped by WarehouseID. Omit all SKU items that have 3 or more items on hand from the sum, and name the sum TotalItemsOnHandLT3 and display the results in descending order of TotalItemsOnHandLT3.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

For Microsoft SQL Server, Oracle Database, and MySQL:

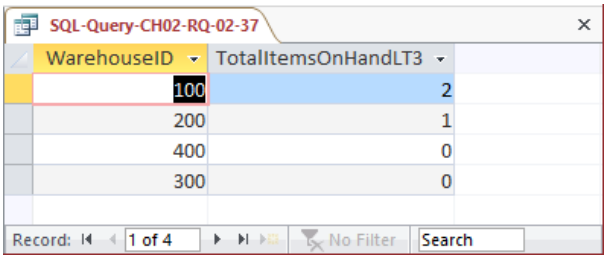
```
SELECT      WarehouseID, SUM(QuantityOnHand) AS TotalItemsOnHandLT3
FROM        INVENTORY
WHERE       QuantityOnHand < 3
GROUP BY    WarehouseID
ORDER BY    TotalItemsOnHandLT3 DESC;
```

For Microsoft Access:

Unfortunately, Microsoft Access cannot process the ORDER BY clause because it contains an aliased computed result. To correct this, we use an SQL statement with the un-aliased computation:

```
SELECT      WarehouseID, SUM(QuantityOnHand) AS TotalItemsOnHandLT3
FROM        INVENTORY
WHERE       QuantityOnHand < 3
GROUP BY    WarehouseID
ORDER BY    SUM(QuantityOnHand) DESC;
```

The results, presented below in Access, are identical in all 4 DBMSs:



2.38 Write an SQL statement to display the WarehouseID and the sum of QuantityOnHand grouped by WarehouseID. Omit all SKU items that have 3 or more items on hand from the sum, and name the sum TotalItemsOnHandLT3. Show Warehouse ID only for warehouses having fewer than 2 SKUs in their TotalItemsOnHandLT3. Display the results in descending order of TotalItemsOnHandLT3.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

For Microsoft SQL Server, Oracle Database and MySQL:

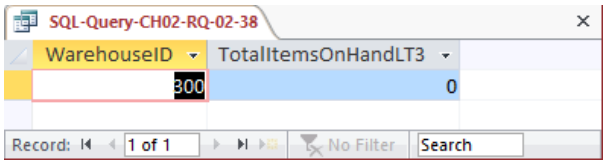
```
SELECT      WarehouseID, SUM(QuantityOnHand) AS TotalItemsOnHandLT3
FROM        INVENTORY
WHERE       QuantityOnHand < 3
GROUP BY    WarehouseID
HAVING      COUNT(*) < 2
ORDER BY    TotalItemsOnHandLT3 DESC;
```

For Microsoft Access:

Unfortunately, Microsoft Access cannot process the ORDER BY clause because it contains an aliased computed result. To correct this, we use an SQL statement with the un-aliased computation:

```
SELECT      WarehouseID, SUM(QuantityOnHand) AS TotalItemsOnHandLT3
FROM        INVENTORY
WHERE       QuantityOnHand < 3
GROUP BY    WarehouseID
HAVING      COUNT(*) < 2
ORDER BY    SUM(QuantityOnHand) DESC;
```

The results, presented below in Access, are identical in all 4 DBMSs:



2.39 In your answer to Review Question 2.38, 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.

Use both the INVENTORY and WAREHOUSE tables to answer Review Questions 2.40 through 2.52:

2.40 Write an SQL statement to display the SKU, SKU_Description, WarehouseID, WarehouseCity, and WarehouseState for all items stored in the Atlanta, Bangor, or Chicago warehouse. Do not use the IN keyword.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   SKU, SKU_Description,
         WAREHOUSE.WarehouseID, WarehouseCity, WarehouseState
FROM     INVENTORY, WAREHOUSE
WHERE    INVENTORY.WarehouseID=WAREHOUSE.WarehouseID
   AND   (WarehouseCity = 'Atlanta'
          OR   WarehouseCity = 'Bangor'
          OR   WarehouseCity = 'Chicago');
```

SQL-Query-CH02-RQ-02-40

SKU	SKU_Description	WarehouseID	WarehouseCity	WarehouseState
100100	Std. Scuba Tank, Yellow	100	Atlanta	GA
100200	Std. Scuba Tank, Magenta	100	Atlanta	GA
101100	Dive Mask, Small Clear	100	Atlanta	GA
101200	Dive Mask, Med Clear	100	Atlanta	GA
201000	Half-dome Tent	100	Atlanta	GA
202000	Half-dome Tent Vestibule	100	Atlanta	GA
301000	Light Fly Climbing Harness	100	Atlanta	GA
302000	Locking Carabiner, Oval	100	Atlanta	GA
100100	Std. Scuba Tank, Yellow	200	Chicago	IL
100200	Std. Scuba Tank, Magenta	200	Chicago	IL
101100	Dive Mask, Small Clear	200	Chicago	IL
101200	Dive Mask, Med Clear	200	Chicago	IL
201000	Half-dome Tent	200	Chicago	IL
202000	Half-dome Tent Vestibule	200	Chicago	IL
301000	Light Fly Climbing Harness	200	Chicago	IL
302000	Locking Carabiner, Oval	200	Chicago	IL
100100	Std. Scuba Tank, Yellow	300	Bangor	ME
100200	Std. Scuba Tank, Magenta	300	Bangor	ME
101100	Dive Mask, Small Clear	300	Bangor	ME
101200	Dive Mask, Med Clear	300	Bangor	ME
201000	Half-dome Tent	300	Bangor	ME
202000	Half-dome Tent Vestibule	300	Bangor	ME
301000	Light Fly Climbing Harness	300	Bangor	ME
302000	Locking Carabiner, Oval	300	Bangor	ME

Record: 1 of 24 No Filter Search

2.41 Write an SQL statement to display the SKU, SKU_Description, WarehouseID, WarehouseCity, and WarehouseState for all items stored in the Atlanta, Bangor, or Chicago warehouse. Use the IN keyword.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT    SKU, SKU_Description,
          WAREHOUSE.WarehouseID, WarehouseCity, WarehouseState
FROM      INVENTORY, WAREHOUSE
WHERE     INVENTORY.WarehouseID=WAREHOUSE.WarehouseID
   AND    WarehouseCity IN ('Atlanta', 'Bangor' ,'Chicago');
```

SQL-Query-CH02-RQ-02-41

SKU	SKU_Description	WarehouseID	WarehouseCity	WarehouseState
100100	Std. Scuba Tank, Yellow	100	Atlanta	GA
100200	Std. Scuba Tank, Magenta	100	Atlanta	GA
101100	Dive Mask, Small Clear	100	Atlanta	GA
101200	Dive Mask, Med Clear	100	Atlanta	GA
201000	Half-dome Tent	100	Atlanta	GA
202000	Half-dome Tent Vestibule	100	Atlanta	GA
301000	Light Fly Climbing Harness	100	Atlanta	GA
302000	Locking Carabiner, Oval	100	Atlanta	GA
100100	Std. Scuba Tank, Yellow	200	Chicago	IL
100200	Std. Scuba Tank, Magenta	200	Chicago	IL
101100	Dive Mask, Small Clear	200	Chicago	IL
101200	Dive Mask, Med Clear	200	Chicago	IL
201000	Half-dome Tent	200	Chicago	IL
202000	Half-dome Tent Vestibule	200	Chicago	IL
301000	Light Fly Climbing Harness	200	Chicago	IL
302000	Locking Carabiner, Oval	200	Chicago	IL
100100	Std. Scuba Tank, Yellow	300	Bangor	ME
100200	Std. Scuba Tank, Magenta	300	Bangor	ME
101100	Dive Mask, Small Clear	300	Bangor	ME
101200	Dive Mask, Med Clear	300	Bangor	ME
201000	Half-dome Tent	300	Bangor	ME
202000	Half-dome Tent Vestibule	300	Bangor	ME
301000	Light Fly Climbing Harness	300	Bangor	ME
302000	Locking Carabiner, Oval	300	Bangor	ME

Record: 1 of 24 No Filter Search

2.42 Write an SQL statement to display the SKU, SKU_Description, WarehouseID, WarehouseCity, and WarehouseState of all items not stored in the Atlanta, Bangor, or Chicago warehouse. Do not use the NOT IN keyword.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

NOTE: The symbol for "not equal to" is < >. Since we want the query output for warehouses that are not Atlanta or Bangor or Chicago as a set, we must ask for warehouses that are not in the group (Atlanta **and** Bangor **and** Chicago). 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,
         WAREHOUSE.WarehouseID, WarehouseCity, WarehouseState
FROM     INVENTORY, WAREHOUSE
```

```
WHERE    INVENTORY.WarehouseID=WAREHOUSE.WarehouseID
   AND   WarehouseCity <> 'Atlanta'
   AND   WarehouseCity <> 'Bangor'
   AND   WarehouseCity <> 'Chicago';
```

SQL-Query-CH02-RQ-02-42

SKU	SKU_Description	WarehouseID	WarehouseCity	WarehouseState
100100	Std. Scuba Tank, Yellow	400	Seattle	WA
100200	Std. Scuba Tank, Magenta	400	Seattle	WA
101100	Dive Mask, Small Clear	400	Seattle	WA
101200	Dive Mask, Med Clear	400	Seattle	WA
201000	Half-dome Tent	400	Seattle	WA
202000	Half-dome Tent Vestibule	400	Seattle	WA
301000	Light Fly Climbing Harness	400	Seattle	WA
302000	Locking Carabiner, Oval	400	Seattle	WA

Record: 1 of 8 No Filter Search

2.43 Write an SQL statement to display the SKU, SKU_Description, WarehouseID, WarehouseCity, and WarehouseState of all items not stored in the Atlanta, Bangor, or Chicago warehouse. Use the NOT IN keyword.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   SKU, SKU_Description,
         WAREHOUSE.WarehouseID, WarehouseCity, WarehouseState
FROM     INVENTORY, WAREHOUSE
WHERE    INVENTORY.WarehouseID=WAREHOUSE.WarehouseID
   AND   WarehouseCity NOT IN ('Atlanta', 'Bangor' ,'Chicago');
```

SQL-Query-CH02-RQ-02-43

SKU	SKU_Description	WarehouseID	WarehouseCity	WarehouseState
100100	Std. Scuba Tank, Yellow	400	Seattle	WA
100200	Std. Scuba Tank, Magenta	400	Seattle	WA
101100	Dive Mask, Small Clear	400	Seattle	WA
101200	Dive Mask, Med Clear	400	Seattle	WA
201000	Half-dome Tent	400	Seattle	WA
202000	Half-dome Tent Vestibule	400	Seattle	WA
301000	Light Fly Climbing Harness	400	Seattle	WA
302000	Locking Carabiner, Oval	400	Seattle	WA

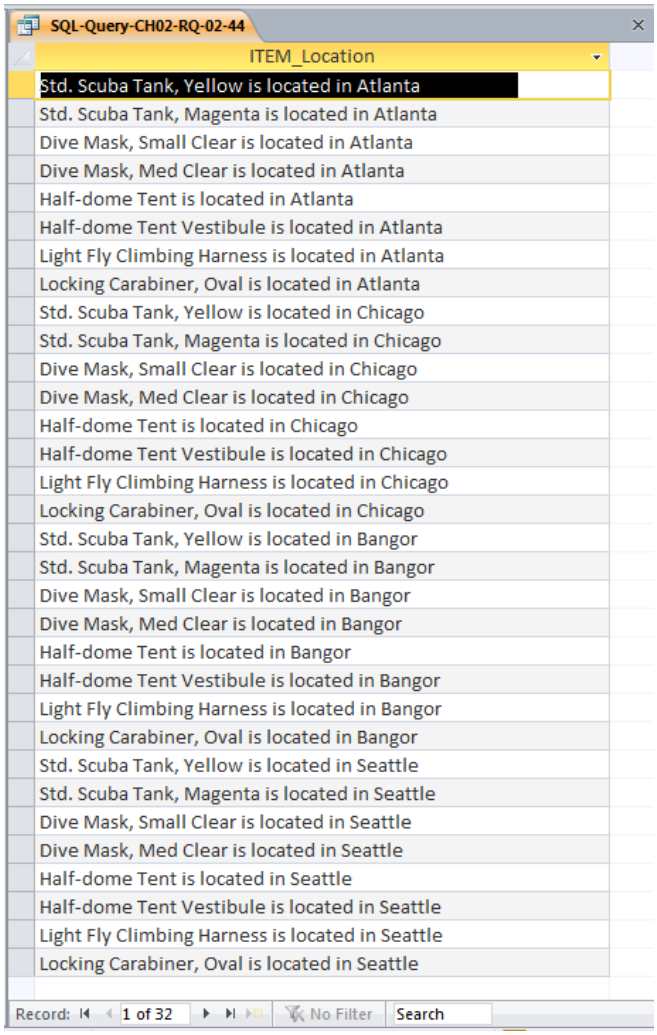
Record: 1 of 8 No Filter Search

2.44 Write an SQL statement to produce a single column called ItemLocation that combines the SKU_Description, the phrase "is located in", and WarehouseCity. Do not be concerned with removing leading or trailing blanks.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that the SQL syntax will vary depending upon the DBMS—see the discussion in Chapter 2.

```
SELECT   SKU_Description+' is located in '
         +WarehouseCity AS ITEM_Location
FROM     INVENTORY, WAREHOUSE
WHERE    INVENTORY.WarehouseID=WAREHOUSE.WarehouseID;
```



SQL-Query-CH02-RQ-02-44

ITEM_Location
Std. Scuba Tank, Yellow is located in Atlanta
Std. Scuba Tank, Magenta is located in Atlanta
Dive Mask, Small Clear is located in Atlanta
Dive Mask, Med Clear is located in Atlanta
Half-dome Tent is located in Atlanta
Half-dome Tent Vestibule is located in Atlanta
Light Fly Climbing Harness is located in Atlanta
Locking Carabiner, Oval is located in Atlanta
Std. Scuba Tank, Yellow is located in Chicago
Std. Scuba Tank, Magenta is located in Chicago
Dive Mask, Small Clear is located in Chicago
Dive Mask, Med Clear is located in Chicago
Half-dome Tent is located in Chicago
Half-dome Tent Vestibule is located in Chicago
Light Fly Climbing Harness is located in Chicago
Locking Carabiner, Oval is located in Chicago
Std. Scuba Tank, Yellow is located in Bangor
Std. Scuba Tank, Magenta is located in Bangor
Dive Mask, Small Clear is located in Bangor
Dive Mask, Med Clear is located in Bangor
Half-dome Tent is located in Bangor
Half-dome Tent Vestibule is located in Bangor
Light Fly Climbing Harness is located in Bangor
Locking Carabiner, Oval is located in Bangor
Std. Scuba Tank, Yellow is located in Seattle
Std. Scuba Tank, Magenta is located in Seattle
Dive Mask, Small Clear is located in Seattle
Dive Mask, Med Clear is located in Seattle
Half-dome Tent is located in Seattle
Half-dome Tent Vestibule is located in Seattle
Light Fly Climbing Harness is located in Seattle
Locking Carabiner, Oval is located in Seattle

Record: 1 of 32 No Filter Search

2.45 Write an SQL statement to show the SKU, SKU_Description, WarehouseID for all items stored in a warehouse managed by 'Lucille Smith'. Use a subquery.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

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

SQL-Query-CH02-RQ-02-45

SKU	SKU_Description	WarehouseID
100100	Std. Scuba Tank, Yellow	200
100200	Std. Scuba Tank, Magenta	200
101100	Dive Mask, Small Clear	200
101200	Dive Mask, Med Clear	200
201000	Half-dome Tent	200
202000	Half-dome Tent Vestibule	200
301000	Light Fly Climbing Harness	200
302000	Locking Carabiner, Oval	200

Record: 1 of 8 No Filter Search

2.46 Write an SQL statement to show the SKU, SKU_Description, and WarehouseID for all items stored in a warehouse managed by 'Lucille Smith'. Use a join, but do not use JOIN ON syntax.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

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

SQL-Query-CH02-RQ-02-46

SKU	SKU_Description	WarehouseID
100100	Std. Scuba Tank, Yellow	200
100200	Std. Scuba Tank, Magenta	200
101100	Dive Mask, Small Clear	200
101200	Dive Mask, Med Clear	200
201000	Half-dome Tent	200
202000	Half-dome Tent Vestibule	200
301000	Light Fly Climbing Harness	200
302000	Locking Carabiner, Oval	200

Record: 1 of 8 No Filter Search

2.47 Write an SQL statement to show the SKU, SKU_Description, WarehouseID for all items stored in a warehouse managed by 'Lucille Smith'. Use a join using JOIN ON syntax.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

For Microsoft SQL Server, Oracle Database, and MySQL:

```
SELECT      SKU, SKU_Description, WAREHOUSE.WarehouseID
FROM        INVENTORY JOIN WAREHOUSE
      ON    INVENTORY.WarehouseID=WAREHOUSE.WarehouseID
WHERE       Manager = 'Lucille Smith';
```

For Microsoft Access:

Microsoft Access requires the SQL JOIN ON syntax INNER JOIN instead of just JOIN:

```
SELECT      SKU, SKU_Description, WAREHOUSE.WarehouseID
FROM        INVENTORY INNER JOIN WAREHOUSE
      ON    INVENTORY.WarehouseID=WAREHOUSE.WarehouseID
WHERE       Manager = 'Lucille Smith';
```

SQL-Query-CH02-RQ-02-47

SKU	SKU_Description	WarehouseID
100100	Std. Scuba Tank, Yellow	200
100200	Std. Scuba Tank, Magenta	200
101100	Dive Mask, Small Clear	200
101200	Dive Mask, Med Clear	200
201000	Half-dome Tent	200
202000	Half-dome Tent Vestibule	200
301000	Light Fly Climbing Harness	200
302000	Locking Carabiner, Oval	200

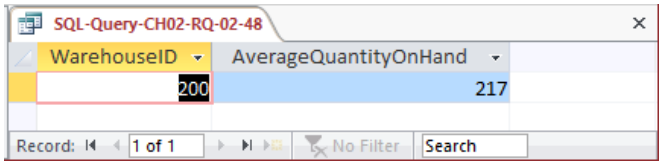
Record: 1 of 8 No Filter Search

2.48 Write an SQL statement to show the WarehouseID and average QuantityOnHand of all items stored in a warehouse managed by 'Lucille Smith'. Use a subquery.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that the "GROUP BY" clause is necessary here since warehouse manager names are not necessarily unique: since the question asks for warehouse ID, there should be one result for each warehouse managed by a 'Lucille Smith'.

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



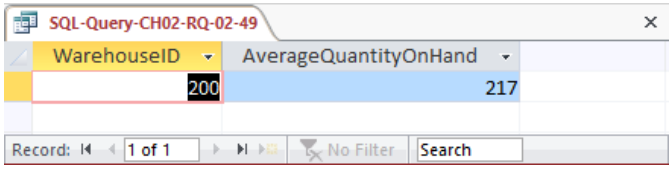
2.49 Write an SQL statement to show the WarehouseID and average QuantityOnHand of all items stored in a warehouse managed by 'Lucille Smith'. Use a join, but do not use JOIN ON syntax.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that the "GROUP BY" clause is necessary here since warehouse manager names are not necessarily unique: since the question asks for warehouse ID, there should be one result for each warehouse managed by a 'Lucille Smith'.

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

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



2.50 Write an SQL statement to show the WarehouseID and average QuantityOnHand of all items stored in a warehouse managed by 'Lucille Smith'. Use a join using JOIN ON syntax.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that the "GROUP BY" clause is necessary here since warehouse manager names are not necessarily unique: since the question asks for warehouse ID, there should be one result for each warehouse managed by a 'Lucille Smith'.

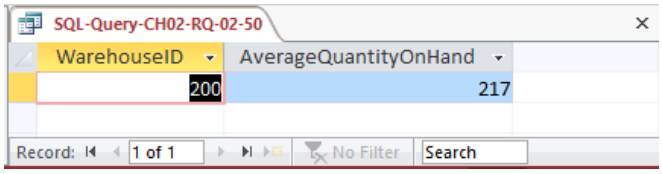
For Microsoft SQL Server, Oracle Database, and MySQL:

```
SELECT      INVENTORY.WarehouseID,
            AVG(QuantityOnHand) AS AverageQuantityOnHand
FROM        INVENTORY JOIN WAREHOUSE
      ON    INVENTORY.WarehouseID=WAREHOUSE.WarehouseID
WHERE       Manager = 'Lucille Smith'
GROUP BY    INVENTORY.WarehouseID;
```

For Microsoft Access:

Microsoft Access requires the SQL JOIN ON syntax INNER JOIN instead of just JOIN:

```
SELECT      INVENTORY.WarehouseID,
            AVG(QuantityOnHand) AS AverageQuantityOnHand
FROM        INVENTORY INNER JOIN WAREHOUSE
      ON    INVENTORY.WarehouseID=WAREHOUSE.WarehouseID
WHERE       Manager = 'Lucille Smith'
GROUP BY    INVENTORY.WarehouseID;
```



2.51 Write an SQL statement to show the WarehouseID, WarehouseCity, WarehouseState, Manager, SKU, SKU_Description, and QuantityOnHand of all items with a Manager of 'Lucille Smith'. Use a join using JOIN ON syntax.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT   WAREHOUSE.WarehouseID, WarehouseCity,
         WarehouseState, Manager,
         SKU, SKU_Description, QuantityOnHand
FROM     INVENTORY INNER JOIN WAREHOUSE
ON    INVENTORY.WarehouseID=WAREHOUSE.WarehouseID
WHERE Manager = 'Lucille Smith';
```

SQL-Query-CH02-RQ-02-51

Warehousel	WarehouseCity	WarehouseState	Manager	SKU	SKU_Description	QuantityOnHand
200	Chicago	IL	Lucille Smith	100100	Std. Scuba Tank, Yellow	100
200	Chicago	IL	Lucille Smith	100200	Std. Scuba Tank, Magenta	75
200	Chicago	IL	Lucille Smith	101100	Dive Mask, Small Clear	0
200	Chicago	IL	Lucille Smith	101200	Dive Mask, Med Clear	50
200	Chicago	IL	Lucille Smith	201000	Half-dome Tent	10
200	Chicago	IL	Lucille Smith	202000	Half-dome Tent Vestibule	1
200	Chicago	IL	Lucille Smith	301000	Light Fly Climbing Harness	250
200	Chicago	IL	Lucille Smith	302000	Locking Carabiner, Oval	1250

Record: 1 of 8 No Filter Search

Note the use of the complete references to **INVENTORY.WarehouseID and WAREHOUSE.WarehouseID**—the query will NOT work without them.

The above version of the query works in Access, SQL Server, Oracle Database, and MySQL. The "INNER" keyword is required in Access, but is optional in SQL Server, Oracle, and MySQL. In addition, this query could benefit from aliasing (range variables) for readability, but that syntax is slightly different in Oracle than in the other three systems (the "AS" keyword is not allowed in Oracle). Thus the most typical, preferred solutions for each system are as follows:

For Microsoft Access:

```
SELECT   W.WarehouseID, WarehouseCity,
         WarehouseState, Manager,
         SKU, SKU_Description, QuantityOnHand
FROM     INVENTORY AS I INNER JOIN WAREHOUSE AS W
ON    I.WarehouseID=W.WarehouseID
WHERE          Manager = 'Lucille Smith';
```

For Oracle Database:

```
SELECT   W.WarehouseID, WarehouseCity,
         WarehouseState, Manager,
         SKU, SKU_Description, QuantityOnHand
FROM     INVENTORY I INNER JOIN WAREHOUSE W
ON    I.WarehouseID=W.WarehouseID
WHERE Manager = 'Lucille Smith';
```

For SQL Server and MySQL:

```
SELECT   W.WarehouseID, WarehouseCity,
         WarehouseState, Manager,
         SKU, SKU_Description, QuantityOnHand
FROM     INVENTORY AS I JOIN WAREHOUSE AS W
ON    I.WarehouseID=W.WarehouseID
WHERE Manager = 'Lucille Smith';
```

2.52 Write an SQL statement to display the WarehouseID, the sum of QuantityOnOrder and sum of QuantityOnHand, grouped by WarehouseID and QuantityOnOrder. Name the sum of QuantityOnOrder as TotalItemsOnOrder and the sum of QuantityOnHand as TotalItemsOnHand. Use only the INVENTORY table in your SQL statement.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT      WarehouseID,
            SUM(QuantityOnOrder) AS TotalItemsOnOrder,
            SUM(QuantityOnHand) AS TotalItemsOnHand
FROM        INVENTORY
GROUP BY    WarehouseID, QuantityOnOrder;
```

SQL-Query-CH02-RQ-02-52

WarehouseID	TotalItemsOnOrder	TotalItemsOnHand
100	0	1250
100	30	200
100	100	2
100	500	310
100	1000	100
200	0	1250
200	50	100
200	75	75
200	750	261
200	1000	50
300	0	925
300	100	100
300	200	300
300	250	0
300	500	500
400	0	900
400	200	0
400	750	250
400	1000	0

Record: 1 of 19 No Filter Search

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

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. In question 2.52 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.

2.54 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 2.53.

(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.

2.55 Write an SQL statement to join WAREHOUSE and INVENTORY and include all rows of WAREHOUSE in your answer, regardless of whether they have any INVENTORY. Run this statement.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that the question doesn't specify which columns to retrieve; we retrieve all columns (but without repeating the join column).

```
SELECT   W.*, I.SKU, I.SKU_Description, I.QuantityOnHand,
   I.QuantityOnOrder
FROM     WAREHOUSE AS W LEFT JOIN INVENTORY AS I
         ON W.WarehouseID = I.WarehouseID;
```

In Oracle, the "AS" keyword is not permitted in the "JOIN" clause, so the Oracle Database solution is:

```
SELECT   W.*, I.SKU, I.SKU_Description, I.QuantityOnHand,
   I.QuantityOnOrder
FROM     WAREHOUSE W LEFT JOIN INVENTORY I
         ON W.WarehouseID = I.WarehouseID;
```

SQL-Query-CH02-RQ-02-55

WarehouseI	Warehouse	Warehouse	Manager	SquareFeet	SKU	SKU_Description	QuantityOnI	QuantityOn
100	Atlanta	GA	Dave Jones	125000	100100	Std. Scuba Tank, Yellow	250	0
100	Atlanta	GA	Dave Jones	125000	100200	Std. Scuba Tank, Magenta	200	30
100	Atlanta	GA	Dave Jones	125000	101100	Dive Mask, Small Clear	0	500
100	Atlanta	GA	Dave Jones	125000	101200	Dive Mask, Med Clear	100	500
100	Atlanta	GA	Dave Jones	125000	201000	Half-dome Tent	2	100
100	Atlanta	GA	Dave Jones	125000	202000	Half-dome Tent Vestibule	10	250
100	Atlanta	GA	Dave Jones	125000	301000	Light Fly Climbing Harness	300	250
100	Atlanta	GA	Dave Jones	125000	302000	Locking Carabiner, Oval	1000	0
200	Chicago	IL	Lucille Smith	100000	100100	Std. Scuba Tank, Yellow	100	50
200	Chicago	IL	Lucille Smith	100000	100200	Std. Scuba Tank, Magenta	75	75
200	Chicago	IL	Lucille Smith	100000	101100	Dive Mask, Small Clear	0	500
200	Chicago	IL	Lucille Smith	100000	101200	Dive Mask, Med Clear	50	500
200	Chicago	IL	Lucille Smith	100000	201000	Half-dome Tent	10	250
200	Chicago	IL	Lucille Smith	100000	202000	Half-dome Tent Vestibule	1	250
200	Chicago	IL	Lucille Smith	100000	301000	Light Fly Climbing Harness	250	250
200	Chicago	IL	Lucille Smith	100000	302000	Locking Carabiner, Oval	1250	0
300	Bangor	ME	Bart Evans	150000	100100	Std. Scuba Tank, Yellow	100	0
300	Bangor	ME	Bart Evans	150000	100200	Std. Scuba Tank, Magenta	100	100
300	Bangor	ME	Bart Evans	150000	101100	Dive Mask, Small Clear	300	200
300	Bangor	ME	Bart Evans	150000	101200	Dive Mask, Med Clear	475	0
300	Bangor	ME	Bart Evans	150000	201000	Half-dome Tent	250	0
300	Bangor	ME	Bart Evans	150000	202000	Half-dome Tent Vestibule	100	0
300	Bangor	ME	Bart Evans	150000	301000	Light Fly Climbing Harness	0	250
300	Bangor	ME	Bart Evans	150000	302000	Locking Carabiner, Oval	500	500
400	Seattle	WA	Dale Rogers	130000	100100	Std. Scuba Tank, Yellow	200	0
400	Seattle	WA	Dale Rogers	130000	100200	Std. Scuba Tank, Magenta	250	0
400	Seattle	WA	Dale Rogers	130000	101100	Dive Mask, Small Clear	450	0
400	Seattle	WA	Dale Rogers	130000	101200	Dive Mask, Med Clear	250	250
400	Seattle	WA	Dale Rogers	130000	201000	Half-dome Tent	0	250
400	Seattle	WA	Dale Rogers	130000	202000	Half-dome Tent Vestibule	0	200
400	Seattle	WA	Dale Rogers	130000	301000	Light Fly Climbing Harness	0	250
400	Seattle	WA	Dale Rogers	130000	302000	Locking Carabiner, Oval	0	1000
500	San Francisco	CA	Grace Jefferson	200000				

Record: 1 of 33 | No Filter | Search

Use both the CATALOG_SKU_2013 and CATALOG_SKU_2014 tables to answer Review Questions 2.56 through 2.60 (for MySQL, 2.56 and 2.57 only):

2.56 *Write an SQL statement to display the SKU, SKU_Description, and Department of all SKUs that appear in* either *the Cape Codd 2013 Catalog (either in the printed catalog or on the Web site)* or *the Cape Codd 2014 catalog (either in the printed catalog or on the Web site)* or *both.*

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2013
UNION
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2014;
```

SQL-Query-CH02-RQ-2-56

SKU	SKU_Description	Department
100100	Std. Scuba Tank, Yellow	Water Sports
100300	Std. Scuba Tank, Light Blue	Water Sports
100400	Std. Scuba Tank, Dark Blue	Water Sports
100500	Std. Scuba Tank, Light Green	Water Sports
100600	Std. Scuba Tank, Dark Green	Water Sports
101100	Dive Mask, Small Clear	Water Sports
101200	Dive Mask, Med Clear	Water Sports
201000	Half-dome Tent	Camping
202000	Half-dome Tent Vestibule	Camping
301000	Light Fly Climbing Harness	Climbing
302000	Locking Carabiner, Oval	Climbing

Record: 1 of 11 No Filter Search

2.57 *Write an SQL statement to display the SKU, SKU_Description, and Department of all SKUs that appear in* either *the Cape Codd 2013 Catalog (only in the printed catalog itself)* or *the Cape Codd 2014 catalog (only in the printed catalog itself)* or *both.*

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2013
WHERE       CatalogPage IS NOT NULL
UNION
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2014
WHERE       CatalogPage IS NOT NULL;
```

SQL-Query-CH02-RQ-02-57

SKU	SKU_Description	Department
100100	Std. Scuba Tank, Yellow	Water Sports
100300	Std. Scuba Tank, Light Blue	Water Sports
101100	Dive Mask, Small Clear	Water Sports
101200	Dive Mask, Med Clear	Water Sports
201000	Half-dome Tent	Camping
202000	Half-dome Tent Vestibule	Camping
301000	Light Fly Climbing Harness	Climbing
302000	Locking Carabiner, Oval	Climbing

Record: 1 of 8 No Filter Search

2.58 *Write an SQL statement to display the SKU, SKU_Description, and Department of all SKUs that appear in* both *the Cape Codd 2013 Catalog (either in the printed catalog or on the Web site)* and *the Cape Codd 2014 catalog (either in the printed catalog or on the Web site).*

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

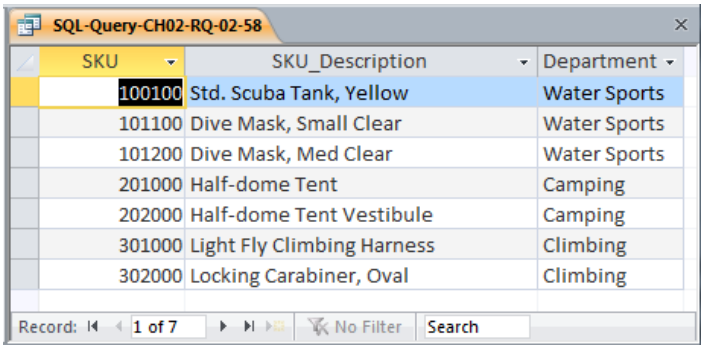
Note that Oracle Database and SQL Server support INTERSECT directly. In MySQL and Access INTERSECT is not supported but can be simulated using a join.

For Oracle and SQL Server:

```
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2013
INTERSECT
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2014;
```

For MySQL and Access:

```
SELECT      DISTINCT CS13.SKU, CS13.SKU_Description, CS13.Department
FROM        CATALOG_SKU_2013 AS CS13
      INNER JOIN  CATALOG_SKU_2014 AS CS14
ON          CS13.SKU = CS14.SKU;
```



2.59 *Write an SQL statement to display the SKU, SKU_Description, and Department of all SKUs that appear in* both *the Cape Codd 2013 Catalog (only in the printed catalog itself)* and *the Cape Codd 2014 catalog (only in the printed catalog itself).*

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that Oracle Database and SQL Server support INTERSECT directly. In MySQL and Access INTERSECT is not supported but can be simulated using a join.

For Oracle and SQL Server:

```
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2013
WHERE       CatalogPage IS NOT NULL
INTERSECT
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2014
WHERE       CatalogPage IS NOT NULL;
```

For MySQL and Access:

```
SELECT      DISTINCT CS13.SKU, CS13.SKU_Description, CS13.Department
FROM        CATALOG_SKU_2013 AS CS13
      INNER JOIN  CATALOG_SKU_2014 AS CS14
ON                      CS13.SKU = CS14.SKU
WHERE CS13.CatalogPage IS NOT NULL AND CS14.CatalogPage IS NOT NULL;
```

SQL-Query-CH02-RQ-02-59

SKU	SKU_Description	Department
100100	Std. Scuba Tank, Yellow	Water Sports
101100	Dive Mask, Small Clear	Water Sports
101200	Dive Mask, Med Clear	Water Sports
201000	Half-dome Tent	Camping
202000	Half-dome Tent Vestibule	Camping
301000	Light Fly Climbing Harness	Climbing
302000	Locking Carabiner, Oval	Climbing

Record: 1 of 7 No Filter Search

2.60 *Write an SQL statement to display the SKU, SKU_Description, and Department of all SKUs that appear in* only *the Cape Codd 2013 Catalog (either in the printed catalog or on the Web site)* and not *in the Cape Codd 2014 catalog (either in the printed catalog or on the Web site).*

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e14-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that Oracle Database and SQL Server support set subtraction directly. In MySQL and Access this operation is not supported but can be simulated using an outer join.

For SQL Server:

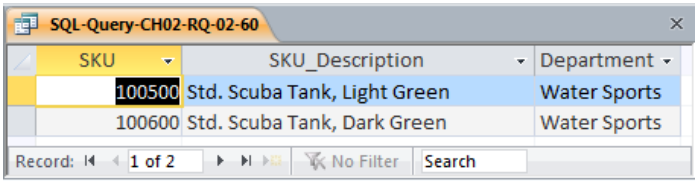
```
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2013
EXCEPT
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2014;
```

For Oracle:

```
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2013
MINUS
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2014;
```

For MySQL and Access:

```
SELECT      DISTINCT CS13.SKU, CS13.SKU_Description, CS13.Department
FROM        CATALOG_SKU_2013 AS CS13
      LEFT OUTER JOIN   CATALOG_SKU_2014 AS CS14
ON          CS13.SKU = CS14.SKU
WHERE       CS14.SKU IS NULL;
```



ANSWERS TO PROJECT QUESTIONS

For this set of project questions, we will extend the Microsoft Access 2013 database for the Wedgewood Pacific Corporation (WPC) that we created in Chapter 1. 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 (see Chapter 1 Project Questions):

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)

The four tables in the revised WPC database schema are shown in Figure 2-41. The column characteristics for the PROJECT table are shown in Figure 2-42, and the column characteristics for the ASSIGNMENT table are shown in Figure 2-44. Data for the PROJECT table are shown in Figure 2-43, and the data for the ASSIGNMENT table are shown in Figure 2-45.

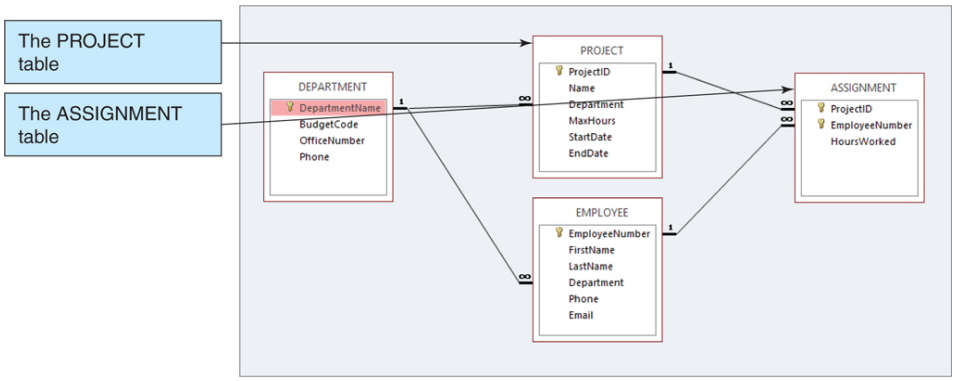


Figure 2-41 – The WPC Database with the PROJECT and ASSIGNMENT Tables

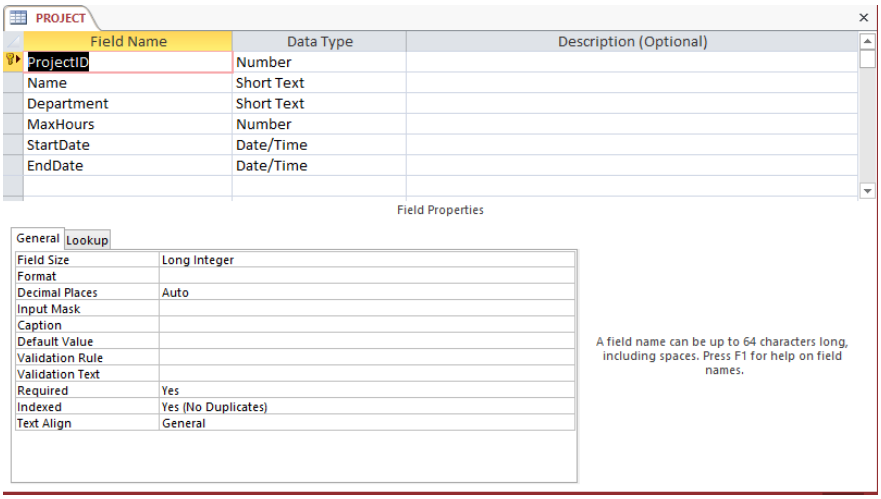
2.61 Figure 2-42 shows the column characteristics for the WPC PROJECT table. Using the column characteristics, create the PROJECT table in the WPC.accdb database.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e14-IM-CH02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

PROJECT

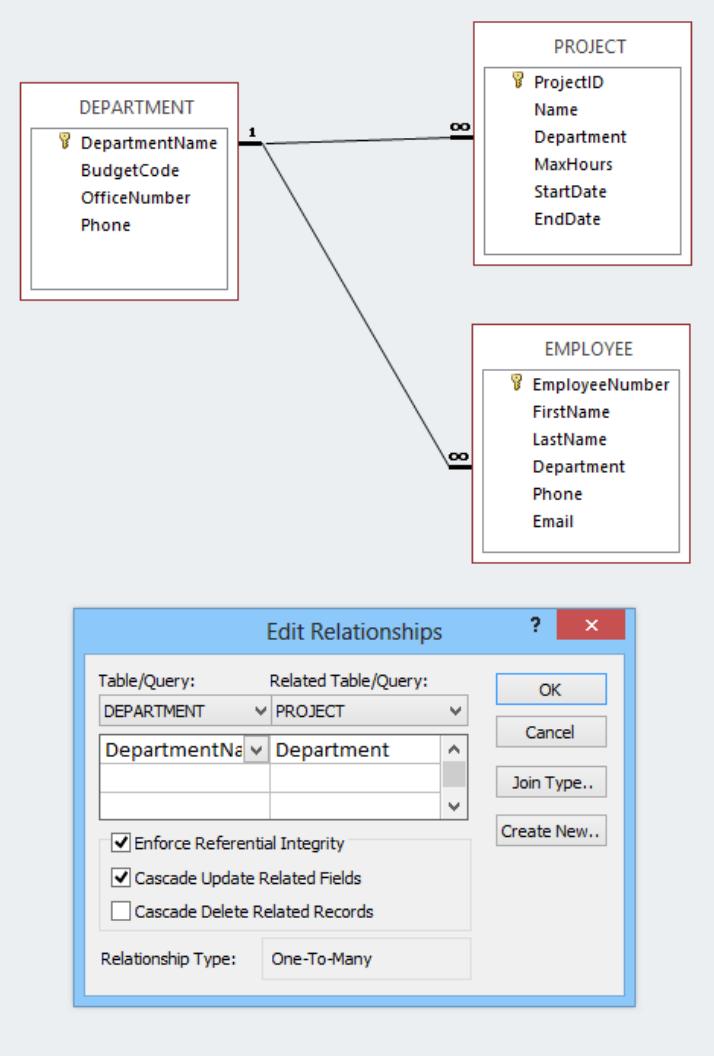
Column Name	Type	Key	Required	Remarks
ProjectID	Integer	Primary Key	DBMS supplied	Surrogate Key
Name	Character (50)	No	Yes	
Department	Character (35)	Foreign Key	Yes	REF: DEPARTMENT
MaxHours	Number (8,2)	No	Yes	
StartDate	Date	No	No	
EndDate	Date	No	No	

Figure 2-42 - Column Characteristics for the PROJECT Table



2.62 Create the relationship and referential integrity constraint between PROJECT and DEPARTMENT. In the Edit Relationship dialog box, enable enforcing of referential integrity and cascading of data updates, but do not enable cascading of data from deleted records. We will define cascading actions in Chapter 6.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e14-IM-CH02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

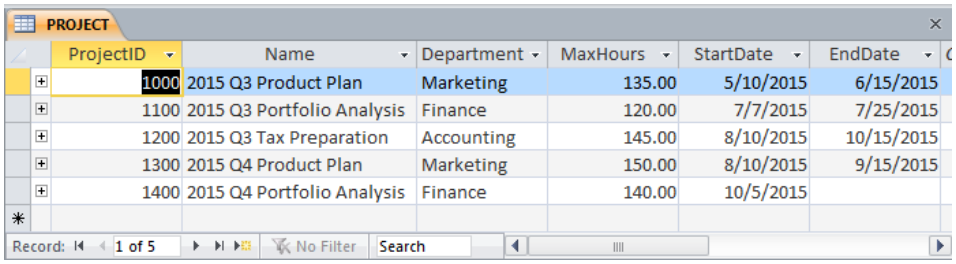


2.63 Figure 2-43 shows the data for the WPC PROJECT table. Using the Datasheet view, enter the data shown in Figure 2-43 into your PROJECT table.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e14-IM-CH02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

ProjectID	Name	Department	MaxHours	StartDate	EndDate
1000	2015 Q3 Product Plan	Marketing	135.00	10-MAY-15	15-JUN-15
1100	2015 Q3 Portfolio Analysis	Finance	120.00	07-JUL-15	25-JUL-15
1200	2015 Q3 Tax Preparation	Accounting	145.00	10-AUG-15	15-OCT-15
1300	2015 Q4 Product Plan	Marketing	150.00	10-AUG-15	15-SEP-15
1400	2015 Q4 Portfolio Analysis	Finance	140.00	05-OCT-15	

Figure 2-43 - Sample Data for the PROJECT Table

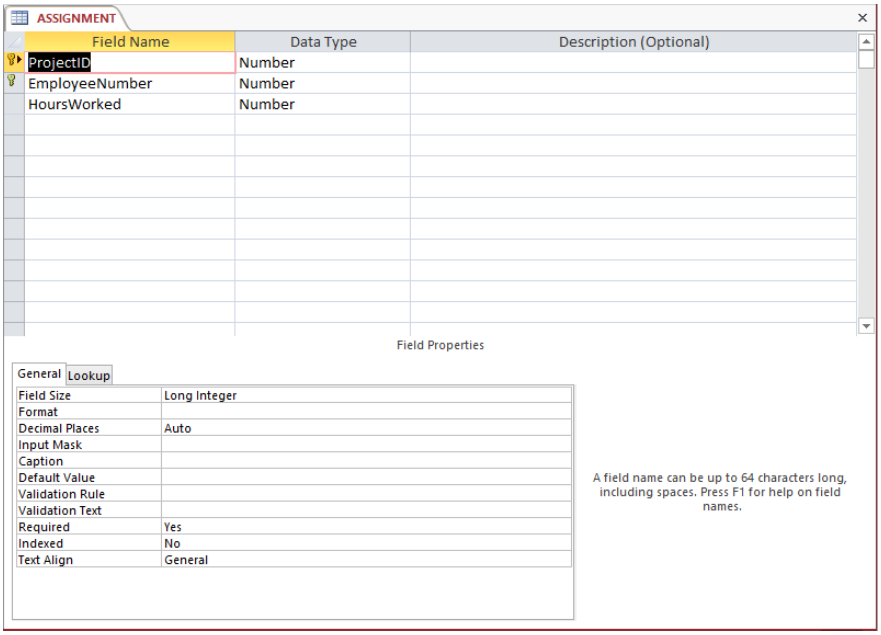


2.64 Figure 2-44 shows the column characteristics for the WPC ASSIGNMENT table. Using the column characteristics, create the ASSIGNMENT table in the WPC.accdb database.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e14-IM-CH02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

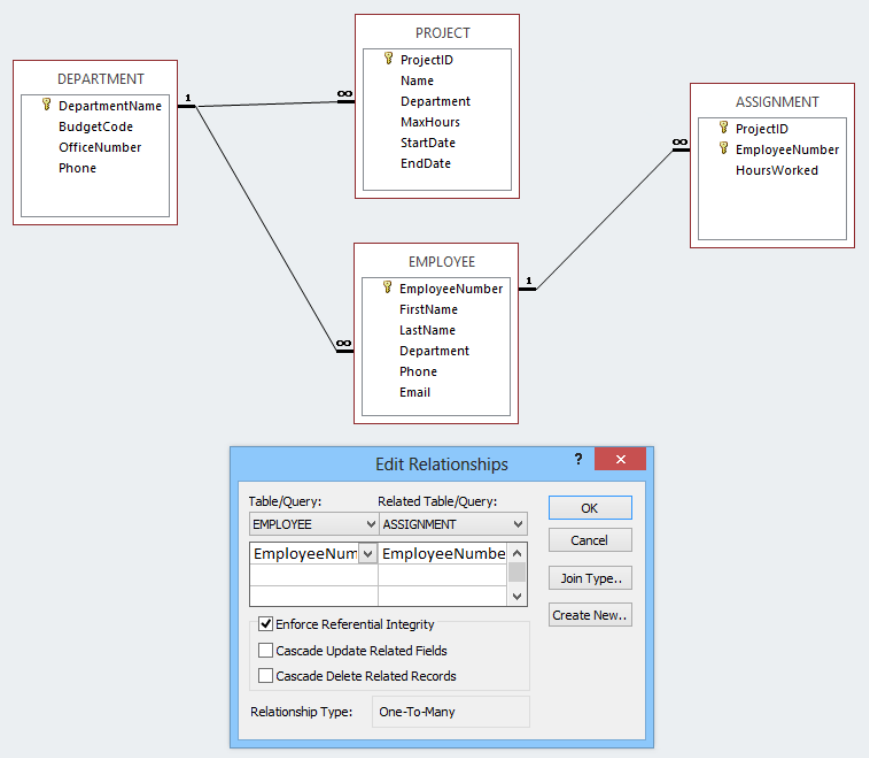
Column Name	Type	Key	Required	Remarks
ProjectID	Integer	Primary Key, Foreign Key	Yes	REF: PROJECT
EmployeeNumber	Integer	Primary Key, Foreign Key	Yes	REF: EMPLOYEE
HoursWorked	Number (6,2)	No	No	

Figure 2-44 - Column Characteristics for the ASSIGNMENT Table



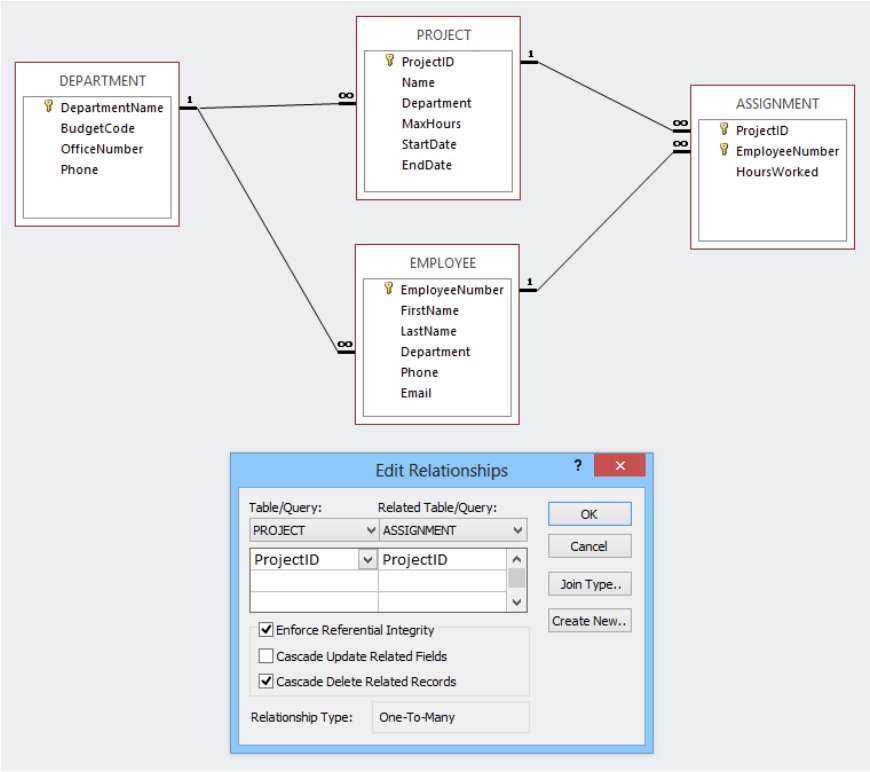
2.65 Create the relationship and referential integrity constraint between ASSIGNMENT and EMPLOYEE. In the Edit Relationship dialog box, enable enforcing of referential integrity, but do not enable either cascading updates or the cascading of data from deleted records.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e14-IM-CH02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).



2.66 *Create the relationship and referential integrity constraint between ASSIGNMENT and PROJECT. In the Edit Relationship dialog box, enable enforcing of referential integrity and cascading of deletes, but do not enable cascading updates.*

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e14-IM-CH02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

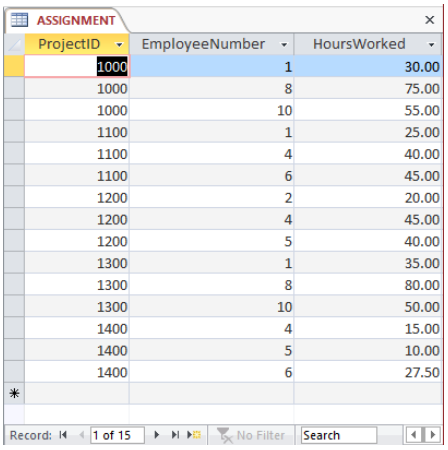


2.67 Figure 2-45 shows the data for the WPC ASSIGNMENT table. Using the Datasheet view, enter the data shown in Figure 2-45 into your ASSIGNMENT table.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e14-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 2-45 - Sample Data for the ASSIGNMENT Table



2.68 *In Project Question 2.63, the table data was entered after referential integrity constraints were created in Project Question 2.62. In Project Question 2.67, the table data was entered after referential integrity constraints were created in Project Questions 2.65 and 2.66. Why was the data entered after the referential integrity constraints were created instead of before the constraints were created?*

Both the PROJECT and ASSIGNMENT tables have foreign keys. PROJECT.Department is the foreign key in PROJECT, and both ASSIGNMENT.ProjectID and ASSIGNMENT.EmployeeNumber are foreign keys in ASSIGNMENT. If data was entered into these columns before the referential integrity constraints were established, it would be possible to enter foreign key data that had no corresponding primary key data. Thus, we establish the referential integrity constraints so that the DBMS will not allow inconsistent data to be entered into the foreign key columns.

2.69 *Using Microsoft 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.*

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e14-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;
```

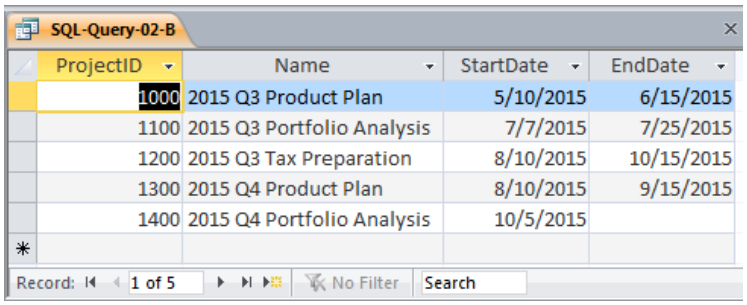
SQL-Query-02-A

ProjectID	Name	Department	MaxHours	StartDate	EndDate
1000	2015 Q3 Product Plan	Marketing	135.00	5/10/2015	6/15/2015
1100	2015 Q3 Portfolio Analysis	Finance	120.00	7/7/2015	7/25/2015
1200	2015 Q3 Tax Preparation	Accounting	145.00	8/10/2015	10/15/2015
1300	2015 Q4 Product Plan	Marketing	150.00	8/10/2015	9/15/2015
1400	2015 Q4 Portfolio Analysis	Finance	140.00	10/5/2015	

Record: 1 of 5 No Filter Search

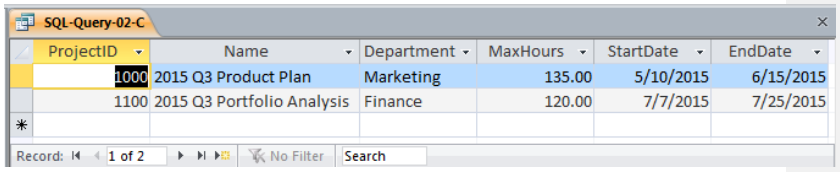
B. *What are the ProjectID, Name, StartDate, and EndDate values of projects in the PROJECT table?*

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

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



C. *What projects in the PROJECT table started before August 1, 2015? Show all the information for each project.*

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

SELECT    *
FROM      PROJECT
WHERE     StartDate < #01-AUG-15#;
```



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;
```

SQL-Query-02-D

ProjectID	Name	Department	MaxHours	StartDate	EndDate
1400	2015 Q4 Portfolio Analysis	Finance	140.00	10/5/2015	

Record: 1 of 1 No Filter Search

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;
```

SQL-Query-02-E

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	360-285-8320
1400	5	Jones	Heather	360-285-8320
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 No Filter Search

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

Note the use of the aliases **ProjectName**, **ProjectDepartment,** and **EmployeePhone**)

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

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

SQL-Query-02-F

ProjectID	ProjectName	ProjectDepartment	EmployeeNumber	LastName	FirstName	EmployeePhone
1000	2015 Q3 Product Plan	Marketing	1	Jacobs	Mary	360-285-8110
1000	2015 Q3 Product Plan	Marketing	8	Jackson	Tom	360-287-8610
1000	2015 Q3 Product Plan	Marketing	10	Numoto	Ken	360-287-8710
1100	2015 Q3 Portfolio Analysis	Finance	4	Caruthers	Tom	360-285-8310
1100	2015 Q3 Portfolio Analysis	Finance	6	Abernathy	Mary	360-285-8410
1100	2015 Q3 Portfolio Analysis	Finance	1	Jacobs	Mary	360-285-8110
1200	2015 Q3 Tax Preparation	Accounting	2	Jackson	Rosalie	360-285-8120
1200	2015 Q3 Tax Preparation	Accounting	4	Caruthers	Tom	360-285-8310
1200	2015 Q3 Tax Preparation	Accounting	5	Jones	Heather	360-285-8320
1300	2015 Q4 Product Plan	Marketing	1	Jacobs	Mary	360-285-8110
1300	2015 Q4 Product Plan	Marketing	8	Jackson	Tom	360-287-8610
1300	2015 Q4 Product Plan	Marketing	10	Numoto	Ken	360-287-8710
1400	2015 Q4 Portfolio Analysis	Finance	4	Caruthers	Tom	360-285-8310
1400	2015 Q4 Portfolio Analysis	Finance	5	Jones	Heather	360-285-8320
1400	2015 Q4 Portfolio Analysis	Finance	6	Abernathy	Mary	360-285-8410

Record: 1 of 15 No Filter Search

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.*

Note the use of the aliases **ProjectName**, **ProjectDepartment**, **DepartmentPhone** and **EmployeePhone**.

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

SELECT    P.ProjectID, Name AS ProjectName,
          D.DepartmentName AS ProjectDepartment,
          D.Phone AS DepartmentPhone,
          E.EmployeeNumber, LastName, FirstName,
          E.Phone AS EmployeePhone
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;
```

SQL-Query-02-G

ProjectID	ProjectName	ProjectDepartment	DepartmentPhone	EmployeeNumber	LastName	FirstName	EmployeePhone
1000	2015 Q3 Product Plan	Marketing	360-287-8700	10	Numoto	Ken	360-287-8710
1000	2015 Q3 Product Plan	Marketing	360-287-8700	8	Jackson	Tom	360-287-8610
1000	2015 Q3 Product Plan	Marketing	360-287-8700	1	Jacobs	Mary	360-285-8110
1100	2015 Q3 Portfolio Analysis	Finance	360-285-8400	1	Jacobs	Mary	360-285-8110
1100	2015 Q3 Portfolio Analysis	Finance	360-285-8400	6	Abernathy	Mary	360-285-8410
1100	2015 Q3 Portfolio Analysis	Finance	360-285-8400	4	Caruthers	Tom	360-285-8310
1200	2015 Q3 Tax Preparation	Accounting	360-285-8300	5	Jones	Heather	360-285-8320
1200	2015 Q3 Tax Preparation	Accounting	360-285-8300	4	Caruthers	Tom	360-285-8310
1200	2015 Q3 Tax Preparation	Accounting	360-285-8300	2	Jackson	Rosalie	360-285-8120
1300	2015 Q4 Product Plan	Marketing	360-287-8700	10	Numoto	Ken	360-287-8710
1300	2015 Q4 Product Plan	Marketing	360-287-8700	8	Jackson	Tom	360-287-8610
1300	2015 Q4 Product Plan	Marketing	360-287-8700	1	Jacobs	Mary	360-285-8110
1400	2015 Q4 Portfolio Analysis	Finance	360-285-8400	6	Abernathy	Mary	360-285-8410
1400	2015 Q4 Portfolio Analysis	Finance	360-285-8400	5	Jones	Heather	360-285-8320
1400	2015 Q4 Portfolio Analysis	Finance	360-285-8400	4	Caruthers	Tom	360-285-8310

Record: 1 of 15 No Filter 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.

Note the use of the aliases **ProjectName**, **ProjectDepartment**, **DepartmentPhone,** and **EmployeePhone**.

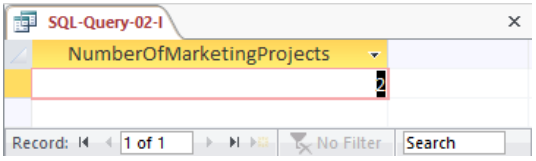
```
/*****   Question H - SQL-Query-02-H   **********************/

SELECT    P.ProjectID, Name AS ProjectName,
          D.DepartmentName AS ProjectDepartment,
          D.Phone AS DepartmentPhone,
          E.EmployeeNumber, LastName, FirstName,
          E.Phone AS EmployeePhone
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;
```

ProjectID	ProjectName	ProjectDepartment	DepartmentPhone	EmployeeNumber	LastName	FirstName	EmployeePhone
1000	2015 Q3 Product Plan	Marketing	360-287-8700	10	Numoto	Ken	360-287-8710
1000	2015 Q3 Product Plan	Marketing	360-287-8700	8	Jackson	Tom	360-287-8610
1000	2015 Q3 Product Plan	Marketing	360-287-8700	1	Jacobs	Mary	360-285-8110
1300	2015 Q4 Product Plan	Marketing	360-287-8700	10	Numoto	Ken	360-287-8710
1300	2015 Q4 Product Plan	Marketing	360-287-8700	8	Jackson	Tom	360-287-8610
1300	2015 Q4 Product Plan	Marketing	360-287-8700	1	Jacobs	Mary	360-285-8110

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

Note the use of the alias **NumberOfMarketingProjects**.

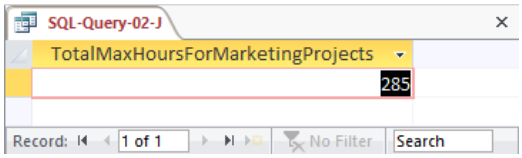
```
/*****   Question I - SQL-Query-02-I   **********************/

SELECT    COUNT(*) AS NumberOfMarketingProjects
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.*

Note the use of the alias **TotalMaxHoursForMarketingProjects**.

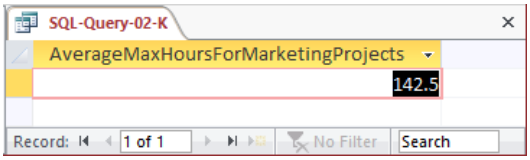
```
/*****   Question J - SQL-Query-02-J   **********************/

SELECT    SUM(MaxHours) AS TotalMaxHoursForMarketingProjects
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.*

Note the use of the alias **AverageMaxHoursForMarketingProjects**.

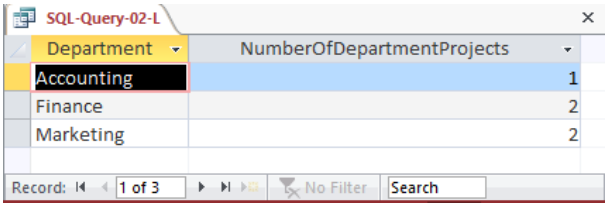
```
/*****   Question K - SQL-Query-02-K   **********************/

SELECT    AVG(MaxHours) AS AverageMaxHoursForMarketingProjects
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.*

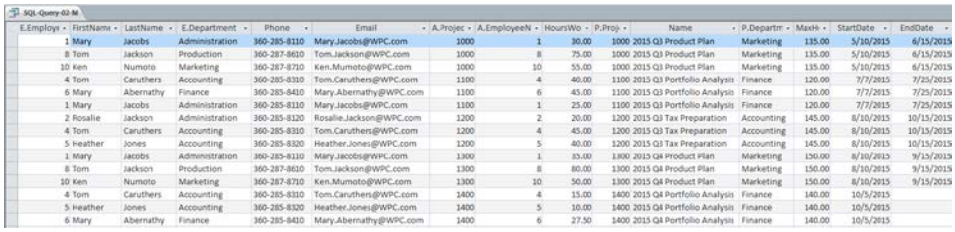
Note the use of the alias **NumberOfDepartmentProjects**.

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

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



M. *Write an SQL statement to join EMPLOYEE, ASSIGNMENT, and PROJECT using the JOIN ON syntax. Run this statement.*

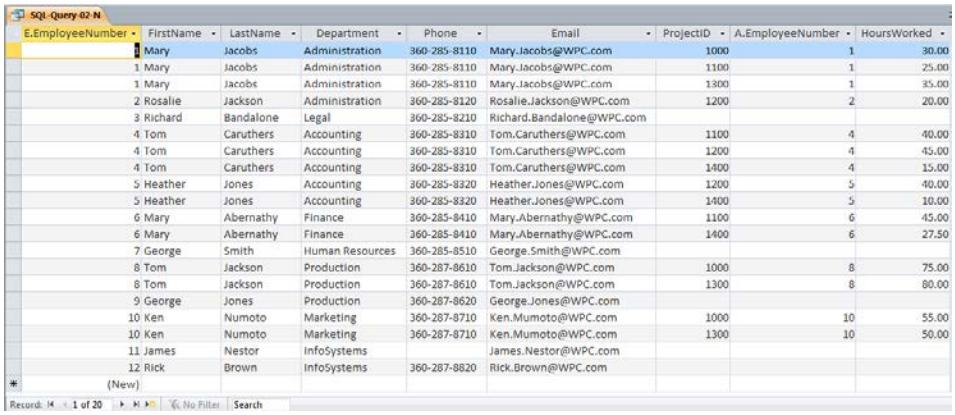
```
SELECT   E.*, A.*, P.*
FROM  (EMPLOYEE AS E INNER JOIN ASSIGNMENT AS A
      ON E.EmployeeNumber = A.EmployeeNumber)
         INNER JOIN PROJECT AS P
            ON A.ProjectID = P.ProjectID;
```



E.Employe	FirstName	LastName	E.Department	Phone	Email	A.Projec	A.EmployeeN	HoursWo	P.Proje	Name	P.Departm	MaxHo	StartDate	EndDate
1	Mary	Jacobs	Administration	360-285-8110	Mary.Jacobs@WPC.com	1000	1	30.00	1000	2015 Q3 Product Plan	Marketing	135.00	5/10/2015	6/15/2015
8	Tom	Jackson	Production	360-287-8610	Tom.Jackson@WPC.com	1000	8	75.00	1000	2015 Q3 Product Plan	Marketing	135.00	5/10/2015	6/15/2015
10	Ken	Numoto	Marketing	360-287-8710	Ken.Mumoto@WPC.com	1000	10	55.00	1000	2015 Q3 Product Plan	Marketing	135.00	5/10/2015	6/15/2015
4	Tom	Caruthers	Accounting	360-285-8310	Tom.Caruthers@WPC.com	1100	4	40.00	1100	2015 Q3 Portfolio Analysis	Finance	120.00	7/7/2015	7/25/2015
6	Mary	Abernathy	Finance	360-285-8410	Mary.Abernathy@WPC.com	1100	6	45.00	1100	2015 Q3 Portfolio Analysis	Finance	120.00	7/7/2015	7/25/2015
1	Mary	Jacobs	Administration	360-285-8110	Mary.Jacobs@WPC.com	1100	1	25.00	1100	2015 Q3 Portfolio Analysis	Finance	120.00	7/7/2015	7/25/2015
2	Rosalie	Jackson	Administration	360-285-8120	Rosalie.Jackson@WPC.com	1200	2	20.00	1200	2015 Q3 Tax Preparation	Accounting	145.00	8/10/2015	10/15/2015
4	Tom	Caruthers	Accounting	360-285-8310	Tom.Caruthers@WPC.com	1200	4	45.00	1200	2015 Q3 Tax Preparation	Accounting	145.00	8/10/2015	10/15/2015
5	Heather	Jones	Accounting	360-285-8320	Heather.Jones@WPC.com	1200	5	40.00	1200	2015 Q3 Tax Preparation	Accounting	145.00	8/10/2015	10/15/2015
1	Mary	Jacobs	Administration	360-285-8110	Mary.Jacobs@WPC.com	1300	1	35.00	1300	2015 Q4 Product Plan	Marketing	150.00	8/10/2015	9/15/2015
8	Tom	Jackson	Production	360-287-8610	Tom.Jackson@WPC.com	1300	8	80.00	1300	2015 Q4 Product Plan	Marketing	150.00	8/10/2015	9/15/2015
10	Ken	Numoto	Marketing	360-287-8710	Ken.Mumoto@WPC.com	1300	10	50.00	1300	2015 Q4 Product Plan	Marketing	150.00	8/10/2015	9/15/2015
4	Tom	Caruthers	Accounting	360-285-8310	Tom.Caruthers@WPC.com	1400	4	15.00	1400	2015 Q4 Portfolio Analysis	Finance	140.00	10/5/2015	
5	Heather	Jones	Accounting	360-285-8320	Heather.Jones@WPC.com	1400	5	10.00	1400	2015 Q4 Portfolio Analysis	Finance	140.00	10/5/2015	
6	Mary	Abernathy	Finance	360-285-8410	Mary.Abernathy@WPC.com	1400	6	27.50	1400	2015 Q4 Portfolio Analysis	Finance	140.00	10/5/2015	

N. *Write an SQL statement to join EMPLOYEE and ASSIGNMENT and include all rows of EMPLOYEE in your answer, regardless of whether they have an ASSIGNMENT. Run this statement.*

```
SELECT   E.*, A.*
FROM  EMPLOYEE AS E LEFT JOIN ASSIGNMENT AS A
      ON E.EmployeeNumber = A.EmployeeNumber;
```



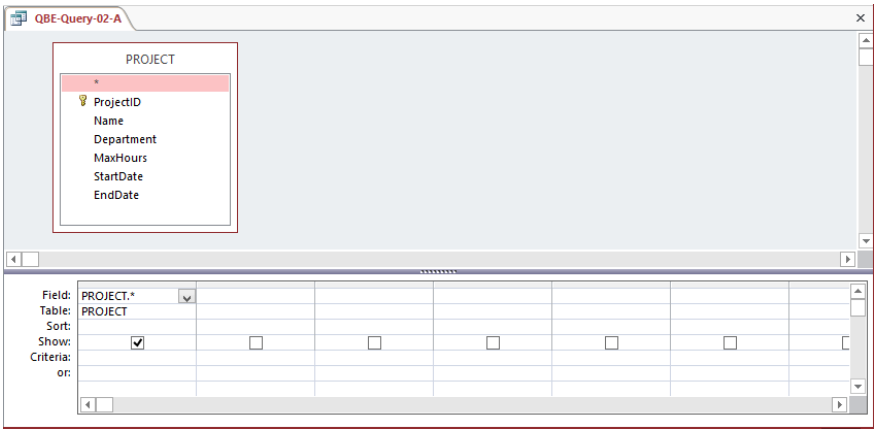
E.EmployeeNumber	FirstName	LastName	Department	Phone	Email	ProjectID	A.EmployeeNumber	HoursWorked
1	Mary	Jacobs	Administration	360-285-8110	Mary.Jacobs@WPC.com	1000	1	30.00
1	Mary	Jacobs	Administration	360-285-8110	Mary.Jacobs@WPC.com	1100	1	25.00
1	Mary	Jacobs	Administration	360-285-8110	Mary.Jacobs@WPC.com	1300	1	35.00
2	Rosalie	Jackson	Administration	360-285-8120	Rosalie.Jackson@WPC.com	1200	2	20.00
3	Richard	Bandalone	Legal	360-285-8210	Richard.Bandalone@WPC.com			
4	Tom	Caruthers	Accounting	360-285-8310	Tom.Caruthers@WPC.com	1100	4	40.00
4	Tom	Caruthers	Accounting	360-285-8310	Tom.Caruthers@WPC.com	1200	4	45.00
4	Tom	Caruthers	Accounting	360-285-8310	Tom.Caruthers@WPC.com	1400	4	15.00
5	Heather	Jones	Accounting	360-285-8320	Heather.Jones@WPC.com	1200	5	40.00
5	Heather	Jones	Accounting	360-285-8320	Heather.Jones@WPC.com	1400	5	10.00
6	Mary	Abernathy	Finance	360-285-8410	Mary.Abernathy@WPC.com	1100	6	45.00
6	Mary	Abernathy	Finance	360-285-8410	Mary.Abernathy@WPC.com	1400	6	27.50
7	George	Smith	Human Resources	360-285-8510	George.Smith@WPC.com			
8	Tom	Jackson	Production	360-287-8610	Tom.Jackson@WPC.com	1000	8	75.00
8	Tom	Jackson	Production	360-287-8610	Tom.Jackson@WPC.com	1300	8	80.00
9	George	Jones	Production	360-287-8620	George.Jones@WPC.com			
10	Ken	Numoto	Marketing	360-287-8710	Ken.Mumoto@WPC.com	1000	10	55.00
10	Ken	Numoto	Marketing	360-287-8710	Ken.Mumoto@WPC.com	1300	10	50.00
11	James	Nestor	InfoSystems		James.Nestor@WPC.com			
12	Rick	Brown	InfoSystems	360-287-8820	Rick.Brown@WPC.com			
(New)								

2.70 Using Microsoft Access QBE, create and run new queries to answer the questions in exercise 2.69. 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.

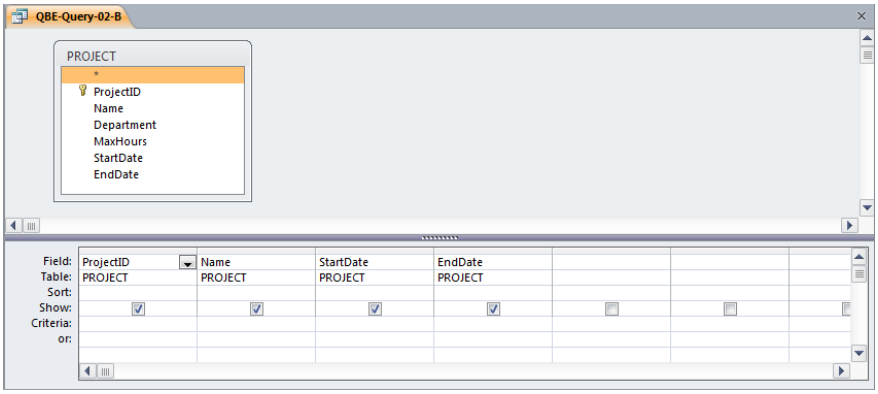
Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e14-IM-CH02-WPC.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

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

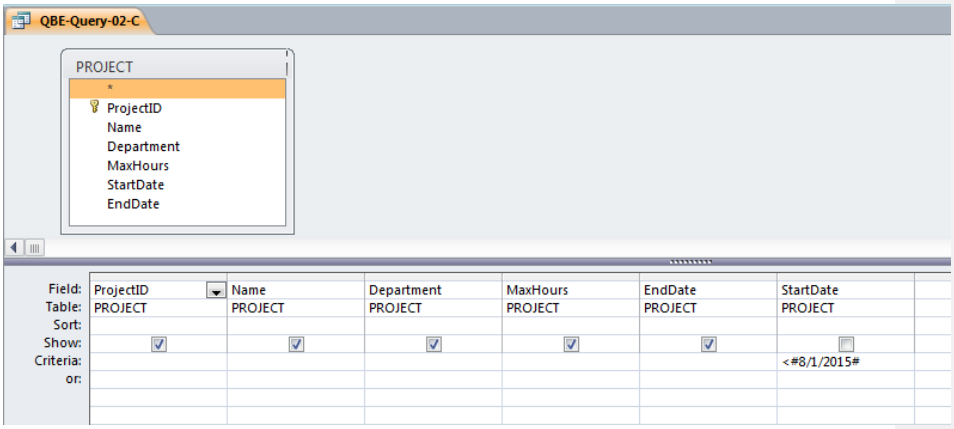
A. What projects are in the PROJECT table? Show all information for each project.



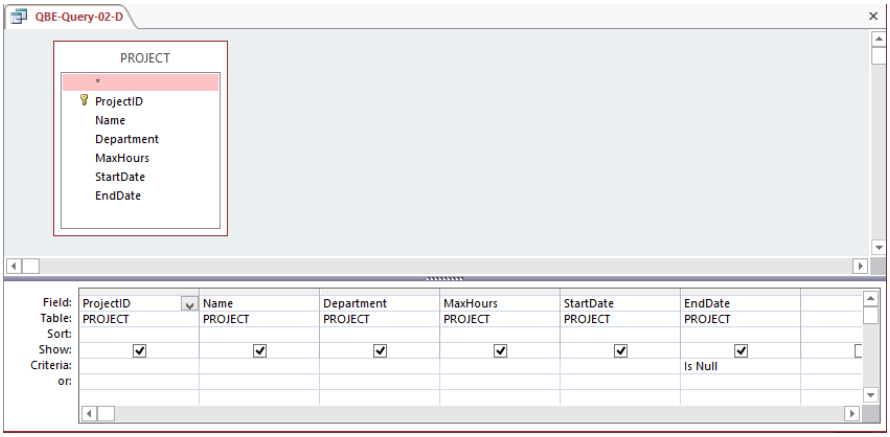
B. What are the ProjectID, Name, StartDate, and EndDate values of projects in the PROJECT table?



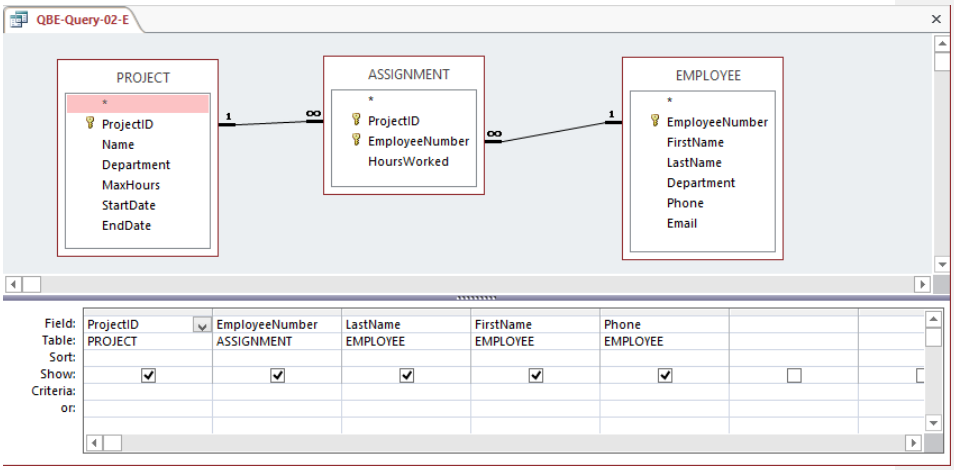
C. *What projects in the PROJECT table started before August 1, 2015? Show all the information for each project.*



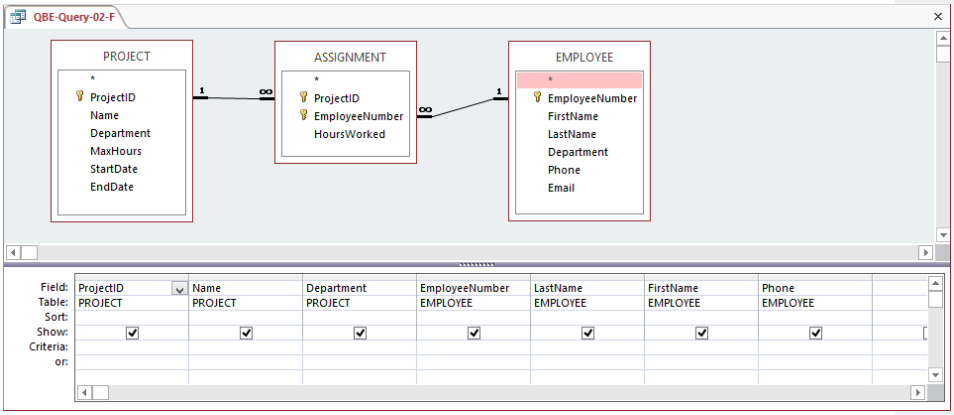
D. *What projects in the PROJECT table have not been completed? Show all the information for each project.*



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

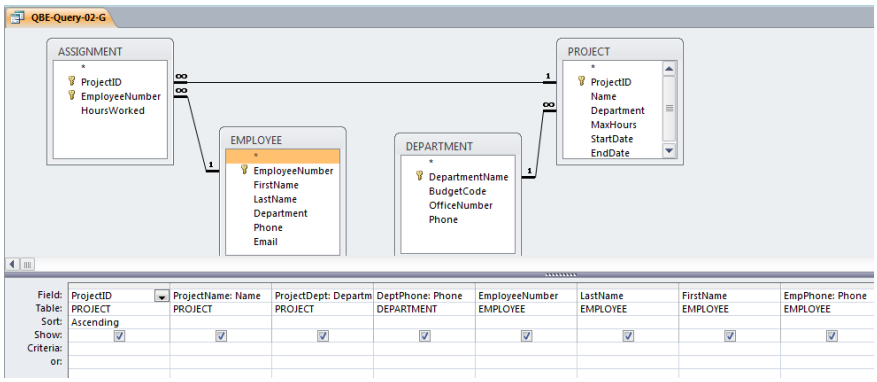


F. Who are the employees assigned to each project? Show ProjectID, Name, and Department. Show EmployeeNumber, LastName, FirstName, and Phone.



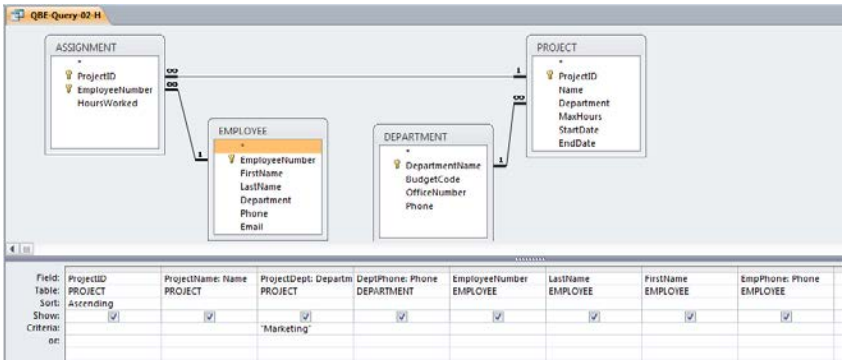
G. Who are the employees assigned to each project? Show ProjectID, Name, Department, and Department Phone. Show EmployeeNumber, LastName, FirstName, and Employee Phone. Sort by ProjectID in ascending order.

This question is more complicated than it seems, in that the default approach of "accepting" all the joins in the QBE query yields an incorrect result. Without deleting the join from EMPLOYEE to DEPARTMENT in the query window (as has been done below; right-click on the relationship line from EMPLOYEE to DEPARTMENT and choose "Delete"), this query will only return assignments in which an EMPLOYEE is assigned to a PROJECT that is in the EMPLOYEE's DEPARTMENT.

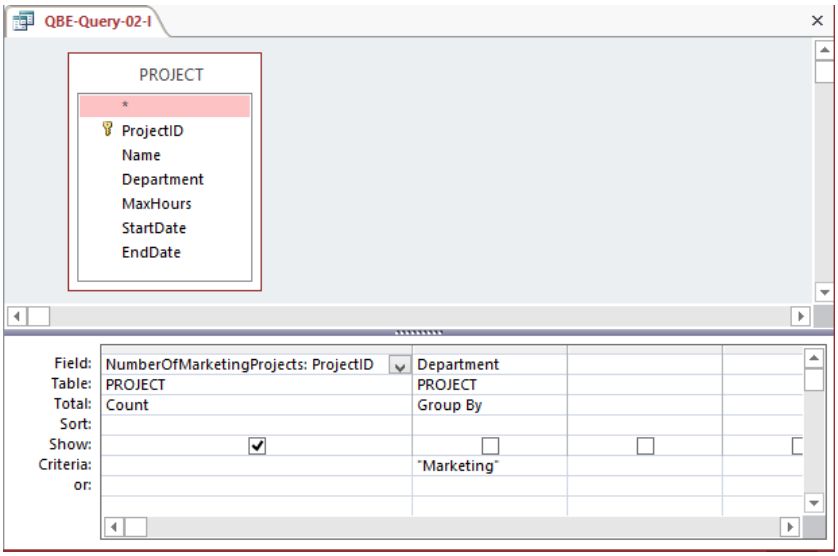


H. Who are the employees assigned to projects run by the marketing department? Show ProjectID, Name, Department, and Department Phone. Show EmployeeNumber, LastName, FirstName, and Employee Phone. Sort by ProjectID in ascending order.

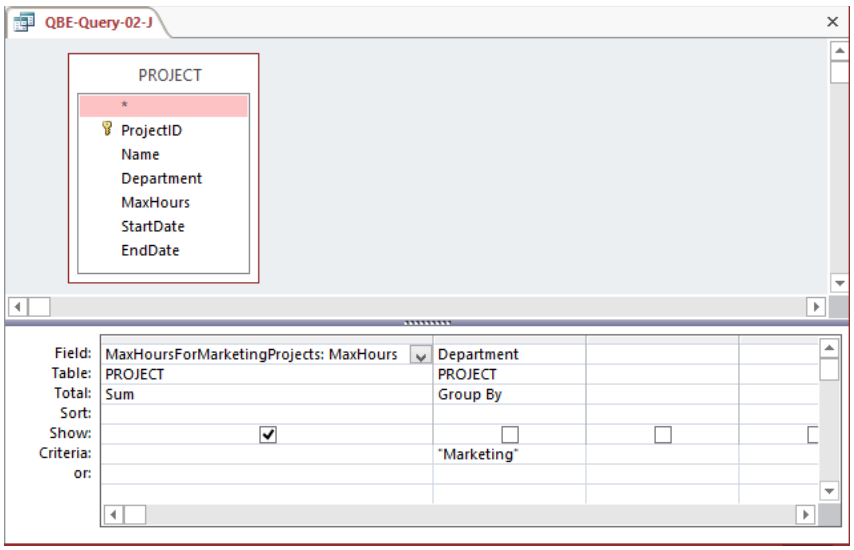
This question is identical to question G except for the restriction to marketing department projects. And, again, this question is more complicated than it seems, in that the default approach of "accepting" all the joins in the QBE query yields an incorrect result. Without deleting the join from EMPLOYEE to DEPARTMENT in the query window (as has been done below; right-click on the relationship line from EMPLOYEE to DEPARTMENT and choose "Delete"), this query will only return assignments in which an EMPLOYEE is assigned to a PROJECT that is in the EMPLOYEE's DEPARTMENT.



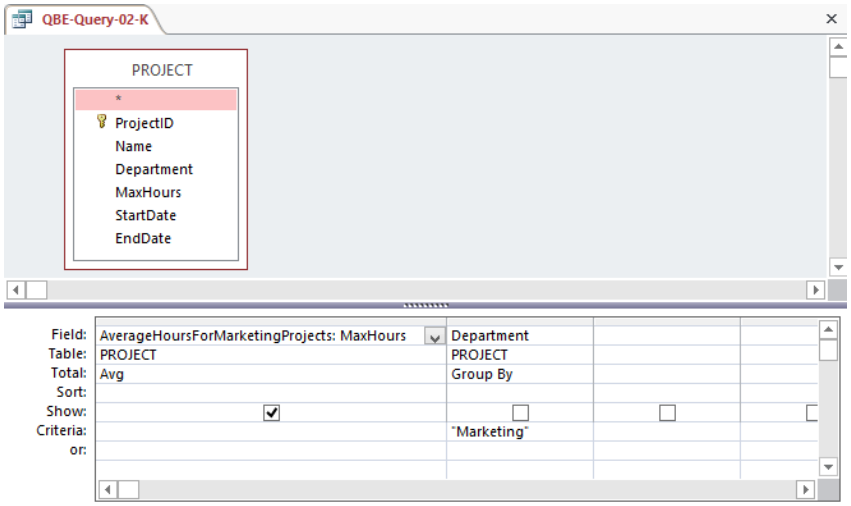
I. How many projects are being run by the marketing department? Be sure to assign an appropriate column name to the computed results.



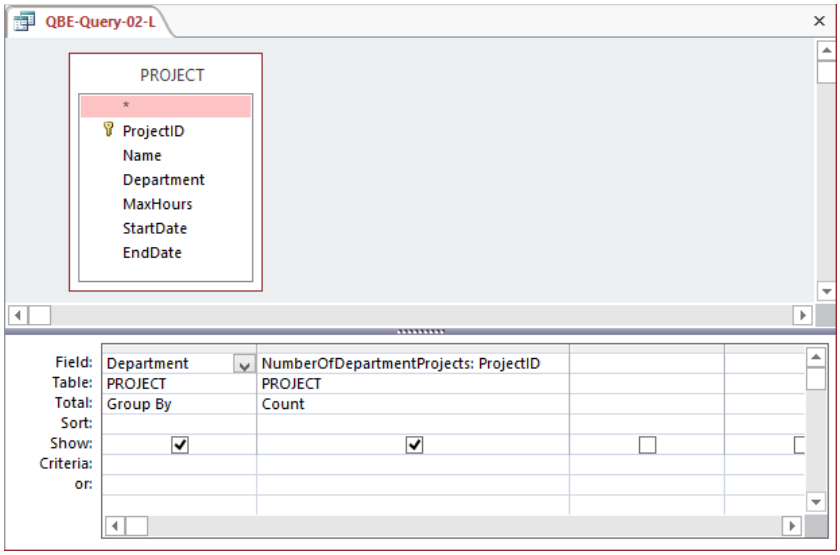
J. What is the total MaxHours of projects being run by the marketing department? Be sure to assign an appropriate column name to the computed results.



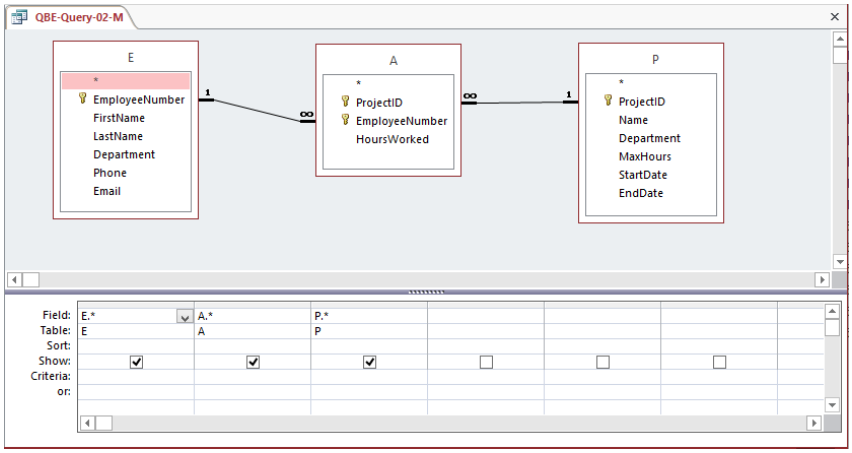
K. *What is the average MaxHours of projects being run by the marketing department? Be sure to assign an appropriate column name to the computed results.*



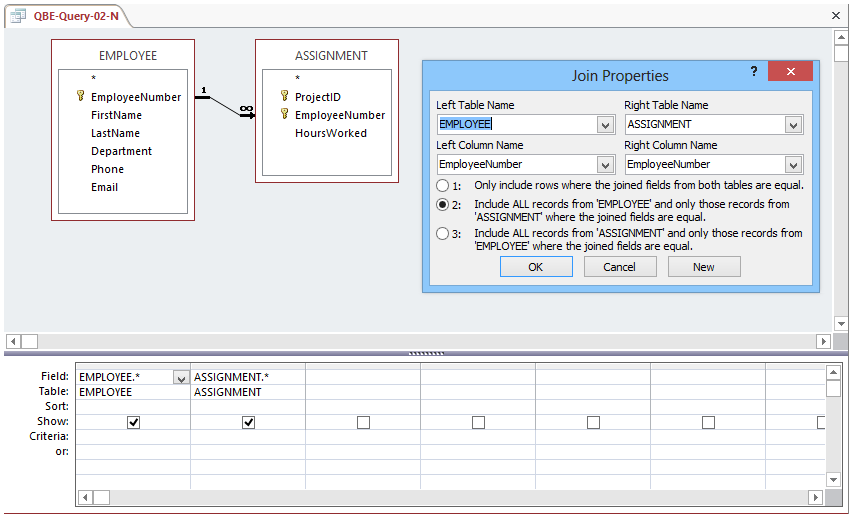
L. *How many projects are being run by each department? Be sure to display each DepartmentName and to assign an appropriate column name to the computed results.*



M. *Write an SQL statement to join EMPLOYEE, ASSIGNMENT, and PROJECT using the JOIN ON syntax. Run this statement.*



N. *Write an SQL statement to join EMPLOYEE and ASSIGNMENT and include all rows of EMPLOYEE in your answer, regardless of whether they have an ASSIGNMENT. Run this statement.*



❖ MARCIA'S DRY CLEANING CASE QUESTIONS

Marcia Wilson owns and operates Marcia's Dry Cleaning, which is an upscale dry cleaner 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 e-mail. 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)

INVOICE (InvoiceNumber, *CustomerNumber*, DateIn, DateOut, TotalAmount)

INVOICE_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 that Marcia has created is named MDC, and the three tables in the MDC database schema are shown in Figure 2-46.

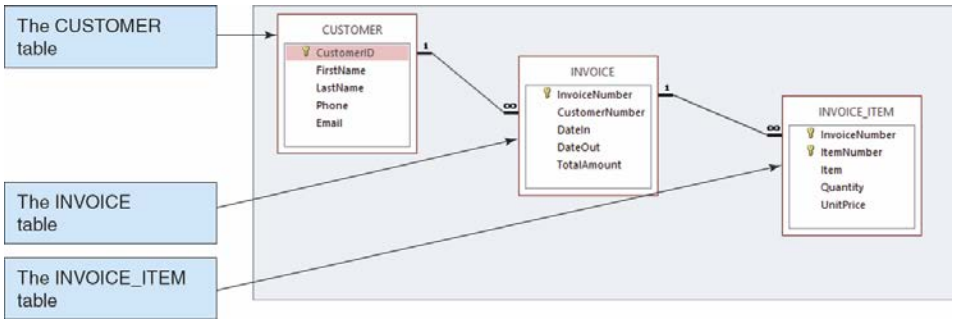


FIGURE 2-46 – The MDC Database

The column characteristics for the tables are shown in Figures 2-47, 2-48, and 2-49. The relationship between CUSTOMER and INVOICE should enforce referential integrity, but not cascade updates nor deletions, while the relationship between INVOICE and INVOICE_ITEM should enforce referential integrity and cascade both updates and deletions. The data for these tables are shown in Figures 2-50, 2-51, and 2-52.

You will need to create and setup a database named MDC-CH02 for use with these case questions. A Microsoft Access 2013 database named MDC_CH02.accdb, and SQL scripts for creating the MDC-CH02 database in Microsoft SQL Server, Oracle Database, and MySQL are available on our Web site at www.pearsonhighered.com/kroenke .

If you are using the Microsoft Access 2013 MDC_CH02.accdb database, simply copy it to an appropriate location in your Documents folder. Otherwise, you will need to use the discussion

and instructions necessary for setting up the MDC_CH02 database in the DBMS product you are using:

- For Microsoft SQL Server 2014, see online Chapter 10A.
- For Oracle Database 12c or Oracle Express Edition 11g Release 2, see online Chapter 10B.
- For MySQL 5.6 Community Server, see online Chapter 10C.

CUSTOMER

Column Name	Type	Key	Required	Remarks
CustomerID	Integer	Primary Key	Yes	Surrogate Key
FirstName	Character (25)	No	Yes	
LastName	Character (25)	No	Yes	
Phone	Character (12)	No	No	
Email	Character (100)	No	No	Use Varchar

Figure 2-47 - Column Characteristics for the CUSTOMER Table

INVOICE

Column Name	Type	Key	Required	Remarks
InvoiceNumber	Integer	Primary Key	Yes	Surrogate Key
CustomerNumber	Integer	Foreign Key	Yes	REF: CUSTOMER
DateIn	Date	No	Yes	
DateOut	Date	No	No	
TotalAmount	Number (8,2)	No	No	

Figure 2-48 - Column Characteristics for the INVOICE Table

INVOICE_ITEM

Column Name	Type	Key	Required	Remarks
InvoiceNumber	Integer	Primary Key, Foreign Key	Yes	REF: INVOICE
ItemNumber	Integer	Primary Key	Yes	Sequential number, but *not* a surrogate key
Item	Character (50)	No	Yes	
Quantity	Integer	No	Yes	
UnitPrice	Number (8,2)	No	Yes	

Figure 2-49 - Column Characteristics for the INVOICE_ITEM Table

CustomerID	FirstName	LastName	Phone	Email
1	Nikki	Kaccaton	723-543-1233	Nikki.Kaccaton@somewhere.com
2	Brenda	Catnazaro	723-543-2344	Brenda.Catnazaro@somewhere.com
3	Bruce	LeCat	723-543-3455	Bruce.LeCat@somewhere.com
4	Betsy	Miller	725-654-3211	Betsy.Miller@somewhere.com
5	George	Miller	725-654-4322	George.Miller@somewhere.com
6	Kathy	Miller	723-514-9877	Kathy.Miller@somewhere.com
7	Betsy	Miller	723-514-8766	Betsy.Miller@elsewhere.com

Figure 2-50 - Sample Data for the MDC Database CUSTOMER table

InvoiceNumber	CustomerNumber	DateIn	DateOut	TotalAmount
2015001	1	04-Oct-15	06-Oct-15	$158.50
2015002	2	04-Oct-15	06-Oct-15	$25.00
2015003	1	06-Oct-15	08-Oct-15	$49.00
2015004	4	06-Oct-15	08-Oct-15	$17.50
2015005	6	07-Oct-15	11-Oct-15	$12.00
2015006	3	11-Oct-15	13-Oct-15	$152.50
2015007	3	11-Oct-15	13-Oct-15	$7.00
2015008	7	12-Oct-15	14-Oct-15	$140.50
2015009	5	12-Oct-15	14-Oct-15	$27.00

Figure 2-51 - Sample Data for the MDC Database INVOICE table

InvoiceNumber	ItemNumber	Item	Quantity	UnitPrice
2015001	1	Blouse	2	$3.50
2015001	2	Dress Shirt	5	$2.50
2015001	3	Formal Gown	2	$10.00
2015001	4	Slacks-Mens	10	$5.00
2015001	5	Slacks-Womens	10	$6.00
2015001	6	Suit-Mens	1	$9.00
2015002	1	Dress Shirt	10	$2.50
2015003	1	Slacks-Mens	5	$5.00
2015003	2	Slacks-Womens	4	$6.00
2015004	1	Dress Shirt	7	$2.50
2015005	1	Blouse	2	$3.50
2015005	2	Dress Shirt	2	$2.50
2015006	1	Blouse	5	$3.50
2015006	2	Dress Shirt	10	$2.50
2015006	3	Slacks-Mens	10	$5.00
2015006	4	Slacks-Womens	10	$6.00
2015007	1	Blouse	2	$3.50
2015008	1	Blouse	3	$3.50
2015008	2	Dress Shirt	12	$2.50
2015008	3	Slacks-Mens	8	$5.00
2015008	4	Slacks-Womens	10	$6.00
2015009	1	Suit-Mens	3	$9.00

Figure 2-52 - Sample Data for the MDC Database INVOICE_ITEM table

Once you have setup your MDC_CH02 database, create an SQL script name MDC-CH02-CQ.sql, and use it to record and store SQL statements that answer each of the following questions (if the question requires a written answer, use and SQL comment to record your answer):

A. Show all data in each of the tables.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-A-CUSTOMER *** */

SELECT   *
FROM     CUSTOMER;
```

Note there are two customers both named Betsy Miller.

SQL-Query-MDC-A-CUSTOMER

CustomerID	FirstName	LastName	Phone	Email
1	Nikki	Kaccaton	723-543-1233	Nikki.Kaccaton@somewhere.com
2	Brenda	Catnazaro	723-543-2344	Brenda.Catnazaro@somewhere.com
3	Bruce	LeCat	723-543-3455	Bruce.LeCat@somewhere.com
4	Betsy	Miller	725-654-3211	Betsy.Miller@somewhere.com
5	George	Miller	725-654-4322	George.Miller@somewhere.com
6	Kathy	Miller	723-514-9877	Kathy.Miller@somewhere.com
7	Betsy	Miller	723-514-8766	Betsy.Miller@elsewhere.com

Record: 1 of 7 No Filter Search

```
/* *** SQL-Query-MDC-A-INVOICE *** */

SELECT   *
FROM     INVOICE;
```

SQL-Query-MDC-A-INVOICE

InvoiceNumber	CustomerNumber	DateIn	DateOut	TotalAmount
2015001	1	10/4/2015	10/6/2015	$158.50
2015002	2	10/4/2015	10/6/2015	$25.00
2015003	1	10/6/2015	10/8/2015	$49.00
2015004	4	10/6/2015	10/8/2015	$17.50
2015005	6	10/7/2015	10/11/2015	$12.00
2015006	3	10/11/2015	10/13/2015	$152.50
2015007	3	10/11/2015	10/13/2015	$7.00
2015008	7	10/12/2015	10/14/2015	$140.50
2015009	5	10/12/2015	10/14/2015	$27.00

Record: 1 of 9 No Filter Search

```
/* *** SQL-Query-MDC-A-INVOICE-ITEM *** */

SELECT   *
FROM     INVOICE_ITEM;
```

SQL-Query-MDC-A-INVOICE-ITEM

InvoiceNumber	ItemNumber	Item	Quantity	UnitPrice
2015001	1	Blouse	2	$3.50
2015001	2	Dress Shirt	5	$2.50
2015001	3	Formal Gown	2	$10.00
2015001	4	Slacks-Mens	10	$5.00
2015001	5	Slacks-Womens	10	$6.00
2015001	6	Suit-Mens	1	$9.00
2015002	1	Dress Shirt	10	$2.50
2015003	1	Slacks-Mens	5	$5.00
2015003	2	Slacks-Womens	4	$6.00
2015004	1	Dress Shirt	7	$2.50
2015005	1	Blouse	2	$3.50
2015005	2	Dress Shirt	2	$2.50
2015006	1	Blouse	5	$3.50
2015006	2	Dress Shirt	10	$2.50
2015006	3	Slacks-Mens	10	$5.00
2015006	4	Slacks-Womens	10	$6.00
2015007	1	Blouse	2	$3.50
2015008	1	Blouse	3	$3.50
2015008	2	Dress Shirt	12	$2.50
2015008	3	Slacks-Mens	8	$5.00
2015008	4	Slacks-Womens	10	$6.00
2015009	1	Suit-Mens	3	$9.00

Record: 22 of 22 No Filter Search

B. List the LastName, FirstName, and Phone of all customers.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-B *** */

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

SQL-Query-MDC-B

LastName	FirstName	Phone
Kaccaton	Nikki	723-543-1233
Catnazaro	Brenda	723-543-2344
LeCat	Bruce	723-543-3455
Miller	Betsy	725-654-3211
Miller	George	725-654-4322
Miller	Kathy	723-514-9877
Miller	Betsy	723-514-8766

Record: 1 of 7 No Filter Search

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

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-C *** */

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

SQL-Query-MDC-C

LastName	FirstName	Phone
Kaccaton	Nikki	723-543-1233

Record: 1 of 1 No Filter Search

D. List the LastName, FirstName, Phone, DateIn, and DateOut of all orders in excess of $100.00.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-D *** */

SELECT   LastName, FirstName, Phone, DateIn, DateOut
FROM     CUSTOMER, INVOICE
WHERE    TotalAmount > 100
   AND   CUSTOMER.CustomerID = INVOICE.CustomerNumber;
```

SQL-Query-MDC-D

LastName	FirstName	Phone	DateIn	DateOut
Kaccaton	Nikki	723-543-1233	10/4/2015	10/6/2015
LeCat	Bruce	723-543-3455	10/11/2015	10/13/2015
Miller	Betsy	723-514-8766	10/12/2015	10/14/2015

Record: 1 of 3 No Filter Search

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

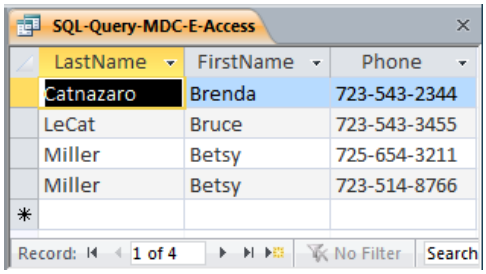
Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

The correct SQL-92 statement for Oracle Database, SQL Server, and MySQL, which uses the wildcard %, is:

```
/* *** SQL-Query-MDC-E *** */

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

/* *** SQL-Query-MDC-E-Access *** */
```

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

```
/* *** SQL-Query-MDC-E-Access *** */

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



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

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that LIKE comparisons will not always work the way you expect: You need to understand when the comparisons are case-sensitive and when they are not. Before running any query involving LIKE, run a small test query to determine whether your DBMS as configured by your DBA is comparing with case sensitivity or not. If you are using Oracle Database, MySQL, or SQL Server, there are ways to force a LIKE comparison to be case-sensitive or case-insensitive; those details are beyond the scope of this text. Microsoft Access, by default, is case-insensitive. To do a case-sensitive LIKE comparison in Microsoft Access, use the "instr" function instead of "LIKE" (see DBP-e14-IM-CH02-MDC.accdb for the solution).

The previous paragraph explains why, in general, you may get different results than those presented below for Access (the Access results are for a default, case-insensitive query). If you are using a DBMS in which the comparisons are case-sensitive, then only the first row in the results below will appear.

The correct SQL-92 statement, for Oracle Database, MySQL, and SQL Server, which uses the wildcard %, is:

```
/* *** SQL-Query-MDC-F *** */

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

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

```
/* *** SQL-Query-MDC-F-Access *** */

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

SQL-Query-MDC-F-Access

LastName	FirstName	Phone
Kaccaton	Nikki	723-543-1233
Catnazaro	Brenda	723-543-2344
LeCat	Bruce	723-543-3455

Record: 1 of 3 No Filter Search

G. *List the LastName, FirstName, and Phone for all customers whose second and third digits (from the left) of their phone number are 23. For example, any phone number with an area code of '723' would meet the criteria.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at 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, of 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:

```
/* *** SQL-Query-MDC-G *** */

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

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

```
/* *** SQL-Query-MDC-G-Access *** */

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

SQL-Query-MDC-G-Access

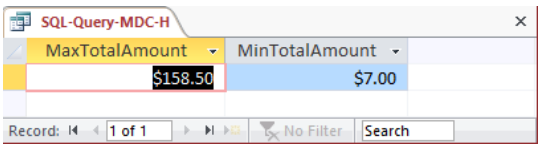
LastName	FirstName	Phone
Kaccaton	Nikki	723-543-1233
Catnazaro	Brenda	723-543-2344
LeCat	Bruce	723-543-3455
Miller	Kathy	723-514-9877
Miller	Betsy	723-514-8766

Record: 1 of 5 No Filter Search

H. Determine the maximum and minimum TotalAmount.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-H *** */

SELECT   MAX (TotalAmt) AS MaxTotalAmount,
         MIN (TotalAmt) AS MinTotalAmount
FROM     INVOICE;
```

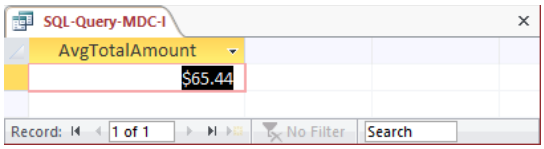


I. Determine the average TotalAmount.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at 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 [ ]).

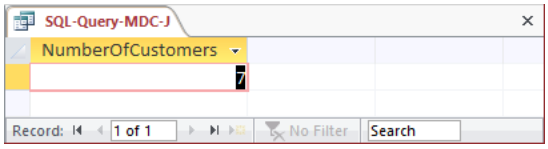
```
/* *** SQL-Query-MDC-I *** */

SELECT   AVG (TotalAmt) AS AvgTotalAmount
FROM     INVOICE;
```



J. Count the number of customers.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

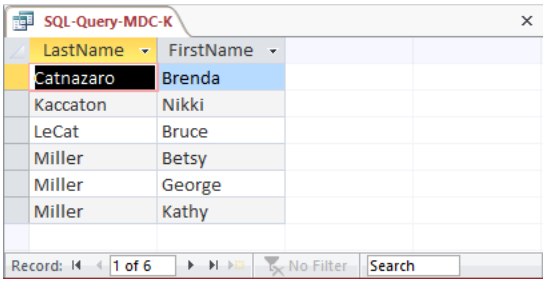
```
/* *** SQL-Query-MDC-J *** */

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 *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-K *** */

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 *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-L *** */

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

SQL-Query-MDC-L

LastName	FirstName	Last_First_Combination_Count
Catnazaro	Brenda	1
Kaccaton	Nikki	1
LeCat	Bruce	1
Miller	Betsy	2
Miller	George	1
Miller	Kathy	1

Record: 1 of 6 No Filter Search

M. Show the LastName, FirstName, and Phone of all customers who have had an order with TotalAmount greater than $100.00. 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 *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-M *** */

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

SQL-Query-MDC-M

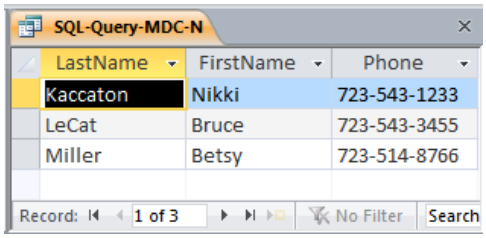
LastName	FirstName	Phone
Kaccaton	Nikki	723-543-1233
LeCat	Bruce	723-543-3455
Miller	Betsy	723-514-8766

Record: 1 of 3 No Filter Search

N. Show the LastName, FirstName and Phone of all customers who have had an order with TotalAmount greater than $100.00. Use a join, but do not use JOIN ON syntax. 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 *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-N *** */

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



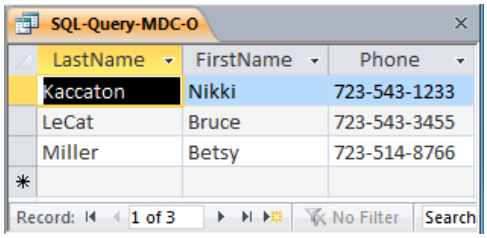
O. Show the LastName, FirstName and Phone of all customers who have had an order with TotalAmount greater than $100.00. Use a join using JOIN ON syntax. Present the results sorted by LastName in ascending order and then FirstName in descending order.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-O *** */

SELECT   CUSTOMER.LastName, CUSTOMER.FirstName, CUSTOMER.Phone
FROM     CUSTOMER JOIN INVOICE
    ON   CUSTOMER.CustomerID = INVOICE.CustomerNumber
WHERE    INVOICE.TotalAmount>100;
```

Note that for Microsoft Access, we must use the INNER JOIN syntax:

```
/* *** SQL-Query-MDC-O *** */

SELECT   CUSTOMER.LastName, CUSTOMER.FirstName, CUSTOMER.Phone
FROM     CUSTOMER INNER JOIN INVOICE
    ON   CUSTOMER.CustomerID = INVOICE.CustomerNumber
WHERE    INVOICE.TotalAmount>100;
```

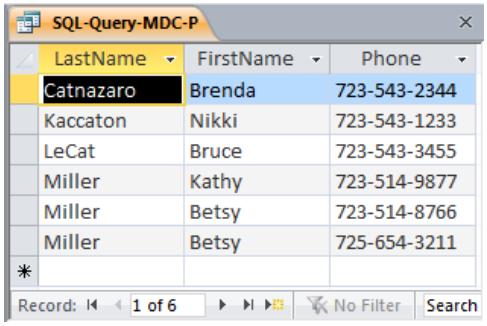


P. *Show the LastName, FirstName and Phone of all customers who have had an order with an Item named "Dress Shirt". Use a subquery. Present the results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note the solution below uses 2 subqueries; other correct solutions are possible that use one subquery and a join (the question does not specify that two subqueries must be used).

```
/* *** SQL-Query-MDC-P *** */

SELECT   LastName, FirstName, Phone
FROM     CUSTOMER
WHERE    CustomerID IN
         (SELECT  CustomerNumber
         FROM     INVOICE
         WHERE    InvoiceNumber IN
                  (SELECT InvoiceNumber
                  FROM INVOICE_ITEM
                  WHERE Item = 'Dress Shirt'))
ORDER BY LastName, FirstName DESC;
```



Q. Show the LastName, FirstName and Phone of all customers who have had an order with an Item named "Dress Shirt". Use a join, but do not use JOIN ON syntax. 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 *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MDC-Q-Access *** */

SELECT   LastName, FirstName, Phone
FROM     CUSTOMER, INVOICE, INVOICE_ITEM
WHERE    CUSTOMER.CustomerID = INVOICE.CustomerNumber
   AND   INVOICE.InvoiceNumber = INVOICE_ITEM.InvoiceNumber
   AND   INVOICE_ITEM.Item = 'Dress Shirt'
ORDER BY LastName, FirstName DESC;
```

SQL-Query-MDC-Q

LastName	FirstName	Phone
Catnazaro	Brenda	723-543-2344
Kaccaton	Nikki	723-543-1233
LeCat	Bruce	723-543-3455
Miller	Kathy	723-514-9877
Miller	Betsy	723-514-8766
Miller	Betsy	725-654-3211

Record: 1 of 6 No Filter Search

R. Show the LastName, FirstName and Phone of all customers who have had an order with an Item named "Dress Shirt". Use a join using JOIN ON syntax. 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 *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

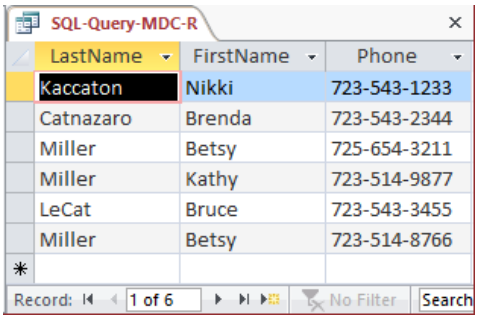
For Oracle Database, SQL Server, and MySQL:

```
/* *** SQL-Query-MDC-R *** */

SELECT   CUSTOMER.LastName, CUSTOMER.FirstName,
         CUSTOMER.Phone
FROM     (CUSTOMER JOIN INVOICE
      ON CUSTOMER.CustomerID = INVOICE.CustomerNumber)
         JOIN INVOICE_ITEM
```

```
                ON INVOICE.InvoiceNumber = INVOICE_ITEM.InvoiceNumber
WHERE    INVOICE_ITEM.Item='Dress Shirt';
```

Note that for Microsoft Access, we must use the INNER JOIN syntax:

```
/* *** SQL-Query-MDC-R-Access *** */

SELECT   CUSTOMER.LastName, CUSTOMER.FirstName,
         CUSTOMER.Phone
FROM     (CUSTOMER INNER JOIN INVOICE
      ON  CUSTOMER.CustomerID = INVOICE.CustomerNumber)
          INNER JOIN INVOICE_ITEM
                ON INVOICE.InvoiceNumber = INVOICE_ITEM.InvoiceNumber
WHERE    INVOICE_ITEM.Item ='Dress Shirt';
```



S. *Show the LastName, FirstName, and Phone of all customers who have had an order with an Item named "Dress Shirt". Use a combination of a join using JOIN ON syntax 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 *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

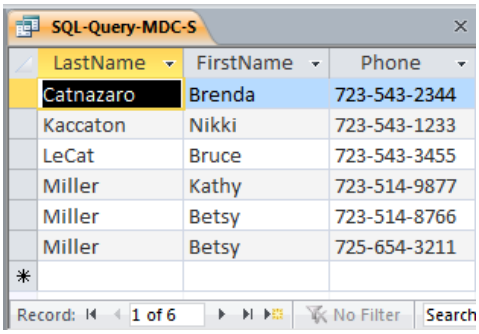
Note that multiple correct solutions are possible here; this solution joins CUSTOMER with INVOICE and uses INVOICE_ITEM by itself in the subquery. Another solution would use CUSTOMER by itself in the main query then a subquery that contains a join of INVOICE and INVOICE_ITEM. Both versions are presented in the solution files.

For SQL Server, MySQL, and Oracle Database:

```
/* *** SQL-Query-MDC *** */

SELECT   LastName, FirstName, Phone
FROM     CUSTOMER JOIN INVOICE
ON       CUSTOMER.CustomerID = INVOICE.CustomerNumber
   WHERE INVOICE.InvoiceNumber IN
                  (SELECT InvoiceNumber
                   FROM INVOICE_ITEM
                   WHERE Item = 'Dress Shirt')
ORDER BY LastName, FirstName DESC;
```

The Access version requires the "INNER JOIN" syntax:

```
/* *** SQL-Query-MDC-Access *** */

SELECT   LastName, FirstName, Phone
FROM     CUSTOMER INNER JOIN INVOICE
ON       CUSTOMER.CustomerID = INVOICE.CustomerNumber
   WHERE INVOICE.InvoiceNumber IN
                  (SELECT InvoiceNumber
                   FROM INVOICE_ITEM
                   WHERE Item = 'Dress Shirt')
ORDER BY LastName, FirstName DESC;
```



T. *Show the LastName, FirstName, Phone, and TotalAmount of all customer orders that included an Item named "Dress Shirt". Also show the LastName, FirstName, and Phone of* all other customers. *Present results sorted by TotalAmount in ascending order, then LastName in ascending order, and then FirstName in descending order.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that this is a very challenging question! The best solution involves adding the 'Dress Shirt' restriction to the inner JOIN before performing the LEFT JOIN, otherwise (if we put the 'Dress Shirt' restriction in the WHERE clause) every customer will have an invoice so the LEFT JOIN will not produce any NULLs, and we will get an incorrect result from the query. Examples of this are not covered in the text, but at the same time, the text does not say you can't do it either.

The LEFT JOIN solution for Oracle Database, MySQL, and SQL Server:

```
/* *** SQL-Query-MDC-T *** */

SELECT LastName, FirstName, Phone, TotalAmount
FROM CUSTOMER LEFT JOIN (INVOICE JOIN INVOICE_ITEM
   ON INVOICE.InvoiceNumber = INVOICE_ITEM.InvoiceNumber AND
         INVOICE_ITEM.Item = 'Dress Shirt')
  ON CustomerID = CustomerNumber
ORDER BY TotalAmount, LastName, FirstName DESC;
```

Note that Microsoft Access does not allow nesting an INNER JOIN inside a LEFT or RIGHT JOIN. It also disallows adding the non-join condition to the "ON" clause. So in order to create a solution in Access, we must either (1) use a more complicated version of the query with a UNION but without an OUTER JOIN or (2) create and save an intermediate query (view) to be used in the final query. Note that these two approaches will also work with Oracle, SQL Server, or MySQL.

```
/* *** SQL-Query-MDC-T-UNION *** */

SELECT LastName, FirstName, Phone, TotalAmount
FROM CUSTOMER C, INVOICE I, INVOICE_ITEM II
WHERE C.CustomerID = I.CustomerNumber AND I.InvoiceNumber =
   II.InvoiceNumber AND II.Item = 'Dress Shirt'
UNION SELECT LastName, FirstName, Phone, NULL
FROM CUSTOMER
WHERE CustomerID NOT IN
   (SELECT CustomerNumber
    FROM INVOICE I, INVOICE_ITEM II
    WHERE I.InvoiceNumber = II.InvoiceNumber AND II.Item = 'Dress
Shirt')
ORDER BY TotalAmount, LastName, FirstName DESC;
```

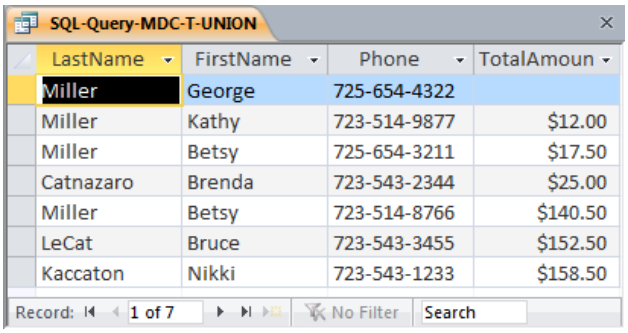
The other approach using Access involves writing and saving an intermediate query (also called a "view"; see Chapter 7). We first write and save a query that produces the CustomerNumber and TotalAmount for all invoices involving a 'Dress Shirt':

```
/* *** SQL-Query-MDC-T-Temp *** */

SELECT CustomerNumber, TotalAmount
FROM INVOICE I, INVOICE_ITEM II
WHERE I.InvoiceNumber = II.InvoiceNumber AND II.Item = 'Dress
Shirt';
```

Now we can use that temporary query as if it were just another table to produce the final result:

```
/* *** SQL-Query-MDC-T-Final *** */

SELECT LastName, FirstName, Phone, TotalAmount
FROM CUSTOMER AS C LEFT OUTER JOIN [SQL-Query-MDC-T-Temp] AS T
  ON C.CustomerID = T.CustomerNumber
ORDER BY TotalAmount, LastName, FirstName DESC;
```

The results below are the same for all correct versions of this query, with the possible exception of where the NULL TotalAmounts are presented: In Access, NULL comes before all values; in Oracle, it comes last, etc.



ANSWERS TO THE QUEEN ANNE CURIOSITY SHOP PROJECT QUESTIONS

The Queen Anne Curiosity Shop is an upscale home furnishings store in a well-to-do urban neighborhood. It sells both antiques and current-production household items that complement or are useful with the antiques. For example, the store sells antique dining room tables and new tablecloths. The antiques are purchased from both individuals and wholesalers, and the new items are purchased from distributors. The store's customers include individuals, owners of bed-and-breakfast operations, and local interior designers who work with both individuals and small businesses. The antiques are unique, though some multiple items, such as dining room chairs, may be available as a set (sets are never broken). The new items are not unique, and an item may be reordered if it is out of stock. New items are also available in various sizes and colors (for example, a particular style of tablecloth may be available in several sizes and in a variety of colors).

Assume that The Queen Anne Curiosity Shop designs a database with the following tables:

CUSTOMER (CustomerID, LastName, FirstName, Address, City, State, ZIP, Phone, Email)

ITEM (ItemID, ItemDescription, CompanyName, PurchaseDate, ItemCost, ItemPrice)

SALE (SaleID, *CustomerID*, SaleDate, SubTotal, Tax, Total)

SALE_ITEM (*SaleID*, SaleItemID, *ItemID*, ItemPrice)

The referential integrity constraints are:

CustomerID in SALE must exist in CustomerID in CUSTOMER

SaleID in SALE_ITEM must exist in SaleID in SALE

ItemID in SALE_ITEM must exist in ItemID in ITEM

Assume that CustomerID of CUSTOMER, ItemID of ITEM, SaleID of SALE, and SaleItemID of SALE_ITEM are all surrogate keys with values as follows:

CustomerID Start at 1 Increment by 1

ItemID Start at 1 Increment by 1

SaleID Start at 1 Increment by 1

The database that The Queen Anne Curiosity Shop has created is named QACS, and the four tables in the QACS database schema are shown in Figure 2-53.

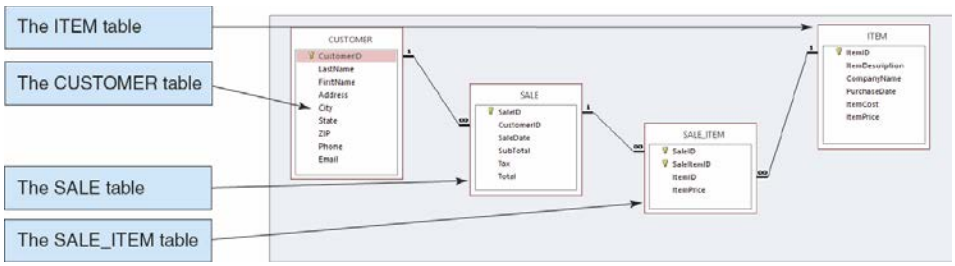


Figure 2-53 – The QACS Database

The column characteristics for the tables are shown in Figures 2-54, 2-55, 2-56, and 2-57. The relationships CUSTOMER-to-SALE and ITEM-to-SALE_ITEM should enforce referential integrity, but not cascade updates nor deletions, while the relationship between SALE and SALE_ITEM should enforce referential integrity and cascade both updates and deletions. The data for these tables are shown in Figures 2-58, 2-59, 2-60, and 2-61.

CUSTOMER

Column Name	Type	Key	Required	Remarks
CustomerID	Integer	Primary Key	Yes	Surrogate Key
LastName	Character (25)	No	Yes	
FirstName	Character (25)	No	Yes	
Address	Character (35)	No	No	
City	Character (35)	No	No	
State	Character (2)	No	No	
ZIP	Character (10)	No	No	
Phone	Character (12)	No	Yes	
Email	Character (100)	No	Yes	Use Varchar

Figure 2-54 - Column Characteristics for the QACS Database CUSTOMER Table

SALE

Column Name	Type	Key	Required	Remarks
SaleID	Integer	Primary Key	Yes	Surrogate Key
CustomerID	Integer	Foreign Key	Yes	REF: CUSTOMER
SaleDate	Date	No	Yes	
SubTotal	Number (15,2)	No	No	
Tax	Number (15,2)	No	No	
Total	Number (15,2)	No	No	

Figure 2-55 - Column Characteristics for the QACS Database SALE Table

SALE_ITEM

Column Name	Type	Key	Required	Remarks
SaleID	Integer	Primary Key, Foreign Key	Yes	REF: SALE
SaleItemID	Integer	Primary Key	Yes	Sequential number, but *not* a surrogate key
ItemID	Integer	Foreign Key	Yes	REF: ITEM
ItemPrice	Number (9,2)	No	No	

Figure 2-56 - Column Characteristics for the QACS Database SALE_ITEM Table

ITEM

Column Name	Type	Key	Required	Remarks
ItemID	Integer	Primary Key	Yes	Surrogate Key
ItemDescription	Character (255)	No	Yes	Use Varchar
CompanyName	Character (100)	No	Yes	
PurchaseDate	Date	No	Yes	
ItemCost	Number (9,2)	No	Yes	
ItemPrice	Number (9,2)	No	Yes	

Figure 2-57 - Column Characteristics for the QACS Database ITEM Table

CustomerID	LastName	FirstName	Address	City	State	ZIP	Phone	Email
1	Shire	Robert	6225 Evanston Ave N	Seattle	WA	98103	206-524-2433	Rober.Shire@somewhere.com
2	Goodyear	Katherine	7335 11th Ave NE	Seattle	WA	98105	206-524-3544	Katherine.Goodyear@somewhere.com
3	Bancroft	Chris	12605 NE 6th Street	Bellevue	WA	98005	425-635-9788	Chris.Bancroft@somewhere.com
4	Griffith	John	335 Aloha Street	Seattle	WA	98109	206-524-4655	John.Griffith@somewhere.com
5	Tierney	Doris	14510 NE 4th Street	Bellevue	WA	98005	425-635-8677	Doris.Tierney@somewhere.com
6	Anderson	Donna	1410 Hillcrest Parkway	Mt. Vernon	WA	98273	360-538-7566	Donna.Anderson@elsewhere.com
7	Svane	Jack	3211 42nd Street	Seattle	WA	98115	206-524-5766	Jack.Svane@somewhere.com
8	Walsh	Denesha	6712 24th Avenue NE	Redmond	WA	98053	425-635-7566	Denesha.Walsh@somewhere.com
9	Enquist	Craig	534 15th Street	Bellingham	WA	98225	360-538-6455	Craig.Enquist@elsewhere.com
10	Anderson	Rose	6823 17th Ave NE	Seattle	WA	98105	206-524-6877	Rose.Anderson@elsewhere.com

Figure 2-58 – Sample Data for the QACS Database CUSTOMER Table

SaleID	CustomerID	SaleDate	SubTotal	Tax	Total
1	1	12/14/2014	$3,500.00	$290.50	$3,790.50
2	2	12/15/2014	$1,000.00	$83.00	$1,083.00
3	3	12/15/2014	$50.00	$4.15	$54.15
4	4	12/23/2014	$45.00	$3.74	$48.74
5	1	1/5/2015	$250.00	$20.75	$270.75
6	5	1/10/2015	$750.00	$62.25	$812.25
7	6	1/12/2015	$250.00	$20.75	$270.75
8	2	1/15/2015	$3,000.00	$249.00	$3,249.00
9	5	1/25/2015	$350.00	$29.05	$379.05
10	7	2/4/2015	$14,250.00	$1,182.75	$15,432.75
11	8	2/4/2015	$250.00	$20.75	$270.75
12	5	2/7/2015	$50.00	$4.15	$54.15
13	9	2/7/2015	$4,500.00	$373.50	$4,873.50
14	10	2/11/2015	$3,675.00	$305.03	$3,980.03
15	2	2/11/2015	$800.00	$66.40	$866.40

Figure 2-59 - Sample Data for the QACS Database SALE Table

SaleID	SaleItemID	ItemID	ItemPrice
1	1	1	$3,000.00
1	2	2	$500.00
2	1	3	$1,000.00
3	1	4	$50.00
4	1	5	$45.00
5	1	6	$250.00
6	1	7	$750.00
7	1	8	$250.00
8	1	9	$1,250.00
8	2	10	$1,750.00
9	1	11	$350.00
10	1	19	$5,000.00
10	2	21	$8,500.00
10	3	22	$750.00
11	1	17	$250.00
12	1	24	$50.00
13	1	20	$4,500.00
14	1	12	$3,200.00
14	2	14	$475.00
15	1	23	$800.00

Figure 2-60 - Sample Data for the QACS Database SALE_ITEM Table

ItemID	ItemDescription	CompanyName	PurchaseDate	ItemCost	ItemPrice
1	Antique Desk	European Specialties	11/7/2014	$1,800.00	$3,000.00
2	Antique Desk Chair	Andrew Lee	11/10/2014	$300.00	$500.00
3	Dining Table Linens	Linens and Things	11/14/2014	$600.00	$1,000.00
4	Candles	Linens and Things	11/14/2014	$30.00	$50.00
5	Candles	Linens and Things	11/14/2014	$27.00	$45.00
6	Desk Lamp	Lamps and Lighting	11/14/2014	$150.00	$250.00
7	Dining Table Linens	Linens and Things	11/14/2014	$450.00	$750.00
8	Book Shelf	Denise Harrion	11/21/2014	$150.00	$250.00
9	Antique Chair	New York Brokerage	11/21/2014	$750.00	$1,250.00
10	Antique Chair	New York Brokerage	11/21/2014	$1,050.00	$1,750.00
11	Antique Candle Holder	European Specialties	11/28/2014	$210.00	$350.00
12	Antique Desk	European Specialties	1/5/2015	$1,920.00	$3,200.00
13	Antique Desk	European Specialties	1/5/2015	$2,100.00	$3,500.00
14	Antique Desk Chair	Specialty Antiques	1/6/2015	$285.00	$475.00
15	Antique Desk Chair	Specialty Antiques	1/6/2015	$339.00	$565.00
16	Desk Lamp	General Antiques	1/6/2015	$150.00	$250.00
17	Desk Lamp	General Antiques	1/6/2015	$150.00	$250.00
18	Desk Lamp	Lamps and Lighting	1/6/2015	$144.00	$240.00
19	Antique Dining Table	Denesha Walsh	1/10/2015	$3,000.00	$5,000.00
20	Antique Sideboard	Chris Bancroft	1/11/2015	$2,700.00	$4,500.00
21	Dining Table Chairs	Specialty Antiques	1/11/2015	$5,100.00	$8,500.00
22	Dining Table Linens	Linens and Things	1/12/2015	$450.00	$750.00
23	Dining Table Linens	Linens and Things	1/12/2015	$480.00	$800.00
24	Candles	Linens and Things	1/17/2015	$30.00	$50.00
25	Candles	Linens and Things	1/17/2015	$36.00	$60.00

Figure 2-61 - Sample Data for the QACS Database ITEM Table

You will need to create and setup a database named QACS_CH02 for use with The Queen Anne Curiosity Shop project questions. A Microsoft Access 2013 database named QACS_CH02.accdb, and SQL scripts for creating the QACS_CH02 database in Microsoft SQL Server, Oracle Database, and MySQL are available on our Web site at www.pearsonhighered.com/kroenke.

If you are using the Microsoft Access 2013 QACS_CH02.accdb database, simply copy it to an appropriate location in your Documents folder. Otherwise, you will need to use the discussion and instructions necessary for setting up the QACS_CH02 database in the DBMS product you are using:

- *For Microsoft SQL Server 2014, see online Chapter 10A.*
- *For Oracle Database 12c or Oracle Express Edition 11g Release 2, see online Chapter 10B.*
- *For MySQL 5.6 Community Server, see online Chapter 10C.*

Once you have setup your QACS_CH02 database, create an SQL script named QACSCH02-CQ.sql, and use it to record and store SQL statements that answer each of the following questions (if the question requires a written answer, use an SQL comment to record your answer):

A. Show all data in each of the tables.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-QACS-A-CUSTOMER *** */

SELECT   *
FROM     CUSTOMER;
```

SQL-Query-QACS-A-CUSTOMER

CustomerID	LastName	FirstName	Address	City	State	ZIP	Phone	Email
1	Shire	Robert	6225 Evanston	Seattle	WA	98103	206-524-2433	Robert.Shire@somewhere.com
2	Goodyear	Katherine	7335 11th Ave I	Seattle	WA	98105	206-524-3544	Katherine.Goodyear@somewhere.com
3	Bancroft	Chris	12605 NE 6th St	Bellevue	WA	98005	425-635-9788	Chris.Bancroft@somewhere.com
4	Griffith	John	335 Aloha Stre	Seattle	WA	98109	206-524-4655	John.Griffith@somewhere.com
5	Tierney	Doris	14510 NE 4th St	Bellevue	WA	98005	425-635-8677	Doris.Tierney@somewhere.com
6	Anderson	Donna	1410 Hillcrest F	Mt. Vernon	WA	98273	360-538-7566	Donna.Anderson@elsewhere.com
7	Svane	Jack	3211 42nd Stre	Seattle	WA	98115	206-524-5766	Jack.Svane@somewhere.com
8	Walsh	Denesha	6712 24th Aven	Redmond	WA	98053	425-635-7566	Denesha.Walsh@somewhere.com
9	Enquist	Craig	534 15th Street	Bellingham	WA	98225	360-538-6455	Craig.Enquist@elsewhere.com
10	Anderson	Rose	6823 17th Ave I	Seattle	WA	98105	206-524-6877	Rose.Anderson@elsewhere.com
(New)								

Record: 1 of 10 No Filter Search

```
/* *** SQL-Query-QACS-A-SALE *** */

SELECT    *
FROM      SALE;
```

SQL-Query-QACS-A-SALE

SaleID	CustomerID	SaleDate	SubTotal	Tax	Total
1	1	12/14/2014	$3,500.00	$290.50	$3,790.50
2	2	12/15/2014	$1,000.00	$83.00	$1,083.00
3	3	12/15/2014	$50.00	$4.15	$54.15
4	4	12/23/2014	$45.00	$3.74	$48.74
5	1	1/5/2015	$250.00	$20.75	$270.75
6	5	1/10/2015	$750.00	$62.25	$812.25
7	6	1/12/2015	$250.00	$20.75	$270.75
8	2	1/15/2015	$3,000.00	$249.00	$3,249.00
9	5	1/25/2015	$350.00	$29.05	$379.05
10	7	2/4/2015	$14,250.00	$1,182.75	$15,432.75
11	8	2/4/2015	$250.00	$20.75	$270.75
12	5	2/7/2015	$50.00	$4.15	$54.15
13	9	2/7/2015	$4,500.00	$373.50	$4,873.50
14	10	2/11/2015	$3,675.00	$305.03	$3,980.03
15	2	2/11/2015	$800.00	$66.40	$866.40
(New)					

Record: 1 of 15 No Filter Search

```
/* *** SQL-Query-QACS-A-SALE-ITEM *** */

SELECT    *
FROM      SALE_ITEM;
```

SQL-Query-QACS-A-SALE-ITEM

SaleID	SaleItemID	ItemID	ItemPrice
1	1	1	$3,000.00
1	2	2	$500.00
2	1	3	$1,000.00
3	1	4	$50.00
4	1	5	$45.00
5	1	6	$250.00
6	1	7	$750.00
7	1	8	$250.00
8	1	9	$1,250.00
8	2	10	$1,750.00
9	1	11	$350.00
10	1	19	$5,000.00
10	2	21	$8,500.00
10	3	22	$750.00
11	1	17	$250.00
12	1	24	$50.00
13	1	20	$4,500.00
14	1	12	$3,200.00
14	2	14	$475.00
15	1	23	$800.00

Record: 1 of 20 No Filter Search

```
/* *** SQL-Query-QACS-A-ITEM *** */

SELECT   *
FROM     ITEM;
```

SQL-Query-QACS-A-ITEM

ItemID	ItemDescription	CompanyName	PurchaseDate	ItemCost	ItemPrice
1	Antique Desk	European Specialties	11/7/2014	$1,800.00	$3,000.00
2	Antique Desk Chair	Andrew Lee	11/10/2014	$300.00	$500.00
3	Dining Table Linens	Linens and Things	11/14/2014	$600.00	$1,000.00
4	Candles	Linens and Things	11/14/2014	$30.00	$50.00
5	Candles	Linens and Things	11/14/2014	$27.00	$45.00
6	Desk Lamp	Lamps and Lighting	11/14/2014	$150.00	$250.00
7	Dining Table Linens	Linens and Things	11/14/2014	$450.00	$750.00
8	Book Shelf	Denise Harrison	11/21/2014	$150.00	$250.00
9	Antique Chair	New York Brokerage	11/21/2014	$750.00	$1,250.00
10	Antique Chair	New York Brokerage	11/21/2014	$1,050.00	$1,750.00
11	Antique Candle Holder	European Specialties	11/28/2014	$210.00	$350.00
12	Antique Desk	European Specialties	1/5/2015	$1,920.00	$3,200.00
13	Antique Desk	European Specialties	1/5/2015	$2,100.00	$3,500.00
14	Antique Desk Chair	Specialty Antiques	1/6/2015	$285.00	$475.00
15	Antique Desk Chair	Specialty Antiques	1/6/2015	$339.00	$565.00
16	Desk Lamp	General Antiques	1/6/2015	$150.00	$250.00
17	Desk Lamp	General Antiques	1/6/2015	$150.00	$250.00
18	Desk Lamp	Lamps and Lighting	1/6/2015	$144.00	$240.00
19	Antique Dining Table	Denesha Walsh	1/10/2015	$3,000.00	$5,000.00
20	Antique Sideboard	Chris Bancroft	1/11/2015	$2,700.00	$4,500.00
21	Dining Table Chairs	Specialty Antiques	1/11/2015	$5,100.00	$8,500.00
22	Dining Table Linens	Linens and Things	1/12/2015	$450.00	$750.00
23	Dining Table Linens	Linens and Things	1/12/2015	$480.00	$800.00
24	Candles	Linens and Things	1/17/2015	$30.00	$50.00
25	Candles	Linens and Things	1/17/2015	$36.00	$60.00
(New)					

Record: 1 of 25 | No Filter | Search

B. List the LastName, FirstName, and Phone of all customers.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-QACS-B *** */

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

SQL-Query-QACS-B

LastName	FirstName	Phone
Shire	Robert	206-524-2433
Goodyear	Katherine	206-524-3544
Bancroft	Chris	425-635-9788
Griffith	John	206-524-4655
Tierney	Doris	425-635-8677
Anderson	Donna	360-538-7566
Svane	Jack	206-524-5766
Walsh	Denesha	425-635-7566
Enquist	Craig	360-538-6455
Anderson	Rose	206-524-6877

Record: 1 of 10 No Filter Search

C. List the LastName, FirstName, and Phone for all customers with a FirstName of 'John'.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-QACS-C *** */

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

SQL-Query-QACS-C

LastName	FirstName	Phone
Griffith	John	206-524-4655

Record: 1 of 1 No Filter Search

D. List the LastName, FirstName, Phone, SaleDate, and Total of all sales in excess of $100.00.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-QACS-D *** */

SELECT LastName, FirstName, Phone, SaleDate, Total
FROM CUSTOMER, SALE
WHERE CUSTOMER.CustomerID = SALE.CustomerID
  AND Total > 100;
```

SQL-Query-QACS-D

LastName	FirstName	Phone	SaleDate	Total
Shire	Robert	206-524-2433	12/14/2014	$3,790.50
Goodyear	Katherine	206-524-3544	12/15/2014	$1,083.00
Shire	Robert	206-524-2433	1/5/2015	$270.75
Tierney	Doris	425-635-8677	1/10/2015	$812.25
Anderson	Donna	360-538-7566	1/12/2015	$270.75
Goodyear	Katherine	206-524-3544	1/15/2015	$3,249.00
Tierney	Doris	425-635-8677	1/25/2015	$379.05
Svane	Jack	206-524-5766	2/4/2015	$15,432.75
Walsh	Denesha	425-635-7566	2/4/2015	$270.75
Enquist	Craig	360-538-6455	2/7/2015	$4,873.50
Anderson	Rose	206-524-6877	2/11/2015	$3,980.03
Goodyear	Katherine	206-524-3544	2/11/2015	$866.40

Record: 1 of 12 No Filter Search

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

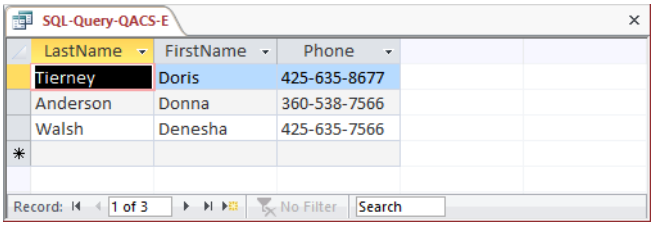
Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

For SQL Server, Oracle Database, and MySQL:

```
/* *** SQL-Query-QACS-E *** */

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

For Microsoft Access:

```
/* *** SQL-Query-QACS-E *** */

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



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

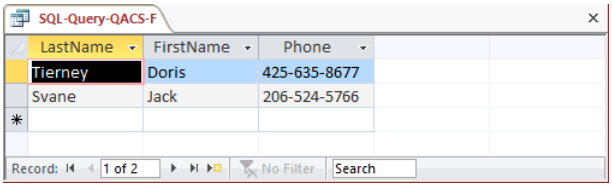
Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

For SQL Server, Oracle Database, and MySQL:

```
/* *** SQL-Query-QACS-F *** */

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

For Microsoft Access:

```
/* *** SQL-Query-QACS-F *** */

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



G. *List the LastName, FirstName, and Phone for all customers whose eighth and ninth digits (starting from the left) of their phone number are 56. For example, a phone number ending in "567" would meet the criteria.*

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle

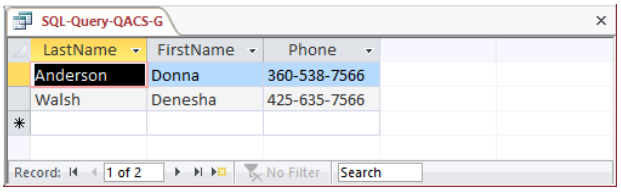
Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

For SQL Server, Oracle Database, and MySQL:

```
/* *** SQL-Query-QACS-G *** */

SELECT        LastName, FirstName, Phone
FROM          CUSTOMER
WHERE         Phone LIKE '%56_';
```

For Microsoft Access:

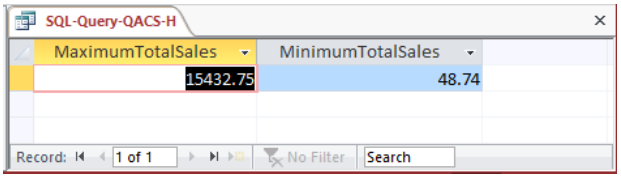
```
/* *** SQL-Query-QACS-G *** */

SELECT        LastName, FirstName, Phone
FROM          CUSTOMER
WHERE         Phone LIKE '*56?';
```



H. Determine the maximum and minimum sales Total.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-QACS-H *** */

SELECT        MAX (Total) as MaximumTotalSales,
              MIN (Total) as MinimumTotalSales
FROM          SALE;
```

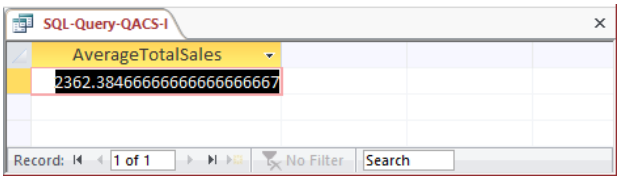


I. Determine the average sales Total.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle

Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

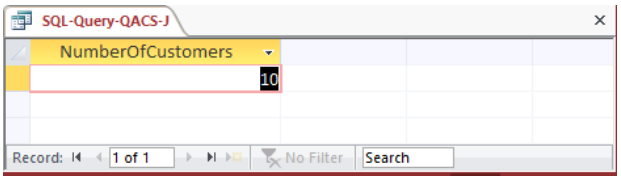
```
/* *** SQL-Query-QACS-I *** */

SELECT        AVG (Total) as AverageTotalSales
FROM          SALE;
```



J. Count the number of customers.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

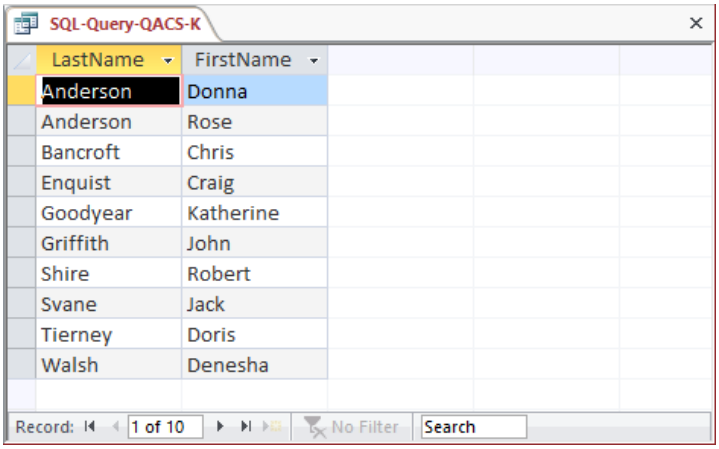
```
/* *** SQL-Query-QACS-J *** */

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



K. Group customers by LastName and then by FirstName.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

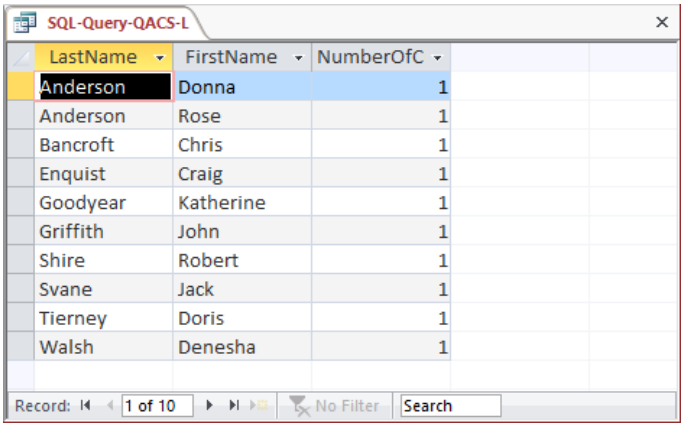
```
/* *** SQL-Query-QACS-K *** */

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



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

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

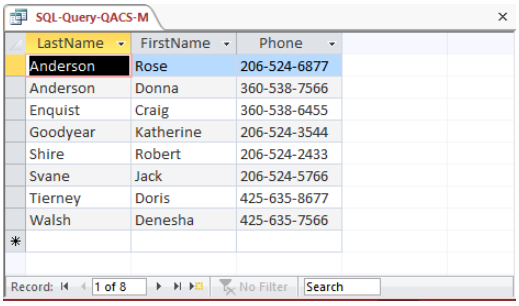
```
/* *** SQL-Query-QACS-L *** */

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



M. *Show the LastName, FirstName, and Phone of all customers who have had an order with Total greater than $100.00. Use a subquery. Present the results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-QACS-M *** */

SELECT        LastName, FirstName, Phone
FROM          CUSTOMER
WHERE         CustomerID IN
                    (SELECT CustomerID
                     FROM SALE
                     WHERE Total > 100)
ORDER BY      LastName, FirstName DESC;
```



N. *Show the LastName, FirstName, and Phone of all customers who have had an order with Total greater than $100.00. Use a join, but do not use JOIN ON syntax. Present results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-QACS-N *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER, SALE
WHERE     CUSTOMER.CustomerID = SALE.CustomerID
   AND    Total > 100;

/*       For each CUSTOMER only once:                                    */

SELECT    DISTINCT LastName, FirstName, Phone
FROM      CUSTOMER, SALE
WHERE     CUSTOMER.CustomerID = SALE.CustomerID
   AND    Total > 100;
```

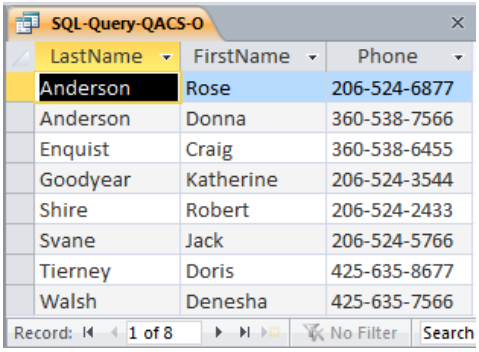
O. Show the LastName, FirstName, and Phone of all customers who have had an order with Total greater than $100.00. Use a join using JOIN ON syntax. Present results sorted by LastName in ascending order and then FirstName in descending order.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-QACS-O *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER JOIN SALE
   ON     CUSTOMER.CustomerID = SALE.CustomerID
WHERE     Total  > 100
ORDER BY  LastName, FirstName DESC;

/*       For each CUSTOMER only once:                                    */

SELECT    DISTINCT LastName, FirstName, Phone
FROM      CUSTOMER JOIN SALE
   ON     CUSTOMER.CustomerID = SALE.CustomerID
WHERE     Total  > 100
ORDER BY  LastName, FirstName DESC;
```

Note that for Microsoft Access, we must use the INNER JOIN syntax:

```
SELECT    DISTINCT LastName, FirstName, Phone
FROM      CUSTOMER INNER JOIN SALE
   ON     CUSTOMER.CustomerID = SALE.CustomerID
WHERE     Total  > 100
ORDER BY  LastName, FirstName DESC;
```



P. *Show the LastName, FirstName, and Phone of all customers who who have bought an Item named 'Desk Lamp'. Use a subquery. Present results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-QACS-P *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER
WHERE     CustomerID IN
          (SELECT   CustomerID
           FROM     SALE
           WHERE    SaleID IN
                    (SELECT   SaleID
                     FROM     SALE_ITEM
                     WHERE    ItemID IN
                              (SELECT   ItemID
                               FROM     ITEM
                               WHERE    ItemDescription = 'Desk Lamp')))
ORDER BY  LastName, FirstName DESC;
```

SQL-Query-QACS-P

LastName	FirstName	Phone
Shire	Robert	206-524-2433
Walsh	Denesha	425-635-7566
*		

Record: 1 of 2 No Filter Search

Q. *Show the LastName, FirstName, and Phone of all customers who have bought an Item named 'Desk Lamp'. Use a join, but do not use JOIN ON syntax. Present results sorted by LastName in ascending order and then FirstName in descending order.*

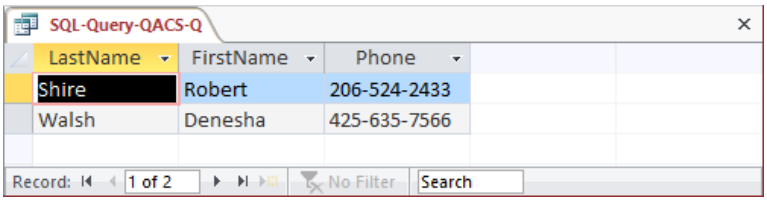
Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

For SQL Server, MySQL, and Microsoft Access:

```
/* *** SQL-Query-QACS-Q *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER AS C,
          SALE AS S,
          SALE_ITEM AS SI,
          ITEM AS I
WHERE     C.CustomerID = S.CustomerID
   AND    S.SaleID = SI.SaleID
   AND    SI.ItemID = I.ItemID
   AND    ItemDescription = 'Desk Lamp'
ORDER BY  LastName, FirstName DESC;
```

For Oracle Database, which doesn't allow "AS" in alias (range variable) declarations:

```
/* *** SQL-Query-QACS-Q-Oracle *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER C,
          SALE S,
          SALE_ITEM SI,
          ITEM I
WHERE     C.CustomerID = S.CustomerID
   AND    S.SaleID = SI.SaleID
   AND    SI.ItemID = I.ItemID
   AND    ItemDescription = 'Desk Lamp'
ORDER BY  LastName, FirstName DESC;
```



R. Show the LastName, FirstName, and Phone of all customers who have bought an Item named 'Desk Lamp'. Use a join using JOIN ON syntax. Present results sorted by LastName in ascending order and then FirstName in descending order.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

For MySQL and SQL Server:

```
/* *** SQL-Query-QACS-R *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER AS C JOIN SALE AS S
```

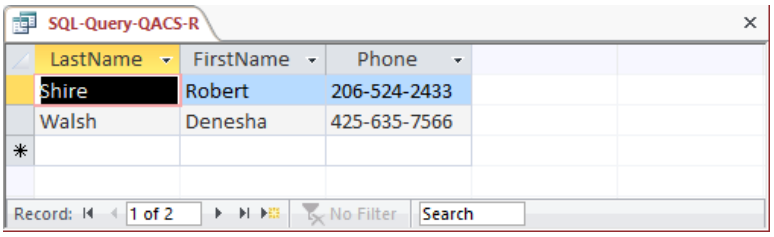
```
    ON      C.CustomerID = S.CustomerID
            JOIN    SALE_ITEM AS SI
               ON  S.SaleID = SI.SaleID
                   JOIN    ITEM AS I
                      ON  SI.ItemID = I.ItemID
WHERE       ItemDescription = 'Desk Lamp'
ORDER BY    LastName, FirstName DESC;
```

For Oracle, which does not allow "AS" in alias declarations:

```
/* *** SQL-Query-QACS-R *** */

SELECT      LastName, FirstName, Phone
FROM        CUSTOMER C JOIN SALE S
    ON      C.CustomerID = S.CustomerID
            JOIN    SALE_ITEM SI
               ON  S.SaleID = SI.SaleID
                   JOIN    ITEM I
                      ON  SI.ItemID = I.ItemID
WHERE       ItemDescription = 'Desk Lamp'
ORDER BY    LastName, FirstName DESC;
```

Note that for Microsoft Access, we must use the INNER JOIN syntax with grouping of the INNER JOINS:

```
SELECT      LastName, FirstName, Phone
FROM        ((CUSTOMER AS C INNER JOIN SALE AS S
    ON      C.CustomerID = S.CustomerID)
            INNER JOIN    SALE_ITEM AS SI
               ON         S.SaleID = SI.SaleID)
                   INNER JOIN    ITEM AS I
                      ON         SI.ItemID = I.ItemID
WHERE       ItemDescription = 'Desk Lamp'
ORDER BY    LastName, FirstName DESC;
```



S. *Show the LastName, FirstName, and Phone of all customers who have bought an Item named 'Desk Lamp'. Use a combination of a join in JOIN ON syntax and a subquery. Present results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle

Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

For SQL Server and MySQL:

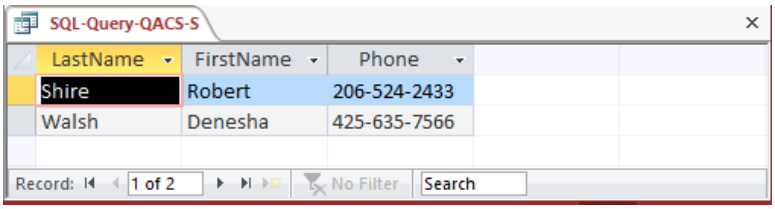
```
/* *** SQL-Query-QACS-S *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER AS C JOIN SALE AS S
ON        C.CustomerID = S.CustomerID
WHERE     SaleID IN
                    (SELECT   SaleID
                     FROM     SALE_ITEM
                     WHERE    ItemID IN
                              (SELECT   ItemID
                               FROM     ITEM
                               WHERE    ItemDescription = 'Desk Lamp'))
ORDER BY  LastName, FirstName DESC;
```

For Oracle Database, which disallows "AS" in alias declarations:

```
/* *** SQL-Query-QACS-S *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER C JOIN SALE S
ON        C.CustomerID = S.CustomerID
WHERE     SaleID IN
                    (SELECT   SaleID
                     FROM     SALE_ITEM
                     WHERE    ItemID IN
                              (SELECT   ItemID
                               FROM     ITEM
                               WHERE    ItemDescription = 'Desk Lamp'))
ORDER BY  LastName, FirstName DESC;
```

For Microsoft Access, which requires "INNER" in the join syntax:

```
/* *** SQL-Query-QACS-S *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER AS C INNER JOIN SALE AS S
ON        C.CustomerID = S.CustomerID
WHERE     SaleID IN
                    (SELECT   SaleID
                     FROM     SALE_ITEM
                     WHERE    ItemID IN
                              (SELECT   ItemID
                               FROM     ITEM
                               WHERE    ItemDescription = 'Desk Lamp'))
ORDER BY  LastName, FirstName DESC;
```



T. *Show the LastName, FirstName, and Phone of all customers who have bought an Item named 'Desk Lamp'. Use a combination of a join in JOIN ON syntax and a subquery that is different from the combination used for question S. Present results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

For MySQL and SQL Server:

```
/* *** SQL-Query-QACS-T *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER AS C JOIN SALE AS S ON C.CustomerID = S.CustomerID
          JOIN SALE_ITEM AS SI ON S.SaleID = SI.SaleID
WHERE     ItemID IN
             (SELECT       ItemID
              FROM      ITEM AS I
              WHERE     ItemDescription = 'Desk Lamp')
ORDER BY  LastName, FirstName DESC;
```

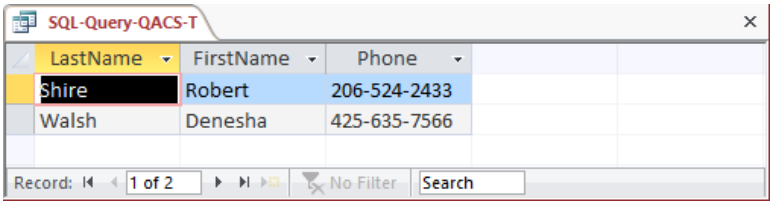
For Oracle Database, which does not allow "AS" in alias declarations:

```
/* *** SQL-Query-QACS-T *** */

SELECT    LastName, FirstName, Phone
FROM      CUSTOMER C JOIN SALE S ON C.CustomerID = S.CustomerID
          JOIN SALE_ITEM SI ON S.SaleID = SI.SaleID
WHERE     ItemID IN
             (SELECT       ItemID
              FROM      ITEM I
              WHERE     ItemDescription = 'Desk Lamp')
ORDER BY  LastName, FirstName DESC;
```

For Microsoft Access, which requires "INNER" in join syntax and parenthesization of multiple joins performed using JOIN syntax:

```
/* *** SQL-Query-QACS-T *** */

SELECT    LastName, FirstName, Phone
FROM      (CUSTOMER AS C INNER JOIN SALE AS S ON C.CustomerID = S.CustomerID)
          INNER JOIN SALE_ITEM AS SI ON  S.SaleID = SI.SaleID
```

```
WHERE     ItemID IN
              (SELECT       ItemID
               FROM      ITEM AS I
               WHERE     ItemDescription = 'Desk Lamp')
ORDER BY  LastName, FirstName DESC;
```



U. *Show the LastName, FirstName, Phone, Item for customers who have bought an Item named 'Desk Lamp'. Also show the LastName, FirstName, and Phone of all the other customers. Present results sorted by Item in ascending order, then LastName in ascending order, and then FirstName in descending order.*

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-QACS.accdb* and in corresponding files for SQL Server, Oracle Database, and MySQL, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that this is a very challenging question! The best solution involves adding the 'Desk Lamp' restriction to the inner JOINs before performing the LEFT JOIN, otherwise (if we put the 'Desk Lamp' restriction in the WHERE clause) every customer will have a sale so the LEFT JOIN will not produce any NULLs, and we will get an incorrect result from the query. Examples of this are not covered in the text, but at the same time, the text does not say you can't do it either.

The LEFT JOIN solution for Oracle Database, MySQL, and SQL Server:

```
SELECT    LastName, FirstName, Phone, ItemDescription
FROM      CUSTOMER LEFT JOIN (SALE
          JOIN   SALE_ITEM
              ON  SALE.SaleID = SALE_ITEM.SaleID
                  JOIN   ITEM
                      ON  SALE_ITEM.ItemID = ITEM.ItemID
                          AND    ITEM.ItemDescription = 'Desk Lamp')
ON     CUSTOMER.CustomerID = SALE.CustomerID
ORDER BY  ItemDescription, LastName, FirstName DESC;
```

Note that Microsoft Access does not allow nesting an INNER JOIN inside a LEFT or RIGHT JOIN. It also disallows adding the non-join condition to the "ON" clause. So in order to create a solution in Access, we must either (1) use a more complicated version of the query with a UNION but without an OUTER JOIN or (2) create and save an intermediate query (view) to be used in the final query. Note that these two approaches will also work with Oracle, SQL Server, or MySQL.

```
/* *** SQL-Query-QACS-U-UNION *** */

SELECT LastName, FirstName, Phone, ItemDescription
FROM CUSTOMER C, SALE S, SALE_ITEM SI, ITEM I
WHERE C.CustomerID = S.CustomerID
   AND         S.SaleID = SI.SaleID
   AND         SI.ItemID = I.ItemID
   AND         ItemDescription = 'Desk Lamp'
UNION
SELECT LastName, FirstName, Phone, NULL
FROM CUSTOMER
WHERE CustomerID NOT IN
  (SELECT CustomerID FROM SALE
   WHERE SaleID IN
    (SELECT SaleID FROM SALE_ITEM
     WHERE ItemID IN
        (SELECT ItemID FROM ITEM
         WHERE ItemDescription = 'Desk Lamp')))
ORDER BY ItemDescription, LastName, FirstName DESC;
```

The other approach using Access involves writing and saving an intermediate query (also called a "view"; see Chapter 7). We first write and save a query that produces the CustomerNumber and ItemDescription for all sales involving a 'Desk Lamp':

```
/* *** SQL-Query-QACS-U-Temp *** */

SELECT CustomerID, ItemDescription
FROM SALE AS S, SALE_ITEM AS SI, ITEM AS I
WHERE S.SaleID = SI.SaleID
   AND         SI.ItemID = I.ItemID
   AND          ItemDescription = 'Desk Lamp';
```

Now we can use that temporary query as if it were just another table to produce the final result:

```
/* *** SQL-Query-QACS-U-Final *** */

SELECT LastName, FirstName, Phone, ItemDescription
FROM CUSTOMER C LEFT OUTER JOIN [SQL-Query-QACS-U-TEMP] T
  ON C.CustomerID = T.CustomerID
ORDER BY ItemDescription, LastName, FirstName DESC;
```

The results below are the same for all correct versions of this query, with the possible exception of where the NULL ItemDescriptions are presented: In Access, NULL comes before all values; in Oracle, it comes last, etc.

SQL-Query-QACS-U-UNION

LastName	FirstName	Phone	ItemDescription
Anderson	Rose	206-524-6877	
Anderson	Donna	360-538-7566	
Bancroft	Chris	425-635-9788	
Enquist	Craig	360-538-6455	
Goodyear	Katherine	206-524-3544	
Griffith	John	206-524-4655	
Svane	Jack	206-524-5766	
Tierney	Doris	425-635-8677	
Shire	Robert	206-524-2433	Desk Lamp
Walsh	Denesha	425-635-7566	Desk Lamp

Record: 1 of 10 No Filter Search

❖ ANSWERS TO MORGAN IMPORTING PROJECT QUESTIONS

James Morgan owns and operates Morgan Importing, which purchases antiques and home furnishings in Asia, ships those items to a warehouse facility in Los Angeles, and then sells these items in the United States. James tracks the Asian purchases and subsequent shipments of these items to Los Angeles by using a database to keep a list of items purchased, shipments of the purchased items, and the items in each shipment. His database includes the following tables:

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

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

SHIPMENT_ITEM (*ShipmentID*, ShipmentItemID, *ItemID*, Value)

In the database schema above, the primary keys are underlined and the foreign keys are shown in italics. The database that James has created is named MI, and the three tables in the MI database schema are shown in Figure 2-62.

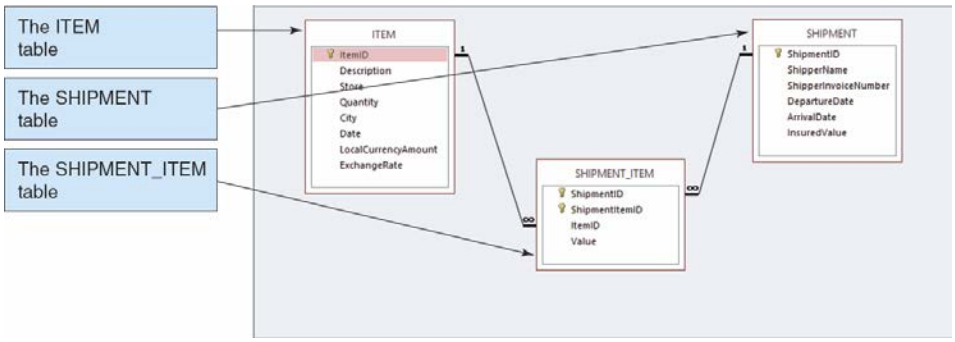


Figure 2-62 – The MI Database

The column characteristics for the tables are shown in Figures 2-63, 2-64, and 2-65. The data for the tables are shown in Figures 2-66, 2-67, and 2-68. The relationship between ITEM and SHIPMENT_ITEM should enforce referential integrity, and although it should cascade updates, it should not cascade deletions. The relationship between SHIPMENT and SHIPMENT_ITEM should enforce referential integrity and cascade both updates and deletions.

You will need to create and setup a database named MI_CH02 for use with the Morgan Importing case questions. A Microsoft Access 2013 database named MI_CH02.accdb, and SQL scripts for creating the MI_CH02 database in Microsoft SQL Server, Oracle Database, and MySQL are available on our Web site at www.pearsonhighered.com/kroenke.
If you are using the Microsoft Access 2013 MDC_CH02.accdb database, simply copy it to an appropriate location in your Documents folder. Otherwise, you will need to use the discussion and instructions necessary for setting up the MI_CH02 database in the DBMS

product you are using:

- *For Microsoft SQL Server 2014, see online Chapter 10A.*
- *For Oracle Database 12c or Oracle Express Edition 11g Release 2, see online Chapter 10B.*
- *For MySQL 5.6 Community Server, see online Chapter 10C.*

Once you have setup your MI_CH02 database, create an SQL script named MICH02-CQ.sql, and use it to record and store SQL statements that answer each of the following questions (if the question requires a written answer, use an SQL comment to record your answer):

ITEM

Column Name	Type	Key	Required	Remarks
ItemID	Integer	Primary Key	Yes	Surrogate Key
Description	Character (255)	No	Yes	Use Varchar
PurchaseDate	Date	No	Yes	
Store	Character (50)	No	Yes	
City	Character (35)	No	Yes	
Quantity	Integer	No	Yes	
LocalCurrencyAmount	Number (18,2)	No	Yes	
ExchangeRate	Number (12,6)	No	Yes	

Figure 2-63 - Column Characteristics for the MI Database ITEM Table

SHIPMENT

Column Name	Type	Key	Required	Remarks
ShipmentID	Integer	Primary Key	Yes	Surrogate Key
ShipperName	Character (35)	No	Yes	
ShipperInvoiceNumber	Integer	No	Yes	
DepartureDate	Date	No	No	
ArrivalDate	Date	No	No	
InsuredValue	Number (12,2)	No	No	

Figure 2-64 - Column Characteristics for the MI Database SHIPMENT Table

SHIPMENT_ITEM

Column Name	Type	Key	Required	Remarks
ShipmentID	Integer	Primary Key, Foreign Key	Yes	REF: SHIPMENT
ShipmentItemID	Integer	Primary Key	Yes	Sequential number, but *not* a surrogate key
ItemID	Integer	Foreign Key	Yes	REF: ITEM
Value	Number (12,2)	No	Yes	

Figure 2-65 - Column Characteristics for the MI Database SHIPMENT_ITEM Table

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

Figure 2-66 - Sample Data for the MI Database ITEM Table

ShipmentID	ShipperName	ShipperInvoiceNumber	DepartureDate	ArrivalDate	InsuredValue
1	ABC Trans-Oceanic	2008651	10-Dec-14	15-Mar-15	$15,000.00
2	ABC Trans-Oceanic	2009012	10-Jan-15	20-Mar-15	$12,000.00
3	Worldwide	49100300	05-May-15	17-Jun-15	$20,000.00
4	International	399400	02-Jun-15	17-Jul-15	$17,500.00
5	Worldwide	84899440	10-Jul-15	28-Jul-15	$25,000.00
6	International	488955	05-Aug-15	11-Sep-15	$18,000.00

Figure 2-67 - Sample Data for the MI Database SHIPMENT Table

ShipmentID	ShipmentItemID	ItemID	Value
3	1	1	$15,000.00
4	1	4	$1,200.00
4	2	3	$9,500.00
4	3	2	$4,500.00

Figure 2-68 - Sample Data for the MI Database SHIPMENT_ITEM Table

A. Show all data in each of the tables.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MI-A-ITEM *** */

SELECT    *
FROM      ITEM;
```

SQL-Query-MI-A-ITEM

ItemID	Description	PurchaseDate	Store	City	Quantity	LocalCurrencyAmount	ExchangeRate
1	QE Dining Set	4/7/2015	Eastern Treasures	Manila	2	403405	0.01774
2	Willow Serving Dishes	7/15/2015	Jade Antiques	Singapore	75	102	0.5903
3	Large Bureau	7/17/2015	Eastern Sales	Singapore	8	2000	0.5903
4	Brass Lamps	7/20/2015	Jade Antiques	Singapore	40	50	0.5903
*						0	0

Record: 1 of 4 | No Filter | Search

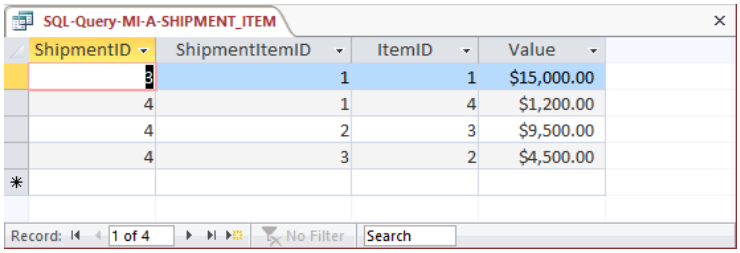
```
/* *** SQL-Query-MI-A-SHIPMENT *** */

SELECT    *
FROM      SHIPMENT;
```

SQL-Query-MI-A-SHIPMENT

ShipmentID	ShipperName	ShipperInvoiceNumber	DepartureDate	ArrivalDate	InsuredValue
1	ABC Trans-Oceanic	2008651	12/10/2014	3/15/2015	$15,000.00
2	ABC Trans-Oceanic	2009012	1/10/2015	3/20/2015	$12,000.00
3	Worldwide	49100300	5/5/2015	6/17/2015	$20,000.00
4	International	399400	6/2/2015	7/17/2015	$17,500.00
5	Worldwide	84899440	7/10/2015	7/28/2015	$25,000.00
6	International	488955	8/5/2015	9/11/2015	$18,000.00
*					

Record: 1 of 6 | No Filter | Search

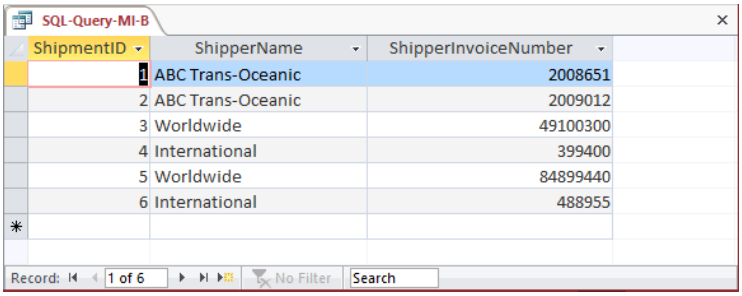
```
/* *** SQL-Query-MI-A-SHIPMENT-ITEM *** */

SELECT    *
FROM      SHIPMENT_ITEM;
```



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

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

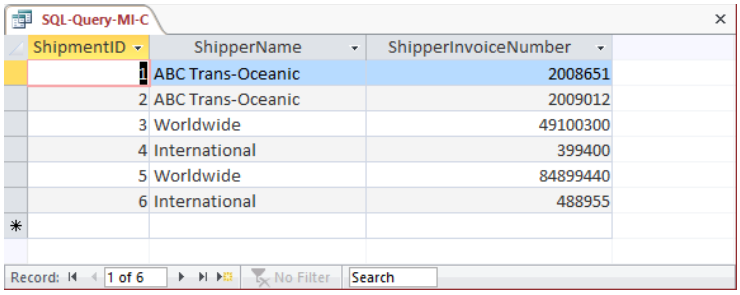
```
/* *** SQL-Query-MI-B *** */

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



C. List the ShipmentID, ShipperName, and ShipperInvoiceNumber for all shipments that have an insured value greater than $10,000.00.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MI-C *** */

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


SQL-Query-MI-C

ShipmentID	ShipperName	ShipperInvoiceNumber
1	ABC Trans-Oceanic	2008651
2	ABC Trans-Oceanic	2009012
3	Worldwide	49100300
4	International	399400
5	Worldwide	84899440
6	International	488955

Record: 1 of 6 No Filter Search

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

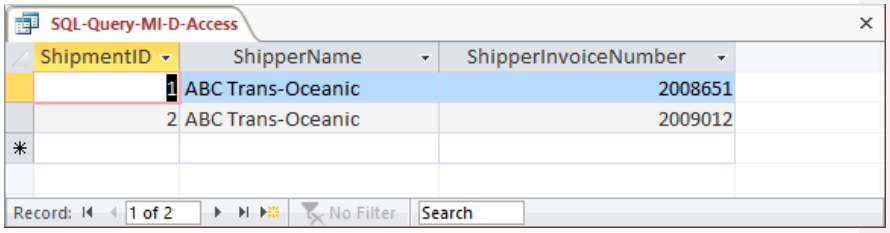
Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

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

```
/* *** SQL-Query-MI-D *** */

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

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

```
/* *** SQL-Query-MI-D-Access *** */

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


SQL-Query-MI-D-Access

ShipmentID	ShipperName	ShipperInvoiceNumber
1	ABC Trans-Oceanic	2008651
2	ABC Trans-Oceanic	2009012

Record: 1 of 2 No Filter Search

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

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

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

```
/* *** SQL-Query-MI-E *** */

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

Formatted: Indent: Left: 0"

Formatted: IM-Answer

Microsoft Access stores dates as strings so we can use the wildcard *, which gives the following SQL statement:

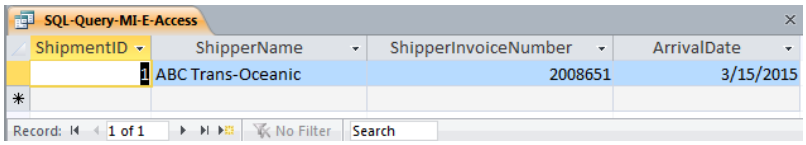
```
/* *** SQL-Query-MI-E-Access *** */

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

Oracle does not store date data type values as strings, so the following Oracle-specific form of the query must be used to extract the month:

```
/* *** SQL-Query-MI-E-Oracle *** */

SELECT    ShipmentID, ShipperName, ShipperInvoiceNumber, ArrivalDate
FROM      SHIPMENT
WHERE     EXTRACT (MONTH FROM DepartureDate) = 12;
```

MySQL- and SQL Server also do~~es~~ not store date data type values as strings, so the following ~~MySQL specific~~ form of the query must be used to extract the month. This version of the query also works with Access:

```
/* *** SQL-Query-MI-E- MySQL *** */

SELECT    ShipmentID, ShipperName, ShipperInvoiceNumber, ArrivalDate
FROM      SHIPMENT
WHERE     MONTH (DepartureDate) = 12;
```



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

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

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

```
/* *** SQL-Query-MI-F *** */

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

Formatted: Tab stops: 2.01", Left + Not at 1" + 1.25" + 1.5"

Microsoft Access stores dates as strings so we can use the wildcards * and ?, which give the following SQL statement:

```
/* *** SQL-Query-MI-F-Access-A *** */

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

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

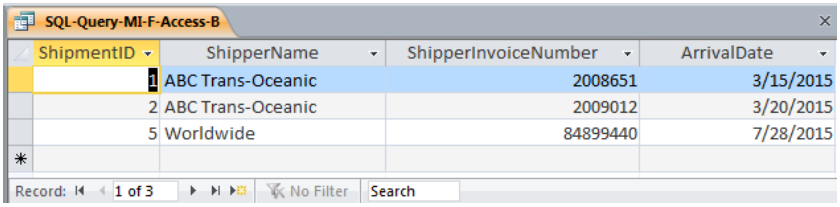
```
/* *** SQL-Query-MI-F-Access-B *** */

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

Oracle does not store date data type values as strings, so the following Oracle-specific form of the query must be used to extract the day of the month:

```
/* *** SQL-Query-MI-F-Oracle *** */

SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber, ArrivalDate
FROM     SHIPMENT
WHERE    EXTRACT (DAY FROM DepartureDate) = 10;
```

MySQL and SQL Server also do~~es~~ not store date data type values as strings, so the following ~~MySQL specific~~ form of the query must be used to extract the day of the month. This query also works in Access:

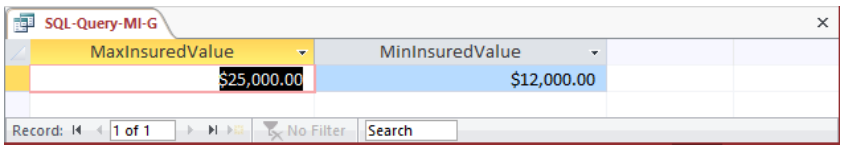
```
/* *** SQL-Query-MI-FF-MySQL *** */

SELECT   ShipmentID, ShipperName, ShipperInvoiceNumber, ArrivalDate
FROM     SHIPMENT
WHERE    DAY (DepartureDate) = 10;
```



G. Determine the maximum and minimum InsuredValue.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

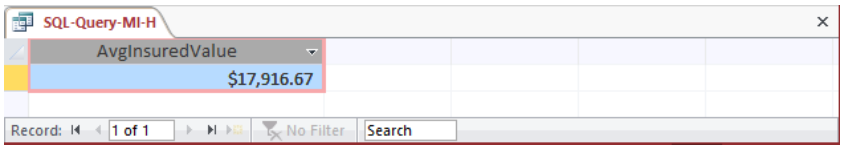
```
/* *** SQL-Query-MI-G *** */

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



H. Determine the average InsuredValue.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

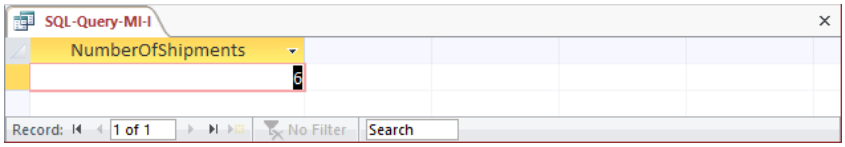
```
/* *** SQL-Query-MI-H *** */

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



I. Count the number of shipments.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MI-I *** */

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



J. Show ItemID, Description, Store, and a calculated column named USCurrencyAmount that is equal to LocalCurrencyAmount times the ExchangeRate for all rows of ITEM.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MI-J *** */

SELECT   ItemID, Description, Store,
         LocalCurrencyAmount * ExchangeRate AS USCurrencyAmount
FROM     ITEM;
```

SQL-Query-MI-J

ItemID	Description	Store	USCurrencyAmount
1	QE Dining Set	Eastern Treasures	7156.4047
2	Willow Serving Dishes	Jade Antiques	60.2106
3	Large Bureau	Eastern Sales	1180.6
4	Brass Lamps	Jade Antiques	29.515

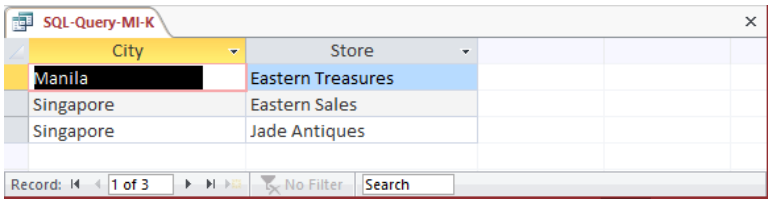
Record: 1 of 4 No Filter Search

K. Group item purchases by City and Store.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MI-K *** */

SELECT      City, Store
FROM        ITEM
```

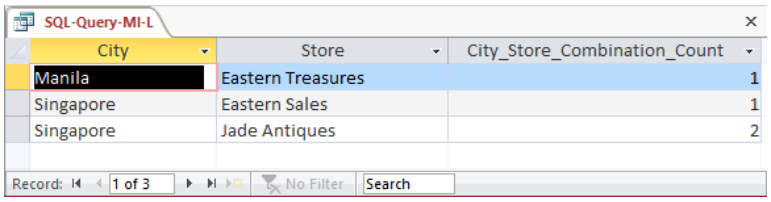
```
GROUP BY    City, Store;
```



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

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MI-L *** */

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

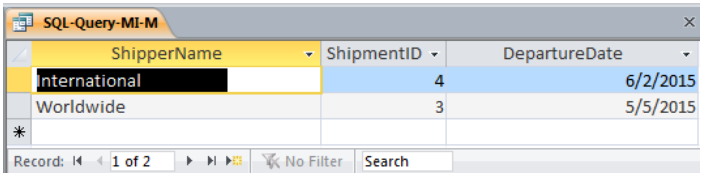


M. Show the ShipperName, ShipmentID and DepartureDate of all shipments that have an item with a value of $1,000.00 or more. Use a subquery. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MI-M *** */

SELECT   ShipperName, ShipmentID, DepartureDate
FROM     SHIPMENT
WHERE    ShipmentID IN
         (SELECT  ShipmentID
          FROM    SHIPMENT_ITEM
          WHERE   Value >= 1000)
```

```
ORDER BY ShipperName, DepartureDate DESC;
```

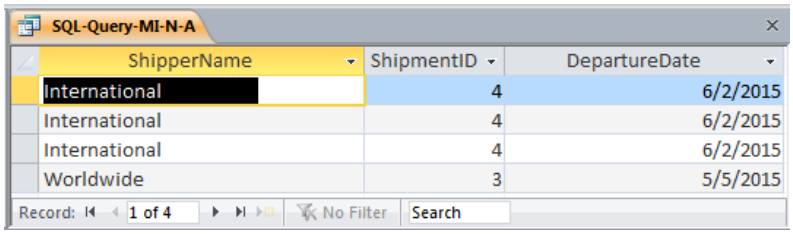


N. Show the ShipperName, ShipmentID, and DepartureDate of all shipments that have an item with a value of $1000.00 or more. Use a join. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

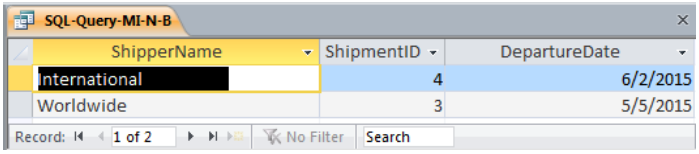
This question is a little more complicated than it appears. Note how the following queries determine that there are actually only two shipments that meet the criteria.

```
/* *** SQL-Query-MI-N-A *** */

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



Note that the three lines for International are actually only one shipment, so we can use DISTINCT to remove the duplication (shipment 4 has three items valued over $1000). Note also that we can use the *greater than or equal to* operator >= to simplify the WHERE clause. The final query is:

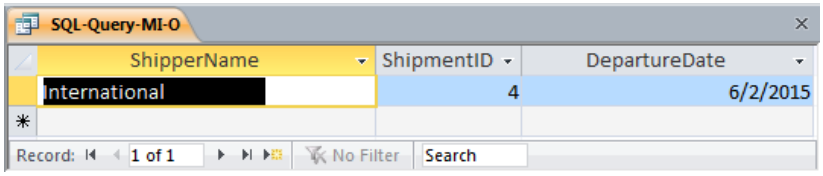
```
/* *** SQL-Query-MI-N-B *** */

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



O. *Show the ShipperName, ShipmentID, and DepartureDate of the shipments for items that were purchased in Singapore. Use a subquery. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.*

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MI-O *** */

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



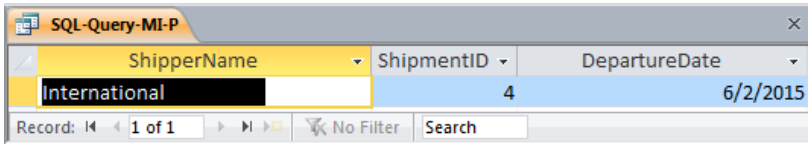
P. *Show the ShipperName, ShipmentID, and DepartureDate of all shipments that have an item that was purchased in Singapore. Use a join, but do not use JOIN ON syntax. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.*

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all 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 guarantee the appropriate answer.

```
/* *** SQL-Query-MI-P *** */

SELECT   DISTINCT ShipperName, SHIPMENT.ShipmentID, DepartureDate
FROM     SHIPMENT, SHIPMENT_ITEM, ITEM
```

```
WHERE    SHIPMENT.ShipmentID = SHIPMENT_ITEM.ShipmentID
   AND   SHIPMENT_ITEM.ItemID = ITEM.ItemID
   AND   City = 'Singapore'
ORDER BY ShipperName, DepartureDate DESC;
```



Q. Show the ShipperName, ShipmentID, and DepartureDate of all shipments that have an item that was purchased in Singapore. Use a join using JOIN ON syntax. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.

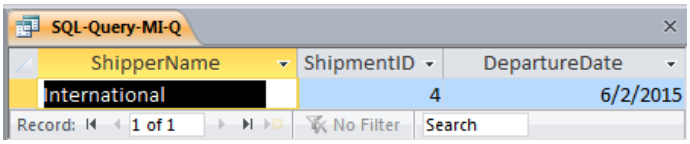
Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

For Oracle Database, MySQL, and SQL Server:

```
/* *** SQL-Query-MI-Q *** */

SELECT DISTINCT SHIPMENT.ShipperName, SHIPMENT_ITEM.ShipmentID,
   SHIPMENT.DepartureDate
FROM ITEM JOIN (SHIPMENT JOIN SHIPMENT_ITEM ON SHIPMENT.ShipmentID =
SHIPMENT_ITEM.ShipmentID) ON ITEM.ItemID = SHIPMENT_ITEM.ItemID
WHERE ITEM.City='Singapore'
ORDER BY ShipperName, DepartureDate DESC;
```

Note that for Microsoft Access, we must use the INNER JOIN syntax:

```
/* *** SQL-Query-MI-Q *** */

SELECT DISTINCT SHIPMENT.ShipperName, SHIPMENT_ITEM.ShipmentID,
   SHIPMENT.DepartureDate
FROM ITEM INNER JOIN (SHIPMENT INNER JOIN SHIPMENT_ITEM ON
SHIPMENT.ShipmentID = SHIPMENT_ITEM.ShipmentID) ON ITEM.ItemID =
SHIPMENT_ITEM.ItemID
WHERE ITEM.City='Singapore'
ORDER BY ShipperName, DepartureDate DESC;
```



R. *Show the ShipperName, ShipmentID, the DepartureDate of the 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 Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

```
/* *** SQL-Query-MI-R *** */

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

SQL-Query-MI-R

ShipperName	ShipmentID	DepartureDate	Value
International	4	6/2/2015	$1,200.00
International	4	6/2/2015	$9,500.00
International	4	6/2/2015	$4,500.00

Record: 1 of 3 No Filter Search

S. *Show the ShipperName, ShipmentID, the DepartureDate of the shipment, and Value for items that were purchased in Singapore. Also show the ShipperName, ShipmentID, and DepartureDate for all other shipments. Present results sorted by Value in ascending order, then ShipperName in ascending order, and then DepartureDate in descending order.*

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

Note that this is a very challenging question! The best solution involves adding the 'Singapore' restriction to the inner JOIN before performing the LEFT JOIN, otherwise (if we put the 'Singapore' restriction in the WHERE clause) every shipment will have an item so the LEFT JOIN will not produce any NULLs, and we will get an incorrect result from the query. Examples of this are not covered in the text, but at the same time, the text does not say you can't do it either.

The LEFT JOIN solution for Oracle Database, MySQL, and SQL Server:

```
/* *** SQL-Query-MI-S *** */

SELECT ShipperName, SHIPMENT.ShipmentID, DepartureDate, Value
```

```
FROM SHIPMENT LEFT JOIN (ITEM JOIN SHIPMENT_ITEM
   ON ITEM.ItemID = SHIPMENT_ITEM.ItemID AND
         ITEM.City = 'Singapore')
  ON SHIPMENT.ShipmentID = SHIPMENT_ITEM.ShipmentID
ORDER BY Value, ShipperName, DepartureDate DESC;
```

Note that Microsoft Access does not allow nesting an INNER JOIN inside a LEFT or RIGHT JOIN. It also disallows adding the non-join condition to the "ON" clause. So in order to create a solution in Access, we must either (1) use a more complicated version of the query with a UNION but without an OUTER JOIN or (2) create and save an intermediate query (view) to be used in the final query. Note that these two approaches will also work with Oracle, SQL Server, or MySQL.

```
/* *** SQL-Query-MI-S-UNION *** */

SELECT ShipperName, S.ShipmentID, DepartureDate, Value
FROM SHIPMENT S, ITEM I, SHIPMENT_ITEM SI
WHERE S.ShipmentID = SI.ShipmentID AND I.ItemID = SI.ItemID
    AND I.City = 'Singapore'
UNION SELECT ShipperName, ShipmentID, DepartureDate, NULL
FROM SHIPMENT
WHERE ShipmentID NOT IN
   (SELECT ShipmentID
    FROM ITEM I, SHIPMENT_ITEM SI
    WHERE I.ItemID = SI.ItemID AND I.City = 'Singapore')
ORDER BY Value, ShipperName, DepartureDate DESC;
```

The other approach using Access involves writing and saving an intermediate query (also called a "view"; see Chapter 7). We first write and save a query that produces the ShipmentID and Value for all shipments involving an item from Singapore:

```
/* *** SQL-Query-MI-S-Temp *** */

SELECT ShipmentID, Value
FROM ITEM I, SHIPMENT_ITEM SI
WHERE I.ItemID = SI.ItemID AND I.City = 'Singapore';
```

Now we can use that temporary query as if it were just another table to produce the final result:

```
/* *** SQL-Query-MI-S-Final *** */

SELECT ShipperName, S.ShipmentID, DepartureDate, Value
FROM SHIPMENT AS S LEFT OUTER JOIN [SQL-Query-MI-S-TEMP] AS T
  ON S.ShipmentID = T.ShipmentID
ORDER BY Value, ShipperName, DepartureDate DESC;
```

The results below are the same for all correct versions of this query, with the possible exception of where the NULL Values are presented: In Access, NULL comes before all values; in Oracle, it comes last, etc.