



## Summary

Wow! That was a full chapter!

Structured Query Language (SQL) was developed by IBM and has been endorsed by the ANSI SQL-92 and following standards. SQL is a data sublanguage that can be embedded into full programming languages or submitted directly to the DBMS. Knowing SQL is critical for knowledge workers, application programmers, and database administrators.

All DBMS products process SQL. Microsoft Access hides SQL, but Microsoft SQL Server, Oracle Database, and MySQL require that you use it.

We are primarily interested in five categories of SQL statements: DML, DDL, SQL/PSM statements, TCL, and DCL. DML statements include statements for querying data and for inserting, updating, and deleting data. This chapter addresses only DML query statements. Additional DML statements, DDL and SQL/PSM are discussed in Chapter 7. TCL and DCL are discussed in Chapter 9.

The examples in this chapter are based on three tables extracted from the operational database at Cape Codd Outdoor Sports. Such database extracts are common and important. Sample data for the three tables is shown in Figure 2-5.

The basic structure of an SQL query statement is SELECT/FROM/WHERE. The columns to be selected are listed after SELECT, the table(s) to process is listed after FROM, and any restrictions on data values are listed after WHERE. In a WHERE clause, character and date data values must be enclosed in single quotes. Numeric data need not be

enclosed in quotes. You can submit SQL statements directly to Microsoft Access, Microsoft SQL Server, Oracle Database, and MySQL, as described in this chapter.

This chapter explained the use of the following SQL clauses: SELECT, FROM, WHERE, ORDER BY, GROUP BY, and HAVING. This chapter explained the use of the following SQL keywords: DISTINCT, DESC, ASC, AND, OR, IN, NOT IN, BETWEEN, LIKE, % (\* for Microsoft Access), \_ (? for Microsoft Access), SUM, AVG, MIN, MAX, COUNT, and AS. You should know how to mix and match these features to obtain the results you want. By default, the WHERE clause is applied before the HAVING clause.

You can query multiple tables using subqueries and joins. Subqueries are nested queries that use the SQL keywords IN and NOT IN. An SQL SELECT expression is placed inside parentheses. Using a subquery, you can display data from the top table only. A join is created by specifying multiple table names in the FROM clause. An SQL WHERE clause is used to obtain an equijoin. In most cases, equijoins are the most sensible option. Joins can display data from multiple tables. In Chapter 8, you will learn another type of subquery that can perform work that is not possible with joins.

Some people believe the JOIN ON syntax is an easier form of join. Rows that have no match in the join condition are dropped from the join results when using a regular, or INNER, join. To keep such rows, use a LEFT OUTER or RIGHT OUTER join rather than an INNER join.



## Key Terms

/\* and \*/

ad-hoc queries

American National Standards Institute (ANSI)

AVG

business intelligence (BI) systems

correlated subquery

COUNT

CRUD

data control language (DCL)

data definition language (DDL)

data manipulation language (DML)

data mart

data sublanguage

data warehouse

data warehouse DBMS

equijoin

Extensible Markup Language (XML)

Extract, Transform, and Load (ETL) system

graphical user interface (GUI)

inner join

International Organization for Standardization (ISO)

join

join operation

joining the two tables

MAX

Microsoft Access asterisk (\*) wildcard character

Microsoft Access question mark (?)

wildcard character

MIN	SQL NOT IN operator
query by example (QBE)	SQL ON keyword
schema	SQL OR operator
SQL AND operator	SQL ORDER BY clause
SQL AS keyword	SQL outer join
SQL asterisk (*) wildcard character	SQL percent sign (%) wildcard character
SQL BETWEEN keyword	SQL query
SQL built-in functions	SQL RIGHT JOIN syntax
SQL comment	SQL right outer join
SQL DESC keyword	SQL script file
SQL DISTINCT keyword	SQL SELECT clause
SQL expression	SQL SELECT/FROM/WHERE framework
SQL FROM clause	SQL Server Compatible Syntax (ANSI 92)
SQL GROUP BY clause	SQL TOP {NumberOfRows} property
SQL HAVING clause	SQL underscore (_) wildcard character
SQL IN operator	SQL WHERE clause
SQL inner join	SQL/Persistent stored modules (SQL/PSM)
SQL INNER JOIN phrase	stock-keeping unit (SKU)
SQL JOIN keyword	Structured Query Language (SQL)
SQL JOIN ON syntax	subquery
SQL join operator	SUM
SQL LEFT JOIN syntax	TableName.ColumnName syntax
SQL left outer join	Transaction control language (TCL)
SQL LIKE keyword	



## Review Questions

- 2.1** What is a business intelligence (BI) system?
- 2.2** What is an ad-hoc query?
- 2.3** What does SQL stand for, and what is SQL?
- 2.4** What does SKU stand for? What is an SKU?
- 2.5** Summarize how data were altered and filtered in creating the Cape Codd data extraction.
- 2.6** Explain, in general terms, the relationships among the RETAIL\_ORDER, ORDER\_ITEM, and SKU\_DATA tables.
- 2.7** Summarize the background of SQL.
- 2.8** What is SQL-92? How does it relate to the SQL statements in this chapter?
- 2.9** What features have been added to SQL in versions subsequent to the SQL-92?
- 2.10** Why is SQL described as a data sublanguage?
- 2.11** What does DML stand for? What are DML statements?
- 2.12** What does DDL stand for? What are DDL statements?
- 2.13** What is the SQL SELECT/FROM/WHERE framework?
- 2.14** Explain how Microsoft Access uses SQL.
- 2.15** Explain how enterprise-class DBMS products use SQL.

The Cape Codd Outdoor Sports sale extraction database has been modified to include two additional tables, the INVENTORY table and the WAREHOUSE table. The table schemas for these tables, together with the RETAIL\_ORDER, ORDER\_ITEM, and SKU\_DATA 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)

The five tables in the revised Cape Codd database schema are shown in Figure 2-24. The column characteristics for the WAREHOUSE table are shown in Figure 2-25, and the column characteristics for the INVENTORY table are shown in Figure 2-26. The data for the WAREHOUSE table are shown in Figure 2-27, and the data for the INVENTORY table are shown in Figure 2-28.

If at all possible, you should run your SQL solutions to the following questions against an actual database. A Microsoft Access database named *Cape-Codd.accdb* is available on our Web site ([www.pearsonhighered.com/kroenke](http://www.pearsonhighered.com/kroenke)) that contains all the

Figure 2-24

The Cape Codd Database with the WAREHOUSE and INVENTORY Tables

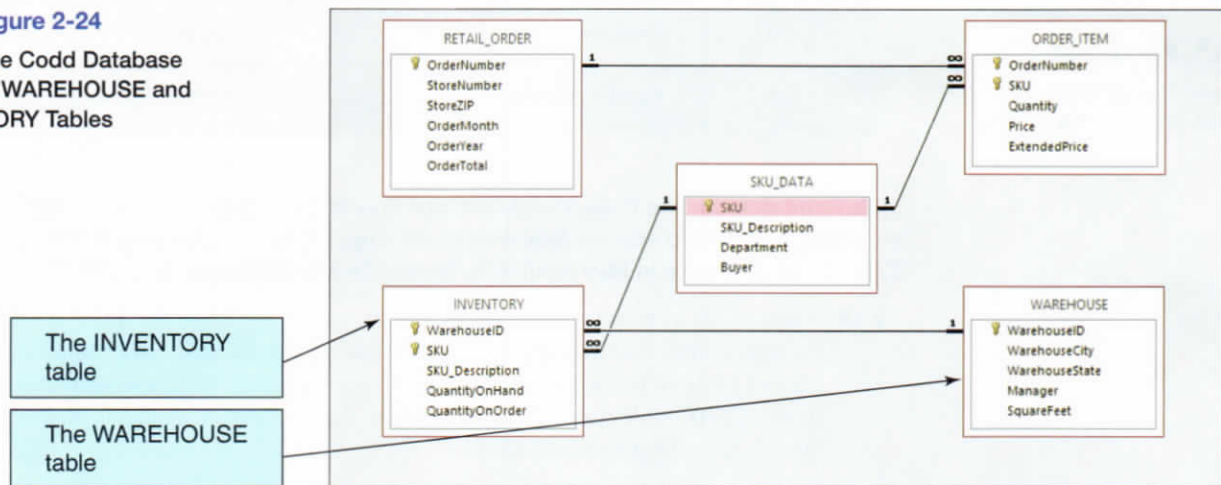


Figure 2-25

WAREHOUSE

Column Characteristics for the Cape Codd Database WAREHOUSE Table

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

**INVENTORY**

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

 **Figure 2-26**

Column Characteristics for  
the Cape Codd Database  
INVENTORY Table

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

 **Figure 2-27**

Cape Codd Database  
WAREHOUSE Table Data

**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.**

**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 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?

**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.

**2.18** Write an SQL statement to display SKU\_Description and SKU.

**2.19** Write an SQL statement to display WarehouseID.

**2.20** Write an SQL statement to display unique WarehouseIDs.

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

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

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

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

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 2-28


Cape Codd Database  
INVENTORY Table Data

- 2.25** Write an SQL statement to display the SKU, SKU\_Description, and WarehouseID for products that have a QuantityOnHand equal to 0. Sort the results in ascending order by WarehouseID.
- 2.26** Write an SQL statement to display the SKU, SKU\_Description, and WarehouseID for products that have a QuantityOnHand greater than 0. Sort the results in descending order by WarehouseID and in ascending order by SKU.
- 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.
- 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.
- 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.
- 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.
- 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'.
- 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'.
- 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.
- 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.
- 2.35** Explain the difference between the SQL built-in functions COUNT and SUM.
- 2.36** Write an SQL statement to display the WarehouseID and the sum of QuantityOnHand, grouped by WarehouseID. Name the sum TotalItemsOnHand. Display the results in descending order of TotalItemsOnHand.
- 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. Display the results in descending order of TotalItemsOnHandLT3.
- 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.
- 2.39** In your answer to Review Question 2.38, was the WHERE clause or the HAVING clause applied first? Why?

**Use both the INVENTORY and WAREHOUSE tables to answer Review Questions 2.40 through 2.55:**

- 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.
- 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.

- 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.
- 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.
- 2.44 Write an SQL statement to produce a single column called ItemLocation that combines the SKU\_Description, the phrase "is in a warehouse in", and WarehouseCity. Do not be concerned with removing leading or trailing blanks.
- 2.45 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 subquery.
- 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.
- 2.47 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 using JOIN ON syntax.
- 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.
- 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.
- 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.
- 2.51 Write an SQL statement to display the WarehouseID, the sum of QuantityOnOrder, and the sum of QuantityOnHand, grouped by WarehouseID and QuantityOnOrder. Name the sum of QuantityOnOrder as TotalItemsOnOrder and the sum of QuantityOnHand as TotalItemsOnHand.
- 2.52 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.
- 2.53 Explain why you cannot use a subquery in your answer to Review Question 2.51.
- 2.54 Explain how subqueries and joins differ.
- 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.



## Project Questions

For this set of project questions, we will extend the Microsoft Access 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:

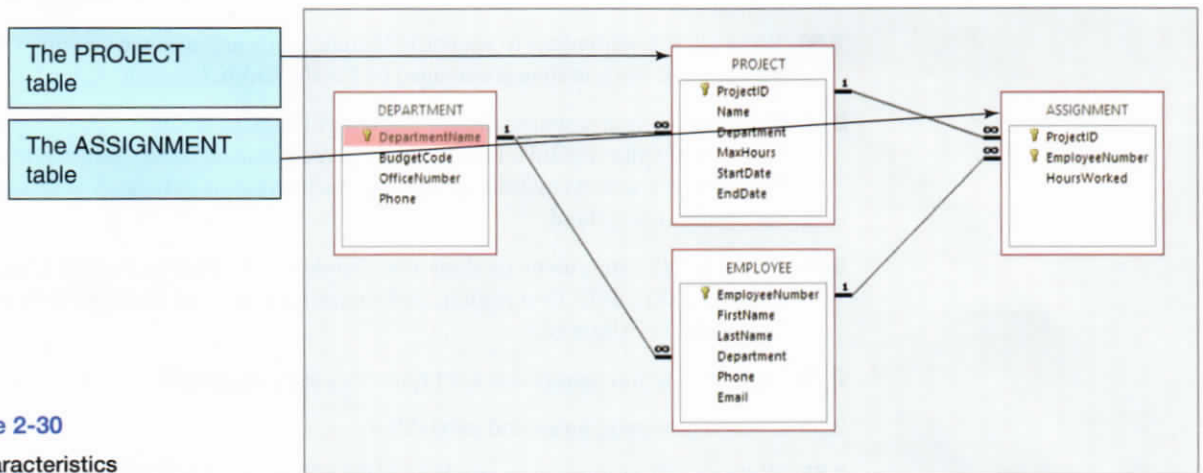
PROJECT (ProjectID, Name, Department, MaxHours, StartDate, EndDate)

ASSIGNMENT (ProjectID, EmployeeNumber, HoursWorked)

The four tables in the revised WPC database schema are shown in Figure 2-29. The column characteristics for the PROJECT table are shown in Figure 2-30, and the column characteristics for the ASSIGNMENT table are shown in Figure 2-32. Data for the PROJECT table are shown in Figure 2-31, and the data for the ASSIGNMENT table are shown in Figure 2-33.

**Figure 2-29**  
The WPC Database with the PROJECT and ASSIGNMENT Tables

**2.56** Figure 2-30 shows the column characteristics for the WPC PROJECT table. Using the column characteristics, create the PROJECT table in the WPC.accdb database.




**Figure 2-30**  
Column Characteristics for the WPC Database PROJECT Table

**PROJECT**

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





ProjectID	Name	Department	MaxHours	StartDate	EndDate
1000	2013 Q3 Product Plan	Marketing	135.00	10-MAY-13	15-JUN-13
1100	2013 Q3 Portfolio Analysis	Finance	120.00	07-JUL-13	25-JUL-13
1200	2013 Q3 Tax Preparation	Accounting	145.00	10-AUG-13	15-OCT-13
1300	2013 Q4 Product Plan	Marketing	150.00	10-AUG-13	15-SEP-13
1400	2013 Q4 Portfolio Analysis	Finance	140.00	05-OCT-13	

 **Figure 2-31**  
Sample Data  
for the WPC  
Database  
PROJECT Table

### ASSIGNMENT

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

 **Figure 2-32**  
Column  
Characteristics for  
the WPC Database  
ASSIGNMENT  
Table

 **Figure 2-33**  
Sample Data for the  
WPC Database  
ASSIGNMENT Table


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

- 2.57** Create the relationship and referential integrity constraint between PROJECT and DEPARTMENT. Enable enforcing of referential integrity and cascading of data updates, but do *not* enable cascading of data from deleted records.
- 2.58** Figure 2-31 shows the data for the WPC PROJECT table. Using the Datasheet view, enter the data shown in Figure 2-31 into your PROJECT table.
- 2.59** Figure 2-32 shows the column characteristics for the WPC ASSIGNMENT table. Using the column characteristics, create the ASSIGNMENT table in the WPC.accdb database.
- 2.60** Create the relationship and referential integrity constraint between ASSIGNMENT and EMPLOYEE. Enable enforcing of referential integrity, but do *not* enable either cascading updates or the cascading of data from deleted records.
- 2.61** Create the relationship and referential integrity constraint between ASSIGNMENT and PROJECT. Enable enforcing of referential integrity and cascading of deletes, but do *not* enable cascading updates.
- 2.62** Figure 2-33 shows the data for the WPC ASSIGNMENT table. Using the Datasheet view, enter the data shown in Figure 2-33 into your ASSIGNMENT table.
- 2.63** In Review Question 2.58, the table data was entered after referential integrity constraints were created in Review Question 2.57. In Review Question 2.62, the table data was entered after referential integrity constraints were created in Review Questions 2.59 and 2.60. Why was the data entered after the referential integrity constraints were created instead of before the constraints were created?
- 2.64** 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.
- 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, 2013? 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, EmployeeNumber, LastName, FirstName, and Phone.
  - F.** Who are the employees assigned to each project? The ProjectID, Name, and Department. Show EmployeeNumber, LastName, FirstName, and Phone.
  - G.** Who are the employees assigned to each project? Show ProjectID, Name, Department, and Department Phone. Show EmployeeNumber, LastName, FirstName, and Employee Phone. Sort by ProjectID, in ascending order.
  - 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.
  - 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.
- 2.65 Using Microsoft Access QBE, create and run new queries to answer the questions in Project Question 2.64. 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.

The following questions refer to the NDX table data as described starting on page 74. You can obtain a copy of this data in the Microsoft Access database *DBP-e13-NDX.accdb* from the text's Web site ([www.pearsonhighered.com/kroenke](http://www.pearsonhighered.com/kroenke)).

- 2.66 Write SQL queries to produce the following results:
- A. The ChangeClose on Fridays.
  - B. The minimum, maximum, and average ChangeClose on Fridays.
  - C. The average ChangeClose grouped by TYear. Show TYear.
  - D. The average ChangeClose grouped by TYear and TMonth. Show TYear and TMonth.
  - E. The average ChangeClose grouped by TYear, TQuarter, TMonth shown in descending order of the average (you will have to give a name to the average in order to sort by it). Show TYear, TQuarter, and TMonth. Note that months appear in alphabetical and not calendar order. Explain what you need to do to obtain months in calendar order.
  - F. The difference between the maximum ChangeClose and the minimum ChangeClose grouped by TYear, TQuarter, TMonth shown in descending order of the difference (you will have to give a name to the difference in order to sort by it). Show TYear, TQuarter, and TMonth.
  - G. The average ChangeClose grouped by TYear shown in descending order of the average (you will have to give a name to the average in order to sort by it). Show only groups for which the average is positive.
  - H. Display a single field with the date in the form day/month/year. Do not be concerned with trailing blanks.
- 2.67 It is possible that volume (the number of shares traded) has some correlation with the direction of the stock market. Use the SQL you have learned in this chapter to investigate this possibility. Develop at least five different SQL statements in your investigation.



## Case Questions

### 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*)

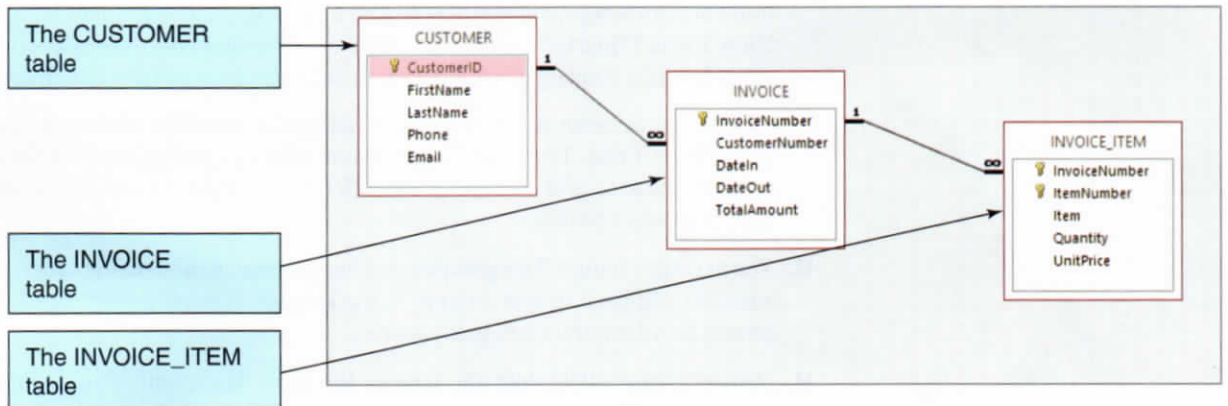
In the database schema above, the primary keys are underlined and the foreign keys are shown in italics. The database that Marcia has created is named MDC, and the three tables in the MDC database schema are shown in Figure 2-34.

The column characteristics for the tables are shown in Figures 2-35, 2-36, and 2-37. 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 2-38, 2-39, and 2-40.

We recommend that you create a Microsoft Access 2013 database named *MDC-CH02.accdB* using the database schema, column characteristics, and data shown above and then use this database to test your solutions to the questions in this section. Alternatively, 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](http://www.pearsonhighered.com/kroenke).

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

**Figure 2-34**  
The MDC Database




- A. Show all data in each of the tables.
- B. List the LastName, FirstName, and Phone of all customers.

**Figure 2-35**  
Column Characteristics  
for the MDC Database  
CUSTOMER Table

CUSTOMER				
Column Name	Type	Key	Required	Remarks
CustomerID	AutoNumber	Primary Key	Yes	Surrogate Key
FirstName	Text (25)	No	Yes	
LastName	Text (25)	No	Yes	
Phone	Text (12)	No	No	
Email	Text (100)	No	No	


**INVOICE**

Column Name	Type	Key	Required	Remarks
InvoiceNumber	Number	Primary Key	Yes	Long Integer
CustomerNumber	Number	Foreign Key	Yes	Long Integer
DateIn	Date	No	Yes	
DateOut	Date	No	No	
TotalAmount	Currency	No	No	Two Decimal Places


 **Figure 2-36**  
Column Characteristics for the MDC Database INVOICE Table

**INVOICE\_ITEM**

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

 **Figure 2-37**  
Column Characteristics for the MDC Database INVOICE\_ITEM Table

- C. List the LastName, FirstName, and Phone for all customers with a FirstName of 'Nikki'.
- D. List the LastName, FirstName, Phone, DateIn, and DateOut of all orders in excess of \$100.00.
- E. List the LastName, FirstName, and Phone of all customers whose first name starts with 'B'.
- F. List the LastName, FirstName, and Phone of all customers whose last name includes the characters 'cat'.

 **Figure 2-38**  
Sample Data for the MDC Database CUSTOMER 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


InvoiceNumber	CustomerNumber	DateIn	DateOut	TotalAmount
2013001	1	04-Oct-13	06-Oct-13	\$158.50
2013002	2	04-Oct-13	06-Oct-13	\$25.00
2013003	1	06-Oct-13	08-Oct-13	\$49.00
2013004	4	06-Oct-13	08-Oct-13	\$17.50
2013005	6	07-Oct-13	11-Oct-13	\$12.00
2013006	3	11-Oct-13	13-Oct-13	\$152.50
2013007	3	11-Oct-13	13-Oct-13	\$7.00
2013008	7	12-Oct-13	14-Oct-13	\$140.50
2013009	5	12-Oct-13	14-Oct-13	\$27.00

 **Figure 2-39**

Sample Data for the MDC Database INVOICE Table

- G.** List the LastName, FirstName, and Phone for all customers whose second and third numbers (from the right) of their phone number are 23.
- H.** Determine the maximum and minimum TotalAmount.
- I.** Determine the average TotalAmount.
- J.** Count the number of customers.
- K.** Group customers by LastName and then by FirstName.
- L.** Count the number of customers having each combination of LastName and FirstName.
- 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.
- 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 results sorted by LastName in ascending order and then FirstName in descending order.
- 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 results sorted by LastName in ascending order and then FirstName in descending order.
- 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 results sorted by LastName in ascending order and then FirstName in descending order.
- 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 results sorted by LastName in ascending order and then FirstName in descending order.
- 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 results sorted by LastName in ascending order and then FirstName in descending order.

InvoiceNumber	ItemNumber	Item	Quantity	UnitPrice
2013001	1	Blouse	2	\$3.50
2013001	2	Dress Shirt	5	\$2.50
2013001	3	Formal Gown	2	\$10.00
2013001	4	Slacks-Mens	10	\$5.00
2013001	5	Slacks-Womens	10	\$6.00
2013001	6	Suit-Mens	1	\$9.00
2013002	1	Dress Shirt	10	\$2.50
2013003	1	Slacks-Mens	5	\$5.00
2013003	2	Slacks-Womens	4	\$6.00
2013004	1	Dress Shirt	7	\$2.50
2013005	1	Blouse	2	\$3.50
2013005	2	Dress Shirt	2	\$2.50
2013006	1	Blouse	5	\$3.50
2013006	2	Dress Shirt	10	\$2.50
2013006	3	Slacks-Mens	10	\$5.00
2013006	4	Slacks-Womens	10	\$6.00
2013007	1	Blouse	2	\$3.50
2013008	1	Blouse	3	\$3.50
2013008	2	Dress Shirt	12	\$2.50
2013008	3	Slacks-Mens	8	\$5.00
2013008	4	Slacks-Womens	10	\$6.00
2013009	1	Suit-Mens	3	\$9.00

 **Figure 2-40**  
Sample Data for the  
MDC Database  
INVOICE\_ITEM Table

- S.** Show the LastName, FirstName, Phone, and TotalAmount of all customers who have had an order with an Item named 'Dress Shirt'. Use a combination of a join and a sub-query. Present results sorted by LastName in ascending order and then FirstName in descending order.
- T.** Show the LastName, FirstName, Phone, and TotalAmount of all customers who have had an order with an Item named 'Dress Shirt'. Also show the LastName, FirstName, and Phone of all other customers. Present results sorted by LastName in ascending order, and then FirstName in descending order.

## The Queen Anne Curiosity Shop



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

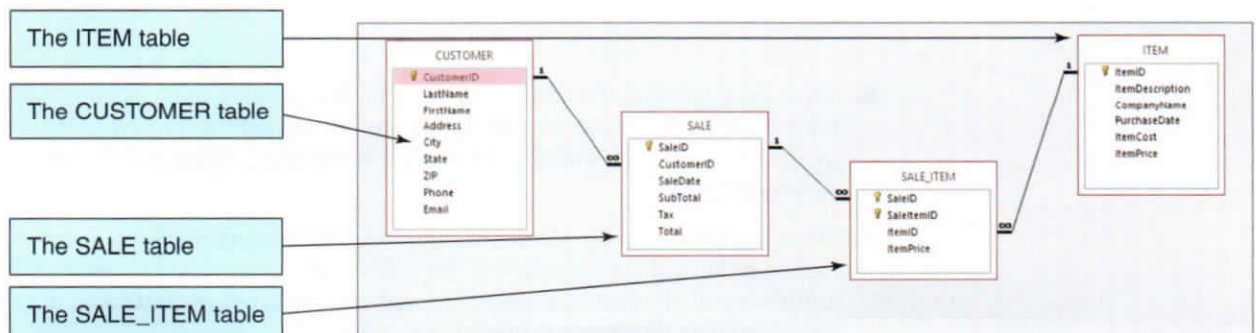
Assume that CustomerID of CUSTOMER, ItemID of ITEM, SaleID of SALE, and SaleItemID of SALE\_ITEM are all surrogate keys with values as follows:

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 2-41.

The column characteristics for the tables are shown in Figures 2-42, 2-43, 2-44, and 2-45. 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 2-46, 2-47, 2-48, and 2-49.

**Figure 2-41**  
The QACS Database





**CUSTOMER**

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

 **Figure 2-42**

Column Characteristics  
for the QACS Database  
CUSTOMER Table

 **Figure 2-43**

Column  
Characteristics for  
the QACS Database  
SALE Table

**SALE**

Column Name	Type	Key	Required	Remarks
SaleID	AutoNumber	Primary Key	Yes	Surrogate Key
CustomerID	Number	Foreign Key	Yes	Long Integer
SaleDate	Date	No	Yes	
SubTotal	Number	No	No	Currency, 2 decimal places
Tax	Number	No	No	Currency, 2 decimal places
Total	Number	No	No	Currency, 2 decimal places

 **Figure 2-44**


Column  
Characteristics  
for the QACS  
Database  
SALE\_ITEM Table

**SALE\_ITEM**

Column Name	Type	Key	Required	Remarks
SaleID	Number	Primary Key, Foreign Key	Yes	Long Integer
SaleItemID	Number	Primary Key	Yes	Long Integer
ItemID	Number	Number	Yes	Long Integer
ItemPrice	Number	No	No	Currency, 2 decimal places

## ITEM

Column Name	Type	Key	Required	Remarks
ItemID	AutoNumber	Primary Key	Yes	Surrogate Key
ItemDescription	Text (255)	No	Yes	
CompanyName	Text (100)	No	Yes	
PurchaseDate	Date	No	Yes	
ItemCost	Number	No	Yes	Currency, 2 decimal places
ItemPrice	Number	No	Yes	Currency, 2 decimal places

 **Figure 2-45**  
Column Characteristics  
for the QACS  
Database ITEM Table


We recommend that you create a Microsoft Access 2013 database named *QACS-CH02.accdb* using the database schema, column characteristics, and data shown above and then use this database to test your solutions to the questions in this section. Alternatively, 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](http://www.pearsonhighered.com/kroenke).

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

- A. Show all data in each of the tables.
- B. List the LastName, FirstName, and Phone of all customers.
- C. List the LastName, FirstName, and Phone for all customers with a FirstName of 'John'.
- D. List the LastName, FirstName, and Phone of all customers with a last name of 'Anderson'.
- E. List the LastName, FirstName, and Phone of all customers whose first name starts with 'D'.
- F. List the LastName, FirstName, and Phone of all customers whose last name includes the characters 'ne'.
- G. List the LastName, FirstName, and Phone for all customers whose second and third numbers (from the right) of their phone number are 56.
- H. Determine the maximum and minimum sales Total.
- I. Determine the average sales Total.
- J. Count the number of customers.
- K. Group customers by LastName and then by FirstName.
- L. Count the number of customers having each combination of LastName and 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.

## ITEM

Column Name	Type	Key	Required	Remarks
ItemID	AutoNumber	Primary Key	Yes	Surrogate Key
ItemDescription	Text (255)	No	Yes	
CompanyName	Text (100)	No	Yes	
PurchaseDate	Date	No	Yes	
ItemCost	Number	No	Yes	Currency, 2 decimal places
ItemPrice	Number	No	Yes	Currency, 2 decimal places

 **Figure 2-45**  
Column Characteristics  
for the QACS  
Database ITEM Table

We recommend that you create a Microsoft Access 2013 database named *QACS-CH02.accdb* using the database schema, column characteristics, and data shown above and then use this database to test your solutions to the questions in this section. Alternatively, 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](http://www.pearsonhighered.com/kroenke).

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

- A. Show all data in each of the tables.
- B. List the LastName, FirstName, and Phone of all customers.
- C. List the LastName, FirstName, and Phone for all customers with a FirstName of 'John'.
- D. List the LastName, FirstName, and Phone of all customers with a last name of 'Anderson'.
- E. List the LastName, FirstName, and Phone of all customers whose first name starts with 'D'.
- F. List the LastName, FirstName, and Phone of all customers whose last name includes the characters 'ne'.
- G. List the LastName, FirstName, and Phone for all customers whose second and third numbers (from the right) of their phone number are 56.
- H. Determine the maximum and minimum sales Total.
- I. Determine the average sales Total.
- J. Count the number of customers.
- K. Group customers by LastName and then by FirstName.
- L. Count the number of customers having each combination of LastName and 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.

CustomerID	LastName	FirstName	Address	City	State	ZIP	Phone	Email
1	Shire	Robert	6225 Evanston Ave N	Seattle	WA	98103	206-524-2433	Robert.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 2-46**

Sample Data for the QACS  
Database CUSTOMER Table

SaleID	CustomerID	SaleDate	SubTotal	Tax	Total
1	1	12/14/2012	\$3,500.00	\$290.50	\$3,790.50
2	2	12/15/2012	\$1,000.00	\$83.00	\$1,083.00
3	3	12/15/2012	\$50.00	\$4.15	\$54.15
4	4	12/23/2012	\$45.00	\$3.74	\$48.74
5	1	1/5/2013	\$250.00	\$20.75	\$270.75
6	5	1/10/2013	\$750.00	\$62.25	\$812.25
7	6	1/12/2013	\$250.00	\$20.75	\$270.75
8	2	1/15/2013	\$3,000.00	\$249.00	\$3,249.00
9	5	1/25/2013	\$350.00	\$29.05	\$379.05
10	7	2/4/2013	\$14,250.00	\$1,182.75	\$15,432.75
11	8	2/4/2013	\$250.00	\$20.75	\$270.75
12	5	2/7/2013	\$50.00	\$4.15	\$54.15
13	9	2/7/2013	\$4,500.00	\$373.50	\$4,873.50
14	10	2/11/2013	\$3,675.00	\$305.03	\$3,980.03
15	2	2/11/2013	\$800.00	\$66.40	\$866.40

 **Figure 2-47**

Sample Data for the QACS  
Database SALE Table

- 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.
- 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.
- 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.
- 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.
- 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.

 Figure 2-48

Sample Data for the QACS  
Database SALE\_ITEM 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

- S.** Show the LastName, FirstName, and Phone of all customers who have bought an Item named 'Desk Lamp'. Use a combination of a join and a subquery. Present results sorted by LastName in ascending order and then FirstName in descending order.
- T.** Show the LastName, FirstName, and Phone of all customers who have bought an Item named 'Desk Lamp'. Use a combination of a join 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.

ItemID	ItemDescription	CompanyName	PurchaseDate	ItemCost	ItemPrice
1	Antique Desk	European Specialties	11/7/2012	\$1,800.00	\$3,000.00
2	Antique Desk Chair	Andrew Lee	11/10/2012	\$300.00	\$500.00
3	Dining Table Linens	Linens and Things	11/14/2012	\$600.00	\$1,000.00
4	Candles	Linens and Things	11/14/2012	\$30.00	\$50.00
5	Candles	Linens and Things	11/14/2012	\$27.00	\$45.00
6	Desk Lamp	Lamps and Lighting	11/14/2012	\$150.00	\$250.00
7	Dining Table Linens	Linens and Things	11/14/2012	\$450.00	\$750.00
8	Book Shelf	Denise Harrion	11/21/2012	\$150.00	\$250.00
9	Antique Chair	New York Brokerage	11/21/2012	\$750.00	\$1,250.00
10	Antique Chair	New York Brokerage	11/21/2012	\$1,050.00	\$1,750.00
11	Antique Candle Holder	European Specialties	11/28/2012	\$210.00	\$350.00
12	Antique Desk	European Specialties	1/5/2013	\$1,920.00	\$3,200.00
13	Antique Desk	European Specialties	1/5/2013	\$2,100.00	\$3,500.00
14	Antique Desk Chair	Specialty Antiques	1/6/2013	\$285.00	\$475.00
15	Antique Desk Chair	Specialty Antiques	1/6/2013	\$339.00	\$565.00
16	Desk Lamp	General Antiques	1/6/2013	\$150.00	\$250.00
17	Desk Lamp	General Antiques	1/6/2013	\$150.00	\$250.00
18	Desk Lamp	Lamps and Lighting	1/6/2013	\$144.00	\$240.00
19	Antique Dining Table	Denesha Walsh	1/10/2013	\$3,000.00	\$5,000.00
20	Antique Sideboard	Chris Bancroft	1/11/2013	\$2,700.00	\$4,500.00
21	Dining Table Chairs	Specialty Antiques	1/11/2013	\$5,100.00	\$8,500.00
22	Dining Table Linens	Linens and Things	1/12/2013	\$450.00	\$750.00
23	Dining Table Linens	Linens and Things	1/12/2013	\$480.00	\$800.00
24	Candles	Linens and Things	1/17/2013	\$30.00	\$50.00
25	Candles	Linens and Things	1/17/2013	\$36.00	\$60.00

 Figure 2-49

QACS Database ITEM Table

**Morgan  
Importing**



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)

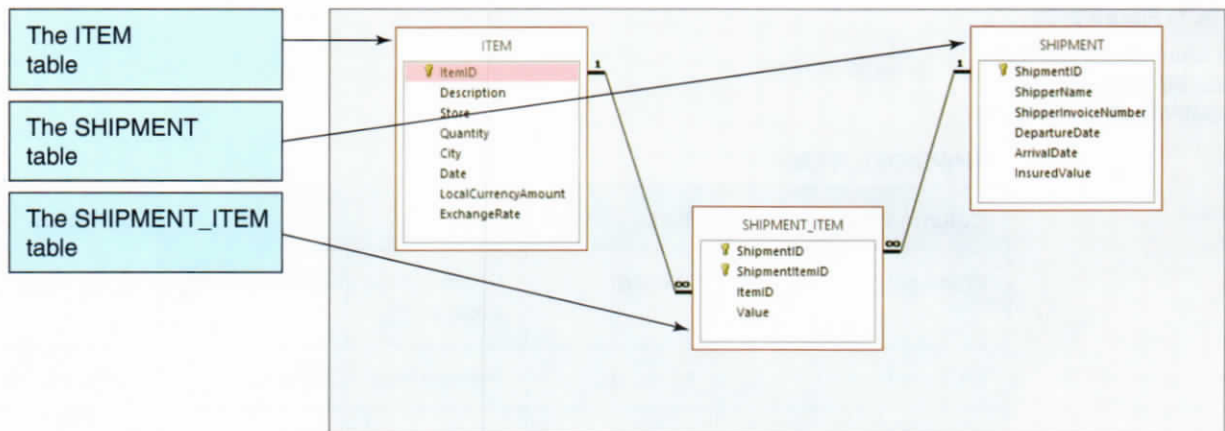
**SHIPMENT** (ShipmentID, ShipperName, ShipperInvoiceNumber, DepartureDate, ArrivalDate, InsuredValue)

**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 2-50.

The column characteristics for the tables are shown in Figures 2-51, 2-52, and 2-53. The data for the tables are shown in Figures 2-54, 2-55, and 2-56. The relationship between ITEM

**Figure 2-50**  
The MI Database



**Figure 2-51**

Column Characteristics for  
the MI Database ITEM Table

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



 **Figure 2-52** SHIPMENT

Column Characteristics for the MI Database SHIPMENT Table

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

 **Figure 2-53**

Column Characteristics for the MI Database SHIPMENT\_ITEM Table

#### SHIPMENT\_ITEM

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

 **Figure 2-54**

Sample Data for the MI Database ITEM Table

ItemID	Description	PurchaseDate	Store	City	Quantity	LocalCurrencyAmount	ExchangeRate
1	QE Dining Set	07-Apr-13	Eastern Treasures	Manila	2	403405	0.01774
2	Willow Serving Dishes	15-Jul-13	Jade Antiques	Singapore	75	102	0.5903
3	Large Bureau	17-Jul-13	Eastern Sales	Singapore	8	2000	0.5903
4	Brass Lamps	20-Jul-13	Jade Antiques	Singapore	40	50	0.5903

ShipmentID	ShipperName	ShipperInvoiceNumber	DepartureDate	ArrivalDate	InsuredValue
1	ABC Trans-Oceanic	2008651	10-Dec-12	15-Mar-13	\$15,000.00
2	ABC Trans-Oceanic	2009012	10-Jan-13	20-Mar-13	\$12,000.00
3	Worldwide	49100300	05-May-13	17-Jun-13	\$20,000.00
4	International	399400	02-Jun-13	17-Jul-13	\$17,500.00
5	Worldwide	84899440	10-Jul-13	28-Jul-13	\$25,000.00
6	International	488955	05-Aug-13	11-Sep-13	\$18,000.00

 Figure 2-55

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 2-56

Sample Data for the MI  
Database SHIPMENT\_ITEM  
Table

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.

We recommend that you create a Microsoft Access 2013 database named *MI-CH02.accdb* using the database schema, column characteristics, and data shown above and then use this database to test your solutions to the questions in this section. Alternatively, 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](http://www.pearsonhighered.com/kroenke).

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

- A. Show all data in each of the tables.
- B. List the ShipmentID, ShipperName, and ShipperInvoiceNumber of all shipments.
- C. List the ShipmentID, ShipperName, and ShipperInvoiceNumber for all shipments that have an insured value greater than \$10,000.00.
- D. List the ShipmentID, ShipperName, and ShipperInvoiceNumber of all shippers whose name starts with 'AB'.
- 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.
- 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.

- G.** Determine the maximum and minimum InsuredValue.
- H.** Determine the average InsuredValue.
- I.** Count the number of shipments.
- 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.
- K.** Group item purchases by City and Store.
- L.** Count the number of purchases having each combination of City and 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.
- N.** Show the ShipperName, ShipmentID, and DepartureDate of all shipments that have an item with a value of \$1,000.00 or more. Use a join. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.
- O.** Show the ShipperName, ShipmentID, and DepartureDate of the shipment for items that were purchased in Singapore. Use a subquery. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.
- 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.
- 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.
- 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.
- 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.