Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SKU: 9780134802749-SOLUTIONS

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

INSTRUCTOR'S MANUAL

TO ACCOMPANY

David M. Kroenke | David J. Auer | Scott L. Vandenberg | Robert C. Yoder

40th Anniversary Edition

DATABASE PROCESSING

Fundamentals, Design, and Implementation

15th Edition

Chapter 2

Introduction to Structured Query Language

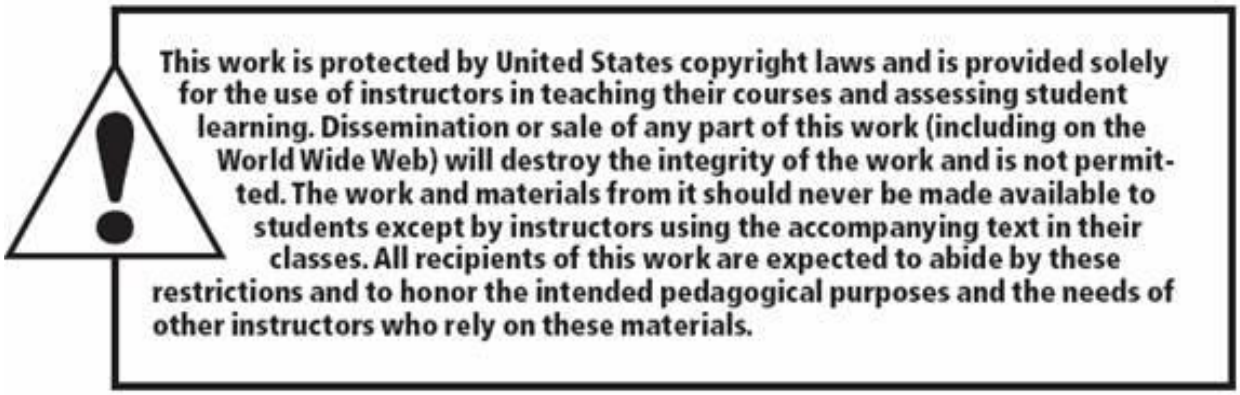


Prepared By

Scott L. Vandenberg

Siena College

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



Instructor's Manual to accompany:

Database Processing: Fundamental, Design, and Implementation (15th Edition)

David M. Kroenke | David J. Auer | Scott L. Vandenberg | Robert C. Yoder

---

Copyright © 2019 Pearson Education, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

❖ 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 on recursive relationships
- 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!

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.

1. If you are using **Microsoft SQL Server 2017** as your DBMS, you should use **Online Chapter 10A – Managing Databases with Microsoft SQL Server 2017**, and cover **pages 10A-1 through 10A-58** to help your students get set up for the SQL work in Chapter 2.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

2. If you are using **Oracle Database 12c Release 2** or **Oracle Database XE** as your DBMS, you should use **Online Chapter 10B – Managing Databases with Oracle Database**, and cover **pages 10B-1 through 10B-55** to help your students get set up for the SQL work in Chapter 2.
3. If you are using **MySQL 5.7** as your DBMS, you should use **Online Chapter 10C – Managing Databases with MySQL 5.7**, and cover **pages 10C-1 through 10C-35** to help your students get set up for the SQL work in Chapter 2.
4. 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

There are no known errors at this time. Any errors that are discovered in the future will be reported and corrected in the Online DBP e15 Errata document, which will be available at http://www.pearsonhighered.com/kroenke.

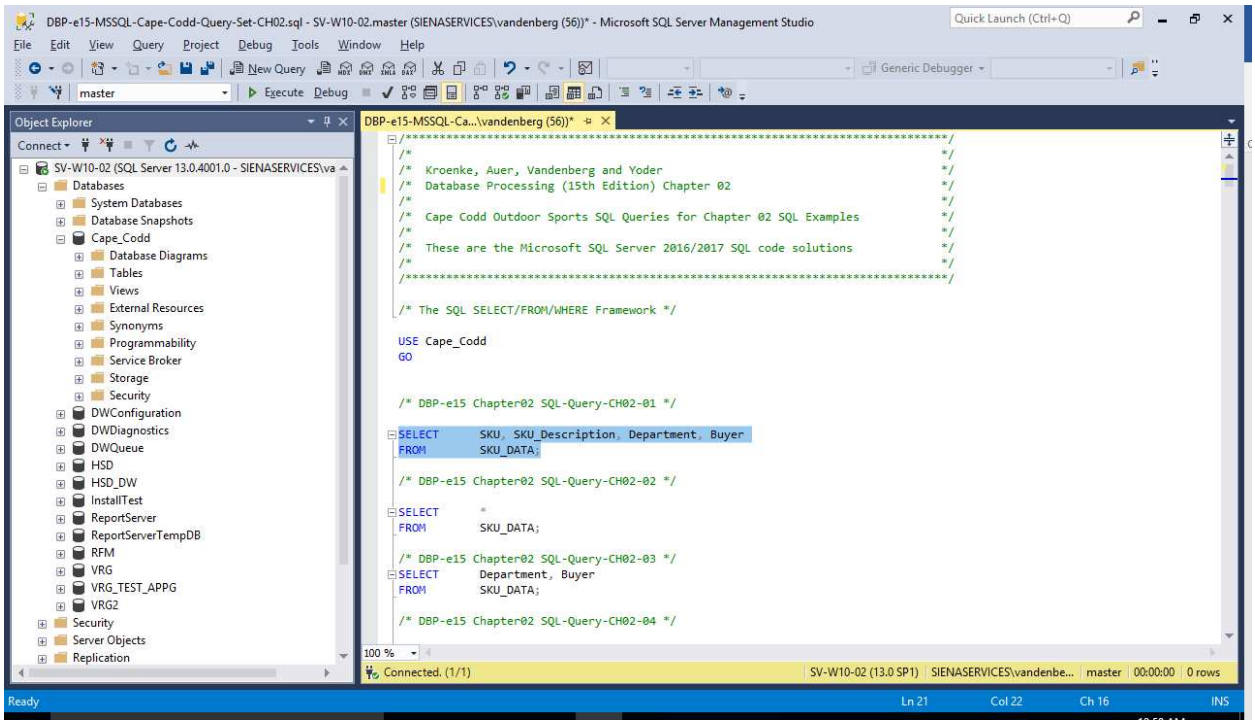
❖ TEACHING SUGGESTIONS

- 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, Exercises, the Marcia's Dry Cleaning Case Questions, and the 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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

- 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-e15-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-e15-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-e15-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 2016 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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

2017. 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 alias" part of the questions.

- Microsoft Access 2016 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 2017, 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 2016) enclosed in delimiters as [ORDER] if the query is going to run

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 2.51.
- 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 most of them are from Access 2016 but some are from older versions of Access: 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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 [04/26/2018]) 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 [2018]).

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, SKU_DATA, and BUYER tables. What is the relationship of these tables to the CATALOG_SKU_2017 and CATALOG_SKU_2018 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). Each SKU has a buyer who purchased the item for Cape Codd. 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:

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

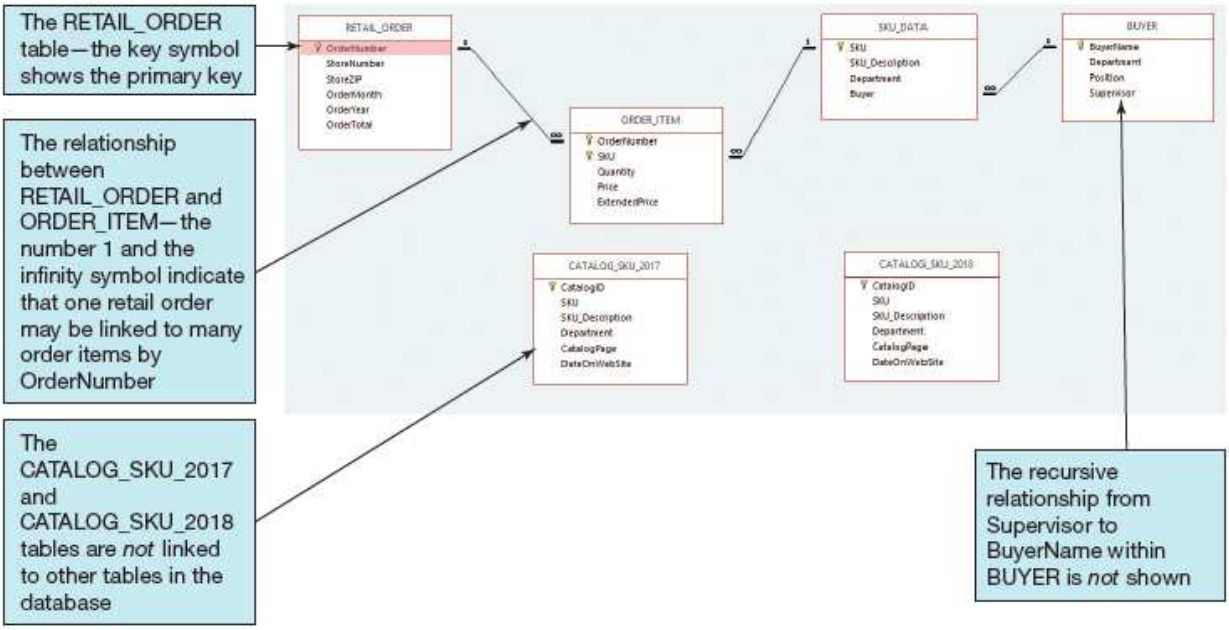


Figure 2-4 – The Cape Codd Database

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. Buyer in SKU_DATA is a foreign key linking to BuyerName in BUYER. Supervisor in BUYER is a foreign key linked to BuyerName in BUYER. 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*)

BUYER (BuyerName, Department, Position, *Supervisor*)

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, although Oracle implements much of that functionality. Alterations since SQL3 have mainly related to incorporating advanced data models (XML, JSON) and temporal data; the primary functionality for our purposes has been fixed for some time. The latest standard is SQL 2016, which added JSON support.

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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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). Also see the solution to Review Question 2.7.

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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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_2016 table. The table schemas for these tables, RETAIL_ORDER, ORDER_ITEM, SKU_DATA, BUYER, CATALOG_SKU_2017, and CATALOG_SKU_2018 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*)

BUYER (BuyerName, Department, Position, *Supervisor*)

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

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

CATALOG_SKU_2016 (CatalogID, SKU, SKU_Description, CatalogPage, DateOnWebSite)

CATALOG_SKU_2017 (CatalogID, SKU, SKU_Description, CatalogPage, DateOnWebSite)

CATALOG_SKU_2018 (CatalogID, SKU, SKU_Description, CatalogPage, DateOnWebSite)

The nine tables in the revised Cape Codd database schema are shown in Figure 2-35. The column characteristics for the WAREHOUSE table are shown in Figure 2-36, the column characteristics for the INVENTORY table are shown in Figure 2-37, and the column characteristics for the CATALOG_SKU_2016 table are shown in Figure 2-38. The data for the WAREHOUSE table are shown in Figure 2-39, the data for the INVENTORY table are shown in Figure 2-40, and the data for the CATALOG_SKU_2016 table are shown in Figure 2-41.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

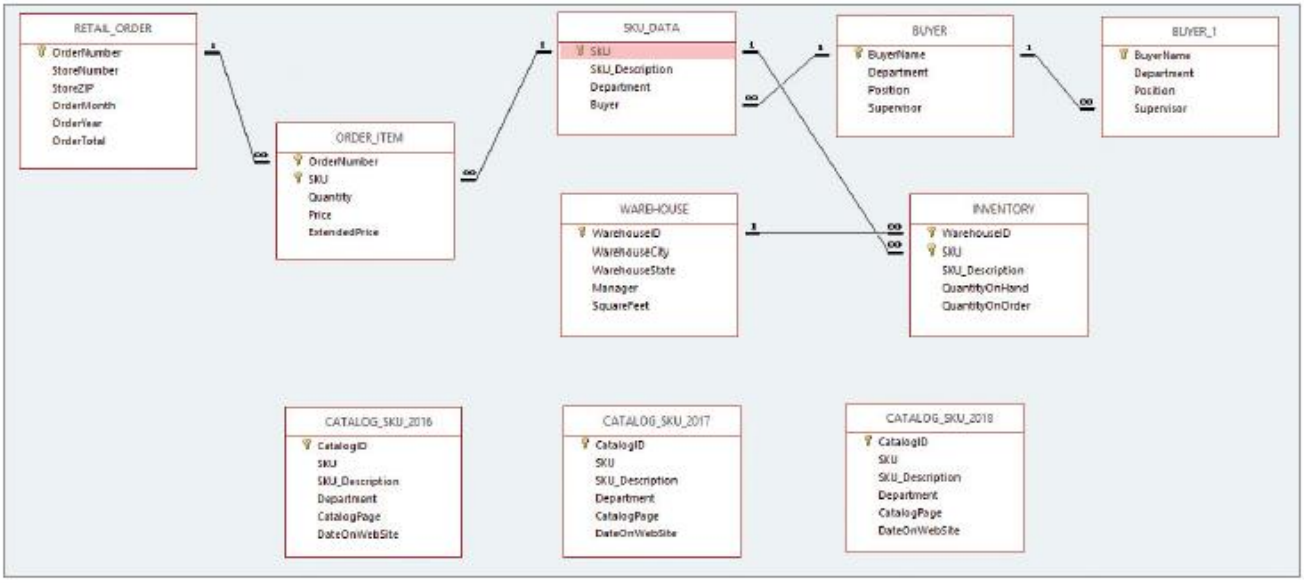


Figure 2-35 – The Cape Codd Database with the WAREHOUSE, INVENTORY, and CATALOG_SKU_2016 tables

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 2-36 - Column Characteristics for the Cape Codd Database WAREHOUSE Table

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 2-37 - Column Characteristics for the Cape Codd Database INVENTORY Table

CATALOG_SKU_2016

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	
DateOnWebSite	Date	No	No	

Figure 2-38 - Column Characteristics for the Cape Codd Database CATALOG_SKU_2016 Table

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 2-39 - Cape Codd Database WAREHOUSE Table Data

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 2-40 - Cape Codd Database INVENTORY Table Data

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

	CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
1	20160001	100100	Std. Scuba Tank, Yellow	Water Sports	23	2016-01-01
2	20160002	100500	Std. Scuba Tank, Light Green	Water Sports	NULL	2016-07-01
3	20160003	100600	Std. Scuba Tank, Light Green	Water Sports	NULL	2016-07-01
4	20160004	101100	Dive Mask, Small Clear	Water Sports	24	2016-01-01
5	20160005	101200	Dive Mask, Med Clear	Water Sports	24	2016-01-01
6	20160006	201000	Half-dome Tent	Camping	45	2016-01-01
7	20160007	202000	Half-dome Tent Vestibule	Camping	47	2016-01-01
8	20160008	301000	Light Fly Climbing Harness	Climbing	76	2016-01-01
9	20160009	302000	Locking Carabiner, Oval	Climbing	78	2016-01-01

Figure 2-41 - Cape Codd Database CATALOG_SKU_2016 Table Data

You will need to create and set up a database named Cape_Codd for use with the Cape Codd review questions. You may have already created this database as suggested in this chapter 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 2016 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 2017, see online Chapter 10A.*
- *For Oracle Database 12c Release 2 or Oracle Database XE, see online Chapter 10B.*
- *For MySQL 5.7 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 2016 when needed. Whenever possible, all results were obtained by running the SQL statements in Microsoft Access 2016, and the corresponding screen shots of the results are shown below. As explained in the text, some queries cannot be run in Microsoft Access 2016, and for those queries the correct result was obtained using Microsoft SQL Server 2017. The SQL statements shown should run with little, if any, modification needed for Oracle

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Database 12c Release 2, Oracle Database Express Edition 11g R2, and MySQL 5.7. A few of the queries illustrate some minor syntactic differences between the systems. In those cases, we have shown the minor changes necessary for Oracle Database and MySQL in this manual. In addition, the solution files for each system of course have working queries for that system.

Solutions to Review Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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 2017, Oracle Database 12c Release 2/Express Edition 11gR2, and MySQL 5.7 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:

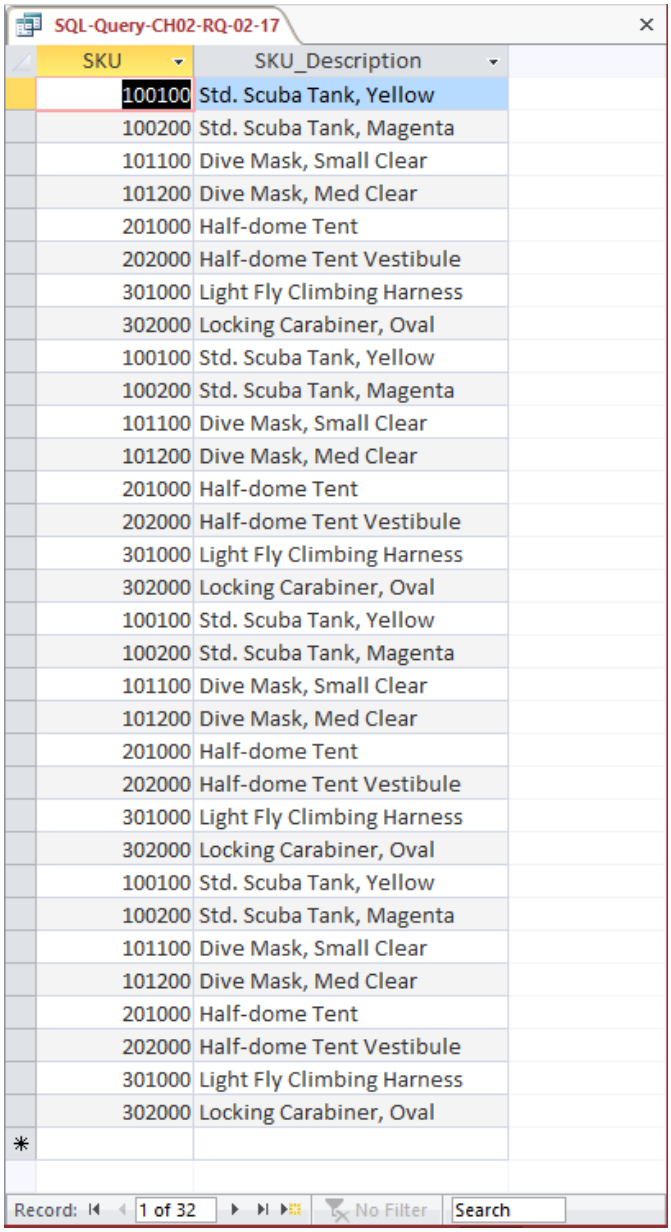
2.17 Write an SQL statement to display SKU and SKU_Description.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



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

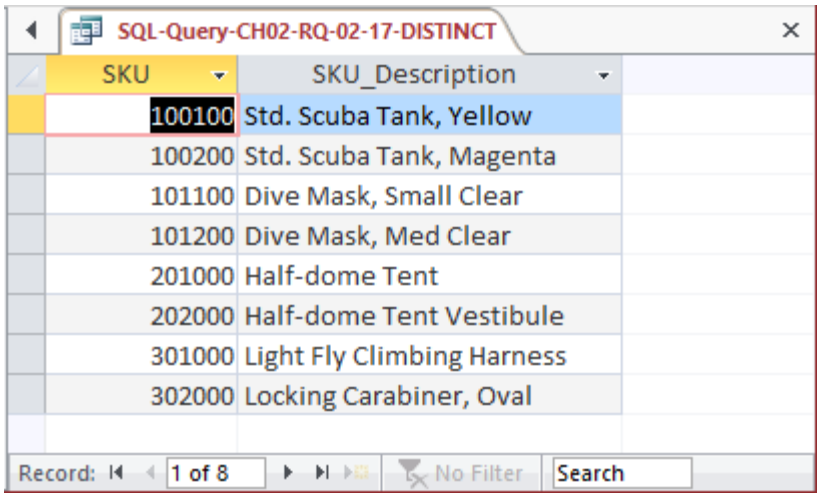
Record: 1 of 32 No Filter Search

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

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


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



2.18 Write an SQL statement to display SKU_Description and SKU.

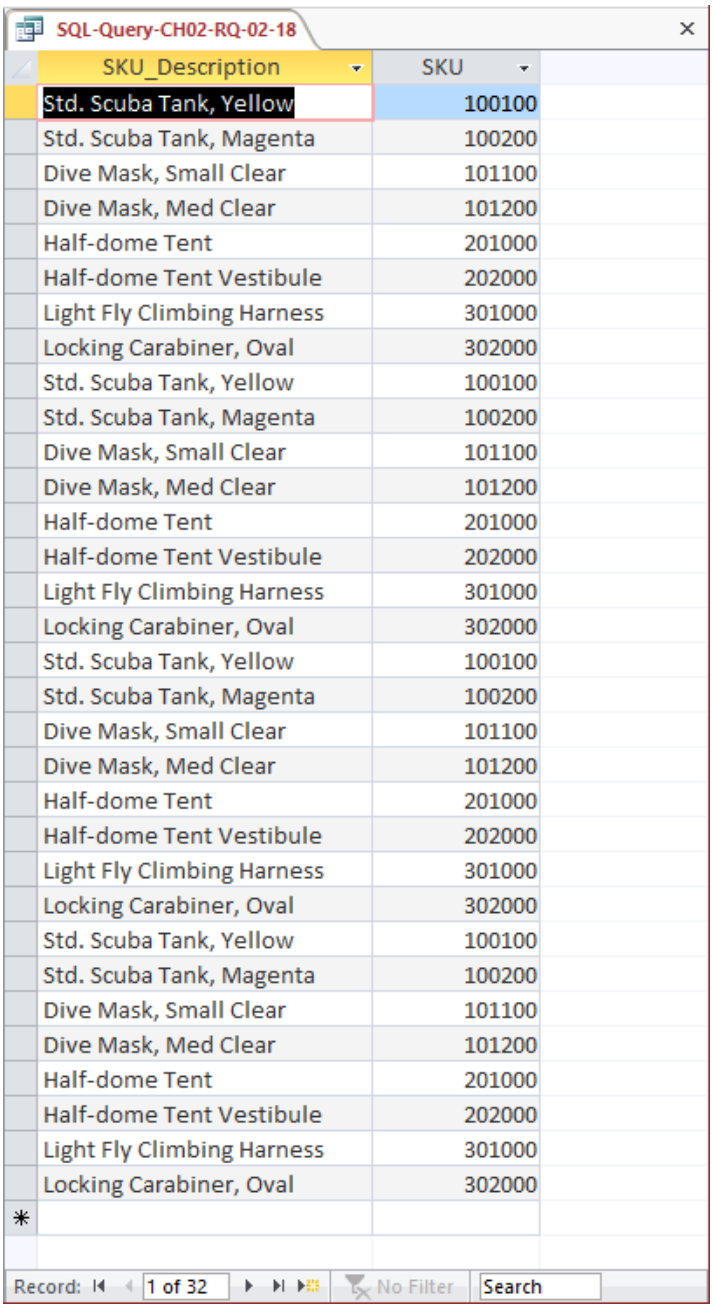
SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

```
SELECT      SKU_Description, SKU

FROM     INVENTORY;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



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

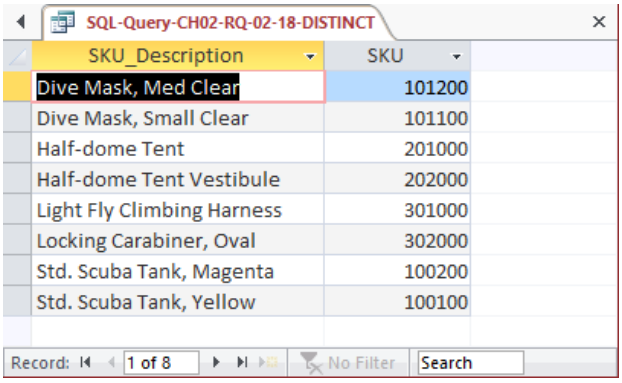
Record: 1 of 32 No Filter Search

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

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


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual


SQL-Query-CH02-RQ-02-18-DISTINCT

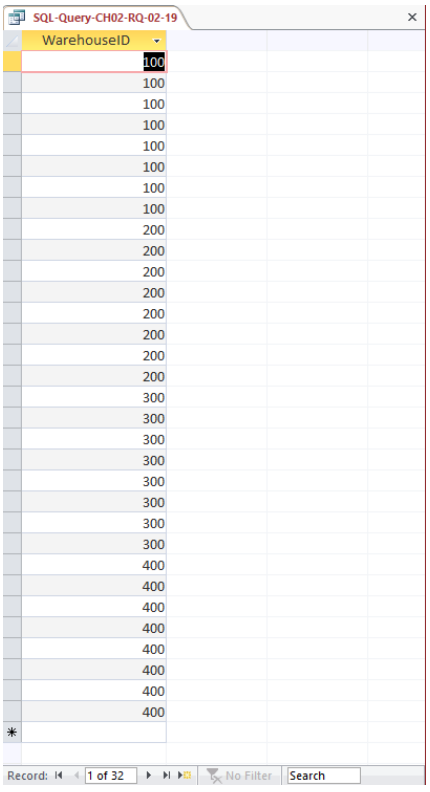
SKU_Description	SKU
Dive Mask, Med Clear	101200
Dive Mask, Small Clear	101100
Half-dome Tent	201000
Half-dome Tent Vestibule	202000
Light Fly Climbing Harness	301000
Locking Carabiner, Oval	302000
Std. Scuba Tank, Magenta	100200
Std. Scuba Tank, Yellow	100100

Record: 1 of 8 No Filter Search

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-e15-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;
```


SQL-Query-CH02-RQ-02-19

WarehouseID
100
100
100
100
100
100
100
100
200
200
200
200
200
200
200
200
300
300
300
300
300
300
300
300
400
400
400
400
400
400
400
400

Record: 1 of 32 No Filter Search

2.20 Write an SQL statement to display unique WarehouseIDs.

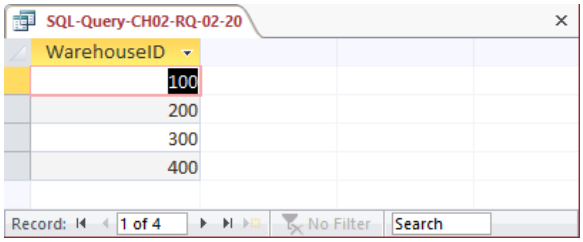
SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

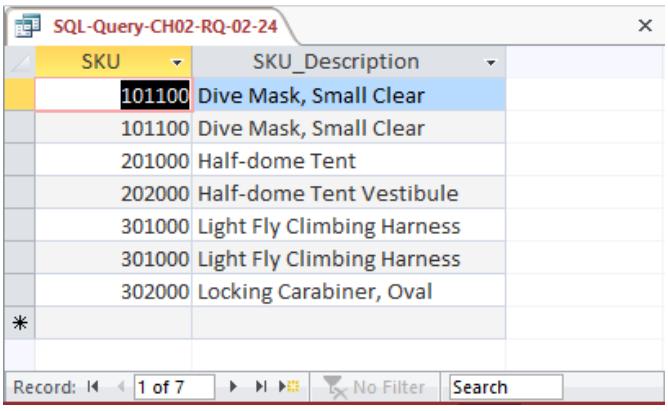
2.24 Write an SQL statement to display the SKU and SKU_Description for products having QuantityOnHand equal to 0.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



SQL-Query-CH02-RQ-02-24

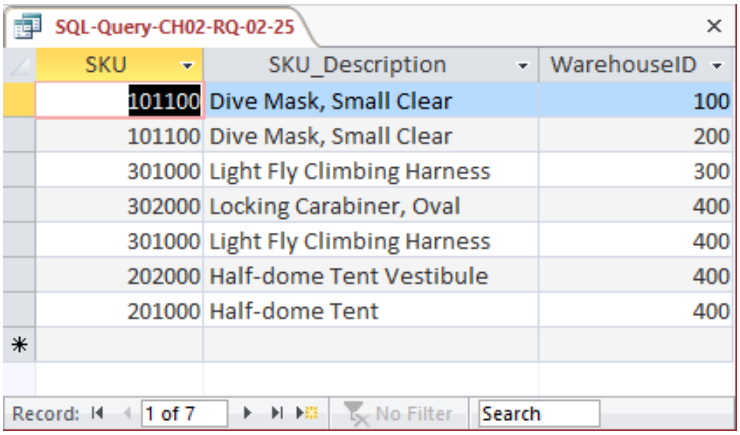
SKU	SKU_Description
101100	Dive Mask, Small Clear
101100	Dive Mask, Small Clear
201000	Half-dome Tent
202000	Half-dome Tent Vestibule
301000	Light Fly Climbing Harness
301000	Light Fly Climbing Harness
302000	Locking Carabiner, Oval

Record: 1 of 7 No Filter Search

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-e15-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;
```



SQL-Query-CH02-RQ-02-25

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

Record: 1 of 7 No Filter Search

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-e15-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).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
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

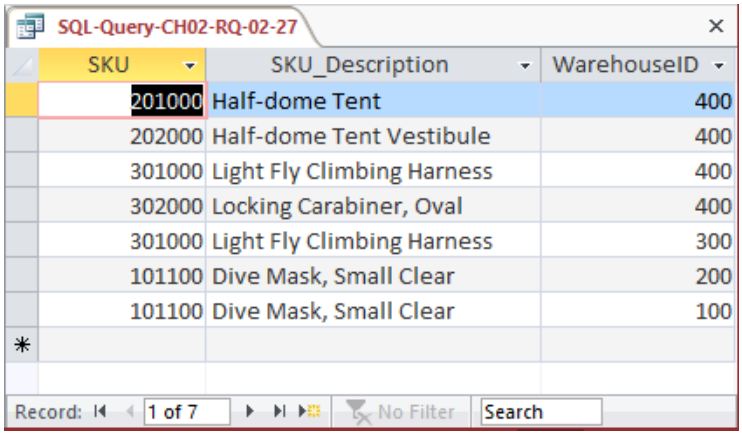
2.27 Write an SQL statement to display SKU, SKU_Description, and WarehouseID for all products that have a QuantityOnHand equal to 0 and a QuantityOnOrder greater than 0. Sort the results in descending order by WarehouseID and in ascending order by SKU.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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

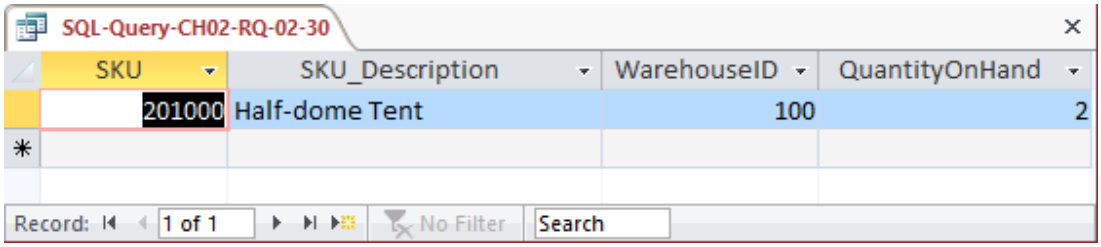
Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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;
```



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-e15-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).

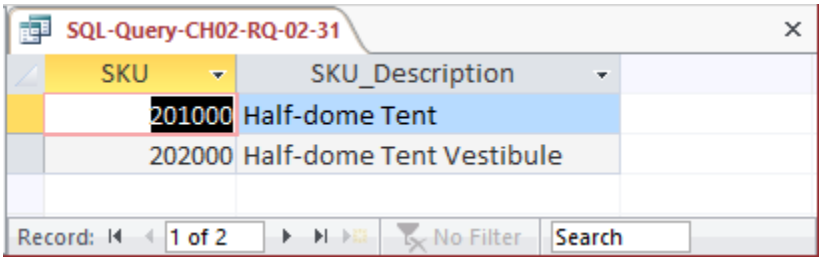
Note that, as discussed in Chapter 2, Microsoft Access 2016 uses wildcard characters that differ from the SQL standard.

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*';
```



Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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).

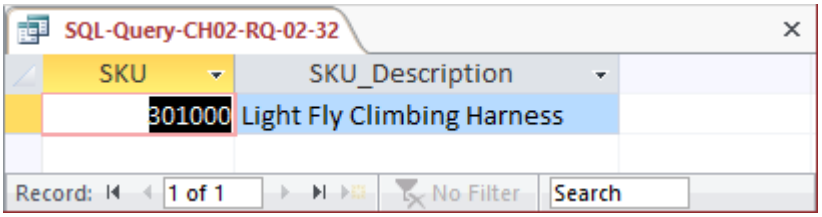
Note that, as discussed in Chapter 2, Microsoft Access 2016 uses wildcard characters that differ from the SQL standard.

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 with 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-e15-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 2016 uses wildcard characters that differ from the SQL standard.

For Microsoft SQL Server, Oracle Database, and MySQL:

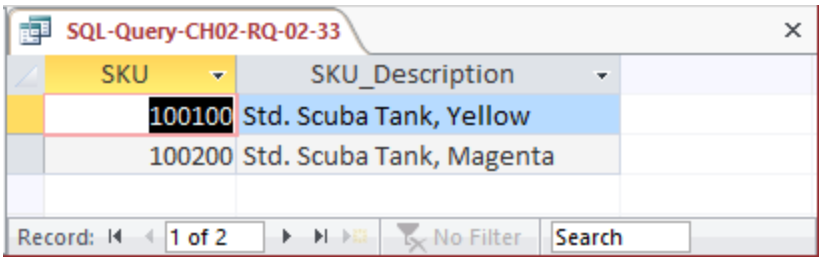
```
SELECT   DISTINCT SKU, SKU_Description
FROM     INVENTORY
WHERE    SKU_Description LIKE '__d%';
```

For Microsoft Access:

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


Copyright © 2019 Pearson Education, Inc.

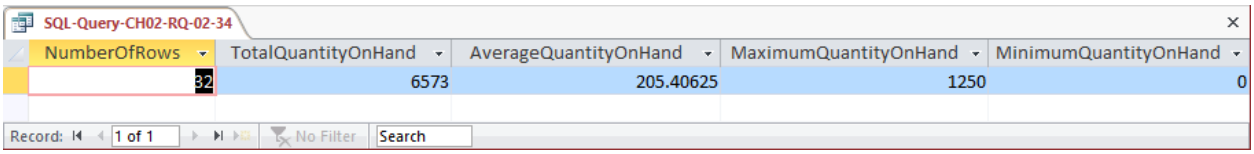
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



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-e15-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-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

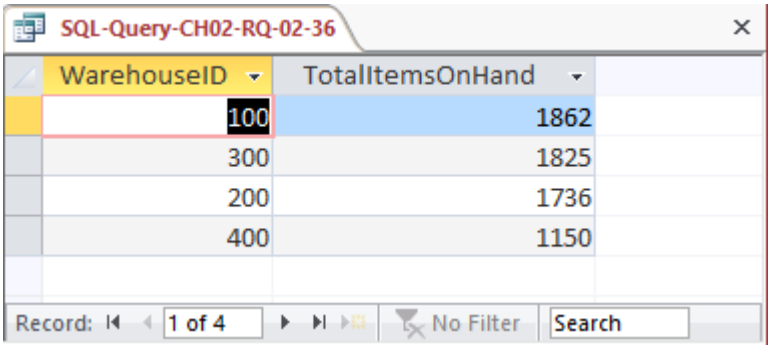
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 three or more items on hand from the sum, 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-e15-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:

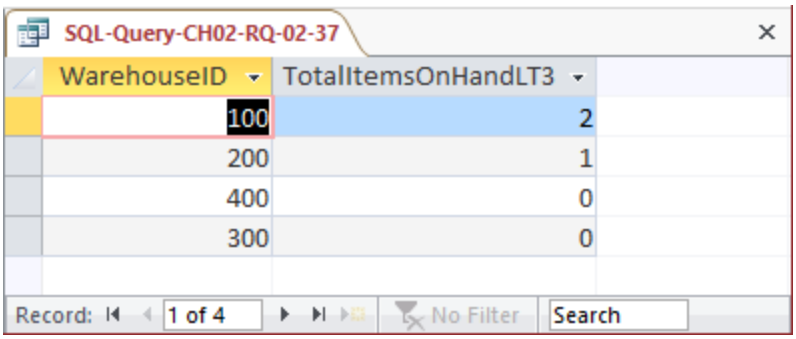
Unfortunately, Microsoft Access cannot process the ORDER BY clause because it contains an aliased computed result. To correct this, we use an SQL statement with the un-aliased computation:

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
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 three or more items on hand from the sum, and name the sum TotalItemsOnHandLT3. Show Warehouse ID only for warehouses having fewer than two 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-e15-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:

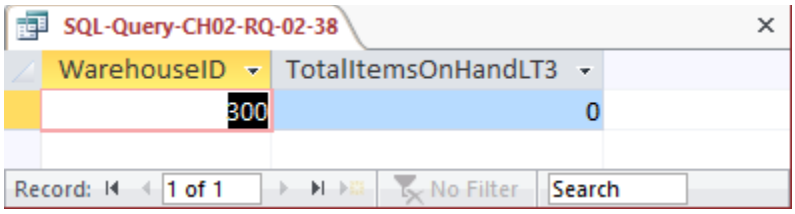
Unfortunately, Microsoft Access cannot process the ORDER BY clause because it contains an aliased computed result. To correct this, we use an SQL statement with the un-aliased computation:

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


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 clause or the HAVING clause 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-e15-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');
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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');
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
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-e15-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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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.

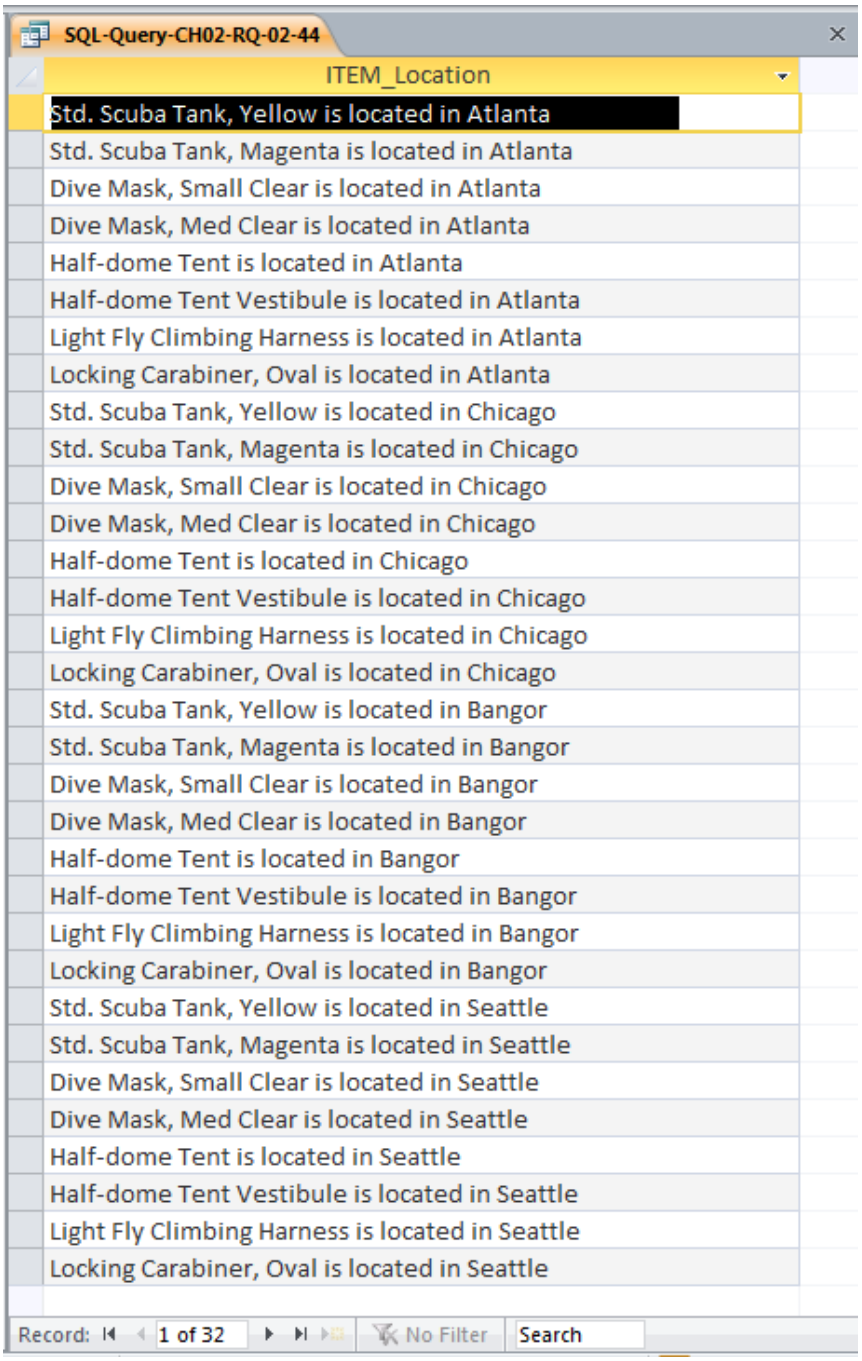
SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Note that the SQL syntax for string concatenation 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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



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, and 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-e15-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).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
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-e15-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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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.

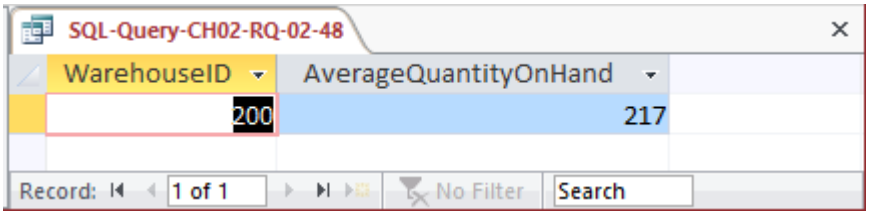
SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-IM-CH02-Cape-Codd-RQ.accdb* and in the corresponding files for SQL Server, Oracle Database, and MySQL, which are all available on the text's Web site (www.pearsonhighered.com/kroenke).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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;
```



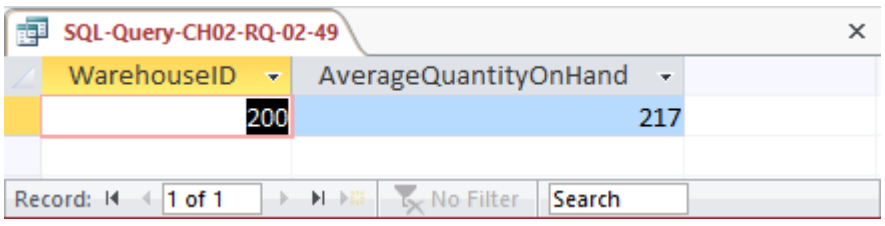
2.49 Write an SQL statement to show the WarehouseID and average QuantityOnHand of all items stored in a warehouse managed by 'Lucille Smith'. Use a join, but do not use JOIN ON syntax.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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.WarehouseID**—the query will NOT work without them.



Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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'.

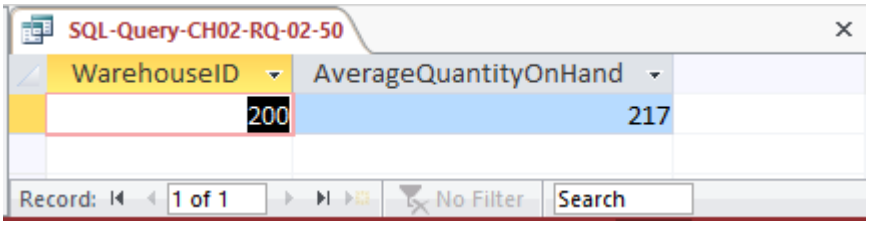
For Microsoft SQL Server, Oracle Database, and MySQL:

```
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;
```



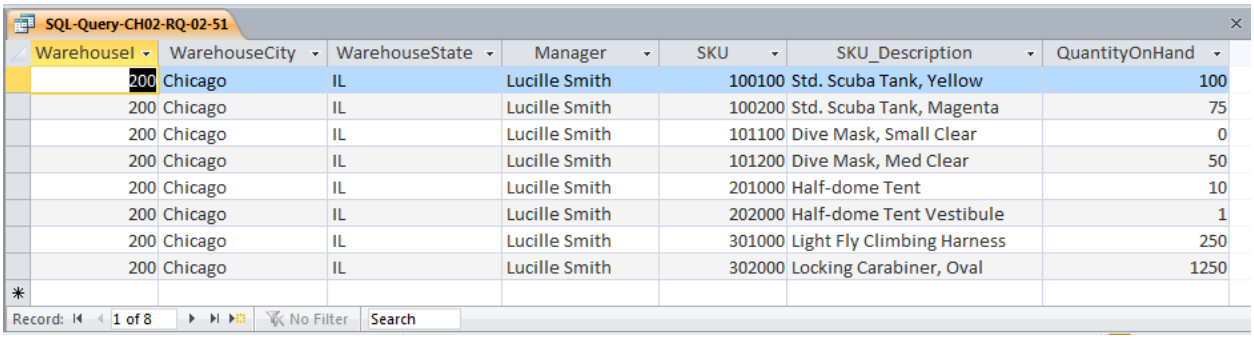
2.51 Write an SQL statement to show the WarehouseID, WarehouseCity, WarehouseState, Manager, SKU, SKU_Description, and QuantityOnHand of all items stored in a warehouse managed by 'Lucille Smith'. Use a join using JOIN ON syntax.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
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

WarehouseI	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';
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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 Review 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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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. Include all columns of both tables, but do not repeat the join columns.

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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   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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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_2016 and CATALOG_SKU_2017 tables to answer Review Questions 2.56 through 2.60 (for Microsoft Access 2016 and MySQL 5.7, 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 2016 Catalog (either in the printed catalog or on the Web site)* or *the Cape Codd 2017 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-e15-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_2016
UNION
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2017;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 2016 Catalog (only in the printed catalog itself)* or *the Cape Codd 2017 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-e15-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_2016
WHERE       CatalogPage IS NOT NULL
UNION
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2017
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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

2.58 Write an SQL statement to display the SKU, SKU_Description, and Department of all SKUs that appear in both *the Cape Codd 2016 Catalog (either in the printed catalog or on the Web site)* and *the Cape Codd 2017 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-e15-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_2016
INTERSECT
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2017;
```

For MySQL and Access:

```
SELECT      DISTINCT CS16.SKU, CS16.SKU_Description, CS16.Department
FROM        CATALOG_SKU_2016 AS CS16
      INNER JOIN  CATALOG_SKU_2017 AS CS17
ON          CS16.SKU = CS17.SKU;
```

SQL-Query-CH02-RQ-02-58

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

2.59 Write an SQL statement to display the SKU, SKU_Description, and Department of all SKUs that appear in both *the Cape Codd 2016 Catalog (only in the printed catalog itself)* and *the Cape Codd 2017 catalog (only in the printed catalog itself).*

SQL Solutions to Project Questions 2.17 – 2.60 are contained in the Microsoft Access database *DBP-e15-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).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

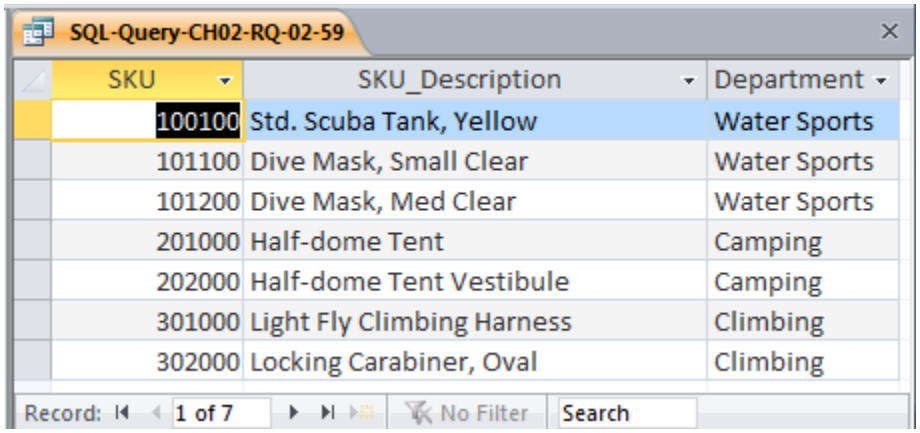
Note that Oracle Database and SQL Server support INTERSECT directly. In MySQL and Access INTERSECT is not supported but can be simulated using a join.

For Oracle and SQL Server:

```
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2016
WHERE       CatalogPage IS NOT NULL
INTERSECT
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2017
WHERE       CatalogPage IS NOT NULL;
```

For MySQL and Access:

```
SELECT      DISTINCT CS16.SKU, CS16.SKU_Description, CS16.Department
FROM        CATALOG_SKU_2016 AS CS16
      INNER JOIN  CATALOG_SKU_2017 AS CS17
ON                      CS16.SKU = CS17.SKU
WHERE CS16.CatalogPage IS NOT NULL AND CS17.CatalogPage IS NOT NULL;
```



2.60 *Write an SQL statement to display the SKU, SKU_Description, and Department of all SKUs that appear in* only *the Cape Codd 2016 Catalog (either in the printed catalog or on the Web site)* and not *in the Cape Codd 2017 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-e15-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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

For SQL Server:

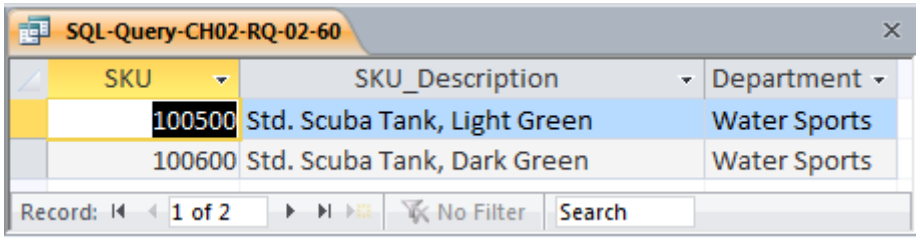
```
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2016
EXCEPT
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2017;
```

For Oracle:

```
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2016
MINUS
SELECT      SKU, SKU_Description, Department
FROM        CATALOG_SKU_2017;
```

For MySQL and Access:

```
SELECT      DISTINCT CS16.SKU, CS16.SKU_Description, CS16.Department
FROM        CATALOG_SKU_2016 AS CS16
      LEFT OUTER JOIN   CATALOG_SKU_2017 AS CS17
ON          CS16.SKU = CS17.SKU
WHERE       CS17.SKU IS NULL;
```



❖ ANSWERS TO EXERCISES

For this set of project questions, we will extend the Microsoft Access 2016 database for Wedgewood Pacific (WP) that we created in Chapter 1. Founded in 1987 in Seattle, Washington, this company manufactures and sells consumer drone aircraft. This is an innovative and rapidly expanding market. In January 2016, the FAA said that 181,000 drones (out of the approximately 700,000 drones that may have been sold during the 2015 Christmas season) had been registered under the new FAA drone registration rules.

WP currently produces three drone models: the Alpha III, the Bravo III, and the Delta IV. These products are created by WP's Research and Development group and produced at WP's production facilities. WP manufactures some of the parts used in the drones, but also purchases some parts from other suppliers.

The company is located in two buildings. One building houses the Administration, Lega., Finance, Accounting, Human Resources, and Sales and Marketing departments, and the second houses the Information Systems, Research and Development, and Production departments. The company database contains data about employees; departments; projects; assets, such as finished goods inventory, parts inventory, and computer equipment; and other

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

aspects of company operations. In the following project questions, we have already created the WP.accdb database with the following two tables (see Chapter 1 Project Questions):

DEPARTMENT (DepartmentName, BudgetCode, OfficeNumber, DepartmentPhone)

EMPLOYEE (EmployeeNumber, FirstName, LastName, *Department*, Position, *Supervisor*, OfficePhone, EmailAddress)

Now we will add in the following two tables:

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

ASSIGNMENT (*ProjectID*, *EmployeeNumber*, HoursWorked)

Where

Department in EMPLOYEE must exist in DepartmentName in DEPARTMENT

Supervisor in EMPLOYEE must exist in EmployeeNumber in EMPLOYEE

Department in PROJECT must exist in DepartmentName in DEPARTMENT

EmployeeNumber in ASSIGNMENT must exist in EmployeeNumber in EMPLOYEE

ProjectID in ASSIGNMENT must exist in ProjectID in PROJECT

The four tables in the revised WP database schema are shown in Figure 2-42 without the recursive relationship in the EMPLOYEE table (which we will add in the following exercise questions). The column characteristics for the PROJECT table are shown in Figure 2-43, and the column characteristics for the ASSIGNMENT table are shown in Figure 2-45. Data for the PROJECT table are shown in Figure 2-44, and the data for the ASSIGNMENT table are shown in Figure 2-46.

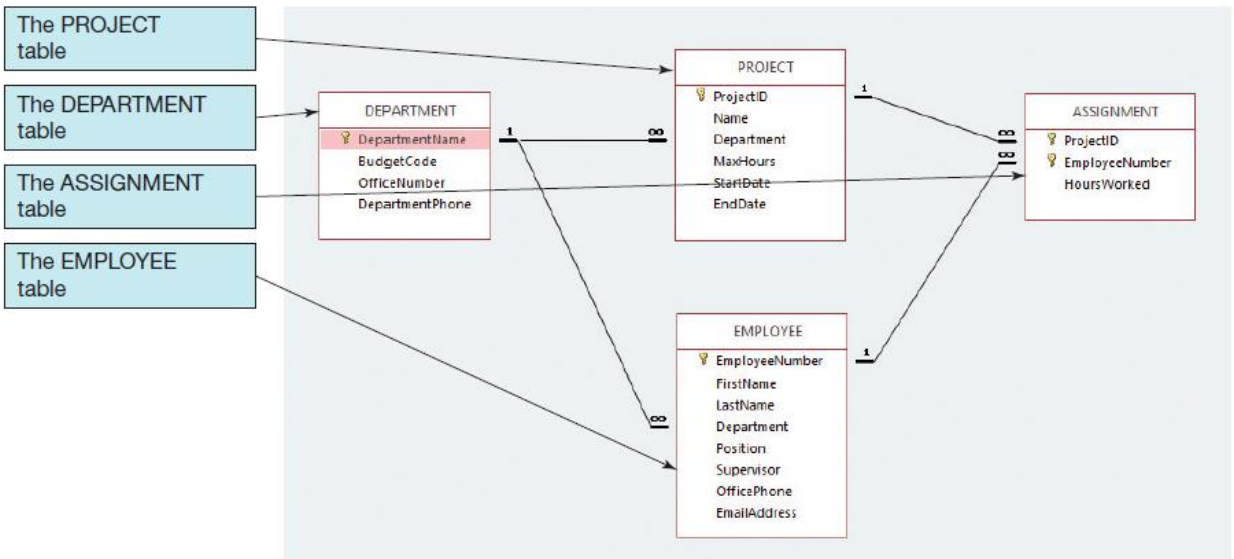


Figure 2-42 – The WP Database with the PROJECT and ASSIGNMENT Tables

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

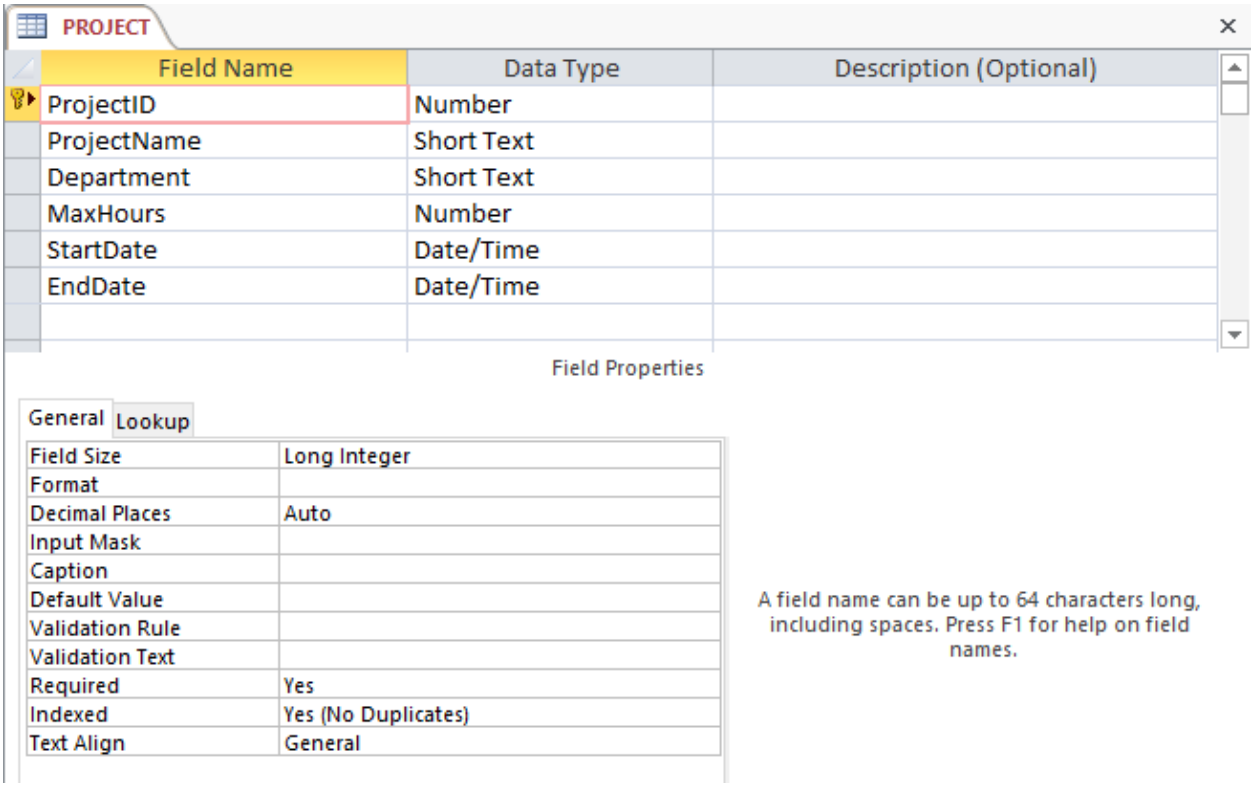
2.61 Figure 2-43 shows the column characteristics for the WP PROJECT table. Using the column characteristics, create the PROJECT table in the WP.accdb database.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e15-IM-CH02-WP.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	Yes	Surrogate Key
ProjectName	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 2-43 - Column Characteristics for the PROJECT Table

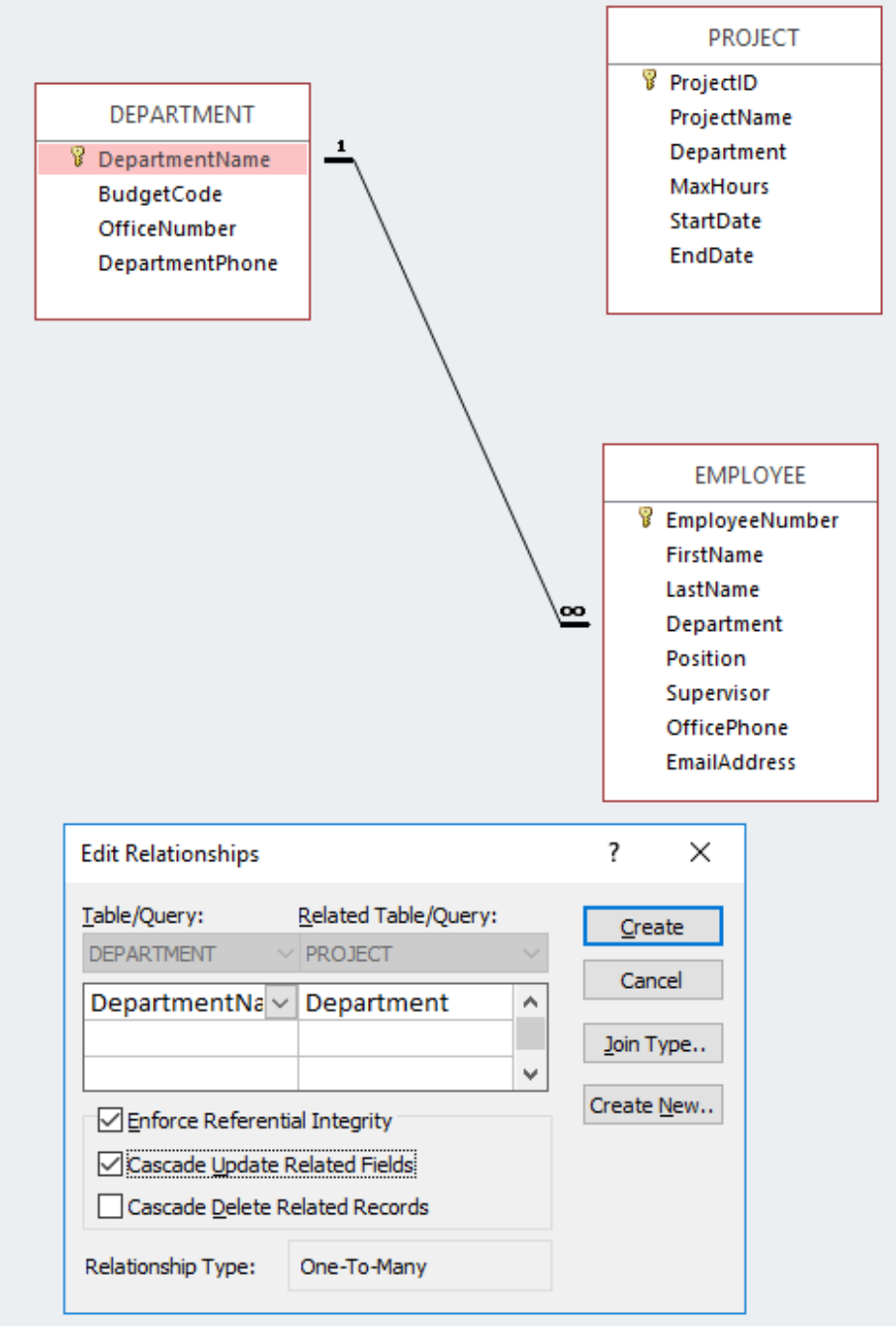


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-IM-CH02-WP.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).



Copyright © 2019 Pearson Education, Inc.

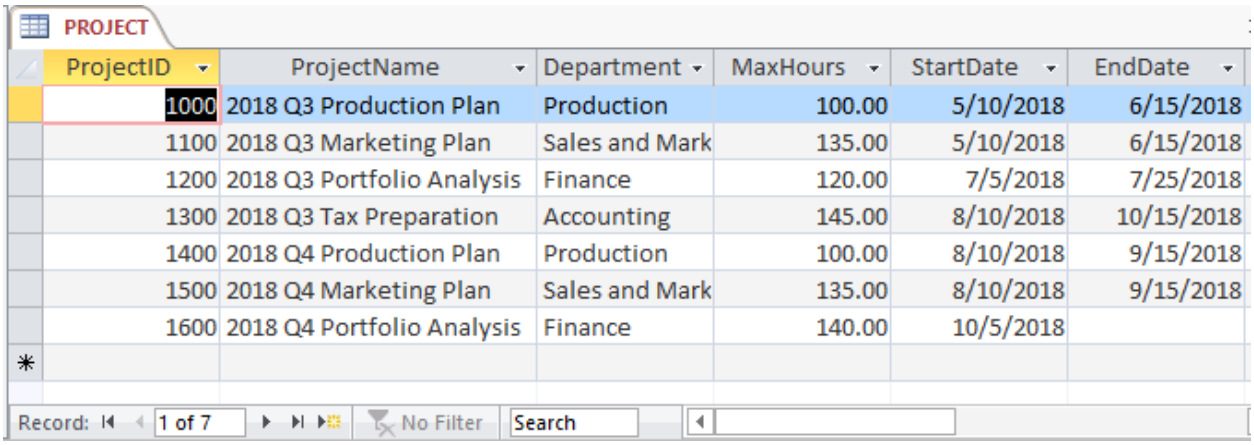
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

2.63 Figure 2-44 shows the data for the WP PROJECT table. Using the Datasheet view, enter the data shown in Figure 2-44 into your PROJECT table.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e15-IM-CH02-WP.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

ProjectID	ProjectName	Department	MaxHours	StartDate	EndDate
1000	2018 Q3 Production Plan	Production	100.00	05/10/18	06/15/18
1100	2018 Q3 Marketing Plan	Sales and Marketing	135.00	05/10/18	06/15/18
1200	2018 Q3 Portfolio Analysis	Finance	120.00	07/05/18	07/25/18
1300	2018 Q3 Tax Preparation	Accounting	145.00	08/10/18	10/15/18
1400	2018 Q4 Production Plan	Production	100.00	08/10/18	09/15/18
1500	2018 Q4 Marketing Plan	Sales and Marketing	135.00	08/10/18	09/15/18
1600	2018 Q4 Portfolio Analysis	Finance	140.00	10/05/18	

Figure 2-44 - Sample Data for the PROJECT Table



2.64 Figure 2-45 shows the column characteristics for the WP ASSIGNMENT table. Using the column characteristics, create the ASSIGNMENT table in the WP.accdb database.

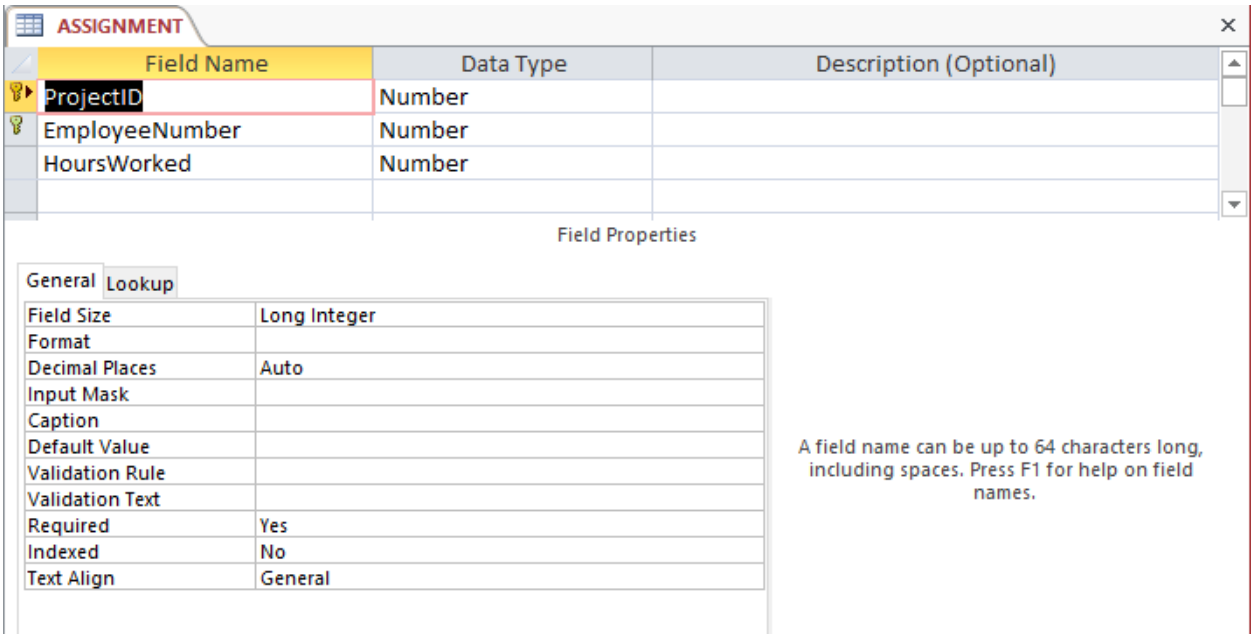
Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e15-IM-CH02-WP.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 2-45 - Column Characteristics for the ASSIGNMENT Table

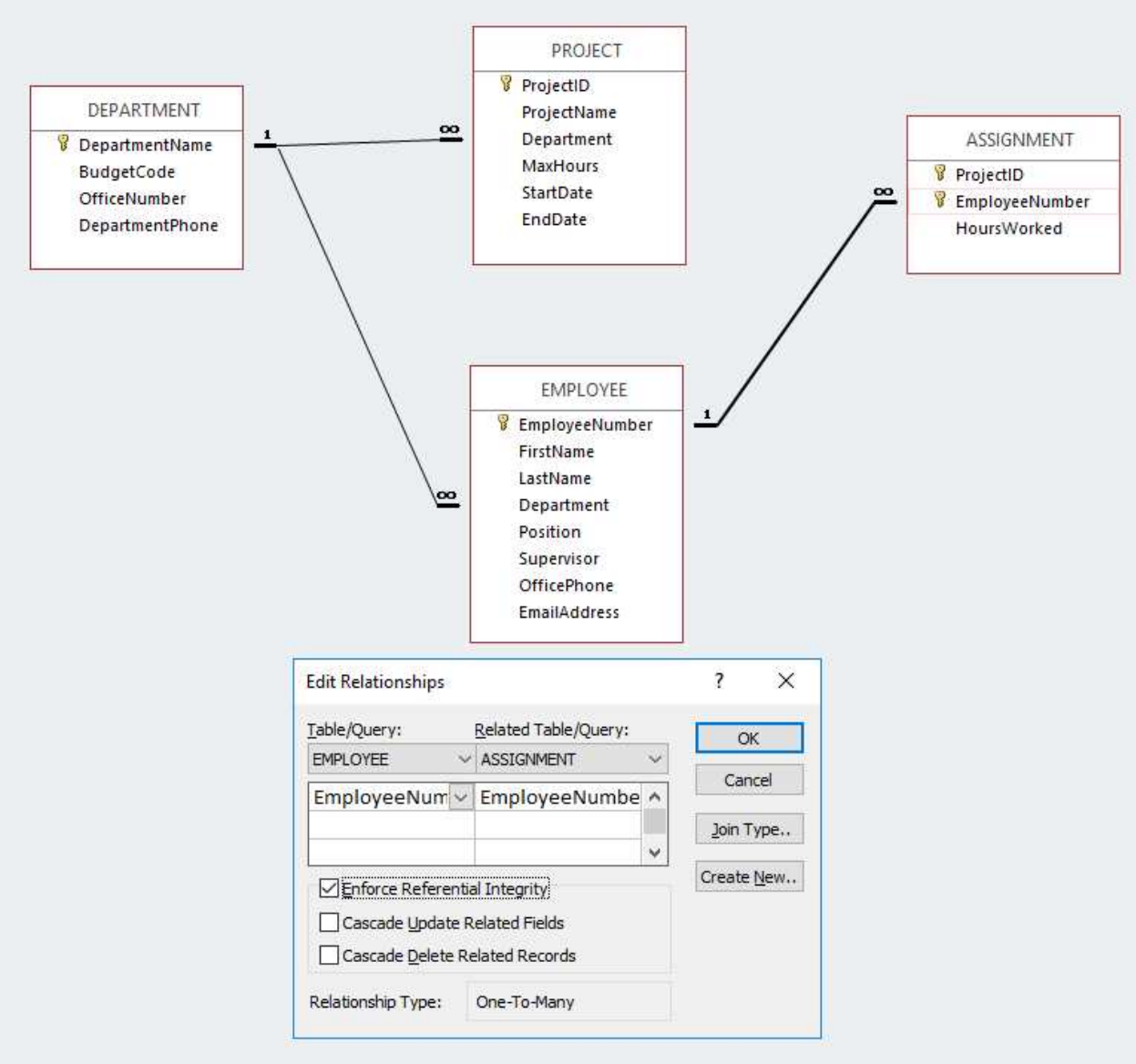


2.65 Create the relationship and referential integrity constraint between ASSIGNMENT and EMPLOYEE. In the Edit Relationship dialog box, enable enforcing of referential integrity, but do not enable either cascading updates or the cascading of data from deleted records.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e15-IM-CH02-WP.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

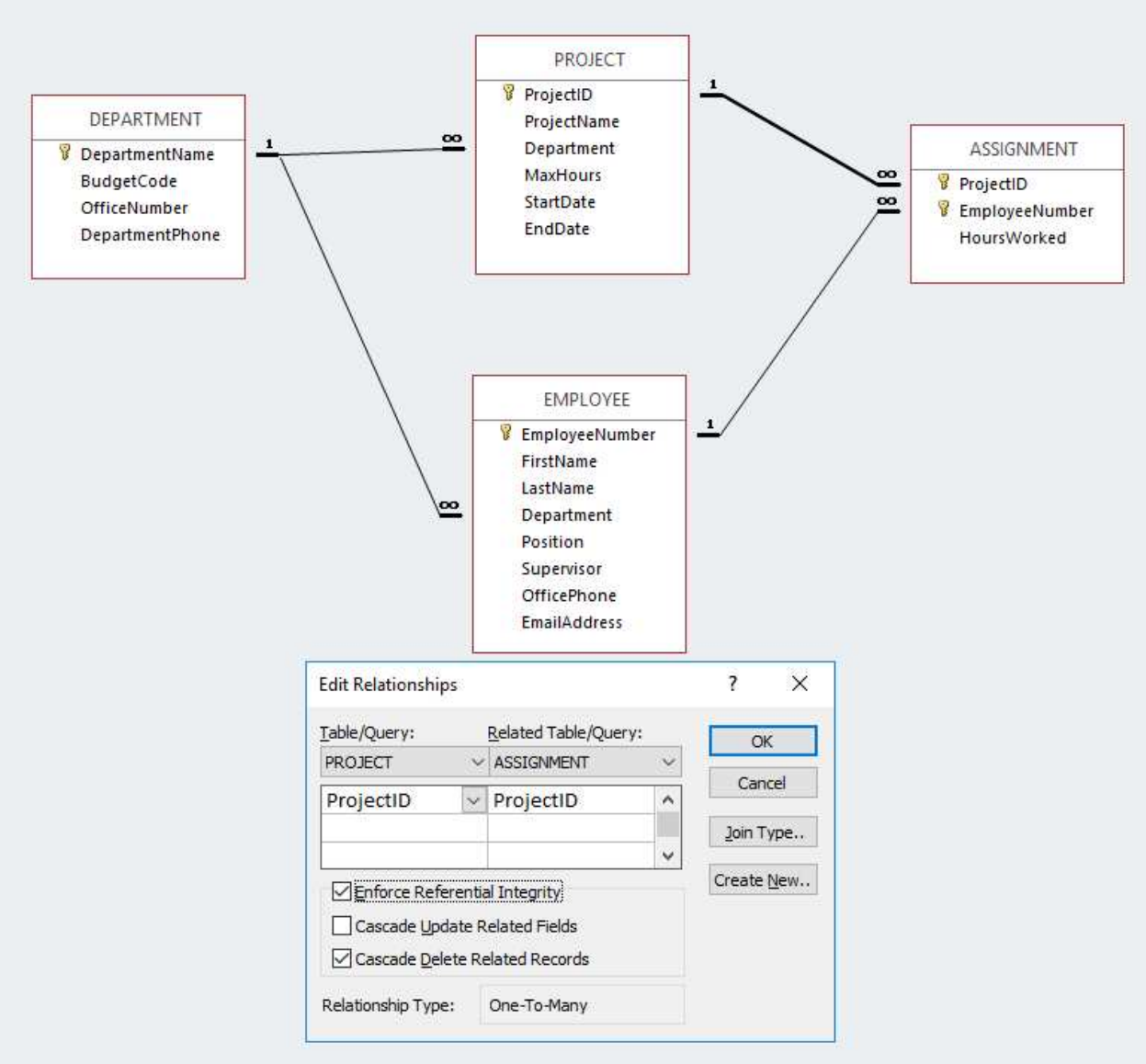


2.66 Create the relationship and referential integrity constraint between ASSIGNMENT and PROJECT. In the Edit Relationship dialog box, enable enforcing of referential integrity and cascading of deletes, but do not enable cascading updates.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e15-IM-CH02-WP.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



2.67 Figure 2-46 shows the data for the WP ASSIGNMENT table. Using the Datasheet view, enter the data shown in Figure 2-46 into your ASSIGNMENT table.

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e15-IM-CH02-WP.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

Copyright © 2019 Pearson Education, Inc.

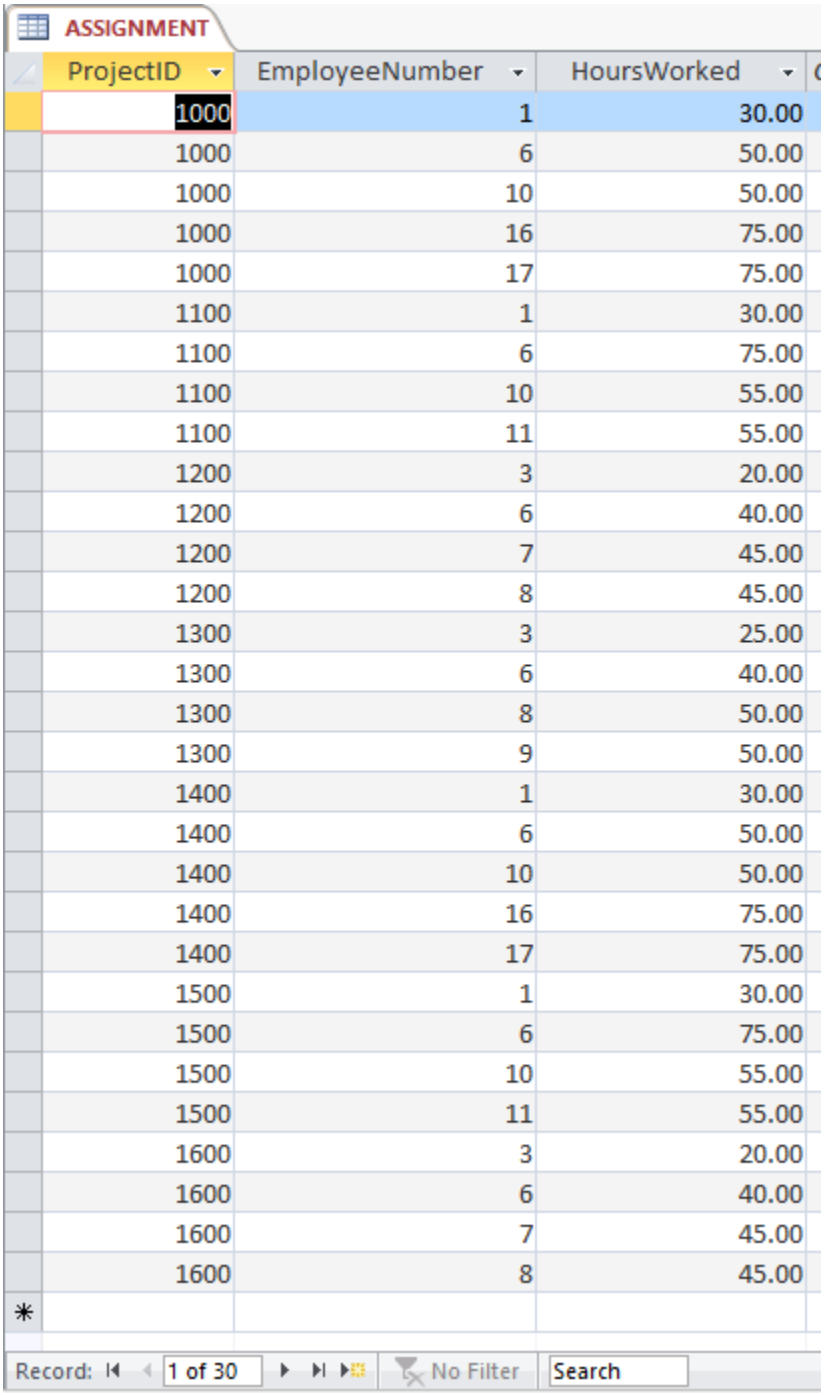
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

ProjectID	EmployeeNumber	HoursWorked
1000	1	30.00
1000	6	50.00
1000	10	50.00
1000	16	75.00
1000	17	75.00
1100	1	30.00
1100	6	75.00
1100	10	55.00
1100	11	55.00
1200	3	20.00
1200	6	40.00
1200	7	45.00
1200	8	45.00
1300	3	25.00
1300	6	40.00
1300	8	50.00
1300	9	50.00
1400	1	30.00
1400	6	50.00
1400	10	50.00
1400	16	75.00
1400	17	75.00
1500	1	30.00
1500	6	75.00
1500	10	55.00
1500	11	55.00
1600	3	20.00
1600	6	40.00
1600	7	45.00
1600	8	45.00

Figure 2-46 - Sample Data for the WP Database ASSIGNMENT Table

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



ASSIGNMENT

ProjectID	EmployeeNumber	HoursWorked
1000	1	30.00
1000	6	50.00
1000	10	50.00
1000	16	75.00
1000	17	75.00
1100	1	30.00
1100	6	75.00
1100	10	55.00
1100	11	55.00
1200	3	20.00
1200	6	40.00
1200	7	45.00
1200	8	45.00
1300	3	25.00
1300	6	40.00
1300	8	50.00
1300	9	50.00
1400	1	30.00
1400	6	50.00
1400	10	50.00
1400	16	75.00
1400	17	75.00
1500	1	30.00
1500	6	75.00
1500	10	55.00
1500	11	55.00
1600	3	20.00
1600	6	40.00
1600	7	45.00
1600	8	45.00

Record: 1 of 30 No Filter Search

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

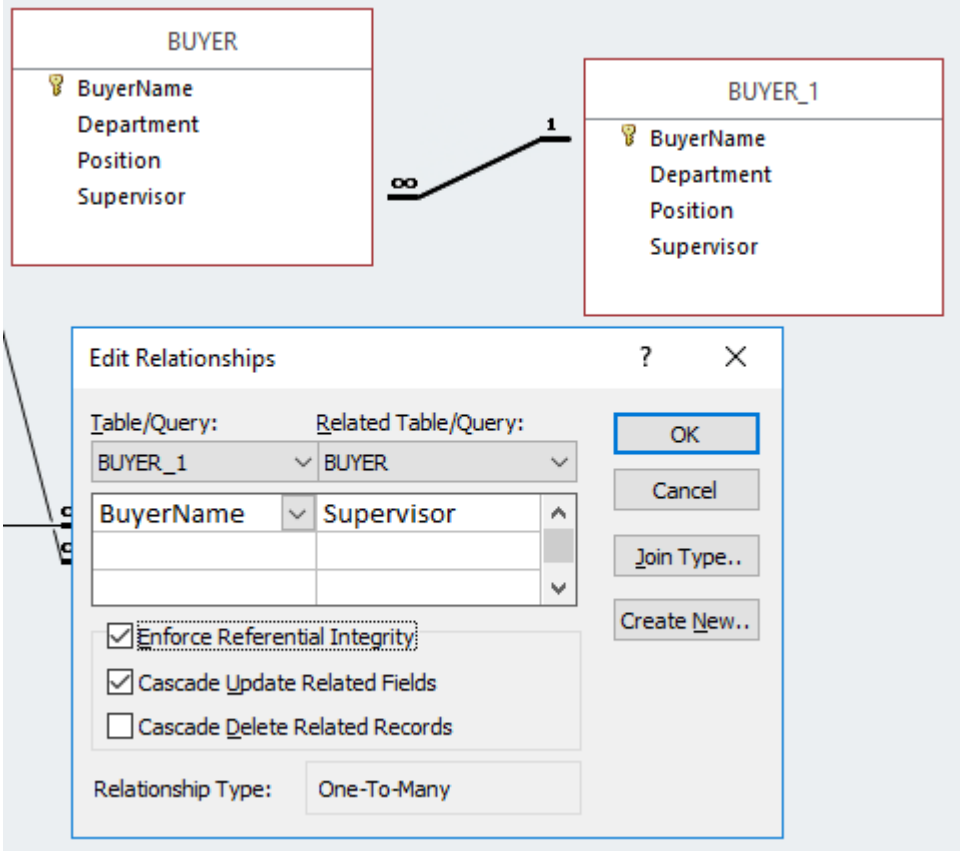
Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 Figure 2-31 for reference, create the recursive relationship and referential integrity constraint between Supervisor and BuyerName in BUYER. In the Edit Relationship dialog box, enable enforcing of referential integrity and cascading up data updates, but do not enable cascading of data from deleted records. HINT: to create a recursive relationship, add another copy of the BUYER table to the relationships window by right-clicking in that window and selecting Show Table.*

Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database DBP-e15-IM-CH02-WP.accdb which is available on the text's Web site (www.pearsonhighered.com/kroenke).



2.70 *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*

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-IM-CH02-WP.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	ProjectName	Department	MaxHours	StartDate	EndDate
1000	2018 Q3 Production Plan	Production	100.00	5/10/2018	6/15/2018
1100	2018 Q3 Marketing Plan	Sales and Mark	135.00	5/10/2018	6/15/2018
1200	2018 Q3 Portfolio Analysis	Finance	120.00	7/5/2018	7/25/2018
1300	2018 Q3 Tax Preparation	Accounting	145.00	8/10/2018	10/15/2018
1400	2018 Q4 Production Plan	Production	100.00	8/10/2018	9/15/2018
1500	2018 Q4 Marketing Plan	Sales and Mark	135.00	8/10/2018	9/15/2018
1600	2018 Q4 Portfolio Analysis	Finance	140.00	10/5/2018	

Record: 1 of 7 No Filter Search

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

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

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

SQL-Query-02-B

ProjectID	ProjectName	StartDate	EndDate
1000	2018 Q3 Production Plan	5/10/2018	6/15/2018
1100	2018 Q3 Marketing Plan	5/10/2018	6/15/2018
1200	2018 Q3 Portfolio Analysis	7/5/2018	7/25/2018
1300	2018 Q3 Tax Preparation	8/10/2018	10/15/2018
1400	2018 Q4 Production Plan	8/10/2018	9/15/2018
1500	2018 Q4 Marketing Plan	8/10/2018	9/15/2018
1600	2018 Q4 Portfolio Analysis	10/5/2018	

Record: 1 of 7 No Filter Search

C. What projects in the PROJECT table started before August 1, 2018? Show all the information for each project.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

SELECT    *
FROM      PROJECT
WHERE     StartDate < #01-AUG-18#;
```

SQL-Query-02-C

ProjectID	ProjectName	Department	MaxHours	StartDate	EndDate
1000	2018 Q3 Production Plan	Production	100.00	5/10/2018	6/15/2018
1100	2018 Q3 Marketing Plan	Sales and Mark	135.00	5/10/2018	6/15/2018
1200	2018 Q3 Portfolio Analysis	Finance	120.00	7/5/2018	7/25/2018

Record: 1 of 3 No Filter Search

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	ProjectName	Department	MaxHours	StartDate	EndDate
1600	2018 Q4 Portfolio Analysis	Finance	140.00	10/5/2018	

Record: 1 of 1 No Filter Search

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

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

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


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL-Query-02-E

ProjectID	EmployeeNumber	LastName	FirstName	OfficePhone
1000	1	Jacobs	Mary	360-285-8110
1100	1	Jacobs	Mary	360-285-8110
1400	1	Jacobs	Mary	360-285-8110
1500	1	Jacobs	Mary	360-285-8110
1200	3	Bandalone	Richard	360-285-8210
1300	3	Bandalone	Richard	360-285-8210
1600	3	Bandalone	Richard	360-285-8210
1000	6	Evans	Ken	360-285-8410
1100	6	Evans	Ken	360-285-8410
1200	6	Evans	Ken	360-285-8410
1300	6	Evans	Ken	360-285-8410
1400	6	Evans	Ken	360-285-8410
1500	6	Evans	Ken	360-285-8410
1600	6	Evans	Ken	360-285-8410
1200	7	Abernathy	Mary	360-285-8420
1600	7	Abernathy	Mary	360-285-8420
1200	8	Caruthers	Tom	360-285-8430
1300	8	Caruthers	Tom	360-285-8430
1600	8	Caruthers	Tom	360-285-8430
1300	9	Jones	Heather	360-285-8440
1000	10	Numoto	Ken	360-285-8510
1100	10	Numoto	Ken	360-285-8510
1400	10	Numoto	Ken	360-285-8510
1500	10	Numoto	Ken	360-285-8510
1100	11	Granger	Linda	360-285-8520
1500	11	Granger	Linda	360-285-8520
1000	16	Smith	Mary	360-285-8810
1400	16	Smith	Mary	360-285-8810
1000	17	Jackson	Tom	360-285-8820
1400	17	Jackson	Tom	360-285-8820

Record: 1 of 30 No Filter Search

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

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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

SELECT     P.ProjectID, ProjectName,
           P.Department AS ProjectDepartment,
           E.EmployeeNumber, LastName, FirstName,
           OfficePhone 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	2018 Q3 Production Plan	Production	1	Jacobs	Mary	360-285-8110
1000	2018 Q3 Production Plan	Production	6	Evans	Ken	360-285-8410
1000	2018 Q3 Production Plan	Production	10	Numoto	Ken	360-285-8510
1000	2018 Q3 Production Plan	Production	16	Smith	Mary	360-285-8810
1000	2018 Q3 Production Plan	Production	17	Jackson	Tom	360-285-8820
1100	2018 Q3 Marketing Plan	Sales and Marketing	1	Jacobs	Mary	360-285-8110
1100	2018 Q3 Marketing Plan	Sales and Marketing	6	Evans	Ken	360-285-8410
1100	2018 Q3 Marketing Plan	Sales and Marketing	10	Numoto	Ken	360-285-8510
1100	2018 Q3 Marketing Plan	Sales and Marketing	11	Granger	Linda	360-285-8520
1200	2018 Q3 Portfolio Analysis	Finance	3	Bandalone	Richard	360-285-8210
1200	2018 Q3 Portfolio Analysis	Finance	6	Evans	Ken	360-285-8410
1200	2018 Q3 Portfolio Analysis	Finance	7	Abernathy	Mary	360-285-8420
1200	2018 Q3 Portfolio Analysis	Finance	8	Caruthers	Tom	360-285-8430
1300	2018 Q3 Tax Preparation	Accounting	3	Bandalone	Richard	360-285-8210
1300	2018 Q3 Tax Preparation	Accounting	6	Evans	Ken	360-285-8410
1300	2018 Q3 Tax Preparation	Accounting	8	Caruthers	Tom	360-285-8430
1300	2018 Q3 Tax Preparation	Accounting	9	Jones	Heather	360-285-8440
1400	2018 Q4 Production Plan	Production	1	Jacobs	Mary	360-285-8110
1400	2018 Q4 Production Plan	Production	6	Evans	Ken	360-285-8410
1400	2018 Q4 Production Plan	Production	10	Numoto	Ken	360-285-8510
1400	2018 Q4 Production Plan	Production	16	Smith	Mary	360-285-8810
1400	2018 Q4 Production Plan	Production	17	Jackson	Tom	360-285-8820
1500	2018 Q4 Marketing Plan	Sales and Marketing	1	Jacobs	Mary	360-285-8110
1500	2018 Q4 Marketing Plan	Sales and Marketing	6	Evans	Ken	360-285-8410
1500	2018 Q4 Marketing Plan	Sales and Marketing	10	Numoto	Ken	360-285-8510
1500	2018 Q4 Marketing Plan	Sales and Marketing	11	Granger	Linda	360-285-8520
1600	2018 Q4 Portfolio Analysis	Finance	3	Bandalone	Richard	360-285-8210
1600	2018 Q4 Portfolio Analysis	Finance	6	Evans	Ken	360-285-8410
1600	2018 Q4 Portfolio Analysis	Finance	7	Abernathy	Mary	360-285-8420
1600	2018 Q4 Portfolio Analysis	Finance	8	Caruthers	Tom	360-285-8430

Record: 1 of 30 No Filter Search

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

Note the use of the aliases **ProjectDept**, and **EmpPhone**.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

SELECT     P.ProjectID, ProjectName,
           D.DepartmentName AS ProjectDept,
           DepartmentPhone,
           E.EmployeeNumber, LastName, FirstName,
           E.OfficePhone AS EmpPhone
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	ProjectDept	DepartmentPhone	EmployeeNum	LastName	FirstName	EmpPhone
1000	2018 Q3 Production Plan	Production	360-285-8800	6	Evans	Ken	360-285-8410
1000	2018 Q3 Production Plan	Production	360-285-8800	10	Numoto	Ken	360-285-8510
1000	2018 Q3 Production Plan	Production	360-285-8800	16	Smith	Mary	360-285-8810
1000	2018 Q3 Production Plan	Production	360-285-8800	17	Jackson	Tom	360-285-8820
1000	2018 Q3 Production Plan	Production	360-285-8800	1	Jacobs	Mary	360-285-8110
1100	2018 Q3 Marketing Plan	Sales and Marketing	360-285-8500	1	Jacobs	Mary	360-285-8110
1100	2018 Q3 Marketing Plan	Sales and Marketing	360-285-8500	6	Evans	Ken	360-285-8410
1100	2018 Q3 Marketing Plan	Sales and Marketing	360-285-8500	10	Numoto	Ken	360-285-8510
1100	2018 Q3 Marketing Plan	Sales and Marketing	360-285-8500	11	Granger	Linda	360-285-8520
1200	2018 Q3 Portfolio Analysis	Finance	360-285-8400	3	Bandalone	Richard	360-285-8210
1200	2018 Q3 Portfolio Analysis	Finance	360-285-8400	6	Evans	Ken	360-285-8410
1200	2018 Q3 Portfolio Analysis	Finance	360-285-8400	7	Abernathy	Mary	360-285-8420
1200	2018 Q3 Portfolio Analysis	Finance	360-285-8400	8	Caruthers	Tom	360-285-8430
1300	2018 Q3 Tax Preparation	Accounting	360-285-8405	8	Caruthers	Tom	360-285-8430
1300	2018 Q3 Tax Preparation	Accounting	360-285-8405	6	Evans	Ken	360-285-8410
1300	2018 Q3 Tax Preparation	Accounting	360-285-8405	9	Jones	Heather	360-285-8440
1300	2018 Q3 Tax Preparation	Accounting	360-285-8405	3	Bandalone	Richard	360-285-8210
1400	2018 Q4 Production Plan	Production	360-285-8800	1	Jacobs	Mary	360-285-8110
1400	2018 Q4 Production Plan	Production	360-285-8800	6	Evans	Ken	360-285-8410
1400	2018 Q4 Production Plan	Production	360-285-8800	10	Numoto	Ken	360-285-8510
1400	2018 Q4 Production Plan	Production	360-285-8800	16	Smith	Mary	360-285-8810
1400	2018 Q4 Production Plan	Production	360-285-8800	17	Jackson	Tom	360-285-8820
1500	2018 Q4 Marketing Plan	Sales and Marketing	360-285-8500	1	Jacobs	Mary	360-285-8110
1500	2018 Q4 Marketing Plan	Sales and Marketing	360-285-8500	6	Evans	Ken	360-285-8410
1500	2018 Q4 Marketing Plan	Sales and Marketing	360-285-8500	10	Numoto	Ken	360-285-8510
1500	2018 Q4 Marketing Plan	Sales and Marketing	360-285-8500	11	Granger	Linda	360-285-8520
1600	2018 Q4 Portfolio Analysis	Finance	360-285-8400	8	Caruthers	Tom	360-285-8430
1600	2018 Q4 Portfolio Analysis	Finance	360-285-8400	3	Bandalone	Richard	360-285-8210
1600	2018 Q4 Portfolio Analysis	Finance	360-285-8400	6	Evans	Ken	360-285-8410
1600	2018 Q4 Portfolio Analysis	Finance	360-285-8400	7	Abernathy	Mary	360-285-8420

Record: 1 of 30 No Filter Search

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

Note the use of the aliases **ProjectDepartment** and **EmployeePhone**.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

SELECT    P.ProjectID, ProjectName,
          D.DepartmentName AS ProjectDepartment,
          DepartmentPhone,
          E.EmployeeNumber, LastName, FirstName,
          E.OfficePhone 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='Sales and Marketing'
ORDER BY  P.ProjectID;
```

SQL-Query-02-H

ProjectID	ProjectName	ProjectDepartment	DepartmentPhone	EmployeeNumber	LastName	FirstName	EmployeePhone
1100	2018 Q3 Marketing Plan	Sales and Marketing	360-285-8500	11	Granger	Linda	360-285-8520
1100	2018 Q3 Marketing Plan	Sales and Marketing	360-285-8500	10	Numoto	Ken	360-285-8510
1100	2018 Q3 Marketing Plan	Sales and Marketing	360-285-8500	6	Evans	Ken	360-285-8410
1100	2018 Q3 Marketing Plan	Sales and Marketing	360-285-8500	1	Jacobs	Mary	360-285-8110
1500	2018 Q4 Marketing Plan	Sales and Marketing	360-285-8500	11	Granger	Linda	360-285-8520
1500	2018 Q4 Marketing Plan	Sales and Marketing	360-285-8500	10	Numoto	Ken	360-285-8510
1500	2018 Q4 Marketing Plan	Sales and Marketing	360-285-8500	6	Evans	Ken	360-285-8410
1500	2018 Q4 Marketing Plan	Sales and Marketing	360-285-8500	1	Jacobs	Mary	360-285-8110

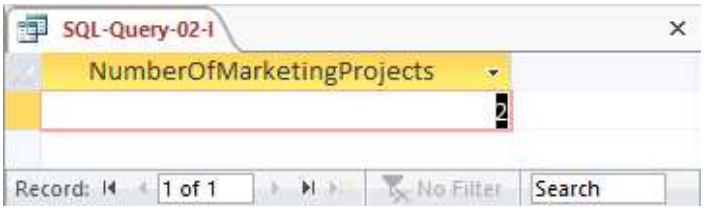
Record: 1 of 8 No Filter Search

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='Sales and 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.*

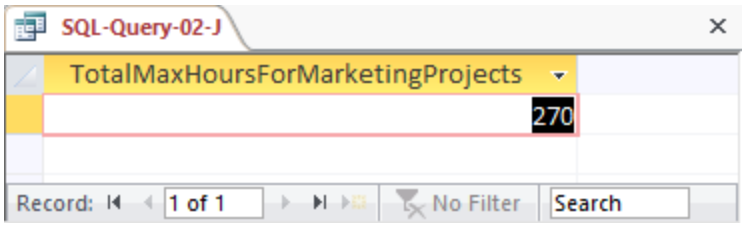
Note the use of the alias **TotalMaxHoursForMarketingProjects**.

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

SELECT    SUM(MaxHours) AS TotalMaxHoursForMarketingProjects
FROM      PROJECT
WHERE     Department='Sales and Marketing';
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

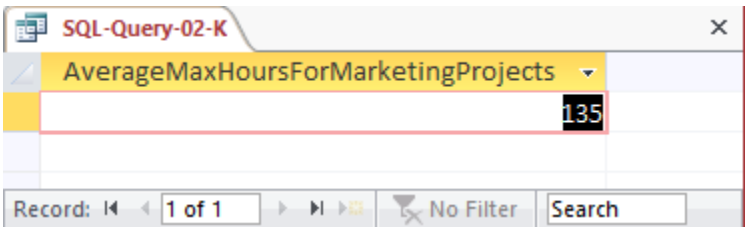


K. *What is the average MaxHours of projects being run by the Marketing department? Be sure to assign an appropriate column name to the computed results.*

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

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

SELECT    AVG(MaxHours) AS AverageMaxHoursForMarketingProjects
FROM      PROJECT
WHERE     Department='Sales and Marketing';
```

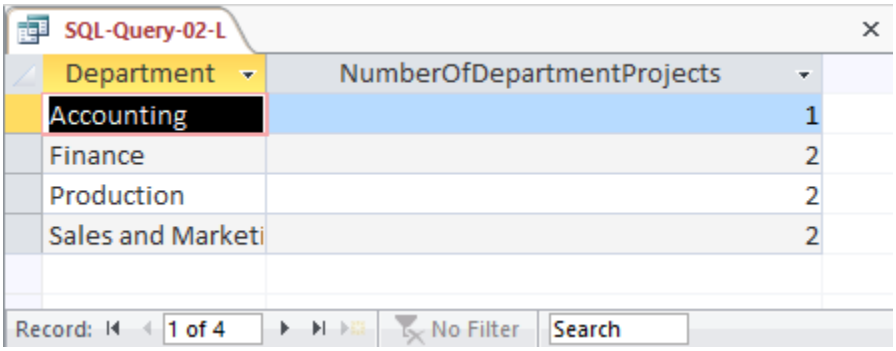


L. *How many projects are being run by each department? Be sure to display each DepartmentName and to assign an appropriate column name to the computed results.*

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

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

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



M. *Who supervises each employee at Wedgewood Pacific? Include the names of employees with no supervisor in the results of the query.*

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
SELECT SUB.EmployeeNumber AS SubNumber, SUB.LastName AS SubLastName,
   SUPER.EmployeeNumber AS SuperNumber, SUPER.LastName AS SuperLastName
FROM EMPLOYEE AS SUB LEFT OUTER JOIN  EMPLOYEE AS SUPER
ON SUB.Supervisor = SUPER.EmployeeNumber;
```

SQL-Query-02-M

SubNumber	SubLastName	SuperNumber	SuperLastNam
1	Jacobs		
3	Bandalone	1	Jacobs
4	Smith	1	Jacobs
12	Nestor	1	Jacobs
6	Evans	1	Jacobs
14	Nguyen	1	Jacobs
16	Smith	1	Jacobs
10	Numoto	1	Jacobs
2	Jackson	1	Jacobs
5	Adams	4	Smith
8	Caruthers	6	Evans
9	Jones	6	Evans
7	Abernathy	6	Evans
11	Granger	10	Numoto
13	Brown	12	Nestor
15	Sleeman	14	Nguyen
17	Jackson	16	Smith
18	Jones	17	Jackson
19	Hayakawa	17	Jackson
20	Stewart	17	Jackson

Record: 1 of 20 No Filter Search

N. 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;
```

Only partial results shown below for this query; there are more rows and columns:

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL-Query-02-N

E.Employe	FirstName	LastName	E.Department	Position	Supervisor	OfficePhone	EmailAddress	A.Projec	A.EmployeeN	HoursWo
1	Mary	Jacobs	Administration	CEO		360-285-8110	Mary.Jacobs@WP.com	1000	1	30
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1000	6	50
10	Ken	Numoto	Sales and Marketir	SM3	1	360-285-8510	Ken.Numoto@WP.com	1000	10	50
16	Mary	Smith	Production	OPS3	1	360-285-8810	Mary.Smith@WP.com	1000	16	75
17	Tom	Jackson	Production	OPS2	16	360-285-8820	Tom.Jackson@WP.com	1000	17	75
1	Mary	Jacobs	Administration	CEO		360-285-8110	Mary.Jacobs@WP.com	1100	1	30
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1100	6	75
10	Ken	Numoto	Sales and Marketir	SM3	1	360-285-8510	Ken.Numoto@WP.com	1100	10	55
11	Linda	Granger	Sales and Marketir	SM2	10	360-285-8520	Linda.Granger@WP.com	1100	11	55
3	Richard	Bandalone	Legal	Attorney	1	360-285-8210	Richard.Bandalone@WP.com	1200	3	20
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1200	6	40
7	Mary	Abernathy	Finance	FA3	6	360-285-8420	Mary.Abernathy@WP.com	1200	7	45
8	Tom	Caruthers	Accounting	FA2	6	360-285-8430	Tom.Caruthers@WP.com	1200	8	45
3	Richard	Bandalone	Legal	Attorney	1	360-285-8210	Richard.Bandalone@WP.com	1300	3	25
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1300	6	40
8	Tom	Caruthers	Accounting	FA2	6	360-285-8430	Tom.Caruthers@WP.com	1300	8	50
9	Heather	Jones	Accounting	FA2	6	360-285-8440	Heather.Jones@WP.com	1300	9	50
1	Mary	Jacobs	Administration	CEO		360-285-8110	Mary.Jacobs@WP.com	1400	1	30
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1400	6	50
10	Ken	Numoto	Sales and Marketir	SM3	1	360-285-8510	Ken.Numoto@WP.com	1400	10	50
16	Mary	Smith	Production	OPS3	1	360-285-8810	Mary.Smith@WP.com	1400	16	75
17	Tom	Jackson	Production	OPS2	16	360-285-8820	Tom.Jackson@WP.com	1400	17	75
1	Mary	Jacobs	Administration	CEO		360-285-8110	Mary.Jacobs@WP.com	1500	1	30
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1500	6	75

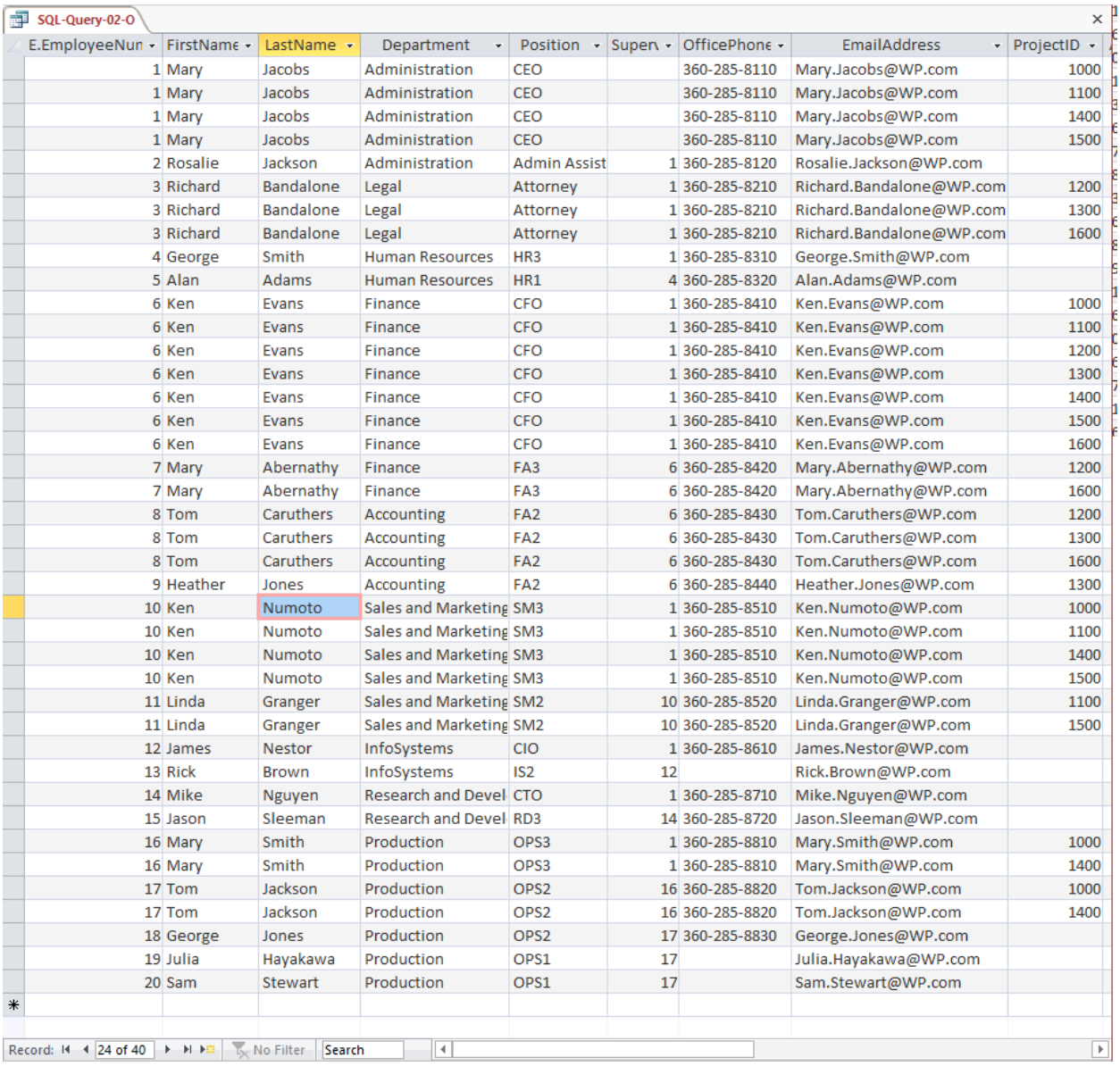
O. *Write an SQL statement to join EMPLOYEE and ASSIGNMENT and include all rows of EMPLOYEE in your answer, regardless of whether they have an ASSIGNMENT. Run this statement.*

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

Note that there are more columns in the result than would fit in the image:

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



SQL-Query-02-O

E.EmployeeNun	FirstName	LastName	Department	Position	Superv	OfficePhone	EmailAddress	ProjectID
1	Mary	Jacobs	Administration	CEO		360-285-8110	Mary.Jacobs@WP.com	1000
1	Mary	Jacobs	Administration	CEO		360-285-8110	Mary.Jacobs@WP.com	1100
1	Mary	Jacobs	Administration	CEO		360-285-8110	Mary.Jacobs@WP.com	1400
1	Mary	Jacobs	Administration	CEO		360-285-8110	Mary.Jacobs@WP.com	1500
2	Rosalie	Jackson	Administration	Admin Assist	1	360-285-8120	Rosalie.Jackson@WP.com	
3	Richard	Bandalone	Legal	Attorney	1	360-285-8210	Richard.Bandalone@WP.com	1200
3	Richard	Bandalone	Legal	Attorney	1	360-285-8210	Richard.Bandalone@WP.com	1300
3	Richard	Bandalone	Legal	Attorney	1	360-285-8210	Richard.Bandalone@WP.com	1600
4	George	Smith	Human Resources	HR3	1	360-285-8310	George.Smith@WP.com	
5	Alan	Adams	Human Resources	HR1	4	360-285-8320	Alan.Adams@WP.com	
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1000
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1100
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1200
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1300
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1400
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1500
6	Ken	Evans	Finance	CFO	1	360-285-8410	Ken.Evans@WP.com	1600
7	Mary	Abernathy	Finance	FA3	6	360-285-8420	Mary.Abernathy@WP.com	1200
7	Mary	Abernathy	Finance	FA3	6	360-285-8420	Mary.Abernathy@WP.com	1600
8	Tom	Caruthers	Accounting	FA2	6	360-285-8430	Tom.Caruthers@WP.com	1200
8	Tom	Caruthers	Accounting	FA2	6	360-285-8430	Tom.Caruthers@WP.com	1300
8	Tom	Caruthers	Accounting	FA2	6	360-285-8430	Tom.Caruthers@WP.com	1600
9	Heather	Jones	Accounting	FA2	6	360-285-8440	Heather.Jones@WP.com	1300
10	Ken	Numoto	Sales and Marketing	SM3	1	360-285-8510	Ken.Numoto@WP.com	1000
10	Ken	Numoto	Sales and Marketing	SM3	1	360-285-8510	Ken.Numoto@WP.com	1100
10	Ken	Numoto	Sales and Marketing	SM3	1	360-285-8510	Ken.Numoto@WP.com	1400
10	Ken	Numoto	Sales and Marketing	SM3	1	360-285-8510	Ken.Numoto@WP.com	1500
11	Linda	Granger	Sales and Marketing	SM2	10	360-285-8520	Linda.Granger@WP.com	1100
11	Linda	Granger	Sales and Marketing	SM2	10	360-285-8520	Linda.Granger@WP.com	1500
12	James	Nestor	InfoSystems	CIO	1	360-285-8610	James.Nestor@WP.com	
13	Rick	Brown	InfoSystems	IS2	12		Rick.Brown@WP.com	
14	Mike	Nguyen	Research and Devel	CTO	1	360-285-8710	Mike.Nguyen@WP.com	
15	Jason	Sleeman	Research and Devel	RD3	14	360-285-8720	Jason.Sleeman@WP.com	
16	Mary	Smith	Production	OPS3	1	360-285-8810	Mary.Smith@WP.com	1000
16	Mary	Smith	Production	OPS3	1	360-285-8810	Mary.Smith@WP.com	1400
17	Tom	Jackson	Production	OPS2	16	360-285-8820	Tom.Jackson@WP.com	1000
17	Tom	Jackson	Production	OPS2	16	360-285-8820	Tom.Jackson@WP.com	1400
18	George	Jones	Production	OPS2	17	360-285-8830	George.Jones@WP.com	
19	Julia	Hayakawa	Production	OPS1	17		Julia.Hayakawa@WP.com	
20	Sam	Stewart	Production	OPS1	17		Sam.Stewart@WP.com	

Record: 24 of 40 No Filter Search

2.71 Using Microsoft Access QBE, create and run new queries to answer the questions in exercise 2.70. 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. HINT: In questions G and H, the default approach of accepting all joins will not work, and you may have to delete some joins from the initial QBE query.

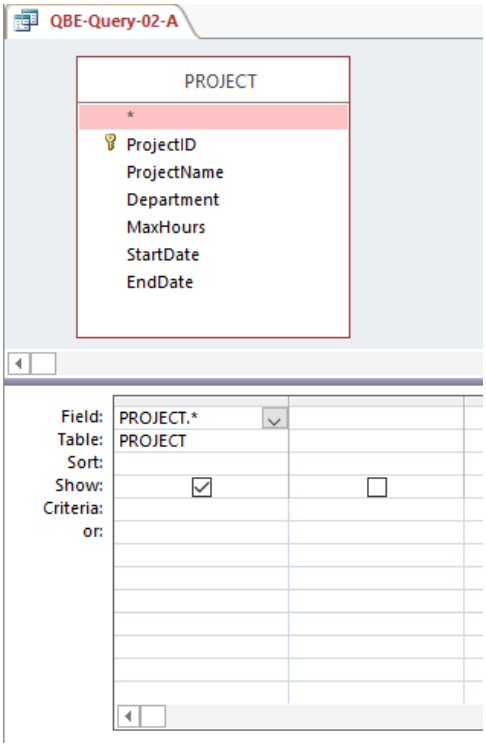
Solutions to Project Questions 2.61 – 2.70 are contained in the Microsoft Access database *DBP-e15-IM-CH02-WP.accdb* which is available on the text's Web site (www.pearsonhighered.com/kroenke).

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.

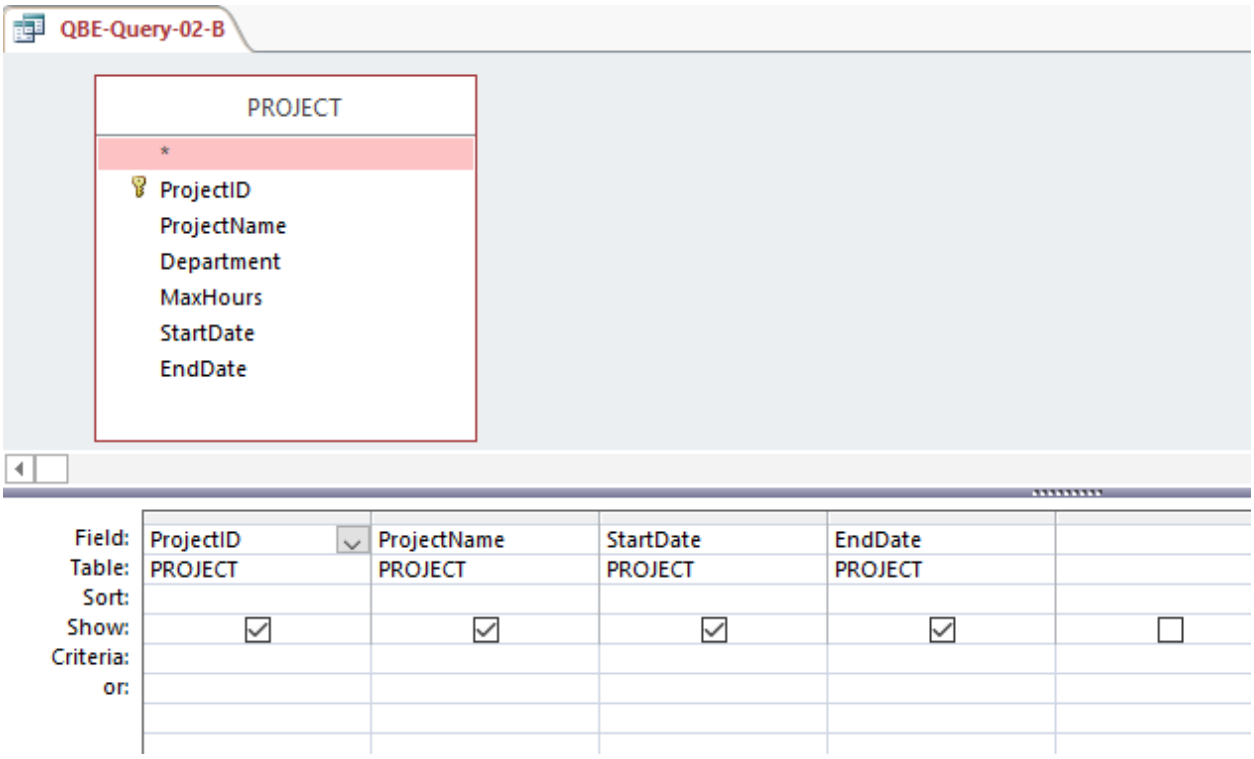
Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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



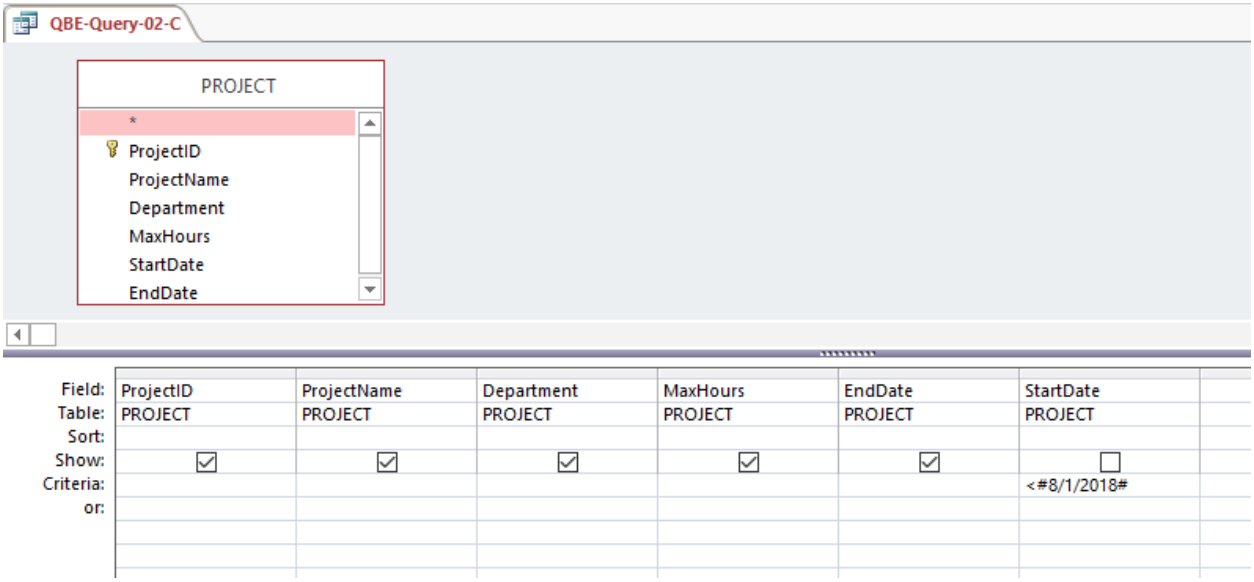
B. What are the ProjectID, ProjectName, StartDate, and EndDate values of projects in the PROJECT table?



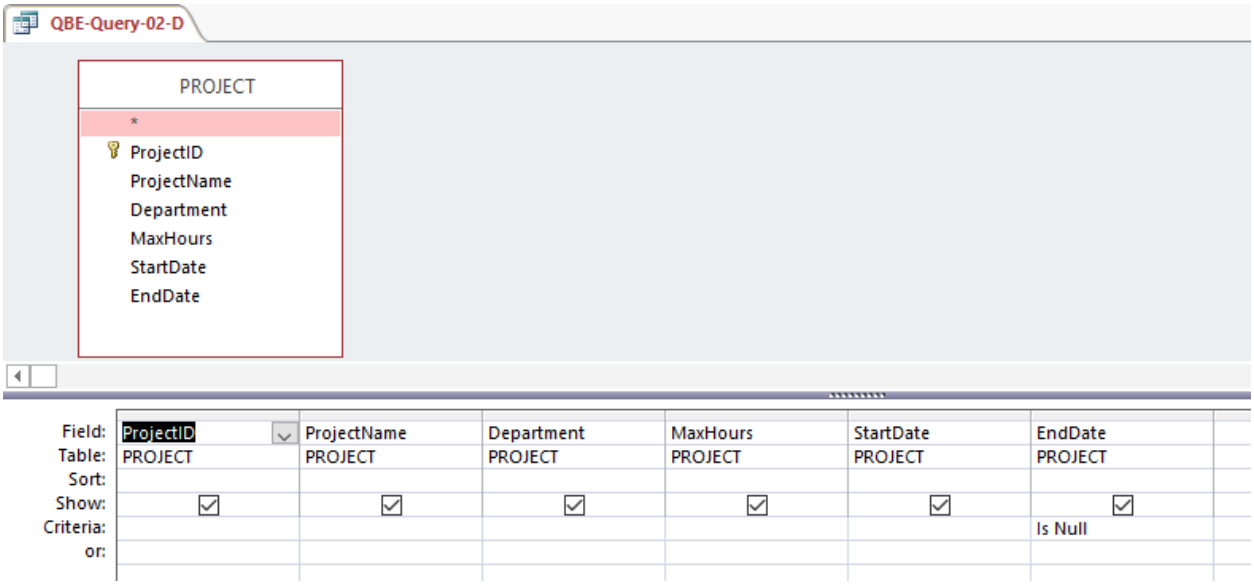
Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

C. What projects in the PROJECT table started before August 1, 2018? Show all the information for each project.



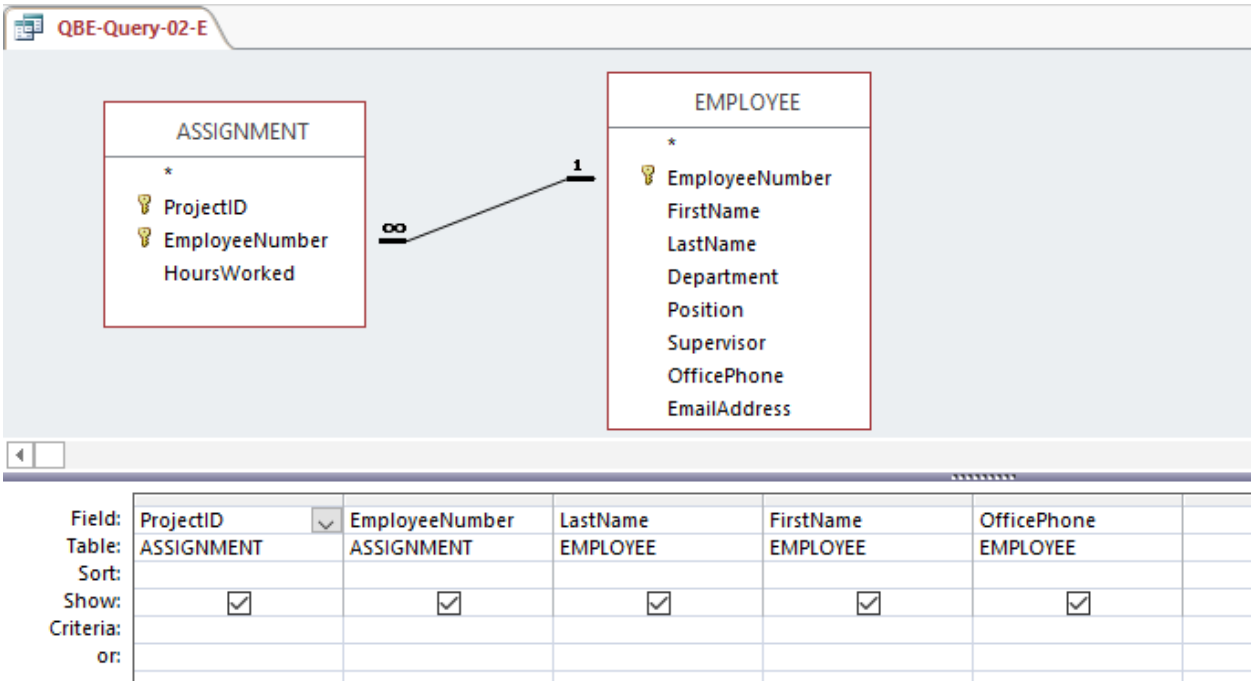
D. What projects in the PROJECT table have not been completed? Show all the information for each project.



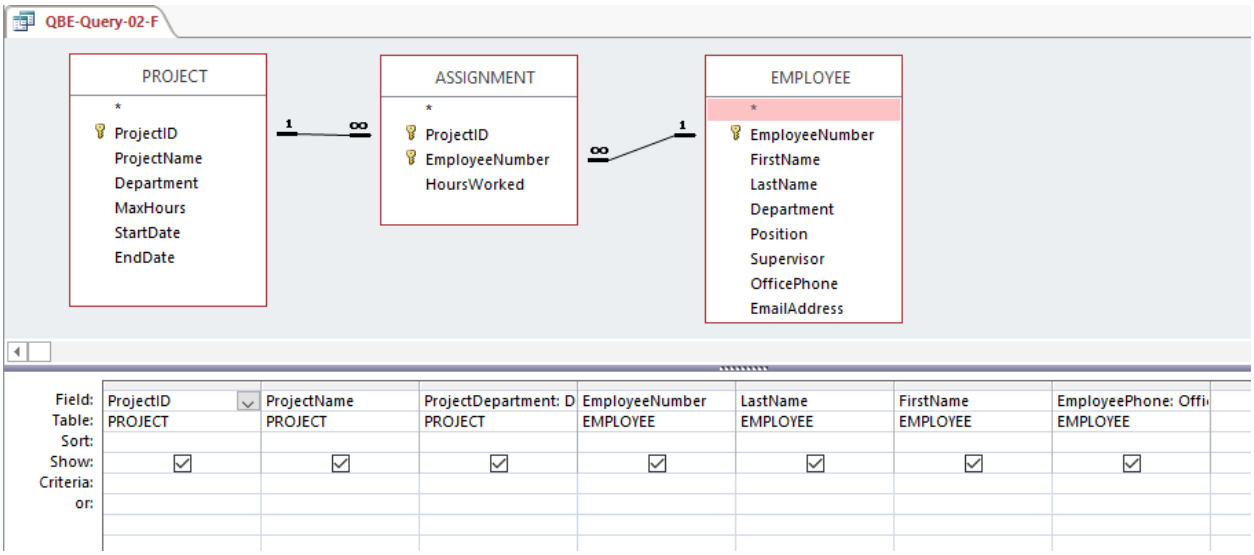
E. Who are the employees assigned to each project? Show ProjectID, Employee-Number, LastName, FirstName, and OfficePhone.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



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



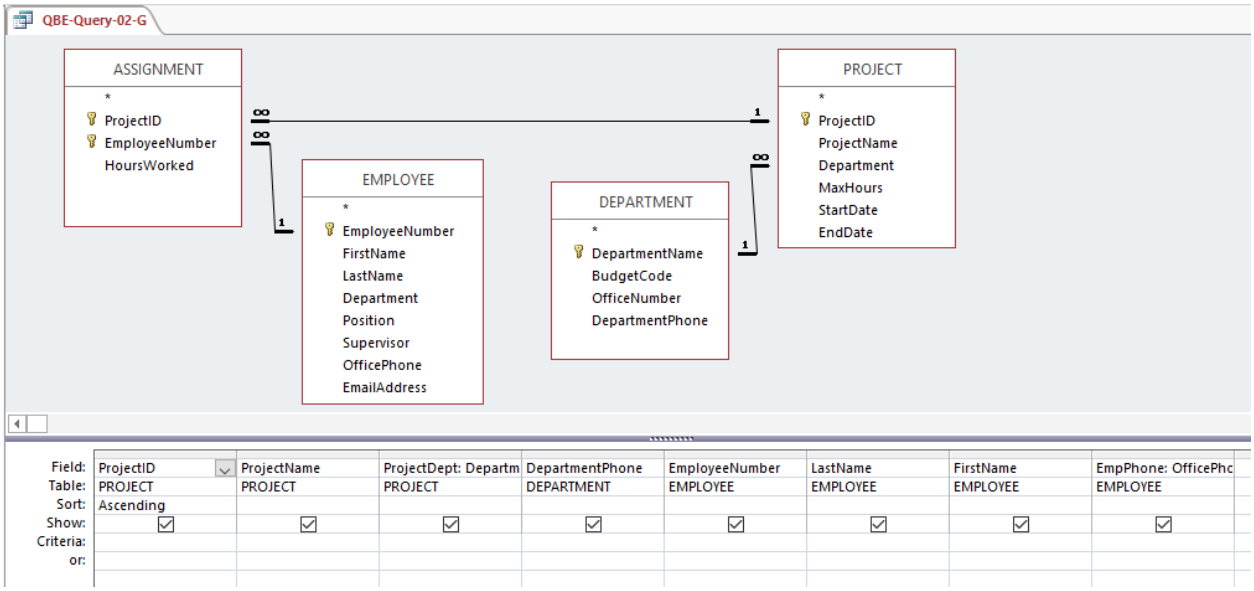
G. Who are the employees assigned to each project? Show ProjectID, ProjectName, Department, and Department Phone. Show EmployeeNumber, LastName, FirstName, and OfficePhone. Sort by ProjectID in ascending order.

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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

query will only return assignments in which an EMPLOYEE is assigned to a PROJECT that is in the EMPLOYEE's DEPARTMENT.

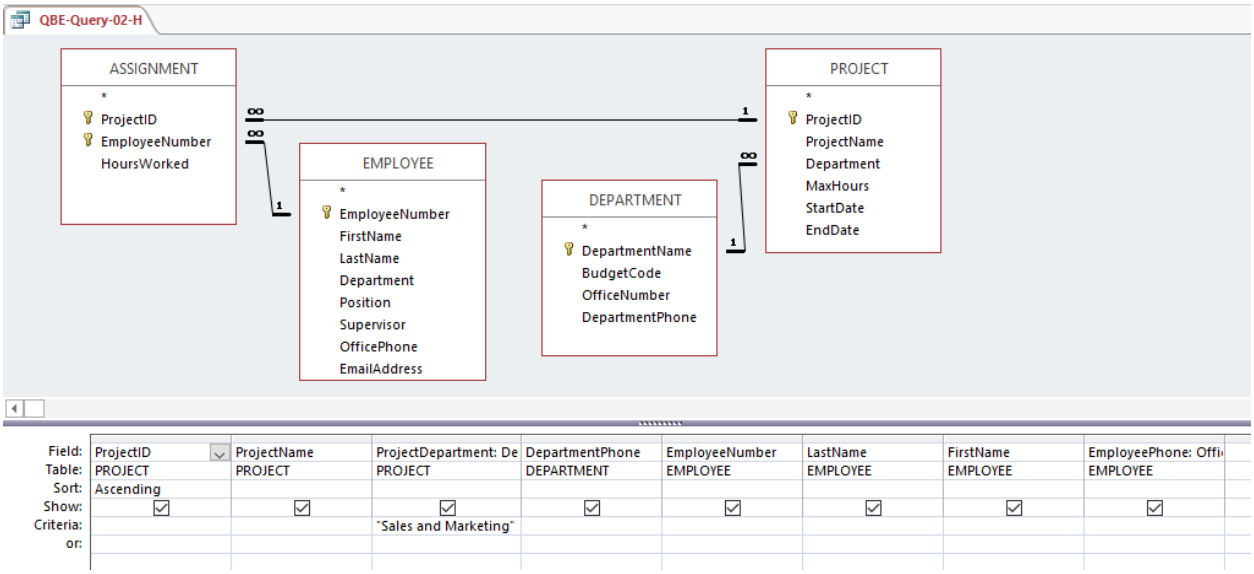


H. Who are the employees assigned to projects run by the marketing department? Show ProjectID, ProjectName, Department, and DepartmentPhone. Show EmployeeNumber, LastName, FirstName, and OfficePhone. Sort by ProjectID in ascending order.

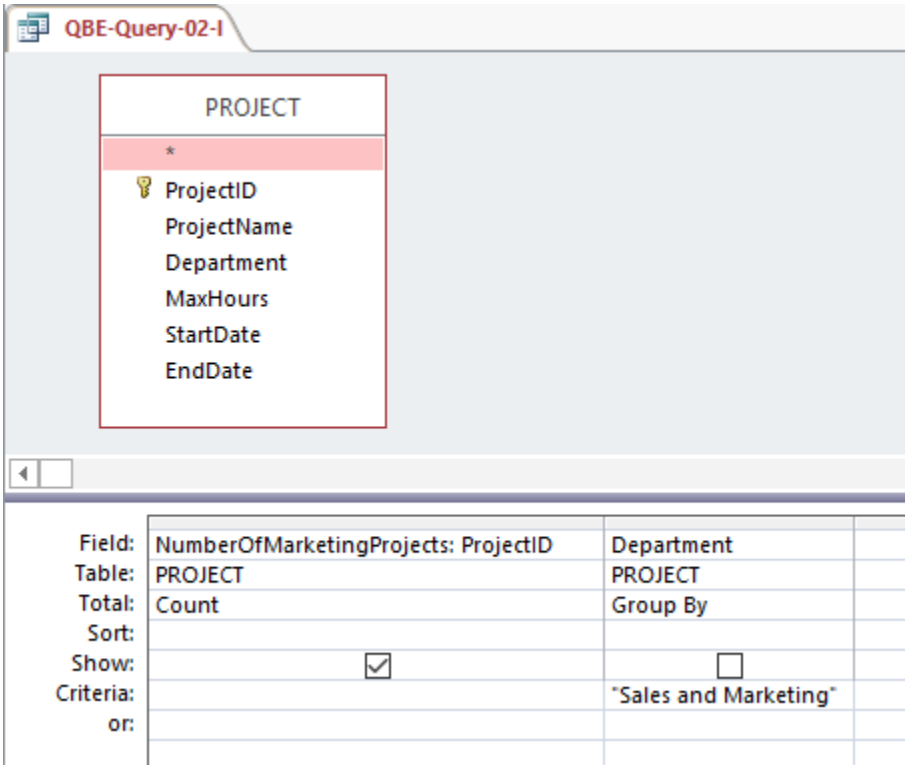
This question is identical to question G except for the restriction to marketing department projects. And, again, this question is more complicated than it seems, in that the default approach of "accepting" all the joins in the QBE query yields an incorrect result. Without deleting the join from EMPLOYEE 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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



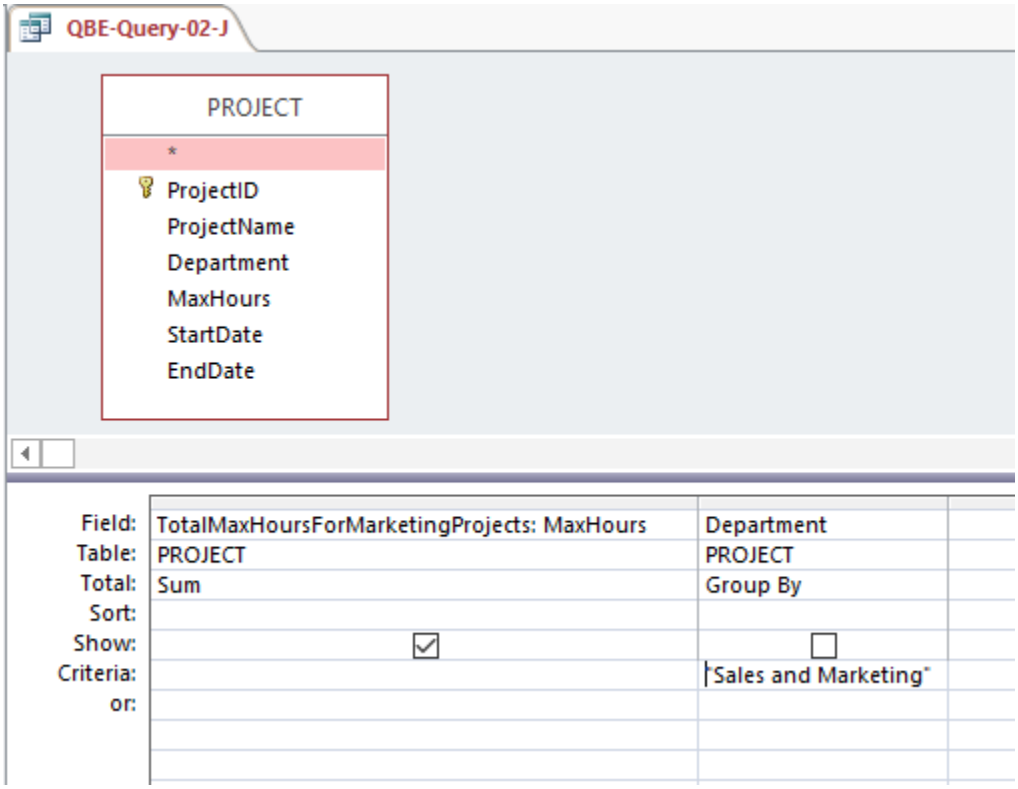
I. How many projects are being run by the Marketing department? Be sure to assign an appropriate column name to the computed results.



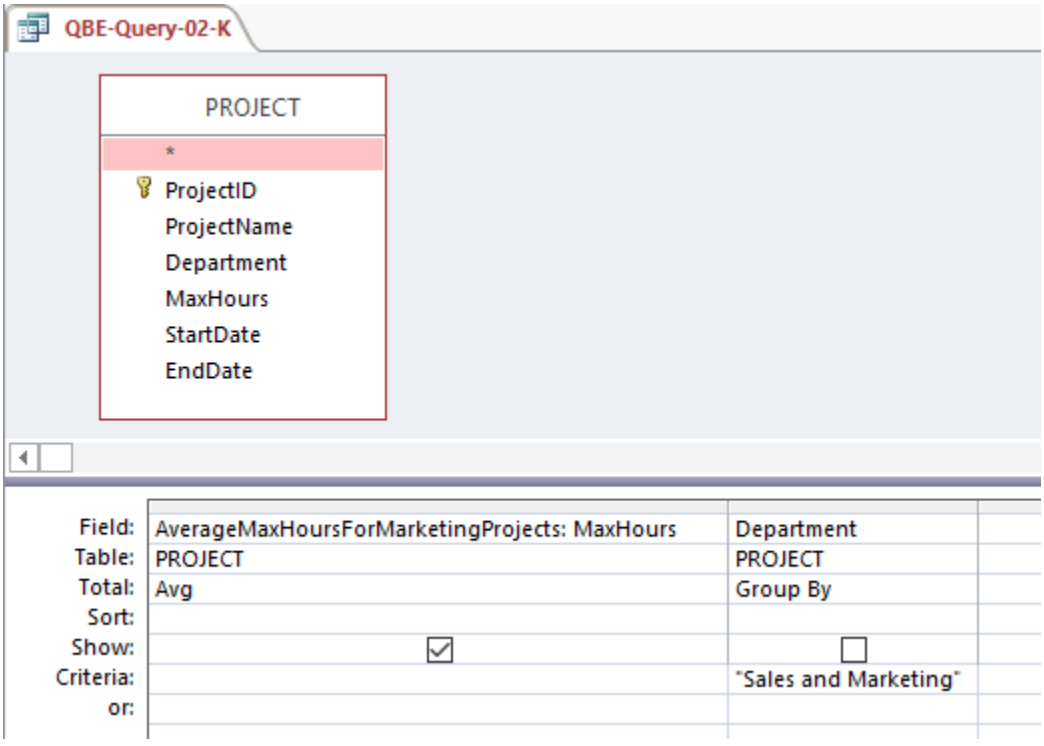
J. What is the total MaxHours of projects being run by the Marketing department? Be sure to assign an appropriate column name to the computed results.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



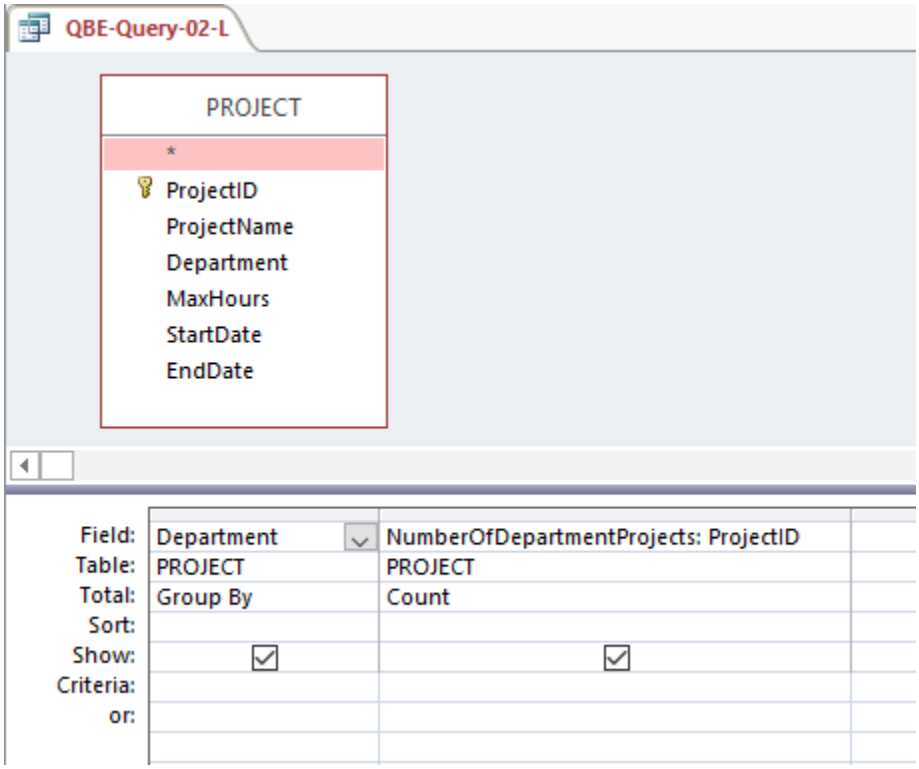
K. What is the average MaxHours of projects being run by the Marketing department? Be sure to assign an appropriate column name to the computed results.



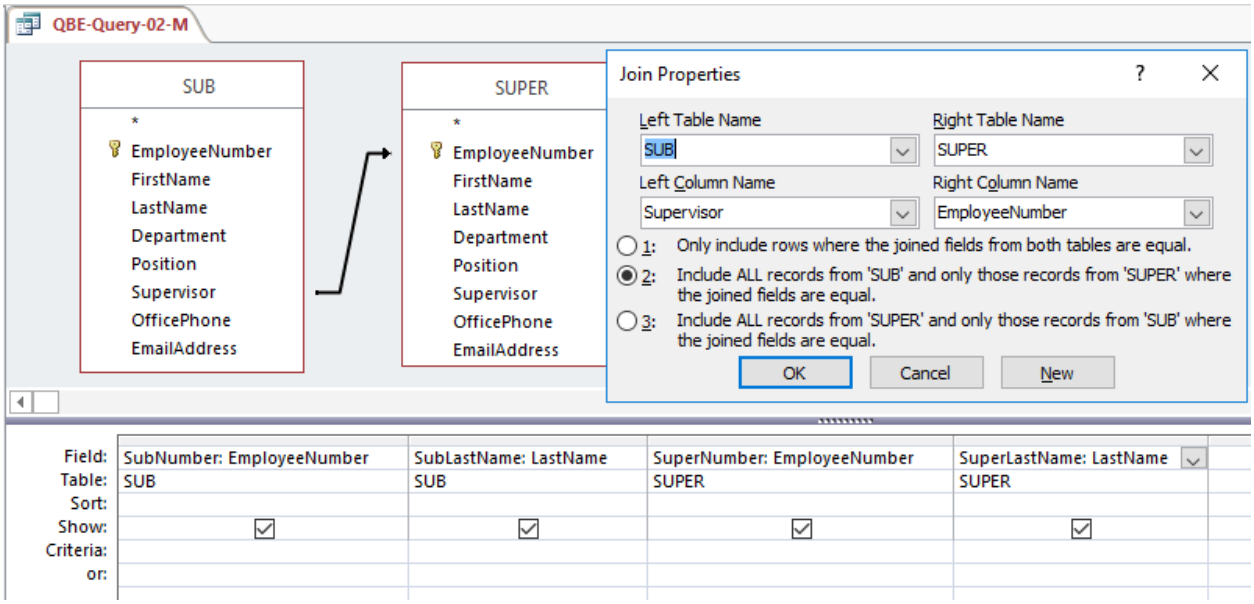
Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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.*



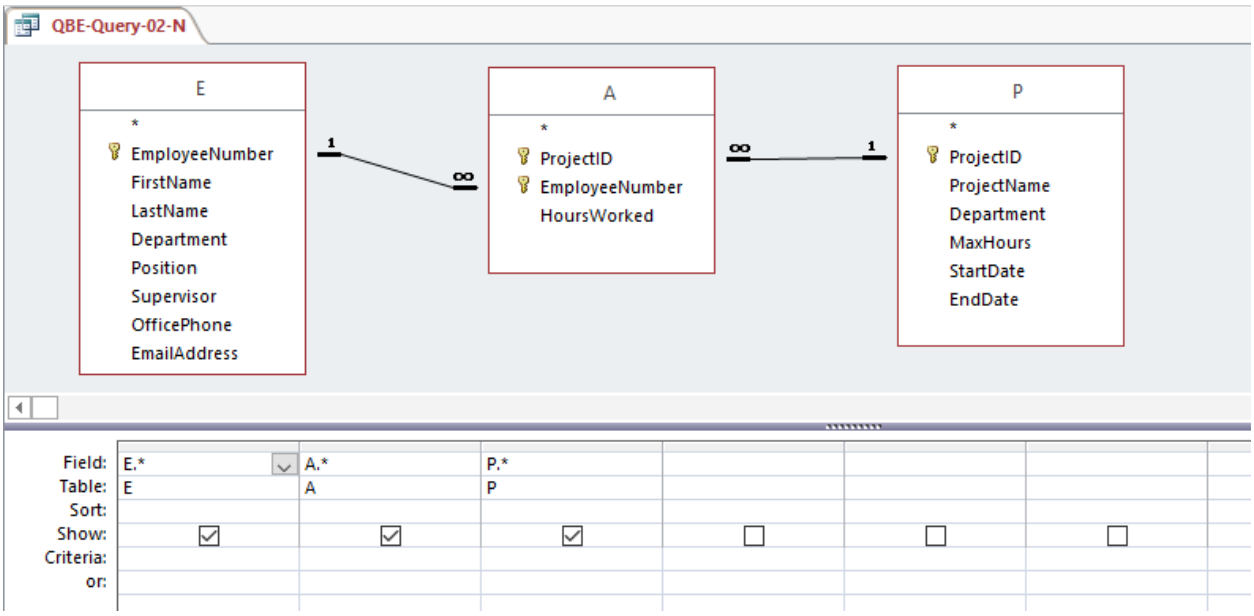
M. *Who supervises each employee at Wedgwood Pacific? Include the names of employees with no supervisor in the results of the query.*



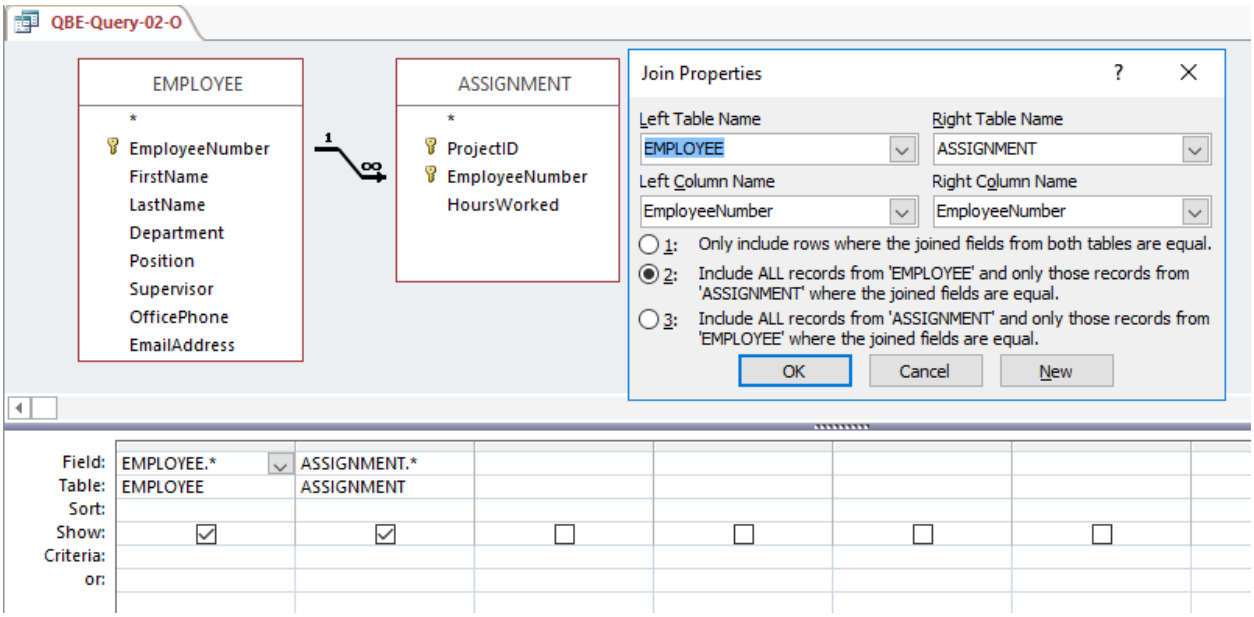
N. *Write an SQL statement to join EMPLOYEE, ASSIGNMENT, and PROJECT using the JOIN ON syntax. Run this statement.*

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



O. *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.*



Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

❖ 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 as follows:

CUSTOMER (CustomerID, FirstName, LastName, Phone, EmailAddress, *ReferredBy*)

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

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

Where

ReferredBy in CUSTOMER must exist in CustomerID in CUSTOMER

CustomerID in INVOICE must exist in CustomerID in CUSTOMER

InvoiceNumber in INVOICE_ITEM must exist in InvoiceNumber in INVOICE

In this database schema, the primary keys are underlined and the foreign keys are shown in italics. Note that CUSTOMER contains a recursive relationship between ReferredBy and CustomerID, where ReferredBy contains the CustomerID value of the existing customer who referred the new customer to Marcia's Dry Cleaning. The database that Marcia has created is named MDC, and the three tables in the MDC database schema are shown in Figure 2-47.

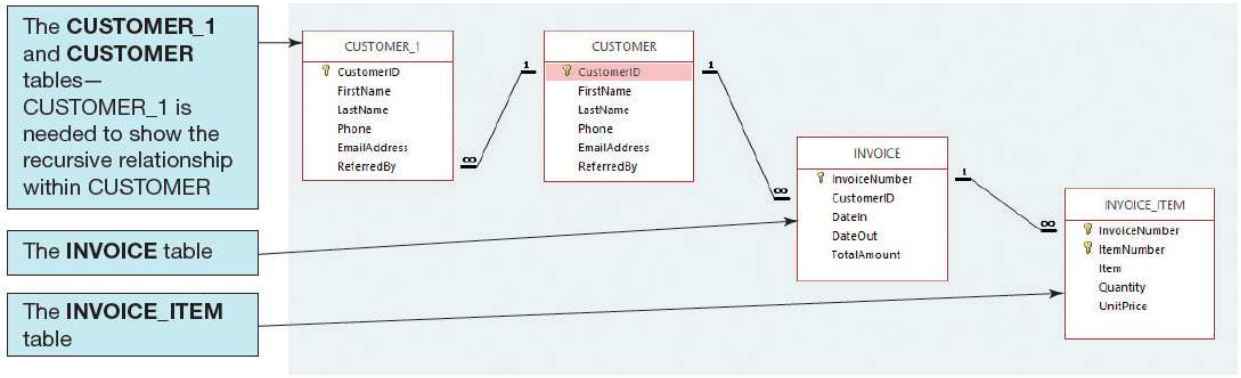


FIGURE 2-47 – The MDC Database

The column characteristics for the tables are shown in Figures 2-48, 2-49, and 2-50. The relationship between CUSTOMER and INVOICE should enforce referential integrity, but not cascade updates or deletions, while the relationship between INVOICE and INVOICE_ITEM

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

should enforce referential integrity and cascade both updates and deletions. The data for these tables are shown in Figures 2-51, 2-52, and 2-53.

You will need to create and setup a database named MDC_CH02 for use with these case questions. A Microsoft Access 2016 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 2016 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 2017, see online Chapter 10A.
- For Oracle Database 12c Release 2 or Oracle Express Edition 11g Release 2, see online Chapter 10B.
- For MySQL 5.7 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	
EmailAddress	Character (100)	No	No	Use Varchar
ReferredBy	Integer	Foreign Key	No	REF: CustomerID

Figure 2-48 - Column Characteristics for the MDC Database CUSTOMER Table

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

INVOICE

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

Figure 2-49 - Column Characteristics for the MDC Database 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 2-50 - Column Characteristics for the MDC Database INVOICE_ITEM Table

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

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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

InvoiceNumber	CustomerNumber	DateIn	DateOut	TotalAmount
2018001	1	04-Oct-18	06-Oct-18	$158.50
2018002	2	04-Oct-18	06-Oct-18	$25.00
2018003	1	06-Oct-18	08-Oct-18	$49.00
2018004	4	06-Oct-18	08-Oct-18	$17.50
2018005	6	07-Oct-18	11-Oct-18	$12.00
2018006	3	11-Oct-18	13-Oct-18	$152.50
2018007	3	11-Oct-18	13-Oct-18	$7.00
2018008	7	12-Oct-18	14-Oct-18	$140.50
2018009	5	12-Oct-18	14-Oct-18	$27.00

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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

InvoiceNumber	ItemNumber	Item	Quantity	UnitPrice
2018001	1	Blouse	2	$3.50
2018001	2	Dress Shirt	5	$2.50
2018001	3	Formal Gown	2	$10.00
2018001	4	Slacks-Mens	10	$5.00
2018001	5	Slacks-Womens	10	$6.00
2018001	6	Suit-Mens	1	$9.00
2018002	1	Dress Shirt	10	$2.50
2018003	1	Slacks-Mens	5	$5.00
2018003	2	Slacks-Womens	4	$6.00
2018004	1	Dress Shirt	7	$2.50
2018005	1	Blouse	2	$3.50
2018005	2	Dress Shirt	2	$2.50
2018006	1	Blouse	5	$3.50
2018006	2	Dress Shirt	10	$2.50
2018006	3	Slacks-Mens	10	$5.00
2018006	4	Slacks-Womens	10	$6.00
2018007	1	Blouse	2	$3.50
2018008	1	Blouse	3	$3.50
2018008	2	Dress Shirt	12	$2.50
2018008	3	Slacks-Mens	8	$5.00
2018008	4	Slacks-Womens	10	$6.00
2018009	1	Suit-Mens	3	$9.00

Figure 2-53 - 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-e15-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 named Betsy Miller.

SQL-Query-MDC-A-CUSTOMER

CustomerID	FirstName	LastName	Phone	EmailAddress	ReferredBy
1	Nikki	Kaccaton	723-543-1233	Nikki.Kaccaton@somewhere.com	
2	Brenda	Catnazaro	723-543-2344	Brenda.Catnazaro@somewhere.com	1
3	Bruce	LeCat	723-543-3455	Bruce.LeCat@somewhere.com	2
4	Betsy	Miller	723-654-3211	Betsy.Miller@somewhere.com	3
5	George	Miller	723-654-4322	George.Miller@somewhere.com	4
6	Kathy	Miller	723-514-9877	Kathy.Miller@somewhere.com	2
7	Betsy	Miller	723-514-8766	Betsy.Miller@elsewhere.com	1
*					0

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
2018001	1	10/4/2018	10/6/2018	$158.50
2018002	2	10/4/2018	10/6/2018	$25.00
2018003	1	10/6/2018	10/8/2018	$49.00
2018004	4	10/6/2018	10/8/2018	$17.50
2018005	6	10/7/2018	10/11/2018	$12.00
2018006	3	10/11/2018	10/13/2018	$152.50
2018007	3	10/11/2018	10/13/2018	$7.00
2018008	7	10/12/2018	10/14/2018	$140.50
2018009	5	10/12/2018	10/14/2018	$27.00
*				

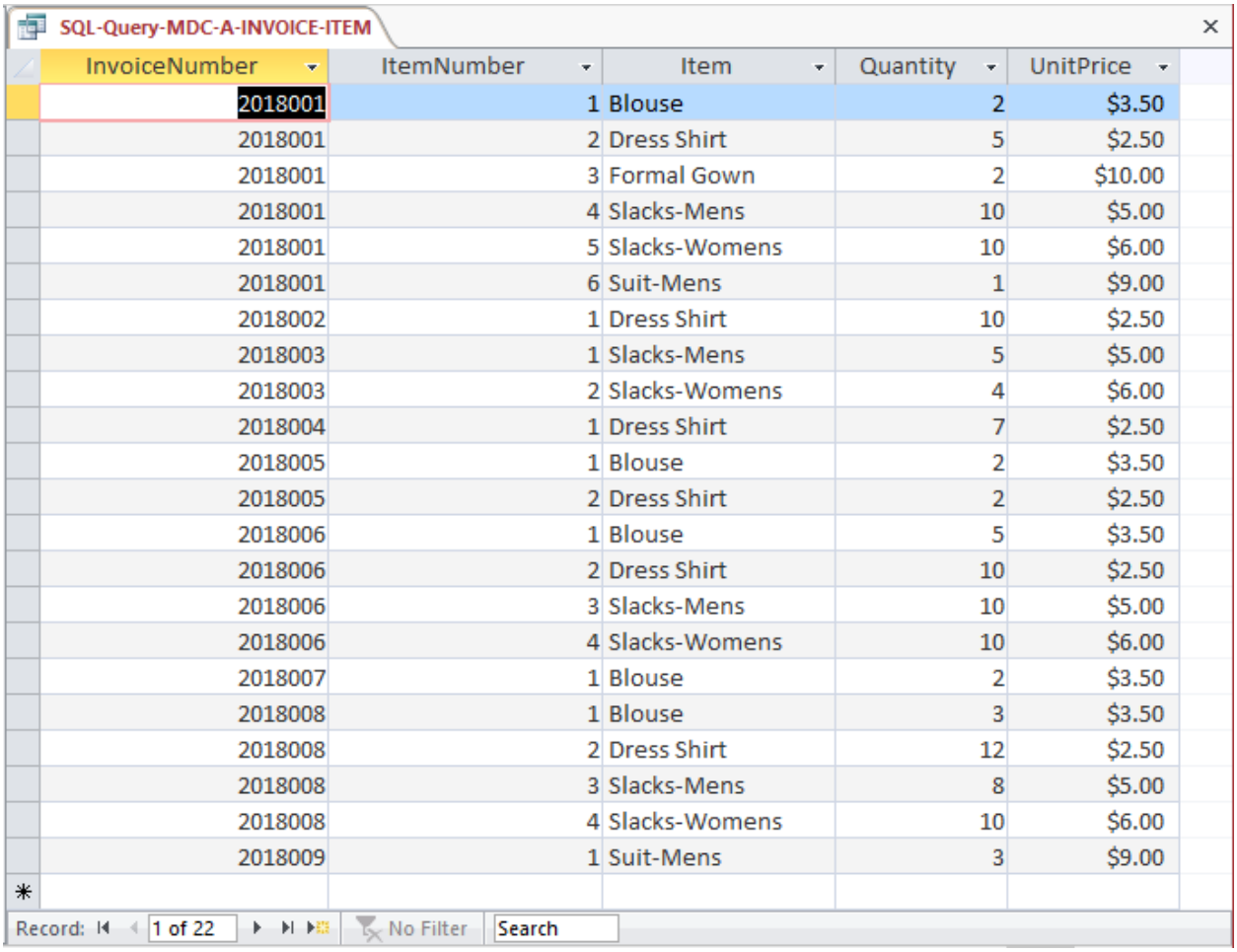
Record: 1 of 9 No Filter Search

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

SELECT    *
FROM      INVOICE_ITEM;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



SQL-Query-MDC-A-INVOICE-ITEM

InvoiceNumber	ItemNumber	Item	Quantity	UnitPrice
2018001	1	Blouse	2	$3.50
2018001	2	Dress Shirt	5	$2.50
2018001	3	Formal Gown	2	$10.00
2018001	4	Slacks-Mens	10	$5.00
2018001	5	Slacks-Womens	10	$6.00
2018001	6	Suit-Mens	1	$9.00
2018002	1	Dress Shirt	10	$2.50
2018003	1	Slacks-Mens	5	$5.00
2018003	2	Slacks-Womens	4	$6.00
2018004	1	Dress Shirt	7	$2.50
2018005	1	Blouse	2	$3.50
2018005	2	Dress Shirt	2	$2.50
2018006	1	Blouse	5	$3.50
2018006	2	Dress Shirt	10	$2.50
2018006	3	Slacks-Mens	10	$5.00
2018006	4	Slacks-Womens	10	$6.00
2018007	1	Blouse	2	$3.50
2018008	1	Blouse	3	$3.50
2018008	2	Dress Shirt	12	$2.50
2018008	3	Slacks-Mens	8	$5.00
2018008	4	Slacks-Womens	10	$6.00
2018009	1	Suit-Mens	3	$9.00

Record: 1 of 22 No Filter Search

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

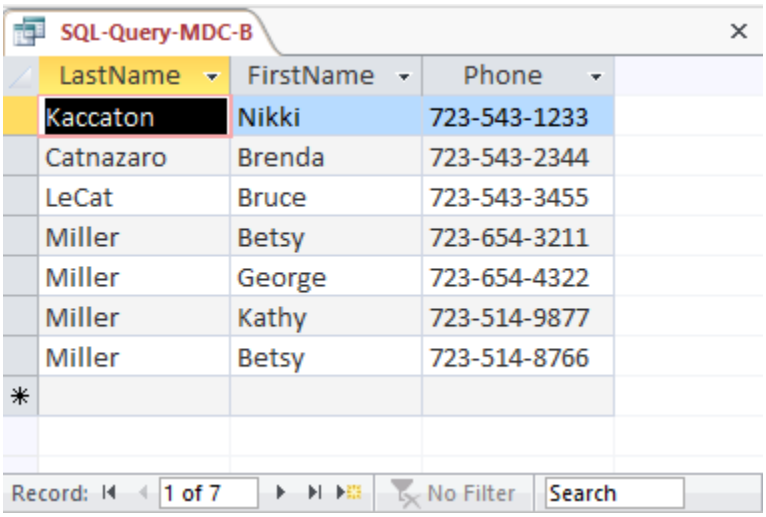
Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

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

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


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

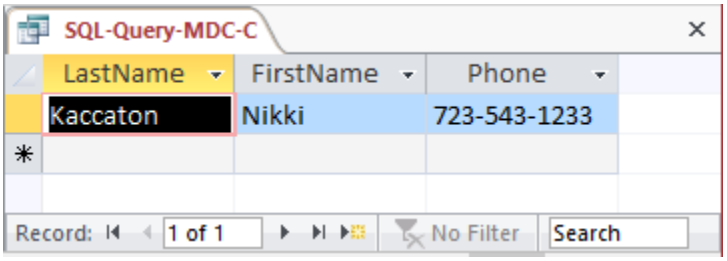


C. List the LastName, FirstName, and Phone for all customers with a FirstName of "Nikki".

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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';
```



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-e15-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.CustomerID;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL-Query-MDC-D

LastName	FirstName	Phone	DateIn	DateOut
Kaccaton	Nikki	723-543-1233	10/4/2018	10/6/2018
LeCat	Bruce	723-543-3455	10/11/2018	10/13/2018
Miller	Betsy	723-514-8766	10/12/2018	10/14/2018

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-e15-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*';
```

SQL-Query-MDC-E-Access

LastName	FirstName	Phone
Catnazaro	Brenda	723-543-2344
LeCat	Bruce	723-543-3455
Miller	Betsy	723-654-3211
Miller	Betsy	723-514-8766

Record: 1 of 4 No Filter Search

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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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-e15-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:

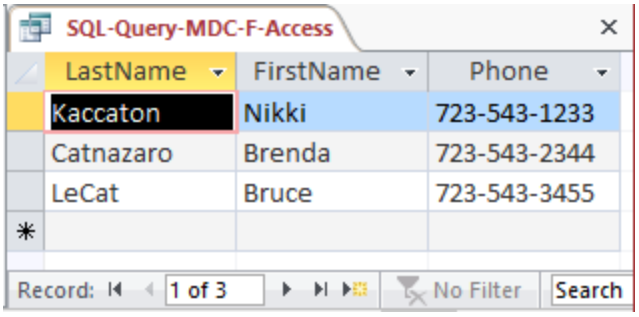
```
/* *** SQL-Query-MDC-F *** */

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*';
```



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.*

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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	Betsy	723-654-3211
Miller	George	723-654-4322
Miller	Kathy	723-514-9877
Miller	Betsy	723-514-8766

Record: 1 of 7 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-e15-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).

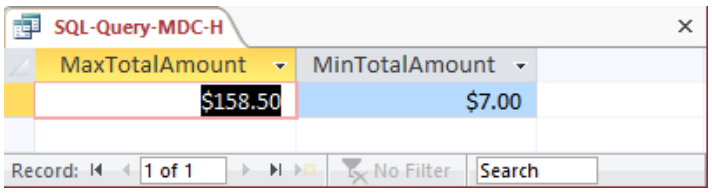
```
/* *** SQL-Query-MDC-H *** */

SELECT   MAX (TotalAmt) AS MaxTotalAmount,
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
        MIN (TotalAmt) AS MinTotalAmount
FROM    INVOICE;
```

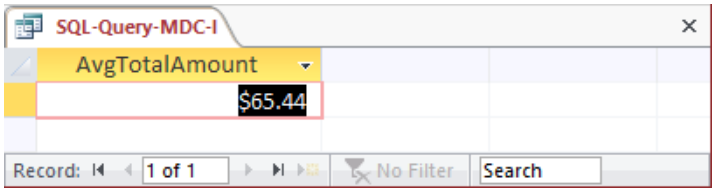


I. Determine the average TotalAmount.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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-I *** */

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

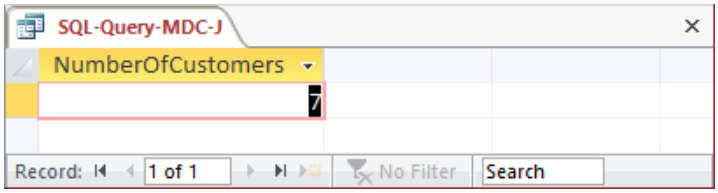


J. Count the number of customers.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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-e15-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL

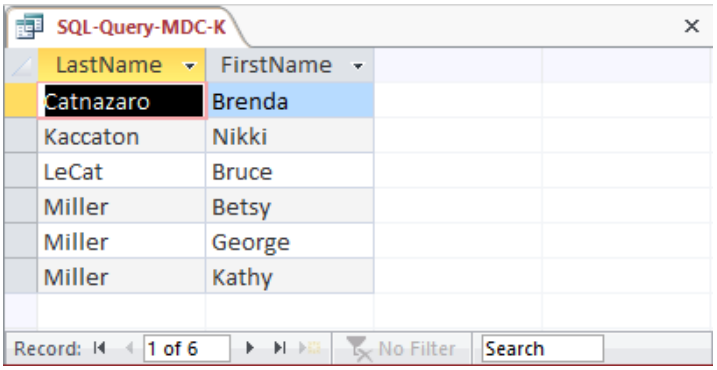
Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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;
```

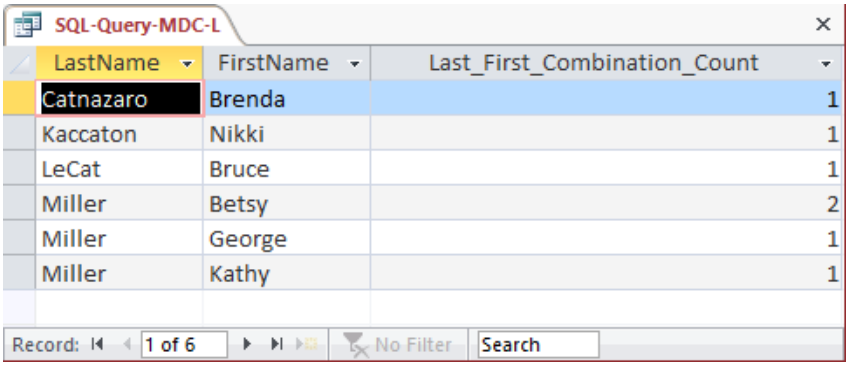


L. *Count the number of customers having each combination of LastName and FirstName.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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;
```



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.*

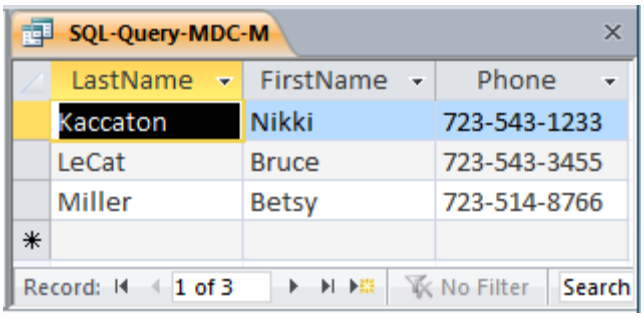
Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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 CustomerID
         FROM INVOICE
         WHERE TotalAmount > 100)
ORDER BY LastName, FirstName DESC;
```

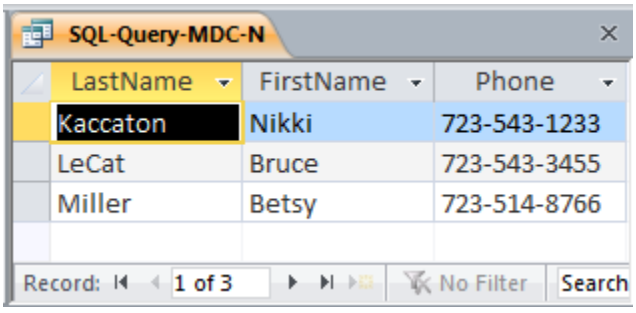


N. *Show the LastName, FirstName and Phone of all customers who have had an order with TotalAmount greater than $100.00. Use a join, but do not use JOIN ON syntax. Present results sorted by LastName in ascending order and then FirstName in descending order.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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.CustomerID
   AND   TotalAmount > 100
ORDER BY LastName, FirstName DESC;
```



Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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).

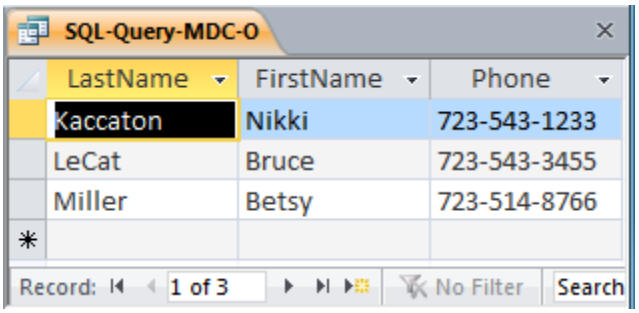
```
/* *** SQL-Query-MDC-O *** */

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

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.CustomerID
WHERE    INVOICE.TotalAmount>100
ORDER BY LastName, FirstName DESC;
```



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.

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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).

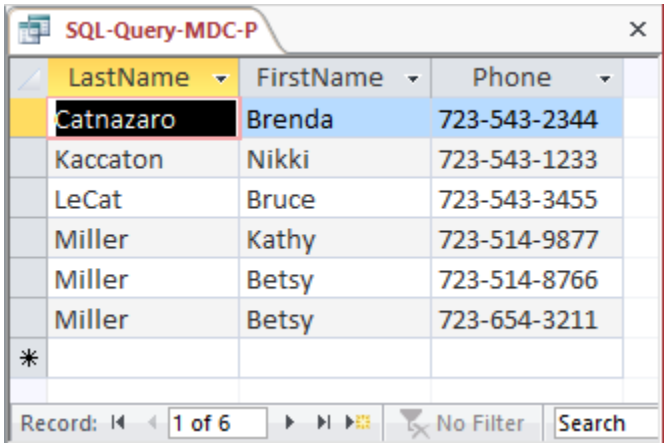
Note the solution below uses 2 subqueries; other correct solutions are possible that use one subquery and a join (the question does not specify that two subqueries must be used).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

SELECT   LastName, FirstName, Phone
FROM     CUSTOMER
WHERE    CustomerID IN
         (SELECT  CustomerID
         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 results sorted by LastName in ascending order and then FirstName in descending order.*

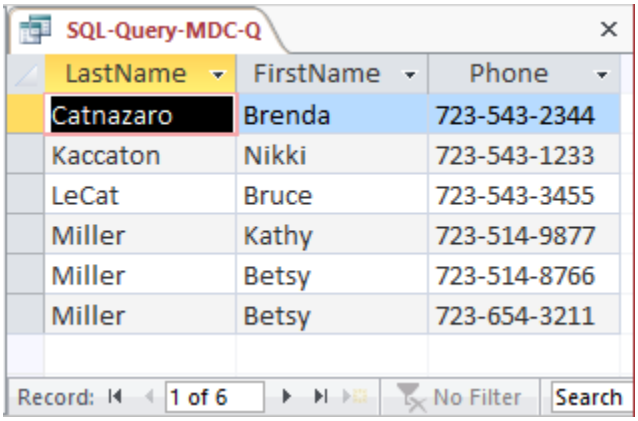
Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e14-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL Server, and MySQL, which are all available at the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

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

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


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



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.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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.CustomerID)
         JOIN INVOICE_ITEM
               ON INVOICE.InvoiceNumber = INVOICE_ITEM.InvoiceNumber
WHERE    INVOICE_ITEM.Item='Dress Shirt'
ORDER BY LastName, FirstName DESC;
```

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.CustomerID)
          INNER JOIN INVOICE_ITEM
               ON INVOICE.InvoiceNumber = INVOICE_ITEM.InvoiceNumber
WHERE    INVOICE_ITEM.Item ='Dress Shirt'
ORDER BY LastName, FirstName DESC;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL-Query-MDC-R

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

Record: 1 of 6 No Filter Search

S. *Who referred each customer to Marcia's Dry Cleaning? Show columns named CustomerLastName, CustomerFirstName, ReferredByLastName, and ReferredByFirstName. Include the names of customers who were not referred by any other customer in the results of the query.*

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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).

```
SELECT CUST.LastName AS CustomerLastName, CUST.FirstName AS
   CustomerFirstName, REFER.LastName AS ReferredByLastName,
   REFER.FirstName AS ReferredByFirstName
FROM CUSTOMER CUST LEFT JOIN CUSTOMER REFER
   ON CUST.ReferredBy = REFER.CustomerID;
```

SQL-Query-MDC-S

CustomerLa:	CustomerFir	ReferredByL	ReferredByF
Kaccaton	Nikki		
Catnazaro	Brenda	Kaccaton	Nikki
Miller	Betsy	Kaccaton	Nikki
LeCat	Bruce	Catnazaro	Brenda
Miller	Kathy	Catnazaro	Brenda
Miller	Betsy	LeCat	Bruce
Miller	George	Miller	Betsy

Record: 1 of 7 No Filter Search

T. *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-e15-IM-CH02-MDC.accdb* and in the corresponding files for Oracle Database, SQL

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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:

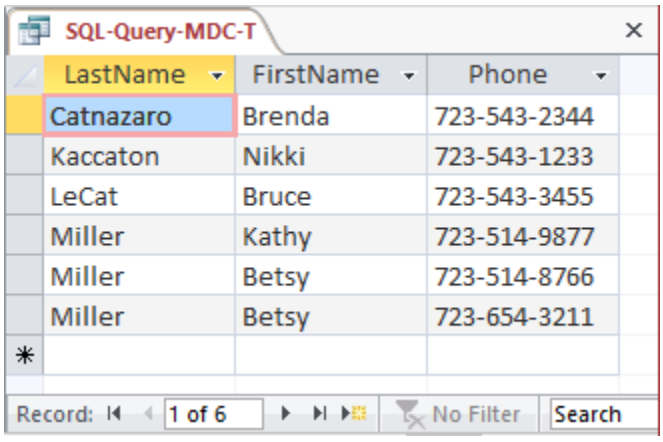
```
/* *** SQL-Query-MDC *** */

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



U. *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.*

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

HINT: In Microsoft Access 2016, you will either need to use a UNION statement or a sequence of two queries to solve this because Microsoft Access disallows nesting an INNER join inside a LEFT OUTER or RIGHT OUTER join. The other DBMS products can complete this question with one query (not a UNION statement).

Solutions to Marcia's Dry Cleaning questions are contained in the Microsoft Access database *DBP-e15-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-U *** */

SELECT LastName, FirstName, Phone, TotalAmount
FROM CUSTOMER C LEFT JOIN (INVOICE I JOIN INVOICE_ITEM II
   ON I.InvoiceNumber = II.InvoiceNumber AND II.Item = 'Dress
              Shirt')
   ON C.CustomerID = I.CustomerID
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-U-UNION *** */

SELECT LastName, FirstName, Phone, TotalAmount
FROM CUSTOMER C, INVOICE I, INVOICE_ITEM II
WHERE C.CustomerID = I.CustomerID AND I.InvoiceNumber =
   II.InvoiceNumber AND II.Item = 'Dress Shirt'
UNION SELECT LastName, FirstName, Phone, NULL
FROM CUSTOMER
WHERE CustomerID NOT IN
   (SELECT CustomerID
    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 CustomerID and TotalAmount for all invoices involving a 'Dress Shirt':

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

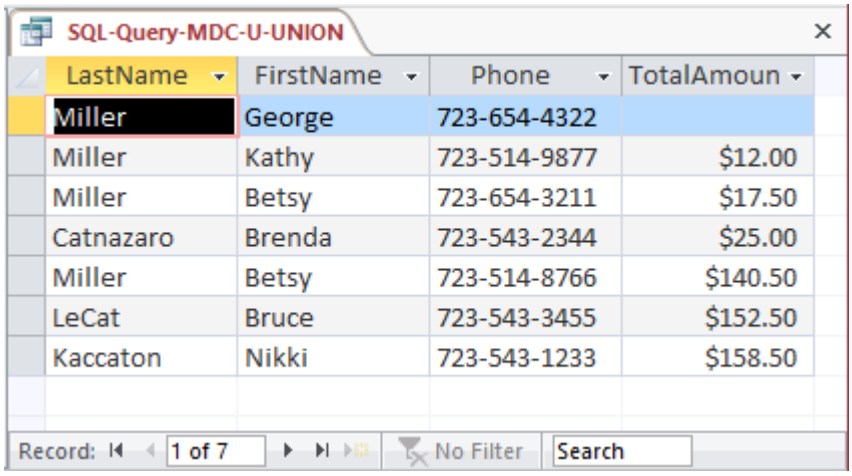
SELECT CustomerID, TotalAmount
FROM INVOICE I, INVOICE_ITEM II
WHERE I.InvoiceNumber = II.InvoiceNumber AND II.Item = 'Dress
Shirt';
```

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

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

SELECT LastName, FirstName, Phone, TotalAmount
FROM CUSTOMER AS C LEFT OUTER JOIN [SQL-Query-MDC-U-Temp] AS T
  ON C.CustomerID = T.CustomerID
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.



Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

❖ 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, EmailAddress, EncryptedPassword, Address, City, State, ZIP, Phone, ReferredBy)

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:

ReferredBy in CUSTOMER must exist in CustomerID in CUSTOMER

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 2-54. Note that CUTOMER contains a recursive relationship between ReferredBy and CustomerID, where ReferredBy contains the CustomerID value of the existing customer who referred the new customer to the Queen Anne Curiosity Shop.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

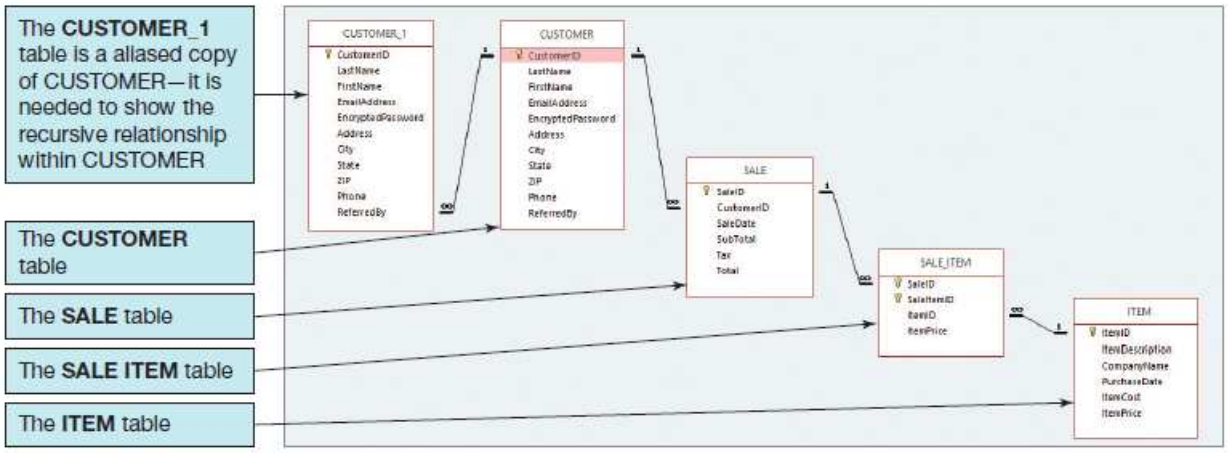


Figure 2-54 – The QACS Database

The column characteristics for the tables are shown in Figures 2-55, 2-56, 2-57, and 2-58. The relationships CUSTOMER-to-SALE and ITEM-to-SALE_ITEM should enforce referential integrity, but not cascade updates or deletions, whereas 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 2-59, 2-60, 2-61, and 2-62.

CUSTOMER

Column Name	Type	Key	Required	Remarks
CustomerID	Integer	Primary Key	Yes	Surrogate Key
LastName	Character (25)	No	Yes	
FirstName	Character (25)	No	Yes	
EmailAddress	Character (100)	No	No	Use Varchar
EncryptedPassword	Character(50)	No	No	Use Varchar
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	
ReferredBy	Integer	Foreign Key	No	REF: CustomerID

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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 2-56 - 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	Yes	

Figure 2-57 - 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 2-58 - Column Characteristics for the QACS Database ITEM Table

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



CustomerID	LastName	FirstName	EmailAddress	EncryptedPassword	Address	City	State	ZIP	Phone	ReferredBy
1	Shire	Robert	Rober.Shire@somewhere.com	56gHj8w	6225 EvanstonAve N	Seattle	WA	98103	206-524-3433	
2	Goodyear	Katherine	Katherine.Goodyear@somewhere.com	fkJU0K24	7335 11th Ave NE	Seattle	WA	98105	206-524-3544	1
3	Bancroft	Chris	Chris.Bancroft@somewhere.com	98bpT4kw	12505 NE 8th Street	Bellevue	WA	98005	425-635-9788	
4	Griffith	John	John.Griffith@somewhere.com	mn8h8jU4	335 Alpha Street	Seattle	WA	98109	206-524-4655	1
5	Tierney	Doris	Doris.Tierney@somewhere.com	as87PP3z	14510 NE 4th Street	Bellevue	WA	98005	425-635-8677	2
6	Anderson	Donna	Donna.Anderson@elsewhere.com	34Gf7e01	1410 Hillcrest Parkway	Mt. Vernon	WA	98273	360-536-7566	3
7	Svane	Jack	Jack.Svane@somewhere.com	wpv7FF9q	3211 42nd Street	Seattle	WA	98115	206-524-5700	1
8	Walsh	Denesha	Denesha.Walsh@somewhere.com	D7gb7T64	6712 24th Avenue NE	Redmond	WA	98053	425-635-7500	5
9	Enquist	Craig	Craig.Enquist@elsewhere.com	gg7ER93t	5341 5th Street	Bellingham	WA	98225	360-536-6455	8
10	Anderson	Rose	Rose.Anderson@elsewhere.com	vx67gH8W	6923 17th Ave NE	Seattle	WA	98105	206-524-6877	3

Figure 2-59 – Sample Data for the QACS Database CUSTOMER Table

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SaleID	CustomerID	SaleDate	SubTotal	Tax	Total
1	1	12/14/2017	$3,500.00	$290.50	$3,790.50
2	2	12/15/2017	$1,000.00	$83.00	$1,083.00
3	3	12/15/2017	$50.00	$4.15	$54.15
4	4	12/23/2017	$45.00	$3.74	$48.74
5	1	1/5/2018	$250.00	$20.75	$270.75
6	5	1/10/2018	$750.00	$62.25	$812.25
7	6	1/12/2018	$250.00	$20.75	$270.75
8	2	1/15/2018	$3,000.00	$249.00	$3,249.00
9	5	1/25/2018	$350.00	$29.05	$379.05
10	7	2/4/2018	$14,250.00	$1,182.75	$15,432.75
11	8	2/4/2018	$250.00	$20.75	$270.75
12	5	2/7/2018	$50.00	$4.15	$54.15
13	9	2/7/2018	$4,500.00	$373.50	$4,873.50
14	10	2/11/2018	$3,675.00	$305.03	$3,980.03
15	2	2/11/2018	$800.00	$66.40	$866.40

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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 2-61 - Sample Data for the QACS Database SALE_ITEM Table

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

ItemID	ItemDescription	CompanyName	PurchaseDate	ItemCost	ItemPrice
1	Antique Desk	European Specialties	11/7/2017	$1,800.00	$3,000.00
2	Antique Desk Chair	Andrew Lee	11/10/2017	$300.00	$500.00
3	Dining Table Linens	Linens and Things	11/14/2017	$600.00	$1,000.00
4	Candles	Linens and Things	11/14/2017	$30.00	$50.00
5	Candles	Linens and Things	11/14/2017	$27.00	$45.00
6	Desk Lamp	Lamps and Lighting	11/14/2017	$150.00	$250.00
7	Dining Table Linens	Linens and Things	11/14/2017	$450.00	$750.00
8	Book Shelf	Denise Harrion	11/21/2017	$150.00	$250.00
9	Antique Chair	New York Brokerage	11/21/2017	$750.00	$1,250.00
10	Antique Chair	New York Brokerage	11/21/2017	$1,050.00	$1,750.00
11	Antique Candle Holders	European Specialties	11/28/2017	$210.00	$350.00
12	Antique Desk	European Specialties	1/5/2018	$1,920.00	$3,200.00
13	Antique Desk	European Specialties	1/5/2018	$2,100.00	$3,500.00
14	Antique Desk Chair	Specialty Antiques	1/6/2018	$285.00	$475.00
15	Antique Desk Chair	Specialty Antiques	1/6/2018	$339.00	$565.00
16	Desk Lamp	General Antiques	1/6/2018	$150.00	$250.00
17	Desk Lamp	General Antiques	1/6/2018	$150.00	$250.00
18	Desk Lamp	Lamps and Lighting	1/6/2018	$144.00	$240.00
19	Antique Dining Table	Denesha Walsh	1/10/2018	$3,000.00	$5,000.00
20	Antique Sideboard	Chris Bancroft	1/11/2018	$2,700.00	$4,500.00
21	Dining Table Chairs	Specialty Antiques	1/11/2018	$5,100.00	$8,500.00
22	Dining Table Linens	Linens and Things	1/12/2018	$450.00	$750.00
23	Dining Table Linens	Linens and Things	1/12/2018	$480.00	$800.00
24	Candles	Linens and Things	1/17/2018	$30.00	$50.00
25	Candles	Linens and Things	1/17/2018	$36.00	$60.00

Figure 2-62 - Sample Data for the QACS Database ITEM Table

You will need to create and set up a database named QACS_CH02 for use with The Queen Anne Curiosity Shop project questions. A Microsoft Access 2016 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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

If you are using the Microsoft Access 2016 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 2017, see online Chapter 10A.*
- *For Oracle Database 12c Release 2 or Oracle Express Edition 11g Release 2, see Online Chapter 10B.*
- *For MySQL 5.7 Community Server, see online Chapter 10C.*

Once you have set up your QACS_CH02 database, create an SQL script named QACSCH02-PQ.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-e15-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	EmailAddress	EncryptedPassword	Address	City	State	ZIP	Phone	ReferredBy
1	Shire	Robert	Robert.Shire@somewhere.com	56gHjj8w	6225 Evanston Ave N	Seattle	WA	98103	206-524-2433	
2	Goodyear	Katherine	Katherine.Goodyear@somewhere.com	fkJU0K24	7335 11th Ave NE	Seattle	WA	98105	206-524-3544	1
3	Bancroft	Chris	Chris.Bancroft@somewhere.com	98bpT4vw	12605 NE 6th Street	Bellevue	WA	98005	425-635-9788	
4	Griffith	John	John.Griffith@somewhere.com	mnBh88t4	335 Aloha Street	Seattle	WA	98109	206-524-4655	1
5	Tierney	Doris	Doris.Tierney@somewhere.com	as87PP3z	14510 NE 4th Street	Bellevue	WA	98005	425-635-8677	2
6	Anderson	Donna	Donna.Anderson@elsewhere.com	34Gf7e0t	1410 Hillcrest Parkway	Mt. Vernon	WA	98273	360-538-7566	3
7	Svane	Jack	Jack.Svane@somewhere.com	wpv7FF9q	3211 42nd Street	Seattle	WA	98115	206-524-5766	1
8	Walsh	Denesha	Denesha.Walsh@somewhere.com	D7gb7T84	6712 24th Avenue NE	Redmond	WA	98053	425-635-7566	5
9	Enquist	Craig	Craig.Enquist@elsewhere.com	gg7ER53t	534 15th Street	Bellingham	WA	98225	360-538-6455	6
10	Anderson	Rose	Rose.Anderson@elsewhere.com	vx67gH8W	6823 17th Ave NE	Seattle	WA	98105	206-524-6877	3
(New)										0

Record: 1 of 10 No Filter Search

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

SELECT    *
FROM      SALE;
```

SQL-Query-QACS-A-SALE

SaleID	CustomerID	SaleDate	SubTotal	Tax	Total
1	1	12/14/2017	$3,500.00	$290.50	$3,790.50
2	2	12/15/2017	$1,000.00	$83.00	$1,083.00
3	3	12/15/2017	$50.00	$4.15	$54.15
4	4	12/23/2017	$45.00	$3.74	$48.74
5	1	1/5/2018	$250.00	$20.75	$270.75
6	5	1/10/2018	$750.00	$62.25	$812.25
7	6	1/12/2018	$250.00	$20.75	$270.75
8	2	1/15/2018	$3,000.00	$249.00	$3,249.00
9	5	1/25/2018	$350.00	$29.05	$379.05
10	7	2/4/2018	$14,250.00	$1,182.75	$15,432.75
11	8	2/4/2018	$250.00	$20.75	$270.75
12	5	2/7/2018	$50.00	$4.15	$54.15
13	9	2/7/2018	$4,500.00	$373.50	$4,873.50
14	10	2/11/2018	$3,675.00	$305.03	$3,980.03
15	2	2/11/2018	$800.00	$66.40	$866.40
*	(New)				

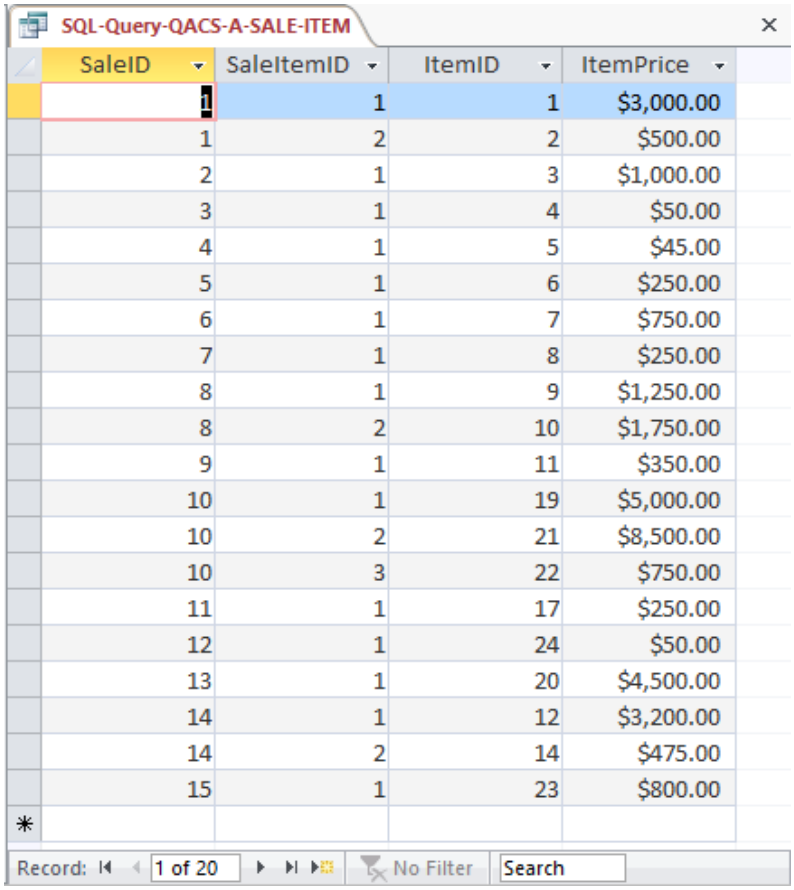
Record: 1 of 15 No Filter Search

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
/* *** 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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

SELECT    *
FROM      ITEM;
```

SQL-Query-QACS-A-ITEM

ItemID	ItemDescription	CompanyName	PurchaseDate	ItemCost	ItemPrice
1	Antique Desk	European Specialties	11/7/2017	$1,800.00	$3,000.00
2	Antique Desk Chair	Andrew Lee	11/10/2017	$300.00	$500.00
3	Dining Table Linens	Linens and Things	11/14/2017	$600.00	$1,000.00
4	Candles	Linens and Things	11/14/2017	$30.00	$50.00
5	Candles	Linens and Things	11/14/2017	$27.00	$45.00
6	Desk Lamp	Lamps and Lighting	11/14/2017	$150.00	$250.00
7	Dining Table Linens	Linens and Things	11/14/2017	$450.00	$750.00
8	Book Shelf	Denise Harrison	11/21/2017	$150.00	$250.00
9	Antique Chair	New York Brokerage	11/21/2017	$750.00	$1,250.00
10	Antique Chair	New York Brokerage	11/21/2017	$1,050.00	$1,750.00
11	Antique Candle Holder	European Specialties	11/28/2017	$210.00	$350.00
12	Antique Desk	European Specialties	1/5/2018	$1,920.00	$3,200.00
13	Antique Desk	European Specialties	1/5/2018	$2,100.00	$3,500.00
14	Antique Desk Chair	Specialty Antiques	1/6/2018	$285.00	$475.00
15	Antique Desk Chair	Specialty Antiques	1/6/2018	$339.00	$565.00
16	Desk Lamp	General Antiques	1/6/2018	$150.00	$250.00
17	Desk Lamp	General Antiques	1/6/2018	$150.00	$250.00
18	Desk Lamp	Lamps and Lighting	1/6/2018	$144.00	$240.00
19	Antique Dining Table	Denesha Walsh	1/10/2018	$3,000.00	$5,000.00
20	Antique Sideboard	Chris Bancroft	1/11/2018	$2,700.00	$4,500.00
21	Dining Table Chairs	Specialty Antiques	1/11/2018	$5,100.00	$8,500.00
22	Dining Table Linens	Linens and Things	1/12/2018	$450.00	$750.00
23	Dining Table Linens	Linens and Things	1/12/2018	$480.00	$800.00
24	Candles	Linens and Things	1/17/2018	$30.00	$50.00
25	Candles	Linens and Things	1/17/2018	$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-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL-Query-QACS-D

LastName	FirstName	Phone	SaleDate	Total
Shire	Robert	206-524-2433	12/14/2017	$3,790.50
Goodyear	Katherine	206-524-3544	12/15/2017	$1,083.00
Shire	Robert	206-524-2433	1/5/2018	$270.75
Tierney	Doris	425-635-8677	1/10/2018	$812.25
Anderson	Donna	360-538-7566	1/12/2018	$270.75
Goodyear	Katherine	206-524-3544	1/15/2018	$3,249.00
Tierney	Doris	425-635-8677	1/25/2018	$379.05
Svane	Jack	206-524-5766	2/4/2018	$15,432.75
Walsh	Denesha	425-635-7566	2/4/2018	$270.75
Enquist	Craig	360-538-6455	2/7/2018	$4,873.50
Anderson	Rose	206-524-6877	2/11/2018	$3,980.03
Goodyear	Katherine	206-524-3544	2/11/2018	$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-e15-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*';
```

SQL-Query-QACS-E

LastName	FirstName	Phone
Tierney	Doris	425-635-8677
Anderson	Donna	360-538-7566
Walsh	Denesha	425-635-7566

Record: 1 of 3 No Filter Search

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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:

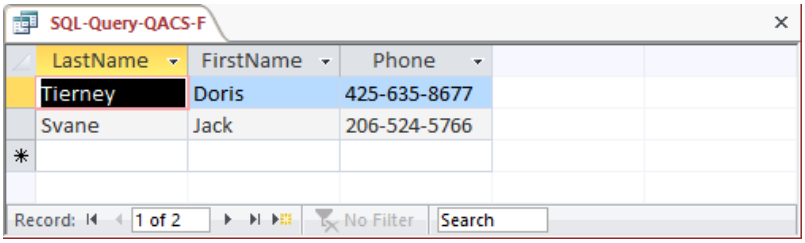
```
/* *** SQL-Query-QACS-F *** */

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-e15-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_';
```

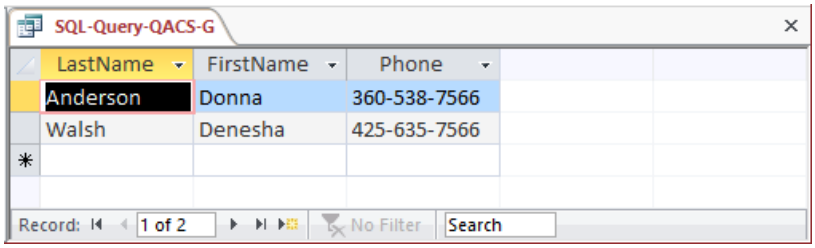
For Microsoft Access:

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

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

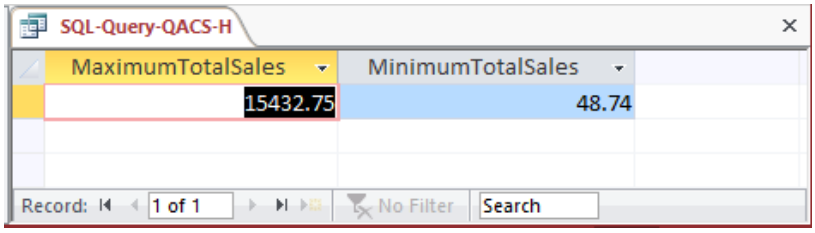


H. Determine the maximum and minimum sales Total.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e15-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;
```

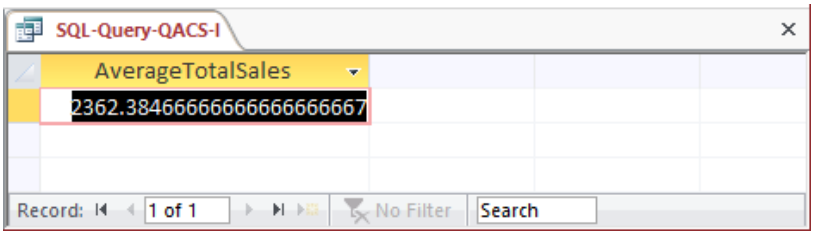


I. Determine the average sales Total.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e15-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;
```



Copyright © 2019 Pearson Education, Inc.

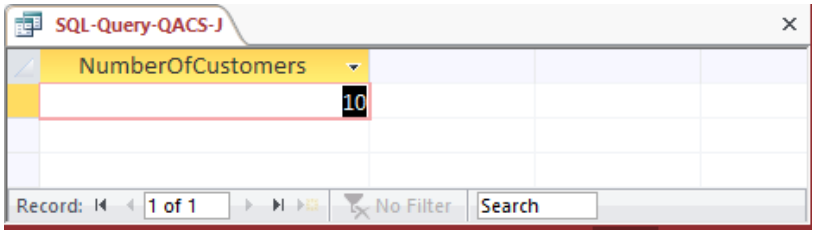
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

J. *Count the number of customers.*

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e15-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;
```

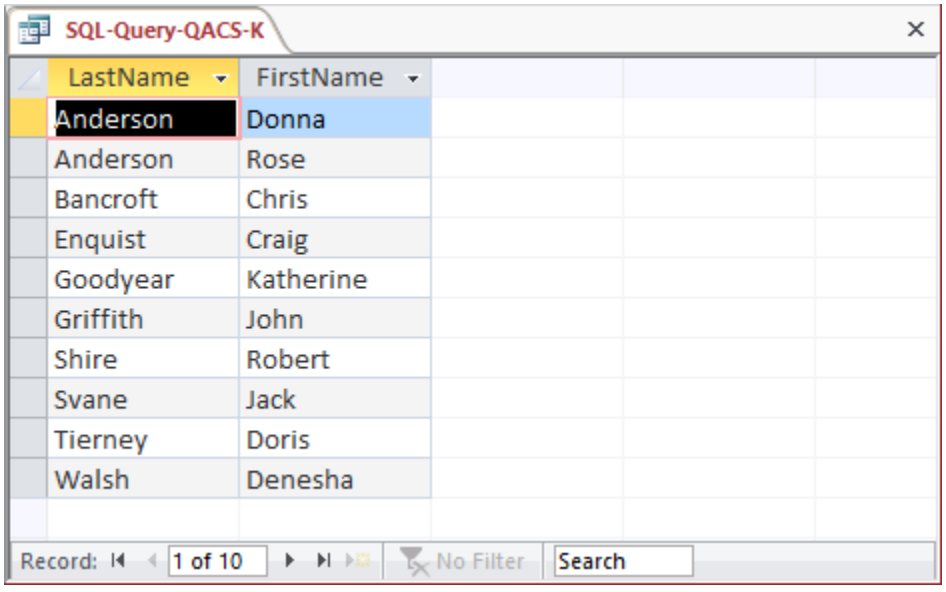


K. *Group customers by LastName and then by FirstName.*

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e15-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;
```



Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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;
```

SQL-Query-QACS-L

LastName	FirstName	NumberOfC
Anderson	Donna	1
Anderson	Rose	1
Bancroft	Chris	1
Enquist	Craig	1
Goodyear	Katherine	1
Griffith	John	1
Shire	Robert	1
Svane	Jack	1
Tierney	Doris	1
Walsh	Denesha	1

Record: 1 of 10 No Filter Search

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-e15-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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL-Query-QACS-M

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

Record: 1 of 8 No Filter Search

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-e15-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
ORDER BY LastName, FirstName DESC;

/*       For each CUSTOMER only once:                                    */

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

SQL-Query-QACS-N

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

Record: 1 of 8 No Filter Search

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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 *** */

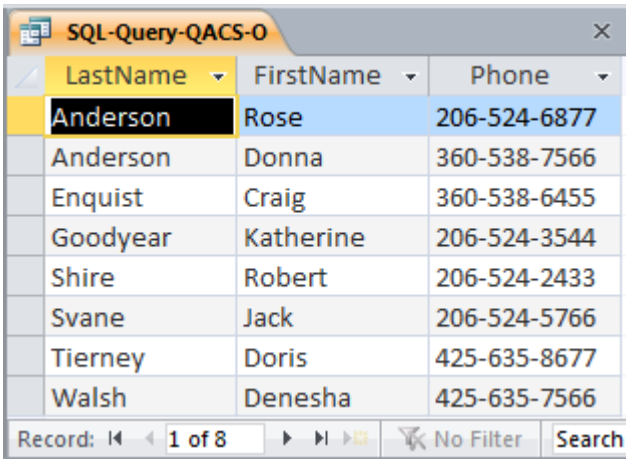
SELECT     LastName, FirstName, Phone
FROM       CUSTOMER JOIN SALE
   ON      CUSTOMER.CustomerID = SALE.CustomerID
WHERE      Total  > 100
ORDER BY   LastName, FirstName DESC;

/*       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.*

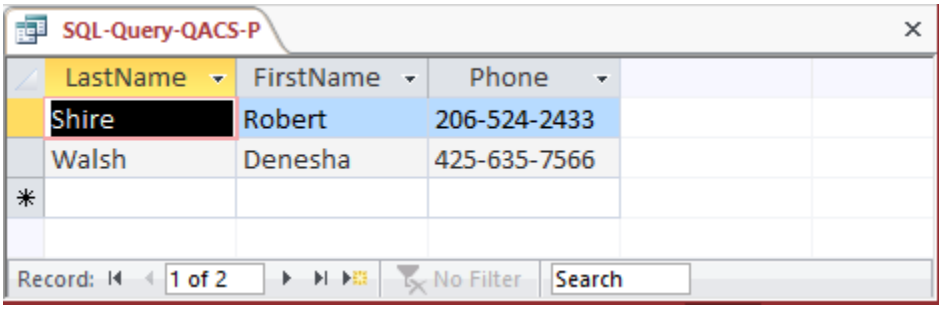
Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e15-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;
```



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-e15-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;
```

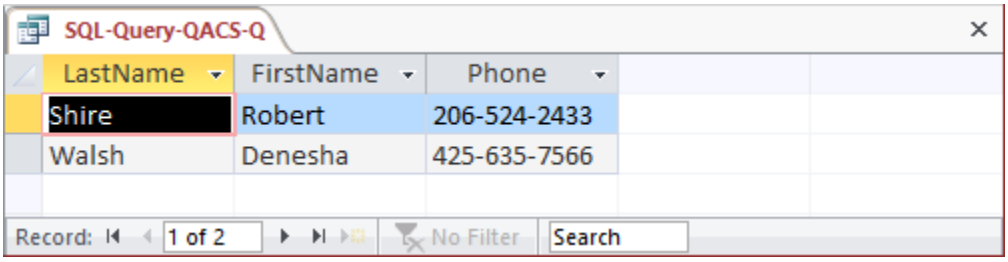

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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:

Copyright © 2019 Pearson Education, Inc.

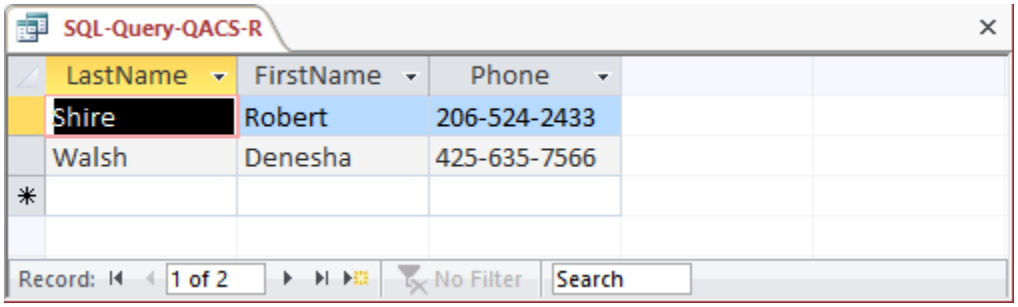
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
/* *** 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-e15-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:

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
/* *** 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:

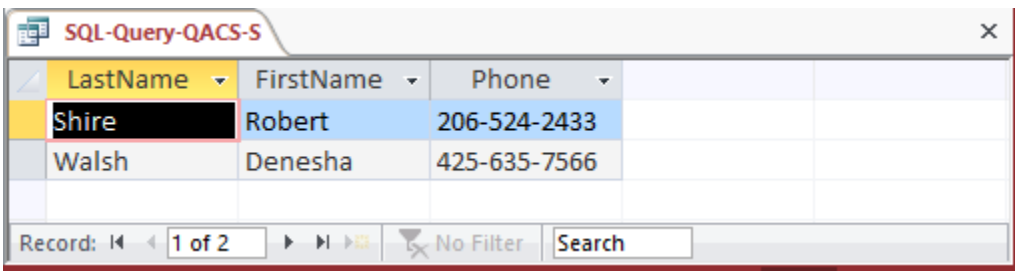
```
/* *** SQL-Query-QACS-S *** */

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;
```



Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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;
```

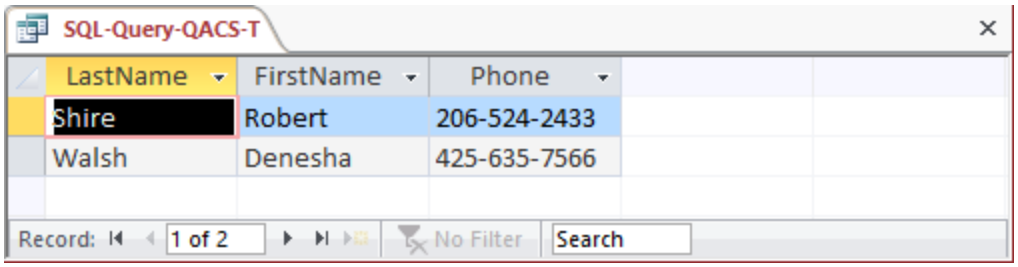
For Microsoft Access, which requires "INNER" in join syntax and parenthesization of multiple joins performed using JOIN syntax:

```
/* *** 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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



U. Show the LastName, FirstName, Phone, and ItemDescription 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. HINT: In Microsoft Access 2016 you will either need to use a UNION statement or a sequence of two queries to solve this, because Microsoft Access disallows nesting an INNER join inside a LEFT or RIGHT join. The other DBMS products can do it with one query (not a UNION statement).

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e15-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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
/* *** 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:

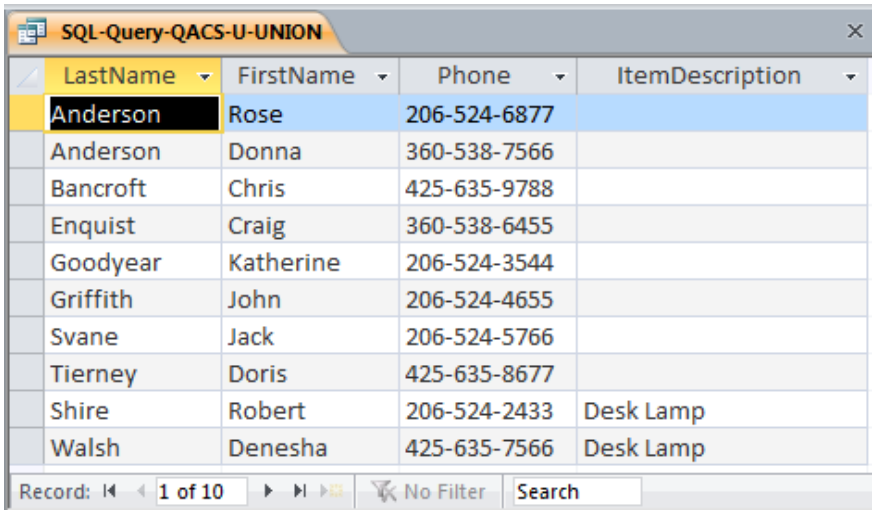
```
/* *** SQL-Query-QACS-U-Final *** */

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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



SQL-Query-QACS-U-UNION

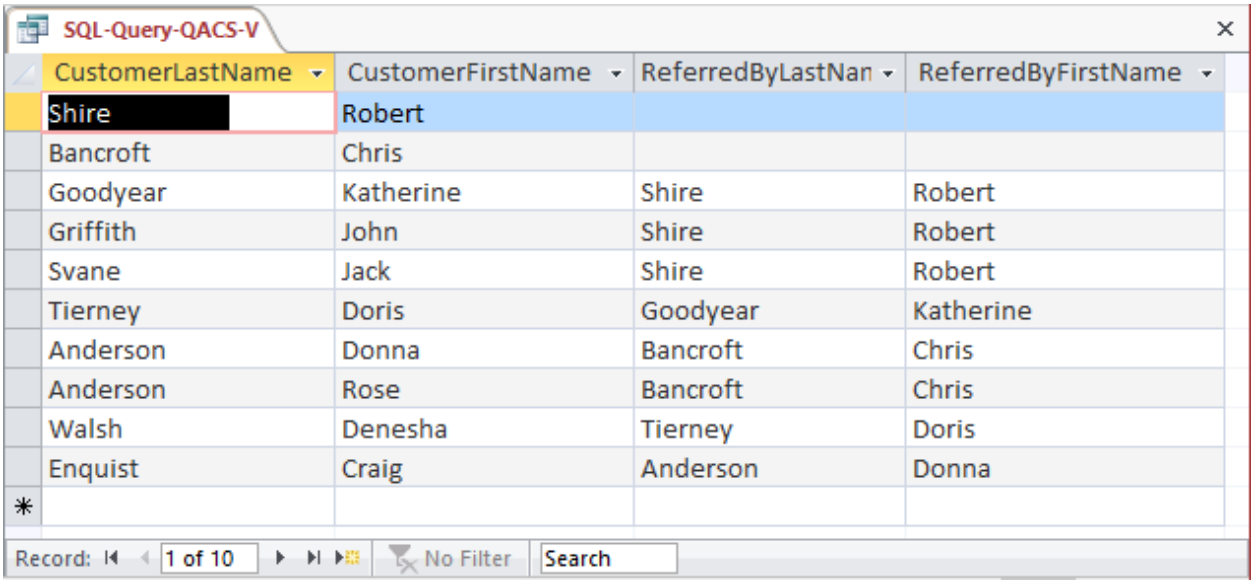
LastName	FirstName	Phone	ItemDescription
Anderson	Rose	206-524-6877	
Anderson	Donna	360-538-7566	
Bancroft	Chris	425-635-9788	
Enquist	Craig	360-538-6455	
Goodyear	Katherine	206-524-3544	
Griffith	John	206-524-4655	
Svane	Jack	206-524-5766	
Tierney	Doris	425-635-8677	
Shire	Robert	206-524-2433	Desk Lamp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

Record: 1 of 10 No Filter Search

V. Who referred each customer to the Queen Anne Curiosity Shop? Show columns named CustomerLastName, CustomerFirstName, ReferredByLastName, and ReferredByFirstName. Include the names of all customers who were not referred by any other customer in the results of the query.

Solutions to The Queen Anne Curiosity Shop questions are contained in the Microsoft Access database *DBP-e15-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).

```
SELECT CUST.LastName AS CustomerLastName, CUST.FirstName AS
      CustomerFirstName, REFER.LastName AS ReferredByLastName,
      REFER.FirstName AS ReferredByFirstName
FROM CUSTOMER CUST LEFT JOIN CUSTOMER REFER
      ON CUST.ReferredBy = REFER.CustomerID;
```



SQL-Query-QACS-V

CustomerLastName	CustomerFirstName	ReferredByLastNan	ReferredByFirstName
Shire	Robert		
Bancroft	Chris		
Goodyear	Katherine	Shire	Robert
Griffith	John	Shire	Robert
Svane	Jack	Shire	Robert
Tierney	Doris	Goodyear	Katherine
Anderson	Donna	Bancroft	Chris
Anderson	Rose	Bancroft	Chris
Walsh	Denesha	Tierney	Doris
Enquist	Craig	Anderson	Donna
*			

Record: 1 of 10 No Filter Search

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

❖ 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, 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 2-63.

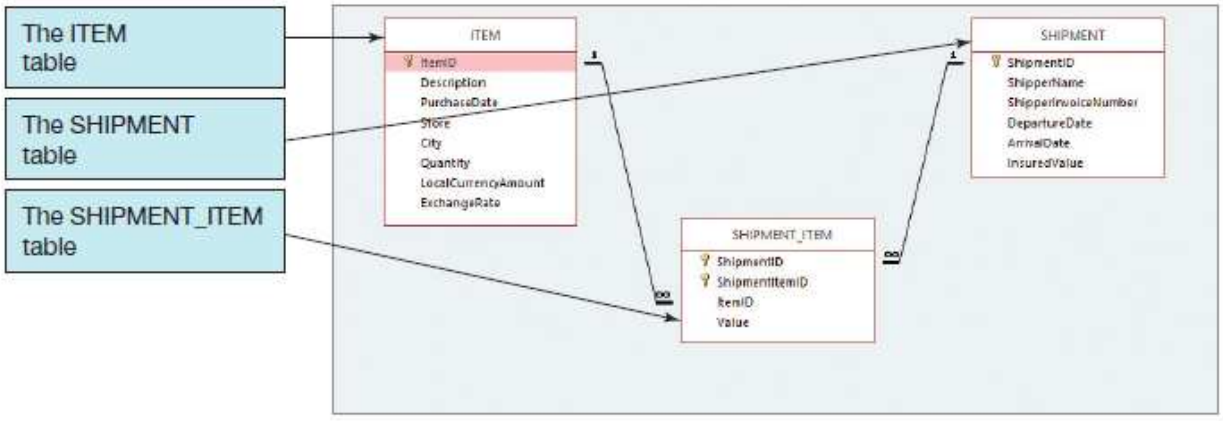


Figure 2-63 – The MI Database

The column characteristics for the tables are shown in Figures 2-64, 2-65, and 2-66. The data for the tables are shown in Figures 2-67, 2-68, and 2-69. 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 set up a database named MI_CH02 for use with the Morgan Importing case questions. A Microsoft Access 2016 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 2016 MI_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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

product you are using:

- *For Microsoft SQL Server 2017, see online Chapter 10A.*
- *For Oracle Database 12c Release 2 or Oracle Express Edition 11g Release 2, see online Chapter 10B.*
- *For MySQL 5.7 Community Server, see online Chapter 10C.*

Once you have set up your MI_CH02 database, create an SQL script named MI-CH02-PQ.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 2-64 - 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	

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-65 - 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 2-66 - Column Characteristics for the MI Database SHIPMENT_ITEM Table

ItemID	Description	PurchaseDate	Store	City	Quantity	LocalCurrencyAmount	ExchangeRate
1	QE Dining Set	07-Apr-18	Eastern Treasures	Manila	2	403405	0.01774
2	Willow Serving Dishes	15-Jul-18	Jade Antiques	Singapore	75	102	0.5903
3	Large Bureau	17-Jul-18	Eastern Sales	Singapore	8	2000	0.5903
4	Brass Lamps	20-Jul-18	Jade Antiques	Singapore	40	50	0.5903

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

ShipmentID	ShipperName	ShipperInvoiceNumber	DepartureDate	ArrivalDate	InsuredValue
1	ABC Trans-Oceanic	2017651	10-Dec-17	15-Mar-18	$15,000.00
2	ABC Trans-Oceanic	2018012	10-Jan-18	20-Mar-18	$12,000.00
3	Worldwide	49100300	05-May-18	17-Jun-18	$20,000.00
4	International	399400	02-Jun-18	17-Jul-18	$17,500.00
5	Worldwide	84899440	10-Jul-18	28-Jul-18	$25,000.00
6	International	488955	05-Aug-18	11-Sep-18	$18,000.00

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

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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 2-69 - 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-e15-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	4/7/2018	Eastern Treasures	Manila	2	403405	0.01774
2	Willow Serving Dishes	7/15/2018	Jade Antiques	Singapore	75	102	0.5903
3	Large Bureau	7/17/2018	Eastern Sales	Singapore	8	2000	0.5903
4	Brass Lamps	7/20/2018	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	2017651	12/10/2017	3/15/2018	$15,000.00
2	ABC Trans-Oceanic	2018012	1/10/2018	3/20/2018	$12,000.00
3	Worldwide	49100300	5/5/2018	6/17/2018	$20,000.00
4	International	399400	6/2/2018	7/17/2018	$17,500.00
5	Worldwide	84899440	7/10/2018	7/28/2018	$25,000.00
6	International	488955	8/5/2018	9/11/2018	$18,000.00
*					

Record: 1 of 6 No Filter Search

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

SELECT    *
FROM      SHIPMENT_ITEM;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL-Query-MI-A-SHIPMENT_ITEM

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

Record: 1 of 4 No Filter Search

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

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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;
```

SQL-Query-MI-B

ShipmentID	ShipperName	ShipperInvoiceNumber
1	ABC Trans-Oceanic	2017651
2	ABC Trans-Oceanic	2018012
3	Worldwide	49100300
4	International	399400
5	Worldwide	84899440
6	International	488955

Record: 1 of 6 No Filter Search

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

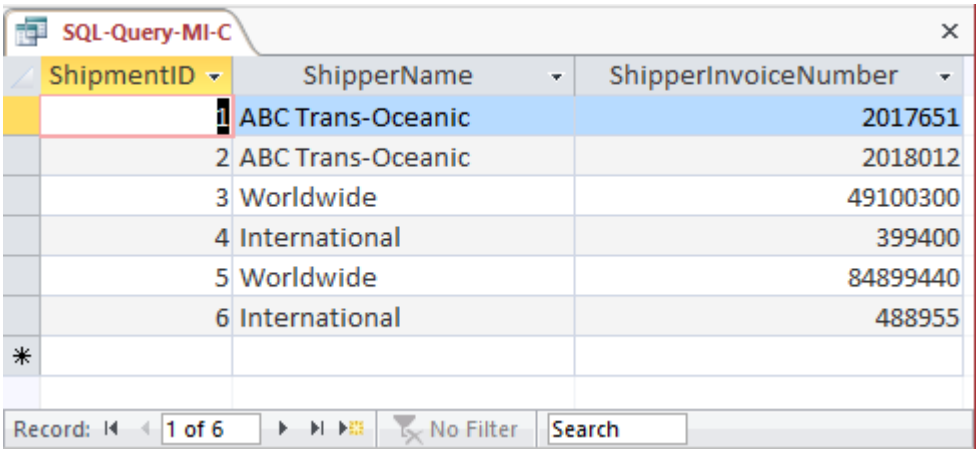
Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

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

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


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



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-e15-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:

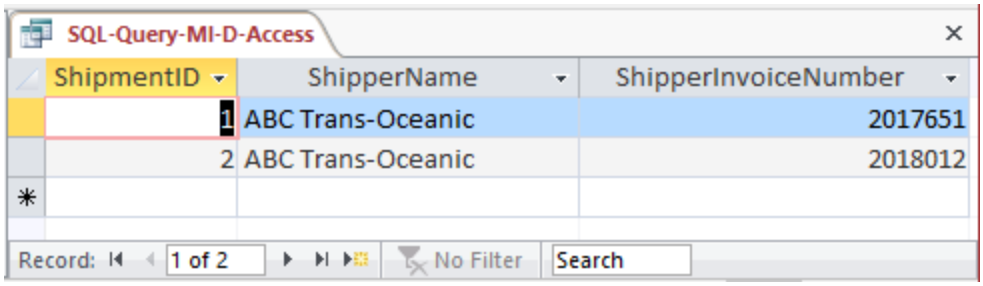
```
/* *** SQL-Query-MI-D *** */

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*';
```



Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

E. List the ShipmentID, ShipperName, ShipperInvoiceNumber, and ArrivalDate of all shipments that departed in December. HINT: For the DBMS you are using, research how to extract a month or day from a date value so it can be compared to a number.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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).

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:

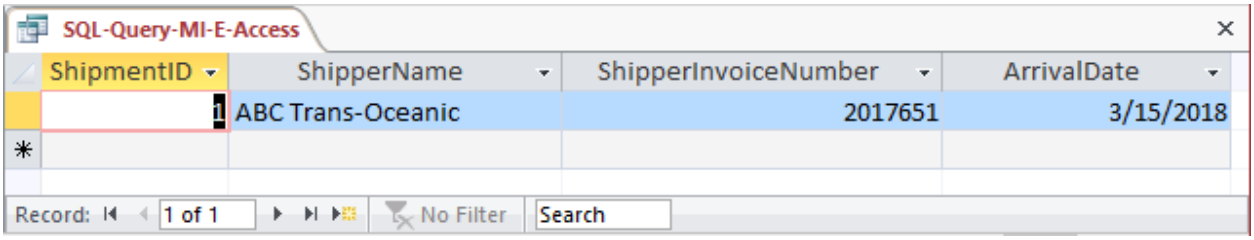
```
/* *** SQL-Query-MI-E-Oracle *** */

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

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

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

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



F. List the ShipmentID, ShipperName, ShipperInvoiceNumber, and ArrivalDate of all shipments that departed on the tenth day of any month. HINT: For the DBMS you are using, research how to extract a month or day from a date value so it can be compared to a number.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL,

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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

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:

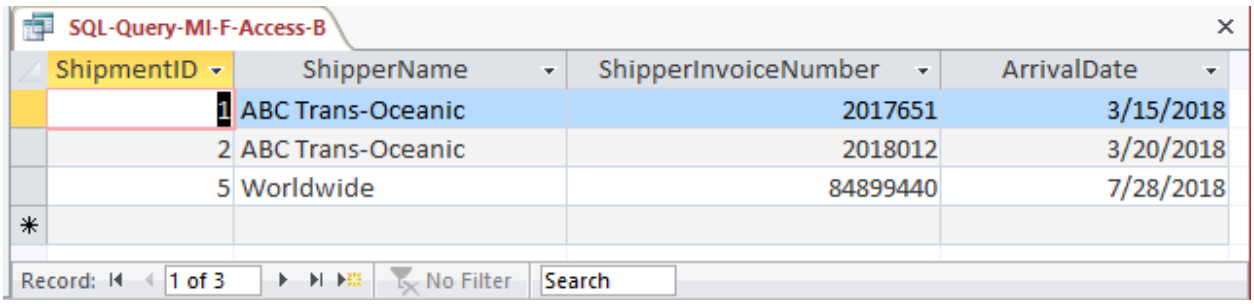
```
/* *** SQL-Query-MI-F-Oracle *** */

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

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

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

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



Copyright © 2019 Pearson Education, Inc.

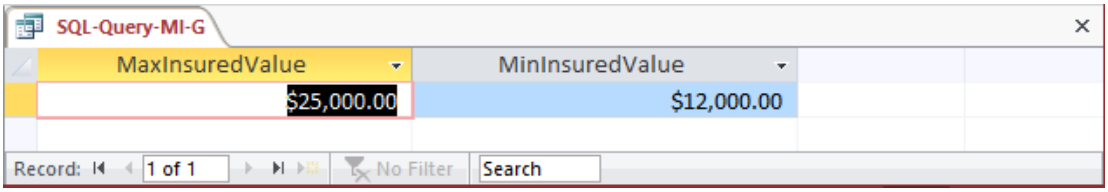
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

G. Determine the maximum and minimum InsuredValue.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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;
```

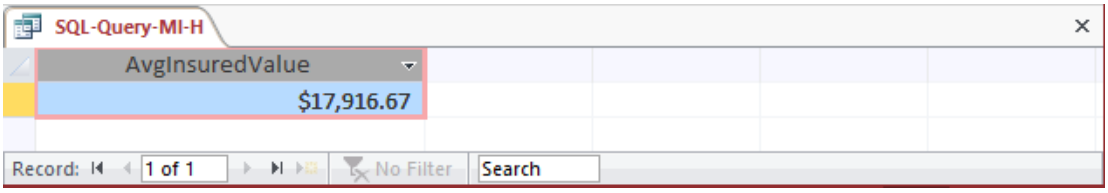


H. Determine the average InsuredValue.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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.

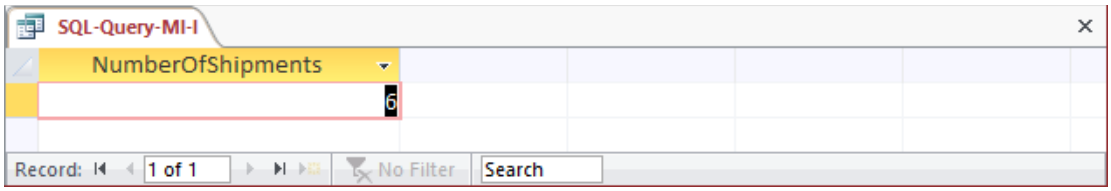
Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-IM-CH02-MI.accdb* and in the corresponding files for Oracle Database, MySQL, and SQL Server, which are all available in the Instructor's Resource Center on the text's Web site (www.pearsonhighered.com/kroenke).

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

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


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

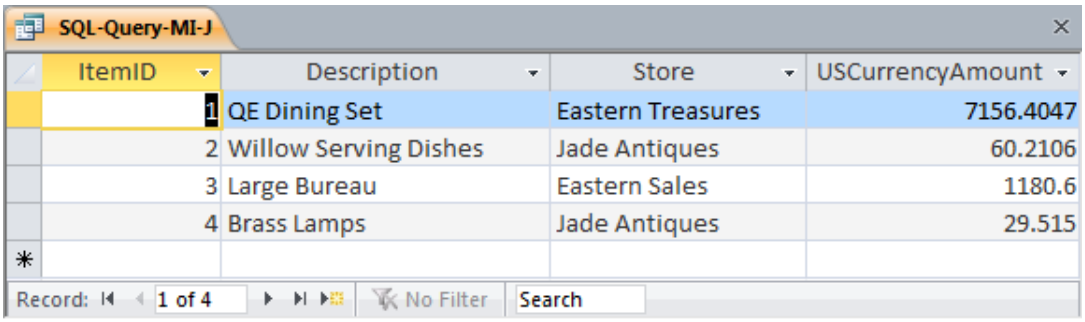


J. Show ItemID, Description, Store, and a calculated column named USCurrencyAmount that is equal to LocalCurrencyAmount multiplied by the ExchangeRate for all rows of ITEM.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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;
```

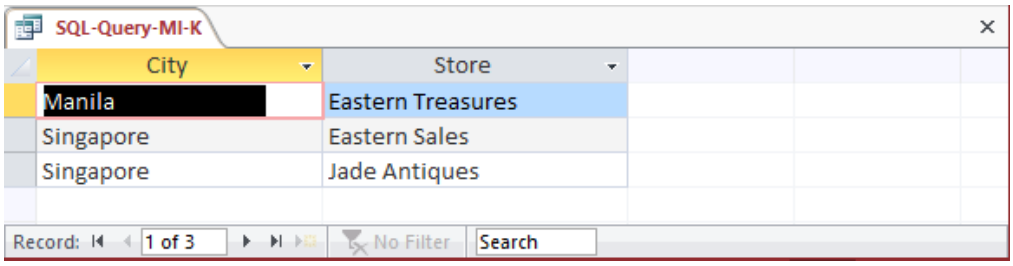


K. Group item purchases by City and Store.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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;
```



Copyright © 2019 Pearson Education, Inc.

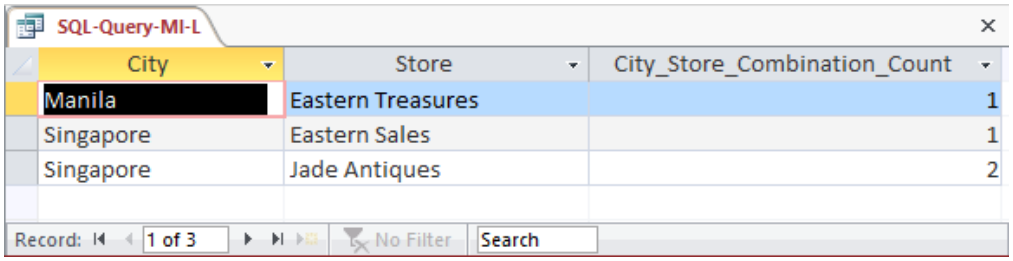
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

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-e15-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;
```

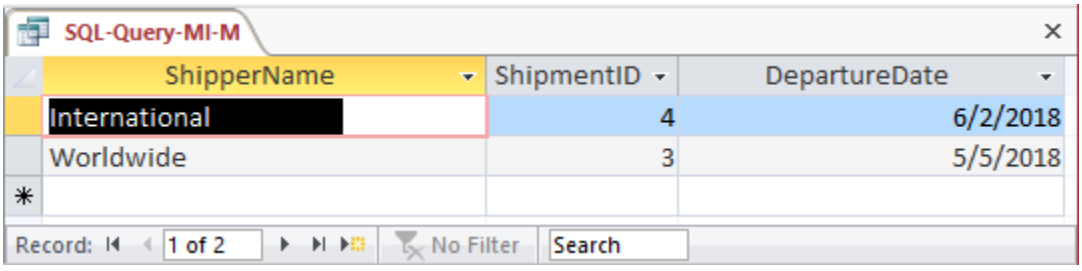


M. Show the ShipperName, ShipmentID and DepartureDate of all shipments that have an item with a value of $1,000.00 or more. Use a subquery. Present results sorted by ShipperName in ascending order and then DepartureDate in descending order.

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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 since when doing the join (instead of the subquery) we may have duplicates in the result if a shipment has more than one item valued at at least $1000. Note how the following query determines 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;
```

SQL-Query-MI-N-A

ShipperName	ShipmentID	DepartureDate
International	4	6/2/2018
International	4	6/2/2018
International	4	6/2/2018
Worldwide	3	5/5/2018

Record: 1 of 4 No Filter Search

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;
```

SQL-Query-MI-N-B

ShipperName	ShipmentID	DepartureDate
International	4	6/2/2018
Worldwide	3	5/5/2018

Record: 1 of 2 No Filter Search

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.*

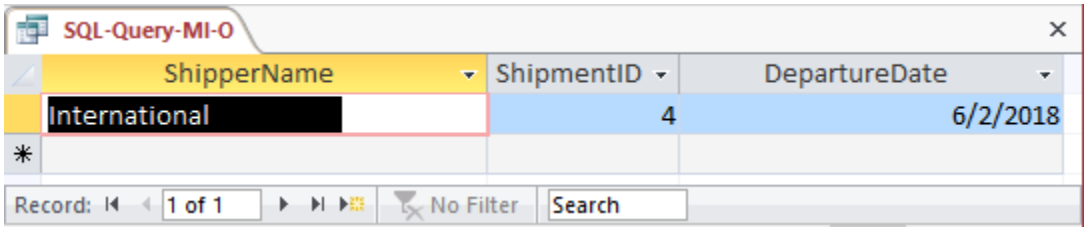
Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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-e15-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).

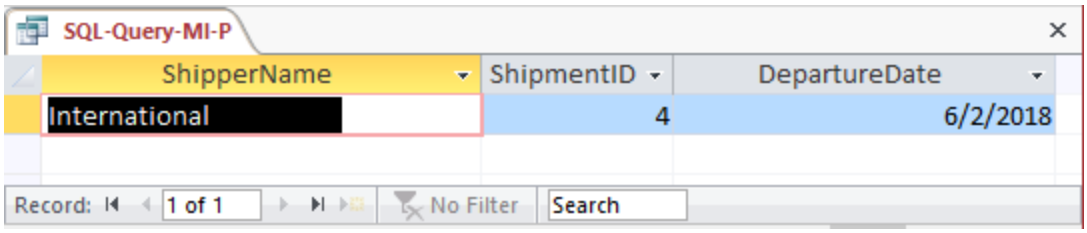
As in question N, we will have to use a DISTINCT keyword to guarantee the appropriate answer.

```
/* *** 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;
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



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-e15-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:

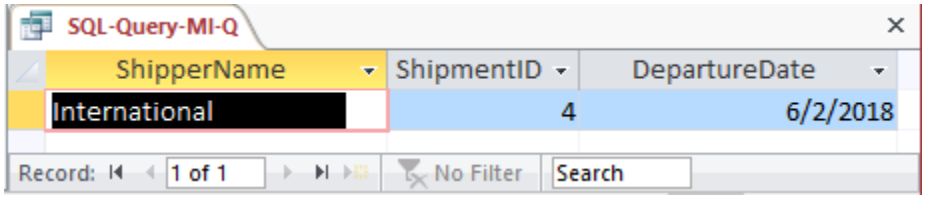
```
/* *** SQL-Query-MI-Q *** */

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.*

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Solutions to Morgan Importing questions are contained in the Microsoft Access database *DBP-e15-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	6/2/2018	$4,500.00
International	4	6/2/2018	$9,500.00
International	4	6/2/2018	$1,200.00

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-e15-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
```


Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

```
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.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL-Query-MI-S-UNION

ShipperNam	ShipmentID	DepartureDa	Value
ABC Trans-Oce	2	1/10/2018	
ABC Trans-Oce	1	12/10/2017	
International	6	8/5/2018	
Worldwide	5	7/10/2018	
Worldwide	3	5/5/2018	
International	4	6/2/2018	$1,200.00
International	4	6/2/2018	$4,500.00
International	4	6/2/2018	$9,500.00

Record: 1 of 8 No Filter Search

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

INSTRUCTOR'S MANUAL

TO ACCOMPANY

David M. Kroenke | David J. Auer | Scott L. Vandenberg | Robert C. Yoder

40th Anniversary Edition

DATABASE PROCESSING

Fundamentals, Design, and Implementation

15th Edition

Appendix B

Getting Started with Systems Analysis and Design

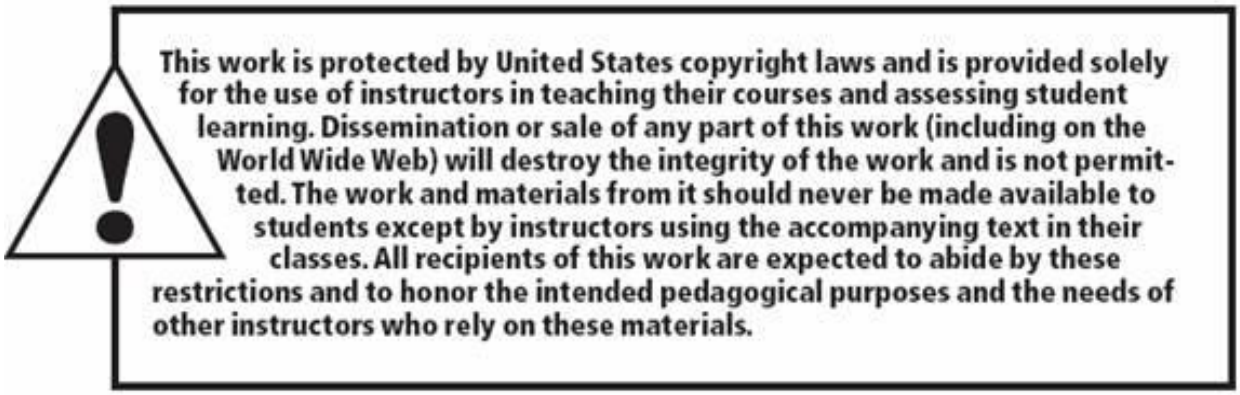


Prepared By

Robert C. Yoder

Siena College

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual



Instructor's Manual to accompany:

Database Processing: Fundamental, Design, and Implementation (15th Edition)

David M. Kroenke | David J. Auer | Scott L. Vandenberg | Robert C. Yoder

---

Copyright © 2019 Pearson Education, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

CHAPTER OBJECTIVES

- To understand information systems
- To understand business processes
- To understand and be able to apply the systems development life cycle (SDLC) model
- To understand business process modeling using Business Processing Modeling Notation (BPMN)
- To be able to gather data and information during requirements analysis
- To understand use cases
- To understand business rules
- To be able to create a user requirements document (URD)
- To be able to create a statement of work (SOW)
- To understand how the topics in this appendix link to *Database Processing (15th Edition)*

ERRATA

There are no known errors at this time. Any errors that are discovered in the future will be reported and corrected in the Online DBP e15 Errata document, which will be available at http://www.pearsonhighered.com/kroenke.

TEACHING SUGGESTIONS

- The general purpose of this appendix is to provide a context for the study of data models in Chapter 5 and database designs in Chapter 6. It can be used as an introduction to or in conjunction with DBP Part II – Database Design, and you should assign it as a reading before your students read Chapter 5 or Chapter 6 depending on how it fits into your presentation of this material.
- However, there is enough significant overlap with Chapter 5 and Chapter 6 topics that you may want to assign it in parts, or, alternatively, have your students reread appropriate sections as you work through the chapters.
- There is also material (on the SDLC Implementation and System Maintenance steps) that can be used with Chapter 9 on database administration and managing multiuser databases. You can tie many of the topics in that chapter back to the SDLC.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

ANSWERS TO REVIEW QUESTIONS

*B.1 What is a **decision**?*

A **decision** is a selected action that we should take in our current situation.

*B.2 What is **data**?*

Data are recorded facts and numbers.

*B.3 What is **information**?*

We can now define **information** as:

- Knowledge derived from data.
- Data presented in a meaningful context.
- Data processed by summing, ordering, averaging, grouping, comparing, or other similar operations.

*B.4 What is a **system**? What is an **information system**?*

A **system** is a set of components that interact to achieve some purpose or goal.

An **information system** is a system that has the goal of producing information.

*B.5 What is a **computer-based information system**? Describe the five components of a computer based information system.*

A **computer-based information system** is an information system comprised of five components: Computer hardware, software, data, procedures, and people.



*B.6 What is a **competitive strategy**?*

A **competitive strategy** is a company's organized response to the **industry structure** of the industry in which it operates, and thus defines how the company will compete within that structure. The industry structure can be defined using Michael Porter's five forces model.

*B.7 Describe Michael Porter's **five forces** model.*

According to Porter's **five forces model**, the structure of an industry is determined by relative strength or weakness within the industry of:

- **The bargaining power of customers**: can the industry's customers influence the industry?
- **The availability ("threat") of substitutable products**: what other products do competitors have?

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

- **The bargaining power of suppliers**: can the industry's suppliers influence the industry?
- **The ease ("threat") of new competitors entering the industry**: how easily can a new company get a start in the industry?
- **The rivalry between competitors**: how many competitors are there, and how much do they really compete?

*B.8 Describe Michael Porter's **primary activities**.*

Michael Porter defines the **primary activities** or **operational activities** of a business as:

- **Inbound logistics:** receiving, storing, and distributing product inputs
- **Manufacturing operations**: transforming inputs into the final product
- **Outbound logistics**: collecting, storing, and distributing the product to buyers
- **Sales and marketing**: convincing customers to buy the product and selling it to them
- **Customer service**: assisting the customers in their use of the product

*B.9 Describe Michael Porter's **support activities**.*

Michael Porter defines the **support activities,** which, as the name implies, support the operational activities, as:

- **Procurement**: managing supplier relationships and buying the product inputs
- **Technology management**: product research and development and new procedures, methods, and techniques
- **Human resources management**: managing employee resources
- **Firm infrastructure management**: general management of the firm, finances, accounting, legal services, and government affairs

*B.10 What is a **business process**?*

A **business process** is a set of **activities** that transform **inputs** into **outputs**.



B.11 How do information systems support business processes?

Information systems support business processes by providing applications that help the process operate more efficiently.

B.12 Describe how information systems include processes.

A generalized conceptual **process** is ***input → process → output***. Information systems include such processes. For example, an inventory control application may include these steps as:

- **Input data**: Using a computer on-screen form, a user will input data into the system, such as how many parts have been taken out of the raw materials inventory.
- **Process**: Another user may query the raw materials database to determine how many of each part remain in the raw materials inventory.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

- **Output**: The answer to the query will be presented to the requesting user as a report detailing the current status of the parts in the raw materials inventory.

*B.13 What is **business process modeling**?*

Business process modeling is a technique to help us understand a business process before we design an information system to support it. To do this, we study the process and document it using business process modeling notation (BPMN). After documenting the process, we will generally understand something about what sort of information system is needed to support the process, and we can then use systems analysis and design methodology to create and maintain the needed information system.

*B.14 What is **systems analysis and design**?*

Systems analysis and design is the process of creating and maintaining information systems. The classic methodology used in systems analysis and design to develop information systems is called the **systems development life cycle (SDLC)**.

*B.15 Describe the **systems development life cycle (SDLC)** model.*

There are different interpretations or conceptualizations of the SDLC, each of which uses a different number of steps. This book uses the same set of steps discussed by David Kroenke, and includes five steps or stages:

1. System definition
2. Requirements analysis
3. Component design
4. Implementation
5. System maintenance

Each step should result in one or more **deliverables**, such as documents, designs, prototypes, data models, database designs, Web screens, and so on.

The **system definition** step is a *process* that starts with the need for an information system to support a business process as its *input*, and produces a **project plan** as its *output*. During this process, we will need to:

- Define the information system project goals and scope
- Assess the feasibility of the project (cost, schedule, technical, organizational)
- Form the project team
- Plan the project (specify tasks, assign personnel, determine task dependencies, set schedules)

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

The **requirements analysis** step is a *process* that starts with the project plan as its *input*, and produces a set of **approved user requirements** as its *output*. During this process, we will need to:

- Conduct user interviews
- Evaluate existing systems
- Determine needed new forms/reports/queries
- Identify needed new application features and functions
- Consider security
- Create the data model
- Consider the five components of an information system—hardware, software, data, procedures, and people

The **component design** step is a *process* that starts with the approved user requirements as its *input*, and produces a final **system design** as its *output*. During this process, we will need to:

- Determine hardware specifications
- Determine program (software) specifications
- Create the database design
- Design business procedures
- Create job descriptions for business personnel

The **implementation** step is a *process* that starts with the final system design as its *input*, and produces a final **system** as its *output*. During this process, we will need to:

- Build system components
- Conduct component tests
- Integrate the components
- Conduct integrated component tests
- Convert to the new system

The **system maintenance** step is a *process* that starts with the implemented system as its *input*, and produces **an updated system** or **a request for system modification using the SDLC** as its *output*. During this process, we will need to:

- Update the system with patches, service packs, and new software releases
- Record and prioritize requests for system changes or enhancements.

*B.16 Define **project scope**.*

Project scope is the work that needs to be done to deliver a product, service, or result with the required functions and features.

*B.17 What is a **use case**? How well does Microsoft Visio 2016 support modeling use cases?*

Use cases are descriptions of the ways users will employ the features and functions of the new information system. A use case consists of a description of the roles users will play when utilizing the new system, together with descriptions of activities' scenarios.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Microsoft Visio 2016 does a good job of supporting modeling use cases, and has a **UML Use Case** diagram template with which to do so.

*B.18 What are **business rules**?*

Business rules are constraints on database activities. Generally, such rules arise from business policy and practice.

*B.19 What is a **user requirements document (URD)**? What purpose does it serve?*

A **user requirements document (URD)** is a requirements analysis deliverable that provides an approved set of user requirements. There is no set format for a URD.

Typically a URD may contain:

- a table of contents
- a revision history
- an introduction
- a general description of the project (including project assumptions and dependencies)
- a data model
- functional requirements
- non-functional requirements (speed and time, capacity, and reliability)
- project delivery requirements

The purpose of a URD is to formalize the project team's understanding of the users' requirements so that the users can review the document. Note that the data model for the database is presented as part of the URD.

*B.20 What is a **statement of work (SOW)**? What purpose does it serve?*

A **statement of work (SOW)** is a possible deliverable for the requirements analysis step of the SDLC. There is no set format for a SOW, and a SOW for an in-company project may be very different from a SOW between a consultant and client.

Typically a SOW may contain:

- A history of the problem or need that generated the project
- An identification of the client for the work
- An identification of who will do the work
- The scope of the work to be done
- The objectives of the work to be done
- Any constraints on the work to be done
- The location of the work (where the work will be done)
- A set of tasks with an associated timeline:
  - An outline of the tasks that will make up the work to be done
  - The time period for the work (start date, finish date, details about how many hours may be worked, etc.)
  - A deliverables schedule
- Criteria for determining whether the project has been successfully completed

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

- A payment schedule
- Signature blocks to record acceptance of the SOW by all parties

ANSWERS TO PROJECT QUESTIONS

B.21 Review the information about Wedgewood Pacific (WP) and the WP database that has been presented in the Project Questions in Chapters 1 and 2.

A. The database is part of an information system to support business operations at Wedgewood Pacific. What specific business process at Wedgewood Pacific Corporation do you think this information system is being developed to support? How does the WP database support the business process?

The WP database is part of an information system being created to support project management at WP. This primarily falls under Porter's support activity of **firm infrastructure management**, although there is some overlap with **human resources management** because of the fact that people are assigned to work on projects.

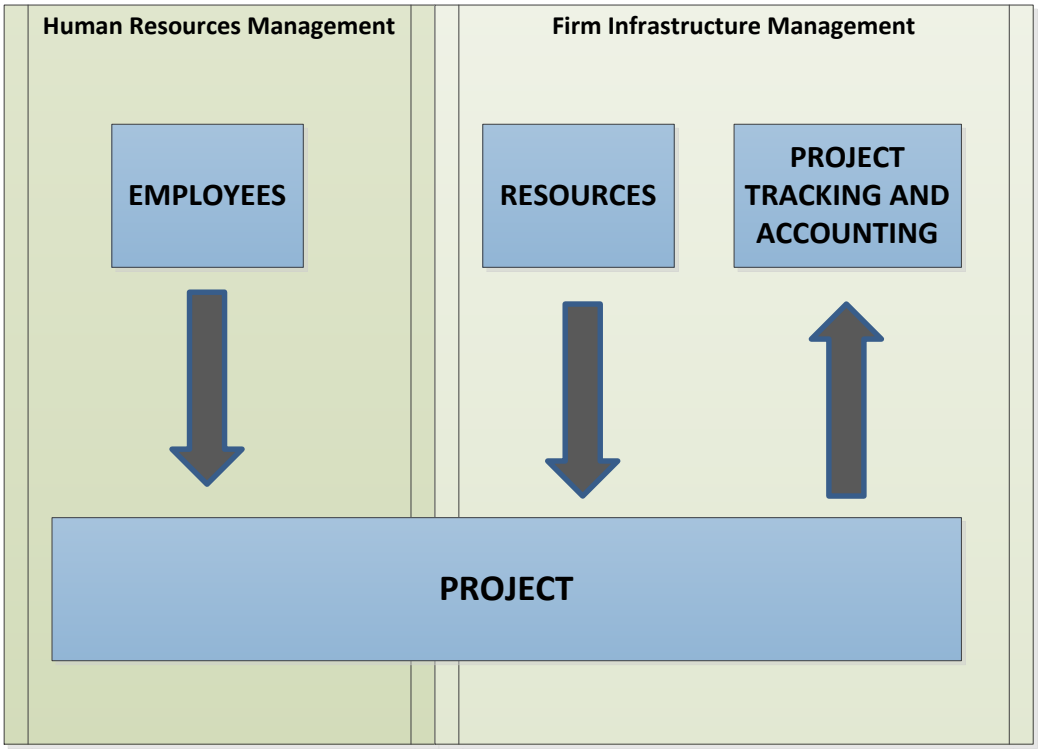
The WP database keeps track of the WP projects, the resources allocated to those projects (employees and MaxHours), project expenses (HoursWorked) and progress on those projects (StartDate and EndDate).

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

B. How does this business process overlay Michael Porter's set of primary and support business activities? Draw a diagram to illustrate this overlap.

The Sales process overlays two of Porter's support activities, but none of Porter's primary activities. The support activities are firm infrastructure management and human resources management. While primary activities have a natural process flow, secondary activities do not. This can make drawing a diagram a bit more difficult, but we can picture this process as:



Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

C. Based on the actual Wedgewood Pacific database as it has been created through the Chapter 2 Exercises, what do you think are the approved requirements for the project? What are the business rules (either stated or implied)?

Approved requirements would include:

	Functional Requirements
R-1	WP human resources management must be able to enter, modify, and delete department data.
R-2	WP human resources management must be able to enter, modify, and delete employee data.
R-3	WP project managers must be able to enter, modify, and delete project data.
R-4	WP human resources management must be able to enter, modify, and delete employee assignment data.
R-5	WP human resources management must be able to view and print department and employee reports.
R-6	WP project managers must be able to view and print project reports.

Business rules would include:

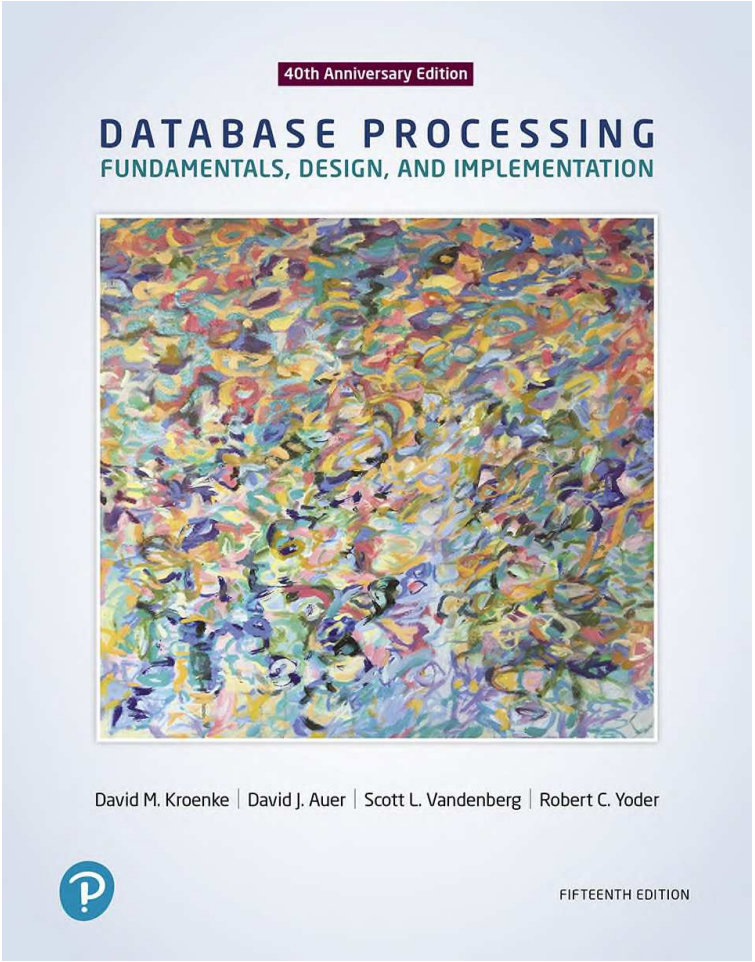
	Business Rules
BR-1	A WP project must be sponsored by a WP department.
BR-2	A WP employee may be assigned to one or more projects.
BR-3	The sum of hours worked on a project by WP employees may not exceed the total hours allocated for that project.

Copyright © 2019 Pearson Education, Inc.

Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Database Processing: Fundamentals, Design, and Implementation

15th Edition



Chapter Two

Part 1

Introduction to Structured Query Language

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Learning Objectives (1 of 2)

- **2.1** To understand the use of extracted datasets in business intelligence (BI) systems
- **2.2** To understand the use of ad-hoc queries in business intelligence (BI) systems
- **2.3** To understand the history and significance of Structured Query Language (SQL)
- **2.4** To understand the SQL SELECT/FROM/WHERE framework as the basis for database queries
- **2.5** To create SQL queries to retrieve data from a single table
- **2.6** To create SQL queries that use the SQL SELECT, FROM, WHERE, ORDER BY, GROUP BY, and HAVING clauses
- **2.7** To create SQL queries that use the SQL DISTINCT, TOP, and TOP PERCENT keywords
- **2.8** To create SQL queries that use the SQL comparison operators, including BETWEEN, LIKE, IN, and IS NULL

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Learning Objectives (2 of 2)

- **2.9** To create SQL queries that use the SQL logical operators, including AND, OR, and NOT
- **2.10** 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
- **2.11** To create SQL queries that retrieve data from a single table while restricting the data based upon data in another table (subquery)
- **2.12** To create SQL queries that retrieve data from multiple tables using the SQL join and JOIN ON operations
- **2.13** To create SQL queries on recursive relationships
- **2.14** To create SQL queries that retrieve data from multiple tables using the SQL OUTER JOIN operation
- **2.15** To create SQL queries that retrieve data from multiple tables using SQL set operators UNION, INTERSECT, and EXCEPT

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Chapter 2 Part 1 Coverage

- This Microsoft PowerPoint Slide Show covers **single table queries** and the **first ten chapter objectives.**

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Business Intelligence (BI) Systems

- **Business intelligence (BI)** systems are information systems that assist managers and other professionals:
  - Assessment
  - Analysis
  - Planning
  - Control

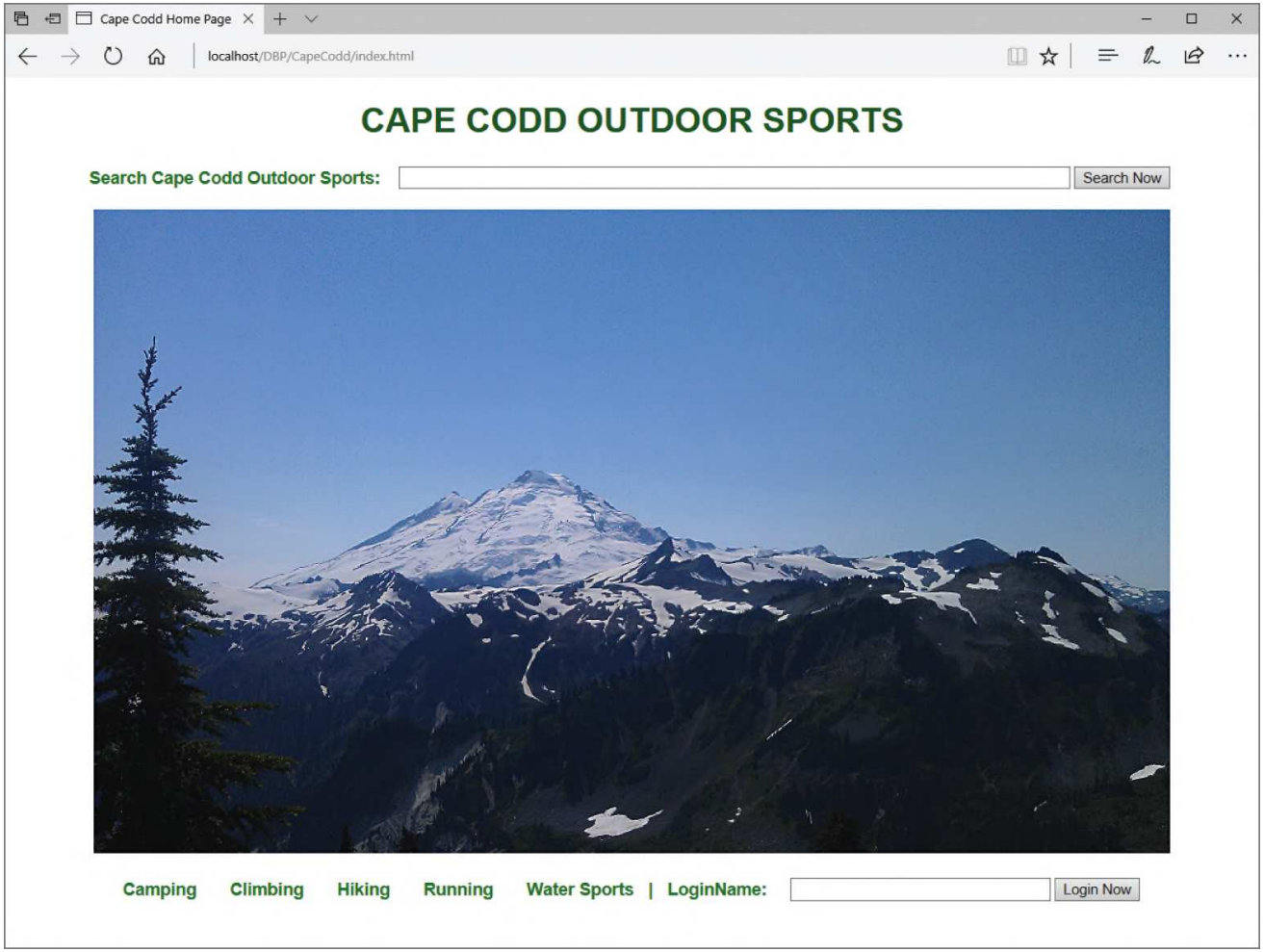
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Ad-Hoc Queries

- **Ad-hoc queries**:
  - Questions that can be answered using database data
  - Example: “How many customers in Portland, Oregon, bought our green baseball cap?”
  - Created by the user as needed, instead of programmed into an application
  - Common in business

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-1
Cape Codd Outdoor Sports



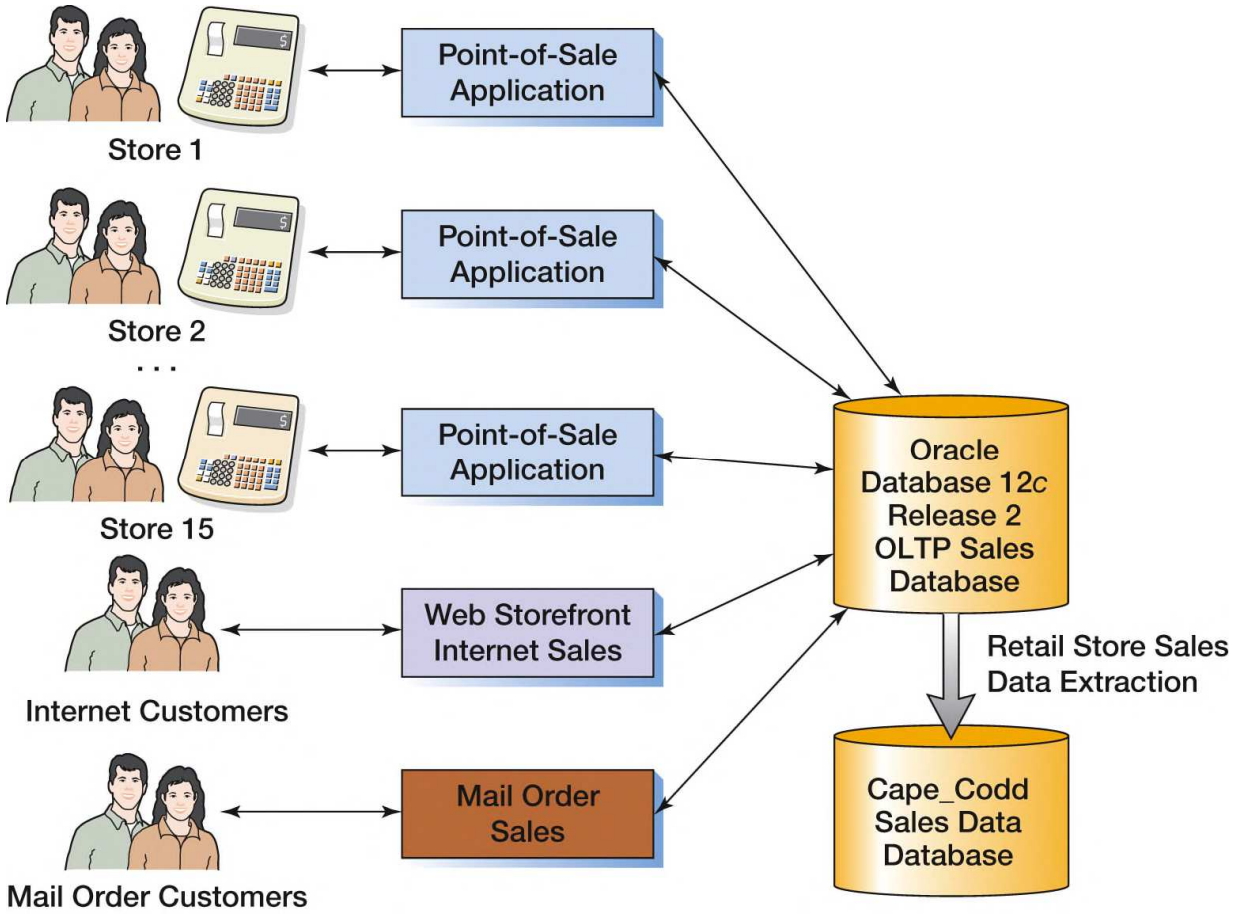
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Cape Codd Outdoor Sports Background

- Cape Codd Outdoor Sports is a fictitious company based on an actual outdoor retail equipment vendor.
- Cape Codd Outdoor Sports:
  - Has 15 retail stores in the United States and Canada
  - Has an online Internet store
  - Has a (postal) mail order department
- All retail sales are recorded in an Oracle Database 12*c* Release 2 DBMS.

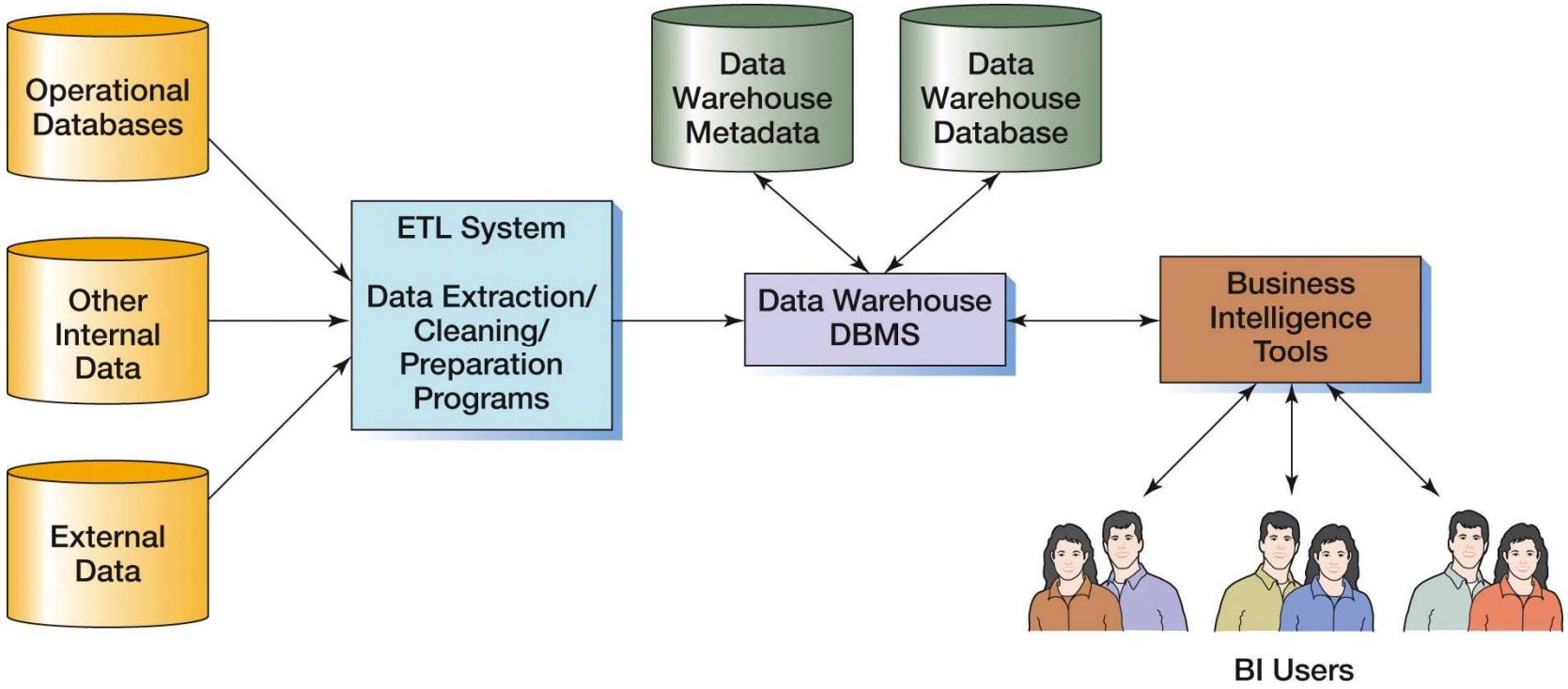
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Figure 2-2
Cape Codd Retail Sales Data Extraction Process



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-3
Components of a Data Warehouse



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Cape Codd Outdoor Sports

Retail Sales and Catalog Context Data Extraction 1

- The Cape Codd marketing department needs an analysis of:
  - In-store sales
  - Catalog content
- The entire database is not needed for this, only an **extraction** of retail sales data and catalog content.
- The data is extracted by the IS department from the **operational database** into a **separate, offline database** for use by the marketing department.

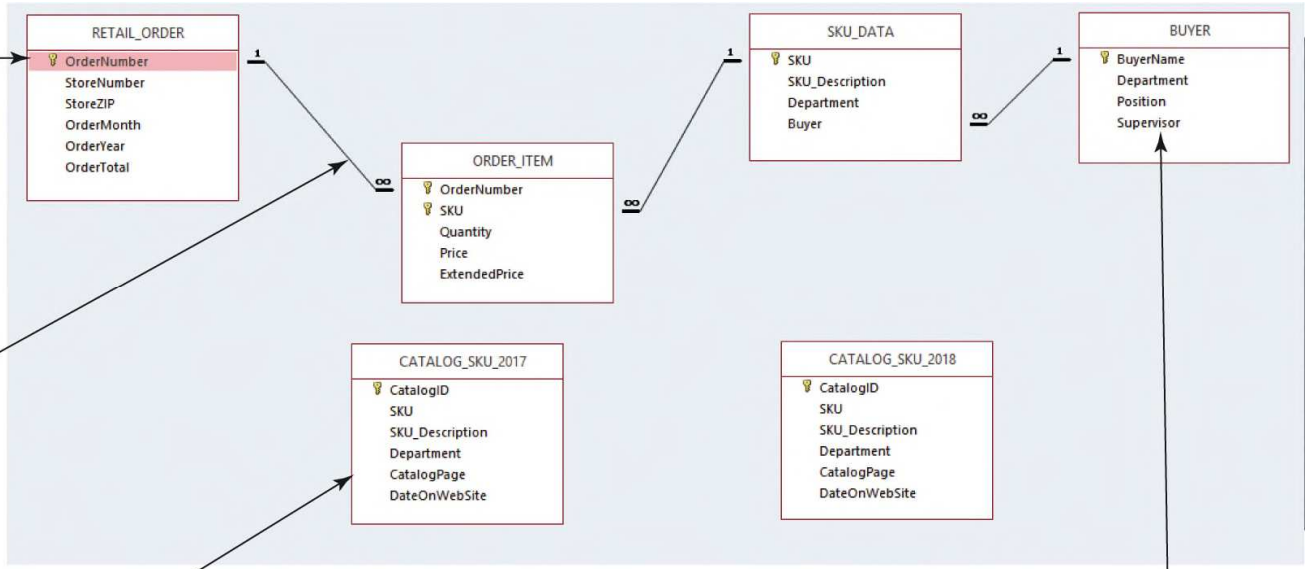
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-4
Extracted Retail Sales Data Database Tables

The RETAIL_ORDER table—the key symbol shows the primary key

The relationship between RETAIL_ORDER and ORDER_ITEM—the number 1 and the infinity symbol indicate that one retail order may be linked to many order items by OrderNumber

The CATALOG_SKU_2017 and CATALOG_SKU_2018 tables are *not* linked to other tables in the database



The recursive relationship from Supervisor to BuyerName within BUYER is *not* shown

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-5 (1 of 2)
Cape Codd Extracted Retail Sales Data Format (1 of 2)

Table	Column	Data Type
RETAIL_ORDER	OrderNumber	Integer
	StoreNumber	Integer
	StoreZIP	Character (9)
	OrderMonth	Character (12)
	OrderYear	Integer
	OrderTotal	Currency
	OrderNumber	Integer
ORDER_ITEM	SKU	Integer
	Quantity	Integer
	Price	Currency
	ExtendedPrice	Currency

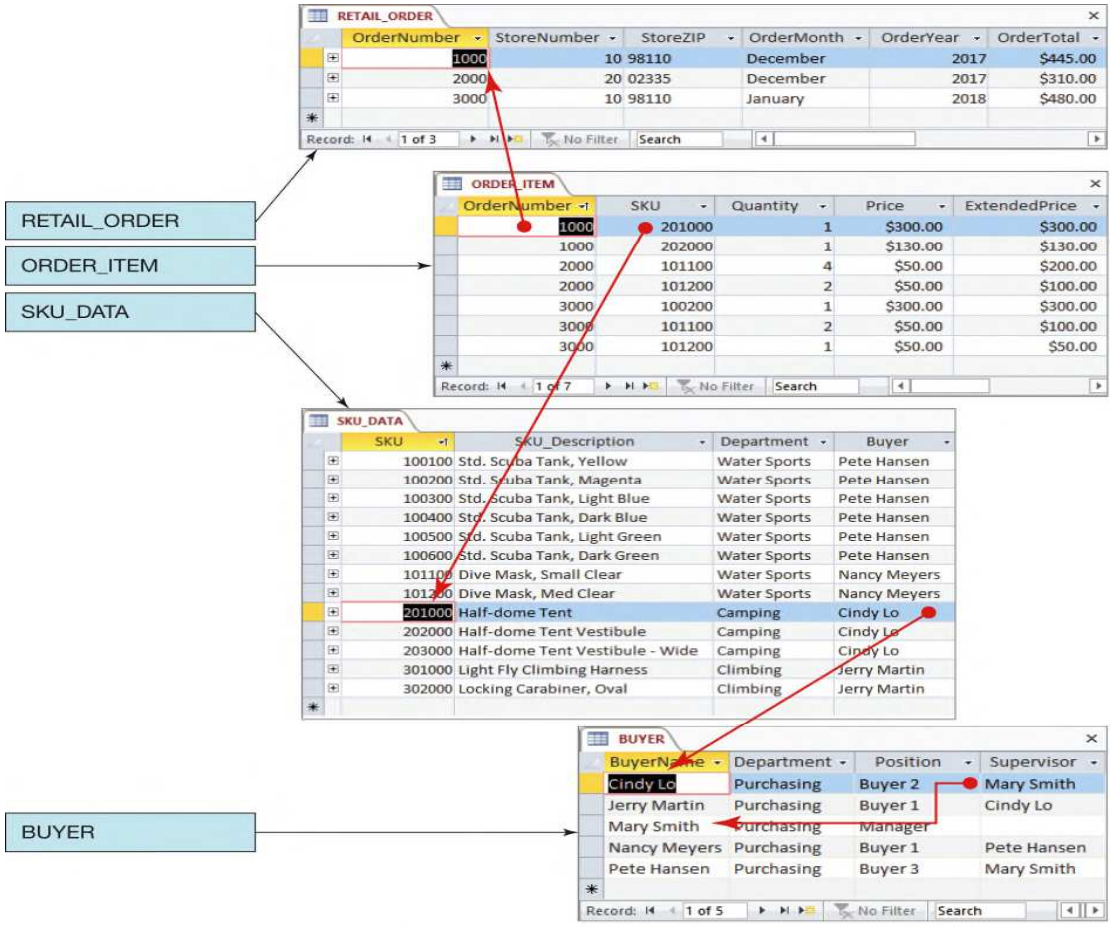
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-5 (2 of 2)
Cape Codd Extracted Retail Sales Data Format (2 of 2)

Table	Column	Data Type
SKU_DATA	SKU	Integer
	SKU_Description	Character (35)
	Department	Character (30)
	Buyer	Character (35)
BUYER	BuyerName	Character (35)
	Department	Character (30)
	Position	Character (10)
	Supervisor	Character (35)
CATALOG_SKU_20##	CatalogID	Integer
	SKU	Integer
	SKU_Description	Character (35)
	Department	Character (30)
	CatalogPage	Integer
	DateOnWebSite	Date

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

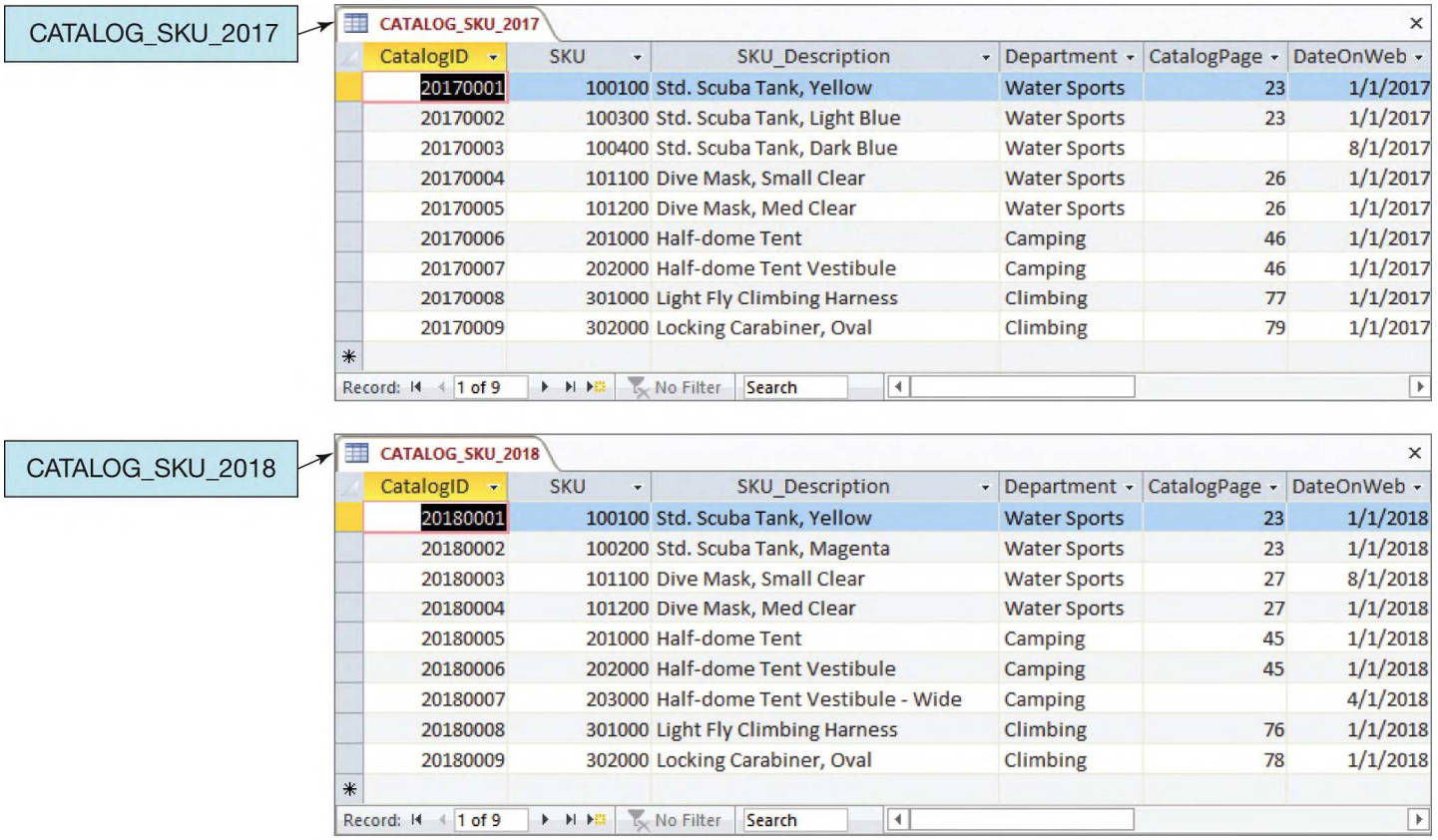
Figure 2-6 (1 of 2)
Extracted Retail Sales Database (1 of 2)



(a) The Linked RETAIL_ORDER, ORDER_ITEM, SKU_DATA, and BUYER Tables

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-6 (2 of 2)
Extracted Retail Sales Database (2 of 2)



CATALOG_SKU_2017

CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWeb
20170001	100100	Std. Scuba Tank, Yellow	Water Sports	23	1/1/2017
20170002	100300	Std. Scuba Tank, Light Blue	Water Sports	23	1/1/2017
20170003	100400	Std. Scuba Tank, Dark Blue	Water Sports		8/1/2017
20170004	101100	Dive Mask, Small Clear	Water Sports	26	1/1/2017
20170005	101200	Dive Mask, Med Clear	Water Sports	26	1/1/2017
20170006	201000	Half-dome Tent	Camping	46	1/1/2017
20170007	202000	Half-dome Tent Vestibule	Camping	46	1/1/2017
20170008	301000	Light Fly Climbing Harness	Climbing	77	1/1/2017
20170009	302000	Locking Carabiner, Oval	Climbing	79	1/1/2017

Record: 1 of 9 No Filter Search

CATALOG_SKU_2018

CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWeb
20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	1/1/2018
20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	1/1/2018
20180003	101100	Dive Mask, Small Clear	Water Sports	27	8/1/2018
20180004	101200	Dive Mask, Med Clear	Water Sports	27	1/1/2018
20180005	201000	Half-dome Tent	Camping	45	1/1/2018
20180006	202000	Half-dome Tent Vestibule	Camping	45	1/1/2018
20180007	203000	Half-dome Tent Vestibule - Wide	Camping		4/1/2018
20180008	301000	Light Fly Climbing Harness	Climbing	76	1/1/2018
20180009	302000	Locking Carabiner, Oval	Climbing	78	1/1/2018

Record: 1 of 9 No Filter Search

(b) The Non-Linked CATALOG_SKU_2017 and CATALOG_SKU_2018 Tables

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Cape Codd Database Available Online (1 of 2)

- Versions of the complete Cape Codd database are available in the downloadable Student Files available at:

  http://www.pearsonhighered.com/kroenke/

- These include versions for:
  - Microsoft Access 2016
  - Microsoft SQL Server 2017
  - Oracle Database XE
  - MySQL 5.7
- **We recommend you actually run all material in a live database!**

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Cape Codd Database Available Online (2 of 2)

- To complete setting up the Cape Codd database, set the referenced materials:
  - For Microsoft SQL Server 2014:
    - See Online Chapter 10A
  - For Oracle Database 12c and Oracle Database XE:
    - See Online Chapter 10B
  - For MySQL 5.6
    - See Online Chapter 10C
- Online chapters 10A, 10B, and 10C are available for download at:

  http://www.pearsonhighered.com/kroenke/

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Structured Query Language

- **Structured Query Language (SQL)** was developed by the IBM Corporation in the late 1970's.
- SQL was endorsed as a U.S. national standard by the American National Standards Institute (ANSI) in 1992 [**SQL-92**].
- Newer versions exist, such as **SQL:2008**, **SQL:2011**, and **SQL:2016** and they incorporate new features including XML, JSON, and some object-oriented concepts. Some of these features are discussed in this book.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL As a Data Sublanguage

- SQL is not a full-featured **programming language**.
  - C, C#
- SQL is a **data sublanguage** for creating and processing database data and metadata.
- SQL is ubiquitous in enterprise-class DBMS products.
- SQL programming is a critical skill.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL Categories

- SQL statements can be divided into five categories:
  - **Data definition language (DDL)**
  - **Data manipulation language (DML)** statements
  - **SQL/Persistent Stored Modules (SQL/PSM)** statements
  - **Transaction control language (TCL)** statements
  - **Data control language (DCL)** statements

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL DDL

- **Data definition language (DDL)** statements
  - Used for creating tables, relationships, and other structures
  - Covered in Chapter 7

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL DML

- **Data manipulation language (DML)** statements
  - Used for:
    - Queries – SQL **SELECT** statement
    - Inserting data – SQL **INSERT** statement
    - Modifying data – SQL **UPDATE** statement
    - Deleting data – SQL **DELETE** statement
  - Covered in this chapter (Chapter 2)

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL SQL/PSM

- **SQL/Persistent Stored Modules (SQL/PSM)** statements
  - Add procedural programming capabilities
    - Variables
    - Control-of-flow statements
  - Covered in Chapters:
    - 7 (general introduction)
    - 10A (SQL Server 2014)
    - 10B (Oracle Database)
    - 10C (MySQL 5.6)

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL TCL

- **Transaction control language (TCL)** statements
  - Used to mark transaction boundaries and control transaction behavior
  - Covered in Chapters:
    - 9 (general introduction)
    - 10A (SQL Server 2017)
    - 10B (Oracle Database XE)
    - 10C (MySQL 5.7)

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL DCL

- **Data control language (DCL)** statements
  - Used to grant (or revoke) database permissions to (from) users and groups
  - Covered in Chapters:
    - 9 (general introduction)
    - 10A (SQL Server 2014)
    - 10B (Oracle Database)
    - 10C (MySQL 5.6)

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

The SQL SELECT Statement

- The fundamental framework for an SQL query is the **SQL SELECT statement**.
  - **SELECT {ColumnName(s)}**
  - **FROM {TableName(s)}**
  - **WHERE {Condition(s)}**
- All SQL statements end with a **semicolon(;)**.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Specific Columns from a Single Table

```
SELECT    SKU, SKU_Description, Department, Buyer
FROM      SKU_DATA;
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	100300	Std. Scuba Tank, Light Blue	Water Sports	Pete Hansen
4	100400	Std. Scuba Tank, Dark Blue	Water Sports	Pete Hansen
5	100500	Std. Scuba Tank, Light Green	Water Sports	Pete Hansen
6	100600	Std. Scuba Tank, Dark Green	Water Sports	Pete Hansen
7	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
8	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
9	201000	Half-dome Tent	Camping	Cindy Lo
10	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
11	203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo
12	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
13	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Selecting All Columns:
The SQL Asterisk (*) Wildcard Character

```
SELECT      *
FROM        SKU_DATA;
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	100300	Std. Scuba Tank, Light Blue	Water Sports	Pete Hansen
4	100400	Std. Scuba Tank, Dark Blue	Water Sports	Pete Hansen
5	100500	Std. Scuba Tank, Light Green	Water Sports	Pete Hansen
6	100600	Std. Scuba Tank, Dark Green	Water Sports	Pete Hansen
7	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
8	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
9	201000	Half-dome Tent	Camping	Cindy Lo
10	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
11	203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo
12	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
13	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Specifying Column Order I

```
SELECT     Department, Buyer
FROM       SKU_DATA;
```

	Department	Buyer
1	Water Sports	Pete Hansen
2	Water Sports	Pete Hansen
3	Water Sports	Pete Hansen
4	Water Sports	Pete Hansen
5	Water Sports	Pete Hansen
6	Water Sports	Pete Hansen
7	Water Sports	Nancy Meyers
8	Water Sports	Nancy Meyers
9	Camping	Cindy Lo
10	Camping	Cindy Lo
11	Camping	Cindy Lo
12	Climbing	Jerry Martin
13	Climbing	Jerry Martin

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Specifying Column Order II

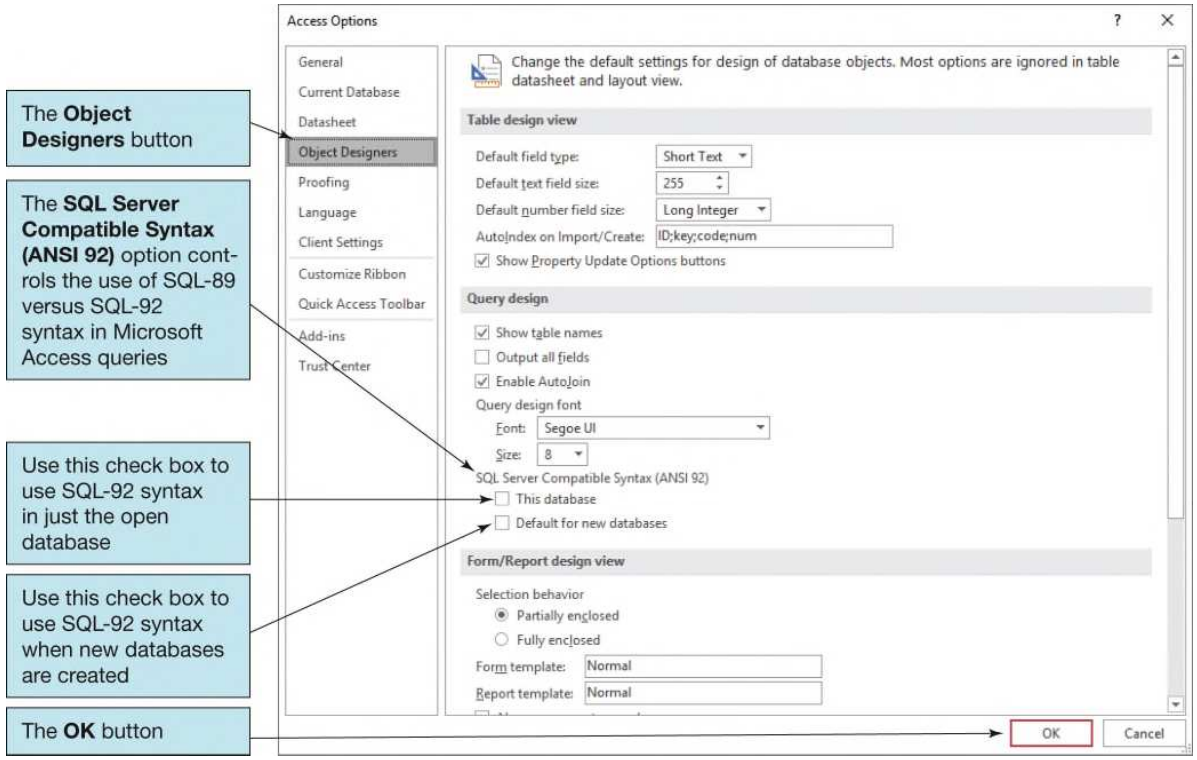
```
SELECT    Buyer, Department
FROM      SKU_DATA;
```

	Buyer	Department
1	Pete Hansen	Water Sports
2	Pete Hansen	Water Sports
3	Pete Hansen	Water Sports
4	Pete Hansen	Water Sports
5	Pete Hansen	Water Sports
6	Pete Hansen	Water Sports
7	Nancy Meyers	Water Sports
8	Nancy Meyers	Water Sports
9	Cindy Lo	Camping
10	Cindy Lo	Camping
11	Cindy Lo	Camping
12	Jerry Martin	Climbing
13	Jerry Martin	Climbing

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

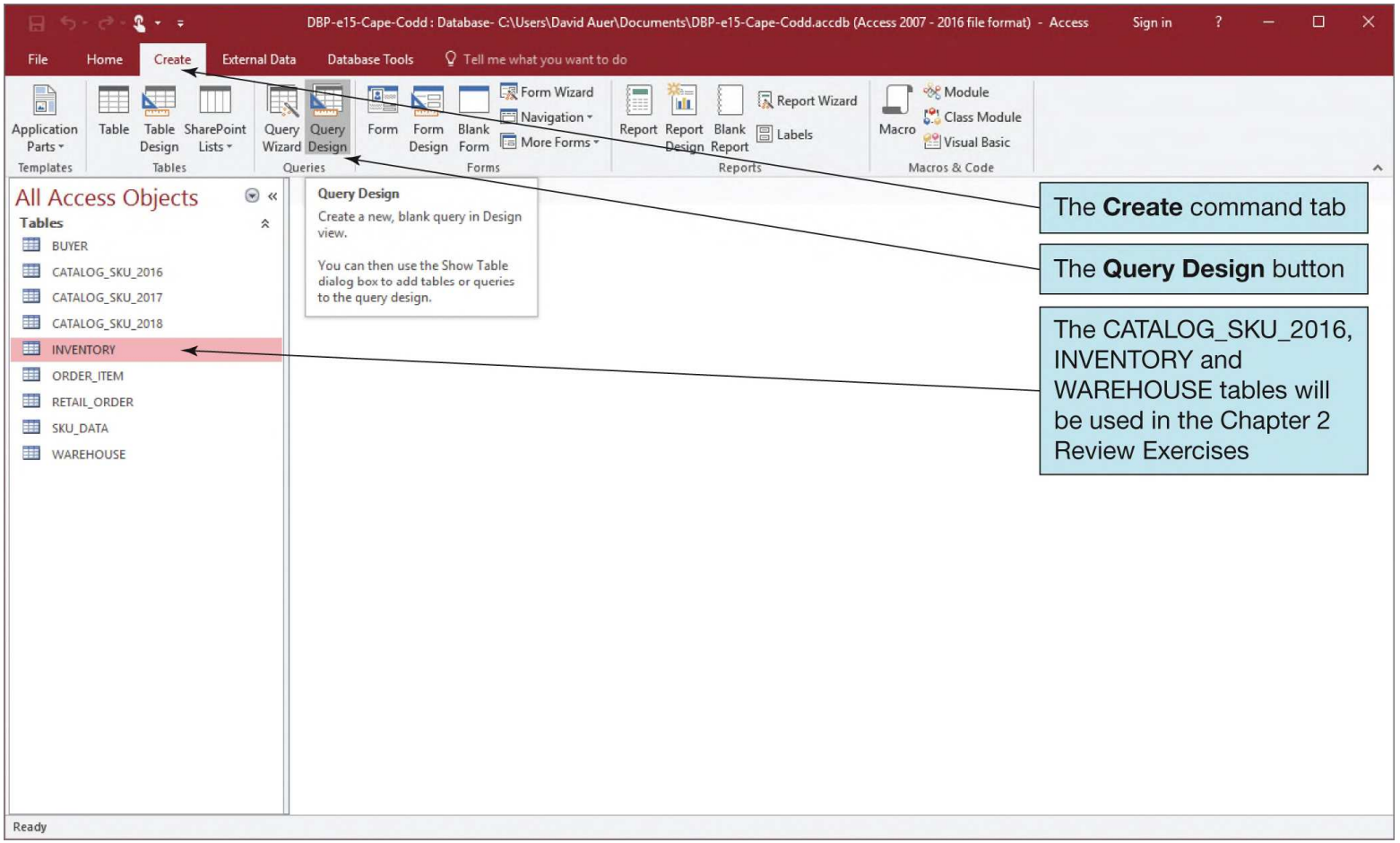
Figure 2-7
MS Access 2016 Options Object Designers Page

The **SQL Server Compatible Syntax (ANSI 92)** options control which version of SQL is used in a Microsoft Access 2016 database. If you check the **This database** check box, you will use SQL-92 syntax in the current database. Or you can check the **Default for new databases** check box to make SQL-92 syntax the default for all new databases you create.



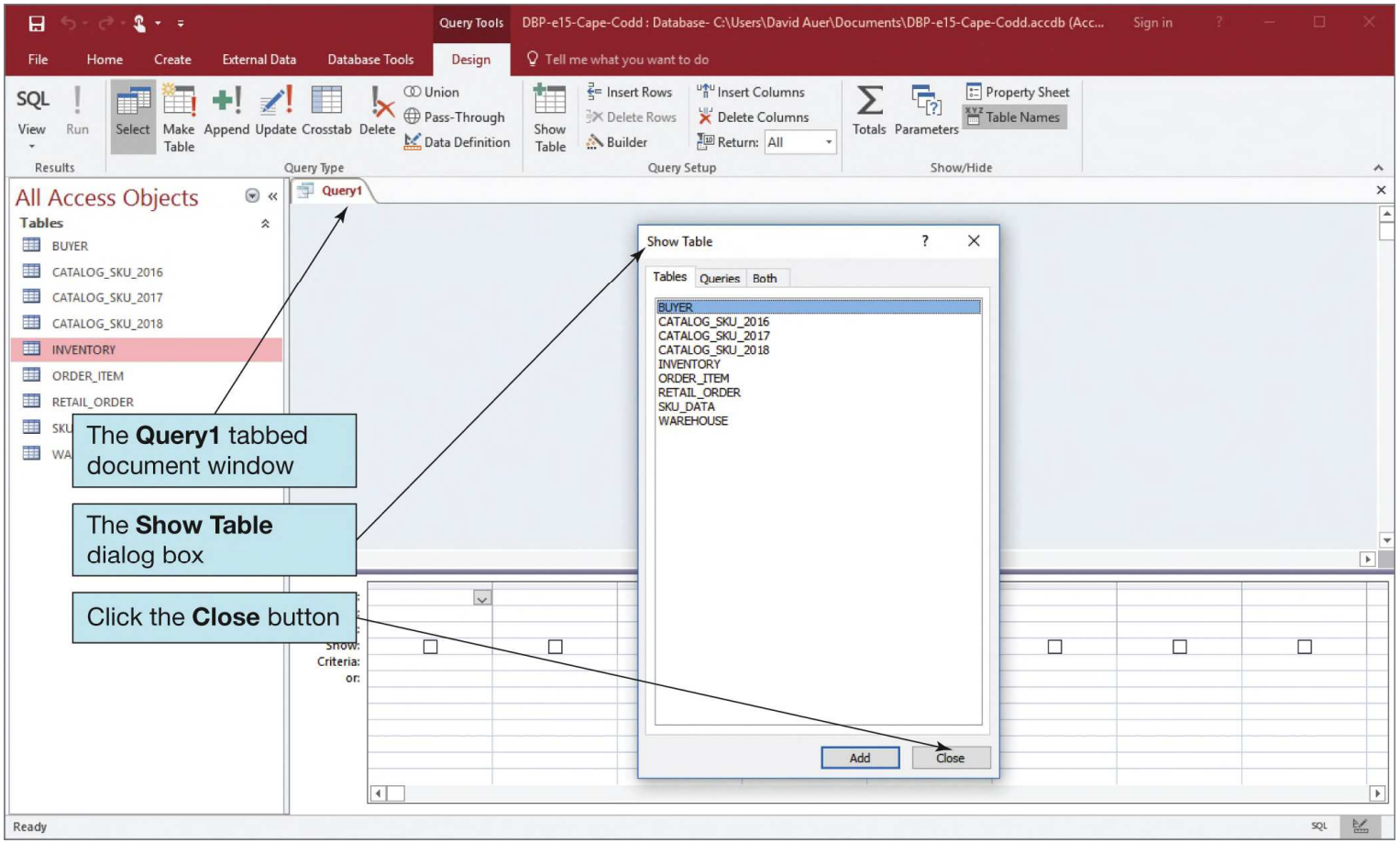
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Figure 2-9
The Create Command Tab



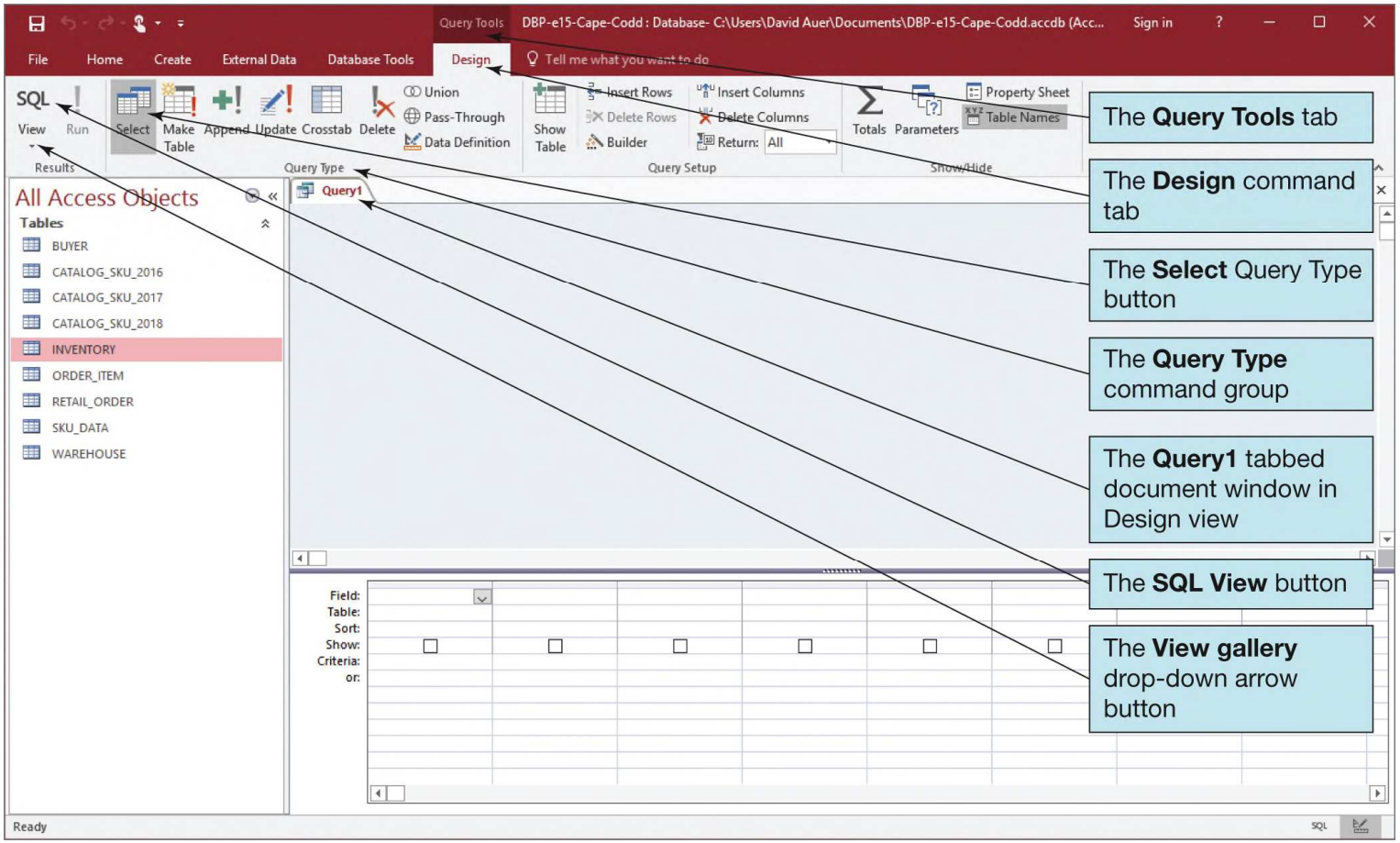
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-10
The Show Table Dialog Box



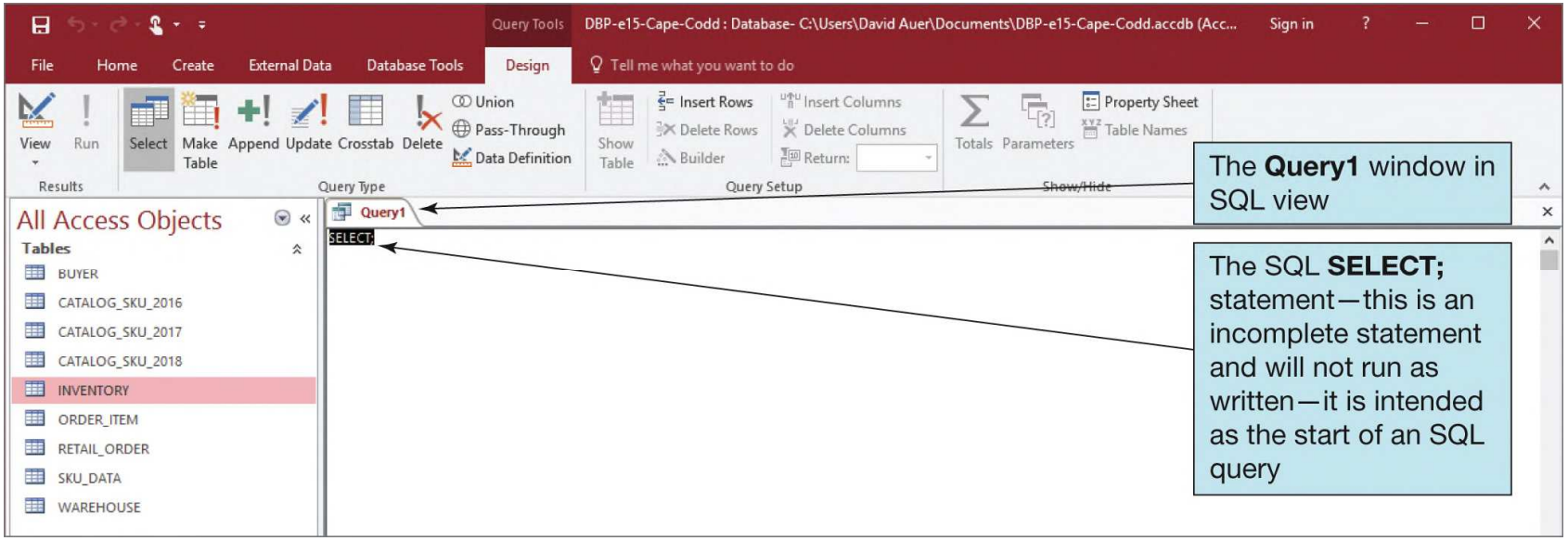
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-11
The Query Tools Contextual Command Tab



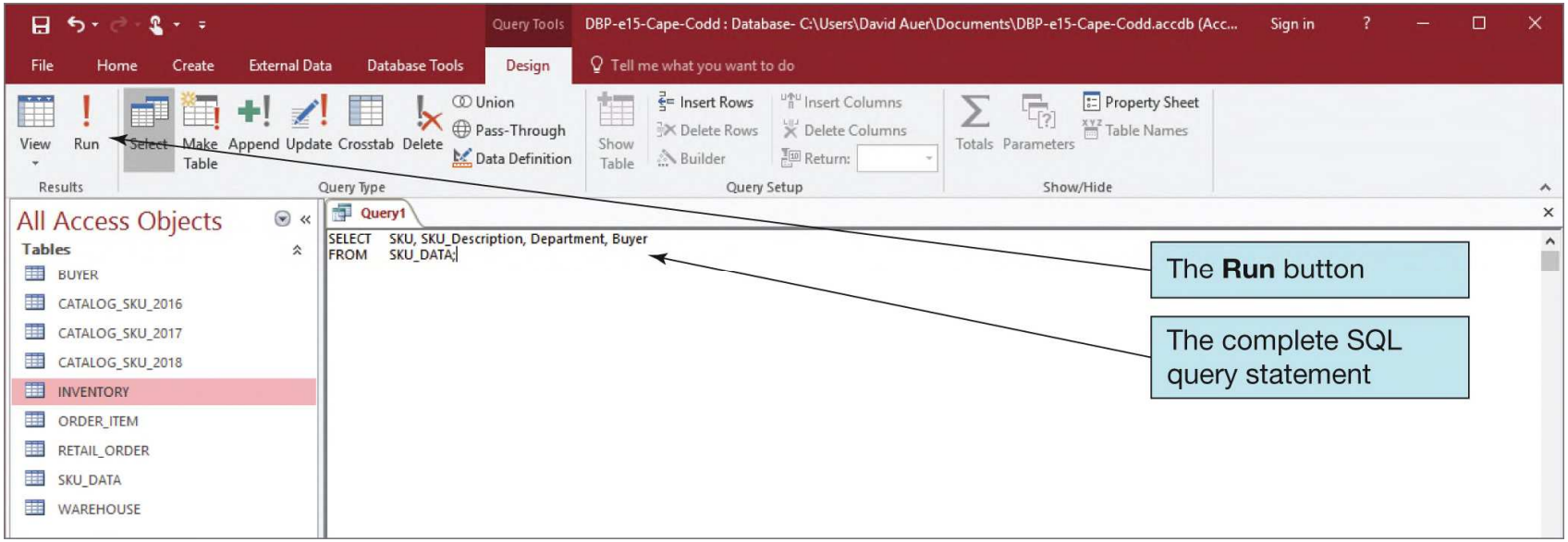
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-12
The Query 1 Window in SQL View



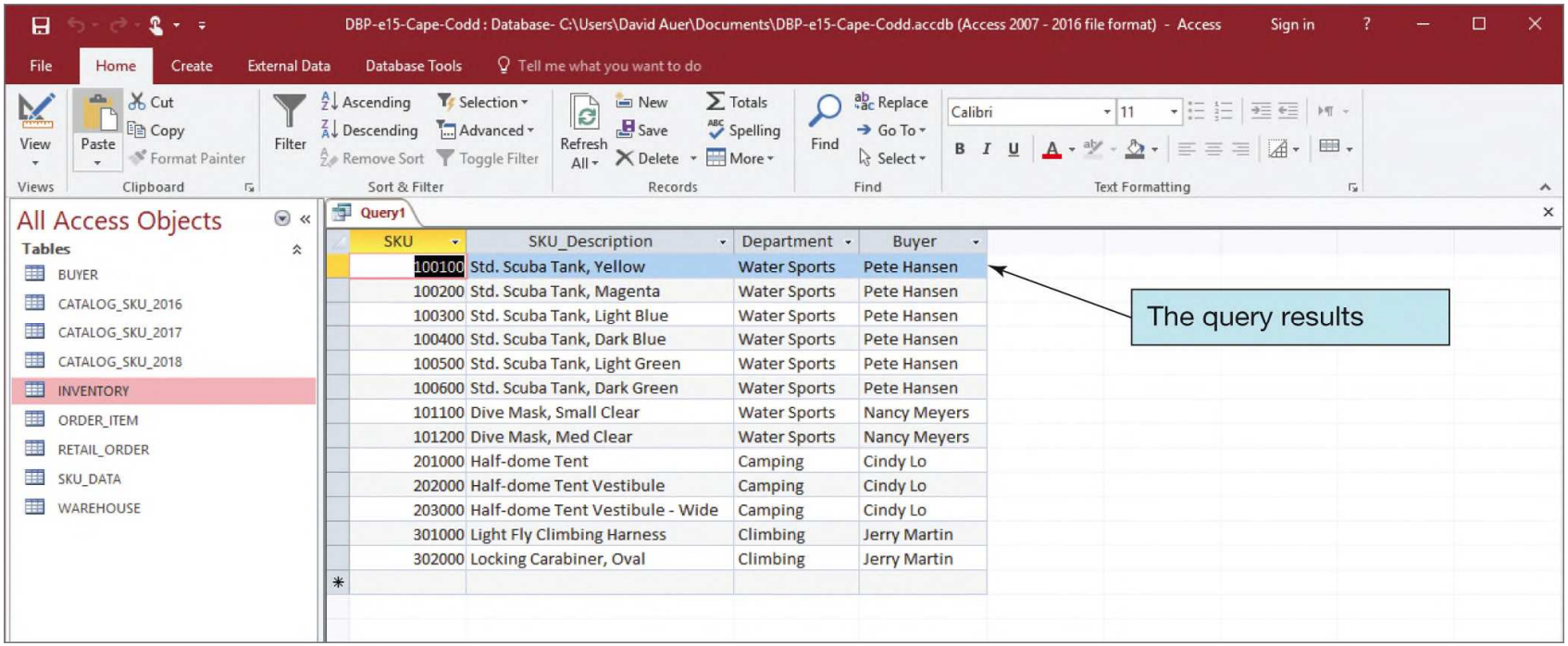
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-13
The SQL Query



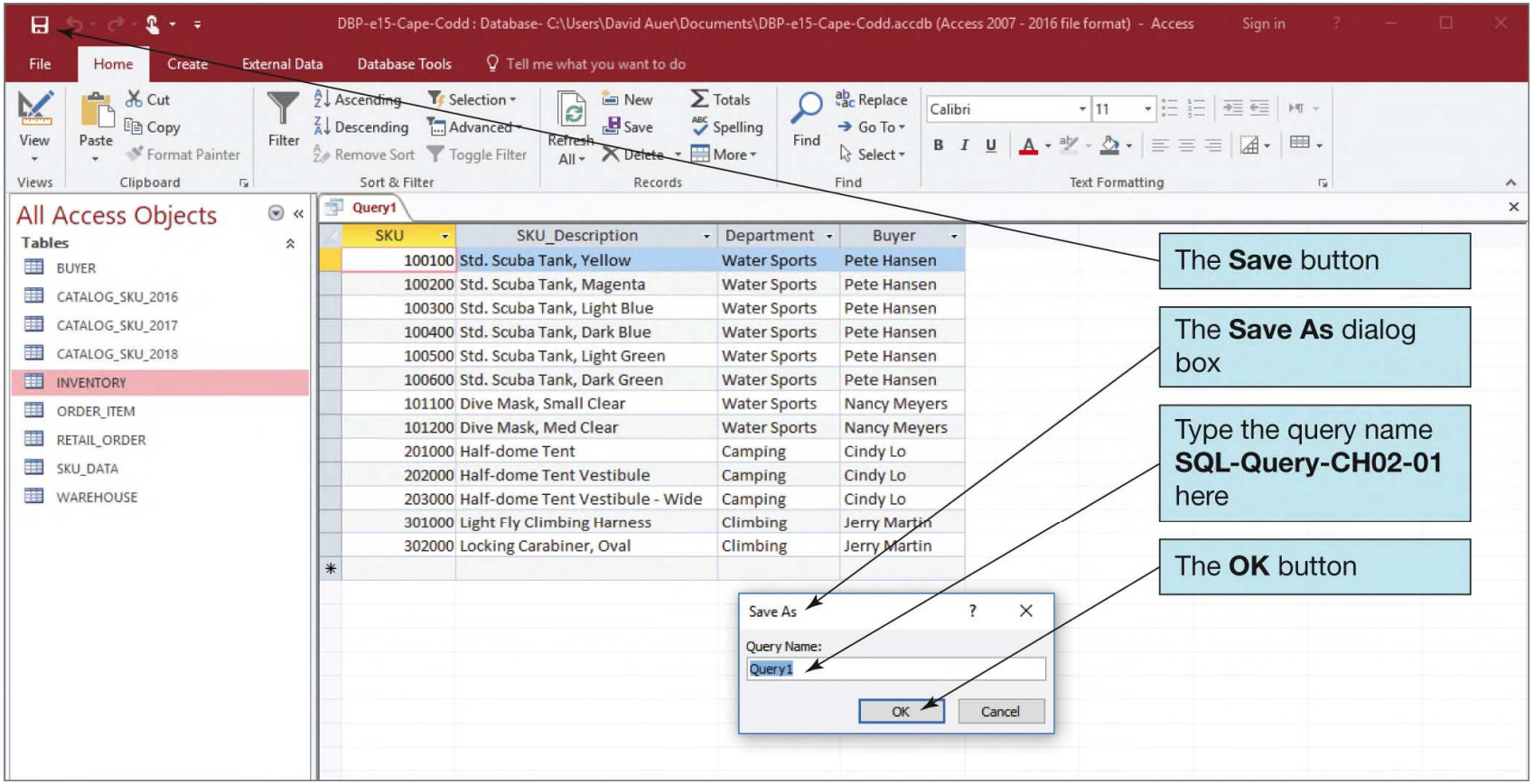
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-14
The SQL Query Results



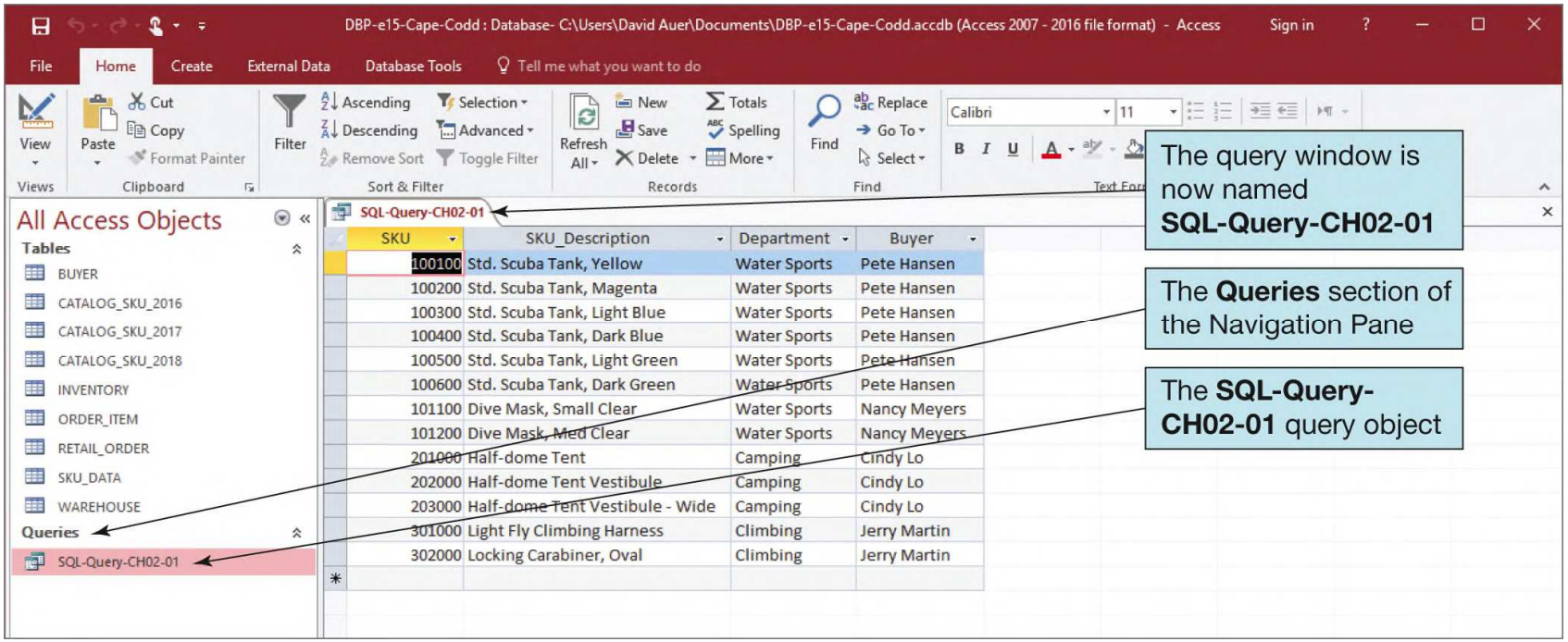
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-15
The Save As Dialog Box



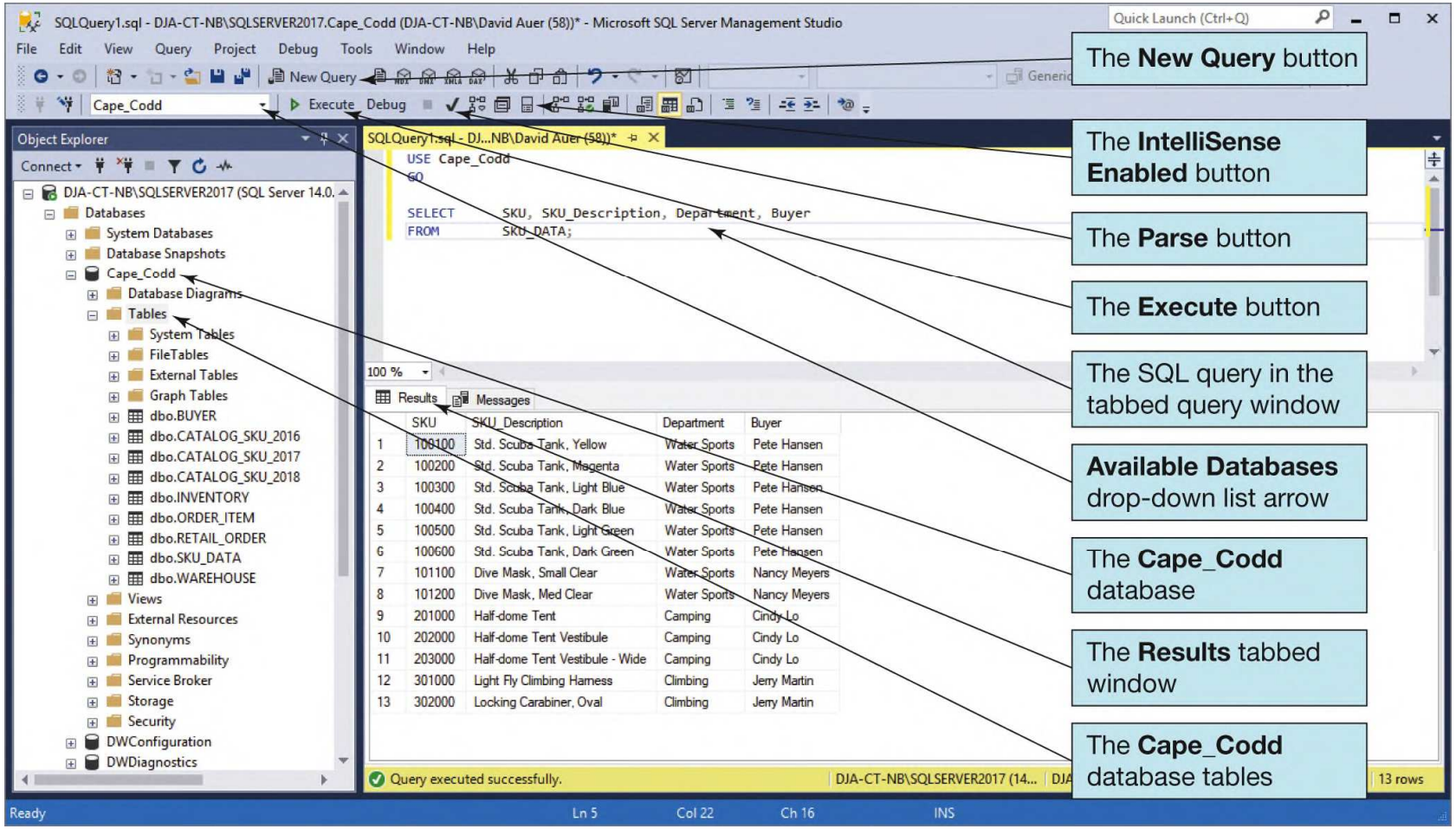
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-16
The Named and Saved Query



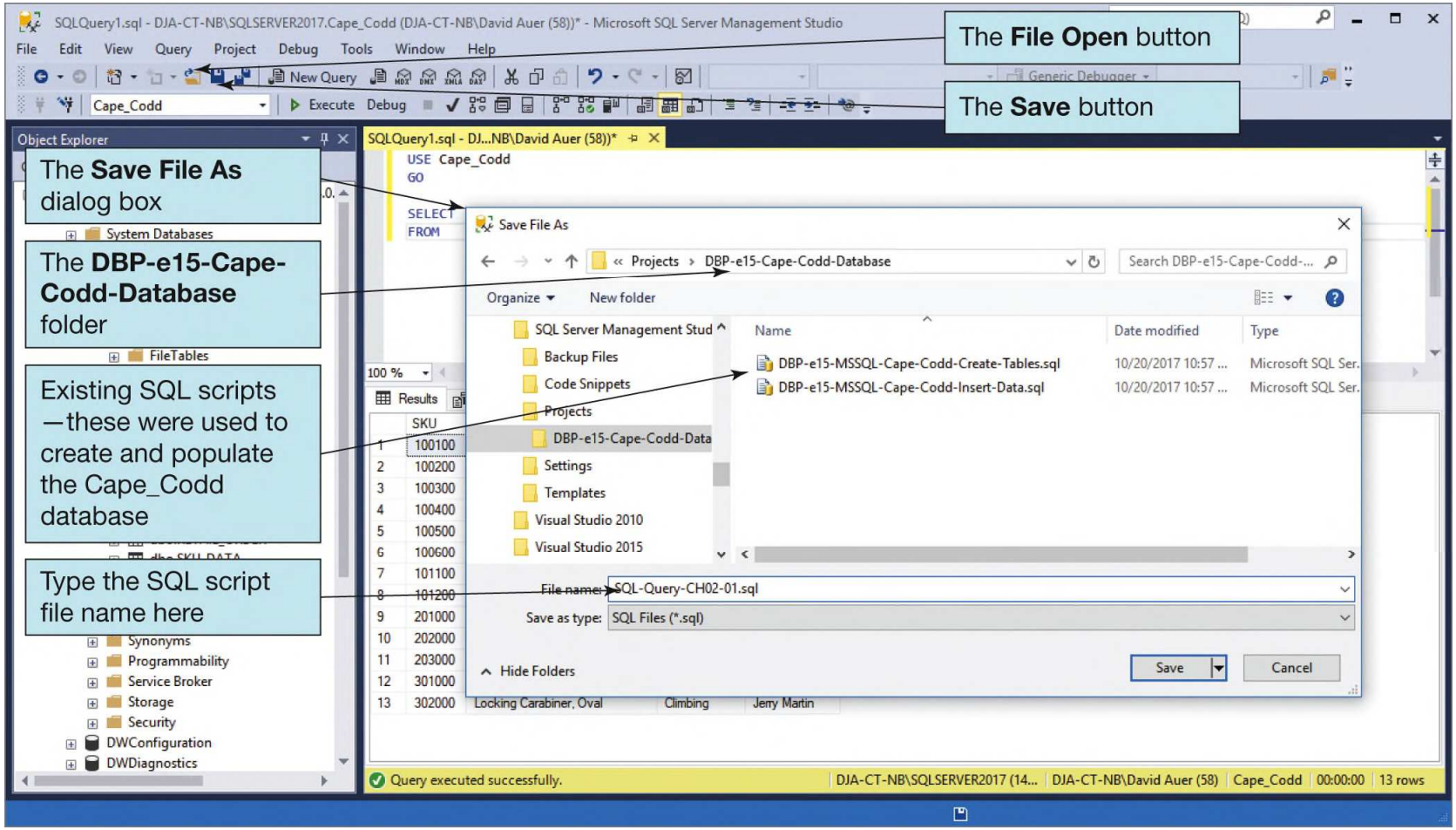
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-17
Running Query in SQL Server Mgmt Studio



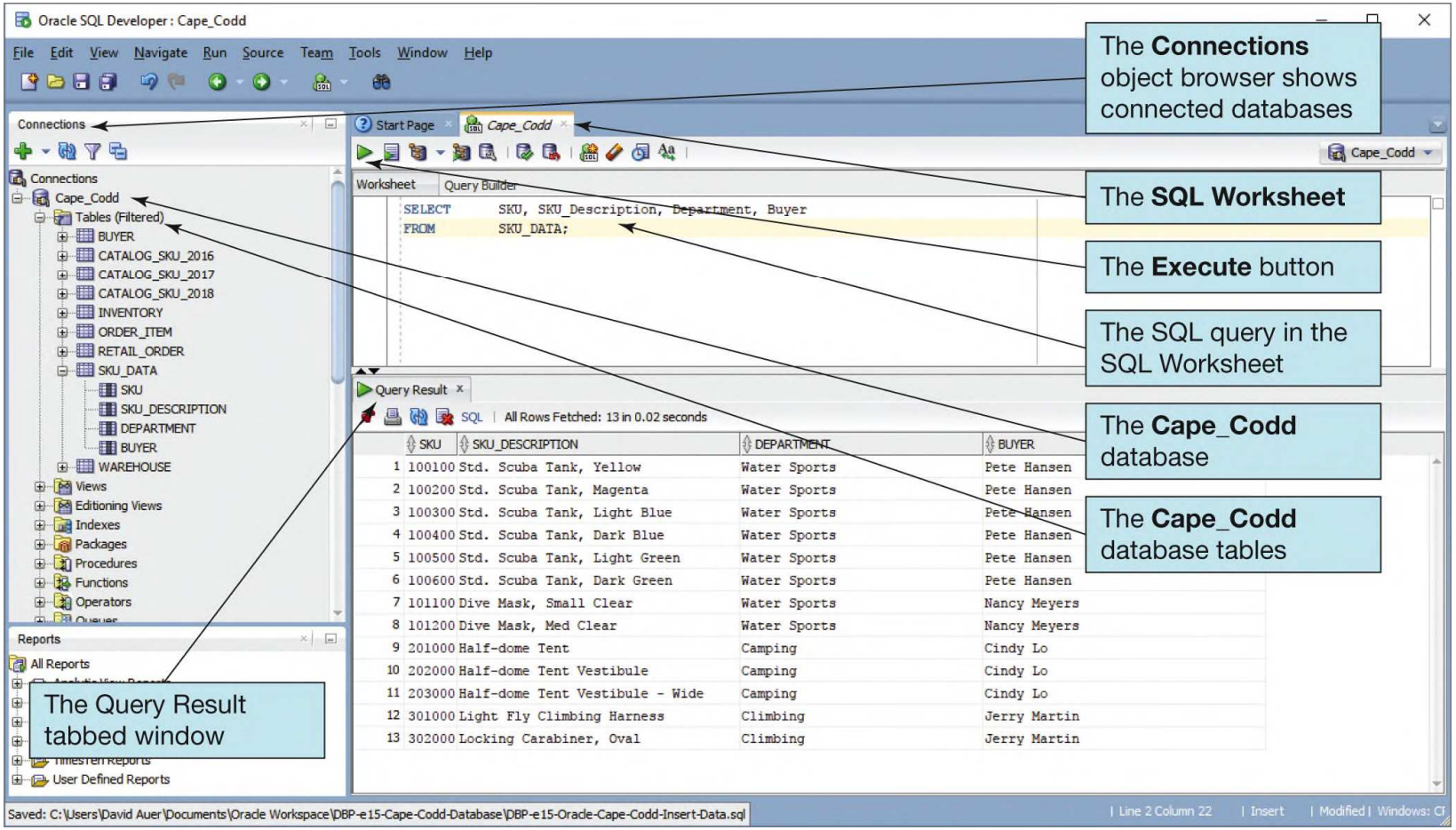
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-18
Saving a Query as an SQL Script in Mgmt Studio



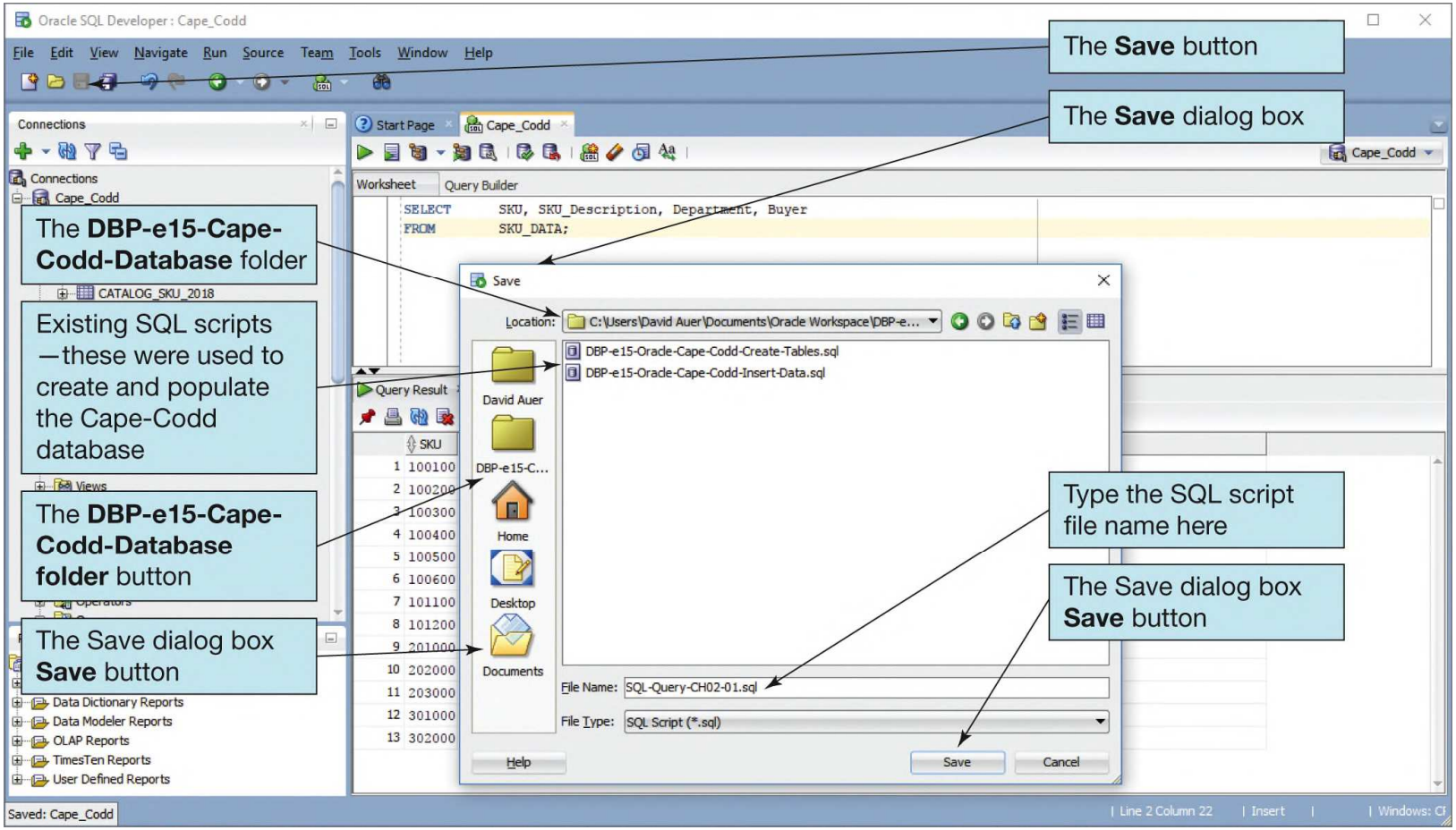
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-19
Running an SQL Query in Oracle SQL Developer



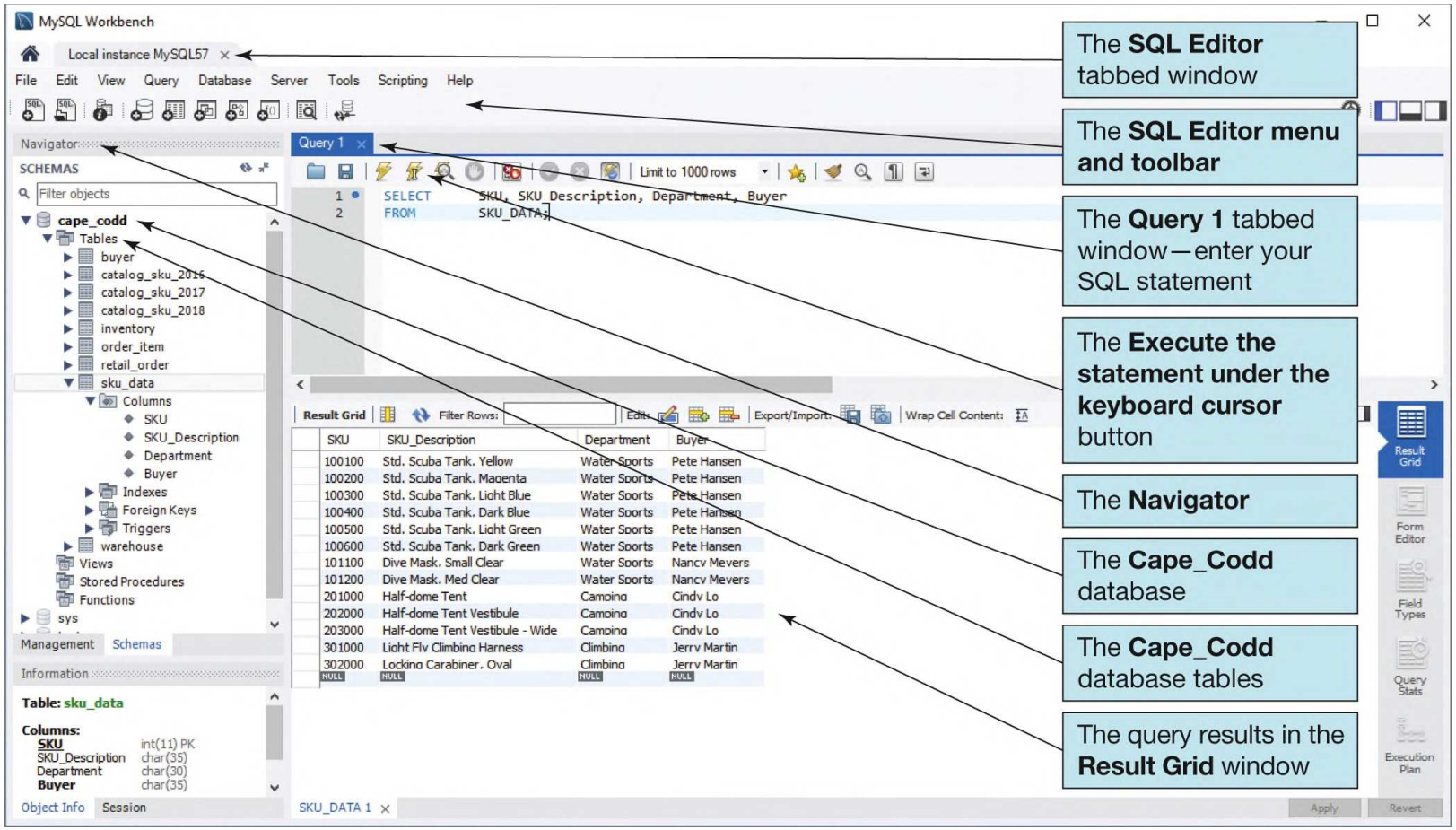
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Figure 2-20
Saving an Oracle SQL Query in Oracle SQL Developer



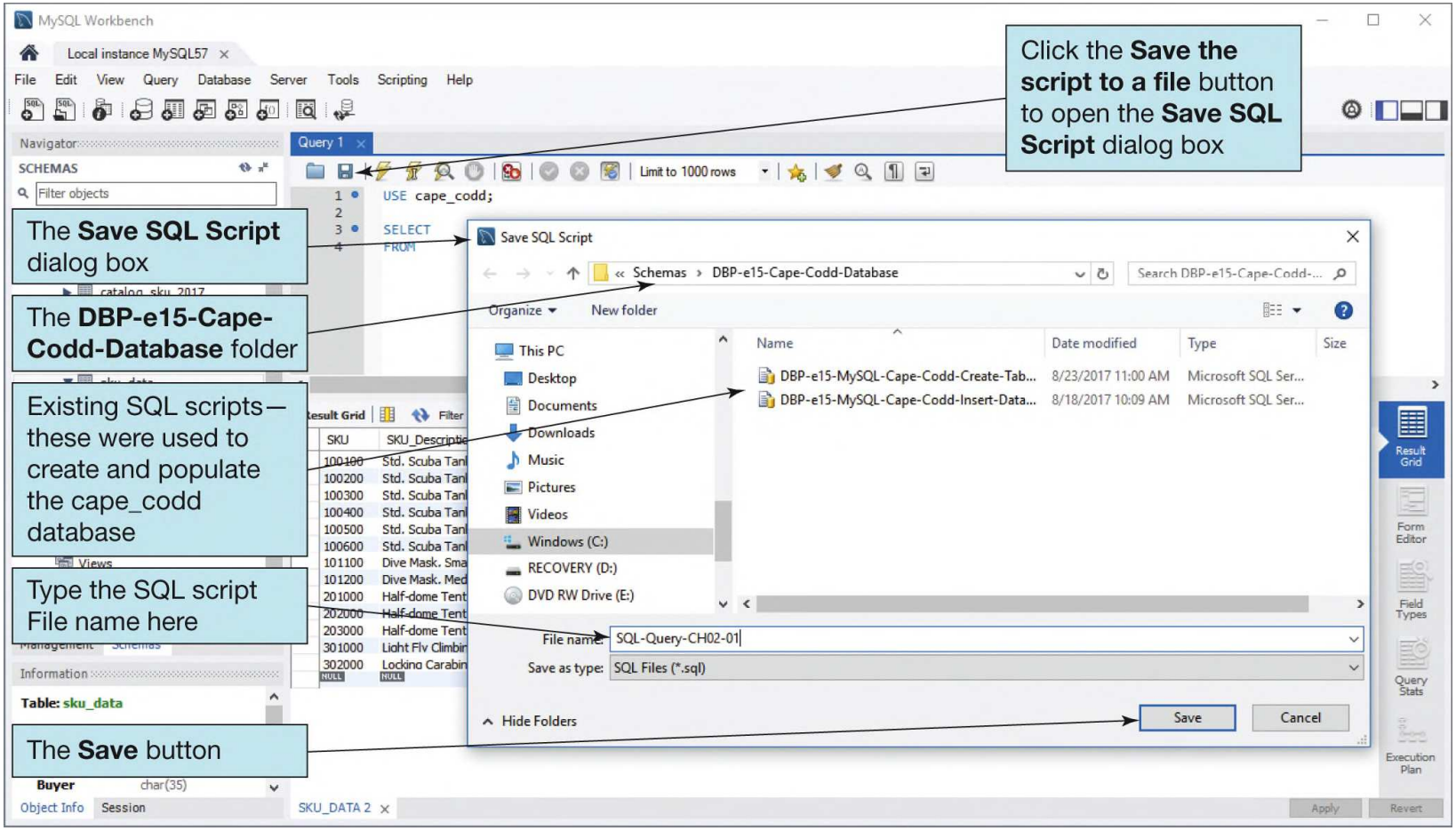
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-21
Running an SQL Query in MySQL Workbench



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-22
Saving a MySQL Query in MySQL Workbench



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Reading Specified Rows from a Single Table
The SQL DISTINCT Keyword

```
/* *** SQL-Query-CH02-05 *** */
SELECT      DISTINCT Buyer, Department
FROM        SKU_DATA;
```

	Buyer	Department
1	Cindy Lo	Camping
2	Jerry Martin	Climbing
3	Nancy Meyers	Water Sports
4	Pete Hansen	Water Sports

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Controlling the Number of Rows from a Single Table I
The SQL TOP {NumberOfRows} Function

```
/* *** SQL-Query-CH02-06 *** */
SELECT      TOP 5 Buyer, Department
FROM        SKU_DATA;
```

	Buyer	Department
1	Pete Hansen	Water Sports
2	Pete Hansen	Water Sports
3	Pete Hansen	Water Sports
4	Pete Hansen	Water Sports
5	Pete Hansen	Water Sports

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Controlling the Number of Rows from a Single Table II
The SQL TOP {Percentage} PERCENT Function

```
/* *** SQL-Query-CH02-07 *** */
SELECT      TOP 75 PERCENT Buyer, Department
FROM        SKU_DATA;
```

	Buyer	Department
1	Pete Hansen	Water Sports
2	Pete Hansen	Water Sports
3	Pete Hansen	Water Sports
4	Pete Hansen	Water Sports
5	Pete Hansen	Water Sports
6	Pete Hansen	Water Sports
7	Nancy Meyers	Water Sports
8	Nancy Meyers	Water Sports
9	Cindy Lo	Camping
10	Cindy Lo	Camping

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Controlling Specified Rows from a Single Table
The SQL WHERE Clause: Character Strings

```
/* *** SQL-Query-CH02-08 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Department = 'Water Sports';
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	100300	Std. Scuba Tank, Light Blue	Water Sports	Pete Hansen
4	100400	Std. Scuba Tank, Dark Blue	Water Sports	Pete Hansen
5	100500	Std. Scuba Tank, Light Green	Water Sports	Pete Hansen
6	100600	Std. Scuba Tank, Dark Green	Water Sports	Pete Hansen
7	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
8	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-23
SQL Comparison Operators

SQL Comparison Operators

Operator	Meaning
=	Is equal to
<>	Is NOT Equal to
<	Is less than
>	Is greater than
<=	Is less than OR equal to
>=	Is greater than OR equal to
IN	Is equal to one of a set of values
NOT IN	Is NOT equal to one of a set of values
BETWEEN	Is within a range of numbers (includes the end points)
NOT BETWEEN	Is NOT within a range of numbers (includes the end points)
LIKE	Matches a sequence of characters
NOT LIKE	Does NOT match a sequence of characters
IS NULL	Is equal to NULL
IS NOT NULL	Is NOT equal to NULL

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Reading Specified Rows from a Single Table I The SQL WHERE Clause: Dates

```
/* *** SQL-Query-CH02-09 *** */
SELECT      *
FROM        CATALOG_SKU_2017
WHERE       DateOnWebSite = '01-JAN-2017';
```

	CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
1	20170001	100100	Std. Scuba Tank, Yellow	Water Sports	23	2017-01-01
2	20170002	100300	Std. Scuba Tank, Light Blue	Water Sports	23	2017-01-01
3	20170004	101100	Dive Mask, Small Clear	Water Sports	26	2017-01-01
4	20170005	101200	Dive Mask, Med Clear	Water Sports	26	2017-01-01
5	20170006	201000	Half-dome Tent	Camping	46	2017-01-01
6	20170007	202000	Half-dome Tent Vestibule	Camping	46	2017-01-01
7	20170008	301000	Light Fly Climbing Harness	Climbing	77	2017-01-01
8	20170009	302000	Locking Carabiner, Oval	Climbing	79	2017-01-01

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Reading Specified Rows from a Single Table II The SQL WHERE Clause: Numbers

```
/* *** SQL-Query-CH02-10 *** */
SELECT      *
FROM        SKU_DATA
WHERE       SKU > 200000;
```

	SKU	SKU_Description	Department	Buyer
1	201000	Half-dome Tent	Camping	Cindy Lo
2	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
3	203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo
4	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
5	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Reading Specified Columns and Rows from a Single Table I
Column Names and the SQL WHERE Clause

```
/* *** SQL-Query-CH02-11 *** */
SELECT      SKU_Description, Department
FROM        SKU_DATA
WHERE       Department = 'Climbing';
```

	SKU_Description	Department
1	Light Fly Climbing Harness	Climbing
2	Locking Carabiner, Oval	Climbing

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Reading Specified Columns and Rows from a Single Table II
Column Names and the SQL WHERE Clause

```
/* *** SQL-Query-CH02-12 *** */
SELECT      SKU_Description, Buyer
FROM        SKU_DATA
WHERE       Department = 'Climbing';
```

- SQL does not require that the column used in the WHERE clause also appear in the SELECT clause column list as seen above.

	SKU_Description	Buyer
1	Light Fly Climbing Harness	Jerry Martin
2	Locking Carabiner, Oval	Jerry Martin

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Sorting the SQL Query Results I
The SQL ORDER BY Clause

```
/* *** SQL-Query-CH02-13 *** */
SELECT          *
FROM            ORDER_ITEM
ORDER BY OrderNumber;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	1000	202000	1	130.00	130.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	101200	1	50.00	50.00
6	3000	101100	2	50.00	100.00
7	3000	100200	1	300.00	300.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Sorting the SQL Query Results II
Ascending and Descending Sort Order

```
/* *** SQL-Query-CH02-16 *** */
SELECT     *
FROM       ORDER_ITEM
ORDER BY   Price DESC, OrderNumber ASC;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	201000	1	300.00	300.00
2	1000	202000	1	130.00	130.00
3	2000	101100	4	50.00	200.00
4	2000	101200	2	50.00	100.00
5	3000	101200	1	50.00	50.00
6	3000	101100	2	50.00	100.00
7	3000	100200	1	300.00	300.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-24
SQL Logical Operators

SQL Logical Operators

Operator	Meaning
AND	Both conditions are TRUE
OR	One of the other or both conditions are TRUE
NOT	Negates the associated condition

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL WHERE Clause Options II
SQL AND Operator

```
/* *** SQL-Query-CH02-18 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Department='Water Sports'
  AND       Buyer='Nancy Meyers';
```

	SKU	SKU_Description	Department	Buyer
1	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL WHERE Clause Options III
SQL OR Operator

```
/* *** SQL-Query-CH02-19 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Department='Camping'
  OR        Department='Climbing';
```

	SKU	SKU_Description	Department	Buyer
1	201000	Half-dome Tent	Camping	Cindy Lo
2	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
3	203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo
4	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
5	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL WHERE Clause Options IV
SQL NOT Operator

```
/* *** SQL-Query-CH02-20 *** */
SELECT     *
FROM       SKU_DATA
WHERE      Department='Water Sports'
  AND NOT  Buyer='Nancy Meyers';
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	100300	Std. Scuba Tank, Light Blue	Water Sports	Pete Hansen
4	100400	Std. Scuba Tank, Dark Blue	Water Sports	Pete Hansen
5	100500	Std. Scuba Tank, Light Green	Water Sports	Pete Hansen
6	100600	Std. Scuba Tank, Dark Green	Water Sports	Pete Hansen

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL WHERE Clause Options V
Sets of Values: SQL IN Operator

```
/* *** SQL-Query-CH02-21 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Buyer IN ('Nancy Meyers', 'Cindy Lo', 'Jerry
            Martin');
```

	SKU	SKU_Description	Department	Buyer
1	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
3	201000	Half-dome Tent	Camping	Cindy Lo
4	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
5	203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo
6	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
7	302000	Locking Carabiner, Oval	Climbing	Jerry Martin

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL WHERE Clause Options VI
Sets of Values: SQL NOT IN Operator

```
/* *** SQL-Query-CH02-22 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Buyer NOT IN ('Nancy Meyers', 'Cindy Lo', 'Jerry
            Martin');
```

	SKU	SKU_Description	Department	Buyer
1	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL WHERE Clause Options VII
Ranges of Values: Using Math Symbols

```
/* *** SQL-Query-CH02-23 *** */
SELECT     *
FROM       ORDER_ITEM
WHERE      ExtendedPrice >= 100
  AND      ExtendedPrice <= 200
ORDER BY   ExtendedPrice;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	3000	101100	2	50.00	100.00
2	2000	101200	2	50.00	100.00
3	1000	202000	1	130.00	130.00
4	2000	101100	4	50.00	200.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL WHERE Clause Options VIII
Ranges of Values: SQL BETWEEN Operator

```
/* *** SQL-Query-CH02-24 *** */
SELECT      *
FROM        ORDER_ITEM
WHERE       ExtendedPrice BETWEEN 100 AND 200 ORDER BY
            ExtendedPrice;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	3000	101100	2	50.00	100.00
2	2000	101200	2	50.00	100.00
3	1000	202000	1	130.00	130.00
4	2000	101100	4	50.00	200.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL WHERE Clause Options IX
Ranges of Values: SQL NOT BETWEEN Operator

```
/* *** SQL-Query-CH02-25 *** */
SELECT       *
FROM         ORDER_ITEM
WHERE        ExtendedPrice NOT BETWEEN 100 AND 200
ORDER BY     ExtendedPrice;
```

	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	3000	101200	1	50.00	50.00
2	1000	201000	1	300.00	300.00
3	3000	100200	1	300.00	300.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL WHERE Clause Options X
Character String Patterns: SQL Wildcard Characters

- The SQL LIKE operator and SQL NOT LIKE operator can be combined with wildcard symbols:
  - the **SQL underscore (_) wildcard character**, which represents a single, unspecified character in a specific position in the character string
  - the **SQL percent sign (%) wildcard character**, which represents any sequence of contiguous, unspecified characters (including spaces) in a specific position in the character string

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL WHERE Clause Options XI
Character String Patterns: SQL Like Operator I

```
/* *** SQL-Query-CH02-26 *** */
SELECT      *
FROM        SKU_DATA
WHERE       Buyer LIKE 'Pete%';
```

	SKU	SKU_Description	Department	Buyer
1	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
3	100300	Std. Scuba Tank, Light Blue	Water Sports	Pete Hansen
4	100400	Std. Scuba Tank, Dark Blue	Water Sports	Pete Hansen
5	100500	Std. Scuba Tank, Light Green	Water Sports	Pete Hansen
6	100600	Std. Scuba Tank, Dark Green	Water Sports	Pete Hansen

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL WHERE Clause Options XII
Character String Patterns: SQL Like Operator II

```
/* *** SQL-Query-CH02-27 *** */
SELECT      *
FROM        SKU_DATA
WHERE       SKU_Description LIKE '%Tent%';
```

	SKU	SKU_Description	Department	Buyer
1	201000	Half-dome Tent	Camping	Cindy Lo
2	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
3	203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL WHERE Clause Options XIII
Character String Patterns: SQL Like Operator III

```
/* *** SQL-Query-CH02-29 *** */
SELECT      *
FROM        SKU_DATA
WHERE       SKU LIKE '%2__';
```

	SKU	SKU_Description	Department	Buyer
1	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL WHERE Clause Options XIV
Using NULL Values: SQL IS NULL Operator

```
/* *** SQL-Query-CH02-30 *** */
SELECT      *
FROM        CATALOG_SKU_2017
WHERE       CatalogPage IS NULL;
```

	CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
1	20170003	100400	Std. Scuba Tank, Dark Blue	Water Sports	NULL	2017-08-01

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL WHERE Clause Options XV
Using NULL Values: SQL IS NOT NULL Operator

```
/* *** SQL-Query-CH02-31 *** */
SELECT      *
FROM        CATALOG_SKU_2017
WHERE       CatalogPage IS NOT NULL;
```

	CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
1	20170001	100100	Std. Scuba Tank, Yellow	Water Sports	23	2017-01-01
2	20170002	100300	Std. Scuba Tank, Light Blue	Water Sports	23	2017-01-01
3	20170004	101100	Dive Mask, Small Clear	Water Sports	26	2017-01-01
4	20170005	101200	Dive Mask, Med Clear	Water Sports	26	2017-01-01
5	20170006	201000	Half-dome Tent	Camping	46	2017-01-01
6	20170007	202000	Half-dome Tent Vestibule	Camping	46	2017-01-01
7	20170008	301000	Light Fly Climbing Harness	Climbing	77	2017-01-01
8	20170009	302000	Locking Carabiner, Oval	Climbing	79	2017-01-01

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-25
SQL Built-in Aggregate Functions

SQL Built-in Aggregate Functions

Function	Meaning
COUNT(*)	Count the number of rows in a table
COUNT ({Name}]	Count the number of rows in the table where column {Name} IS NOT NULL
SUM	Calculate the sum of all values (numeric columns only)
AVG	Calculate the average of all values (numeric columns only)
MIN	Calculate the minimum value of all values
MAX	Calculate the maximum value of all values

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Performing Calculations in SQL Queries II
Using SQL Built-in Aggregate Functions: SUM

```
/* *** SQL-Query-CH02-32 *** */
SELECT      SUM(OrderTotal) AS OrderSum
FROM        RETAIL_ORDER;
```

	OrderSum
1	1235.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Performing Calculations in SQL Queries III

Using SQL Built-in Aggregate Functions: SUM, AVG, MIN AND MAX

```
/* *** SQL-Query-CH02-35 *** */
SELECT      SUM(ExtendedPrice) AS OrderItemSum,
            AVG(ExtendedPrice) AS OrderItemAvg,
            MIN(ExtendedPrice) AS OrderItemMin,
            MAX(ExtendedPrice) AS OrderItemMax
FROM        ORDER_ITEM;
```

	OrderItemSum	OrderItemAvg	OrderItemMin	OrderItemMax
1	1180.00	168.571428	50.00	300.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Performing Calculations in SQL Queries IV
Using SQL Built-in Aggregate Functions: COUNT(*)

```
/* *** SQL-Query-CH02-36 *** */
SELECT      COUNT(*) AS NumberOfRows
FROM        ORDER_ITEM;
```

	NumberOfRows
1	7

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Performing Calculations in SQL Queries V
Using SQL Built-in Aggregate Functions: COUNT({Name})

```
/* *** SQL-Query-CH02-38 *** */
SELECT      COUNT(DISTINCT Department) AS DeptCount
FROM        SKU_DATA;
```

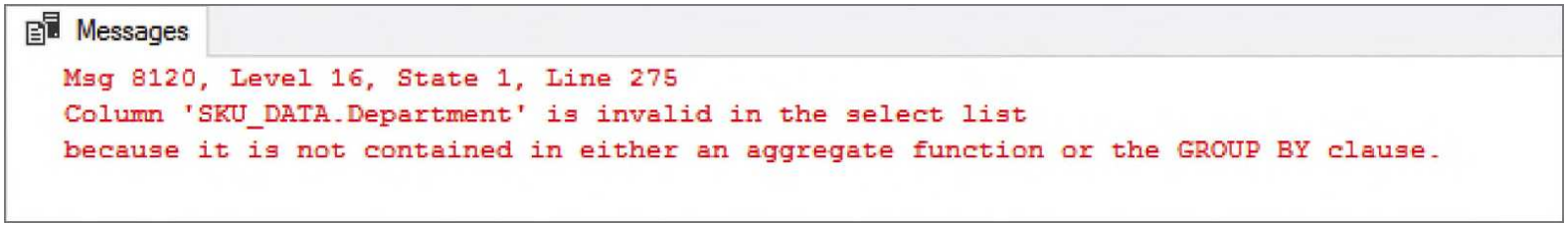
	DeptCount
1	3

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Performing Calculations in SQL Queries VI
SQL Expressions I

- You should be aware of two limitations to SQL built-in functions:
  - 1) You cannot combine table column names with an SQL built-in function.

```
/* *** SQL-Query-CH02-40 *** */
SELECT  Department, COUNT(*)
FROM    SKU_DATA;
```

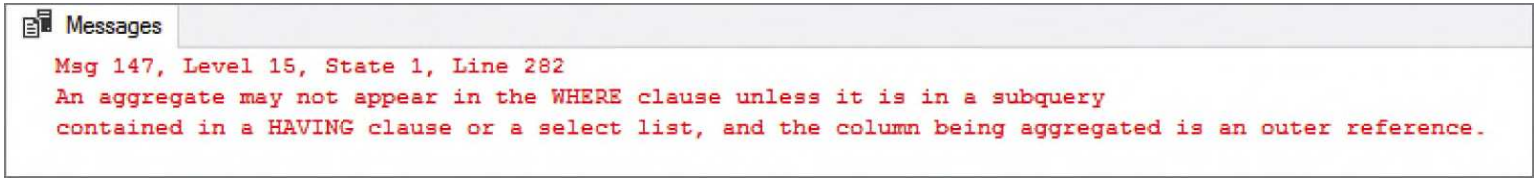


Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Performing Calculations in SQL Queries VII
SQL Expressions II

- 2) You cannot use them (table names) in an SQL WHERE clause because the SQL WHERE clause operates on rows (choosing which rows will be displayed), whereas the aggregate functions operate on columns.

```
/* *** SQL-Query-CH02-41 *** */
SELECT  *
FROM    RETAIL_ORDER
WHERE   OrderTotal > AVG(OrderTotal);
```



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Performing Calculations in SQL Queries VIII Using SQL Expressions I

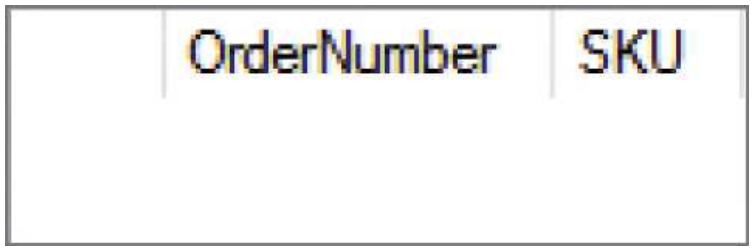
```
/* *** SQL-Query-CH02-42 *** */
SELECT     OrderNumber, SKU, (Quantity * Price) AS EP
FROM       ORDER_ITEM
ORDER BY   OrderNumber, SKU;
```

	OrderNumber	SKU	EP
1	1000	201000	300.00
2	1000	202000	130.00
3	2000	101100	200.00
4	2000	101200	100.00
5	3000	100200	300.00
6	3000	101100	100.00
7	3000	101200	50.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Performing Calculations in SQL Queries IX Using SQL Expressions II

```
/* *** SQL-Query-CH02-44 *** */
SELECT     OrderNumber, SKU
FROM       ORDER_ITEM
WHERE      Quantity * Price) <> ExtendedPrice
ORDER BY   OrderNumber, SKU;
```



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Performing Calculations in SQL Queries X

Using SQL Expressions with String Functions: Concatenation I

```
/* *** SQL-Query-CH02-45 *** */
SELECT     SKU, SKU_Description, (Buyer+' in '+Department) AS
           Sponsor
FROM       SKU_DATA
ORDER BY   SKU;
```

	SKU	SKU_Description	Sponsor	
1	100100	Std. Scuba Tank, Yellow	Pete Hansen	in Water Sports
2	100200	Std. Scuba Tank, Magenta	Pete Hansen	in Water Sports
3	100300	Std. Scuba Tank, Light Blue	Pete Hansen	in Water Sports
4	100400	Std. Scuba Tank, Dark Blue	Pete Hansen	in Water Sports
5	100500	Std. Scuba Tank, Light Green	Pete Hansen	in Water Sports
6	100600	Std. Scuba Tank, Dark Green	Pete Hansen	in Water Sports
7	101100	Dive Mask, Small Clear	Nancy Meyers	in Water Sports
8	101200	Dive Mask, Med Clear	Nancy Meyers	in Water Sports
9	201000	Half-dome Tent	Cindy Lo	in Camping
10	202000	Half-dome Tent Vestibule	Cindy Lo	in Camping
11	203000	Half-dome Tent Vestibule - Wide	Cindy Lo	in Camping
12	301000	Light Fly Climbing Harness	Jerry Martin	in Climbing
13	302000	Locking Carabiner, Oval	Jerry Martin	in Climbing

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Performing Calculations in SQL Queries XI

Using SQL Expressions with String Functions: Concatenation II

This is the Oracle Database syntax:

```
/* *** SQL-Query-CH02-45-Oracle-Database *** */
SELECT     SKU, SKU_Description, (Buyer||' in '||Department)
           AS Sponsor
FROM       SKU_DATA
ORDER BY   SKU;
```

This is the MySQL syntax:

```
/* *** SQL-Query-CH02-45-MySQL *** */
SELECT     SKU, SKU_Description, CONCAT(Buyer, ` in ’,
           Department) AS Sponsor
FROM       SKU_DATA
ORDER BY SKU;
```

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Performing Calculations in SQL Queries XII

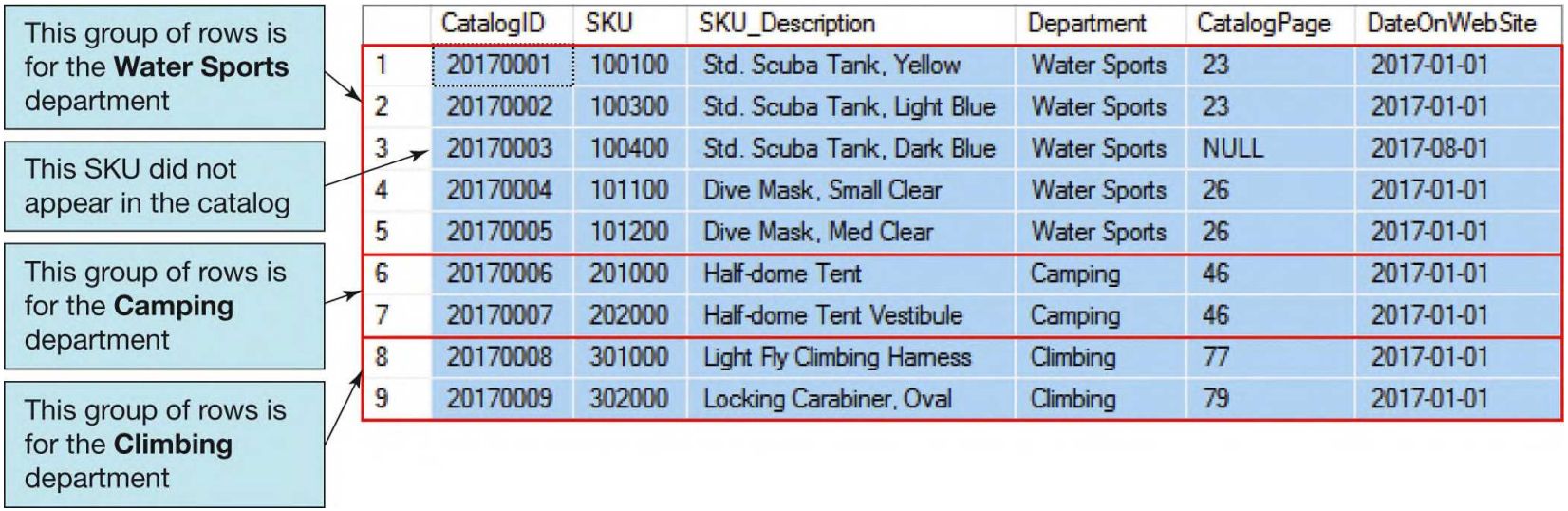
Using SQL Expressions with String Functions: Concatenation III

```
/* *** SQL-Query-CH02-46 *** */
SELECT     SKU, SKU_Description, RTRIM(Buyer)+' in `
           +RTRIM(Department) AS Sponsor
FROM       SKU_DATA
ORDER BY   SKU;
```

	SKU	SKU_Description	Sponsor
1	100100	Std. Scuba Tank, Yellow	Pete Hansen in Water Sports
2	100200	Std. Scuba Tank, Magenta	Pete Hansen in Water Sports
3	100300	Std. Scuba Tank, Light Blue	Pete Hansen in Water Sports
4	100400	Std. Scuba Tank, Dark Blue	Pete Hansen in Water Sports
5	100500	Std. Scuba Tank, Light Green	Pete Hansen in Water Sports
6	100600	Std. Scuba Tank, Dark Green	Pete Hansen in Water Sports
7	101100	Dive Mask, Small Clear	Nancy Meyers in Water Sports
8	101200	Dive Mask, Med Clear	Nancy Meyers in Water Sports
9	201000	Half-dome Tent	Cindy Lo in Camping
10	202000	Half-dome Tent Vestibule	Cindy Lo in Camping
11	203000	Half-dome Tent Vestibule - Wide	Cindy Lo in Camping
12	301000	Light Fly Climbing Harness	Jerry Martin in Climbing
13	302000	Locking Carabiner, Oval	Jerry Martin in Climbing

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

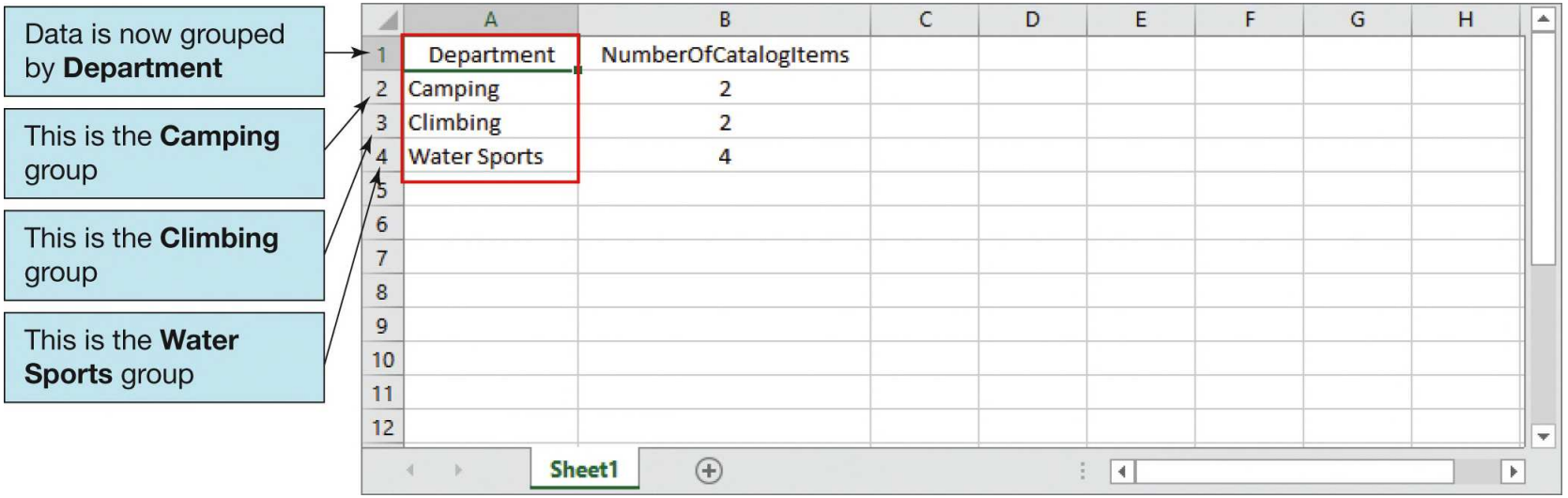
Figure 2-26
Department Groups in the CATALOG_SKU_2017 Table



	CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWebSite
1	20170001	100100	Std. Scuba Tank, Yellow	Water Sports	23	2017-01-01
2	20170002	100300	Std. Scuba Tank, Light Blue	Water Sports	23	2017-01-01
3	20170003	100400	Std. Scuba Tank, Dark Blue	Water Sports	NULL	2017-08-01
4	20170004	101100	Dive Mask, Small Clear	Water Sports	26	2017-01-01
5	20170005	101200	Dive Mask, Med Clear	Water Sports	26	2017-01-01
6	20170006	201000	Half-dome Tent	Camping	46	2017-01-01
7	20170007	202000	Half-dome Tent Vestibule	Camping	46	2017-01-01
8	20170008	301000	Light Fly Climbing Harness	Climbing	77	2017-01-01
9	20170009	302000	Locking Carabiner, Oval	Climbing	79	2017-01-01

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-27
Grouping Data into Department Groups



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Grouping Rows in SQL Queries III
The SQL GROUP BY Clause

```
/* ***     SQL-Query-CH02-48 *** */
SELECT     Department, COUNT(SKU) AS NumberOfCatalogItems
FROM       CATALOG_SKU_2017
WHERE      CatalogPage IS NOT NULL
GROUP BY   Department;
```

	Department	NumberOfCatalogItems
1	Camping	2
2	Climbing	2
3	Water Sports	4

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Grouping Rows in SQL Queries IV
The SQL HAVING Clause

```
/* *** SQL-Query-CH02-49 *** */
SELECT          Department, COUNT(SKU) AS NumberOfCatalogItems
FROM            CATALOG_SKU_2017
WHERE           CatalogPage IS NOT NULL
GROUP  BY       Department
HAVING          COUNT(SKU) > 2;
```

	Department	NumberOfCatalogItems
1	Water Sports	4

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Grouping Rows in SQL Queries V

- The SQL **WHERE** clause specifies which *rows* will be used to determine the groups.
- The SQL **HAVING** clause specifies which *groups* will be used in the final result.
- In general, place **WHERE** before **GROUP BY**. Some DBMS products do not require that placement; but to be safe, always put **WHERE** before **GROUP BY**.
- There is an ambiguity in statements that include both **WHERE** and **HAVING** clauses. The results can vary, so to eliminate this ambiguity SQL *always* applies **WHERE** before **HAVING**.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Grouping Rows in SQL Queries VI
Grouping by More Than One Column

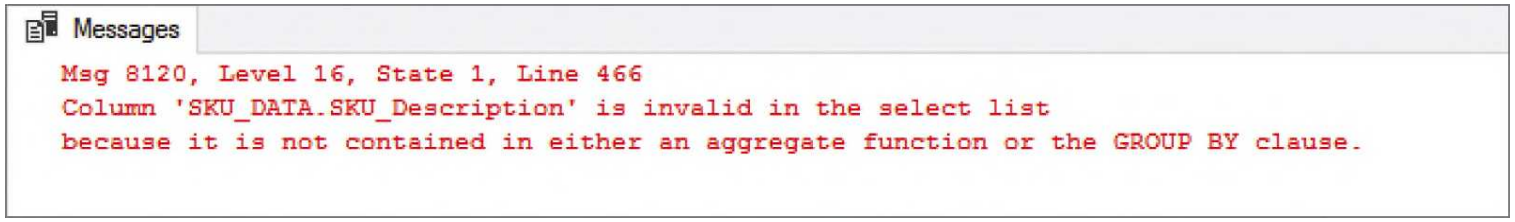
```
/* *** SQL-Query-CH02-50 *** */
SELECT     Department, Buyer, COUNT(SKU) AS
           Dept_Buyer_SKU_Count
FROM       SKU_DATA
GROUP BY   Department, Buyer;
```

	Department	Buyer	Dept_Buyer_SKU_Count
1	Camping	Cindy Lo	3
2	Climbing	Jerry Martin	2
3	Water Sports	Nancy Meyers	2
4	Water Sports	Pete Hansen	6

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Grouping Rows in SQL Queries VII
Column Names that Must Appear in GROUP BY

```
/* *** SQL-Query-CH02-51 *** */
SELECT     Department, SKU, COUNT(SKU) AS Dept_SKU_Count
FROM       SKU_DATA
GROUP BY   Department;
```



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Grouping Rows in SQL Queries VIII
Using the SQL ORDER BY Clause with Groupings

```
/* *** SQL-Query-CH02-52 *** */
SELECT     Department, COUNT(SKU) AS Dept_SKU_Count
FROM       SKU_DATA
WHERE      SKU <> 302000
GROUP BY   Department
HAVING     COUNT(SKU) > 1
ORDER BY   Dept_SKU_Count;
```

	Department	Dept_SKU_Count
1	Camping	3
2	Water Sports	8

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Database Processing
Fundamentals, Design, and Implementation (15th Edition)

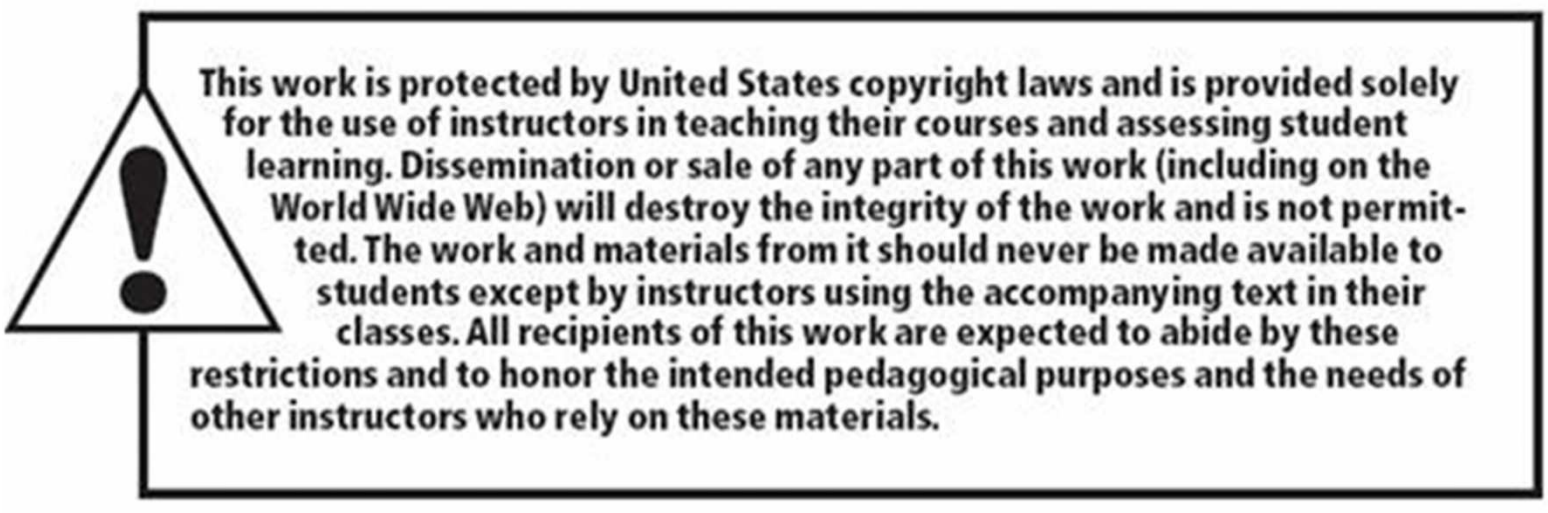
End of Presentation:

Chapter Two

Part 1

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Copyright

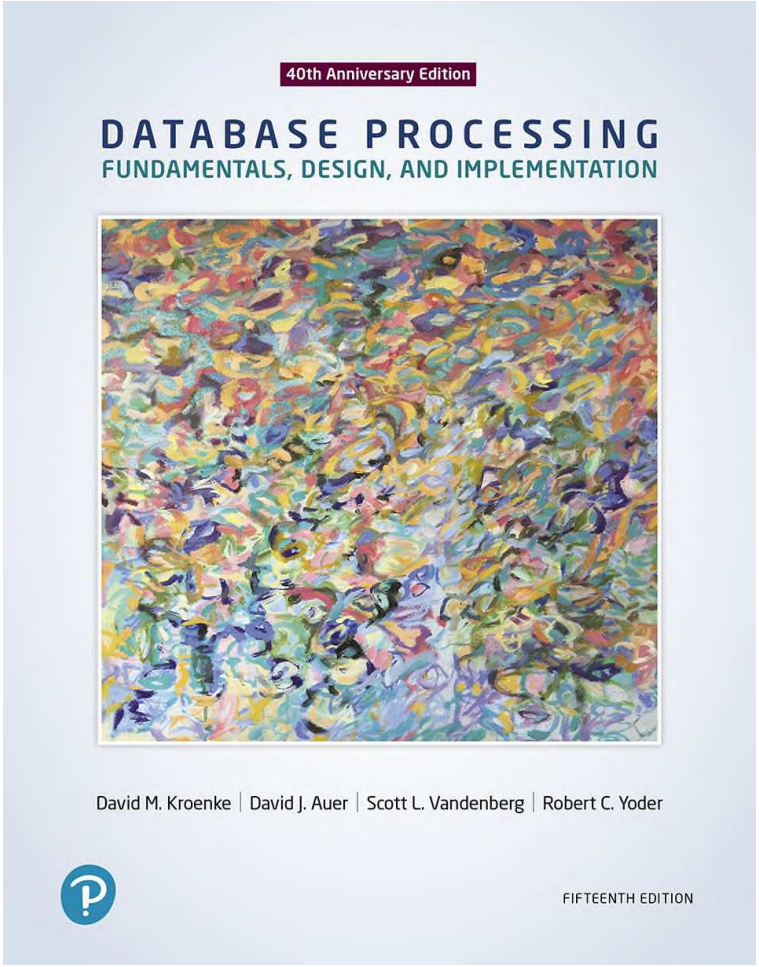


This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Database Processing: Fundamentals, Design, and Implementation

15th Edition



Chapter 2
Part 2
Introduction to Structured Query Language

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Learning Objectives (1 of 2)

- **2.1** To understand the use of extracted datasets in business intelligence (BI) systems
- **2.2** To understand the use of ad-hoc queries in business intelligence (BI) systems
- **2.3** To understand the history and significance of Structured Query Language (SQL)
- **2.4** To understand the SQL SELECT/FROM/WHERE framework as the basis for database queries
- **2.5** To create SQL queries to retrieve data from a single table
- **2.6** To create SQL queries that use the SQL SELECT, FROM, WHERE, ORDER BY, GROUP BY, and HAVING clauses
- **2.7** To create SQL queries that use the SQL DISTINCT, TOP, and TOP PERCENT keywords
- **2.8** To create SQL queries that use the SQL comparison operators, including BETWEEN, LIKE, IN, and IS NULL

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Learning Objectives (2 of 2)

- **2.9** To create SQL queries that use the SQL logical operators, including AND, OR, and NOT
- **2.10** 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
- **2.11** To create SQL queries that retrieve data from a single table while restricting the data based upon data in another table (subquery)
- **2.12** To create SQL queries that retrieve data from multiple tables using the SQL join and JOIN ON operations
- **2.13** To create SQL queries on recursive relationships
- **2.14** To create SQL queries that retrieve data from multiple tables using the SQL OUTER JOIN operation
- **2.15** To create SQL queries that retrieve data from multiple tables using SQL set operators UNION, INTERSECT, and EXCEPT

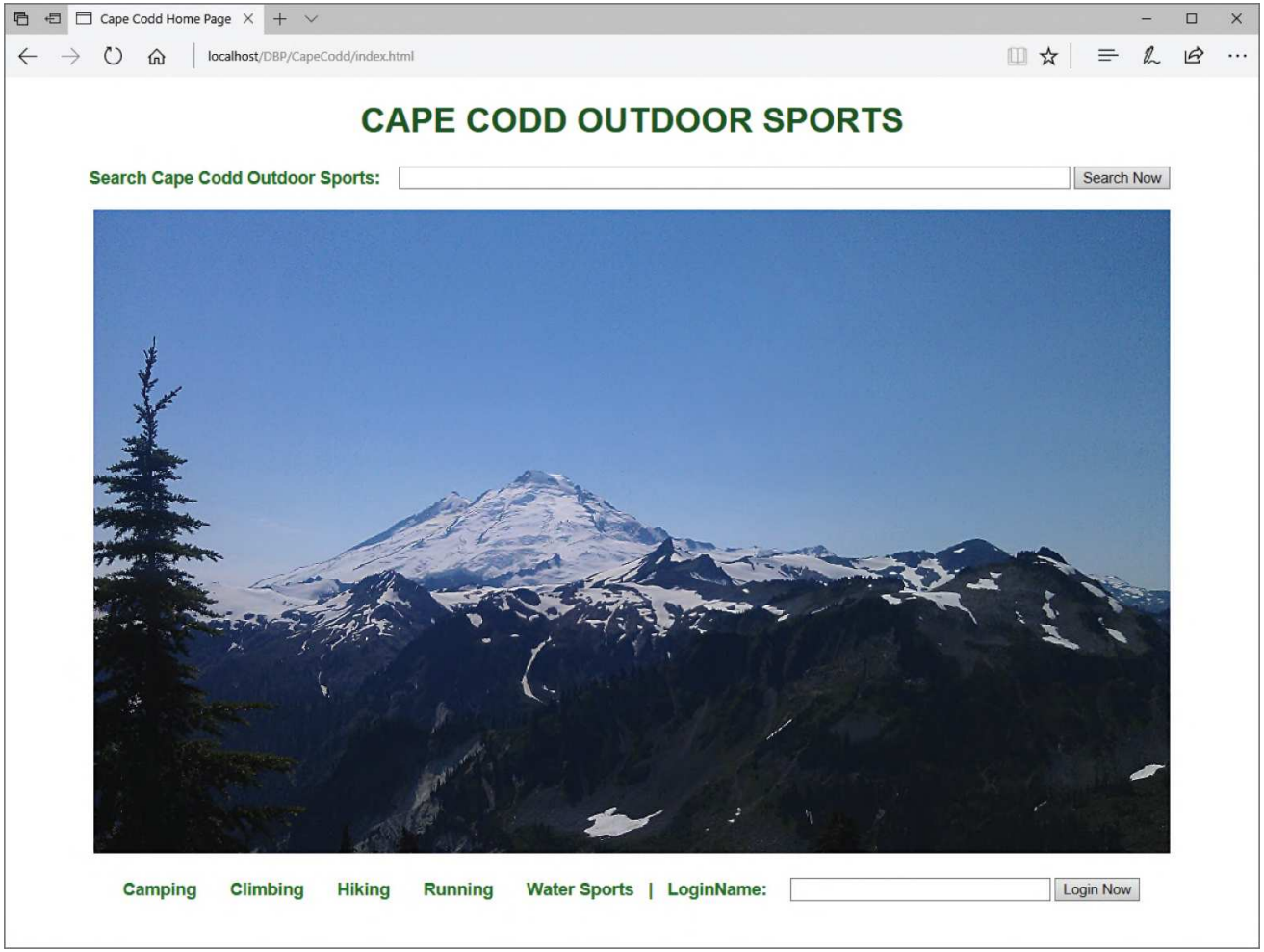
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Chapter 2 Part 2 Coverage

- This Microsoft PowerPoint Slide Show covers **multiple table queries** and **chapter objectives 2.11-2.15.**
- The material in the Microsoft PowerPoint Slide Show for Chapter Two Part One on **single table queries** *must* by covered before discussing the material in this Slide Show.
- A review of the **Cape Codd Outdoor Sports** database is included for continuity.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-1
Cape Codd Outdoor Sports



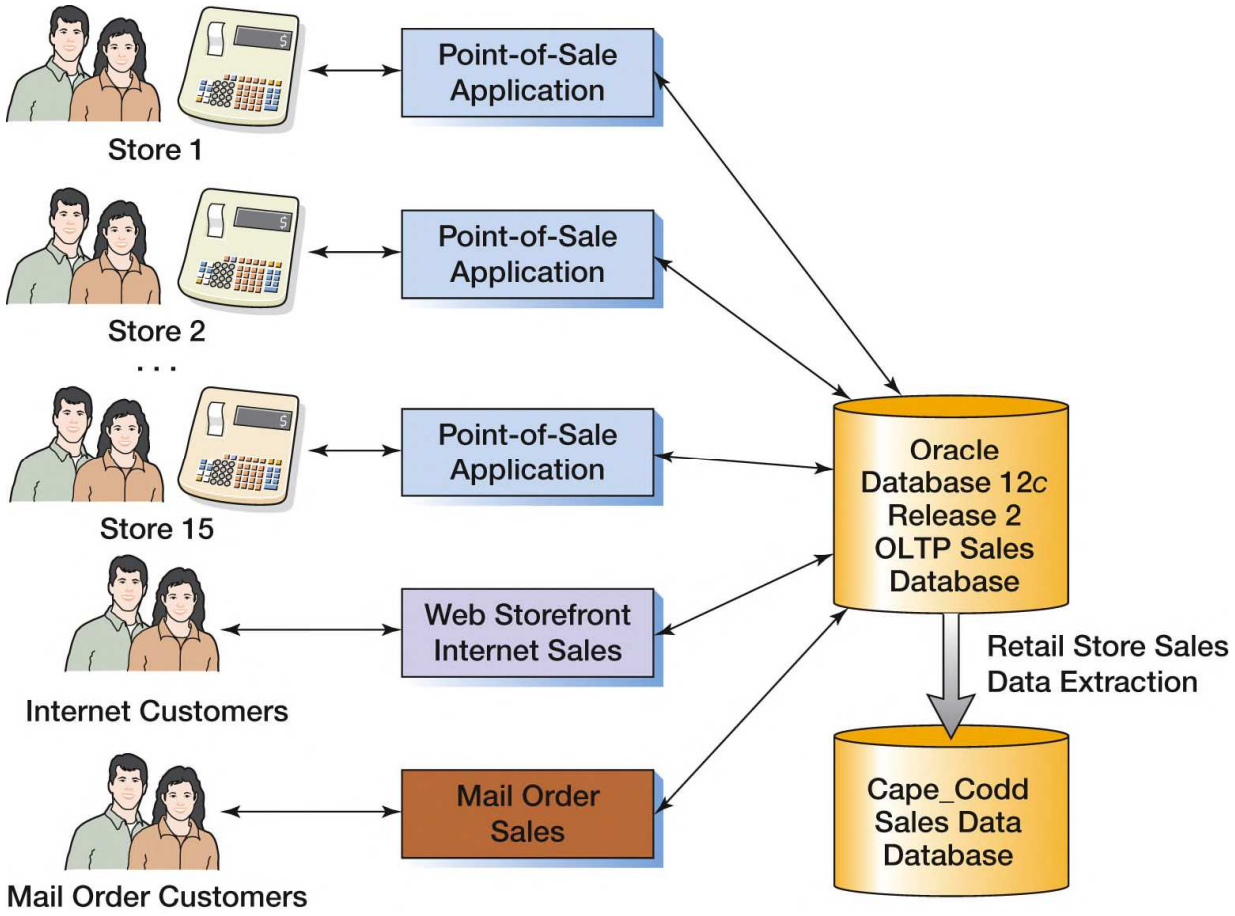
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Cape Codd Outdoor Sports Background

- Cape Codd Outdoor Sports is a fictitious company based on an actual outdoor retail equipment vendor.
- Cape Codd Outdoor Sports:
  - Has 15 retail stores in the United States and Canada
  - Has an online Internet store
  - Has a (postal) mail order department
- All retail sales are recorded in an Oracle Database 12*c* Release 2 DBMS.

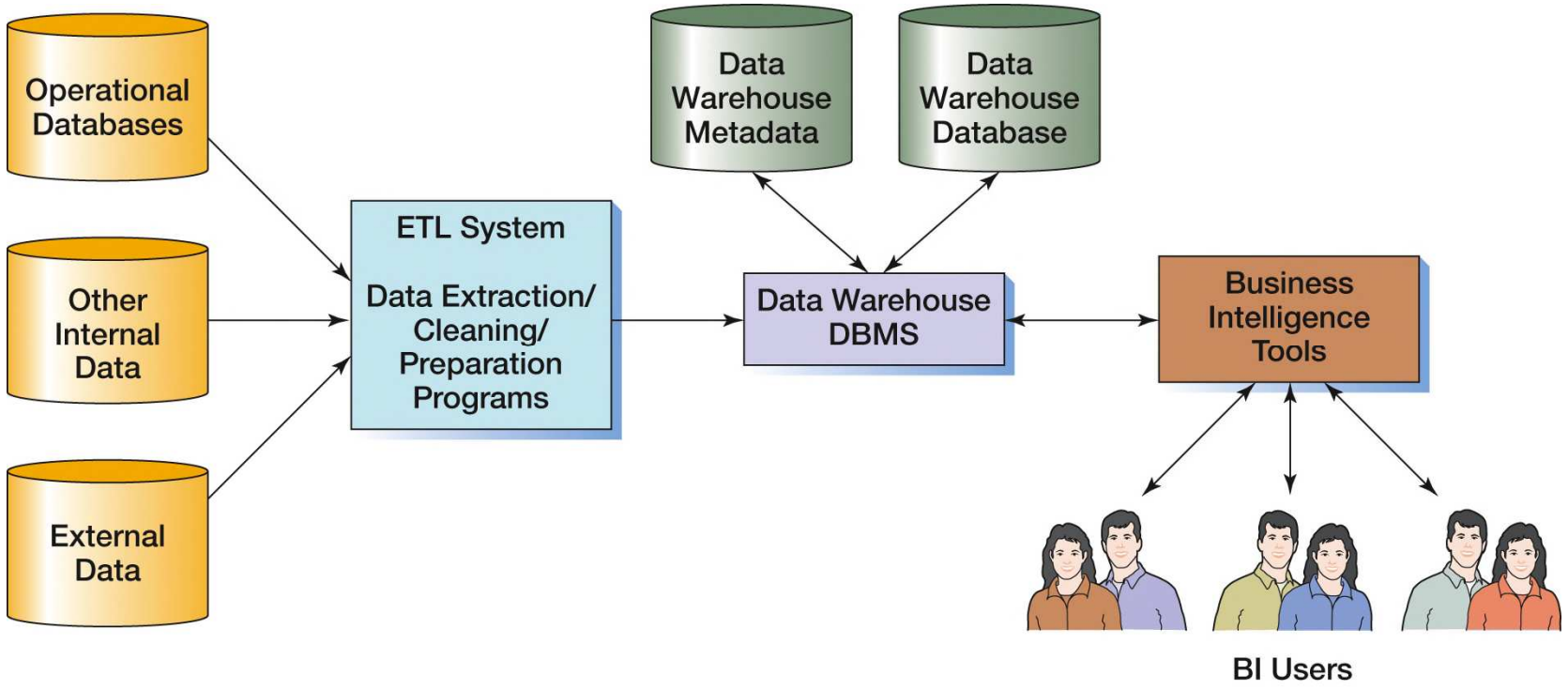
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Figure 2-2
Cape Codd Retail Sales Data Extraction Process



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-3
Components of a Data Warehouse



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Cape Codd Outdoor Sports
Retail Sales and Catalog Context Data Extraction 1

- The Cape Codd marketing department needs an analysis of:
  - In-store sales
  - Catalog content
- The entire database is not needed for this, only an **extraction** of retail sales data and catalog content.
- The data is extracted by the IS department from the **operational database** into a **separate, off-line database** for use by the marketing department.

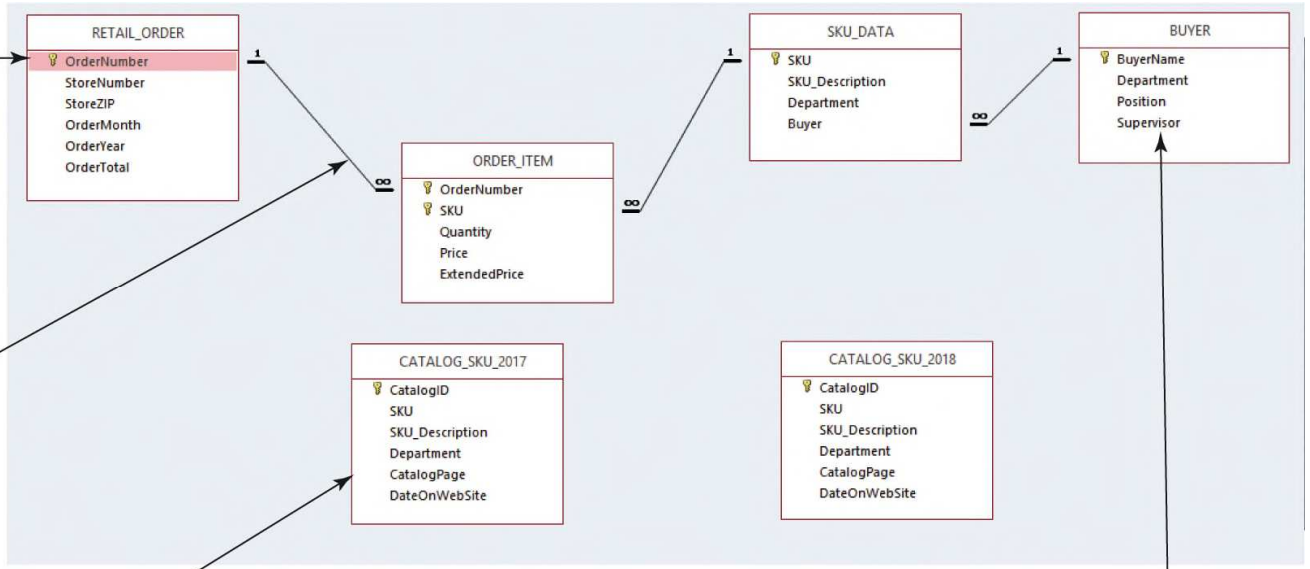
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Figure 2-4
Extracted Retail Sales Data Database Tables

The RETAIL_ORDER table—the key symbol shows the primary key

The relationship between RETAIL_ORDER and ORDER_ITEM—the number 1 and the infinity symbol indicate that one retail order may be linked to many order items by OrderNumber

The CATALOG_SKU_2017 and CATALOG_SKU_2018 tables are *not* linked to other tables in the database



The recursive relationship from Supervisor to BuyerName within BUYER is *not* shown

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Figure 2-5 (1 of 2)
Cape Codd Extracted Retail Sales Data Format (1 of 2)

Table	Column	Data Type
RETAIL_ORDER	OrderNumber	Integer
	StoreNumber	Integer
	StoreZIP	Character (9)
	OrderMonth	Character (12)
	OrderYear	Integer
	OrderTotal	Currency
	OrderNumber	Integer
ORDER_ITEM	SKU	Integer
	Quantity	Integer
	Price	Currency
	ExtendedPrice	Currency

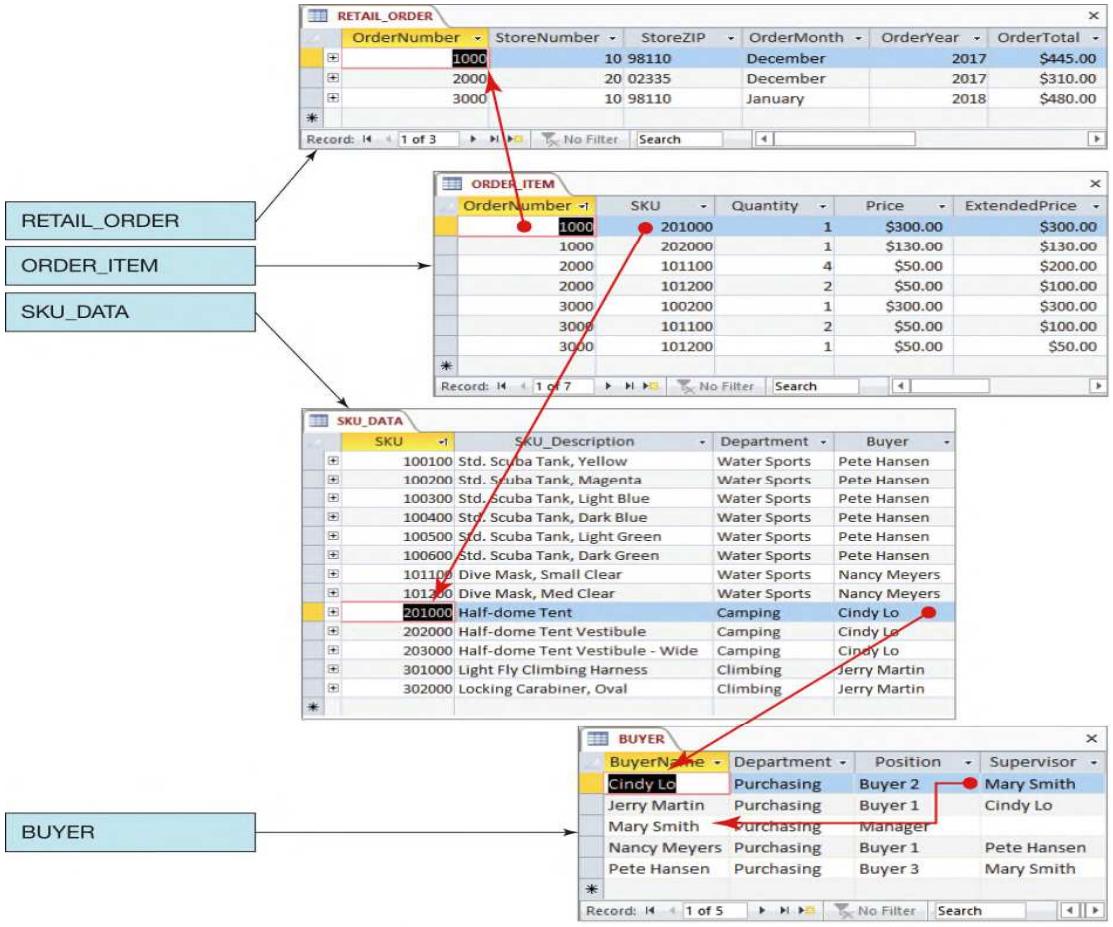
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-5 (2 of 2)
Cape Codd Extracted Retail Sales Data Format (2 of 2)

Table	Column	Data Type
SKU_DATA	SKU	Integer
	SKU_Description	Character (35)
	Department	Character (30)
	Buyer	Character (35)
BUYER	BuyerName	Character (35)
	Department	Character (30)
	Position	Character (10)
	Supervisor	Character (35)
CATALOG_SKU_20##	CatalogID	Integer
	SKU	Integer
	SKU_Description	Character (35)
	Department	Character (30)
	CatalogPage	Integer
	DateOnWebSite	Date

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-6 (1 of 2)
Extracted Retail Sales Database (1 of 2)



(a) The Linked RETAIL_ORDER, ORDER_ITEM, SKU_DATA, and BUYER Tables

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-6 (2 of 2)
Extracted Retail Sales Database (2 of 2)

CATALOG_SKU_2017

CATALOG_SKU_2017

CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWeb
20170001	100100	Std. Scuba Tank, Yellow	Water Sports	23	1/1/2017
20170002	100300	Std. Scuba Tank, Light Blue	Water Sports	23	1/1/2017
20170003	100400	Std. Scuba Tank, Dark Blue	Water Sports		8/1/2017
20170004	101100	Dive Mask, Small Clear	Water Sports	26	1/1/2017
20170005	101200	Dive Mask, Med Clear	Water Sports	26	1/1/2017
20170006	201000	Half-dome Tent	Camping	46	1/1/2017
20170007	202000	Half-dome Tent Vestibule	Camping	46	1/1/2017
20170008	301000	Light Fly Climbing Harness	Climbing	77	1/1/2017
20170009	302000	Locking Carabiner, Oval	Climbing	79	1/1/2017

Record: 1 of 9 No Filter Search

CATALOG_SKU_2018

CATALOG_SKU_2018

CatalogID	SKU	SKU_Description	Department	CatalogPage	DateOnWeb
20180001	100100	Std. Scuba Tank, Yellow	Water Sports	23	1/1/2018
20180002	100200	Std. Scuba Tank, Magenta	Water Sports	23	1/1/2018
20180003	101100	Dive Mask, Small Clear	Water Sports	27	8/1/2018
20180004	101200	Dive Mask, Med Clear	Water Sports	27	1/1/2018
20180005	201000	Half-dome Tent	Camping	45	1/1/2018
20180006	202000	Half-dome Tent Vestibule	Camping	45	1/1/2018
20180007	203000	Half-dome Tent Vestibule - Wide	Camping		4/1/2018
20180008	301000	Light Fly Climbing Harness	Climbing	76	1/1/2018
20180009	302000	Locking Carabiner, Oval	Climbing	78	1/1/2018

Record: 1 of 9 No Filter Search

(b) The Non-Linked CATALOG_SKU_2017 and CATALOG_SKU_2018 Tables

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Cape Codd Database Available Online (1 of 2)

- Versions of the complete Cape Codd database are available in the downloadable Student Files available at:

  http://www.pearsonhighered.com/kroenke/

- These include versions for:
  - Microsoft Access 2016
  - Microsoft SQL Server 2017
  - Oracle Database XE
  - MySQL 5.7
- **We recommend you actually run all material in a live database!**

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Cape Codd Database Available Online (2 of 2)

- To complete setting up the Cape Codd database, set the referenced materials:
  - For Microsoft SQL Server 2014:
    - See Online Chapter 10A
  - For Oracle Database 12c and Oracle Database XE:
    - See Online Chapter 10B
  - For MySQL 5.6
    - See Online Chapter 10C
- Online chapters 10A, 10B, and 10C are available for download at:

  http://www.pearsonhighered.com/kroenke/

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Querying Two or More Tables with SQL

- SQL provides two different techniques for querying data from multiple tables:
  - The **SQL subquery**
  - The **SQL join**
- Although both work with multiple tables, they are used for slightly different purposes.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Querying Multiple Tables with SQL Subqueries (1 of 5) The Logic of Subqueries I

We want to know the revenue for Water Sports items, which have SKU values of 100100, 100200, 101100, and 101200.

Given that we know the SKU values, we can use this query:

```
/* *** SQL-Query-CH02-53 *** */
SELECT    SUM(ExtendedPrice) AS WaterSportsRevenue
FROM      ORDER_ITEM
WHERE     SKU IN (100100, 100200, 100300, 100400, 100500, 100600,
          101100, 101200);
```

	WaterSportsRevenue
1	750.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Querying Multiple Tables with SQL Subqueries (2 of 5) The Logic of Subqueries II

What if we don't know the SKU values? We can determine them with this query:

```
/* *** SQL-Query-CH02-54 *** */
SELECT    SKU
FROM      SKU_DATA
WHERE     Department = 'Water Sports';
```

	SKU
1	100100
2	100200
3	100300
4	100400
5	100500
6	100600
7	101100
8	101200

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Querying Multiple Tables with SQL Subqueries (3 of 5)
The Logic of Subqueries III

We can combine these two as shown in SQL-Query-02-55. The second query, which is enclosed in parentheses, is called an **SQL subquery**. Note how the subquery returns a set of values for use by the **top level query**, and note the use of the **SQL IN keyword**.

```
/* *** SQL-Query-CH02-55 *** */
   SELECT   SUM(ExtendedPrice) AS WaterSportsRevenue
   FROM     ORDER_ITEM
   WHERE    SKU IN
            (SELECT   SKU
             FROM     SKU_DATA
             WHERE    Department = 'Water Sports');
```

	WaterSportsRevenue
1	750.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Querying Multiple Tables with SQL Subqueries (4 of 5)
The Logic of Subqueries IV

- An **SQL subquery** is often described as:
  - A nested query
  - A query within a query
- Note that SQL queries using subqueries are still effectively *single table queries* in the sense that only the ***columns of the top level query can be displayed in the query results***!
- Multiple subqueries can be used to process three or even more tables.
- All parts if the SQL SELECT statement syntax can be applied to the table displaying the results of an SQL query using subqueries.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Querying Multiple Tables with SQL Subqueries (5 of 5)
The Logic of Subqueries V

```
/* *** SQL-Query-CH02-56 *** */
SELECT  DISTINCT Buyer, Department
FROM    SKU_DATA
WHERE   SKU IN
        (SELECT SKU
         FROM   ORDER_ITEM
         WHERE  OrderNumber IN
                (SELECT   OrderNumber
                 FROM     RETAIL_ORDER
                 WHERE    OrderMonth = 'January'
                   AND    OrderYear = 2018));
```

	Buyer	Department
1	Nancy Meyers	Water Sports
2	Pete Hansen	Water Sports

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Querying Multiple Tables with SQL Joins (1 of 10)
The Logic of Joins

- In an **SQL join operation**, the **SQL JOIN operator** is used to combine parts or all of two or more tables.
  - **Explicit join** – the SQL JOIN operator is used as part of the SQL statement.
  - **Implicit join** – the SQL JOIN operator is ***not*** used as part of the SQL statement.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Querying Multiple Tables with SQL Joins (2 of 10)
SQL CROSS JOIN

- An **SQL CROSS JOIN** combines each row in one table with every row in another table.

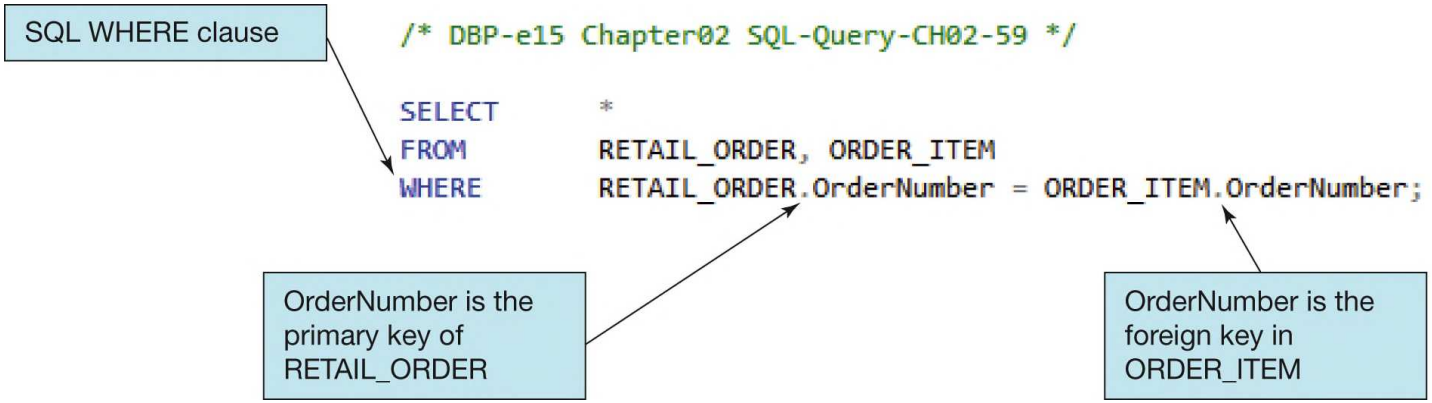
```
/* *** SQL-Query-CH02-58 *** */
SELECT  *
FROM    RETAIL_ORDER, ORDER_ITEM;
```

- This is known mathematically as a **Cartesian product** of the rows in the tables.
- This is **illogical in database work** because we only need rows that somehow logically correspond in the two tables.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Querying Multiple Tables with SQL Joins (3 of 10)
Implicit SQL INNER JOIN I

- By selecting rows by matching by the primary key values of one table with the foreign key values of a second table, we produce an **SQL INNER JOIN**.



- Because the **SQL JOIN keyword** does ***not*** appear in the SQL statement, this is an **implicit join**.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Querying Multiple Tables with SQL Joins (4 of 10)
Implicit SQL INNER JOIN II

```
/* *** SQL-Query-CH02-59 *** */
SELECT      *
FROM        RETAIL_ORDER, ORDER_ITEM
WHERE       RETAIL_ORDER.OrderNumber = ORDER_ITEM.OrderNumber;
```

	OrderNumber	StoreNumber	StoreZip	OrderMonth	OrderYear	OrderTotal	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	3000	10	98110	January	2018	480.00	3000	100200	1	300.00	300.00
2	2000	20	02335	December	2017	310.00	2000	101100	4	50.00	200.00
3	3000	10	98110	January	2018	480.00	3000	101100	2	50.00	100.00
4	2000	20	02335	December	2017	310.00	2000	101200	2	50.00	100.00
5	3000	10	98110	January	2018	480.00	3000	101200	1	50.00	50.00
6	1000	10	98110	December	2017	445.00	1000	201000	1	300.00	300.00
7	1000	10	98110	December	2017	445.00	1000	202000	1	130.00	130.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Querying Multiple Tables with SQL Joins (5 of 10)
Implicit SQL INNER JOIN III

- With an **SQL ORDER BY clause** for easier reading by OrderNumber:

```
/* *** SQL-Query-CH02-60 *** */
SELECT      *
FROM        RETAIL_ORDER, ORDER_ITEM
WHERE       RETAIL_ORDER.OrderNumber = ORDER_ITEM.OrderNumber
ORDER BY    RETAIL_ORDER.OrderNumber, ORDER_ITEM.SKU;
```

	OrderNumber	StoreNumber	StoreZip	OrderMonth	OrderYear	OrderTotal	OrderNumber	SKU	Quantity	Price	ExtendedPrice
1	1000	10	98110	December	2017	445.00	1000	201000	1	300.00	300.00
2	1000	10	98110	December	2017	445.00	1000	202000	1	130.00	130.00
3	2000	20	02335	December	2017	310.00	2000	101100	4	50.00	200.00
4	2000	20	02335	December	2017	310.00	2000	101200	2	50.00	100.00
5	3000	10	98110	January	2018	480.00	3000	100200	1	300.00	300.00
6	3000	10	98110	January	2018	480.00	3000	101100	2	50.00	100.00
7	3000	10	98110	January	2018	480.00	3000	101200	1	50.00	50.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Querying Multiple Tables with SQL Joins (6 of 10)
EQUIJOIN

- The process of using an **SQL JOIN operation** to join two tables is called **joining the two tables**.
- When tables are joined using an **equal to** condition as in the example above, the join is called an **equijoin**.
- When people say "***join***," 99.99999 percent of the time they mean ***equijoin***.
- Multiple joins can be used to process three or even more tables.
- All parts of the SQL SELECT statement syntax can be applied to the table displaying the results of an SQL query using joins.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Querying Multiple Tables with SQL Joins (7 of 10)
The Logic of Joins

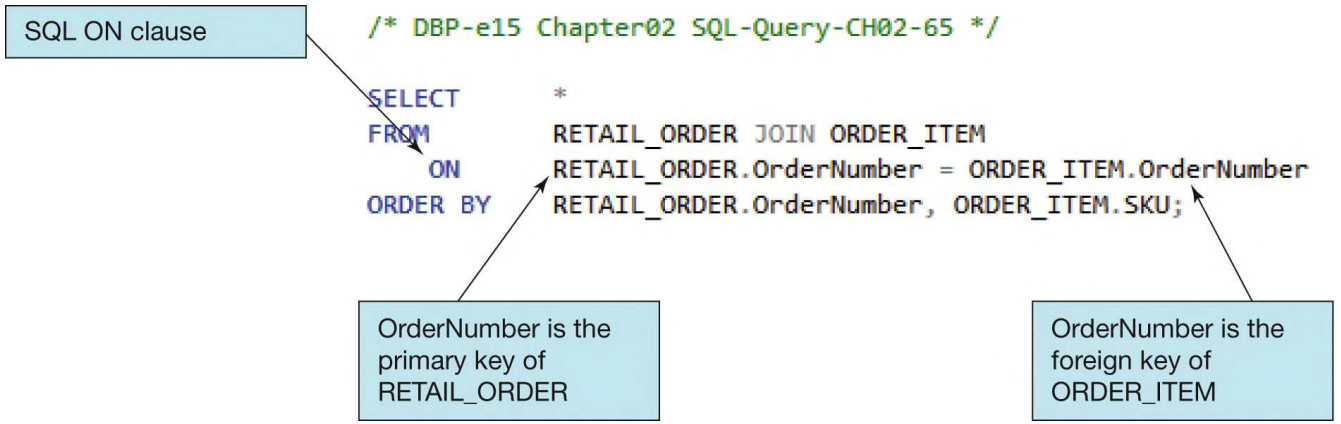
```
/* *** SQL-Query-CH02-62 *** */
SELECT     Buyer, SKU_DATA.SKU, SKU_Description,
           SUM(ExtendedPrice) AS BuyerSKURevenue
FROM       SKU_DATA, ORDER_ITEM
WHERE      SKU_DATA.SKU = ORDER_ITEM.SKU
GROUP BY   Buyer, SKU_DATA.SKU, SKU_Description
ORDER BY   BuyerSKURevenue DESC;
```

	Buyer	SKU	SKU_Description	BuyerSKURevenue
1	Pete Hansen	100200	Std. Scuba Tank, Magenta	300.00
2	Nancy Meyers	101100	Dive Mask, Small Clear	300.00
3	Cindy Lo	201000	Half-dome Tent	300.00
4	Nancy Meyers	101200	Dive Mask, Med Clear	150.00
5	Cindy Lo	202000	Half-dome Tent Vestibule	130.00

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Querying Multiple Tables with SQL Joins (8 of 10)
Explicit SQL INNER JOIN

- Selecting rows by matching by the primary key values of one table with the foreign key values of a second table, we produce an **SQL INNER JOIN**.



- Because the **SQL JOIN keyword *does*** appear in the SQL statement, this is an **explicit join**.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Querying Multiple Tables with SQL Joins (9 of 10)
SQL JOIN ON Syntax

- In **SQL JOIN ON syntax**:
  - The **SQL JOIN keyword** is placed between the table names in the **SQL FROM clause**, where it replaces the comma that previously separated the two table names.
  - The **SQL ON keyword** now leads into an **SQL ON clause**, which includes the statement of matching key values that was previously in an **SQL WHERE clause**.
  - The **SQL WHERE clause** is *no longer used as part of the join*, which makes it easier to read the actual restrictions on the rows in the query in the **SQL WHERE clause** itself.
- The **explicit SQL JOIN ON syntax** is currently considered as the proper way to write SQL join operations, and the older implicit SQL syntax is considered an archaic, older syntax (but it still works).

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Querying Multiple Tables with SQL Joins (10 of 10)
The Logic of Joins

```
/* *** SQL-Query-CH02-66 *** */
SELECT     *
FROM       RETAIL_ORDER JOIN ORDER_ITEM
     ON    RETAIL_ORDER.OrderNumber = ORDER_ITEM.OrderNumber
WHERE      OrderYear = 2017
ORDER BY   RETAIL_ORDER.OrderNumber, ORDER_ITEM.SKU;
```

	OrderNumber	StoreNumber	OrderYear	SKU	SKU_Description	Department
1	1000	10	2017	201000	Half-dome Tent	Camping
2	1000	10	2017	202000	Half-dome Tent Vestibule	Camping
3	2000	20	2017	101100	Dive Mask, Small Clear	Water Sports
4	2000	20	2017	101200	Dive Mask, Med Clear	Water Sports

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL Subqueries versus SQL Joins

- **SQL subqueries** and **SQL joins** both process multiple tables.
- An **SQL subquery** can only be used to retrieve data from the "top table."
- An **SQL join** can be used to obtain data from any number of tables, including the "top table" of the subquery.
- In Chapter 8, we will study the **correlated subquery**. That kind of subquery can do work that is not possible with joins.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL Queries on Recursive Relationships (1 of 3)

- **Recursive** means that something recurs or repeats.
- A **recursive procedure** (in computer programming languages) is a block of code that calls itself.
- A ***recursive relationship*** (in database structures) is a relationship between two columns in the same table.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL Queries on Recursive Relationships (2 of 3)

In the Cape Codd Outdoor Sports data extract below, the BUYER table has a recursive relationship: the Supervisor column holds value of the BuyerName column as data and serves as a foreign key referencing BuyerName as the associated primary key.

```
/* *** SQL-Query-CH02-70 *** */
SELECT     BuyerName, Department, Position, Supervisor
FROM       BUYER
ORDER BY   Position DESC;
```

	BuyerName	Department	Position	Supervisor
1	Mary Smith	Purchasing	Manager	NULL
2	Pete Hansen	Purchasing	Buyer 3	Mary Smith
3	Cindy Lo	Purchasing	Buyer 2	Mary Smith
4	Jerry Martin	Purchasing	Buyer 1	Cindy Lo
5	Nancy Meyers	Purchasing	Buyer 1	Pete Hansen

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL Queries on Recursive Relationships (3 of 3)

The solution is to use table aliases to create two aliased versions of the same single table. Thus, the query below lists each BuyerName and that person's Supervisor:

```
/* *** SQL-Query-CH02-71 *** */
SELECT     S.BuyerName AS SupervisorName, S.Position AS
           SupervisorPosition, B.BuyerName, B.Position
FROM       BUYER S JOIN BUYER B
   ON      S.BuyerName = B.Supervisor ORDER BY S.BuyerName;
```

	SupervisorName	SupervisiorPosition	BuyerName	Position
1	Cindy Lo	Buyer 2	Jerry Martin	Buyer 1
2	Mary Smith	Manager	Cindy Lo	Buyer 2
3	Mary Smith	Manager	Pete Hansen	Buyer 3
4	Pete Hansen	Buyer 3	Nancy Meyers	Buyer 1

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Figure 2-32 (1 of 4)
The Logic of Outer Joins – Example Tables

STUDENT

StudentPK	StudentName	LockerFK
1	Adams	NULL
2	Buchanan	NULL
3	Carter	10
4	Ford	20
5	Hoover	30
6	Kennedy	40
7	Roosevelt	50
8	Truman	60

LOCKER

LockerPK	LockerType
10	Full
20	Full
30	Half
40	Full
50	Full
60	Half
70	Full
80	Full
90	Half

(a) The STUDENT and LOCKER Tables Aligned to Show Row Relationships

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-32 (2 of 4)
INNER JOIN (STUDENT and LOCKER Tables)

Only the rows where LockerFK=LockerPK are shown—Note that some StudentPK and some LockerPK values are not in the results

StudentPK	StudentName	LockerFK	LockerPK	LockerType
3	Carter	10	10	Full
4	Ford	20	20	Full
5	Hoover	30	30	Half
6	Kennedy	40	40	Full
7	Roosevelt	50	50	Full
8	Truman	60	60	Half

(b) INNER JOIN of the STUDENT and LOCKER Tables

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-32 (3 of 4)
LEFT OUTER JOIN (STUDENT and LOCKER Tables)

All rows from STUDENT are shown, even where there is no matching LockerFK=LockerPK value

StudentPK	StudentName	LockerFK	LockerPK	LockerType
1	Adams	NULL	NULL	NULL
2	Buchanan	NULL	NULL	NULL
3	Carter	10	10	Full
4	Ford	20	20	Full
5	Hoover	30	30	Half
6	Kennedy	40	40	Full
7	Roosevelt	50	50	Full
8	Truman	60	60	Half

(c) LEFT OUTER JOIN of the STUDENT and LOCKER Tables

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-32 (4 of 4)
RIGHT OUTER JOIN (STUDENT and LOCKER Tables)

All rows from LOCKER are shown, even where there is no matching LockerFK=LockerPK value

StudentPK	StudentName	LockerFK	LockerPK	LockerType
3	Carter	10	10	Full
4	Ford	20	20	Full
5	Hoover	30	30	Half
6	Kennedy	40	40	Full
7	Roosevelt	50	50	Full
8	Truman	60	60	Half
NULL	NULL	NULL	70	Full
NULL	NULL	NULL	80	Full
NULL	NULL	NULL	90	Half

(d) RIGHT OUTER JOIN of the STUDENT and LOCKER Tables

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL RIGHT OUTER JOIN Example

```
/* *** SQL-Query-CH02-77 *** */
SELECT     OI.OrderNumber, Quantity, SD.SKU, SKU_Description,
           Department, Buyer
FROM       ORDER_ITEM AS OI RIGHT OUTER JOIN SKU_DATA AS SD
   ON      OI.SKU = SD.SKU
ORDER BY   OI.OrderNumber, SD.SKU;
```

	OrderNumber	Quantity	SKU	SKU_Description	Department	Buyer
1	NULL	NULL	100100	Std. Scuba Tank, Yellow	Water Sports	Pete Hansen
2	NULL	NULL	100300	Std. Scuba Tank, Light Blue	Water Sports	Pete Hansen
3	NULL	NULL	100400	Std. Scuba Tank, Dark Blue	Water Sports	Pete Hansen
4	NULL	NULL	100500	Std. Scuba Tank, Light Green	Water Sports	Pete Hansen
5	NULL	NULL	100600	Std. Scuba Tank, Dark Green	Water Sports	Pete Hansen
6	NULL	NULL	203000	Half-dome Tent Vestibule - Wide	Camping	Cindy Lo
7	NULL	NULL	301000	Light Fly Climbing Harness	Climbing	Jerry Martin
8	NULL	NULL	302000	Locking Carabiner, Oval	Climbing	Jerry Martin
9	1000	1	201000	Half-dome Tent	Camping	Cindy Lo
10	1000	1	202000	Half-dome Tent Vestibule	Camping	Cindy Lo
11	2000	4	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
12	2000	2	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers
13	3000	1	100200	Std. Scuba Tank, Magenta	Water Sports	Pete Hansen
14	3000	2	101100	Dive Mask, Small Clear	Water Sports	Nancy Meyers
15	3000	1	101200	Dive Mask, Med Clear	Water Sports	Nancy Meyers

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Mathematical Set Theory

- Mathematicians use the term **set theory** to describe mathematical operations on sets, where a **set** is defined as a group of distinct items.
- A relational database table meets the definition of a set, so it is little wonder that SQL includes a group of **set operators** for use with SQL queries.

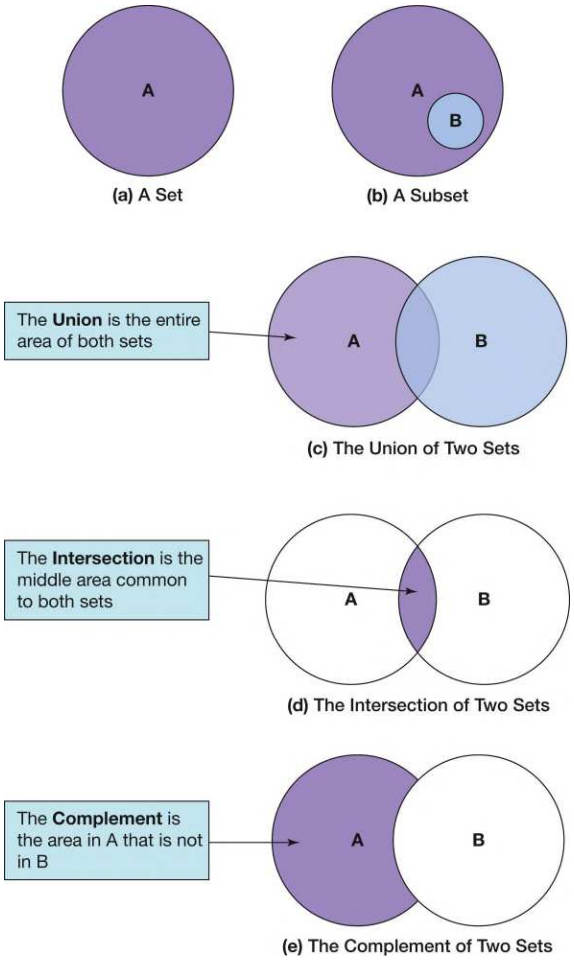
Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Venn Diagrams (1 of 2)

- **Venn diagrams** are the standard method of visualizing sets and their relationships.
  - A **set** is represented by a labeled circle.
  - A **subset** is a portion of a set that is contained entirely within the set.
  - The **union** of two sets represents a set that contains all values in both sets. This is equivalent to an **OR** logical operation (**A OR B**).
  - The **intersection** of two sets represents the area common to both sets. This is equivalent to an **AND** logical operation (**A AND B**).
  - The **complement** of set B in set A represents everything in set A that is not in set B. This is equivalent to a logical operation using **NOT** (**A NOT B**).

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Venn Diagrams (2 of 2)



Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Figure 2-34
SQL Set Operators

Note that in order to use **SQL set operators**, the table columns involved in the operations ***must*** be the same number in each SELECT component, and corresponding columns ***must*** have the same or compatible (e.g., CHAR and VARCHAR) data types!

SQL Set Operators	
Operator	**Meaning**
UNION	The result is all the row values in one or both tables
INTERSECT	The result is all the row values common to both tables
EXCEPT	The result is all the row values in the first table but not the second

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL UNION Operator

"What products were available for sale (by either catalog or Web site) in 2017 and 2018?"

```
/* *** SQL-Query-CH02-78 *** */
SELECT     SKU, SKU_Description, Department
FROM       CATALOG_SKU_2017
UNION
SELECT     SKU, SKU_Description, Department
FROM       CATALOG_SKU_2018;
```

	SKU	SKU_Description	Department
1	100100	Std. Scuba Tank, Yellow	Water Sports
2	100200	Std. Scuba Tank, Magenta	Water Sports
3	100300	Std. Scuba Tank, Light Blue	Water Sports
4	100400	Std. Scuba Tank, Dark Blue	Water Sports
5	101100	Dive Mask, Small Clear	Water Sports
6	101200	Dive Mask, Med Clear	Water Sports
7	201000	Half-dome Tent	Camping
8	202000	Half-dome Tent Vestibule	Camping
9	203000	Half-dome Tent Vestibule - Wide	Camping
10	301000	Light Fly Climbing Harness	Climbing
11	302000	Locking Carabiner, Oval	Climbing

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL ALL Keyword

If we compare the output of SQL-Query-CH02-78 to the data in the CATALOG-SKU_2017 and CATALOG_SKU_2018, we will note that there are **no duplicate rows in the query output**. For example, SKU 201000, the Half-Dome Tent, is in each table, but only appears once in the query output. If, for some reason, we **wanted the duplicated rows to be displayed in the query output** as well, we would simply add the **SQL ALL keyword** to the query:

```
/* *** SQL-Query-CH02-78-ALL *** */
SELECT     SKU, SKU_Description, Department
FROM       CATALOG_SKU_2017
UNION ALL
SELECT     SKU, SKU_Description, Department
FROM       CATALOG_SKU_2018;
```

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

SQL INTERSECT Operator

- "What products were available for sale (by either catalog or Web site) in *both* 2017 and 2018?" [MySQL 5.6 does not support the INTERSECT Operator]

```
/* *** SQL-Query-CH02-79 *** */
SELECT     SKU, SKU_Description, Department
FROM       CATALOG_SKU_2017
INTERSECT
SELECT     SKU, SKU_Description, Department
FROM       CATALOG_SKU_2018;
```

	SKU	SKU_Description	Department
1	100100	Std. Scuba Tank, Yellow	Water Sports
2	101100	Dive Mask, Small Clear	Water Sports
3	101200	Dive Mask, Med Clear	Water Sports
4	201000	Half-dome Tent	Camping
5	202000	Half-dome Tent Vestibule	Camping
6	301000	Light Fly Climbing Harness	Climbing
7	302000	Locking Carabiner, Oval	Climbing

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

SQL EXCEPT Operator

"What products were available for sale (by either catalog or Web site) in 2017 but *not* in 2018?" [Oracle Database calls this the **SQL MINUS operator**, and MySQL 5.6 does not support this operation]

```
/* *** SQL-Query-CH02-80 *** */
SELECT     SKU, SKU_Description, Department
FROM       CATALOG_SKU_2017
EXCEPT
SELECT     SKU, SKU_Description, Department
FROM       CATALOG_SKU_2018;
```

	SKU	SKU_Description	Department
1	100300	Std. Scuba Tank, Light Blue	Water Sports
2	100400	Std. Scuba Tank, Dark Blue	Water Sports

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved

Database Processing
Fundamentals, Design, and Implementation (15th Edition)

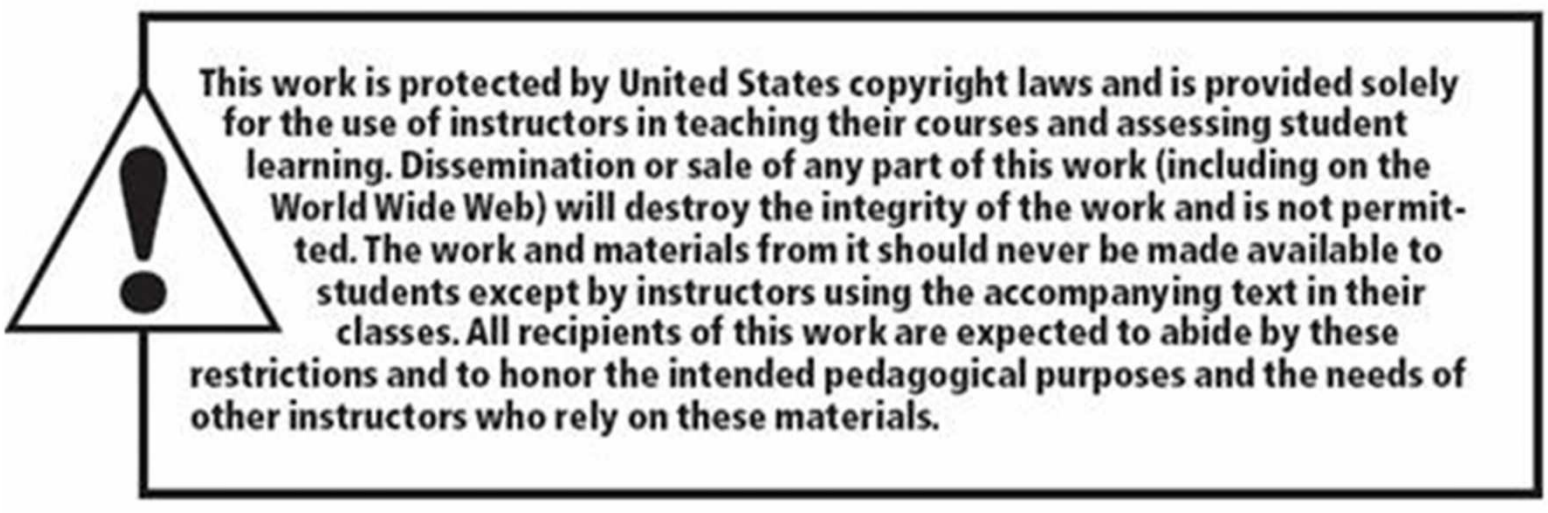
End of Presentation:

Chapter Two

Part 2

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved
Database Processing Fundamentals Design and Implementation 15th edition Kroenke Solutions Manual

Copyright



This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.

Pearson
Copyright © 2019, 2016, 2014 Pearson Education, Inc. All Rights Reserved