MySQL for Beginners

Activity Guide

D61918GC30
Edition 3.0
May 2013
D82066

ORACLE

Author
Mark Lewin

Technical Contributors and Reviewers
Andrew Morgan, Martin Hansson, Bob Falsco, Mat Keep, Georgi Kodinov, Mike Lischke, Guilhem Bichot, Ford Brockman, Sanjay Manwani, Roy Lyseng, Craig McBride, Jeremy Smyth

This book was published using: Oracle Tutor

Table of Contents

Practices for Lesson 1: Introduction to MySQL 1-1
Practices for Lesson 1: Overview 1-2
Practice 1-1: Accessing MySQL Resources Online 1-3
Solution 1-1: Accessing MySQL Resources Online 1-5
Practices for Lesson 2: MySQL Server and Client 2-1
Practices for Lesson 2: Overview 2-2
Practice 2-1: Installing and Starting the MySQL Server 2-3
Solution 2-1: Installing and Starting the MySQL Server 2-8
Practice 2-2: Using the Keyboard Editing and Tee Commands 2-9
Solution 2-2: Using the Keyboard Editing and Tee Commands 2-13
Practice 2-3: Creating the world_innodb Database 2-14
Solutions 2-3: Creating the world_innodb Database 2-15
Practices for Lesson 3: MySQL Database Basics 3-1
Practices for Lesson 3: Overview 3-2
Practice 3-1: Quiz – Database Basics 3-3
Solutions 3-1: Quiz – Database Basics 3-5
Practice 3-2: Identifying the Structure of a Table 3-6
Solutions 3-2: Identifying the Structure of a Table 3-7
Practices for Lesson 4: Database Design 4-1
Practices for Lesson 4: Overview 4-2
Practice 4-1: Quiz – Database Design 4-3
Solutions 4-1: Quiz – Database Design 4-6
Practice 4-2: Evaluating a Database 4-7
Solutions 4-2: Evaluating a Database 4-8
Practice 4-3: Creating a Structure Diagram 4-13
Solution 4-3: Creating a Structure Diagram 4-15
Practices for Lesson 5: Table Data Types 5-1
Practices for Lesson 5: Overview 5-2
Practice 5-1: Quiz – Data Types 5-3
Solutions 5-1: Quiz – Data Types 5-5
Practice 5-2: Explaining the Use of Data Types 5-6
Solutions 5-2: Explaining the Use of Data Types 5-7
Practices for Lesson 6: Database and Table Creation 6-1
Practices for Lesson 6: Overview 6-2
Practice 6-1: Displaying Table Creation Information 6-3
Solutions 6-1: Displaying Table Creation Information 6-4
Practice 6-2: Creating a Database 6-7
Solutions 6-2: Creating a Database 6-11
Practices for Lesson 7: Basic Queries 7-1
Practices for Lesson 7: Overview 7-2
Practice 7-1: Performing Basic Queries 7-3
Solutions 7-1: Performing Basic Queries 7-4
Practice 7-2: Perform Basic Queries Using MySQL Workbench 7-12
Solution 7-2: Perform Basic Queries Using MySQL Workbench 7-14
Practice 7-3: Perform Basic Queries on the Pets Database 7-15
Solution 7-3: Perform Basic Queries on the Pets Database 7-16

Practices for Lesson 8: Database and Table Maintenance 8-1
Practices for Lesson 8: Overview....8-2
Practice 8-1: Removing a Database....8-3
Solutions 8-1: Removing a Database....8-4
Practice 8-2: Creating a New Table and Removing a Table....8-5
Solutions 8-2: Creating a New Table and Removing a Table....8-6
Practice 8-3: Altering Table Columns....8-9
Solutions 8-3: Altering Table Columns....8-10
Practice 8-4: Modifying Table Indexes and Constraints....8-12
Solutions 8-4: Modifying Table Indexes and Constraints....8-13
Practice 8-5: Further Practice....8-16
Solutions 8-5: Further Practice....8-17
Practices for Lesson 9: Table Data Manipulation....9-1
Practices for Lesson 9: Overview....9-2
Practice 9-1: Inserting and Replacing Table Row Data....9-3
Solutions 9-1: Inserting and Replacing Table Row Data....9-4
Practice 9-2: Modifying and Deleting Table Row Data....9-6
Solutions 9-2: Modifying and Deleting Table Row Data....9-7
Practice 9-3: Manipulating Table Row Data in the Pets Database....9-10
Solutions Practice 9-3: Manipulating Table Row Data in the Pets Database....9-13
Practices for Lesson 10: Functions....10-1
Practices for Lesson 10: Overview....10-2
Practice 10-1: Quiz – Functions....10-2
Solutions 10-1: Quiz – Functions....10-4
Practice 10-2: Using Built-In, String, and Temporal Functions....10-5
Solutions 10-2: Using Built-In, String, and Temporal Functions....10-6
Practice 10-3: Using Numeric and Control Flow Functions....10-8
Solutions 10-3: Using Numeric and Control Flow Functions....10-9
Practice 10-4: Using Aggregate Functions....10-11
Solutions 10-4: Using Aggregate Functions....10-12
Practices for Lesson 11: Exporting and Importing Data....11-1
Practices for Lesson 11: Overview....11-2
Practice 11-1: Quiz – Exporting and Importing Data....11-3
Solutions 11-1: Quiz – Exporting and Importing Data....11-5
Practice 11-2: Exporting Files by Using a Query....11-6
Solutions 11-2: Exporting Files by Using a Query....11-7
Practice 11-3: Importing Files from a Data File....11-10
Solutions 11-3: Importing Files from a Data File....11-11
Practice 11-4: Backing Up Database Files with a Utility....11-13
Solutions 11-4: Backing Up Database Files with a Utility....11-14
Practices for Lesson 12: Joining Tables....12-1
Practices for Lesson 12: Overview....12-2
Practice 12-1: Performing Inner and Outer Joins....12-3
Solutions 12-1: Performing Inner and Outer Joins....12-4
Practice 12-2: Creating Queries Requiring Joins....12-8
Solutions 12-2: Creating Queries Requiring Joins....12-9
Optional Practice 12-3: Additional Join Practice....12-15
Solutions 12-3: Additional Join Practice....12-16

Practices for Lesson 13: Table Subqueries 13-1
Practices for Lesson 13: Overview 13-2
Practice 13-1: Performing Different Types of Subqueries 13-3
Solutions 13-1: Performing Different Types of Subqueries 13-4
Practice 13-2: Performing Advanced Subqueries 13-7
Solutions 13-2: Performing Several Advanced Subqueries 13-8
Practices for Lesson 14: MySQL Graphical User Interface Tools 14-1
Practices for Lesson 14: Overview 14-2
Practice 14-1: Creating a Data Model by Using MySQL Workbench 14-3
Practice 14-2: Creating a Server Instance by Using MySQL Workbench 14-6
Practice 14-3: Viewing MySQL Enterprise Monitor Demonstrations 14-8
Optional Practice 14-4: Viewing the MySQL Workbench Demonstration 14-9
Optional Practice 14-5: Creating a Model for the Pets Database by Using MySQL Workbench 14-10
Practices for Lesson 15: Supplemental Information 15-1
Practices for Lesson 15: Overview 15-2
Practice 15-1: Displaying Storage Engine Information 15-3
Solutions 15-1: Displaying Storage Engine Information 15-4
Practice 15-2: Displaying and Creating Views 15-7
Solutions 15-2: Displaying and Creating Views 15-8
Practice 15-3: Obtaining Metadata 15-10
Solutions 15-3: Obtaining Metadata 15-11
OPTIONAL Practice 15-4: Creating a Backup of MySQL Databases 15-15
OPTIONAL Practice 15-4: Create a Backup of MySQL Databases 15-15
OPTIONAL Solutions 15-4: Creating a Backup of MySQL Databases 15-16
OPTIONAL Solutions 15-4: Create a Backup of MySQL Databases 15-16
Practices for Lesson 16: Conclusion 16-1
Practices for Lesson 16 16-2
Appendix A: Glossary of MySQL Terms 17-1
Glossary of MySQL Terms: Overview 17-2
Appendix B: Practice Solution Scripts 18-1
Lesson 2: MySQL Server and Client 18-2
Lesson 3: Database Basics 18-4
Lesson 4: Database Design 18-5
Lesson 5: Data Types 18-6
Lesson 6: Database and Table Creation 18-7
Lesson 7: Basic Queries 18-9
Lesson 8: Database and Table Maintenance 18-12
Lesson 9: Table Data Manipulation 18-16
Lesson 10: Functions 18-20
Lesson 11: Exporting and Importing Data 18-22
Lesson 12: Joining Tables 18-24
Lesson 13: Table Subqueries 18-28
Lesson 14: MySQL GUIs 18-31
Lesson 15: Supplemental Information 18-32

Practices for Lesson 1: Introduction to MySQL

Chapter 1

Practices for Lesson 1: Overview

Practices Overview

This practice introduces you to online MySQL resources.

Assumptions

- A web browser is available for connection to the Internet.
- Your web browser allows connection to MySQL and Oracle websites.

Practice 1-1: Accessing MySQL Resources Online

Overview

In this practice, you review common web pages from the MySQL and Oracle websites that contain information about MySQL products and services.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. Open the MySQL website:
   In your web browser, go to the following URL: www.mysql.com.
   The MySQL home page appears.
2. Open the Products web page from the MySQL home page:
   a. Click the Products tab located at the top of the page.
   b. Scroll down the page to review the product information provided.
      A list of currently available products is presented (in the left navigation menu) with links for further information.
3. Open the MySQL Enterprise Edition web page from the Products page:
   a. Click the MySQL Enterprise Edition link.
   b. Scroll down the page to review the details of MySQL Enterprise Edition.
      Note the extended list of products (in the left navigation menu) with links for further information.
   c. Visit the links for Enterprise Monitor, Enterprise Backup, and Workbench to find out more about these MySQL Enterprise Edition tools.
4. Open the Services web page:
   a. Click the Services tab located at the top of the page.
   b. Scroll down the page to review the service information provided, with links for further information.
5. Open the Training web page:
   a. Click the Learn More link below the MySQL Training title in the main window.
      A list of current training courses available appears.
   b. Select your country from the pull-down menu and click the Go button to be taken to the Oracle training website for your country.
   c. Review the information on courses, learning paths, certification, and purchase options.
6. Open the Support web page:
   a. Go back to the Training page and click the Support link in the left navigation menu.
   b. Scroll down the page to review the support information provided, with links for further information.
   c. Note the Resources area, which links to Oracle website pages for additional information and purchase.

7. Open the MySQL Developer Zone website:

   In your web browser, go to the following URL: dev.mysql.com.

   The MySQL Developer Zone (also known as the MySQL Community) home page appears. This website contains many links to more information and access to a variety of open communication forums. Notice the Downloads and Documentation tabs across the top of the page.

8. Open the News and Events page from MySQL Developer Zone:
   a. Click the News and Events link located at the top of the page.
   b. Use the left navigation menu to review the latest MySQL happenings, from news stories to upcoming seminars.
9. Open the Oracle Products and Services: MySQL web page:
   a. In your web browser, go to the following URL: www.oracle.com/us/products/mysql/.
   b. Scroll down the page to review the MySQL product information provided.

      A list of MySQL products and services appears, with links to further information.
10. Exit the web browser.

Solution 1-1: Accessing MySQL Resources Online

There are no solutions for this practice. See the practice tasks instructions.

Practices for Lesson 2: MySQL Server and Client

Chapter 2

Practices for Lesson 2: Overview

Practices Overview

These practices test your knowledge of the MySQL server and client installation. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you may have to adjust file locations.

Assumptions

- The MySQL Installer file (`mysql-installer-commercial-5.6.10.0.msi`) for the MySQL bundle (including the MySQL server) is in the `D:\stage\MySQL` directory.
- The Microsoft .NET Framework 4 Client Profile has been installed.
- MySQL Enterprise Backup 3.8 and MySQL Enterprise Monitor are installed.
- The `world_innodb` database file (`world_innodb.sql`) is in the `D:\labs` directory.

Practice 2-1: Installing and Starting the MySQL Server

Overview

In this practice, you use MySQL Installer to install and configure MySQL. It installs the following components:

- MySQL Server 5.6.10 (Commercial/Enterprise Edition)
- MySQL Workbench SE 5.2.46
- MySQL Notifier 1.0.3
- MySQL Connector/ODBC 5.2.4
- MySQL Connector/C++ 1.1.2
- MySQL Connector/J 5.1.23
- MySQL Connector/NET 6.6.5
- MySQL Documentation 5.6.10
- MySQL Samples and Examples 5.6.10

Duration

This practice takes approximately 30 minutes to complete.

Tasks

1. Go to the `D:\stage\MySQL` directory by using Windows Explorer.
2. Execute the MySQL Installer by double-clicking the `mysql-installer-commercial-5.6.10.0.msi` installation file.
3. When the installer has finished loading, the Welcome window appears. Select Install MySQL Products.
4. In the License Agreement window, select the check box to accept the terms, and then click the Next button.
5. In the Choosing a Setup Type window:
   a. Select Developer Default.
   b. Enter (or confirm) the “Installation Path” as: `D:\Program Files\MySQL\`.
   c. Enter (or confirm) the “Data Path” as: `D:\ProgramData\MySQL\MySQL Server 5.6\`.
   d. Click the Next button.
      The Check Requirements window appears.
6. The Check Requirements window lists any dependencies on external software:
   - Microsoft .NET Framework 4 (already installed)
   - Microsoft Visual C++ 2010 32-bit run time
   - Microsoft Excel 2007 or greater
   - Visual Studio Tools for Office 2010 run time
7. The green check mark next to .NET Framework 4 shows it is already installed. You need to install the other items in the list.

8. Click the Execute button to install Microsoft Visual C++ 2010 32-bit run time:
   a. Select the check box to accept the license terms.
   b. Click the Install button.
   c. When the Installation is Complete window appears, click the Finish button.
9. There is now a blue arrow next to Microsoft Excel 2007 or greater in the Check Requirements window.
   a. Click the Execute button.

      MySQL Installer checks if Microsoft Excel or Visual Studio are installed. They are not, so it refreshes the list of requirements to exclude these dependencies.

      The required software (Microsoft .NET Framework 4 Client Profile and Microsoft Visual C++ 2010 32-bit runtime) has now been installed and the MySQL installation can proceed.
   b. Click Next.
10. The Installation Progress window lists all MySQL products to be installed:
    - MySQL Server 5.6.10
    - MySQL Workbench SE 5.2.46
    - MySQL Notifier 1.0.3
    - Connector/ODBC 5.2.4
    - Connector/C++ 1.1.2
    - Connector/J 5.1.23
    - Connector/NET 6.6.5
    - MySQL Documentation 5.6.10
    - Samples and Examples 5.6.10.
11. Install the MySQL products:
    a. Click the Execute button.

       The installation process takes a few minutes. When complete, the Status column for all products shows “Install success”.
    b. Click the Next button.
12. Configure the software components in the Configuration Overview window:

    A green arrow points to MySQL Server 5.6.10. Click Next to configure MySQL Server 5.6.10.
13. In the MySQL Server Configuration (1/3) window:
    a. Server Configuration Type: From the Config Type drop-down list, select Development Machine.
    b. Select (or confirm) Enable TCP/IP Networking and Port Number of 3306.
    c. Select (or confirm) “Open Firewall port for network access”.
    d. Click Next.
14. In the MySQL Server Configuration (2/3) window:
    a. In Root Account Password, enter and confirm the password `oracle`.
    b. Click the Next button.
15. In the MySQL Server Configuration (3/3) window:
    a. Enter (or confirm) Windows Service Name is `MySQL56`.
    b. Select (or confirm) “Start the MySQL Server at System Startup”.

c. Select (or confirm) Run Windows Service as “Standard System Account”.
d. Click Next.

16. The Configuration Overview window appears and shows the configuration progress for the MySQL server.

    This takes a few moments. When complete, a green tick appears next to MySQL Server 5.6.10 and the “Action to be performed” column reads “Configuration Complete”.

17. There is now a green arrow next to Samples and Examples, and the Action to be performed column reads “Initial Configuration”.
    a. Click Next to configure Samples and Examples.
    b. Configuration takes a minute or two. When complete, click Next.
18. In the Installation complete window:
    a. Deselect the Start MySQL Workbench after Setup check box.
    b. Click the Finish button.

    The final MySQL Installer window closes. You have installed and configured MySQL Server and the other tools required for this course.

19. Confirm the installation:
    a. Click the Windows Start button (in the lower left corner of the Windows desktop).
    b. Select All Programs.
    c. Select MySQL. The folder includes the following programs:
       - MySQL Workbench 5.2 SE
       - MySQL Connector/Net 6.6.5
       - MySQL Enterprise Backup 3.8 (pre-installed)
       - MySQL Enterprise Monitor (pre-installed)
       - MySQL Installer
       - MySQL Notifier 1.0.3
       - MySQL Server 5.6

    Note: MySQL Enterprise Backup and MySQL Enterprise Monitor are pre-installed for you.

20. Attempt to start the `mysql` client from the Windows command prompt:
    a. Click the Start button (at the lower-left of the screen).
    b. Select Run.
    c. In the Open field, enter `cmd` and press Enter.

    A command-prompt window opens.

    Note: You can also use the Command Prompt shortcut icon on the desktop.

    The prompt shows the current working directory, for example:

    ```
    D:\Users\Administrator>
    ```

    Note: The practice steps use `cmd>` to refer to this prompt from now on. Do not attempt to type the prompt, just enter the commands that appear in bold after it:

    ```
    cmd> enter commands here
    ```

    d. Enter the following at the command prompt:

    ```
    cmd> mysql
    ```

    e. Press Enter.

Windows reports that it cannot find the `mysql` program:

```
cmd> mysql
'mysql' is not recognized as an internal or external command,
operable program or batch file.
```

This is because Windows has not been told where to look for the `mysql` program. So that Windows recognizes the `mysql` client program, you need to add it to the PATH environment variable.

f. Type `exit` and then press Enter to close the command-prompt window.

21. Add the `mysql` client program to the PATH:
    a. Click the Windows Start button.
    b. Right-click the Computer link.
    c. From the context menu that appears, select Properties.
    d. Select Advanced System Settings.
    e. Click the Advanced tab.
    f. Click the "Environment variables" button.

       The Environment variables window opens.
    g. From the list of System variables at the bottom of this window, select Path.
    h. Click Edit.

       The Edit System Variable window opens.
    i. Click somewhere in the Variable Value field and press the End button to move to the end of the line.
    j. Type the following exactly as shown:

```
;D:\Program Files\MySQL\MySQL Server 5.6\bin;
```

       – **Note:** include the semicolons at the beginning and end of the file path.
    k. Click OK to exit the Edit System Variable window.
    l. Click OK to exit the Environment Variables window.
    m. Click OK to exit the System Properties window.
    n. Close the System Window.
22. Connect to the MySQL server by using the `mysql` client program:
    a. Click the Windows Start button.
    b. Select Run.
    c. In the Open field, enter `cmd` and press Enter.

       A command-prompt window opens, with the prompt located at the current user's "home" directory.
    d. Enter the following at the command prompt:

```
cmd> mysql -u root -p
Enter password: oracle

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.6.10-enterprise-commercial-advanced MySQL Enterprise
Server - Advanced Edition (Commercial)
```

```
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

- In this course, the username is `root` and the password used is `oracle`. As you type the password, it appears as ****** on your screen.
- The current version of the MySQL server is displayed.
- The "MySQL connection id" might differ on your machine.
- When the client is started, the standard command prompt is replaced by the `mysql>` prompt.

23. Exit the `mysql` client:

    Enter the following from the `mysql>` prompt:

```
mysql> EXIT
```

    The following message is displayed and the standard command prompt is returned:

```
Bye
cmd>
```

Solution 2-1: Installing and Starting the MySQL Server

There are no solutions for this practice. See the practice instructions.

Practice 2-2: Using the Keyboard Editing and Tee Commands

Overview

In this practice, you set some sensible defaults for the command-line window and then use the keyboard editing methods to work with the `mysql` client. You also create a `tee` file to document all `mysql` session activities.

Duration

This practice takes approximately 5 minutes to complete.

Tasks

Note: Execute each command with the Enter or Return key, unless otherwise specified.

1. Change the default look and behavior of the Windows command prompt:
   a. Right-click the command-prompt window's title bar and select Defaults from the context menu. The Console Windows Properties dialog box appears.
   b. Select the Options tab and make the following change:
      - Edit Options: Select the QuickEdit Mode check box.
   c. Select the Font tab and make the following changes:
      - Font: Select Consolas.
      - Size: Select 18.
   d. Select the Layout tab and make the following changes:
      - Screen Buffer Size: Width: 160, Height: 300
      - Window Size: Width: 160, Height: 40
   e. (Optional) Select the Colors tab.
      - Select the Screen Text radio button.
      - Select a font color from the color palette.
      - Select the Screen Background radio button.
      - Select a background color from the color palette.
   f. Click OK to close the "Console Windows Properties" window.
   g. Close the command-prompt window.
2. Re-open the command-prompt window to apply the changes you made:
   a. Click the Windows Start button.
   b. Select Run.
   c. In the Open field, enter `cmd` and press Enter.
      The new settings are applied.
3. Connect to the MySQL server by using the `mysql` client program.
   Enter the following at the command prompt:

```
cmd> mysql -u root -p
Enter password: oracle
```

4. View help for the `mysql` client commands (non-SQL) by using the `Help` command (`\h`).
   Execute the following statement:

```
mysql> \h
```

It returns the following list of commands and their descriptions:

```
...
List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?         (\?) Synonym for `help'.
clear     (\c) Clear the current input statement.
connect   (\r) Reconnect to the server. Optional arguments are db and
host.
delimiter (\d) Set statement delimiter.
ego       (\G) Send command to mysql server, display result
vertically.
exit      (\q) Exit mysql. Same as quit.
go        (\g) Send command to mysql server.
help      (\h) Display this help.
notee     (\t) Don't write into outfile.
print     (\p) Print current command.
prompt    (\R) Change your mysql prompt.
quit      (\q) Quit mysql.
rehash    (\#) Rebuild completion hash.
source    (\.) Execute an SQL script file. Takes a file name as an
argument.
status    (\s) Get status information from the server.
tee       (\T) Set outfile [to_outfile]. Append everything into given
outfile.
use       (\u) Use another database. Takes database name as argument.
charset   (\C) Switch to another charset. Might be needed for
processing binlog with multi-byte charsets.
warnings  (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.

For server side help, type 'help contents'
```

5. Create a `tee` file to log your `mysql` client session:

   Execute the following statement at the prompt:

```
mysql> tee D:\labs\Lesson2_tee.txt
```

   You get a message that confirms the creation of the `tee` file:

```
Logging to file 'D:\labs\Lesson2_tee.txt'
```

6. Use the keyboard to re-run an earlier command:

   Retrieve the help command by pressing the up-arrow key ↑ twice and execute it again.

   This returns the command before the last command issued at the prompt which, in this case, is the `\h` command.

7. Use the keyboard to return to the very last command entry:

   Press the down-arrow key ↓ once to retrieve the `tee` command. Do not press Enter.

8. Use the keyboard to position the cursor within the command:

   Press the left-arrow ← key four times to position the cursor on the dot "`.`" in the file name.

   This demonstrates the use of the arrow keys to complete command-line edits by positioning the cursor for changes.

9. Use a keyboard editing key to delete portions of a command:

   Using the Backspace key, delete the current `Lesson2_tee` part of the `tee` file name and replace it with `Test`, leaving the rest of the line intact.

```
mysql> tee D:\labs\Test.txt
```

   a. Press the End key to move the cursor to the end of the command.
   b. Press the Enter key to execute the edited command.

10. Cancel the `Test.txt` `tee` file:

    Issue the `notee` command:

```
mysql> notee
```

    It returns the following to confirm the creation of the file:

```
Outfile disabled.
```

11. Return to the first **`tee`** command by pressing the up-arrow key ↑ four times.

    Re-execute the command to restart the `tee` file logging:

```
mysql> tee D:\labs\Lesson2_tee.txt
```

    It returns the following to confirm the creation of the file:

```
Logging to file 'D:\labs\Lesson2_tee.txt'
```

12. Start to cancel the `tee` file, but use the `mysql` command, which aborts the command execution:

    Issue the `notee` command, and then add `\c` to abort:

```
mysql> notee \c
```

    It returns the prompt without executing the command:

```
mysql>
```

13. View the current `Lesson2_tee.txt` file:
    a. Using Windows Explorer, go to the `D:\labs` directory.

       Note: Windows hides file extensions by default. Follow these steps to display them:
       - Press the Alt key to show the Windows Explorer menu bar.
       - Select Tools > Folder options to display the Folder Options window and click the View tab.
       - In the Advanced Settings area, deselect “Hide extensions for known file types”.
       - Click the Apply to Folders button in the Folder views area.
       - In the Folder Views dialog box, click Yes.
       - Click OK to close the Folder Options window.

    b. Double-click the `Lesson2_tee.txt` file. The file opens in Notepad and the contents show all the commands and results from the current session.

       The file shows the commands and results from the first execution of this `tee` file, then starts again from when it was re-executed.
    c. Close the `Lesson2_tee.txt` file.
    d. Close Windows Explorer.

    Note: Do not close the `mysql` client. You use it in the next practice.

Solution 2-2: Using the Keyboard Editing and Tee Commands

There no solutions for this practice. See the practice instructions.

Practice 2-3: Creating the `world_innodb` Database

Overview

In this practice, you create and populate the `world_innodb` database, which you use in practices throughout this course.

Duration

This practice takes approximately 5 minutes to complete.

Tasks

1. Create the `world_innodb` database within the `mysql` client:

   Enter the following at the `mysql` prompt from the previous practice:

   ```
   mysql> CREATE DATABASE world_innodb CHARACTER SET latin1;
   ```

   It returns the following message to indicate that the database was created:

   ```
   Query OK, 1 row affected (0.02 sec)
   ```

   Note: Adding the `latin1` character set to the above database creation statement makes it compatible with the script you use to populate it below.

2. Now you have created the `world_innodb` database, tell the client to use it:

   Execute the following statement at the `mysql>` prompt:

   ```
   mysql> USE world_innodb
   ```

   – **Note:** You do not need a semicolon at the end of a `USE` statement.

   It returns the following message to indicate that the database selection changed:

   ```
   Database changed
   ```

3. Populate the `world_innodb` database with data:

   Execute the following statement at the `mysql>` prompt:

   ```
   mysql> SOURCE D:\labs\world_innodb.sql
   ```

   – **Note:** Do not include a semicolon at the end of the `SOURCE` statement

   MySQL displays multiple instances of the following message while populating `world_innodb`. The process takes several minutes.

   ```
   Query OK, 0 rows affected (0.00 sec)
   ```

   When the script has finished executing, you are returned to the `mysql>` prompt.

   Note: Do not attempt to review the database contents yet. You do this in a later practice.

4. Exit the `mysql` client:

   Enter the following at the `mysql>` prompt:

   ```
   mysql> EXIT
   ```

   The following message appears and control returns to the standard command prompt:

   ```
   Bye
   cmd>
   ```

Solutions 2-3: Creating the `world_innodb` Database

There are no solutions for this practice. See the practice instructions.

Practices for Lesson 3: MySQL Database Basics

Chapter 3

Practices for Lesson 3: Overview

Practices Overview

These practices test your knowledge of database basics. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured MySQL Server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.

Practice 3-1: Quiz – Database Basics

Overview

In this quiz, you answer questions about database basics.

Duration

This practice takes approximately 15 minutes to complete.

Quiz Questions

Choose the best answer from those provided.

1. What does RDBMS stand for?
   a. Relative Database Basic Manipulation System
   b. Relative Data Basic Management System
   c. Relational Database Management System
   d. None of the above
2. An RDBMS organizes and stores data in tables.
   a. True
   b. False
3. An association between database entities is called a _____ and different categories of these exist.
   a. Relationship
   b. Line
   c. System
   d. All of the above
4. An RDBMS is presented as two-dimensional tables that consist of columns and _____.
   a. Schemas
   b. Sets
   c. Values
   d. None of the above
5. A one-to-one relationship exists when you associate a single entity with another single entity.
   a. True
   b. False
6. A row in a database table contains a value for each _____ in the table.
   a. Row
   b. Column
   c. Relationship
7. The SQL acronym stands for Structural Queries Linked.
   a. True
   b. False

8. The set of SQL statements that modify data is called _____ _____ _____.
   a. Data manipulation language
   b. Data definition language
   c. Database modification language
9. `CREATE DATABASE` is a SQL data _____ language statement.
   a. Manipulation
   b. Definition
   c. Modification
   d. None of the above
10. Identify the benefits of using MySQL:
   a. Runs on multiple operating systems
   b. Is highly customizable
   c. Can be open source or with commercial licenses
   d. Supports very large databases
   e. All of the above

Solutions 3-1: Quiz – Database Basics

Quiz Solutions

1. **c**. Relational Database Management System
2. **a**. True
3. **a**. Relationship
4. **d**. None of the above. The correct answer is **rows**.
5. **a**. True
6. **b**. Column.
7. **b**. False. The correct answer is **Structured Query Language**.
8. **a**. Data manipulation language
9. **b**. Definition
10. **e**. All of the above

Practice 3-2: Identifying the Structure of a Table

Overview

In this practice, you identify the relational characteristics of a table.

Duration

This practice takes approximately 10 minutes to complete.

Tasks

The following is a partial list of the contents of the `City` table from the `world_innodb` database. Refer to this for the questions that follow.

```
+----+----------------+-------------+---------------+------------+
| ID | Name           | CountryCode | District      | Population |
+----+----------------+-------------+---------------+------------+
|  1 | Kabul          | AFG         | Kabol         |    1780000 |
|  2 | Qandahar       | AFG         | Qandahar      |     237500 |
|  3 | Herat          | AFG         | Herat         |     186800 |
|  4 | Mazar-e-Sharif | AFG         | Balkh         |     127800 |
|  5 | Amsterdam      | NLD         | Noord-Holland |     731200 |
|  6 | Rotterdam      | NLD         | Zuid-Holland  |     593321 |
|  7 | Den Haag       | NLD         | Zuid-Holland  |     440900 |
|  8 | Utrecht        | NLD         | Utrecht       |     234323 |
...
```

1. How many columns does this table contain?
2. List the names of the columns.
3. How many rows appear in the partial list?
4. Is the city with Amsterdam in the `Name` column in the same `District` as Kabul?
5. What is the `Population` of the city of Utrecht?
6. If you delete the rows (from this list) with a `CountryCode` of AFG, what are the `ID` values of the deleted rows?
7. Can the city of Den Haag belong to more than one `District` in the table structure shown?
8. Can the `District` of Zuid-Holland contain more than one city?

Solutions 3-2: Identifying the Structure of a Table

Tasks

1. **5**
2. **`ID`, `Name`, `CountryCode`, `District`, `Population`**
3. **8**
4. **No**. Amsterdam is in the district of Noord-Holland.
5. **234323**
6. **1, 2, 3,** and **4**
7. **No**. This would result in two values for the `District` column. Only one is possible in the table structure shown.

   You could have two rows with the same `Name` and different values in the `District` column. However, you would need a different way to uniquely identify a city, because `ID` would not work in this instance.
8. **Yes**. Zuid-Holland appears in the Rotterdam and Den Haag rows because both these cities are contained with it.

Practices for Lesson 4: Database Design

Chapter 4

Practices for Lesson 4: Overview

Practices Overview

These practices test your knowledge of database design. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured MySQL Server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.

Note: In this practice the first letters of table names are in uppercase. Windows is not case-sensitive but some operating systems are, so it is good practice to use proper capitalization. The SQL statements are all in uppercase for clarity, but this is not required.

Practice 4-1: Quiz – Database Design

Overview

In this quiz, you answer questions about database design.

Duration

This practice takes approximately 10 minutes to complete.

Quiz Questions

Choose the best answer from those provided for each multiple choice or True/False question.

1. The _____ _____ is a candidate key that best defines exactly one unique row.
   a. Unique key
   b. Foreign key
   c. Relationship key
   d. Primary key
2. The _____ _____ _____ is a high-level, graphical depiction of a data model. It assists in database design.
   a. Entity relationship model
   b. Database modeling process
   c. Entity relationship diagram
3. You use the `SHOW DATABASES` SQL statement to list the available databases.
   a. True
   b. False
4. Which statement shows a list of table column names and settings?
   a. `SHOW TABLE COLUMNS`
   b. `DESCRIBE <table_name>`
   c. `SELECT * FROM <table_name>`
5. The following statement retrieves all the records from the `CountryLanguage` table:

```
mysql> SELECT * FROM CountryLanguage;
```

   a. True
   b. False

6. The following statement displays the `City` table columns and settings. Based on the output, which column is the primary key?

```
mysql> DESCRIBE City;
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| ID          | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name        | char(35) | NO   |     |         |                |
| CountryCode | char(3)  | NO   | MUL |         |                |
| District    | char(20) | NO   |     |         |                |
| Population  | int(11)  | NO   |     | 0       |                |
+-------------+----------+------+-----+---------+----------------+
5 rows in set (0.30 sec)
```

   a. CountryCode
   b. Name
   c. ID
   d. None of the above

7. Using the output from the `DESCRIBE City` statement, which column is a nonunique index that is part of a multiple-column primary key?
   a. Population
   b. Name
   c. ID
   d. None of the above

8. Identify the statement that gives the following output (assume a `USE world_innodb` statement has already been issued).

```
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| city                   |
| country                |
| countrylanguage        |
+------------------------+
3 rows in set (0.19 sec)
```

   a. SHOW TABLES;
   b. SHOW world_innodb;
   c. DESCRIBE TABLES;
   d. SHOW DATABASE TABLES;

9. Poorly designed table structures result in duplicate data, redundant data, and difficulty using the data.
   a. True
   b. False

10. The normalization process ensures database integrity. The first three levels, or _____ _____ are adequate for most databases.
    a. Normalization levels
    b. Normal forms
    c. Normalization forms
    d. None of the above
11. Third normal form is the highest level of normalization that you can achieve within a database.
    a. True
    b. False

Solutions 4-1: Quiz – Database Design

Quiz Solutions

1. **d**. Primary key
2. **c**. Entity relationship diagram
3. **a**. True
4. **b**. `DESCRIBE <table_name>;`
5. **a**. True
6. **c**. `ID`
7. **d**. None of the above. The correct answer is `CountryCode`, which is indicated by `MUL` under the `Key` attribute.
8. **a**. `SHOW TABLES;`
9. **a**. True
10. **b**. Normal forms
11. **b**. False. There are several normal forms above third, although it is rarely necessary for you to go beyond the first three.

Practice 4-2: Evaluating a Database

Overview

In this practice, you view and evaluate the contents of the `world_innodb` database, using the `mysql` client.

> **Note:** The data contained in the `world_innodb` database (with regard to continent, country, and city details) is not accurate. It is used only as a teaching tool.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. Start the `mysql` client.
2. List all the databases available in the current `mysql` session.
3. Change the current database to `world_innodb`.
4. List all the tables in the `world_innodb` database.
5. Show the structure of the `City` table.
6. List the contents of the `City` table.
7. Show the structure of the `Country` table.
8. List the contents of the `Country` table.
9. Show the structure of the `CountryLanguage` table.
10. List the contents of the `CountryLanguage` table.

Note: Keep your `mysql` session open for the next practice.

Solutions 4-2: Evaluating a Database

Tasks

1. Start the `mysql` client:

   Enter the following at the command prompt to log in to the MySQL server and receive the welcome message shown below:

```
cmd> mysql -u root -p
Enter password: oracle

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.6.10-enterprise-commercial-advanced MySQL Enterprise
Server - Advanced Edition (Commercial)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights
reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input
statement.

mysql>
```

2. List all the databases available in the current `mysql` session:

   Compare your statement and results to those shown below:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sakila             |
| test               |
| world              |
| world_innodb       |
+--------------------+
7 rows in set (0.00 sec)
```

   - There are other databases in addition to `world_innodb`. MySQL Installer installed these databases.

3. Change the current database to `world_innodb`:

   Compare your statement and results to those shown below:

```
mysql> USE world_innodb
Database changed
```

4. List all the tables in the `world_innodb` database:

   Compare your statement and results to those shown below:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| city                   |
| country                |
| countrylanguage        |
+------------------------+
3 rows in set (0.13 sec)
```

5. Show the structure of the `City` table:

   Compare your statement and results to those shown below::

```
mysql> DESCRIBE City;
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| ID          | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name        | char(35) | NO   |     |         |                |
| CountryCode | char(3)  | NO   | MUL |         |                |
| District    | char(20) | NO   |     |         |                |
| Population  | int(11)  | NO   |     | 0       |                |
+-------------+----------+------+-----+---------+----------------+
5 rows in set (0.06 sec)
```

   The `DESCRIBE` statement provides information about the columns in the `City` table:

   - `Field`: Column name
   - `Type`: Data type
   - `Null`: Whether the column accepts null values
   - `Key`: Whether the column is part of a key/index
   - `Default`: Default value of the column
   - `Extra`: Any additional settings for the column

6. List the contents of the `City` table:

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM City;
+------+----------------+-------------+--------------+------------+
| ID   | Name           | CountryCode | District     | Population |
+------+----------------+-------------+--------------+------------+
|  1   | Kabul          | AFG         | Kabol        |    1780000 |
|  2   | Qandahar       | AFG         | Qandahar     |     237500 |
|  3   | Herat          | AFG         | Herat        |     186800 |
|  4   | Mazar-e-Sharif | AFG         | Balkh        |     127800 |
|  5   | Amsterdam      | NLD         | Noord-Holland|     731200 |
...
| 4077 | Jabaliya       | PSE         | North Gaza   |     113901 |
| 4078 | Nablus         | PSE         | Nablus       |     100231 |
| 4079 | Rafah          | PSE         | Rafah        |      92020 |
+------+----------------+-------------+--------------+------------+
4079 rows in set (0.01 sec)
```

7. Show the structure of the `Country` table:

   Compare your statement and results to those shown below:

```
mysql> DESCRIBE Country;
+----------------+---------------------+------+-----+---------+-------+
| Field          | Type                | Null | Key | Default | Extra |
+----------------+---------------------+------+-----+---------+-------+
| Code           | char(3)             | NO   | PRI |         |       |
| Name           | char(52)            | NO   |     |         |       |
| Continent      | enum('Asia',        |      |     |         |       |
|                | 'Europe',           |      |     |         |       |
|                | 'North America'     |      |     |         |       |
|                | 'Africa'            |      |     |         |       |
|                | 'Oceania',          |      |     |         |       |
|                | 'Antarctica',       |      |     |         |       |
|                | 'South America')    | NO   |     | Asia    |       |
| Region         | char(26)            | NO   |     |         |       |
| SurfaceArea    | float(10,2)         | NO   |     | 0.00    |       |
| IndepYear      | smallint(6)         | YES  |     | NULL    |       |
| Population     | int(11)             | NO   |     | 0       |       |
| LifeExpectancy | float(3,1)          | YES  |     | NULL    |       |
| GNP            | float(10,2)         | YES  |     | NULL    |       |
| GNPOld         | float(10,2)         | YES  |     | NULL    |       |
| LocalName      | char(45)            | NO   |     |         |       |
| GovernmentForm | char(45)            | NO   |     |         |       |
| HeadOfState    | char(60)            | YES  |     | NULL    |       |
| Capital        | int(11)             | YES  |     | NULL    |       |
| Code2          | char(2)             | NO   |     |         |       |
+----------------+---------------------+------+-----+---------+-------+
15 rows in set (0.06 sec)
```

8. List the contents of the `Country` table:

   The statement shown below uses the `\G` terminator, which displays table contents vertically. This makes the output more readable when there are a large number of columns. The `\G` terminator is case-sensitive and must always be in uppercase.

```
mysql> SELECT * FROM Country\G
*************************** 1. row ***************************
          Code: ABW
          Name: Aruba
     Continent: North America
        Region: Caribbean
   SurfaceArea: 193.00
     IndepYear: NULL
    Population: 103000
LifeExpectancy: 78.4
           GNP: 828.00
        GNPOld: 793.00
     LocalName: Aruba
GovernmentForm: Nonmetropolitan Territory of The Netherlands
   HeadOfState: Beatrix
       Capital: 129
         Code2: AW
...
*************************** 238. row ***************************
          Code: ZMB
          Name: Zambia
     Continent: Africa
        Region: Eastern Africa
   SurfaceArea: 752618.00
```

```
     IndepYear: 1964
    Population: 9169000
LifeExpectancy: 37.2
           GNP: 3377.00
        GNPOld: 3922.00
     LocalName: Zambia
GovernmentForm: Republic
   HeadOfState: Frederick Chiluba
       Capital: 3162
         Code2: ZM
*************************** 239. row ***************************
          Code: ZWE
          Name: Zimbabwe
     Continent: Africa
        Region: Eastern Africa
   SurfaceArea: 390757.00
     IndepYear: 1980
    Population: 11669000
LifeExpectancy: 37.8
           GNP: 5951.00
        GNPOld: 8670.00
     LocalName: Zimbabwe
GovernmentForm: Republic
   HeadOfState: Robert G. Mugabe
       Capital: 4068
         Code2: ZW
239 rows in set (0.00 sec)
```

9. Show the structure of the `CountryLanguage` table:

   Compare your statement and results to those shown below:

```
mysql> DESCRIBE CountryLanguage;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| CountryCode | char(3)       | NO   | PRI |         |       |
| Language    | char(30)      | NO   | PRI |         |       |
| IsOfficial  | enum('T','F') | NO   |     | F       |       |
| Percentage  | float(4,1)    | NO   |     | 0.0     |       |
+-------------+---------------+------+-----+---------+-------+
4 rows in set (0.11 sec)
```

10. List the contents of the `CountryLanguage` table:

    Compare your statement and results to those shown below:

```
mysql> SELECT * FROM CountryLanguage;
+-------------+------------+------------+------------+
| CountryCode | Language   | IsOfficial | Percentage |
+-------------+------------+------------+------------+
| ABW         | Dutch      | T          |        5.3 |
| ABW         | English    | F          |        9.5 |
| ABW         | Papiamento | F          |       76.7 |
...
| ZWE         | English    | T          |        2.2 |
| ZWE         | Ndebele    | F          |       16.2 |
| ZWE         | Nyanja     | F          |        2.2 |
| ZWE         | Shona      | F          |       72.1 |
+-------------+------------+------------+------------+
984 rows in set (0.01 sec)
```

Note: Keep your `mysql` session open for the next practice.

Practice 4-3: Creating a Structure Diagram

Overview

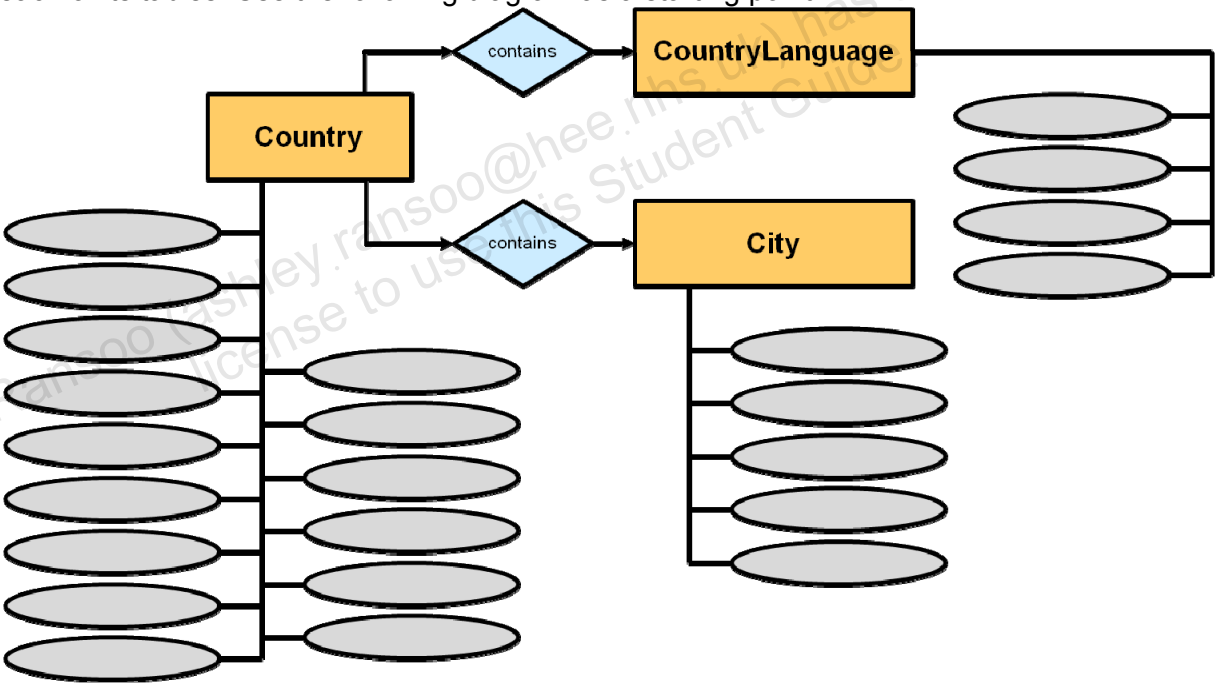
In this practice, you use SQL `DESCRIBE` statements and an entity relationship diagram to evaluate the design of the `world_innodb` database. Use your open `mysql` session to issue any statements you feel will help you better understand `world_innodb`.

Duration

This practice takes approximately 25 minutes to complete.

Tasks

1. Explain the purpose of the `world_innodb` database, using the output from the `DESCRIBE` statements in the previous practice.
2. In the `world_innodb` database, language information for countries is stored in a dedicated table. Based on the structure of the `CountryLanguage` table, what do users need to know about country languages?
3. Create a structure diagram for the `world_innodb` tables by filling in the column names for each of its tables. Use the following diagram as a starting point:



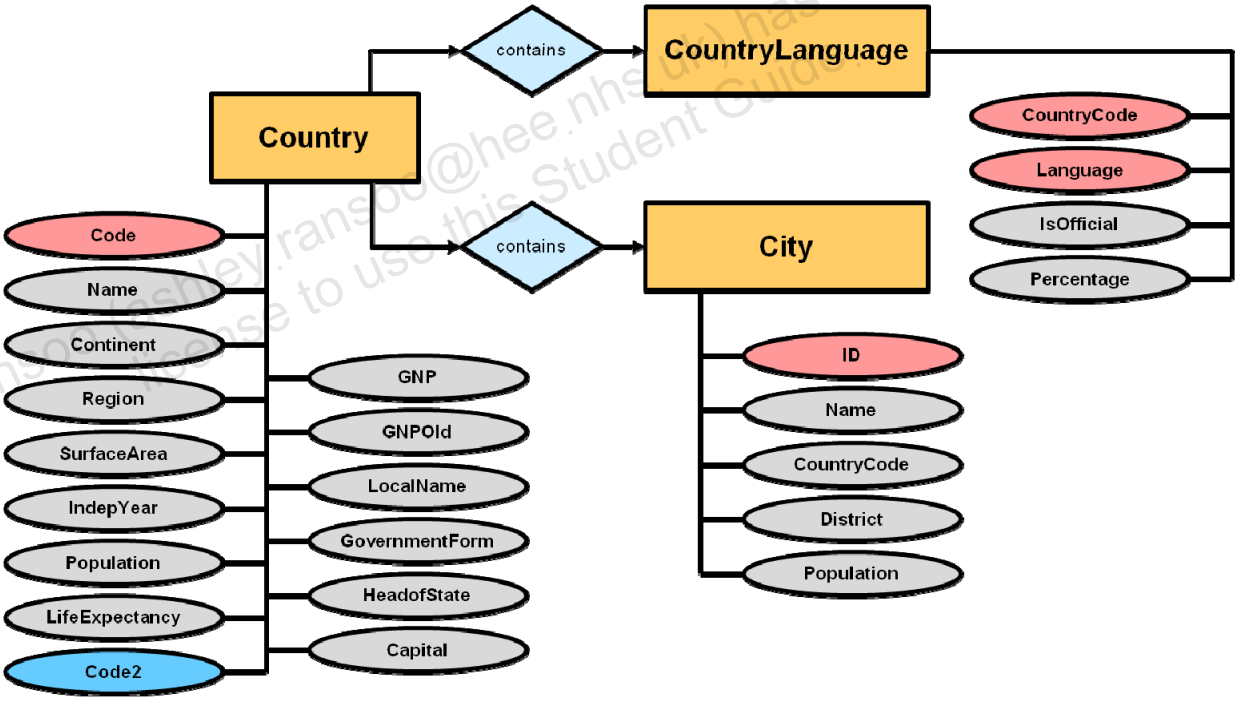
4. Which columns in each table can uniquely identify a record? Indicate these on the table. These are the candidate keys.
5. Cities and countries have their own tables. Continents do not. Why?
6. Why is `CountryLanguage` a separate table? Why not store language information in the `Country` table?
7. Indicate on the diagram which column(s) in each table is/are the primary key.
8. Explain why the `CountryLanguage` table has a composite primary key.
9. Exit the `mysql` client.

Note: The `world_innodb` database is not fully normalized. If you choose to, you can note the improvements necessary to get the database to third normal form.

Solution 4-3: Creating a Structure Diagram

Tasks

1. Explain the purpose of the `world_innodb` database, using the output from the `DESCRIBE` statements in the previous practice. The answer is as follows:
   - The database is being used to store, retrieve and otherwise manipulate information about countries, including continent, language and city data.
2. In the `world_innodb` database, language information for countries is stored in a dedicated table. Based on the structure of the `CountryLanguage` table, what do users need to know about country languages? The answer is as follows:
   - The language spoken
   - The official language
   - The percentage of the population that speaks the language
   - The country the information relates to
3. Create a structure diagram for the `world_innodb` tables by filling in the column names for each of its tables. The completed diagram is as follows:



4. Which columns in each table can uniquely identify a record? Indicate these on the table. These are the candidate keys. The answer is as follows:
   a. `Country`: Code *or* Code2
   b. `City`: ID
   c. `CountryLanguage`: CountryCode *and* Language

5. Cities and countries have their own tables. Continents do not. Why? The answer is as follows:
   - The primary tables are `Country` and `City`. Continents do not need a table of their own because all the information required is at the country and city level.
6. Why is `CountryLanguage` a separate table? Why not store language information in the `Country` table? The answer is as follows:
   - Some countries have more than one national language. If each of these were stored in the `Country` table it would result in redundant rows.
7. Indicate on the diagram which column(s) in each table is/are the primary key. The answer is as follows:
   - `Country`: Code
   - `City`: ID
   - `Country`: CountryCode *and* Language – a composite key.
8. Explain why the `CountryLanguage` table has a composite primary key.
   - The `CountryLanguage` table requires both the CountryCode and Language columns to uniquely identify each row.
9. Exit the `mysql` client:

   Enter the following at the `mysql>` prompt:

```
mysql> EXIT
```

   MySQL displays an exit message and the command window returns to the standard prompt:

```
Bye
cmd>
```

Practices for Lesson 5: Table Data Types

Chapter 5

Practices for Lesson 5: Overview

Practices Overview

These practices test your knowledge of table data types. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured the MySQL server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.

Practice 5-1: Quiz – Data Types

Overview

In this quiz, you answer questions about MySQL data types.

Duration

This practice takes approximately 10 minutes to complete.

Quiz Questions

Choose the best answer from those provided.

1. Data types belong to four different categories: Numeric, Character, Binary, and Temporal.
   a. True
   b. False
2. Numeric data types are suitable only for whole numbers.
   a. True
   b. False
3. The _____ data type represents approximate-value numbers that have an integer part, a fractional part, or both.
   a. `FLOAT`
   b. `INTEGER`
   c. `FIXED`
   d. None of the above
4. These data types are included in the Temporal category:
   a. `TIME`
   b. `DATE`
   c. `DATETIME`
   d. `TIMESTAMP`
   e. All of the above
5. The following statement displays the `CountryLanguage` table columns and settings. The output shows that the data type of the `Language` column is `CHAR`. The _____ data type is also suitable for this column, and allows for variations in the number of characters.

```
mysql> DESC CountryLanguage;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| CountryCode | char(3)       | NO   | PRI |         |       |
| Language    | char(30)      | NO   | PRI |         |       |
| IsOfficial  | enum('T','F') | NO   |     | F       |       |
| Percentage  | float(4,1)    | NO   |     | 0.0     |       |
+-------------+---------------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

   a. `TINYINT`
   b. `VARCHAR`
   c. `BIGCHAR`

6. The data type of the `IsOfficial` column is `ENUM`. You can see this in the output from the `DESCRIBE CountryLanguage` statement in question number 5. `ENUM` is used to restrict the value of column entries to a fixed, enumerated string. The only values allowed are `T` (true) or `F` (false).
   a. True
   b. False
7. _____ _____ data types (like `BLOB`) store values that represent binary data, such as video and audio files.
   a. Character string
   b. Fixed-point
   c. Integer class
   d. Binary string
8. _____ is a SQL keyword used to define data types that permit missing values.
   a. `UNKNOWN`
   b. `DEFAULT`
   c. `NULL`

Solutions 5-1: Quiz – Data Types

Quiz Solutions

1. **a**. True
2. **b**. False. Numeric data types can be whole, fractional or exact-value numbers, as well as `BIT`-field values.
3. **a**. `FLOAT`
4. **e**. All of the above
5. **b**. `VARCHAR`
6. **a**. True
7. **d**. Binary string
8. **c**. `NULL`

Practice 5-2: Explaining the Use of Data Types

Overview

In this practice, you explain the use of data types in a given table example.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

The following is the data structure of the `Country` table (including the column data types), from the `world_innodb` database. Refer to this chart to answer the questions below.

```
+----------------+--------------------+------+-----+---------+-------+
| Field          | Type               | Null | Key | Default | Extra |
+----------------+--------------------+------+-----+---------+-------+
| Code           | char(3)            | NO   | PRI |         |       |
| Name           | char(52)           | NO   |     |         |       |
| Continent      | enum('Asia',       |      |     |         |       |
|                | 'Europe',          |      |     |         |       |
|                | 'North America'    |      |     |         |       |
|                | 'Africa'           |      |     |         |       |
|                | 'Oceania',         |      |     |         |       |
|                | 'Antarctica',      |      |     |         |       |
|                | 'South America')   | NO   |     | Asia    |       |
| Region         | char(26)           | NO   |     |         |       |
| SurfaceArea    | float(10,2)        | NO   |     | 0.00    |       |
| IndepYear      | smallint(6)        | YES  |     | NULL    |       |
| Population     | int(11)            | NO   |     | 0       |       |
| LifeExpectancy | float(3,1)         | YES  |     | NULL    |       |
| GNP            | float(10,2)        | YES  |     | NULL    |       |
| GNPOld         | float(10,2)        | YES  |     | NULL    |       |
| LocalName      | char(45)           | NO   |     |         |       |
| GovernmentForm | char(45)           | NO   |     |         |       |
| HeadOfState    | char(60)           | YES  |     | NULL    |       |
| Capital        | int(11)            | YES  |     | NULL    |       |
| Code2          | char(2)            | NO   |     |         |       |
+----------------+--------------------+------+-----+---------+-------+
```

1. The primary country codes all contain three characters. Does it make sense that the `Code` column is `CHAR(3)` or should it be expanded to allow for more characters?
2. What other data type could the `Name` column use? Why?
3. Could the `Continent` column use a different data type besides `ENUM`? Why?
4. Which continent is used if none is specified in a query?
5. Is `SMALLINT(6)` the best setting for the `IndepYear` column? If not, what is?
6. Why is `FLOAT(10,2)` used for the `GNP` column? What do the 10 and 2 integers mean?
7. Would you ever change the maximum value in the `FLOAT` data type used for the `GNP` column?
8. Why is the `HeadOfState` column set to `CHAR(60)`?
9. Does it make sense that the `Capital` column is set to accept `NULL` values? Why?

Solutions 5-2: Explaining the Use of Data Types

Tasks

1. The primary country codes all contain three characters. Does it make sense that the `Code` column is `CHAR(3)` or should it be expanded to allow for more characters? Why? The answers are as follows:
   - Yes. The number of characters does not change. Therefore, this is the most efficient data type setting.
2. What other data type could the `Name` column use? Why? The answers are as follows:
   - It could also use `VARCHAR(52)`. This uses less storage space for smaller strings. The storage requirement for a `VARCHAR` depends on the character set and the number of characters in the string (including trailing spaces, which a `VARCHAR` retains.)
3. Could the `Continent` column use a different data type besides `ENUM`? Why? The answers are as follows:
   - No. Values are restricted to one selection from a list of strings. `ENUM` is the only data type that suits this purpose.
4. Which continent is used if none is specified in a query? The answer is as follows:
   - `Asia`, which is set as the `DEFAULT` value.
5. Is `SMALLINT(6)` the best setting for the `IndepYear` column? If not, what is? The answers are as follows:
   - No. `SMALLINT(4)` is better because this column needs to display only four characters. Note that the display width does not affect how the actual value is stored, so no data will be lost.
6. Why is `FLOAT(10,2)` used for the `GNP` column? What do the 10 and 2 integers mean? The answers are as follows:
   - It expresses financial data that has an integer and fractional part. `DECIMAL` could be a better choice because it stores exact values (`FLOAT` stores only approximate values).
   - The first integer is the maximum number of decimal digits that can be stored. The second integer is the number of digits after the decimal point.
7. Would you ever change the maximum value in the `FLOAT` data type used for the `GNP` column? The answer is as follows:
   - Yes, if the maximum value for any country's `GNP` were to rise to $100,000,000.00.
8. Why is the `HeadOfState` column set to `CHAR(60)`? The answer is as follows:
   - At least one record in the table could be up to 60 characters long.
9. Does it make sense that the `Capital` column is set to accept `NULL` values? Why?
   - Yes, because not all countries have a capital specified.

Practices for Lesson 6: Database and Table Creation

Chapter 6

Practices for Lesson 6: Overview

Practices Overview

These practices test your knowledge of database and table creation. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured MySQL Server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.

Note: In this practice, the first letters of table names are in uppercase. Windows is not case-sensitive but some operating systems are, so it is good practice to use proper capitalization. The SQL statements are all in uppercase for clarity, but this is not required.

Practice 6-1: Displaying Table Creation Information

Overview

In this practice, you:

- Show the SQL statement syntax required to create a specific table
- Create a new table
- Show indexes for a table

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. Start the `mysql` client and set the current database to `world_innodb`.
2. Execute `SHOW CREATE TABLE` to display the statement used to create the `Country` table.
3. Use the results of the `SHOW CREATE TABLE` statement to create a new table called `Country2` with the same attributes, but do not specify the `ENGINE` or `CHARSET`.
4. Execute the `SHOW TABLES` statement to confirm that the new table now exists.
5. Use the `SHOW INDEXES` statement to display the primary key for the `Country2` table.
6. Identify the indexes in the `City` table.
7. Exit the `mysql` client.

   Note: The changes you make to the `world_innodb` database in these lessons are cumulative. Do not attempt to undo any changes you make to the database in this or future practices.

Solutions 6-1: Displaying Table Creation Information

Tasks

1. Start the `mysql` client and set the current database to `world_innodb`:

   Compare your statement and results to those shown below:

```
cmd> mysql -u root -p
Enter password: oracle
...

mysql> USE world_innodb
Database changed
```

2. Execute `SHOW CREATE TABLE` to display the statement used to create the `Country` table.

   Compare your statement and results to those shown below:

```
mysql> SHOW CREATE TABLE Country\G
*************************** 1. row ***************************
       Table: Country
Create Table: CREATE TABLE `country` (
  `Code` char(3) NOT NULL DEFAULT '',
  `Name` char(52) NOT NULL DEFAULT '',
  `Continent` enum('Asia','Europe','North America','Africa',
    'Oceania','Antarctica','South America')
    NOT NULL DEFAULT 'Asia',
  `Region` char(26) NOT NULL DEFAULT '',
  `SurfaceArea` float(10,2) NOT NULL DEFAULT '0.00',
  `IndepYear` smallint(6) DEFAULT NULL,
  `Population` int(11) NOT NULL DEFAULT '0',
  `LifeExpectancy` float(3,1) DEFAULT NULL,
  `GNP` float(10,2) DEFAULT NULL,
  `GNPOld` float(10,2) DEFAULT NULL,
  `LocalName` char(45) NOT NULL DEFAULT '',
  `GovernmentForm` char(45) NOT NULL DEFAULT '',
  `HeadOfState` char(60) DEFAULT NULL,
  `Capital` int(11) DEFAULT NULL,
  `Code2` char(2) NOT NULL DEFAULT '',
  PRIMARY KEY (`Code`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

3. Use the results of the `SHOW CREATE TABLE` statement to create a new table called `Country2` with the same attributes, but do not specify the `ENGINE` or `CHARSET`.

   Compare your statement and results to those shown below:

```
mysql> CREATE TABLE `Country2` (
    -> `Code` char(3) NOT NULL DEFAULT '',
    -> `Name` char(52) NOT NULL DEFAULT '',
    -> `Continent` enum('Asia','Europe','North
America','Africa', 'Oceania','Antarctica','South America')
NOT NULL DEFAULT 'Asia',
    -> `Region` char(26) NOT NULL DEFAULT '',
    -> `SurfaceArea` float(10,2) NOT NULL DEFAULT '0.00',
    -> `IndepYear` smallint(6) DEFAULT NULL,
```

```
    -> `Population` int(11) NOT NULL DEFAULT '0',
    -> `LifeExpectancy` float(3,1) DEFAULT NULL,
    -> `GNP` float(10,2) DEFAULT NULL,
    -> `GNPOld` float(10,2) DEFAULT NULL,
    -> `LocalName` char(45) NOT NULL DEFAULT '',
    -> `GovernmentForm` char(45) NOT NULL DEFAULT '',
    -> `HeadOfState` char(60) DEFAULT NULL,
    -> `Capital` int(11) DEFAULT NULL,
    -> `Code2` char(2) NOT NULL DEFAULT '',
    -> PRIMARY KEY (`Code`)
    -> );
Query OK, 0 rows affected (0.14 sec)
```

- The quotes used around table and column names (``) are known as backticks. They are not necessary, but can aid clarity and allow you to include special characters and reserved words in the names.
- Consider using MySQL Workbench to enter this long and complex statement.

4. Execute the `SHOW TABLES` statement to confirm that the new table now exists:

   Compare your statement and results to those shown below:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| city                   |
| country                |
| country2               |
| countrylanguage        |
+------------------------+
4 rows in set (0.16 sec)
```

5. Use the `SHOW INDEXES` statement to determine the primary key for the `Country2` table:

   Compare your statement and results to those shown below:

```
mysql> SHOW INDEX FROM Country2\G
*************************** 1. row ***************************
        Table: country2
   Non_unique: 0
     Key_name: PRIMARY
 Seq_in_index: 1
  Column_name: Code
    Collation: A
  Cardinality: 0
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
1 row in set (0.00 sec)
```

- The result shows only one index. It is the primary key on the `Code` column.

6. Identify the indexes in the `City` table.

   Compare your statement and results to those shown below:

```
mysql> SHOW INDEX FROM City\G
***************** 1. row ********************
        Table: city
   Non_unique: 0
     Key_name: PRIMARY
 Seq_in_index: 1
  Column_name: ID
    Collation: A
  Cardinality: 4051
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
***************** 2. row ********************
        Table: city
   Non_unique: 1
     Key_name: CountryCode
 Seq_in_index: 1
  Column_name: CountryCode
    Collation: A
  Cardinality: 368
     Sub_part: NULL
       Packed: NULL
         Null:
   Index_type: BTREE
      Comment:
Index_comment:
2 rows in set (0.00 sec)
```

   – The result shows two indexes: a primary key on the `ID` column and a second index on the `CountryCode` column.

   – Note that the cardinality values can vary from those shown.

7. Exit the `mysql` client:

   Enter the following at the `mysql>` prompt:

```
mysql> EXIT
```

   MySQL displays an exit message and the command window returns to the standard prompt:

```
Bye
cmd>
```

Practice 6-2: Creating a Database

Overview

In this practice, you create a new database and its tables. This database is for a veterinary clinic and consists of information about pets and their owners. You will start with a spreadsheet and go through some initial design steps before creating the database and populating its tables. Although this is a small database, keep in mind that it must leave room for growth.

Important Note: Save your work and ensure that the database remains in a consistent state so that it you can build on it in future practices.

Duration

This practice takes approximately 60 minutes to complete.

The following is a spreadsheet containing details of pets and their owners. You will use this information to create the database:

Pet Name	Owner	Phone	Type	Category	Gender	Birth date	Death date
Fluffy	Harold	15554159855	Cat	Mammal	F	2003-02-04	
Claws	Gwen	15551234567	Cat	Mammal	M	2004-03-17	
Buffy	Harold	15554159855	Dog	Mammal	F	1999-05-13	
Fang	Benny	15553456789	Dog	Mammal	M	2000-08-27	
Bowser	Diane	15554567890	Dog	Mammal	M	1989-08-31	2009-07-29
Chirpy	Gwen	15551234567	Parrot	Bird	F	2008-09-11	
Whistler	Gwen	15551234567	Canary	Bird		2007-12-09	
Slim	Benny	15553456789	Snake	Reptile	M	2006-04-29	
Puffball	Diane	15554567890	Hamster	Mammal	F	2009-03-30	
Opus	Caryn	15554444444	Ferret	Mammal	M		
Rocky	Chris	15556666666	Dog	Mammal	M	2008-04-04	2013-02-11
Koko	Benny	15553456789	Dog	Mammal	M	1997-02-08	
Scruffy	Gwen	15551234567	Cat	Mammal	M	2008-04-17	

Note: Not all the information is available for every pet. Therefore, some of the columns must allow null values.

Tasks

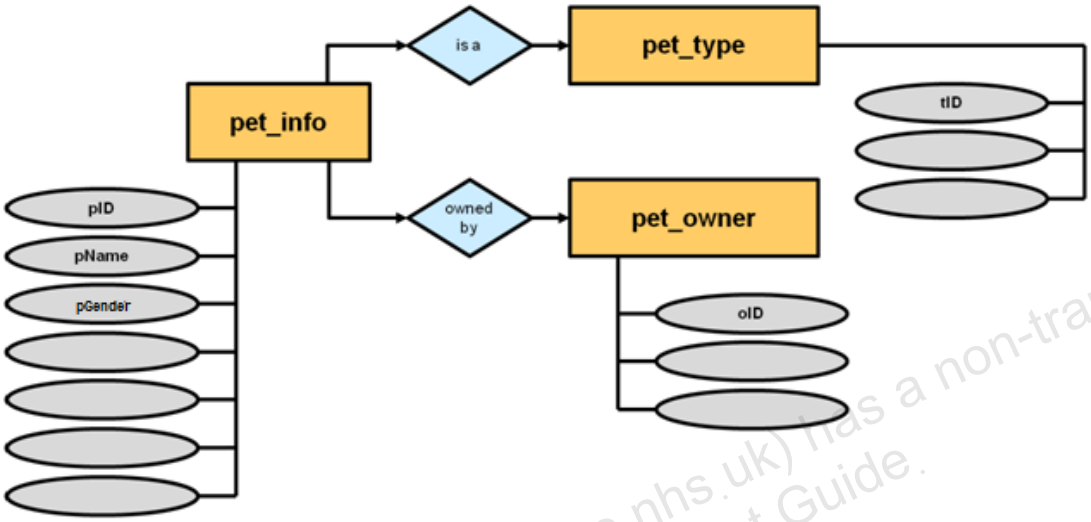
1. Start the database design process by answering the following questions:
   a. What is the primary purpose of the database?
   b. Considering its purpose, what is a good name for this database?
   c. Does any owner have more than one pet?
   d. Does any pet have more than one owner?
   e. Can more than one pet have the same name?
   f. Can you assign the same pet type to more than one pet?
   g. Can a pet have more than one pet type?
   h. Can you assign the same pet type to more than one category?

i. Can a category have more than one pet type?

j. Can a pet have more than one gender?

k. Would any table(s) from this database benefit from an extra column to uniquely identify each record?

2. Draw a structure diagram (like you did for the `world_innodb` database in an earlier practice) to show the tables and columns required. Use the diagram below as a starting point. The structure is partially normalized for you.



Note: Each table has a unique identifier so that tables can reference each other by using foreign keys, where applicable.

3. The normalization process resulted in the following tables. Review these tables. You will use them to create the `Pets` database:

   a. `pet_info` table:

pID*	pName	pGender	pBday	pDday	oID**	tID***
1	Fluffy	F	2003-02-04	NULL	1	1
2	Claws	M	2004-03-17	NULL	2	1
3	Buffy	F	1999-05-13	NULL	1	2
4	Fang	M	2000-08-27	NULL	3	2
5	Bowser	M	1989-08-31	2009-07-29	4	2
6	Chirpy	F	2008-09-11	NULL	2	3
7	Whistler	NULL	2007-12-09	NULL	2	4
8	Slim	M	2006-04-29	NULL	3	5
9	Puffball	F	2009-03-30	NULL	4	1
10	Opus	M	NULL	NULL	5	1
11	Rocky	M	1998-04-04	2013-02-11	6	1
12	Koko	M	1997-02-08	NULL	3	1
13	Scruffy	M	2008-04-17	NULL	2	1

* = Primary key, ** = Foreign key, references the `owners` table (`oID`), ***= Foreign key, references the `pet_types` table (`tID`)

b. `owners` table:

oID*	oName	oPhone
1	Harold	15554159855
2	Gwen	15551234567
3	Benny	15553456789
4	Diane	15554567890
5	Caryn	15554444444
6	Chris	15556666666

* = Primary key

c. `pet_types` table:

tID*	pType	pCategory
1	Cat	Mammal
2	Dog	Mammal
3	Parrot	Bird
4	Canary	Bird
5	Snake	Reptile
6	Hamster	Mammal
7	Ferret	Mammal

* = Primary key

4. Decide on the column data types and other desired attributes by answering the following questions for each column:
   a. Does each row need to be unique, or are duplicates allowed?
   b. Which (if any) of the column options must you use? For example: `NOT NULL`.
   c. Which category of data type is relevant for the column? For example: Numeric or character string?
   d. Which is the most appropriate data type within that category? For example: `INT` or `CHAR(20)`.

   Note: Remember that although you have sample data already, your design needs to accommodate more being added later on. For example, ensure that the data type for the pet name is large enough to support a relatively long name, not just the longest name in the current data set.

`pet_info` table (use the first column as an example):

Attributes	pID*	pName	pGender	pBday	pDday	oID**	tID***
Unique?	Yes						
Options?	`NOT NULL, AUTO_INCREMENT`						
Category?	Integer						
Data type?	`INT`						

Note: You do not need to provide a specific value for the `INT` data type (unlike, for example, `CHAR(30)`).

`owners` table:

Attributes	oID*	oName	oPhone
Unique?			
Options?			
Category?			
Data type?			

`pet_types` table:

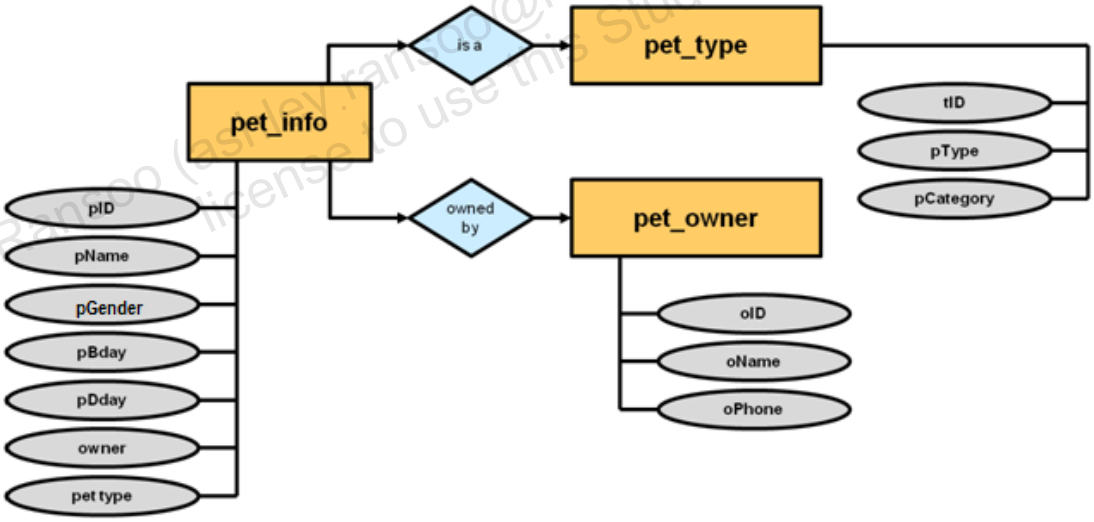
Attributes	tID*	pType	pCategory
Unique?			
Options?			
Category?			
Data type?			

5. Create the `Pets` database in a `mysql` client session.
6. Confirm that the `Pets` database is in the list of available databases.
7. Change the current database to `Pets`.
8. Use your plan to create the empty tables, including their primary keys.
   Note: Do not create the foreign keys yet. You do this in a later lesson.
9. Confirm that the tables are available.
10. View the table structure for each table. Use the `DESCRIBE <table name>` statement for this.
11. Exit the `mysql` client.
    Note: You now have a database with three empty tables. You populate them with data in a later practice.

Solutions 6-2: Creating a Database

Tasks

1. Start the database design process by answering the following questions:
   a. What is the primary purpose of this database? **This database is for a veterinary clinic and consists of information about pets and their owners.**
   b. Considering its purpose, what is a good name for this database? **"Pets"**
   c. Does any owner have more than one pet? **Yes**
   d. Does any pet have more than one owner? **No. Not for the purpose of this practice**.
   e. Can more than one pet have the same name? **Yes**
   f. Can you assign the same pet type to more than one pet? **Yes**
   g. Can a pet have more than one pet type? **No**
   h. Can you assign the same pet type to more than one category? **No**
   i. Can a category have more than one pet type? **Yes**
   j. Can a pet have more than one gender? **No**
   k. Would any table(s) from this database benefit from an extra column to uniquely identify each record? **Yes, all of them could benefit from an identifier to ensure that each row is unique.**
2. Draw a structure diagram (like you did for the `world_innodb` database in an earlier practice) to show the tables and columns required. Use the diagram below as a starting point, which has been partially normalized for you.



3. Review the final table design shown in task 3.

4. Decide on the column data types and other desired attributes by answering the following questions for each column:

`pet_info` table (use the first column as an example):

Attributes	**pID***	**pName**	**pGender**	**pBday**	**pDday**	**oID****	**tID*****
Unique?	Yes	No	No	No	No	No	No
Options?	`NOT NULL, AUTO_INCREMENT`	`NOT NULL`	`DEFAULT NULL`	`DEFAULT NULL`	`DEFAULT NULL`	`NOT NULL`	`NOT NULL`
Category?	Integer	String	String	Temporal	Temporal	Integer	Integer
Data type?	`INT`	`VARCHAR (20)`	`ENUM ('M','F')`	`DATE`	`DATE`	`INT`	`INT`

`owners` table:

Attributes	**oID***	**oName**	**oPhone**
Unique?	Yes	No	No
Options?	`NOT NULL, AUTO_INCREMENT`	`NOT NULL`	`NOT NULL`
Category?	Integer	String	String/Number
Data type?	`INT`	`VARCHAR(20)`	`CHAR(11)`

`pet_types` table:

Attributes	**tID***	**pType**	**pCategory**
Unique?	Yes	Yes	No
Options?	`NOT NULL, AUTO_INCREMENT`	`NOT NULL`	`NOT NULL`
Category?	Integer	String	String
Data type?	`INT`	`VARCHAR(20)`	`VARCHAR(20)`

5. Create the `Pets` database in a `mysql` client session:
   a. Log in to the `mysql` client program:

```
cmd> mysql -u root -p
Enter password: oracle
...
```

   b. Compare your statement and results to those shown below:

```
mysql> CREATE DATABASE Pets;
Query OK, 1 row affected (0.05 sec)
```

6. Confirm that the `Pets` database is in the list of available databases:

   Compare your statement and results to those shown below:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
```

```
| performance_schema |
| pets               |
| sakila             |
| test               |
| world              |
| world_innodb       |
+--------------------+
8 rows in set (0.00 sec)
```

7. Change the current database to `Pets`:

   Compare your statement and results to those shown below:

```
mysql> USE Pets
Database changed
```

8. Use your plan to create the empty tables, including their primary keys.

   Compare your statement and results to those shown below:

```
mysql> CREATE TABLE pet_info (
    -> pID INT NOT NULL AUTO_INCREMENT,
    -> pName VARCHAR(20) NOT NULL,
    -> pGender ENUM('M', 'F') DEFAULT NULL,
    -> pBday DATE DEFAULT NULL,
    -> pDday DATE DEFAULT NULL,
    -> oID INT NOT NULL,
    -> tID INT NOT NULL,
    -> PRIMARY KEY (pID)
    -> );
Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TABLE owners (
    -> oID INT NOT NULL AUTO_INCREMENT,
    -> oName VARCHAR(20) NOT NULL,
    -> oPhone CHAR(11) NOT NULL,
    -> PRIMARY KEY (oID)
    -> );
Query OK, 0 rows affected (0.09 sec)

mysql> CREATE TABLE pet_types (
    -> tID INT NOT NULL AUTO_INCREMENT,
    -> pType VARCHAR(20) NOT NULL,
    -> pCategory VARCHAR(20) NOT NULL,
    -> PRIMARY KEY (tID)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

Note: Do not create the foreign keys yet. You do this in a later lesson.

9. Confirm that the tables are available:

   Compare your statement and results to those shown below:

```
mysql> SHOW TABLES;
+----------------+
| Tables_in_pets |
+----------------+
| owners         |
| pet_info       |
| pet_types      |
+----------------+
3 rows in set (0.00 sec)
```

10. View the table structure for each table. Use the `DESCRIBE <table name>` statement for this.

    Compare your statement and results to those shown below:

```
mysql> DESC pet_info;
+---------+---------------+------+-----+---------+----------------+
| Field   | Type          | Null | Key | Default | Extra          |
+---------+---------------+------+-----+---------+----------------+
| pID     | int(11)       | NO   | PRI | NULL    | auto_increment |
| pName   | varchar(20)   | NO   |     | NULL    |                |
| pGender | enum('M','F') | YES  |     | NULL    |                |
| pBday   | date          | YES  |     | NULL    |                |
| pDday   | date          | YES  |     | NULL    |                |
| oID     | int(11)       | NO   |     | NULL    |                |
| tID     | int(11)       | NO   |     | NULL    |                |
+---------+---------------+------+-----+---------+----------------+
7 rows in set (0.09 sec)

mysql> DESC owners;
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| oID    | int(11)     | NO   | PRI | NULL    | auto_increment |
| oName  | varchar(20) | NO   |     | NULL    |                |
| oPhone | char(11)    | NO   |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
3 rows in set (0.08 sec)

mysql> DESC pet_types;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| tID       | int(11)     | NO   | PRI | NULL    | auto_increment |
| pType     | varchar(20) | NO   |     | NULL    |                |
| pCategory | varchar(20) | NO   |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
3 rows in set (0.09 sec)
```

    - `DESC` is a shortened version of the `DESCRIBE` statement.
    - Note that the `INT` data type defaults to a display width value of 11. This does not affect the size of the value that can be stored.

Note: You now have a database with three empty tables. You populate them with data in a later practice.

11. Exit the `mysql` client:

    Enter the following at the `mysql>` prompt:

```
mysql> EXIT
```

    The following message appears and Control returns to the standard command prompt:

```
Bye
cmd>
```

Practices for Lesson 7: Basic Queries

Chapter 7

Practices for Lesson 7: Overview

Practices Overview

These practices test your knowledge of basic queries. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed the MySQL server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.
- You have created the `Pets` database and defined its tables.
- You can access MySQL Workbench if you choose to complete the practices using this tool.

Note: In this practice, the first letters of table names are in uppercase. Windows is not case-sensitive but some operating systems are, so it is good practice to use proper capitalization. The SQL statements are all in uppercase for clarity, but this is not required.

Practice 7-1: Performing Basic Queries

Overview

In this practice, you query the `world_innodb` database using the `mysql` client.

Duration

This practice takes approximately 20 minutes to complete.

Tasks

1. Start the `mysql` client and set the current database to `world_innodb`.
2. Use a `DESCRIBE` statement to see which columns are available for querying in the `Country` table.
3. Execute a `SELECT` statement that retrieves the `Continent` column data from the `Country` table.
4. Change the preceding `SELECT` statement to include the `Name` column from the `Country` table.
5. Execute a `SELECT` statement that retrieves all the `Region` column data from the `Country` table.
6. Change the preceding `SELECT` statement to retrieve only the distinct `Region` column data from the `Country` table.
7. Execute a `SELECT` statement that retrieves all columns from the `City` table where the identification number is 3875.

   Hint: Use the `*` symbol to indicate that you want all column data.
8. Execute a `SELECT` statement that retrieves names and population figures from the `Country` table where the population is less than 1000.
9. Execute a `SELECT` statement that retrieves the names of all cities from the `City` table in descending `Name` order.
10. Use a `DESCRIBE` statement to see which columns are available for querying in the `CountryLanguage` table.
11. Execute a `SELECT` statement that retrieves the country code and language from the `CountryLanguage` table where the language is Swedish, in descending order of `CountryCode`.
12. Execute a `SELECT` statement that retrieves the name of the cities from the `City` table in ascending alphabetical order, and limit the number of rows to 10.
13. Execute a `SELECT` statement that retrieves the country code and language from the `CountryLanguage` table where the language is Chinese, in descending order of country code. Limit the result to two rows.
14. Execute a `SELECT` statement that retrieves all columns for countries where the GNP is greater than the old GNP, in order of country name. Limit the result to three rows.

    Hint: Use the `\G` terminator to get a more readable result.
15. Exit the `mysql` client.

Solutions 7-1: Performing Basic Queries

Tasks

1. Start the `mysql` client and set the current database to `world_innodb`:

   Enter the following at the command prompt, and receive the results shown below:

```
cmd> mysql -u root -p
Enter password: oracle
...

mysql> USE world_innodb
Database changed
```

2. Use a `DESCRIBE` statement to see which columns are available for querying in the `Country` table.

   Compare your statement and results to those shown below:

```
mysql> DESC Country;
+----------------+-------------------------+------+-----+---------+-------+
| Field          | Type                    | Null | Key | Default | Extra |
+----------------+-------------------------+------+-----+---------+-------+
| Code           | char(3)                 | NO   | PRI |         |       |
| Name           | char(52)                | NO   |     |         |       |
| Continent      | enum('Asia',            |      |     |         |       |
                       'Europe',
                       'North America',
                       'Africa',
                       'Oceania',
                       'Antarctica',
                       'South America')    | NO   |     | Asia    |       |
| Region         | char(26)                | NO   |     |         |       |
| SurfaceArea    | float(10,2)             | NO   |     | 0.00    |       |
| IndepYear      | smallint(6)             | YES  |     | NULL    |       |
| Population     | int(11)                 | NO   |     | 0       |       |
| LifeExpectancy | float(3,1)              | YES  |     | NULL    |       |
| GNP            | float(10,2)             | YES  |     | NULL    |       |
| GNPOld         | float(10,2)             | YES  |     | NULL    |       |
| LocalName      | char(45)                | NO   |     |         |       |
| GovernmentForm | char(45)                | NO   |     |         |       |
| HeadOfState    | char(60)                | YES  |     | NULL    |       |
| Capital        | int(11)                 | YES  |     | NULL    |       |
| Code2          | char(2)                 | NO   |     |         |       |
+----------------+-------------------------+------+-----+---------+-------+
15 rows in set (0.02 sec)
```

3. Execute a `SELECT` statement that retrieves the `Continent` column data from the `Country` table:

   Compare your statement and results to those shown below:

```
mysql> SELECT Continent FROM Country;
+---------------+
| Continent     |
+---------------+
| North America |
| Asia          |
| Africa        |
| North America |
| Europe        |
| Europe        |
```

```
| North America |
| Asia          |
| South America |
| Asia          |
| Oceania       |
| Antarctica    |
| Antarctica    |
| North America |
| Oceania       |
| Europe        |
| Asia          |
...
| South America |
| North America |
| North America |
| Asia          |
| Oceania       |
| Oceania       |
| Oceania       |
| Asia          |
| Europe        |
| Africa        |
| Africa        |
| Africa        |
+---------------+
239 rows in set (0.45 sec)
```

4. Change the preceding `SELECT` statement to include the `Name` column from the `Country` table.

   Compare your statement and results to those shown below:

```
mysql> SELECT Continent, Name FROM Country;
+---------------+--------------------------------------------+
| Continent     | Name                                       |
+---------------+--------------------------------------------+
| North America | Aruba                                      |
| Asia          | Afghanistan                                |
| Africa        | Angola                                     |
| North America | Anguilla                                   |
| Europe        | Albania                                    |
| Europe        | Andorra                                    |
| North America | Netherlands Antilles                       |
| Asia          | United Arab Emirates                       |
| South America | Argentina                                  |
| Asia          | Armenia                                    |
| Oceania       | American Samoa                             |
| Antarctica    | Antarctica                                 |
| Antarctica    | French Southern territories                |
| North America | Antigua and Barbuda                        |
| Oceania       | Australia                                  |
| Europe        | Austria                                    |
| Asia          | Azerbaijan                                 |
...
| South America | Venezuela                                  |
| North America | Virgin Islands, British                    |
| North America | Virgin Islands, U.S.                       |
| Asia          | Vietnam                                    |
| Oceania       | Vanuatu                                    |
```

```
| Oceania        | Wallis and Futuna                                  |
| Oceania        | Samoa                                              |
| Asia           | Yemen                                              |
| Europe         | Yugoslavia                                         |
| Africa         | South Africa                                       |
| Africa         | Zambia                                             |
| Africa         | Zimbabwe                                           |
+----------------+----------------------------------------------------+
239 rows in set (0.01 sec)
```

5. Execute a `SELECT` statement that retrieves all the `Region` column data from the `Country` table.

   Compare your statement and results to those shown below:

```
mysql> SELECT Region FROM Country;
+--------------------------+
| Region                   |
+--------------------------+
| Caribbean                |
| Southern and Central Asia |
| Central Africa           |
| Caribbean                |
| Southern Europe          |
| Southern Europe          |
| Caribbean                |
| Middle East              |
| South America            |
| Middle East              |
| Polynesia                |
| Antarctica               |
| Antarctica               |
...
| South America            |
| Caribbean                |
| Caribbean                |
| Southeast Asia           |
| Melanesia                |
| Polynesia                |
| Polynesia                |
| Middle East              |
| Southern Europe          |
| Southern Africa          |
| Eastern Africa           |
| Eastern Africa           |
+--------------------------+
239 rows in set (0.00 sec)
```

6. Change the preceding `SELECT` statement to retrieve only the distinct `Region` column data from the `Country` table.

   Compare your statement and results to those shown below:

```
mysql> SELECT DISTINCT Region FROM Country;
+--------------------------+
| Region                   |
+--------------------------+
| Caribbean                |
| Southern and Central Asia |
| Central Africa           |
```

```
| Southern Europe           |
| Middle East               |
| South America             |
| Polynesia                 |
| Antarctica                |
| Australia and New Zealand |
| Western Europe            |
| Eastern Africa            |
| Western Africa            |
| Eastern Europe            |
| Central America           |
| North America             |
| Southeast Asia            |
| Southern Africa           |
| Eastern Asia              |
| Nordic Countries          |
| Northern Africa           |
| Baltic Countries          |
| Melanesia                 |
| Micronesia                |
| British Islands           |
| Micronesia/Caribbean      |
+---------------------------+
25 rows in set (0.16 sec)
```

7. Execute a `SELECT` statement that retrieves all columns from the `City` table where the identification number is 3875.

   Compare your statement and results to those shown below:

```
mysql> SELECT *
    -> FROM City
    -> WHERE ID = 3875;
+------+---------+-------------+-----------+------------+
| ID   | Name    | CountryCode | District  | Population |
+------+---------+-------------+-----------+------------+
| 3875 | Madison | USA         | Wisconsin |     208054 |
+------+---------+-------------+-----------+------------+
1 row in set (0.09 sec)
```

8. Execute a `SELECT` statement that retrieves names and population figures from the `Country` table where the population is less than 1000.

   Compare your statement and results to those shown below:

```
mysql> SELECT Name, Population
    -> FROM Country
    -> WHERE Population < 1000;
+--------------------------------------------+------------+
| Name                                       | Population |
+--------------------------------------------+------------+
| Antarctica                                 |          0 |
| French Southern territories                |          0 |
| Bouvet Island                              |          0 |
| Cocos (Keeling) Islands                    |        600 |
| Heard Island and McDonald Islands          |          0 |
| British Indian Ocean Territory             |          0 |
| Pitcairn                                   |         50 |
| South Georgia and the South Sandwich Islands |          0 |
```

```
| United States Minor Outlying Islands          |           0 |
+-----------------------------------------------+-------------+
9 rows in set (0.00 sec)
```

9. Execute a `SELECT` statement that retrieves the names of all cities from the `City` table in descending `Name` order.

   Compare your statement and results to those shown below:

```
mysql> SELECT Name
    -> FROM City
    -> ORDER BY Name DESC;
+-----------------------------------+
| Name                              |
+-----------------------------------+
| ´s-Hertogenbosch                  |
| Šumen                             |
| Štšolkovo                         |
| Šostka                            |
| Šiauliai                          |
| Šahty                             |
| Öskemen                           |
| Örebro                            |
| [San Cristóbal de] la Laguna      |
| Århus                             |
| Zytomyr                           |
| Zürich                            |
| Zwolle                            |
| Zwickau                           |
...
| Abilene                           |
| Abiko                             |
| Abidjan                           |
| Abha                              |
| Aberdeen                          |
| Abeokuta                          |
| Abbotsford                        |
| Abakan                            |
| Abaetetuba                        |
| Abadan                            |
| Aba                               |
| Aalborg                           |
| Aachen                            |
| A Coruña (La Coruña)              |
+-----------------------------------+
4079 rows in set (0.03 sec)
```

10. Use a `DESCRIBE` statement to see which columns are available for querying in the `Country` table.

    Compare your statement and results to those shown below:

```
mysql> DESC CountryLanguage;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| CountryCode | char(3)       | NO   | PRI |         |       |
| Language    | char(30)      | NO   | PRI |         |       |
| IsOfficial  | enum('T','F') | NO   |     | F       |       |
| Percentage  | float(4,1)    | NO   |     | 0.0     |       |
```

```
+-------------+---------------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

11. Execute a `SELECT` statement that retrieves the country code and language from the `CountryLanguage` table where the language is Swedish, in descending order of `CountryCode`.

    Compare your statement and results to those shown below:

```
mysql> SELECT CountryCode, Language
    -> FROM CountryLanguage
    -> WHERE Language = 'Swedish'
    -> ORDER BY CountryCode DESC;
+-------------+----------+
| CountryCode | Language |
+-------------+----------+
| SWE         | Swedish  |
| NOR         | Swedish  |
| FIN         | Swedish  |
| DNK         | Swedish  |
+-------------+----------+
4 rows in set (0.05 sec)
```

12. Execute a `SELECT` statement that retrieves the name of the cities from the `City` table in ascending alphabetical order, and limit the number of rows to 10.

    Compare your statement and results to those shown below:

```
mysql> SELECT Name
    -> FROM City
    -> ORDER BY Name ASC
    -> LIMIT 10;
+----------------------+
| Name                 |
+----------------------+
| A Coruña (La Coruña) |
| Aachen               |
| Aalborg              |
| Aba                  |
| Abadan               |
| Abaetetuba           |
| Abakan               |
| Abbotsford           |
| Abeokuta             |
| Aberdeen             |
+----------------------+
10 rows in set (0.02 sec)
```

13. Execute a `SELECT` statement that retrieves the country code and language from the `CountryLanguage` table where the language is Chinese, in descending order of country code. Limit the result to two rows.

    Compare your statement and results to those shown below:

```
mysql> SELECT CountryCode, Language
    -> FROM CountryLanguage
    -> WHERE Language = 'Chinese'
    -> ORDER BY CountryCode DESC
```

```
    -> LIMIT 2;
+-------------+----------+
| CountryCode | Language |
+-------------+----------+
| VNM         | Chinese  |
| USA         | Chinese  |
+-------------+----------+
2 rows in set (0.00 sec)
```

14. Execute a `SELECT` statement that retrieves all columns for countries where the GNP is greater than the old GNP, in order of country name. Limit the result to three rows.

    Compare your statement and results to those shown below:

```
mysql> SELECT *
    -> FROM Country
    -> WHERE GNP > GNPOld
    -> ORDER BY Name
    -> LIMIT 3\G
*************************** 1. row ***************************
          Code: ALB
          Name: Albania
     Continent: Europe
        Region: Southern Europe
   SurfaceArea: 28748.00
     IndepYear: 1912
    Population: 3401200
LifeExpectancy: 71.6
           GNP: 3205.00
        GNPOld: 2500.00
     LocalName: Shqipëria
GovernmentForm: Republic
   HeadOfState: Rexhep Mejdani
       Capital: 34
         Code2: AL
*************************** 2. row ***************************
          Code: DZA
          Name: Algeria
     Continent: Africa
        Region: Northern Africa
   SurfaceArea: 2381741.00
     IndepYear: 1962
    Population: 31471000
LifeExpectancy: 69.7
           GNP: 49982.00
        GNPOld: 46966.00
     LocalName: Al-Jaza'ir/Algérie
GovernmentForm: Republic
   HeadOfState: Abdelaziz Bouteflika
       Capital: 35
         Code2: DZ
*************************** 3. row ***************************
          Code: ATG
          Name: Antigua and Barbuda
     Continent: North America
        Region: Caribbean
   SurfaceArea: 442.00
```

```
     IndepYear: 1981
    Population: 68000
LifeExpectancy: 70.5
           GNP: 612.00
        GNPOld: 584.00
     LocalName: Antigua and Barbuda
GovernmentForm: Constitutional Monarchy
   HeadOfState: Elisabeth II
       Capital: 63
         Code2: AG
3 rows in set (0.00 sec)
```

15. Exit the `mysql` client:

    Enter the following at the `mysql>` prompt:

```
mysql> EXIT
```

    The following message appears and Control returns to the standard command prompt:

```
Bye
cmd>
```

Practice 7-2: Perform Basic Queries Using MySQL Workbench

Overview

In this practice, you use the MySQL Workbench GUI to perform basic `SELECT` statements. You run the SQL Development module, and set the options to connect to the MySQL server.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. Open MySQL Workbench by selecting it from the MySQL programs:
   a. Select the Windows Start menu.
   b. Select the All Programs menu.
   c. Select MySQL.
   d. Select MySQL Workbench 5.2 SE.

      The MySQL Workbench window appears and displays the primary function modules: SQL Development, Data Modeling, and Server Administration.
2. To use the SQL Development module for queries:

   Click the Open Connection to Start Querying link. The Connect to Database window appears.
3. Enter the server connection information in the Connect to Database window:
   a. Stored Connection: leave unselected
   b. Connection Method: Standard (TCP/IP)
   c. Hostname: localhost (or local system IP address)
   d. Port: 3306
   e. Username: root
   f. Password: Click the Store in Vault button. Enter 'oracle' as the password and click OK.
   g. Default Schema: `world_innodb`
   h. Click OK.
   i. The SQL Editor tab opens, with `world_innodb` selected in the list of schemas. A new query tab (Query 1) opens within the SQL Editor.
4. Execute this `SELECT...FROM` statement (from the previous practice) using the SQL Editor:
   a. Enter this statement in the Query 1 tab:

```
SELECT Continent, Name FROM Country
```

      – Note that the semicolon (`;`) terminator is not required within the SQL Editor.
   b. Click the first button with the gold lightning bolt icon (Execute) at the top of the Query 1 tab to execute the query.
   c. A new results tab (Country 1) appears below the Query 1 tab and contains the results of your query. Confirm that they are identical to the results from the same query in the previous practice.
      – Use the scroll bars to scroll both horizontally and vertically, if needed.

5. Execute this `SELECT ...DISTINCT` statement (from the previous practice) using the SQL Editor:
   a. Delete the previous statement and enter this one:

```
SELECT DISTINCT Region FROM Country
```

   b. Click the Execute button (or press CTRL + ENTER).
6. Execute this `SELECT ...WHERE` statement (from the previous practice) using the SQL Editor:
   a. Delete the previous statement and enter this one:

```
SELECT Name FROM Country
WHERE Population < 1000
```

   b. Click the Execute button (or press CTRL + ENTER).
7. Execute this `SELECT ... ORDER BY` statement (from the previous practice) using the SQL Editor:
   a. Delete the previous statement and enter this one:

```
SELECT CountryCode, Language FROM CountryLanguage
WHERE Language = 'Swedish'
ORDER BY CountryCode DESC
```

8. Execute this `SELECT ... LIMIT` statement (from the previous practice) using the SQL Editor:
   a. Delete the previous statement and enter this one:

```
SELECT Name FROM City
ORDER BY Name ASC
LIMIT 10
```

9. Close the SQL Editor and the MySQL Workbench:
   a. From the File menu, select Exit.
   b. The Workbench window closes.

Solution 7-2: Perform Basic Queries Using MySQL Workbench

There are no solutions for this practice. See the practice task instructions.

Practice 7-3: Perform Basic Queries on the Pets Database

Overview

In this practice, you query the `Pets` database you created in lesson 6. Because currently there is no data in the database, you start by inserting a few rows into the `pet_info` table. Use either the command-line client or the SQL Editor to write and execute the SQL statements. See practice 7-2 for instructions for working with MySQL Workbench.

Note: The course covers the `INSERT` statement in detail in a later lesson.

Duration

This practice takes approximately 35 minutes to complete.

Tasks

1. Start the `mysql` client or MySQL Workbench and set the database to `Pets`.
2. Use a `DESCRIBE` statement to show the structure of the `pet_info` table.
3. Enter the following statement to add three rows of data to the table:

```
INSERT INTO pet_info (pName, pGender, pBday, pDday, oID, tID)
VALUES ('Fluffy', 'F', '2003-02-04', NULL, 1, 1),
('Claws', 'M', '2004-03-17', NULL, 2, 1),
('Buffy', 'F', '1999-05-13', NULL, 1, 2);
```

   – Now that you have some data in the `pet_info` table, you are ready to query the `Pets` database and answer questions about the data.
4. Show all tables in the `Pets` database. Confirm that the `pet_info` table exists.
5. Show all the data in the `pet_info` table.
6. Show only the first row in the `pet_info` table.
7. Who owns Fluffy?

   Note: The owner ID is the only information you have available at this time.
8. What are the names of the cats (pet type ID of 1) born after January 1, 1993?
9. List the distinct genders of all pets.
10. What is the name of the animal which is not a cat (pet type ID not equal to 1)?
11. List the pet IDs and names for Claws and Buffy.
12. List all pet IDs and names for the owner (ID 1) of pets of type 2 or 3.
13. List all pets and their birthdays in ascending order of birth date.
14. List all pet IDs, names, and their birthdays, in descending order of birth date.
15. Exit the `mysql` client or MySQL Workbench.

Solution 7-3: Perform Basic Queries on the Pets Database

Tasks

Note: The solutions use the mysql client, but you can also use MySQL Workbench. The results are the same regardless of which tool you use. See practice 7-2 for instructions for working with MySQL Workbench.

1. Start the `mysql` client or MySQL Workbench and set the database to `Pets`.
   a. Enter the following at the command prompt, and receive the results shown below:

```
cmd> mysql -u root -p
Enter password: oracle
...

mysql> USE Pets
Database changed
```

2. Use a `DESCRIBE` statement to show the structure of the `pet_info` table.
   a. Compare your statement and results to those shown below:

```
mysql> DESC pet_info;
+---------+---------------+------+-----+---------+----------------+
| Field   | Type          | Null | Key | Default | Extra          |
+---------+---------------+------+-----+---------+----------------+
| pID     | int(11)       | NO   | PRI | NULL    | auto_increment |
| pName   | varchar(20)   | NO   |     | NULL    |                |
| pGender | enum('M','F') | YES  |     | NULL    |                |
| pBday   | date          | YES  |     | NULL    |                |
| pDday   | date          | YES  |     | NULL    |                |
| oID     | int(11)       | NO   |     | NULL    |                |
| tID     | int(11)       | NO   |     | NULL    |                |
+---------+---------------+------+-----+---------+----------------+
7 rows in set (0.02 sec)
```

3. Enter the following statement to add three rows of data to the table:
   a. Compare your statement and results to those shown below:

```
mysql> INSERT INTO pet_info (pName, pGender, pBday, pDday, oID,
tID)
    -> VALUES ('Fluffy', 'F', '2003-02-04', NULL, 1, 1),
    -> ('Claws', 'M', '2004-03-17', NULL, 2, 1),
    -> ('Buffy', 'F', '1999-05-13', NULL, 1, 2);
Query OK, 3 rows affected (0.06 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

4. Show all tables in the `Pets` database. Confirm that the `pet_info` table exists.
   a. Compare your statement and results to those shown below:

```
mysql> SHOW TABLES;
+---------------+
| Tables_in_pets |
+---------------+
| owners         |
| pet_info       |
| pet_types      |
+---------------+
```

```
3 rows in set (0.00 sec)
```

5. Show all the data in the `pet_info` table.
   a. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info;
+-----+--------+---------+------------+-------+-----+-----+
| pID | pName  | pGender | pBday      | pDday | oID | tID |
+-----+--------+---------+------------+-------+-----+-----+
|   1 | Fluffy | F       | 2003-02-04 | NULL  |   1 |   1 |
|   2 | Claws  | M       | 2004-03-17 | NULL  |   2 |   1 |
|   3 | Buffy  | F       | 1999-05-13 | NULL  |   1 |   2 |
+-----+--------+---------+------------+-------+-----+-----+
3 rows in set (0.05 sec)
```

6. Show only the first row contained in the `pet_info` table.
   a. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info LIMIT 1;
+-----+--------+---------+------------+-------+-----+-----+
| pID | pName  | pGender | pBday      | pDday | oID | tID |
+-----+--------+---------+------------+-------+-----+-----+
|   1 | Fluffy | F       | 2003-02-04 | NULL  |   1 |   1 |
+-----+--------+---------+------------+-------+-----+-----+
1 row in set (0.05 sec)
```

7. Who owns Fluffy?
   a. Compare your statement and results to those shown below:

```
mysql> SELECT oID FROM pet_info WHERE pName = 'Fluffy';
+-----+
| oID |
+-----+
|   1 |
+-----+
1 row in set (0.08 sec)
```

8. What are the names of the cats (pet type ID of 1) born after January 1, 2003?
   a. Compare your statement and results to those shown below:

```
mysql> SELECT pName FROM pet_info
    -> WHERE tID = 1 AND pBday > '2003-01-01';
+--------+
| pName  |
+--------+
| Fluffy |
| Claws  |
+--------+
2 rows in set (0.02 sec)
```

9. List the distinct genders of all pets.
   a. Compare your statement and results to those shown below:

```
mysql> SELECT DISTINCT pGender FROM pet_info;
+---------+
| pGender |
+---------+
| F       |
| M       |
+---------+
```

```
2 rows in set (0.05 sec)
```

10. What is the name and type (ID) of the animal which is not a cat (pet type ID not equal to 1)?
    a. Compare your statement and results to those shown below:

```
mysql> SELECT pName, tID FROM pet_info WHERE tID != 1;
+-------+-----+
| pName | tID |
+-------+-----+
| Buffy |   2 |
+-------+-----+
1 row in set (0.01 sec)
```

11. List the pet IDs and names for Claws and Buffy.
    a. Compare your statement and results to those shown below:

```
mysql> SELECT pID, pName FROM pet_info
    -> WHERE pName IN ('Claws', 'Buffy');
+-----+-------+
| pId | pName |
+-----+-------+
|   2 | Claws |
|   3 | Buffy |
+-----+-------+
2 rows in set (0.09 sec)
```

12. List all pet IDs and names for the owner (ID 1) of pets of type 2 or 3.
    a. Compare your statement and results to those shown below:

```
mysql> SELECT pID, pName from pet_info
    -> WHERE oID = 1
    -> AND (tID = 2 OR tID=3);
+-----+-------+
| pID | pName |
+-----+-------+
|   3 | Buffy |
+-----+-------+
1 row in set (0.00 sec)
```

13. List all pets and their birthdays in ascending order of birth date.
    a. Compare your statement and results to those shown below:

```
mysql> SELECT pName, pBday FROM pet_info
    -> ORDER BY pBday ASC;
+--------+------------+
| pName  | pBday      |
+--------+------------+
| Buffy  | 1999-05-13 |
| Fluffy | 2003-02-04 |
| Claws  | 2004-03-17 |
+--------+------------+
3 rows in set (0.03 sec)
```

14. List all pet IDs, names, and their birthdays in descending order of birth date.
    a. Compare your statement and results to those shown below:

```
mysql> SELECT pID, pName, pBday FROM pet_info
    -> ORDER BY pBday DESC;
```

```
+-----+--------+------------+
| pID | pName  | pBday      |
+-----+--------+------------+
|   2 | Claws  | 2004-03-17 |
|   1 | Fluffy | 2003-02-04 |
|   3 | Buffy  | 1999-05-13 |
+-----+--------+------------+
3 rows in set (0.01 sec)
```

15. Exit the `mysql` client or MySQL Workbench.

Practices for Lesson 8: Database and Table Maintenance

Chapter 8

Practices for Lesson 8: Overview

Practices Overview

These practices test your knowledge of database and table maintenance. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured the MySQL server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.
- You can access MySQL Workbench if you choose to complete the practices using this tool.

Practice 8-1: Removing a Database

Overview

In this practice, you create a new database and then drop it.

Duration

This practice takes approximately 10 minutes to complete.

Tasks

1. Create a new database called `db1`.
2. Show the list of current databases and confirm that the new database is there.
3. Issue a `DROP DATABASE` statement to remove the entire `db1` database.
4. Show the list of current databases and confirm that the database is gone.

Note: Keep your `mysql` session open for the next practice.

Solutions 8-1: Removing a Database

Tasks

1. Create a new database called `db1`.

   Log in to the `mysql` client program and execute the `CREATE DATABASE` statement shown below:

```
cmd> mysql -uroot -poracle
...

mysql> CREATE DATABASE db1;
Query OK, 1 row affected (0.14 sec)
```

2. Show the list of current databases and confirm that the new database is there.

   Compare your statement and results to those shown below:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| db1                |
| mysql              |
| performance_schema |
| pets               |
| sakila             |
| test               |
| world              |
| world_innodb       |
+--------------------+
9 rows in set (0.00 sec)
```

3. Issue a `DROP DATABASE` statement to remove the entire `db1` database.

   Compare your statement and results to those shown below:

```
mysql> DROP DATABASE db1;
Query OK, 0 rows affected (0.19 sec)
```

4. Show the list of current databases and confirm that the database has gone.

   Compare your statement and results to those shown below:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| pets               |
| sakila             |
| test               |
| world              |
| world_innodb       |
+--------------------+
8 rows in set (0.00 sec)
```

Note: Keep your `mysql` session for the next practice.

Practice 8-2: Creating a New Table and Removing a Table

Overview

In this practice, you create a new table containing records from an existing table. Later, you remove the table and its data.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. View the `CREATE TABLE` statement for the `City` table from the `world_innodb` database.
2. Issue a `CREATE TABLE...SELECT` statement to create a new table called `GelderlandDist` with the `Name`, `District`, and `CountryCode` values from the `City` table, for all cities in the Gelderland district.

   Hint: The new table name does not require quotation marks, but the `District` name does.
3. Show the list of tables. Confirm that the `GelderlandDist` table is now on the list.
4. Select all the rows from the `GelderlandDist` table. Confirm that the table contains the columns specified and four rows of corresponding data.
5. Issue a `CREATE TABLE...LIKE` statement to create a new table called `GelderlandDist2` with the same structure as the `GelderlandDist` table. Execute the appropriate statements to confirm that the structure is the same.
6. Show the list of tables. Confirm that the `GelderlandDist2` table is now on the list.
7. Issue the `DROP TABLE` statement to delete the entire `GelderlandDist2` table. Suppress any errors that would result if the table did not exist.
8. Show the list of tables. Confirm that the `GelderlandDist2` table is gone.

Note: Keep your `mysql` session open to use in the next practice.

Solutions 8-2: Creating a New Table and Removing a Table

Tasks

1. View the `CREATE TABLE` statement for the `City` table from the `world_innodb` database.

   Compare your statement and results to those shown below:

```
mysql> USE world_innodb
Database changed

mysql> SHOW CREATE TABLE City\G
*************************** 1. row ***************************
       Table: City
Create Table: CREATE TABLE `city` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`),
  CONSTRAINT `city_ibfk_1` FOREIGN KEY (`CountryCode`)
     REFERENCES `country` (`Code`)
) ENGINE=InnoDB AUTO_INCREMENT=4080 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

   – Returns the `CREATE TABLE` statement used for `City`.

2. Issue a `CREATE TABLE...SELECT` statement to create a new table called `GelderlandDist` with the `Name`, `District`, and `CountryCode` values from the `City` table, for all cities in the Gelderland district.

   Compare your statement and results to those shown below:

```
mysql> CREATE TABLE GelderlandDist AS
    ->   SELECT Name, District, CountryCode
    ->   FROM City
    ->   WHERE District = 'Gelderland';
Query OK, 4 rows affected (0.22 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

3. Show the list of tables. Confirm that the `GelderlandDist` table is now on the list.

   Compare your statement and results to those shown below:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| city                   |
| country                |
| country2               |
| countrylanguage        |
| gelderlanddist         |
+------------------------+
5 rows in set (0.22 sec)
```

4. Select all the rows from the `GelderlandDist` table. Confirm that the table contains the columns specified and four rows of corresponding data.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM GelderlandDist;
+-----------+------------+-------------+
| Name      | District   | CountryCode |
+-----------+------------+-------------+
| Apeldoorn | Gelderland | NLD         |
| Nijmegen  | Gelderland | NLD         |
| Arnhem    | Gelderland | NLD         |
| Ede       | Gelderland | NLD         |
+-----------+------------+-------------+
4 rows in set (0.00 sec)
```

5. Issue a `CREATE TABLE...LIKE` statement to create a new table called `GelderlandDist2` with the same structure as the `GelderlandDist` table. Execute the appropriate statements to confirm that the structure is the same.

   a. Create the new table:

```
mysql> CREATE TABLE GelderlandDist2 LIKE GelderlandDist;
Query OK, 0 rows affected (0.01 sec)
```

   b. Confirm that the structure is the same as the `GelderlandDist` table:

```
mysql> DESC GelderlandDist2;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| Name         | char(20) | NO   | PRI |         |       |
| District     | char(20) | NO   |     |         |       |
| CountryCode  | char(3)  | NO   |     |         |       |
+--------------+----------+------+-----+---------+-------+
3 rows in set (0.05 sec)
```

6. Show the list of tables. Confirm that the `GelderlandDist2` table is now on the list.

   Compare your statement and results to those shown below:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| city                   |
| country                |
| country2               |
| countrylanguage        |
| gelderlanddist         |
| gelderlanddist2        |
+------------------------+
6 rows in set (0.22 sec)
```

7. Issue the `DROP TABLE` statement to delete the entire `GelderlandDist2` table. Suppress any errors that would result if the table did not exist.

   Compare your statement and results to those shown below:

```
mysql> DROP TABLE IF EXISTS GelderlandDist2;
Query OK, 0 rows affected (0.03 sec)
```

8. Show the list of tables. Confirm that the `GelderlandDist2` table is gone.

   Compare your statement and results to those shown below:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| city                   |
| country                |
| country2               |
| countrylanguage        |
| gelderlanddist         |
+------------------------+
5 rows in set (0.33 sec)
```

   – `GelderlandDist2` is no longer on the list.

Practice 8-3: Altering Table Columns

Overview

In this practice, you modify the data type of a column and add a new column to a table.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. Use the `DESCRIBE` statement to view the structure of the `GelderlandDist` table.
2. Use an `ALTER TABLE` statement to modify the `Name` column in the `GelderlandDist` table to have a character data type with a length of `20`.
3. View the table structure to confirm the change.
4. Use an `ALTER TABLE` statement to add a new column called `Inauguration`, which holds date information and does not allow `NULL` values.
5. Confirm the addition of the new column.
6. List all the row data in the table to inspect the new column's values.

Note: Keep your `mysql` session open to use in the next practice.

Solutions 8-3: Altering Table Columns

Tasks

1. Use the `DESCRIBE` statement to view the structure of the `GelderlandDist` table.

   Compare your statement and results to those shown below:

```
mysql> DESCRIBE GelderlandDist;
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| Name        | char(35) | NO   |     |         |       |
| District    | char(20) | NO   |     |         |       |
| CountryCode | char(3)  | NO   |     |         |       |
+-------------+----------+------+-----+---------+-------+
3 rows in set (0.38 sec)
```

   – The output shows the current attributes for each column in the table.

2. Use an `ALTER TABLE` statement to modify the `Name` column in the `GelderlandDist` table to have a character data type with a length of **20**.

   Compare your statement and results to those shown below:

```
mysql> ALTER TABLE GelderlandDist
    -> MODIFY Name char(20);
Query OK, 4 rows affected (0.14 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

3. View the table structure to confirm the change.

   Compare your statement and results to those shown below:

```
mysql> DESCRIBE GelderlandDist;
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| Name        | char(20) | YES  |     | NULL    |       |
| District    | char(20) | NO   |     |         |       |
| CountryCode | char(3)  | NO   |     |         |       |
+-------------+----------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

   – Note that the `Null` column value for the `Name` field is now `YES`, indicating that the column allows null values. Columns allow null values by default when you do not specify `NOT NULL`.

4. Use an `ALTER TABLE` statement to add a new column called `Inauguration`, which holds date information and does not allow `NULL` values.

   Compare your statement and results to those shown below:

```
mysql> ALTER TABLE GelderlandDist
    -> ADD Inauguration DATE NOT NULL;
Query OK, 4 rows affected (0.01 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

   Note: If the `NO_ZERO_IN_DATE` or `NO_ZERO_DATE` SQL modes are set, you get an error.

5. Confirm the addition of the new column.

   Compare your statement and results to those shown below:

```
mysql> DESCRIBE GelderlandDist;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| Name         | char(20) | YES  |     | NULL    |       |
| District     | char(20) | NO   |     |         |       |
| CountryCode  | char(3)  | NO   |     |         |       |
| Inauguration | date     | NO   |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

6. List all the row data in the table to inspect the new column's values.

   Compare your statement and results to those shown below::

```
mysql> SELECT * FROM GelderlandDist;
+-----------+------------+-------------+--------------+
| Name      | District   | CountryCode | Inauguration |
+-----------+------------+-------------+--------------+
| Apeldoorn | Gelderland | NLD         | 0000-00-00   |
| Nijmegen  | Gelderland | NLD         | 0000-00-00   |
| Arnhem    | Gelderland | NLD         | 0000-00-00   |
| Ede       | Gelderland | NLD         | 0000-00-00   |
+-----------+------------+-------------+--------------+
4 rows in set (0.00 sec)
```

   – Note that the `Inauguration` column values have been set to an "empty date".

Note: Keep your `mysql` session open for the next practice.

Practice 8-4: Modifying Table Indexes and Constraints

Overview

In this practice, you modify the table index and constraints.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. Show the `CREATE TABLE` statement for the `City` table.
2. Issue an `ALTER TABLE` statement to add an index to the `Name` column in the `City` table. Call the index `CityName`.
3. Show the `CREATE TABLE` statement for the `City` table. Examine the part of the statement that adds the `CityName` index.
4. Use an `ALTER TABLE` statement to drop the `CityName` key/index from the `City` table.
5. Confirm that the index has gone.
6. Show the `GelderlandDist` table structure. You are going to alter this table structure, so review it first.
7. Issue an `ALTER TABLE` statement to add a primary key to the `Name` column in the `GelderlandDist` table.
8. Confirm the existence of the primary key.

   Note: Keep your `mysql` session open for the next practice.

   Important: The changes you make to the `world_innodb` database in these lessons are cumulative. Do not attempt to undo any changes you make to the database in this or future practices.

Solutions 8-4: Modifying Table Indexes and Constraints

Tasks

1. Show the `CREATE TABLE` statement for the `City` table.

   Compare your statement and results to those shown below::

```
mysql> SHOW CREATE TABLE City\G
*********************** 1. row ***********************
       Table: City
Create Table: CREATE TABLE `city` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`),
  CONSTRAINT `city_ibfk_1` FOREIGN KEY (`CountryCode`)
     REFERENCES `country` (`Code`)
) ENGINE=InnoDB AUTO_INCREMENT=4080 DEFAULT CHARSET=latin1
1 row in set (0.08 sec)
```

2. Issue an `ALTER TABLE` statement to add an index to the `Name` column in the `City` table. Call the index `CityName`.

   Compare your statement and results to those shown below:

```
mysql> ALTER TABLE City
    -> ADD INDEX CityName(Name);
Query OK, 0 rows affected (0.16 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

3. Show the `CREATE TABLE` statement for the `City` table. Examine the part of the statement that adds the `CityName` index.

   Compare your statement and results to those shown below:

```
mysql> SHOW CREATE TABLE City\G
*************************** 1. row ***************************
       Table: City
Create Table: CREATE TABLE `city` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`),
  KEY `CityName` (`Name`),
  CONSTRAINT `city_ibfk_1` FOREIGN KEY (`CountryCode`)
     REFERENCES `country` (`Code`)
) ENGINE=InnoDB AUTO_INCREMENT=4080 DEFAULT CHARSET=latin1
```

```
1 row in set (0.00 sec)
```

4. Use an `ALTER TABLE` statement to drop the `CityName` key/index from the `City` table.

   Compare your statement and results to those shown below:

```
mysql> ALTER TABLE City
    -> DROP INDEX CityName;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

5. Confirm that the index has gone.

   Compare your statement and results to those shown below::

```
mysql> SHOW CREATE TABLE City\G
*********************** 1. row ***********************
       Table: City
Create Table: CREATE TABLE `city` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`),
  CONSTRAINT `city_ibfk_1` FOREIGN KEY (`CountryCode`)
     REFERENCES `country` (`Code`)
) ENGINE=InnoDB AUTO_INCREMENT=4080 DEFAULT CHARSET=latin1
1 row in set (0.08 sec)
```

   – The `CityName` index no longer appears in the statement.

6. Show the `GelderlandDist` table structure.

   Compare your statement and results to those shown below:

```
mysql> DESCRIBE GelderlandDist;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| Name         | char(20) | YES  |     | NULL    |       |
| District     | char(20) | NO   |     |         |       |
| CountryCode  | char(3)  | NO   |     |         |       |
| Inauguration | date     | NO   |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

   – The `DESCRIBE` statement also shows information about a table's structure.

7. Issue an `ALTER TABLE` statement to add a primary key to the `Name` column in the `GelderlandDist` table.

   Compare your statement and results to those shown below:

```
mysql> ALTER TABLE GelderlandDist
    -> ADD PRIMARY KEY(Name);
```

```
Query OK, 4 rows affected (0.03 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

8. Confirm the existence of the primary key.

   Compare your statement and results to those shown below::

```
mysql> DESC GelderlandDist;
+--------------+----------+------+-----+---------+-------+
| Field        | Type     | Null | Key | Default | Extra |
+--------------+----------+------+-----+---------+-------+
| Name         | char(20) | NO   | PRI |         |       |
| District     | char(20) | NO   |     |         |       |
| CountryCode  | char(3)  | NO   |     |         |       |
| Inauguration | date     | NO   |     | NULL    |       |
+--------------+----------+------+-----+---------+-------+
4 rows in set (0.05 sec)
```

   – The `PRI` indicator in the `KEY` field shows that the `Name` is now the primary key. Note that the `Name` column does not allow null values now that it is the primary key.

Note: Keep your `mysql` session open for the next practice.

Important: The changes you make to the `world_innodb` database in these lessons are cumulative. Do not attempt to undo any changes you make to the database in this or future practices.

Practice 8-5: Further Practice

Overview

In this practice, you reinforce your knowledge of database and table maintenance with these additional tasks:

- Creating a new table
- Adding, modifying, and removing column indexes and constraints
- Dropping a table

Duration

This practice takes approximately 20 minutes to complete.

Tasks

1. Show the `CREATE TABLE` statement for the `City` table. You are going to use `City` as the basis of a new table, so review its structure first.
2. Add a new table to `world_innodb` called `Big_Cities` using a `SELECT` on the `City` table. Create `Big_Cities` so it contains the `ID`, `Name`, and `Population` columns for all cities with a population greater than eight million. Confirm the addition of the table.
3. Examine the table structure to confirm that the specified columns are included and their data types match.
4. Display all the records in the `Big_Cities` table.
5. Add a `Founded` column to store the date each city was established. Allow `NULL` values. View the table structure to confirm.
6. View the contents of the `Big_Cities` table and confirm that the new column exists. Note that all its values are nulls.
7. Remove the `Founded` column. View the structure to confirm its deletion.
8. Change the `ID` column so that it accepts null values. View the table structure to confirm.
9. Make the `ID` column the primary key. Confirm the change.
10. Create an index (called `Pop`) on the `Population` column. Examine the structure of `Big_Cities` and its `CREATE TABLE` statement to confirm the change.
11. Remove the index on the `Population` column. Confirm the change.
12. Remove the `Big_Cities` table. View the list of tables in the `world_innodb` database to confirm that it has gone.
13. Exit the `mysql` client.

Solutions 8-5: Further Practice

Tasks

1. Show the `CREATE TABLE` statement for the `City` table.

   Compare your statement and results to those shown below:

```
mysql> SHOW CREATE TABLE City\G
*************************** 1. row ***************************
       Table: city
Create Table: CREATE TABLE `city` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`),
  CONSTRAINT `city_ibfk_1` FOREIGN KEY (`CountryCode`) REFERENCES
`country` (`Code`)
) ENGINE=InnoDB AUTO_INCREMENT=4080 DEFAULT CHARSET=latin1
1 row in set (0.00 sec)
```

2. Create a new table called `Big_Cities` by using a `SELECT` on the `City` table. `Big_Cities` should contain the `ID`, `Name`, and `Population` columns for all cities with a population greater than eight million. Confirm the addition of the table.

   a. Compare your statement and results to those shown below:

```
mysql> CREATE TABLE Big_Cities
    -> SELECT id, name, population from City
    -> WHERE population > 8000000;
Query OK, 10 rows affected (0.03 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

   b. Confirm the addition of the table:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| big_cities             |
| city                   |
| country                |
| country2               |
| countrylanguage        |
| gelderlanddist         |
+------------------------+
6 rows in set (0.01 sec)
```

3. Examine the table structure to confirm that the specified columns are included and their data types match.

   Compare your statement and results to those shown below:

```
mysql> DESCRIBE Big_Cities;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| id         | int(11)  | NO   |     | 0       |       |
| name       | char(35) | NO   |     |         |       |
| population | int(11)  | NO   |     | 0       |       |
+------------+----------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

   – The three specified columns are present with the same attributes as those in the `City` table.

4. Display all the records in the `Big_Cities` table.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM Big_Cities;
+------+------------------+------------+
| id   | name             | population |
+------+------------------+------------+
|  206 | São Paulo        |    9968485 |
|  939 | Jakarta          |    9604900 |
| 1024 | Mumbai (Bombay)  |   10500000 |
| 1890 | Shanghai         |    9696300 |
| 2331 | Seoul            |    9981619 |
| 2515 | Ciudad de México |    8591309 |
| 2822 | Karachi          |    9269265 |
| 3357 | Istanbul         |    8787958 |
| 3580 | Moscow           |    8389200 |
| 3793 | New York         |    8008278 |
+------+------------------+------------+
10 rows in set (0.00 sec)
```

5. Add a `Founded` column to store the date each city was established. Allow `NULL` values. View the table structure to confirm.

   a. Compare your statement and results to those shown below:

```
mysql> ALTER TABLE Big_Cities ADD COLUMN Founded DATE NULL;
Query OK, 0 rows affected (0.32 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   b. View the structure to confirm:

```
mysql> DESC Big_Cities;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| id         | int(11)  | NO   |     | 0       |       |
| name       | char(35) | NO   |     |         |       |
| population | int(11)  | NO   |     | 0       |       |
| Founded    | date     | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
4 rows in set (0.03 sec)
```

   – The new column has been added and allows null values.

6. View the contents of the `Big_Cities` table and confirm that the new column exists.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM Big_Cities;
+------+------------------+------------+---------+
| id   | name             | population | Founded |
+------+------------------+------------+---------+
|  206 | Sпo Paulo        |    9968485 | NULL    |
|  939 | Jakarta          |    9604900 | NULL    |
| 1024 | Mumbai (Bombay)  |   10500000 | NULL    |
| 1890 | Shanghai         |    9696300 | NULL    |
| 2331 | Seoul            |    9981619 | NULL    |
| 2515 | Ciudad de MΘxico |    8591309 | NULL    |
| 2822 | Karachi          |    9269265 | NULL    |
| 3357 | Istanbul         |    8787958 | NULL    |
| 3580 | Moscow           |    8389200 | NULL    |
| 3793 | New York         |    8008278 | NULL    |
+------+------------------+------------+---------+
10 rows in set (0.00 sec)
```

   – The new column defaults to null values.

7. Remove the `Founded` column.

   a. Compare your statement and results to those shown below:

```
mysql> ALTER TABLE Big_Cities DROP Founded;
Query OK, 0 rows affected (0.32 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   b. View the structure to confirm its deletion:

```
mysql> DESC Big_Cities;
+------------+---------+------+-----+---------+-------+
| Field      | Type    | Null | Key | Default | Extra |
+------------+---------+------+-----+---------+-------+
| id         | int(11) | NO   |     | 0       |       |
| name       | char(35)| NO   |     |         |       |
| population | int(11) | NO   |     | 0       |       |
+------------+---------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

8. Change the `ID` column so that it accepts null values. View the table structure to confirm.

   a. Compare your statement and results to those shown below:

```
mysql> ALTER TABLE Big_Cities MODIFY ID INT(11) NULL;
Query OK, 0 rows affected (0.32 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   b. View the structure to confirm:

```
mysql> DESC Big_Cities;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| ID         | int(11)  | YES  |     | NULL    |       |
| name       | char(35) | NO   |     |         |       |
| population | int(11)  | NO   |     | 0       |       |
+------------+----------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

9. Make the `ID` column the primary key. Confirm the change.
   a. Compare your statement and results to those shown below:

```
mysql> ALTER TABLE Big_Cities ADD PRIMARY KEY (ID);
Query OK, 0 rows affected (0.57 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   b. Confirm the change:

```
mysql> DESC Big_Cities;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| ID         | int(11)  | NO   | PRI |         |       |
| name       | char(35) | NO   |     |         |       |
| population | int(11)  | NO   |     | 0       |       |
+------------+----------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

10. Create an index (called `Pop`) on the `Population` column.
   a. Compare your statement and results to those shown below:

```
mysql> ALTER TABLE Big_Cities ADD INDEX Pop (Population);
Query OK, 0 rows affected (0.17 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

   b. Examine the structure of `Big_Cities`:

```
mysql> DESC Big_Cities;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| ID         | int(11)  | NO   | PRI |         |       |
| name       | char(35) | NO   |     |         |       |
| population | int(11)  | NO   | MUL | 0       |       |
+------------+----------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

   – The `MUL` in the `Key` field of the `population` column indicates that it participates in a multi-column index.

   c. View the `CREATE TABLE` statement for `Big_Cities`:

```
mysql> SHOW CREATE TABLE Big_Cities\G
*************************** 1. row ***********************
       Table: Big_Cities
Create Table: CREATE TABLE `big_cities` (
  `ID` int(11) NOT NULL DEFAULT '0',
  `name` char(35) NOT NULL DEFAULT '',
  `population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `Pop` (`population`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.03 sec)
```

11. Remove the index on the `Population` column.
   a. Compare your statement and results to those shown below:

```
mysql> ALTER TABLE Big_Cities DROP INDEX Pop;
Query OK, 0 rows affected (0.16 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

b. Confirm the change:

```
mysql> DESC Big_Cities;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| ID         | int(11)  | NO   | PRI |         |       |
| name       | char(35) | NO   |     |         |       |
| population | int(11)  | NO   |     | 0       |       |
+------------+----------+------+-----+---------+-------+
3 rows in set (0.02 sec)
```

– The `Key` attribute for the `population` column is empty.

c. Display the `CREATE TABLE` statement for `Big_Cities`:

```
mysql> SHOW CREATE TABLE Big_Cities\G
*************************** 1. row ***********************
       Table: Big_Cities
Create Table: CREATE TABLE `big_cities` (
  `ID` int(11) NOT NULL DEFAULT '0',
  `name` char(35) NOT NULL DEFAULT '',
  `population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.03 sec)
```

– The `Pop` key index has gone.

12. Remove the `Big_Cities` table.

a. Compare your statement and results to those shown below:

```
mysql> DROP TABLE Big_Cities;
Query OK, 0 rows affected (0.13 sec)
```

b. View the list of tables in the `world_innodb` database to confirm:

```
mysql> SHOW TABLES;
+------------------------+
| Tables_in_world_innodb |
+------------------------+
| city                   |
| country                |
| country2               |
| countrylanguage        |
| gelderlanddist         |
+------------------------+
5 rows in set (0.01 sec)
```

– The `Big_Cities` table has gone.

13. Exit the `mysql` client.

Practices for Lesson 9: Table Data Manipulation

Chapter 9

Practices for Lesson 9: Overview

Practices Overview

These practices test your knowledge of table data manipulation. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you may need to adjust file locations.

Assumptions

- You have installed and configured MySQL Server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.

Note: In this practice the first letters of table names are in uppercase. Windows is not case-sensitive but some operating systems are, so it is good practice to use proper capitalization. The SQL statements are all in uppercase for clarity, but this is not required.

Practice 9-1: Inserting and Replacing Table Row Data

Overview

In this practice, you add new records to a table and replace existing records.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. Use an `INSERT INTO` statement to add a single row to the `GelderlandDist` table. The new row includes the city name Sakila, with a district of Gelderland, country code of SQL, and an inauguration date of July 01, 2001.

   Hint: Use the date format: YYYY-MM-DD. Enclose column values within single quotation marks.
2. List the contents of the `GelderlandDist` table to confirm that the new row is in the table.
3. Insert two more rows into the `GelderlandDist` table:
   - Row 1: The city name MySQLland, with a country code of MYS, and inauguration date of August 04, 1984
   - Row 2: The city name Fantasia, with country code of FNT, and inauguration date of January 1, 1950

   Both cities are in the Gelderland district.
4. List the contents of the `GelderlandDist` table to confirm that the new rows are there.
5. Use a `REPLACE INTO` statement to replace the MySQLland record, so that it is in the same country as Sakila.

   Hint: Save yourself some typing. Use the up-arrow key (↑) to retrieve the last `INSERT INTO` statement. Change `INSERT` to `REPLACE` and provide the new values.
6. List the contents of the `GelderlandDist` table to verify the change.

   Note: Keep your `mysql` session open for the next practice.

Solutions 9-1: Inserting and Replacing Table Row Data

Tasks

1. Use an `INSERT INTO` statement to add a single row to the `GelderlandDist` table. The new row includes the city name Sakila, with a district of Gelderland, country code of SQL, and an inauguration date of July 01, 2001.
   a. Log in to the `mysql` client program:

```
cmd> mysql -u root -p
Enter password: oracle
...
```

   b. Compare your statements and results to those shown below:

```
mysql> USE world_innodb
Database changed

mysql> INSERT INTO GelderlandDist (Name, District, CountryCode,
Inauguration)
    -> VALUES ('Sakila', 'Gelderland', 'SQL', '2001-07-01');
Query OK, 1 row affected (0.09 sec)
```

   – The result shows that the operation affects a single row.

2. List the contents of the `GelderlandDist` table to confirm that the new row is in the table:

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM GelderlandDist;
+-----------+------------+-------------+--------------+
| Name      | District   | CountryCode | Inauguration |
+-----------+------------+-------------+--------------+
| Apeldoorn | Gelderland | NLD         | 0000-00-00   |
| Arnhem    | Gelderland | NLD         | 0000-00-00   |
| Ede       | Gelderland | NLD         | 0000-00-00   |
| Nijmegen  | Gelderland | NLD         | 0000-00-00   |
| Sakila    | Gelderland | SQL         | 2001-07-01   |
+-----------+------------+-------------+--------------+
5 rows in set (0.00 sec)
```

   – Returns a list of all rows and columns in the table including the new record

3. Insert two more rows into the `GelderlandDist` table:
   – Row 1: The city name MySQLland, with a country code of MYS, and inauguration date of August 04, 1984
   – Row 2: The city name Fantasia, with country code of FNT, and inauguration date of January 1, 1950

   Compare your statement and results to those shown below:

```
mysql> INSERT INTO GelderlandDist (Name, District, CountryCode,
Inauguration)
    -> VALUES ('MySQLland', 'Gelderland', 'MYS', '1984-08-04'),
    ->        ('Fantasia', 'Gelderland', 'FNT', '1950-01-01');
Query OK, 2 rows affected (0.02 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

4. List the contents of the `GelderlandDist` table to confirm that the new rows are there.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM GelderlandDist;
+-----------+------------+-------------+--------------+
| Name      | District   | CountryCode | Inauguration |
+-----------+------------+-------------+--------------+
| Apeldoorn | Gelderland | NLD         | 0000-00-00   |
| Arnhem    | Gelderland | NLD         | 0000-00-00   |
| Ede       | Gelderland | NLD         | 0000-00-00   |
| Fantasia  | Gelderland | FNT         | 1950-01-01   |
| MySQLland | Gelderland | MYS         | 1984-08-04   |
| Nijmegen  | Gelderland | NLD         | 0000-00-00   |
| Sakila    | Gelderland | SQL         | 2001-07-01   |
+-----------+------------+-------------+--------------+
7 rows in set (0.30 sec)
```

   – Returns a list of all rows and columns in the table including the new records

5. Use a `REPLACE INTO` statement to replace the MySQLland record, so that it is in the same country as Sakila.

   Compare your statement and results to those shown below:

```
mysql> REPLACE INTO GelderlandDist (Name, District, CountryCode,
Inauguration)
    -> VALUES ('MySQLland', 'Gelderland', 'SQL', '1984-08-04');
Query OK, 2 rows affected (0.08 sec)
```

   – You need to provide values for all columns, even those that are unchanged.

6. List the contents of the `GelderlandDist` table to verify the change.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM GelderlandDist;
+-----------+------------+-------------+--------------+
| Name      | District   | CountryCode | Inauguration |
+-----------+------------+-------------+--------------+
| Apeldoorn | Gelderland | NLD         | 0000-00-00   |
| Arnhem    | Gelderland | NLD         | 0000-00-00   |
| Ede       | Gelderland | NLD         | 0000-00-00   |
| Fantasia  | Gelderland | FNT         | 1950-01-01   |
| MySQLland | Gelderland | SQL         | 1984-08-04   |
| Nijmegen  | Gelderland | NLD         | 0000-00-00   |
| Sakila    | Gelderland | SQL         | 2001-07-01   |
+-----------+------------+-------------+--------------+
7 rows in set (0.30 sec)
```

   – Returns a list of all rows and columns in the table and shows MySQLland in the country “SQL”

Note: Keep your `mysql` session open to use in the next practice.

Practice 9-2: Modifying and Deleting Table Row Data

Overview

In this practice, you modify and delete records from the `GelderlandDist` table.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. Use an `UPDATE` statement on the `GelderlandDist` table to set the inauguration date for the city named Ede to May 17, 1880.

   Hint: Use the date format: YYYY-MM-DD. Enclose column values within single quotation marks.
2. List the contents of the `GelderlandDist` table to verify the change.
3. Update the `GelderlandDist` table so that the first two cities with a country code of NLD (when ordered alphabetically by `Name`) use the code FOO instead.
4. List the contents of the `GelderlandDist` table to verify the change.
5. Issue a `DELETE` statement to remove a single row from the `GelderlandDist` table, with a country code of FOO.
6. List the contents of the `GelderlandDist` table to verify the change.
7. List all cities in the `City` table with the country code FOO.
8. Try to delete one row from the `City` table with the country code FOO.

Solutions 9-2: Modifying and Deleting Table Row Data

Tasks

1. Use an `UPDATE` statement on the `GelderlandDist` table to set the inauguration date for the city named Ede to May 17, 1880.

   Compare your statement and results to those shown below:

```
mysql> UPDATE GelderlandDist
    -> SET Inauguration = '1880-05-17'
    -> WHERE Name = 'Ede';
Query OK, 1 row affected (0.14 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

2. List the contents of the `GelderlandDist` table to verify the change.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM GelderlandDist;
+-----------+------------+-------------+--------------+
| Name      | District   | CountryCode | Inauguration |
+-----------+------------+-------------+--------------+
| Apeldoorn | Gelderland | NLD         | 0000-00-00   |
| Arnhem    | Gelderland | NLD         | 0000-00-00   |
| Ede       | Gelderland | NLD         | 1880-05-17   |
| Fantasia  | Gelderland | FNT         | 1950-01-01   |
| MySQLland | Gelderland | SQL         | 1984-08-04   |
| Nijmegen  | Gelderland | NLD         | 0000-00-00   |
| Sakila    | Gelderland | SQL         | 2001-07-01   |
+-----------+------------+-------------+--------------+
7 rows in set (0.00 sec)
```

3. Update the `GelderlandDist` table so that the first two cities with a country code of NLD (when ordered alphabetically by `Name`) use the code FOO instead.

   Compare your statement and results to those shown below:

```
mysql> UPDATE GelderlandDist
    -> SET CountryCode = 'FOO'
    -> WHERE CountryCode = 'NLD'
    -> ORDER BY Name
    -> LIMIT 2;
Query OK, 2 rows affected (0.06 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

4. List the contents of the `GelderlandDist` table to verify the change.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM GelderlandDist;
+-----------+------------+-------------+--------------+
| Name      | District   | CountryCode | Inauguration |
+-----------+------------+-------------+--------------+
| Apeldoorn | Gelderland | FOO         | 0000-00-00   |
| Arnhem    | Gelderland | FOO         | 0000-00-00   |
| Ede       | Gelderland | NLD         | 1880-05-17   |
| Fantasia  | Gelderland | FNT         | 1950-01-01   |
| MySQLland | Gelderland | SQL         | 1984-08-04   |
| Nijmegen  | Gelderland | NLD         | 0000-00-00   |
| Sakila    | Gelderland | SQL         | 2001-07-01   |
+-----------+------------+-------------+--------------+
7 rows in set (0.00 sec)
```

   – The updated rows are at the top of the table, due to the default alphabetical ordering.

5. Issue a `DELETE` statement to remove a single row from the `GelderlandDist` table, with a country code of FOO.

   Compare your statement and results to those shown below:

```
mysql> DELETE FROM GelderlandDist
    -> WHERE CountryCode = 'FOO'
    -> LIMIT 1;
Query OK, 1 row affected (0.02 sec)
```

6. List the contents of the `GelderlandDist` table to verify the change.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM GelderlandDist;
+-----------+------------+-------------+--------------+
| Name      | District   | CountryCode | Inauguration |
+-----------+------------+-------------+--------------+
| Arnhem    | Gelderland | FOO         | 0000-00-00   |
| Ede       | Gelderland | NLD         | 1880-05-17   |
| Fantasia  | Gelderland | FNT         | 1950-01-01   |
| MySQLland | Gelderland | SQL         | 1984-08-04   |
| Nijmegen  | Gelderland | NLD         | 0000-00-00   |
| Sakila    | Gelderland | SQL         | 2001-07-01   |
+-----------+------------+-------------+--------------+
6 rows in set (0.00 sec)
```

   – The first record (Apeldoorn) has been deleted.

7. List all cities in the `City` table with the country code FOO.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM City WHERE CountryCode = 'FOO';
Empty set (0.05 sec)
```

   – There are no rows in the City table with a country code of FOO.

8. Try to delete one row from the `City` table with the country code FOO.

   Compare your statement and results to those shown below:

```
mysql> DELETE FROM City
    -> WHERE CountryCode = 'FOO'
    -> LIMIT 1;
Query OK, 0 rows affected (0.02 sec)
```

   – There are no records with a country code of FOO, so the `DELETE` does not affect any rows.

Practice 9-3: Manipulating Table Row Data in the `Pets` Database

Overview

In this practice, you make the following changes to the data in the `Pets` database:

- Add table row data.
- Replace current table row data.
- Modify table row data.
- Delete table rows.

Duration

This practice takes approximately 40 minutes to complete.

Tasks

1. Change the current database to `Pets`.
2. List all records in the `pet_info` table.
3. Using the pet information chart below:
   a. Insert the ten remaining records into the `pet_info` table
      - Use the correct column name and order.
      - Not every pet (row) has values for all columns. Use `NULL` for these columns.
      - You added data for Fluffy, Claws, and Buffy in a previous practice. Start with the data for Fang in the fourth row.

pID*	pName	pGender	pBday	pDday	oID**	tID***
1	Fluffy	F	2003-02-04	NULL	1	1
2	Claws	M	2004-03-17	NULL	2	1
3	Buffy	F	1999-05-13	NULL	1	2
4	Fang	M	2000-08-27	NULL	3	2
5	Bowser	M	1989-08-31	2009-07-29	4	2
6	Chirpy	F	2008-09-11	NULL	2	3
7	Whistler	NULL	2007-12-09	NULL	2	4
8	Slim	M	2006-04-29	NULL	3	5
9	Puffball	F	2009-03-30	NULL	4	1
10	Opus	M	NULL	NULL	5	1
11	Rocky	M	1998-04-04	2013-02-11	6	1
12	Koko	M	1997-02-08	NULL	3	1
13	Scruffy	M	2008-04-17	NULL	2	1

   b. List the contents of the `pet_info` table to confirm the changes.

4. Using the owner information chart below:
   a. Insert the records into the `owners` table

oID*	oName	oPhone
1	Harold	15554159855
2	Gwen	15551234567
3	Benny	15553456789
4	Diane	15554567890
5	Caryn	15554444444
6	Chris	15556666666

   b. List the contents of the `owners` table to confirm the changes.
5. Using the pet type information chart below:
   a. Insert the rows into the `pet_types` table

tID*	pType	pCategory
1	Cat	Mammal
2	Dog	Mammal
3	Parrot	Bird
4	Canary	Bird
5	Snake	Reptile
6	Hamster	Mammal
7	Ferret	Mammal

   b. List the contents of the `pet_types` table to confirm the changes
6. Update Whistler's details in the `pet_info` table:
   a. Set the gender to male.
   b. Review the table data to confirm the change.
7. Benny's snake, Slim, is actually an iguana. You must correct this. The update affects two tables: `pet_info` and `pet_types`:
   a. Add an Iguana record to the `pet_types` table, which belongs to the Reptile category.
      Note: Do not attempt to include foreign keys at this time. You add foreign keys to the `Pets` database in a later lesson.
   b. Modify the `pet_info` table to use Slim's new `tID`.
   c. Confirm the changes to both tables.
8. Harold has given Buffy the dog to Benny. Make the necessary changes to the database.
   a. Update the `pet_info` table to assign Buffy to Benny.
      Hint: Check the `owners` table first to confirm the owner IDs.
   b. Confirm the change.

9. Two of the owners, Caryn and Chris, marry and share a new phone number: 16163429988.
   a. Update Chris and Caryn's records in the owners table with the new number.
      Hint: Use a `WHERE...IN` statement to include both owner names in the update.
   b. Confirm the changes.
10. All the animals born before 2000 are no longer patients. Remove them from the database.
   a. Delete all the animals from the `pet_info` table that were born before 2000.
   b. Confirm the change.
11. After removing the pets in the previous step, there is an owner without any pets.
   a. Determine who the owner is and delete that person's record from the `owners` table.
   b. Confirm the change.
12. Diane has moved and is not bringing her hamster Puffball to the clinic anymore. The clinic wants to reuse her ID number for a new client. Make the necessary changes to the database.
   a. Use `REPLACE INTO` to delete Puffball's record and replace it with the new pet's details:
      - The new pet is a female hamster named Chewy, owned by Olga.
      - Specify Puffball's ID in the statement.
      - Use the next available number for the owner ID.
      - Get the correct type ID for this pet from the `pet_types` table.
   b. Confirm the change.
13. Add the new owner Olga to the owners table. Her phone number is 18563330000. You must specify the owner ID.
14. Exit the `mysql` client.

    Note: You use the `Pets` database in future practices, so do not make any further changes to the tables.

Solutions Practice 9-3: Manipulating Table Row Data in the `Pets` Database

Tasks

1. Change the current database to `Pets`.

   Compare your statement and results to those shown below:

```
mysql> USE pets
Database changed
```

2. List all the records in the `pet_info` table. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info;
+-----+--------+---------+------------+-------+-----+-----+
| pID | pName  | pGender | pBday      | pDday | oID | tID |
+-----+--------+---------+------------+-------+-----+-----+
|   1 | Fluffy | F       | 2003-02-04 | NULL  |   1 |   1 |
|   2 | Claws  | M       | 2004-03-17 | NULL  | 2 |   1 |
|   3 | Buffy  | F       | 1999-05-13 | NULL  |   1 |   2 |
+-----+--------+---------+------------+-------+-----+-----+
3 rows in set (0.09 sec)
```

3. Using the pet information chart:

   a. Insert the ten remaining records into the `pet_info` table. Compare your statement and results to those shown below:

```
mysql> INSERT INTO pet_info (pName, pGender, pBday, pDday, oID, tID)
    -> VALUES
    -> ('Fang','M','2000-08-27',NULL,3,2),
    -> ('Bowser','M','1989-08-31','2009-07-29',4,2),
    -> ('Chirpy','F','2008-09-11',NULL,2,3),
    -> ('Whistler',NULL,'2007-12-09',NULL,2,4),
    -> ('Slim','M','2006-04-29',NULL,3,5),
    -> ('Puffball','F','2009-03-30',NULL,4,1),
    -> ('Opus','F',NULL,NULL,5,1),
    -> ('Rocky','M','1998-04-04','2013-02-11',6,1),
    -> ('Koko','M','1997-02-08',NULL,3,1),
    -> ('Scruffy','M','2008-04-17',NULL,2,1);
Query OK, 10 rows affected (0.09 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

   – Note that `AUTO_INCREMENT` assigns the `pID` automatically.

b. List the contents of the `pet_info` table to confirm the changes. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info;
+-----+---------+---------+------------+------------+-----+-----+
| pID | pName   | pGender | pBday      | pDday      | oID | tID |
+-----+---------+---------+------------+------------+-----+-----+
|   1 | Fluffy  | F       | 2003-02-04 | NULL       |   1 |   1 |
|   2 | Claws   | M       | 2004-03-17 | NULL       |   2 |   1 |
|   3 | Buffy   | F       | 1999-05-13 | NULL       |   1 |   2 |
|   4 | Fang    | M       | 2000-08-27 | NULL       |   3 |   2 |
|   5 | Bowser  | M       | 1989-08-31 | 2009-07-29 |   4 |   2 |
|   6 | Chirpy  | F       | 2008-09-11 | NULL       |   2 |   3 |
|   7 | Whistler| NULL    | 2007-12-09 | NULL       |   2 |   4 |
|   8 | Slim    | M       | 2006-04-29 | NULL       |   3 |   5 |
|   9 | Puffball| F       | 2009-03-30 | NULL       |   4 |   1 |
|  10 | Opus    | F       | NULL       | NULL       |   5 |   1 |
|  11 | Rocky   | M       | 1998-04-04 | 2013-02-11 |   6 |   1 |
|  12 | Koko    | M       | 1997-02-08 | NULL       |   3 |   1 |
|  13 | Scruffy | M       | 2008-04-17 | NULL       |   2 |   1 |
+-----+---------+---------+------------+------------+-----+-----+
13 rows in set (0.00 sec)
```

4. Using the owner information chart:
   a. Insert the records into the `owners` table. Compare your statement and results to those shown below:

```
mysql> INSERT INTO owners (oName, oPhone)
    -> VALUES
    -> ('Harold', '15554159855'),
    -> ('Gwen', '15551234567'),
    -> ('Benny', '15553456789'),
    -> ('Diane', '15554567890'),
    -> ('Caryn', '15554444444'),
    -> ('Chris', '15556666666');
Query OK, 6 rows affected (0.02 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

   b. List the contents of the `owners` table to confirm the changes. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM owners;
+-----+--------+-------------+
| oID | oName  | oPhone      |
+-----+--------+-------------+
|   1 | Harold | 15554159855 |
|   2 | Gwen   | 15551234567 |
|   3 | Benny  | 15553456789 |
|   4 | Diane  | 15554567890 |
|   5 | Caryn  | 15554444444 |
|   6 | Chris  | 15556666666 |
+-----+--------+-------------+
6 rows in set (0.00 sec)
```

5. Using the pet type information chart:
   a. Insert the rows into the `pet_types` table. Compare your statement and results to those shown below:

```
mysql> INSERT INTO pet_types (pType, pCategory)
    -> VALUES
    -> ('Cat', 'Mammal'),
    -> ('Dog', 'Mammal'),
    -> ('Parrot', 'Bird'),
    -> ('Canary', 'Bird'),
    -> ('Snake', 'Reptile'),
    -> ('Hamster', 'Mammal'),
    -> ('Ferret', 'Mammal');
Query OK, 7 rows affected (0.01 sec)
Records: 7  Duplicates: 0  Warnings:
```

   b. List the contents of the `pet_types` table to confirm the changes. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_types;
+-----+---------+-----------+
| tID | pType   | pCategory |
+-----+---------+-----------+
|   1 | Cat     | Mammal    |
|   2 | Dog     | Mammal    |
|   3 | Parrot  | Bird      |
|   4 | Canary  | Bird      |
|   5 | Snake   | Reptile   |
|   6 | Hamster | Mammal    |
|   7 | Ferret  | Mammal    |
+-----+---------+-----------+
7 rows in set (0.00 sec)
```

6. Update Whistler's details in the `pet_info` table:
   a. Set the gender to male. Compare your statement and results to those shown below:

```
mysql> UPDATE pet_info
    -> SET pGender = 'M'
    -> WHERE pName = 'Whistler';
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

b. Review the table data to confirm the change. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info;
+-----+----------+---------+------------+------------+-----+-----+
| pID | pName    | pGender | pBday      | pDday      | oID | tID |
+-----+----------+---------+------------+------------+-----+-----+
|   1 | Fluffy   | F       | 2003-02-04 | NULL       |   1 |   1 |
|   2 | Claws    | M       | 2004-03-17 | NULL       |   2 |   1 |
|   3 | Buffy    | F       | 1999-05-13 | NULL       |   1 |   2 |
|   4 | Fang     | M       | 2000-08-27 | NULL       |   3 |   2 |
|   5 | Bowser   | M       | 1989-08-31 | 2009-07-29 |   4 |   2 |
|   6 | Chirpy   | F       | 2008-09-11 | NULL       |   2 |   3 |
|   7 | Whistler | M       | 2007-12-09 | NULL       |   2 |   4 |
|   8 | Slim     | M       | 2006-04-29 | NULL       |   3 |   5 |
|   9 | Puffball | F       | 2009-03-30 | NULL       |   4 |   1 |
|  10 | Opus     | F       | NULL       | NULL       |   5 |   1 |
|  11 | Rocky    | M       | 1998-04-04 | 2013-02-11 |   6 |   1 |
|  12 | Koko     | M       | 1997-02-08 | NULL       |   3 |   1 |
|  13 | Scruffy  | M       | 2008-04-17 | NULL       |   2 |   1 |
+-----+----------+---------+------------+------------+-----+-----+
13 rows in set (0.00 sec)
```

7. Benny's snake, Slim, is actually an iguana. You must correct this. The update affects two tables: `pet_info` and `pet_types`.

   a. Add an Iguana record to the `pet_types` table, which belongs to the Reptile category. Compare your statement and results to those shown below:

```
mysql> INSERT INTO pet_types (pType, pCategory)
    -> VALUES ('Iguana', 'Reptile');
Query OK, 1 row affected (0.03 sec)
```

   Note: Do not attempt to include foreign keys at this time. You add foreign keys to the Pets database in a later lesson.

   b. Modify the `pet_info` table to use Slim's new `tID`. Compare your statement and results to those shown below:

```
mysql> UPDATE pet_info
    -> SET tID = 8
    -> WHERE pName = 'Slim';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

c. Confirm the changes to both tables. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info;
+-----+----------+---------+------------+------------+-----+-----+
| pID | pName    | pGender | pBday      | pDday      | oID | tID |
+-----+----------+---------+------------+------------+-----+-----+
|   1 | Fluffy   | F       | 2003-02-04 | NULL       |   1 |   1 |
|   2 | Claws    | M       | 2004-03-17 | NULL       |   2 |   1 |
|   3 | Buffy    | F       | 1999-05-13 | NULL       |   1 |   2 |
|   4 | Fang     | M       | 2000-08-27 | NULL       |   3 |   2 |
|   5 | Bowser   | M       | 1989-08-31 | 2009-07-29 |   4 |   2 |
|   6 | Chirpy   | F       | 2008-09-11 | NULL       |   2 |   3 |
|   7 | Whistler | M       | 2007-12-09 | NULL       |   2 |   4 |
|   8 | Slim     | M       | 2006-04-29 | NULL       |   3 |   8 |
|   9 | Puffball | F       | 2009-03-30 | NULL       |   4 |   1 |
|  10 | Opus     | F       | NULL       | NULL       |   5 |   1 |
|  11 | Rocky    | M       | 1998-04-04 | 2013-02-11 |   6 |   1 |
|  12 | Koko     | M       | 1997-02-08 | NULL       |   3 |   1 |
|  13 | Scruffy  | M       | 2008-04-17 | NULL       |   2 |   1 |
+-----+----------+---------+------------+------------+-----+-----+
13 rows in set (0.00 sec)
```

8. Harold has given Buffy the dog to Benny. Make the necessary changes to the database.
   a. Update the `pet_info` table to assign Buffy to Benny. Compare your statements and results to those shown below:

```
mysql> SELECT * FROM owners;
+-----+--------+-------------+
| oID | oName  | oPhone      |
+-----+--------+-------------+
|   1 | Harold | 15554159855 |
|   2 | Gwen   | 15551234567 |
|   3 | Benny  | 15553456789 |
...
+-----+--------+-------------+

mysql> UPDATE pet_info SET oID = 3
    -> WHERE oID = 1 AND pName = 'Buffy';
Query OK, 1 row affected (4.16 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

b. Confirm the change. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info;
+-----+----------+---------+------------+------------+-----+-----+
| pID | pName    | pGender | pBday      | pDday      | oID | tID |
+-----+----------+---------+------------+------------+-----+-----+
|   1 | Fluffy   | F       | 2003-02-04 | NULL       |   1 |   1 |
|   2 | Claws    | M       | 2004-03-17 | NULL       |   2 |   1 |
|   3 | Buffy    | F       | 1999-05-13 | NULL       |   3 |   2 |
. . .
13 rows in set (0.00 sec)
```

9. Two of the owners, Caryn and Chris, marry and share a new phone number: 16163429988.
   a. Update Chris and Caryn's records in the owners table with the new number. Compare your statement and results to those shown below:

```
mysql> UPDATE owners
    -> SET oPhone = '16163429988'
    -> WHERE oName IN ('Caryn', 'Chris');
Query OK, 2 rows affected (0.13 sec)
Rows matched: 2  Changed: 2  Warnings: 0
```

   b. Confirm the changes. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM owners;
+-----+--------+-------------+
| oID | oName  | oPhone      |
+-----+--------+-------------+
|   1 | Harold | 15554159855 |
|   2 | Gwen   | 15551234567 |
|   3 | Benny  | 15553456789 |
|   4 | Diane  | 15554567890 |
|   5 | Caryn  | 16163429988 |
|   6 | Chris  | 16163429988 |
+-----+--------+-------------+
6 rows in set (0.02 sec)
```

10. All the animals born before 2000 are no longer patients. Remove them from the database.
    a. Delete all the animals from the `pet_info` table that were born before 2000. Compare your statement and results to those shown below:

```
mysql> DELETE FROM pet_info
    -> WHERE pBday < '2000-01-01';
Query OK, 4 rows affected (0.05 sec)
```

b. Confirm the change. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info;
+-----+----------+---------+------------+-------+-----+-----+
| pID | pName    | pGender | pBday      | pDday | oID | tID |
+-----+----------+---------+------------+-------+-----+-----+
|   1 | Fluffy   | F       | 2003-02-04 | NULL  |   1 |   1 |
|   2 | Claws    | M       | 2004-03-17 | NULL  |   2 |   1 |
|   4 | Fang     | M       | 2000-08-27 | NULL  |   3 |   2 |
|   6 | Chirpy   | F       | 2008-09-11 | NULL  |   2 |   3 |
|   7 | Whistler | M       | 2007-12-09 | NULL  |   2 |   4 |
|   8 | Slim     | M       | 2006-04-29 | NULL  |   3 |   8 |
|   9 | Puffball | F       | 2009-03-30 | NULL  |   4 |   1 |
|  10 | Opus     | F       | NULL       | NULL  |   5 |   1 |
|  13 | Scruffy  | M       | 2008-04-17 | NULL  |   2 |   1 |
+-----+----------+---------+------------+-------+-----+-----+
9 rows in set (0.00 sec)
```

– Note that there are now only nine rows, as opposed to the original 13.

11. After removing the pets in the previous step, there is an owner without any pets.

a. Determine who the owner is and delete that person's record from the `owners` table. Compare your statements and results to those shown below:

```
mysql> SELECT DISTINCT oID FROM pet_info;
+-----+
| oID |
+-----+
|   1 |
|   2 |
|   3 |
|   4 |
|   5 |
+-----+
5 rows in set (0.09 sec)

mysql> SELECT * FROM owners;
+-----+--------+-------------+
| oID | oName  | oPhone      |
+-----+--------+-------------+
|   1 | Harold | 15554159855 |
|   2 | Gwen   | 15551234567 |
|   3 | Benny  | 15553456789 |
|   4 | Diane  | 15554567890 |
|   5 | Caryn  | 16163429988 |
|   6 | Chris  | 16163429988 |
+-----+--------+-------------+
6 rows in set (0.00 sec)

mysql> DELETE FROM owners
    -> WHERE oID = 6;
Query OK, 1 row affected (0.05 sec)
```

– The **oID** number 6 (Chris) is missing from the `pet_info` table.

b. Confirm the change. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM owners;
+-----+--------+-------------+
| oID | oName  | oPhone      |
+-----+--------+-------------+
|   1 | Harold | 15554159855 |
|   2 | Gwen   | 15551234567 |
|   3 | Benny  | 15553456789 |
|   4 | Diane  | 15554567890 |
|   5 | Caryn  | 16163429988 |
+-----+--------+-------------+
5 rows in set (0.00 sec)
```

12. Diane has moved and is not bringing her hamster Puffball to the clinic any more. The clinic wants to reuse her ID number for a new client. Make the necessary changes to the database.

    a. Use `REPLACE INTO` to delete Puffball's record and replace it with the new pet's details. Compare your statement and results to those shown below:

```
mysql> REPLACE INTO pet_info (pID, pName, pGender, pBday, pDday,
oID, tID)
    -> VALUES (9,'Chewy','F', NULL, NULL,6,6);
Query OK, 2 rows affected (0.03 sec)
```

    b. Confirm the change. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info;
+-----+----------+---------+------------+-------+-----+-----+
| pID | pName    | pGender | pBday      | pDday | oID | tID |
+-----+----------+---------+------------+-------+-----+-----+
|   1 | Fluffy   | F       | 2003-02-04 | NULL  |   1 |   1 |
|   2 | Claws    | M       | 2004-03-17 | NULL  |   2 |   1 |
|   4 | Fang     | M       | 2000-08-27 | NULL  |   3 |   2 |
|   6 | Chirpy   | F       | 2008-09-11 | NULL  |   2 |   3 |
|   7 | Whistler | M       | 2007-12-09 | NULL  |   2 |   4 |
|   8 | Slim     | M       | 2006-04-29 | NULL  |   3 |   8 |
|   9 | Chewy    | F       | NULL       | NULL  |   6 |   6 |
|  10 | Opus     | F       | NULL       | NULL  |   5 |   1 |
|  13 | Scruffy  | M       | 2008-04-17 | NULL  |   2 |   1 |
+-----+----------+---------+------------+-------+-----+-----+
9 rows in set (0.00 sec)
```

13. Add the new owner Olga to the owners table. Her phone number is 18563330000. You must specify the owner ID.

    a. Compare your statement and results to those shown below:

```
mysql> INSERT INTO owners (oID, oName, oPhone)
    -> VALUES (6,'Olga','18563330000');
Query OK, 1 row affected (0.02 sec)
```

b. Confirm the change. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM owners;
+-----+--------+-------------+
| oID | oName  | oPhone      |
+-----+--------+-------------+
|   1 | Harold | 15554159855 |
|   2 | Gwen   | 15551234567 |
|   3 | Benny  | 15553456789 |
|   4 | Diane  | 15554567890 |
|   5 | Caryn  | 16163429988 |
|   6 | Olga   | 18563330000 |
+-----+--------+-------------+
6 rows in set (0.00 sec)
```

14. Exit the `mysql` client:

    Note: You use the `Pets` database in future practices, so do not make any further changes to the tables.

Practices for Lesson 10: Functions

Chapter 10

Practices for Lesson 10: Overview

Practices Overview

These practices test your knowledge of functions. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured the MySQL server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.
- You can access MySQL Workbench if you choose to complete the practices using this tool.

Note: In this practice, the first letters of table names are in uppercase. Windows is not case-sensitive but some operating systems are, so it is good practice to use proper capitalization. The SQL statements are all in uppercase for clarity, but this is not required.

Practice 10-1: Quiz – Functions

Overview

In this quiz, you answer questions about MySQL functions.

Duration

This practice takes approximately 10 minutes to complete.

Quiz Questions

Choose the best answer from those provided.

1. Parentheses are optional in function calls.
   a. True
   b. False
2. The following query calculates the difference between the old and current gross national product of Montserrat. The result is `NULL` because the value of `GNPOld` is `NULL`. This means you cannot determine the result of the subtraction.

```
mysql> SELECT Name, GNP, GNP - GNPOld
-> FROM Country
-> WHERE Name = 'Montserrat';
+------------+--------+---------------+
| Name       | GNP    | GNP - GNPOld  |
+------------+--------+---------------+
| Montserrat | 109.00 | NULL          |
+------------+--------+---------------+
```

   a. True
   b. False

3. What is the default format of the `TIMESTAMP` date component?
   a. HH:MI:SS YYYY-MM-DD
   b. `TIMESTAMP`
   c. YYYY-MM-DD HH:MI:SS
   d. All of the above
4. The `CHAR_LENGTH` function returns the number of _____ in a specified string.
   a. Numerals
   b. Characters
   c. Strings
   d. Bytes
5. `ROUND` is a _____ function that rounds a number to the _____ integer.
   a. String, exact
   b. Temporal, nearest
   c. Numerical, lowest
   d. Numerical, closest
6. MySQL does not support geometrical and trigonometric functions.
   a. True
   b. False
7. Which aggregate function calculates the average of a group of rows?
   a. `WITH ROLLUP`
   b. `CALC()`
   c. `GROUP_CONCAT()`
   d. None of the above
8. In a `SELECT` statement, if a `WHERE` clause selects 20 rows and a `GROUP BY` arranges them into four groups of five rows each, a summary function produces a value for each of the four groups.
   a. True
   b. False
9. There can never be a space between a function name and the subsequent parenthesis.
   a. True
   b. False

Solutions 10-1: Quiz – Functions

Quiz Solutions

1. **b**. False
2. **a**. True
3. **c**. YYYY-MM-DD HH:MI:SS
4. **b**. Characters
5. **d**. Numerical, closest
6. **b**. False
7. **d**. The function to use is `AVG()`. `WITH ROLLUP` is a modifier (not a function) that is used with the `AVG()` function to perform a “super-aggregate” operation.
8. **a**. True
9. **b**. False. You can allow spaces by setting the `IGNORE_SPACE` SQL mode.

Practice 10-2: Using Built-In, String, and Temporal Functions

Overview

In this practice, you use built-in, string, and temporal functions.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. Show the version of MySQL that is currently running on your system.
2. Compare the string sort order of the words “awake” to “asleep”, and “awake” to “awake” and “asleep” to “awake”, all in one function call.
3. Combine the following words to result in a complete sentence: I ' 'am ' 'mostly ' 'awake!'
4. Determine the partial string that starts on the sixth character of the string “HarryMonkey”.
5. Look up the available formats for the `DATE_FORMAT()` function in the online reference: http://dev.mysql.com/doc/refman/5.6/en/date-and-time-functions.html
6. Show the current day of the week, date, month, and year in the following format: “Tuesday the 4th of June in the year 2009”.
7. What weekday is it 500 days from now?

Note: Keep your `mysql` session open for the next practice.

Solutions 10-2: Using Built-In, String, and Temporal Functions

Tasks

1. Show the version of MySQL that is currently running on your system:

   Compare your statement and results to those shown below:

```
shell> mysql -uroot -poracle
...

mysql> SELECT VERSION();
+---------------------------------------+
| VERSION()                             |
+---------------------------------------+
| 5.6.10-enterprise-commercial-advanced |
+---------------------------------------+
1 row in set (0.09 sec)
```

2. Compare the string sort order of the words “awake” to “asleep”, and “awake” to “awake” and “asleep” to “awake”, all in one function call:

   Compare your statement and results to those shown below:

```
mysql> SELECT strcmp('awake','asleep'),
    -> strcmp('awake','awake'),
    -> strcmp('asleep','awake');
+-------------------------+-----------------------+-------------------------+
| strcmp('awake','asleep')| strcmp('awake','awake')| strcmp('asleep','awake')|
+-------------------------+-----------------------+-------------------------+
|                       1 |                     0 |                      -1 |
+-------------------------+-----------------------+-------------------------+
1 row in set (0.05 sec)
```

   – Returns three comparisons, showing the number of character differences.

3. Combine the following words to result in a complete sentence: 'I ' 'am ' 'mostly ' 'awake!'

   Compare your statement and results to those shown below:

```
mysql> SELECT CONCAT('I ','am ','mostly ', 'awake!');
+----------------------------------------+
| CONCAT('I ','am ','mostly ', 'awake!') |
+----------------------------------------+
| I am mostly awake!                     |
+----------------------------------------+
1 row in set (0.02 sec)
```

   – Returns the completed sentence without the quotation marks

4. Determine the partial string that starts on the sixth character of the string “HarryMonkey”:

   Compare your statement and results to those shown below:

```
mysql> SELECT SUBSTRING('HarryMonkey', 6);
+-----------------------------+
| SUBSTRING('HarryMonkey', 6) |
+-----------------------------+
| Monkey                      |
+-----------------------------+
1 row in set (0.02 sec)
```

   – Returns the last six characters of the original string.

5. Look up the available formats for the `DATE_FORMAT()` function in the online reference: http://dev.mysql.com/doc/refman/5.6/en/date-and-time-functions.html
   - There is a table that lists the format specifiers under the `DATE_FORMAT` entry.
6. Show the current day of the week, date, month, and year in the following format: "Tuesday the 4th of June in the year 2009":

   Compare your statement and results to those shown below:

```
mysql> SELECT DATE_FORMAT(NOW(),
    -> '%W the %D of %M in the year %Y');
+-----------------------------------------------------+
| DATE_FORMAT(NOW(), '%W the %D of %M in the year %Y') |
+-----------------------------------------------------+
| Tuesday the 2nd of April in the year 2013           |
+-----------------------------------------------------+
1 row in set (0.03 sec)
```

   - Returns the current date in the format specified
7. What weekday is it 500 days from now?

   Compare your statement and results to those shown below:

```
mysql> SELECT DAYNAME(NOW() + INTERVAL 500 DAY);
+-----------------------------------+
| DAYNAME(NOW() + INTERVAL 500 DAY) |
+-----------------------------------+
| Friday                            |
+-----------------------------------+
1 row in set (0.01 sec)
```

   - The result depends on when you execute the function.

   Note: Keep your `mysql` session open for the next practice.

Practice 10-3: Using Numeric and Control Flow Functions

Overview

In this practice, you use numeric and control flow functions.

Duration

This practice takes approximately 10 minutes to complete.

Tasks

1. Round the numbers -8.6 and 8.6 down to the nearest, smaller integer.
2. Examine the year of independence column for each record in the `Country` table and use a `CASE` statement to put the dates into appropriate categories. Display the results in descending year order.

Year	Text
Before 1300	'Ancient'
Before 1800	'Really Old'
Before 1900	'Not Old'
Before 2000	'New'
All others	'Brand New'

Note: Keep your `mysql` session open for the next practice.

Solutions 10-3: Using Numeric and Control Flow Functions

Tasks

1. Round the numbers -8.6 and 8.6 down to nearest (smaller) integer.

   Compare your statement and results to those shown below:

```
mysql> SELECT FLOOR(-8.6), FLOOR(8.6);
+-------------+------------+
| FLOOR(-8.6) | FLOOR(8.6) |
+-------------+------------+
|          -9 |          8 |
+-------------+------------+
1 row in set (0.03 sec)
```

   – Shows the numbers rounded down

2. Examine the year of independence column for each record in the `Country` table and use a `CASE` statement to put the dates into appropriate categories. Display the results in descending year order.

   Compare your statement and results to those shown below:

```
mysql> USE world_innodb
Database changed

mysql> SELECT IndepYear, Name,
    -> CASE
    -> WHEN IndepYear < 1300 then 'Ancient'
    -> WHEN IndepYear < 1800 then 'Really Old'
    -> WHEN IndepYear < 1900 then 'Not Old'
    -> WHEN IndepYear < 2000 then 'New'
    -> ELSE 'Brand New'
    -> END
    -> FROM Country
    -> ORDER BY IndepYear DESC;
+-----------+--------------------------+-----------------------------+
| IndepYear | Name                     | CASE                        |
...
+-----------+--------------------------+-----------------------------+
|      1994 | Palau                    | New                         |
|      1993 | Czech Republic           | New                         |
|      1993 | Slovakia                 | New                         |
...
|      1878 | Romania                  | Not Old                     |
|      1867 | Luxembourg               | Not Old                     |
|      1867 | Canada                   | Not Old                     |
...
|      1776 | United States            | Really Old                  |
|      1769 | Nepal                    | Really Old                  |
|      1581 | Netherlands              | Really Old                  |
...
|      1278 | Andorra                  | Ancient                     |
|      1143 | Portugal                 | Ancient                     |
|      1066 | United Kingdom           | Ancient                     |
...
|      NULL | Turks and Caicos Islands | Brand New                   |
|      NULL | French Polynesia         | Brand New                   |
```

```
|      NULL | Svalbard and Jan Mayen    | Brand New                      |
...
```

Note: Keep your `mysql` session open for the next practice.

Practice 10-4: Using Aggregate Functions

Overview

In this practice, you use aggregate functions.

Duration

This practice takes approximately 20 minutes to complete.

Tasks

1. List all the continents from the `Country` table, and the total population of each continent.
2. List the average life expectancy on each continent, rounded to the closest integer.
3. List the average population of each city in the `City` table, where that figure is over 500,000. Group the results by country code.
4. Find the five most common government forms in the world.
5. List the average surface area per country on each continent.
6. List the average surface area per country on each continent, and the average surface area of all countries.
7. Exit the `mysql` client.

Solutions 10-4: Using Aggregate Functions

Tasks

1. List all the continents from the `Country` table, and the total population of each continent:

   Compare your statement and results to those shown below:

```
mysql> USE world_innodb
Database changed

mysql> SELECT Continent, SUM(Population)
    -> FROM Country
    -> GROUP BY Continent;
+---------------+-----------------+
| Continent     | SUM(Population) |
+---------------+-----------------+
| Asia          |      3705025700 |
| Europe        |       730074600 |
| North America |       482993000 |
| Africa        |       784475000 |
| Oceania       |        30401150 |
| Antarctica    |               0 |
| South America |       345780000 |
+---------------+-----------------+
7 rows in set (0.00 sec)
```

2. List the average life expectancy on each continent, rounded to the closest integer.

   Compare your statement and results to those shown below:

```
mysql> SELECT Continent, ROUND(AVG(LifeExpectancy))
    -> FROM Country
    -> GROUP BY Continent;
+---------------+----------------------------+
| Continent     | ROUND(AVG(LifeExpectancy)) |
+---------------+----------------------------+
| Asia          |                         67 |
| Europe        |                         75 |
| North America |                         73 |
| Africa        |                         53 |
| Oceania       |                         70 |
| Antarctica    |                       NULL |
| South America |                         71 |
+---------------+----------------------------+
7 rows in set (0.00 sec)
```

3. List the average population of each city in the `City` table, where that figure is over 500,000. Group the results by country code:

   Compare your statement and results to those shown below:

```
mysql> SELECT CountryCode, AVG(Population) AS AvgPop
    -> FROM City
    -> GROUP BY CountryCode
    -> HAVING AVG(Population) > 500000;
+-------------+--------------+
| CountryCode | AvgPop       |
```

```
+-------------+-------------+
| AFG         |  583025.0000 |
| AGO         |  512320.0000 |
| ARM         |  544366.6667 |
| AUS         |  808119.0000 |
| AZE         |  616000.0000 |
| CAF         |  524000.0000 |
| CIV         |  638227.4000 |
| CMR         |  503222.0000 |
| COD         |  548034.1667 |
| COG         |  725000.0000 |
| COL         |  532920.7895 |
| EGY         |  542785.9189 |
| GIN         | 1090610.0000 |
| HKG         | 1650316.5000 |
| IRQ         |  595069.4000 |
| KOR         |  557141.3286 |
| LBN         |  670000.0000 |
| LBR         |  850000.0000 |
| LBY         |  674251.7500 |
| MLI         |  809552.0000 |
| MNG         |  773700.0000 |
| PAK         |  534690.5932 |
| PER         |  552147.3636 |
| SGP         | 4017733.0000 |
| SLE         |  850000.0000 |
| THA         |  662763.4167 |
| UGA         |  890800.0000 |
| URY         | 1236000.0000 |
+-------------+-------------+
28 rows in set (0.23 sec)
```

– The optional AS clause provides an alias for the resulting column of data.

4. Find the five most common government forms in the world:

Compare your statement and results to those shown below:

```
mysql> SELECT GovernmentForm, COUNT(GovernmentForm) AS
Governments
    -> FROM Country
    -> GROUP BY GovernmentForm
    -> ORDER BY Governments DESC
    -> LIMIT 5;
+--------------------------------+-------------+
| GovernmentForm                 | Governments |
+--------------------------------+-------------+
| Republic                       |         122 |
| Constitutional Monarchy        |          29 |
| Federal Republic               |          15 |
| Dependent Territory of the UK  |          12 |
| Monarchy                       |           5 |
+--------------------------------+-------------+
5 rows in set (0.00 sec)
```

5. List the average surface area per country on each continent.

   Compare your statement and results to those shown below:

```
mysql> SELECT Continent, AVG(SurfaceArea) AS AverageSurfaceArea
    -> FROM Country
    -> GROUP BY Continent;
+---------------+--------------------+
| Continent     | AverageSurfaceArea |
+---------------+--------------------+
| Asia          |      625117.745098 |
| Europe        |      501068.128261 |
| North America |      654445.135135 |
| Africa        |      521558.224138 |
| Oceania       |      305867.642857 |
| Antarctica    |     2626420.200000 |
| South America |     1276066.142857 |
+---------------+--------------------+
```

6. List the average surface area per country on each continent, and the average surface area of all countries:

   Compare your statement and results to those shown below:

```
mysql> SELECT Continent, AVG(SurfaceArea) AS AverageSurfaceArea
    -> FROM Country
    -> GROUP BY Continent
    -> WITH ROLLUP;
+---------------+--------------------+
| Continent     | AverageSurfaceArea |
+---------------+--------------------+
| Asia          |      625117.745098 |
| Europe        |      501068.128261 |
| North America |      654445.135135 |
| Africa        |      521558.224138 |
| Oceania       |      305867.642857 |
| Antarctica    |     2626420.200000 |
| South America |     1276066.142857 |
| NULL          |      623248.146025 |
+---------------+--------------------+
8 rows in set (0.00 sec)
```

   – Note that the average of all countries appears as a **NULL** column at the bottom of the list.

7. Exit the `mysql` client.

Practices for Lesson 11: Exporting and Importing Data

Chapter 11

Practices for Lesson 11: Overview

Practices Overview

These practices test your knowledge of exporting and importing database data. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured the MySQL server.
- You have created and populated the `world_innodb` database.
- You can access the mysql client from a command-line prompt.

Note: In this practice, the first letters of table names are in uppercase. Windows is not case-sensitive but some operating systems are, so it is good practice to use proper capitalization. The SQL statements are all in uppercase for clarity, but this is not required.

Practice 11-1: Quiz – Exporting and Importing Data

Overview

In this quiz, you answer questions about exporting and importing data.

Duration

This practice takes approximately 15 minutes to complete.

Quiz Questions

Choose the best answer from those provided.

1. You can export data by using a MySQL query.
   a. True
   b. False
2. Which of the following are valid reasons to export your database data?
   a. If tables are lost or damaged
   b. If there is a system crash, to minimize data loss
   c. If a copy of the database is needed on another server
   d. To load data into external applications
   e. To transfer data between RDBMSs
   f. All of the above
3. Use the `SELECT` statement with the _____ clause to write the results directly into a file.
   a. `INTO OUTFILE`
   b. `INSERT INTO`
   c. `LOAD DATA INFILE`
   d. `LOAD DATA OUTFILE`
4. Which of the following clauses do you use to send query results to a text file containing comma-separated values?
   a. `OUTFILE WITH ','`
   b. `FIELDS TERMINATED BY ','`
   c. `ENCLOSED BY ','`
   d. Any of the above
5. The _____ utility writes table contents to a file.
   a. `mysqlimport`
   b. `INTO OUTFILE`
   c. `mysqloutfile`
   d. `mysqldump`
6. Use the _____ operator with `mysqldump` to indicate the export file name and location.
   a. `<`
   b. `=`
   c. `>`
   d. `:`
7. You cannot import data from the shell prompt.

a. True
b. False

8. A SQL statement file (`.sql`) can create and populate tables. Use the _____ statement within the MySQL client to execute this file.
   a. `mysql-import`
   b. `INPUT FILE`
   c. `IMPORT`
   d. `SOURCE`
9. Which of the following statements imports a file containing data only (not SQL statements)?
   a. `LOAD DATA INFILE`
   b. `LOAD DATA OUTFILE`
   c. `LOAD SOURCE FILE`
   d. None of the above.
10. The `mysqlimport` client program imports data files into tables.
    a. True
    b. False
11. Which option displays detailed usage information for the import/export utilities?
    a. `-h`
    b. `-info`
    c. `--help`
12. When using the `mysqlimport` utility, the destination tables must already be present and the input files can contain only _____ values.
    a. Column
    b. Data
    c. Database
    d. Table
13. When importing a file by using `LOAD DATA INFILE`, MySQL assumes that the file resides on the server host, unless you specify otherwise.
    a. True
    b. False

Solutions 11-1: Quiz – Exporting and Importing Data

Quiz Solutions

1. **a**. True
2. **f**. All of the above
3. **a**. `INTO OUTFILE`
4. **b**. `FIELDS TERMINATED BY ','`
5. **d**. `mysqldump`
6. **c**. `>` (redirect)
7. **b**. False
8. **d**. `SOURCE`
9. **a**. `LOAD DATA INFILE`
10. **a**. True
11. **c**. `--help`
12. **b**. Data
13. **a**. True

Practice 11-2: Exporting Files by Using a Query

Overview

In this practice, you use `SELECT …INTO OUTFILE` to export database data.

Assumptions

You have access to the `Pets` database, which you populated with data in the "Data Manipulation" practice.

Duration

This practice takes approximately 20 minutes to complete.

Tasks

1. Set the `Pets` database as the default database.
2. Execute a `SELECT...INTO OUTFILE` statement to export the `pet_info` table to a text file. Put this file in the `D:\labs` directory. Examine the text file and confirm that it contains all the rows and columns from the table.

   Hints:

   – In the Oracle classroom environment, use Notepad++ to view the text files in the correct format.

   – The contents are the same as the output of a `SELECT * FROM pet_info` statement.
3. Execute a query to export the `owners` table to a text file in CSV format. Put this file in the `D:\labs` directory. Examine the text file and note the format. Confirm that it contains all the rows and columns from the table.

   Note: In the CSV format, a carriage return character terminates each row, double quotation marks enclose each value, and commas separate the values.
4. Using the `world_innodb` database, back up the `CountryLanguage` table to a standard text file. Put this file in the `D:\labs` directory. Examine the text file and confirm that it contains all the rows and columns from the table.

Note: Keep your `mysql` session open for the next practice.

Solutions 11-2: Exporting Files by Using a Query

Tasks

1. Set the `Pets` database as the default database.
   a. Log in to the `mysql` client program:

```
cmd> mysql -u root -p
Enter password: oracle
...
```

   b. Compare your statement and results to those shown below:

```
mysql> USE Pets
Database changed
```

2. Execute a `SELECT...INTO OUTFILE` statement to export the `pet_info` table to a text file. Put this file in the `D:\labs` directory.
   a. Compare your statement and results to those shown below:

```
mysql> SELECT * INTO OUTFILE 'D:/labs/pet_info.txt'
    -> FROM pet_info;
Query OK, 9 rows affected (0.00 sec)
```

   – The statement creates a text file in the `D:\labs` directory.
   b. Open the text file in Notepad++ and examine the contents:

```
1    Fluffy     F     2003-02-04          \N    1     1
2    Claws      M     2004-03-17          \N    2     1
4    Fang       M     2000-08-27          \N    3     2
6    Chirpy     F     2008-09-11          \N    2     3
7    Whistler   M     2007-12-09          \N    2     4
8    Slim       M     2006-04-29          \N    3     8
9    Chewy      F     \N                  \N    6     6
10   Opus       F     \N                  \N    5     1
13   Scruffy    M     2008-04-17          \N    2     1
```

   c. Confirm that it contains all the rows and columns from the table. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info;
+-----+----------+---------+------------+-------+-----+-----+
| pID | pName    | pGender | pBday      | pDday | oID | tID |
+-----+----------+---------+------------+-------+-----+-----+
|   1 | Fluffy   | F       | 2003-02-04 | NULL  |   1 |   1 |
|   2 | Claws    | M       | 2004-03-17 | NULL  |   2 |   1 |
|   4 | Fang     | M       | 2000-08-27 | NULL  |   3 |   2 |
|   6 | Chirpy   | F       | 2008-09-11 | NULL  |   2 |   3 |
|   7 | Whistler | M       | 2007-12-09 | NULL  |   2 |   4 |
|   8 | Slim     | M       | 2006-04-29 | NULL  |   3 |   8 |
|   9 | Chewy    | F       | NULL       | NULL  |   6 |   6 |
|  10 | Opus     | F       | NULL       | NULL  |   5 |   1 |
|  13 | Scruffy  | M       | 2008-04-17 | NULL  |   2 |   1 |
+-----+----------+---------+------------+-------+-----+-----+
9 rows in set (0.00 sec)
```

3. Execute a query to export the `owners` table to a text file in CSV format. Put this file in the `D:\labs` directory.
   a. Compare your statement and results to those shown below:

```
mysql> SELECT *
    -> INTO OUTFILE 'D:/labs/owners.txt'
    -> FIELDS TERMINATED BY ','
    -> ENCLOSED BY '"'
    -> LINES TERMINATED BY '\r'
    -> FROM owners;
Query OK, 6 rows affected (0.02 sec)
```

   – The statement creates a text file in the `D:\labs` directory.

   b. Inspect the contents of the text file by using Notepad++ and note the format. Confirm that it contains all the rows and columns from the table:

```
"1","Harold","15554159855"
"2","Gwen","15551234567"
"3","Benny","15553456789"
"4","Diane","15554567890"
"5","Caryn","16163429988"
"6","Olga","18563330000"
```

4. Using the `world_innodb` database, back up the `CountryLanguage` table to a standard text file. Put this file in the `D:\labs` directory.
   a. Compare your statement and results to those shown below:

```
mysql> USE world_innodb
Database changed

mysql> SELECT * INTO OUTFILE 'D:/labs/CountryLanguage.txt'
    -> FROM CountryLanguage;
Query OK, 984 rows affected (0.00 sec)
```

   b. Inspect the contents of the text file by using Notepad++. Confirm that it contains all the rows and columns from the table.

```
ABW  Dutch      T    5.3
ABW  English    F    9.5
ABW  Papiamento F    76.7
ABW  Spanish    F    7.4
AFG  Balochi    F    0.9
AFG  Dari       T    32.1
AFG  Pashto     T    52.4
AFG  Turkmenian F    1.9
AFG  Uzbek      F    8.8
AGO  Ambo       F    2.4
...
ZAF  Zulu       T    22.7
ZMB  Bemba      F    29.7
ZMB  Chewa      F    5.7
ZMB  Lozi       F    6.4
```

```
ZMB  Nsenga     F     4.3
ZMB  Nyanja     F     7.8
ZMB  Tongan     F     11.0
ZWE  English    T     2.2
ZWE  Ndebele    F     16.2
ZWE  Nyanja     F     2.2
ZWE  Shona      F     72.1
```

Practice 11-3: Importing Files from a Data File

Overview

In this practice, you use a `LOAD DATA INFILE` statement to import database data.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

1. In the `Pets` database, create a table like `pet_info` called `pet_info2`. Check that the new table exists in the database.
2. Use a `LOAD DATA INFILE` statement to import the data from the `pet_info.txt` file (from the previous practice) into the new `pet_info2` table.
3. List all the records in the new `pet_info2` table and confirm that it contains all the data.
4. In the `world_innodb` database, create a table like `CountryLanguage` called `CountryLanguage2`.
5. Import the `CountryLanguage.txt` file created in the previous practice into the `CountryLanguage2` table. Confirm that the new table contains the correct number of rows.

Note: Keep your `mysql` session open for the next practice.

Solutions 11-3: Importing Files from a Data File

Tasks

1. In the `Pets` database, create a table like `pet_info` called `pet_info2`.
   a. Compare your statement and results to those shown below:

```
mysql> USE Pets

mysql> CREATE TABLE pet_info2 LIKE pet_info;
Query OK, 0 rows affected (0.08 sec)
```

   b. Check that the new table exists in the database:

```
mysql> SHOW TABLES LIKE 'pet_info%';
+----------------------------+
| Tables_in_pets (pet_info%) |
+----------------------------+
| pet_info                   |
| pet_info2                  |
+----------------------------+
2 rows in set (0.02 sec)
```

2. Use a `LOAD DATA INFILE` statement to import the data from the `pet_info.txt` file (from the previous practice) into the new `pet_info2` table.
   Compare your statement and results to those shown below:

```
mysql> LOAD DATA INFILE 'D:/labs/pet_info.txt'
    -> INTO TABLE pet_info2;
Query OK, 9 rows affected (0.00 sec)
Records: 9  Deleted: 0  Skipped: 0  Warnings: 0
```

3. List all the records in the new `pet_info2` table and confirm that it contains all the data.
   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM pet_info2;
+-----+----------+---------+------------+-------+-----+-----+
| pID | pName    | pGender | pBday      | pDday | oID | tID |
+-----+----------+---------+------------+-------+-----+-----+
|   1 | Fluffy   | F       | 2003-02-04 | NULL  |   1 |   1 |
|   2 | Claws    | M       | 2004-03-17 | NULL  |   2 |   1 |
|   4 | Fang     | M       | 2000-08-27 | NULL  |   3 |   2 |
|   6 | Chirpy   | F       | 2008-09-11 | NULL  |   2 |   3 |
|   7 | Whistler | M       | 2007-12-09 | NULL  |   2 |   4 |
|   8 | Slim     | M       | 2006-04-29 | NULL  |   3 |   8 |
|   9 | Chewy    | F       | NULL       | NULL  |   6 |   6 |
|  10 | Opus     | F       | NULL       | NULL  |   5 |   1 |
|  13 | Scruffy  | M       | 2008-04-17 | NULL  |   2 |   1 |
+-----+----------+---------+------------+-------+-----+-----+
9 rows in set (0.00 sec)
```

4. In the `world_innodb` database, create a table like `CountryLanguage` called `CountryLanguage2`.

   Compare your statement and results to those shown below:

```
mysql> USE world_innodb
Database changed

mysql> CREATE TABLE CountryLanguage2 LIKE CountryLanguage;
Query OK, 0 rows affected (0.03 sec)
```

5. Import the `CountryLanguage.txt` file created in the previous practice into the `CountryLanguage2` table.

   a. Compare your statement and results to those shown below:

```
mysql> LOAD DATA INFILE 'D:/labs/CountryLanguage.txt'
    -> INTO TABLE CountryLanguage2;
Query OK, 984 rows affected (0.08 sec)
Records: 984  Deleted: 0  Skipped: 0  Warnings: 0
```

   b. Confirm that the new table contains the correct number of rows:

```
mysql> SELECT COUNT(*) FROM CountryLanguage2;
+----------+
| COUNT(*) |
+----------+
|      984 |
+----------+
1 row in set (0.06 sec)
```

Note: Keep your `mysql` session open for the next practice.

Practice 11-4: Backing Up Database Files with a Utility

Overview

In this practice, you use the `mysqldump` utility to back up a database.

Duration

This practice takes approximately 25 minutes to complete.

Tasks

Tip: Open two separate command-prompt windows. Work with the shell commands in one window and the MySQL statements in the other. This avoids having to enter and exit the `mysql` client repeatedly.

1. Use `mysqldump` to create a SQL backup of the `world_innodb` database in the `D:\labs` directory. Examine the contents of the backup file.
2. Create another backup file, this time with the `--skip-opt` flag. Compare the two files. What is the difference?
3. Use the `mysql` client to create a new database called `world_innodb2`. Confirm that the new database exists. Exit the `mysql` client.
4. Use one of your new dump files as part of the `mysql` client startup command to import the contents of `world_innodb` to `world_innodb2`. Examine the `world_innodb2` database to confirm.
5. Exit the `mysql` client.
6. Use a MySQL utility to create a backup of just the `Country` table. Ensure that the output file is tab-delimited and written to `d:\labs`. This creates `country.txt` and `country.sql` files: review them both.

Solutions 11-4: Backing Up Database Files with a Utility

Tasks

1. Use `mysqldump` to create a SQL backup of the `world_innodb` database in the `D:\labs` directory.
   a. Execute the following at the command prompt:

```
cmd> mysqldump -uroot -poracle world_innodb >
       D:/labs/world_innodb_backup.sql
```

   b. Examine the contents of the backup file:

```
-- MySQL dump 10.13  Distrib 5.6.10, for Win64 (x86_64)
--
-- Host: localhost    Database: world_innodb
-- ------------------------------------------------------
-- Server version      5.6.10-enterprise-commercial-advanced

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `city`
--

DROP TABLE IF EXISTS `city`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `city` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`),
  CONSTRAINT `city_ibfk_1` FOREIGN KEY (`CountryCode`) REFERENCES
`country` (`Code`)
) ENGINE=InnoDB AUTO_INCREMENT=4080 DEFAULT CHARSET=latin1;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;


--
-- Dumping data for table `city`
--


LOCK TABLES `city` WRITE;
/*!40000 ALTER TABLE `city` DISABLE KEYS */;
INSERT INTO `city` VALUES
(1,'Kabul','AFG','Kabol',1780000),(2,'Qandahar','AFG','Qandahar',23750
0),(3,'Herat','AFG','Herat',186800),(4,'Mazar-e-
Sharif','AFG','Balkh',127800),(5,'Amsterdam','NLD','Noord-
Holland',731200),(6,'Rotterdam','NLD','Zuid-
Holland',593321),(7,'Haag','NLD','Zuid-
Holland',440900),(8,'Utrecht','NLD','Utrecht',234323),(9,'Eindhoven','
NLD','Noord-Brabant',201843),(10,'Tilburg','NLD','Noord-
Brabant',193238),(11,'Groningen','NLD','Groningen',172701),(12,'Breda'
,'NLD','Noord-Brabant',160398),
...
```

2. Create another backup file, this time with the `--skip-opt` flag.
   a. Execute the following at the command prompt:

```
cmd> mysqldump -uroot -poracle --skip-opt world_innodb >
       D:/labs/world_innodb_backup2.sql
```

   b. Compare the two files. What is the difference?

```
-- MySQL dump 10.13  Distrib 5.6.10, for Win64 (x86_64)
--
-- Host: localhost    Database: world_innodb
-- ------------------------------------------------------
-- Server version      5.6.10-enterprise-commercial-advanced
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;


--
-- Table structure for table `city`
--


/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `city` (
  `ID` int(11) NOT NULL,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
```

```
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  KEY `CountryCode` (`CountryCode`),
  CONSTRAINT `city_ibfk_1` FOREIGN KEY (`CountryCode`) REFERENCES
`country` (`Code`)
);
/*!40101 SET character_set_client = @saved_cs_client */;


--
-- Dumping data for table `city`
--


INSERT INTO `city` VALUES (1,'Kabul','AFG','Kabol',1780000);
INSERT INTO `city` VALUES (2,'Qandahar','AFG','Qandahar',237500);
INSERT INTO `city` VALUES (3,'Herat','AFG','Herat',186800);
INSERT INTO `city` VALUES (4,'Mazar-e-Sharif','AFG','Balkh',127800);
INSERT INTO `city` VALUES (5,'Amsterdam','NLD','Noord-
Holland',731200);
INSERT INTO `city` VALUES (6,'Rotterdam','NLD','Zuid-Holland',593321);
INSERT INTO `city` VALUES (7,'Haag','NLD','Zuid-Holland',440900);
INSERT INTO `city` VALUES (8,'Utrecht','NLD','Utrecht',234323);
...
```

– There are no `DROP DATABASE`, `LOCK TABLES`, or `ALTER TABLE` statements. A separate `INSERT` statement populates each row.

3. Use the `mysql` client to create a new database called `world_innodb2`.
   a. Compare your statement and results to those shown below:

```
mysql> CREATE DATABASE world_innodb2;
Query OK, 1 row affected (0.02 sec)
```

   b. Confirm that the new database exists:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| pets               |
| sakila             |
| test               |
| world              |
| world_innodb       |
| world_innodb2      |
+--------------------+
9 rows in set (0.03 sec)
```

c. Exit the `mysql` client. Enter the following at the `mysql>` prompt:

```
mysql> EXIT
```

4. Use one of your new dump files as part of the `mysql` client startup command to import the contents of `world_innodb` to `world_innodb2`.

a. Execute the following at the command prompt:

```
cmd> mysql -uroot -poracle world_innodb2 <
D:/labs/world_innodb_backup2.sql
```

– This can take several minutes to complete.

b. Examine the `world_innodb2` database to confirm.

```
cmd> mysql -uroot -poracle
...
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| pets               |
| sakila             |
| test               |
| world              |
| world_innodb       |
| world_innodb2      |
+--------------------+
9 rows in set (0.05 sec)

mysql> USE world_innodb2
Database changed

mysql> SHOW TABLES;
+-------------------------+
| Tables_in_world_innodb2 |
+-------------------------+
| city                    |
| country                 |
| country2                |
| countrylanguage         |
| countrylanguage2        |
| gelderlanddist          |
+-------------------------+
6 rows in set (0.02 sec)
```

5. Exit the `mysql` client:

6. Use a MySQL utility to create a backup of just the `Country` table. Ensure that the output file is tab-delimited and written to `d:\labs`.
   a. Execute the following at the command prompt:

```
cmd> mysqldump -uroot -poracle --tab=D:/labs
       world_innodb Country
```

   b. Review the `country.txt` file:

```
ABW   Aruba  North America Caribbean     193.00 \N     103000 78.4   828.00 793.00
      Aruba  Nonmetropolitan Territory of The Netherlands   Beatrix       129
      AW
AFG   Afghanistan   Asia   Southern and Central Asia  652090.00     1919
      22720000      45.9   5976.00       \N     Afganistan/Afqanestan
      Islamic Emirate     Mohammad Omar 1      AF
AGO   Angola Africa Central Africa       1246700.00    1975   12878000      38.3
      6648.00       7984.00       Angola Republic     José Eduardo dos Santos
      56     AO
AIA   Anguilla      North America Caribbean     96.00  \N     8000   76.1   63.20
      \N     Anguilla      Dependent Territory of the UK     Elisabeth II  62
      AI
...
BDI   Burundi       Africa Eastern Africa       27834.00      1962   6695000
      46.2   903.00 982.00 Burundi/Uburundi    Republic      Pierre Buyoya 552
      BI
BEL   Belgium       Europe Western Europe       30518.00      1830   10239000
      77.8   249704.00     243948.00     België/Belgique     Constitutional
Monarchy, Federation      Albert II     179    BE
BEN   Benin  Africa Western Africa       112622.00     1960   6097000       50.2
      2357.00       2141.00       Bénin  Republic      Mathieu
...
TON   Tonga  Oceania       Polynesia     650.00 1970   99000  67.9   146.00 170.00
      Tonga  Monarchy      Taufa'ahau Tupou IV 3334   TO
TTO   Trinidad and Tobago North America Caribbean     5130.00       1962
      1295000       68.0  6232.00       5867.00       Trinidad and Tobago
      Republic      Arthur N. R. Robinson       3336   TT
TUN   Tunisia       Africa Northern Africa      163610.00     1956   9586000
      73.7   20026.00      18898.00      Tunis/Tunisie Republic      Zine al-
Abidine Ben Ali     3349   TN
TUR   Turkey Asia   Middle East   774815.00     1923   66591000      71.0
      210721.00     189122.00     Türkiye       Republic      Ahmet Necdet Sezer
      3358   TR
...
YUG   Yugoslavia    Europe Southern Europe      102173.00     1918   10640000
      72.4   17000.00      \N     Jugoslavija   Federal Republic    Vojislav
Koštunica   1792   YU
ZAF   South Africa  Africa Southern Africa      1221037.00    1910   40377000
      51.1   116729.00     129092.00     South Africa  Republic      Thabo Mbeki
      716    ZA
ZMB   Zambia Africa Eastern Africa       752618.00     1964   9169000       37.2
      3377.00       3922.00       Zambia Republic     Frederick Chiluba   3162
      ZM
ZWE   Zimbabwe      Africa Eastern Africa       390757.00     1980   11669000
      37.8   5951.00       8670.00       Zimbabwe      Republic
```

c. Review the `country.sql` file:

```
-- MySQL dump 10.13  Distrib 5.6.10, for Win64 (x86_64)
--
-- Host: localhost    Database: world_innodb
-- ------------------------------------------------------
-- Server version  5.6.10-enterprise-commercial-advanced

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `country`
--

DROP TABLE IF EXISTS `country`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `country` (
  `Code` char(3) NOT NULL DEFAULT '',
  `Name` char(52) NOT NULL DEFAULT '',
  `Continent` enum('Asia','Europe','North
America','Africa','Oceania','Antarctica','South America') NOT NULL DEFAULT
'Asia',
  `Region` char(26) NOT NULL DEFAULT '',
  `SurfaceArea` float(10,2) NOT NULL DEFAULT '0.00',
  `IndepYear` smallint(6) DEFAULT NULL,
  `Population` int(11) NOT NULL DEFAULT '0',
  `LifeExpectancy` float(3,1) DEFAULT NULL,
  `GNP` float(10,2) DEFAULT NULL,
  `GNPOld` float(10,2) DEFAULT NULL,
  `LocalName` char(45) NOT NULL DEFAULT '',
  `GovernmentForm` char(45) NOT NULL DEFAULT '',
  `HeadOfState` char(60) DEFAULT NULL,
  `Capital` int(11) DEFAULT NULL,
  `Code2` char(2) NOT NULL DEFAULT '',
  PRIMARY KEY (`Code`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
...
-- Dump completed on 2013-04-04 13:00:40
```

Practices for Lesson 12: Joining Tables

Chapter 12

Practices for Lesson 12: Overview

Practices Overview

These practices test your knowledge of table joins. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured the MySQL server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.
- You can access MySQL Workbench if you choose to complete the practices by using this tool.

Note: In this practice the first letters of table names are in uppercase. Windows is not case-sensitive but some operating systems are, so it is good practice to use proper capitalization. The SQL statements are all in uppercase for clarity, but this is not required.

Practice 12-1: Performing Inner and Outer Joins

Overview

In this practice, you use the `INNER JOIN`, `LEFT JOIN`, and `RIGHT JOIN` keywords to perform multi-table queries.

Duration

This practice takes approximately 20 minutes to complete.

Tasks

1. Plan and execute a query to display the country and district details for the city of San Antonio:
   a. Issue SQL statements to display the structure of all the tables in the `world_innodb` database. Answer the following questions:
      - Which table contains city and district names?
      - Which table contains country names?
   b. Before constructing your join, execute a query to ensure that there is a city named San Antonio in the `City` table.
   c. Use a `SELECT...INNER JOIN` statement to combine the `Country` and `City` tables. Match records on the `Code` column from the `Country` table and the `CountryCode` column from the `Country` table. Filter the results to retrieve the country and district for the city of San Antonio.

      Hint: Use the proper syntax to specify which table each column in your query belongs to: `<table>.<column>`
2. List all capital cities with the countries they belong to. Use an `INNER JOIN` statement to combine the `City` and `Country` tables. Use the aliases “CapitalName” for cities and “CountryName” for countries.
3. Display the names and capital cities of countries with the codes CHE and ATA. One of these countries has a capital city record associated with it and the other does not, but both countries must appear in the query output. Use appropriate aliases for the columns in the resultset.

   Hint: Use a `SELECT...LEFT JOIN` statement to join the `Country` and `City` tables where the identifier in the `Country` table’s `Capital` column matches the `City` table’s `ID` column. Base the `LEFT JOIN` on the `Country` table to ensure that every country features in the results.
4. Repeat previous step using a `RIGHT JOIN` to base the query on the `City` table and compare the results.

Note: Keep your `mysql` session open for the next practice.

Solutions 12-1: Performing Inner and Outer Joins

Tasks

1. Plan and execute a query to display the country and district details for the city of San Antonio:
   a. Issue SQL statements to display the structure of all the tables in the `world_innodb` database.

```
cmd> mysql -uroot -poracle
...

mysql> USE world_innodb;
Database changed.

mysql> DESC City;
+-------------+----------+------+-----+---------+----------------+
| Field       | Type     | Null | Key | Default | Extra          |
+-------------+----------+------+-----+---------+----------------+
| ID          | int(11)  | NO   | PRI | NULL    | auto_increment |
| Name        | char(35) | NO   |     |         |                |
| CountryCode | char(3)  | NO   | MUL |         |                |
| District    | char(20) | NO   |     |         |                |
| Population  | int(11)  | NO   |     | 0       |                |
+-------------+----------+------+-----+---------+----------------+
5 rows in set (0.02 sec)

mysql> DESC Country;
+----------------+--------------------+------+-----+---------+-------+
| Field          | Type               | Null | Key | Default | Extra |
+----------------+--------------------+------+-----+---------+-------+
| Code           | char(3)            | NO   | PRI |         |       |
| Name           | char(52)           | NO   |     |         |       |
| Continent      | enum('Asia',       |      |     |         |       |
|                | 'Europe',          |      |     |         |       |
|                | 'North America'    |      |     |         |       |
|                | 'Africa'           |      |     |         |       |
|                | 'Oceania',         |      |     |         |       |
|                | 'Antarctica',      |      |     |         |       |
|                | 'South America')   | NO   |     | Asia    |       |
| Region         | char(26)           | NO   |     |         |       |
| SurfaceArea    | float(10,2)        | NO   |     | 0.00    |       |
| IndepYear      | smallint(6)        | YES  |     | NULL    |       |
| Population     | int(11)            | NO   |     | 0       |       |
| LifeExpectancy | float(3,1)         | YES  |     | NULL    |       |
| GNP            | float(10,2)        | YES  |     | NULL    |       |
| GNPOld         | float(10,2)        | YES  |     | NULL    |       |
| LocalName      | char(45)           | NO   |     |         |       |
| GovernmentForm | char(45)           | NO   |     |         |       |
| HeadOfState    | char(60)           | YES  |     | NULL    |       |
| Capital        | int(11)            | YES  |     | NULL    |       |
| Code2          | char(2)            | NO   |     |         |       |
+----------------+--------------------+------+-----+---------+-------+
15 rows in set (0.06 sec)
```

```
mysql> DESC CountryLanguage;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| CountryCode | char(3)       | NO   | PRI |         |       |
| Language    | char(30)      | NO   | PRI |         |       |
| IsOfficial  | enum('T','F') | NO   |     | F       |       |
| Percentage  | float(4,1)    | NO   |     | 0.0     |       |
+-------------+---------------+------+-----+---------+-------+
4 rows in set (0.01 sec)
```

- Which table contains city and district names? **City**
- Which table contains country names? **Country**

b. Before constructing your join, execute a query to ensure that there is a city named San Antonio in the `City` table. Compare your statement and results to those shown below:

```
mysql> SELECT Name FROM City
    -> WHERE Name = 'San Antonio';
+-------------+
| Name        |
+-------------+
| San Antonio |
+-------------+
1 row in set (0.14 sec)
```

c. Use a `SELECT...INNER JOIN` statement to combine the `Country` and `City` tables. Match records on the `Code` column from the `Country` table and the `CountryCode` column from the `Country` table. Filter the results to retrieve the country and district for the city of San Antonio. Compare your statement and results to those shown below:

```
mysql> SELECT Country.Name, City.District
    -> FROM Country
    -> INNER JOIN City
    -> ON CountryCode = Code
    -> WHERE City.Name = 'San Antonio';
+---------------+----------+
| Name          | District |
+---------------+----------+
| United States | Texas    |
+---------------+----------+
1 row in set (0.11 sec)
```

2. List all capital cities with the countries they belong to. Use an `INNER JOIN` statement to combine the `City` and `Country` tables. Use the aliases “CapitalName” for cities and “CountryName” for countries. Compare your statement and results to those shown below:

```
mysql> SELECT City.Name AS CapitalName, Country.Name
    -> AS CountryName
    -> FROM City
    -> INNER JOIN Country
    -> ON City.ID = Country.Capital;
+----------------------------------+--------------------------------------+
| CapitalName                      | CountryName                          |
+----------------------------------+--------------------------------------+
| Oranjestad                       | Aruba                                |
| Kabul                            | Afghanistan                          |
| Luanda                           | Angola                               |
| The Valley                       | Anguilla                             |
| Tirana                           | Albania                              |
| Andorra la Vella                 | Andorra                              |
| Willemstad                       | Netherlands Antilles                 |
| Abu Dhabi                        | United Arab Emirates                 |
| Buenos Aires                     | Argentina                            |
...
| Cittα del Vaticano               | Holy See (Vatican City State)        |
| Kingstown                        | Saint Vincent and the Grenadines     |
| Caracas                          | Venezuela                            |
| Road Town                        | Virgin Islands, British              |
| Charlotte Amalie                 | Virgin Islands, U.S.                 |
| Hanoi                            | Vietnam                              |
| Port-Vila                        | Vanuatu                              |
| Mata-Utu                         | Wallis and Futuna                    |
| Apia                             | Samoa                                |
| Sanaa                            | Yemen                                |
| Beograd                          | Yugoslavia                           |
| Pretoria                         | South Africa                         |
| Lusaka                           | Zambia                               |
| Harare                           | Zimbabwe                             |
+----------------------------------+--------------------------------------+
232 rows in set (0.58 sec)
```

3. Display the names and capital cities of countries with the codes CHE and ATA. One of these countries has a capital city record associated with it and the other does not, but both countries must appear in the query output. Use appropriate aliases for the columns in the resultset. Compare your statement and results to those shown below:

```
mysql> SELECT co.Name AS CountryName, ci.Name AS CityName
    -> FROM Country AS co
    -> LEFT JOIN City AS ci
    -> ON co.Capital = ci.ID
    -> WHERE co.Code IN ('CHE', 'ATA');
+-------------+----------+
| CountryName | CityName |
+-------------+----------+
| Antarctica  | NULL     |
| Switzerland | Bern     |
+-------------+----------+
2 rows in set (0.02 sec)
```

4. Repeat previous step using a `RIGHT JOIN` to base the query on the `City` table and compare the results.

   Compare your statement and results to those shown below:

```
mysql> SELECT co.Name AS CountryName, ci.Name AS CityName
    -> FROM Country AS co
    -> RIGHT JOIN city AS ci
    -> ON co.Capital = ci.ID
    -> WHERE co.Code IN ('CHE', 'ATA');
+-------------+----------+
| CountryName | CityName |
+-------------+----------+
| Switzerland | Bern     |
+-------------+----------+
1 row in set (0.00 sec)
```

   – These results include only records from the `City` table with matching records in the `Country` table.

Note: Keep your `mysql` session open for the next practice.

Practice 12-2: Creating Queries Requiring Joins

Overview

In this practice, you construct the appropriate join query to answer each of the following questions.

Duration

This practice takes approximately 30 minutes to complete.

Tasks

Use the `world_innodb` database to answer the following questions:

1. Which languages are spoken in the country of Sweden? Include the country name and language in the result.
2. Which countries have entries in the `CountryLanguage` table? Include the country name and each of its languages in the output.

   Hint: Use an `INNER JOIN` to ignore records that do not match.
3. Modify the query used in the previous step to include every country, even when there is no corresponding entry in the `CountryLanguage` table. What is the difference in the output?

   Hint: Use the `LEFT JOIN` statement instead.
4. Which country has the largest number of cities?

   Hint: Use the `GROUP BY` and `ORDER BY` clauses with your query to display the correct information.
5. Which countries have at least one city with more than 7 million (7 000 000) inhabitants?

   Hint: Use `DISTINCT` so that each country appears in the output only once.

Use the `Pets` database to answer the following questions:

6. Show the structure of all tables in the `Pets` database.
7. Which owners have female pets, and a phone number starting with 1555?
8. What are the names and types of all pets, and their corresponding owner IDs?
   a. Answer the above query with a `RIGHT JOIN` of the `pet_info` and `pet_types` tables.
   b. Answer the above query with a `LEFT JOIN` of the `pet_info` and `pet_types` tables.
   c. Answer the above query with an `INNER JOIN` of the `pet_info` and `pet_types` tables.
   d. What is the difference in the output?
9. If you are not going to complete the optional practice, exit the `mysql` client.

Solutions 12-2: Creating Queries Requiring Joins

Tasks

Use the `world_innodb` database to answer the following questions:

1. Which languages are spoken in the country of Sweden? Include the country name and language in the result.

   Compare your statement and results to those shown below:

```
mysql> SELECT Name, Language
    -> FROM CountryLanguage, Country
    -> WHERE CountryCode = Code
    -> AND Name = 'Sweden';
+--------+---------------------------+
| Name   | Language                  |
+--------+---------------------------+
| Sweden | Arabic                    |
| Sweden | Finnish                   |
| Sweden | Norwegian                 |
| Sweden | Southern Slavic Languages |
| Sweden | Spanish                   |
| Sweden | Swedish                   |
+--------+---------------------------+
6 rows in set (0.00 sec)
```

2. Which countries have entries in the `CountryLanguage` table? Include the country name and each of its languages in the output.

   Compare your statement and results to those shown below:

```
mysql> SELECT Name, Language
    -> FROM CountryLanguage
    -> INNER JOIN Country
    -> ON CountryCode = Code;
+----------------------------------------+---------------------------+
| Name                                   | Language                  |
+----------------------------------------+---------------------------+
| Aruba                                  | Dutch                     |
| Aruba                                  | English                   |
| Aruba                                  | Papiamento                |
| Aruba                                  | Spanish                   |
| Afghanistan                            | Balochi                   |
...
| Djibouti                               | Somali                    |
| Dominica                               | Creole English            |
| Dominica                               | Creole French             |
| Denmark                                | Arabic                    |
| Denmark                                | Danish                    |
...
| Niger                                  | Tamashek                  |
| Norfolk Island                         | English                   |
| Nigeria                                | Bura                      |
...
| Yugoslavia                             | Slovak                    |
| South Africa                           | Afrikaans                 |
```

```
...                                      |
| Zambia                                 | Tongan                     |
| Zimbabwe                               | English                    |
| Zimbabwe                               | Ndebele                    |
| Zimbabwe                               | Nyanja                     |
| Zimbabwe                               | Shona                      |
+----------------------------------------+----------------------------+
984 rows in set (0.00 sec)
```

3. Modify the query used in the previous step to include every country, even when there is no corresponding entry in the `CountryLanguage` table. What is the difference in the output?
   a. Compare your statement and results to those shown below:

```
mysql> SELECT Name, Language
    -> FROM Country
    -> LEFT JOIN CountryLanguage
    -> ON CountryCode = Code;
+----------------------------------------+----------------------------+
| Name                                   | Language                   |
+----------------------------------------+----------------------------+
| Aruba                                  | Dutch                      |
...
| Antarctica                             | NULL                       |
| French Southern territories            | NULL                       |
...
| South Georgia/South Sandwich Islands   | NULL                       |
| Saint Helena                           | English                    |
...
| Zimbabwe                               | Shona                      |
+----------------------------------------+----------------------------+
990 rows in set (0.00 sec)
```

   b. What is the difference in the output? The number of rows in the result is different. The previous query returned 984 rows and this task returned 990 rows. This is because this result includes countries with no assigned language, indicated by a null value in the `Language` column.

4. Which country has the largest number of cities?

   Compare your statement and results to those shown below:

```
mysql> SELECT Country.Name, COUNT(City.Name) AS Cities
    -> FROM Country
    -> INNER JOIN City
    -> ON City.CountryCode = Country.Code
    -> GROUP BY Country.Name
    -> ORDER BY Cities DESC
    -> LIMIT 1;
```

```
+-------+--------+
| Name  | Cities |
+-------+--------+
| China |    363 |
+-------+--------+
1 row in set (0.01 sec)
```

5. Which countries have at least one city with more than 7 million (7 000 000) inhabitants?
   Compare your statement and results to those shown below:

```
mysql> SELECT DISTINCT Country.Name FROM Country
    -> INNER JOIN City ON Code = CountryCode
    -> WHERE City.Population > 7000000;
+--------------------+
| Name               |
+--------------------+
| Brazil             |
| China              |
| United Kingdom     |
| Indonesia          |
| India              |
| Japan              |
| South Korea        |
| Mexico             |
| Pakistan           |
| Russian Federation |
| Turkey             |
| United States      |
+--------------------+
12 rows in set (0.00 sec)
```

Use the `Pets` database to answer the following questions:

6. Show the structure of all tables in the `Pets` database.
   Compare your statement and results to those shown below:

```
mysql> USE Pets
Database changed

mysql> DESC pet_info;
+---------+---------------+------+-----+---------+----------------+
| Field   | Type          | Null | Key | Default | Extra          |
+---------+---------------+------+-----+---------+----------------+
| pID     | int(11)       | NO   | PRI | NULL    | auto_increment |
| pName   | varchar(20)   | NO   |     | NULL    |                |
| pGender | enum('M','F') | YES  |     | NULL    |                |
| pBday   | date          | YES  |     | NULL    |                |
| pDday   | date          | YES  |     | NULL    |                |
| oID     | int(11)       | NO   |     | NULL    |                |
| tID     | int(11)       | NO   |     | NULL    |                |
+---------+---------------+------+-----+---------+----------------+
7 rows in set (0.05 sec)
```

```
mysql> DESC owners;
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| oID    | int(11)     | NO   | PRI | NULL    | auto_increment |
| oName  | varchar(20) | NO   |     | NULL    |                |
| oPhone | char(11)    | NO   |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
3 rows in set (0.00 sec)

mysql> DESC pet_types;
+-----------+-------------+------+-----+---------+----------------+
| Field     | Type        | Null | Key | Default | Extra          |
+-----------+-------------+------+-----+---------+----------------+
| tID       | int(11)     | NO   | PRI | NULL    | auto_increment |
| pType     | varchar(20) | NO   |     | NULL    |                |
| pCategory | varchar(20) | NO   |     | NULL    |                |
+-----------+-------------+------+-----+---------+----------------+
3 rows in set (0.01 sec)
```

7. Which owners have female pets, and a phone number starting with 1555?
   Compare your statement and results to those shown below:

```
mysql> SELECT owners.oName
    -> FROM owners
    -> JOIN pet_info
    -> ON owners.oID = pet_info.oID
    -> WHERE oPhone LIKE '1555%' AND pGender = 'F';
+--------+
| oName  |
+--------+
| Harold |
| Gwen   |
+--------+
2 rows in set (0.00 sec)
```

8. What are the names and types of all pets, and their corresponding owner IDs?
   a. Answer the above query with a `RIGHT JOIN` of the `pet_info` and `pet_types` tables. Compare your statement and results to those shown below:

```
mysql> SELECT oID, pName, pet_types.pType
    -> FROM pet_info
    -> RIGHT JOIN pet_types
    -> ON pet_info.tID = pet_types.tID;
+------+----------+---------+
| oID  | pName    | pType   |
+------+----------+---------+
|    1 | Fluffy   | Cat     |
|    2 | Claws    | Cat     |
|    5 | Opus     | Cat     |
|    2 | Scruffy  | Cat     |
|    3 | Fang     | Dog     |
|    2 | Chirpy   | Parrot  |
|    2 | Whistler | Canary  |
```

```
| NULL | NULL      | Snake   |
|    6 | Chewy     | Hamster |
| NULL | NULL      | Ferret  |
|    3 | Slim      | Iguana  |
+------+-----------+---------+
11 rows in set (0.00 sec)
```

– The result includes all pets in `pet_info` and two records without a name or owner, indicated by null values in those columns. The right join bases the query on the `pet_types` table so every pet type appears in the query, regardless of whether there is a corresponding entry in the `pet_info` table.

b. Answer the above query with a `LEFT JOIN` of the `pet_info` and `pet_types` tables. Compare your statement and results to those shown below:

```
mysql> SELECT oID, pName, pet_types.pType
    -> FROM pet_info
    -> LEFT JOIN pet_types
    -> ON pet_info.tID = pet_types.tID;
+-----+----------+---------+
| oID | pName    | pType   |
+-----+----------+---------+
|   1 | Fluffy   | Cat     |
|   2 | Claws    | Cat     |
|   3 | Fang     | Dog     |
|   2 | Chirpy   | Parrot  |
|   2 | Whistler | Canary  |
|   3 | Slim     | Iguana  |
|   6 | Chewy    | Hamster |
|   5 | Opus     | Cat     |
|   2 | Scruffy  | Cat     |
+-----+----------+---------+
9 rows in set (0.00 sec)
```

– The result includes an entry for each pet in the `pet_info` table. It does not include the unassigned pet types (Snake and Ferret) from the previous query because the left join bases the query on the `pet_info` table.

c. Answer the above query with an `INNER JOIN` of the `pet_info` and `pet_types` tables. Compare your statement and results to those shown below:

```
mysql> SELECT oID, pName, pet_types.pType
    -> FROM pet_info
    -> INNER JOIN pet_types
    -> ON pet_info.tID = pet_types.tID;
+-----+----------+---------+
| oID | pName    | pType   |
+-----+----------+---------+
|   1 | Fluffy   | Cat     |
|   2 | Claws    | Cat     |
|   3 | Fang     | Dog     |
|   2 | Chirpy   | Parrot  |
|   2 | Whistler | Canary  |
|   3 | Slim     | Iguana  |
|   6 | Chewy    | Hamster |
|   5 | Opus     | Cat     |
|   2 | Scruffy  | Cat     |
+-----+----------+---------+
```

```
9 rows in set (0.00 sec)
```

   - The results are identical to a `LEFT JOIN`, and ignore all non-matching records.

9. If you are not going to complete the optional practice, exit the `mysql` client.

Optional Practice 12-3: Additional Join Practice

Overview

In this practice, you continue to test your knowledge of joins by using your answers from the previous practice.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

Answer the following questions using the `world_innodb` database and your answers from the previous practice:

1. **From Task 1 in 12-2:** Does adding `DISTINCT` make a difference? Why?
2. **From Task 1 in 12-2:** How do you list the languages in reverse alphabetical order?
3. **From Task 2 in 12-2:** How do you limit the results to French-speaking countries?
4. **From Task 3 in 12-2:** How do you limit the results to countries with no matching entries in the `CountryLanguage` table?
5. **From Task 3 in 12-2:** Which table are the nulls from? Why do they appear in the results?
6. **From Task 3 in 12-2:** What join displays all the records in the `CountryLanguage` table that have no matches in the `Country` table?
7. **From Task 5 in 12-2:** What happens if you remove the `DISTINCT` keyword?
8. Exit the `mysql` client.

Solutions 12-3: Additional Join Practice

Tasks

1. From Task 1 in 12-2: Does adding `DISTINCT` make a difference? Why?

   Compare your statement and results to those shown below:

```
mysql> SELECT DISTINCT Name, Language
    -> FROM CountryLanguage, Country
    -> WHERE CountryCode = Code
    -> AND Name = 'Sweden';
+--------+--------------------------+
| Name   | Language                 |
+--------+--------------------------+
| Sweden | Arabic                   |
| Sweden | Finnish                  |
| Sweden | Norwegian                |
| Sweden | Southern Slavic Languages |
| Sweden | Spanish                  |
| Sweden | Swedish                  |
+--------+--------------------------+
6 rows in set (0.00 sec)
```

   – Adding `DISTINCT` does not change the result because all the rows are unique.

2. From Task 1 in 12-2: How do you list the languages in reverse alphabetical order?

   Compare your statement and results to those shown below:

```
mysql> SELECT Name, Language
    -> FROM CountryLanguage, Country
    -> WHERE CountryCode = Code
    -> AND Name = 'Sweden'
    -> ORDER BY Language DESC;
+--------+--------------------------+
| Name   | Language                 |
+--------+--------------------------+
| Sweden | Swedish                  |
| Sweden | Spanish                  |
| Sweden | Southern Slavic Languages |
| Sweden | Norwegian                |
| Sweden | Finnish                  |
| Sweden | Arabic                   |
+--------+--------------------------+
6 rows in set (0.02 sec)
```

3. From Task 2 in 12-2: How do you limit the results to French-speaking countries?

   Compare your statement and results to those shown below:

```
mysql> SELECT Name, Language
    -> FROM CountryLanguage
    -> LEFT JOIN Country
    -> ON CountryCode = Code
    -> WHERE Language = 'French';
+--------------------------+----------+
| Name                     | Language |
```

```
+--------------------------+----------+
| Andorra                  | French   |
| Burundi                  | French   |
| Belgium                  | French   |
| Canada                   | French   |
| Switzerland              | French   |
| France                   | French   |
| Guadeloupe               | French   |
| Haiti                    | French   |
| Italy                    | French   |
| Lebanon                  | French   |
| Luxembourg               | French   |
| Monaco                   | French   |
| Madagascar               | French   |
| Martinique               | French   |
| Mauritius                | French   |
| Mayotte                  | French   |
| New Caledonia            | French   |
| French Polynesia         | French   |
| Rwanda                   | French   |
| Saint Pierre and Miquelon | French   |
| Sao Tome and Principe    | French   |
| Seychelles               | French   |
| United States            | French   |
| Virgin Islands, U.S.     | French   |
| Vanuatu                  | French   |
+--------------------------+----------+
25 rows in set (0.02 sec)
```

4. From Task 3 in 12-2: How do you limit the results to countries with no matching entries in the `CountryLanguage` table?

   Compare your statement and results to those shown below:

```
mysql> SELECT Name, Language
    -> FROM Country
    -> LEFT JOIN CountryLanguage
    -> ON CountryCode = Code
    -> WHERE Language IS NULL;
+-----------------------------------------------+----------+
| Name                                          | Language |
+-----------------------------------------------+----------+
| Antarctica                                    | NULL     |
| French Southern territories                   | NULL     |
| Bouvet Island                                 | NULL     |
| Heard Island and McDonald Islands             | NULL     |
| British Indian Ocean Territory                | NULL     |
| South Georgia and the South Sandwich Islands  | NULL     |
+-----------------------------------------------+----------+
6 rows in set (0.00 sec)
```

5. From Task 3 in 12-2: Which table are the nulls from? Why do they appear in the results?
   - This is a left join, so the results are from the `Country` table.

6. From Task 3 in 12-2: What join displays all the records in the `CountryLanguage` table that have no matches in the `Country` table?

   Compare your statement and results to those shown below:

```
mysql> SELECT Name, Language
    -> FROM Country
    -> RIGHT JOIN CountryLanguage
    -> ON CountryCode = Code
    -> WHERE Language IS NULL;
Empty set (0.00 sec)
```

7. From Task 5 in 12-2: What happens if you remove the `DISTINCT` keyword?

   Compare your statement and results to those shown below:

```
mysql> SELECT Country.Name FROM Country
    -> INNER JOIN City ON Code = CountryCode
    -> WHERE City.Population > 7000000;
+--------------------+
| Name               |
+--------------------+
| Brazil             |
| China              |
| China              |
| United Kingdom     |
| Indonesia          |
| India              |
| India              |
| Japan              |
| South Korea        |
| Mexico             |
| Pakistan           |
| Russian Federation |
| Turkey             |
| United States      |
+--------------------+
14 rows in set (0.00 sec)
```

   – Without `DISTINCT`, two duplicate country rows appear in the results. This is because these countries have more than one city with a population of 7 000 000 or above.

8. Exit the `mysql` client.

Practices for Lesson 13: Table Subqueries

Chapter 13

Practices for Lesson 13: Overview

Practices Overview

These practices test your knowledge of subqueries. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured the MySQL server.
- You have created and populated the `world_innodb` database.
- You can access the `mysql` client from a command-line prompt.
- You can access MySQL Workbench if you choose to complete the practices using this tool.

Note: In this practice, the first letters of table names are in uppercase. Windows is not case-sensitive but some operating systems are, so it is a good practice to use proper capitalization. The SQL statements are all in uppercase for clarity, but this is not required.

Practice 13-1: Performing Different Types of Subqueries

Overview

In this practice, you use subqueries in the `FROM` and `WHERE` clauses of a `SELECT` statement.

Duration

This practice takes approximately 20 minutes to complete.

Tasks

1. Write a query with a non-correlated subquery that retrieves the first three cities in the `City` table with a population larger than New York. List the results in order of population.
   a. The first step is to write the subquery, which must not be dependent on the outer query. The subquery syntax is as follows:

```
SELECT Population
FROM City
WHERE Name = 'New York'
```

   b. Run the subquery. This returns the value the outer query needs in its `WHERE` condition.
   c. Write the outer query to complete the statement.
2. Write a query that retrieves all Nordic countries, with a correlated subquery that calculates the number of cities per country. The outer query syntax is as follows:

```
SELECT Country.Name,
     (<subquery>)
AS CityCount
FROM Country
WHERE Region = 'Nordic Countries'
```

   a. Write the subquery to complete the full statement and execute it.
   b. Is the subquery correlated or non-correlated?
3. Write a query with a non-correlated subquery that retrieves all the languages spoken in Singapore, in descending alphabetical order of `Language`. The subquery syntax is as follows:

```
SELECT Code
FROM Country
WHERE Name='Singapore'
```

   a. Run the subquery. This returns the value the outer query uses in its `WHERE` condition.
   b. Write the outer query to complete the statement.

Note: Keep your `mysql` session open for the next practice.

Solutions 13-1: Performing Different Types of Subqueries

Tasks

1. Write a query with a non-correlated subquery that retrieves the first three cities in the `City` table with a population larger than New York. List the results in order of population.
   a. Log in to the `mysql` client and set `world_innodb` as the default database:

```
cmd> mysql -uroot -poracle
...
mysql> USE world_innodb;
Database changed.
```

   b. Run the subquery. This returns the value the outer query needs in its `WHERE` condition. Compare your statement and results to those shown below:

```
mysql> SELECT Population
    -> FROM City
    -> WHERE Name = 'New York';
+------------+
| Population |
+------------+
| 8008278    |
+------------+
1 row in set (0.02 sec)
```

   – The subquery returns the population of New York City.

   c. Write the outer query to complete the statement. Compare your statement and results to those shown below:

```
mysql> SELECT Name
    -> FROM City
    -> WHERE Population >
    ->     (SELECT Population
    ->      FROM City
    ->      WHERE Name = 'New York')
    -> ORDER BY Population
    -> LIMIT 3;
+------------------+
| Name             |
+------------------+
| Moscow           |
| Ciudad de México |
| Istanbul         |
+------------------+
3 rows in set (0.00 sec)
```

2. Write a query that retrieves all Nordic countries, with a correlated subquery that calculates the number of cities per country. The outer query syntax is as follows:

```
SELECT Country.Name,
     (<subquery>)
AS CityCount
FROM Country
WHERE Region = 'Nordic Countries'
```

   a. Write the subquery to complete the full statement and execute it. Compare your statement and results to those shown below:

```
mysql> SELECT Country.Name,
    ->      (SELECT COUNT(*)
    ->       FROM City
    ->       WHERE CountryCode = Country.Code)
    -> AS CityCount
    -> FROM Country
    -> WHERE Region = 'Nordic Countries';
+------------------------+-----------+
| Name                   | CityCount |
+------------------------+-----------+
| Denmark                |         5 |
| Finland                |         7 |
| Faroe Islands          |         1 |
| Iceland                |         1 |
| Norway                 |         5 |
| Svalbard and Jan Mayen |         1 |
| Sweden                 |        15 |
+------------------------+-----------+
7 rows in set (0.06 sec)
```

   – Returns a list of the seven countries in the Nordic Region, and the number of cities associated with each country

   b. Is the subquery correlated or non-correlated?

   – It is a correlated subquery, because it requires information from the outer query (the `Country` table).

3. Write a query with a non-correlated subquery that retrieves all the languages spoken in Singapore, in descending alphabetical order of `Language`. The subquery syntax is as follows:

```
SELECT Code
FROM Country
WHERE Name='Singapore'
```

a. Run the subquery. This returns the value the outer query needs in its `WHERE` condition. Compare your statement and results to those shown below:

```
mysql> SELECT Code
    -> FROM Country
    -> WHERE Name='Singapore';
+------+
| Code |
+------+
| SGP  |
+------+
1 row in set (0.00 sec)
```

– The subquery returns a scalar value containing the country code for Singapore.

b. Write the outer query to complete the statement. Compare your statement and results to those shown below:

```
mysql> SELECT Language
    -> FROM CountryLanguage
    -> WHERE CountryCode =
    ->     (SELECT Code
    ->      FROM Country
    ->      WHERE Name='Singapore')
    -> ORDER BY Language DESC;
+----------+
| Language |
+----------+
| Tamil    |
| Malay    |
| Chinese  |
+----------+
3 rows in set (0.00 sec)
```

Note: Keep your `mysql` session open for the next practice.

Practice 13-2: Performing Advanced Subqueries

Overview

In this practice, you construct advanced subqueries to answer specific questions.

Duration

This practice takes approximately 45 minutes to complete.

Tasks

1. Which continents have countries where more than 50% of the population speaks English? Does this query use a correlated subquery? Why or why not?

   Hint: Use a subquery in the `WHERE` clause.

2. List the countries in the European continent where the inhabitants speak Spanish. Can you write this query without a subquery?

   Hint: Use a subquery to find the codes for the countries where they speak Spanish, and then compare it to the codes of European countries to filter out all other continents.

   Hint: Use a subquery in the `WHERE` clause. Use the `AND` keyword to filter by both continent and country code values.

3. Which country has the most populous city in the world?

   Hint: Join the `Country` and `City` tables to get the population and city information.

   Hint: Use a subquery to determine the city with the largest population, and then retrieve the associated country name.

4. The following statement uses a correlated subquery to find the South American country with the smallest population:

```
SELECT * FROM Country c1
WHERE Continent = 'South America'
AND Population =
     (SELECT MIN(Population)
      FROM Country c2
      WHERE c2.Continent = c1.Continent)\G
```

   a. Execute the query to display the results.

   b. Run the subquery independently of the main query. What is the result?

5. Rewrite the previous statement (from task 4) using a non-correlated subquery. Explain what makes the new subquery non-correlated. For extra credit, rewrite the full query by using aliases for the table names.

6. Exit the `mysql` client.

Solutions 13-2: Performing Several Advanced Subqueries

Tasks

1. Which continents have countries in which more than 50% of the population speaks English?
   a. First, write and test the subquery. Compare your statement and results to those shown below:

```
mysql> SELECT CountryCode
    -> FROM CountryLanguage
    -> WHERE Language='English'
    -> AND Percentage > 50;
+-------------+
| CountryCode |
+-------------+
| AUS         |
| BLZ         |
| BMU         |
| CAN         |
| GBR         |
| GIB         |
| IRL         |
| NZL         |
| TTO         |
| USA         |
| VIR         |
+-------------+
11 rows in set (0.00 sec)
```

   b. Complete the query. Compare your statement and results to those shown below:

```
mysql> SELECT DISTINCT Continent
    -> FROM Country
    -> WHERE Code IN
    ->     (SELECT CountryCode
    ->      FROM CountryLanguage
    ->      WHERE Language='English'
    ->      AND Percentage>50);
+---------------+
| Continent     |
+---------------+
| Oceania       |
| North America |
| Europe        |
+---------------+
3 rows in set (0.02 sec)
```

   c. Does this query use a correlated subquery? Why or why not?
      – No. The subquery can run alone with no dependency on the outer query.

2. List the countries in the European continent where the inhabitants speak Spanish.
   a. First, write and test the subquery. Compare your statement and results to those shown below:

```
mysql> SELECT CountryCode
    -> FROM CountryLanguage
    -> WHERE Language = 'Spanish';
+-------------+
| CountryCode |
+-------------+
| ABW         |
| AND         |
| ARG         |
| BLZ         |
| BOL         |
| CAN         |
...
| USA         |
| VEN         |
| VIR         |
+-------------+
28 rows in set (0.05 sec)
```

   b. Complete the query. Compare your statement and results to those shown below:

```
mysql> SELECT Name
    -> FROM Country
    -> WHERE Continent = 'Europe'
    -> AND Code IN
    ->     (SELECT CountryCode
    ->      FROM CountryLanguage
    ->      WHERE Language = 'Spanish')
    -> ORDER BY Name;
+---------+
| Name    |
+---------+
| Andorra |
| France  |
| Spain   |
| Sweden  |
+---------+
4 rows in set (0.00 sec)
```

   c. Can you write this query without a subquery?
      – No.

3. Which country has the most populous city in the world?
   a. First, write and test the subquery. Compare your statement and results to those shown below:

```
mysql> SELECT MAX(Population) FROM City;
+-----------------+
| MAX(Population) |
+-----------------+
|        10500000 |
+-----------------+
1 row in set (0.05 sec)
```

   b. Complete the query. Compare your statement and results to those shown below:

```
mysql> SELECT Country.Name
    -> FROM Country JOIN City
    -> ON Country.Code=City.CountryCode
    -> WHERE City.Population =
    ->     (SELECT MAX(Population) FROM City);
+-------+
| Name  |
+-------+
| India |
+-------+
1 row in set (0.03 sec)
```

4. The following statement uses a correlated subquery to find the South American country with the smallest population.
   a. Execute the query to display the results:

```
mysql> SELECT * FROM Country c1
    -> WHERE Continent = 'South America'
    -> AND Population =
    ->     (SELECT MIN(Population)
    ->      FROM Country c2
    ->      WHERE c2.Continent = c1.Continent)\G
*************************** 1. row **************
          Code: FLK
          Name: Falkland Islands
     Continent: South America
        Region: South America
   SurfaceArea: 12173.00
     IndepYear: NULL
    Population: 2000
LifeExpectancy: NULL
           GNP: 0.00
        GNPOld: NULL
     LocalName: Falkland Islands
GovernmentForm: Dependent Territory of the UK
   HeadOfState: Elisabeth II
       Capital: 763
         Code2: FK
1 row in set (0.01 sec)
```

b. Run the subquery independently of the main query. What is the result? Compare your statement and results to those shown below:

```
mysql> SELECT MIN(Population)
    -> FROM Country c2
    -> WHERE c2.Continent = c1.Continent;
ERROR 1054 (42S22): Unknown column 'c1.Continent' in 'where clause'
```

5. Rewrite the previous statement (from task 4) using a non-correlated subquery.

a. First, write and test the subquery. Compare your statement and results to those shown below:

```
mysql> SELECT MIN(Population) FROM Country
    -> WHERE Continent = 'South America';
+-----------------+
| MIN(Population) |
+-----------------+
|            2000 |
+-----------------+
1 row in set (0.00 sec)
```

b. Complete the query. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM Country
    -> WHERE Continent = 'South America'
    -> AND Population =
    ->     (SELECT MIN(Population)
    ->      FROM Country
    ->      WHERE Continent = 'South America')\G
*************************** 1. row ***************************
          Code: FLK
          Name: Falkland Islands
     Continent: South America
        Region: South America
   SurfaceArea: 12173.00
     IndepYear: NULL
    Population: 2000
LifeExpectancy: NULL
           GNP: 0.00
        GNPOld: NULL
     LocalName: Falkland Islands
GovernmentForm: Dependent Territory of the UK
   HeadOfState: Elisabeth II
       Capital: 763
         Code2: FK
1 row in set (0.00 sec)
```

c. Explain what makes the new subquery non-correlated.

– The subquery does not depend on the outer query, and can run independently of it.

d. For extra credit, rewrite the full query by using aliases for the table names. Compare your statement and results to those shown below:

```
mysql> SELECT * FROM Country c1
    -> WHERE Continent = 'South America'
    -> AND Population =
    ->     (SELECT MIN(Population)
    ->     FROM Country c2
    ->     WHERE c2.Continent = c1.Continent)\G
*********************** 1. row ***********************
          Code: FLK
          Name: Falkland Islands
     Continent: South America
        Region: South America
   SurfaceArea: 12173.00
     IndepYear: NULL
    Population: 2000
LifeExpectancy: NULL
           GNP: 0.00
        GNPOld: NULL
     LocalName: Falkland Islands
GovernmentForm: Dependent Territory of the UK
   HeadOfState: Elisabeth II
       Capital: 763
         Code2: FK
1 row in set (0.02 sec)
```

6. Exit the `mysql` client.

Practices for Lesson 14: MySQL Graphical User Interface Tools

Chapter 14

Practices for Lesson 14: Overview

Practices Overview

These practices test your knowledge of MySQL Graphical User Interface tools. They target the Windows operating system provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured the MySQL server.
- You have created and populated the `world_innodb` database.
- You have created and populated the `Pets` database and have completed all the practice steps that target this database, up to and including the Practices for Lesson 9, “Table Data Manipulation.”
- MySQL Workbench 5.2 SE is installed.
- The MySQL Enterprise Monitor demonstration files are available in the `D:\labs\demo\Enterprise_Monitor_demos` directory.
- The MySQL Workbench demonstration files are available in the `D:\labs\demo\Workbench_demos` directory.

Practice 14-1: Creating a Data Model by Using MySQL Workbench

Overview

In this practice, you use the Data Modeling module from Workbench to create an EER diagram for the `world_innodb` database.

Duration

This practice takes approximately 20 minutes to complete.

Tasks

Note: There are no solutions for this practice.

1. Open MySQL Workbench from the MySQL programs group:
   a. Open the Windows Start menu.
   b. Select All Programs.
   c. Select MySQL.
   d. Select MySQL Workbench 5.2 SE.
      – The primary MySQL Workbench window (Workbench Central) opens.
2. Open the Data Modeling module:
   Under the Data Modeling module heading, click the Create New EER Model link.
      – The Model Editor opens in the MySQL Model tab and displays the `mydb` default schema.
3. Delete the default schema:
   a. Click the `mydb` tab in the Physical Schemata panel.
   b. Click the delete button (-) at the far right of the panel header.
   c. The `mydb` schema disappears from the Physical Schemata panel.
4. Add the `world_innodb` database schema to the Physical Schemata panel:
   a. Click the add button (+) on the panel header. A new window called new_schema1 appears at the bottom of the screen.
   b. Enter `world_innodb` in the Name field of the new schema window. The schema name gets updated in the Physical Schemata panel and the new schema window tab header as you type.
   c. Click the close button (X) in the new schema window.
   d. A message box appears, asking if you want to rename all schema occurrences. Click Yes.
   e. You have added the `world_innodb` schema to the Physical Schemata panel.
5. Import the `world_innodb` database data and create an EER diagram:
   a. Select the File > Import > Reverse Engineer MySQL Create Script menu option.
      – The Reverse Engineer MySQL Script dialog box appears.
   b. On the Input and Options page:
      – Click Browse and navigate to the `world_innodb.sql` file in the `D:\labs` directory.
      – Select LATIN1 from the File Encoding list.
      – Select the “Place imported objects on a diagram” check box.

c. Click the Execute button.
   - The Reverse Engineering Progress page displays “Import Finished. Finished parsing MySQL SQL script”.

d. Click Next
   - The Results page displays “SQL Import Finished Successfully”.

e. Click Finish
   - You have created a new EER diagram for the `world_innodb` database.

6. Review the new `world_innodb` EER diagram:
   a. The tables appear in the center of the canvas. Drag them to different parts of the canvas so you can see them all clearly.
   b. Confirm that the tables and their column definitions match those in the SQL input file.
7. Examine the table relationships and foreign keys:
   a. Move the mouse cursor over the relationship line between the `Country` and `CountryLanguage` tables. This highlights the line and the table columns that participate in the relationship.
   b. Double-click the relationship line.
      - The Relationship window opens below the EER diagram.
      - Click the Foreign Key tab at the bottom of the Relationship Window and review the relationship details.
      - Close the Relationship window.
   c. Repeat steps a and b to examine the relationship between the `Country` and `City` tables.
   d. Close the EER diagram tab with the close button (X) to the right of the tab name.
8. View the table information:
   a. Click the MySQL Model tab at the top of the Workbench window.
   b. Double-click the `City` table icon in the `world_innodb` schema. The City table window appears below the Physical Schemata panel.
   c. Click on each of the following tabs at the bottom of the City table window to review the table settings: Columns, Indexes, Foreign Keys, Triggers, Partitioning, Options, Inserts, and Privileges.
   d. Repeat steps a, b, and c for the `Country` and `CountryLanguage` tables.
   e. Close the City table window.
9. Modify the `City` table:
   a. Double-click the `City` table icon in the `world_innodb` schema window.
   b. In the `Columns` tab:
      - Double-click the `Name` column under Column Name.
      - Change the name of the column to `City_Name`.
   c. Close the City table window, which saves the change.
10. Export the `world_innodb` database to a SQL script:
    a. Select the File > Export > Forward Engineer SQL CREATE Script menu option.
       - The Forward Engineer SQL Script dialog box appears.
    b. On the SQL Export Options page:

- Set the Output SQL Script File to `D:\labs\world_innodb_copy.sql`.
- Select the following options:
  - Generate DROP Statement Before Each CREATE Statement
  - Do Not Create Users, Only Export Privileges
  - Generate INSERT Statements For Tables
- Click Next

c. On the SQL Object Export Filter page:
- Select Export MySQL Table Objects
- Click Next

d. On the Review SQL Script page:
- Note that the contents are in the form of a SQL statement file, like the original `world_innodb.sql` file.
- Find the `CREATE TABLE` statement for the `City` table and confirm that your column name change (from `Name` to `City_Name`) is included.
- Click Finish.

e. Open the `D:\labs\world_innodb_copy.sql` file in Notepad++ and examine the content.

11. Save the model:
    a. Select the File > Save Model As menu option.
    b. Save the model as `D:\labs\world_innodb_copy.mwb`.
12. Close the Data Modeling window:
    Click the close button (X) to the right of the MySQL Model tab name.
    - The MySQL Model window closes, leaving the MySQL Workbench window open.

Note: Keep MySQL Workbench open for the next practice.

Practice 14-2: Creating a Server Instance by Using MySQL Workbench

Overview

In this practice, you use the Workbench Server Administration module to create a new server instance for the `world_innodb` database.

Duration

This practice takes approximately 20 minutes to complete.

Tasks

Note: There are no solutions for this practice.

1. Open the Server Administration module:

   From the Workbench Central window, under the Server Administration module heading, click New Server Instance.

   - The Create New Server Instance Profile dialog box appears.

2. Create a new server profile:
   a. On the Specify Host Machine page:
      - Select the localhost radio button
      - Click Next
   b. On the Database Connection page:
      - For Connection Name, enter `MyConnection`
      - For Default Schema, enter `world_innodb`
      - Leave all other options as their defaults
      - Click Next
   c. A pop-up window appears, prompting you for the root user's password. Enter `oracle` and click OK.
   d. The Test DB Connection page indicates that the connection was successful. Click Next.
   e. On the Windows Management page, accept the defaults and click Next.
   f. The Test Settings page reports that "Testing host machine settings is done." Click Next.
   g. In the Review Settings pop-up window, click Continue**.**
   h. On the Complete Setup page:
      - For Server Instance Name, enter `mysql_profile`
      - Click Finish
3. Connect to the new MySQL server instance:
   a. Click the Server Administration link just under the Server Administration module heading.
   b. In the Server Administration dialog box select "mysql_profile" and click OK.
   c. A pop-up window appears, prompting you for the root user's password. Enter `oracle` and click OK.

4. View the server management options and settings for your new server instance, available under the MANAGEMENT navigation menu on the left:
   a. Select Server Status to show the name, details, and “health” status of the server.
   b. Select Startup/Shutdown to show whether the server is running, and to stop or start the server.
   c. Select Status and System Variables to show the current settings of all status and system variables.
   d. Select Server Logs to show the type and location of the log file.
5. View the configuration options and settings for your new server instance, available under the CONFIGURATION navigation menu on the left:

   Select Options File to show a tabbed display of the various mysql configuration options.
6. View the configuration options and settings for your new server instance, available under the DATA EXPORT/RESTORE navigation menu on the left:
   a. Select Data Export to show the various options for exporting data.
   b. Select Data Import/Restore to show the various options for importing data.
7. Close the Server Administration window:

   Click the close button (X) in the Admin tab header.
8. Close MySQL Workbench:

   Select the File > Exit menu option.

Practice 14-3: Viewing MySQL Enterprise Monitor Demonstrations

Overview

In this practice, you view a web-based demonstration that shows the primary features of Enterprise Monitor.

Duration

This practice takes approximately 10 minutes to complete.

Tasks

Note: There are no solutions for this practice.

1. Open the MySQL Enterprise Monitor Dashboard demonstration in your web browser:
   a. Go to the `D:\labs\demo\Enterprise_Monitor_demos` directory in Windows Explorer, and double-click the `memdashboard.html` file.
   b. The demonstration opens in the Mozilla Firefox web browser.
   c. Click the play button in the center of the page to start the demonstration
   d. The demonstration lasts for approximately 3.5 minutes.
2. Open MySQL Enterprise Monitor Query Analyzer demonstration in your web browser:
   a. Go to the `D:\labs\demo\Enterprise_Monitor_demos` directory in Windows Explorer, and double-click the `memqueryanalyzer.html` file.
   b. The demonstration opens in the Mozilla Firefox web browser.
   c. Click the play button in the center of the page to start the demonstration
   d. The demonstration lasts for approximately 5 minutes.

   Note: If you are working outside of the Oracle classroom environment, you can access these demonstrations from the MySQL website at: http://www.mysql.com/products/enterprise/demo.html

Optional Practice 14-4: Viewing the MySQL Workbench Demonstration

Overview

In this practice, you view a web-based demonstration that shows the primary features of MySQL Workbench.

Duration

This practice takes approximately 15 minutes to complete

Tasks

Note: There are no solutions for this practice.

1. Open the MySQL Workbench demonstration in your web browser.
   a. Go to the `D:\labs\demo\Workbench_demos` directory in Windows Explorer, and double-click the `MySQL-Workbench.html` file.
   b. The demonstration opens in the Mozilla Firefox web browser.
   c. Click the play button in the center of the page to start the demonstration
   d. The demonstration lasts for approximately 14 minutes.

   Note: If you are working outside of the Oracle classroom environment, you can access this demonstration on YouTube at:
   http://www.youtube.com/watch?v=X_umYKqKaF0

Optional Practice 14-5: Creating a Model for the `Pets` Database by Using MySQL Workbench

Overview

In this practice, you make a backup of the `Pets` database, and then use the Data Modeling module in Workbench to create an EER diagram for `Pets`.

Duration

This practice takes approximately 15 minutes to complete.

Tasks

Note: There are no solutions for this practice.

1. Create a backup SQL file of the `Pets` database with the `mysqldump` utility, in the `D:\labs` directory.
   a. Execute the following at the command prompt:

```
cmd> mysqldump -uroot -poracle Pets > D:/labs/Pets.sql
```

   b. Open the `D:\labs\Pets.sql` backup file in Notepad++:

```
-- MySQL dump 10.13  Distrib 5.6.10, for Win64 (x86_64)
--
-- Host: localhost    Database: Pets
-- ------------------------------------------------------
-- Server version  5.6.10-enterprise-commercial-advanced
...
--
-- Table structure for table `owners`
--
...
CREATE TABLE `owners` (
  `oID` int(11) NOT NULL AUTO_INCREMENT,
  `oName` varchar(20) NOT NULL,
  `oPhone` char(11) NOT NULL,
  PRIMARY KEY (`oID`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;
...
-- Dumping data for table `owners`
--
...
INSERT INTO `owners` VALUES
(1,'Harold','15554159855'),(2,'Gwen','15551234567'),(3,'Benny','15553456789'),
(4,'Diane','15554567890'),(5,'Caryn','16163429988'),(6,'Olga','18563330000');
...
--
-- Table structure for table `pet_info`
--
...
CREATE TABLE `pet_info` (
  `pID` int(11) NOT NULL AUTO_INCREMENT,
  `pName` varchar(20) NOT NULL,
```

```
  `pGender` enum('M','F') DEFAULT NULL,
  `pBday` date DEFAULT NULL,
  `pDday` date DEFAULT NULL,
  `oID` int(11) NOT NULL,
  `tID` int(11) NOT NULL,
  PRIMARY KEY (`pID`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8;
...
--
-- Dumping data for table `pet_info`
--
...
INSERT INTO `pet_info` VALUES (1,'Fluffy','F','2003-02-
04',NULL,1,1),(2,'Claws','M','2004-03-17',NULL,2,1),(4,'Fang','M','2000-08-
27',NULL,3,2),(6,'Chirpy','F','2008-09-11',NULL,2,3),(7,'Whistler','M','2007-
12-09',NULL,2,4),(8,'Slim','M','2006-04-
29',NULL,3,8),(9,'Chewy','F',NULL,NULL,6,6),(10,'Opus','F',NULL,NULL,5,1),(13,
'Scruffy','M','2008-04-17',NULL,2,1);
...
--
-- Table structure for table `pet_info2`
--
...
CREATE TABLE `pet_info2` (
  `pID` int(11) NOT NULL AUTO_INCREMENT,
  `pName` varchar(20) NOT NULL,
  `pGender` enum('M','F') DEFAULT NULL,
  `pBday` date DEFAULT NULL,
  `pDday` date DEFAULT NULL,
  `oID` int(11) NOT NULL,
  `tID` int(11) NOT NULL,
  PRIMARY KEY (`pID`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8;
...
--
-- Dumping data for table `pet_info2`
--
...
INSERT INTO `pet_info2` VALUES (1,'Fluffy','F','2003-02-
04',NULL,1,1),(2,'Claws','M','2004-03-17',NULL,2,1),(4,'Fang','M','2000-08-
27',NULL,3,2),(6,'Chirpy','F','2008-09-11',NULL,2,3),(7,'Whistler','M','2007-
12-09',NULL,2,4),(8,'Slim','M','2006-04-
29',NULL,3,8),(9,'Chewy','F',NULL,NULL,6,6),(10,'Opus','F',NULL,NULL,5,1),(13,
'Scruffy','M','2008-04-17',NULL,2,1);
...
--
-- Table structure for table `pet_types`
--
...
CREATE TABLE `pet_types` (
  `tID` int(11) NOT NULL AUTO_INCREMENT,
  `pType` varchar(20) NOT NULL,
```

```
  `pCategory` varchar(20) NOT NULL,
  PRIMARY KEY (`tID`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8;
...
--
-- Dumping data for table `pet_types`
--
...
INSERT INTO `pet_types` VALUES
(1,'Cat','Mammal'),(2,'Dog','Mammal'),(3,'Parrot','Bird'),(4,'Canary','Bird'),
(5,'Snake','Reptile'),(6,'Hamster','Mammal'),(7,'Ferret','Mammal'),(8,'Iguana'
,'Reptile');
...
-- Dump completed on 2013-04-16 11:37:51
```

2. Start MySQL Workbench.
3. Open the Data Modeling module. (See the instructions for Practice 14-1.)
4. Add the existing `Pets` database as a schema in the Physical Schemata panel:
   a. Right-click the `mydb` schema and select Delete 'mydb'.
   b. Click the add button (+), and enter the name `Pets` in the new_schema1 window.
   c. Click the close button (x) in the new schema window.
   d. A message box appears, asking if you want to rename all schema occurrences. Click Yes.
   e. You added the `Pets` database to the Physical Schemata panel.
5. Import the `Pets` database data:
   a. Select the File > Import > Reverse Engineer MySQL Create Script menu option.
      – The Reverse Engineer MySQL Script dialog box appears.
   b. On the Input and Options page:
      – Click Browse and navigate to the `Pets.sql` file in the `D:\labs` directory
      – Select UTF8 from the File Encoding list
      – Select the "Place imported objects on a diagram" check box
   c. Click Execute
      – The Reverse Engineering Progress page displays "Import Finished. Finished parsing MySQL SQL script".

d. Click Next
   - The Results page displays “SQL Import Finished Successfully”.
e. Click Finish
   - You have created a new EER diagram for the `Pets` database.

6. Review the new `Pets` EER diagram:
   a. The tables appear in the center of the canvas. Drag them to different parts of the canvas so you can see them all clearly.
   b. Confirm that the tables and their column definitions match those in the SQL input file.
7. Create relationships between the tables:
   a. Add a foreign key from the `pet_info` table to the `owners` table. Select the Place a Relationship Using Existing Columns tool at the very bottom of the vertical toolbar (on the left side of the canvas. You might need to maximize the Workbench window to see this tool.)
      - Click the `oID` column in the `pet_info` table.
      - Click the `oID` column in the `owners` table.

        A relationship line appears on the canvas. This shows that the `owners` table has a one-to-many relationship with the `pet_info` table using the `oID` column.
   b. Move the cursor over the relationship line. This highlights the line and the two related columns.
   c. Double-click the relationship line. The Relationship window opens below the EER diagram.
   d. Click the Foreign Key tab at the bottom of the Relationship Window and review the relationship details.
      - The relationship specifies that one owner ID in the `owners` table can relate to many owner IDs in the `pet_info` table.
   e. Close the Relationship window.
   f. Repeat steps a. to d. to create a relationship between the `pet_info` and `pet_types` tables, using their `tID` columns.
8. View the `Pets` table information:
   a. Click the `MySQL Model` tab.
   b. Double-click the `pet_info` table icon in the `Pets` schema window.
   c. Click the Foreign Keys tab at the bottom of the `pet_info` table window, and ensure that it lists the foreign keys for the two relationships you created.
   d. Close the `pet_info` table window.
9. Save the model:
   a. Select the File > Save Model As menu option.
   b. Save the model as `D:\labs\Pets.mwb`.
10. Close MySQL Workbench.

Practices for Lesson 15: Supplemental Information

Chapter 15

Practices for Lesson 15: Overview

Practices Overview

These practices test your knowledge of views, storage engines, database metadata, and backups. They target the Windows operating system, provided in Oracle classrooms. For non-Oracle classrooms, you might need to adjust file locations.

Assumptions

- You have installed and configured the MySQL server.
- You have created and populated the `world_innodb` database.
- You have created and populated the `Pets` database and have completed all the practice steps that target this database, up to and including the Practices for Lesson 9: “Table Data Manipulation”.
- You can access the `mysql` client from a command-line prompt.

Note: In this practice, the first letters of table names are in uppercase. Windows is not case-sensitive but some operating systems are, so it is good practice to use proper capitalization. The SQL statements are all in uppercase for clarity, but this is not required.

Practice 15-1: Displaying Storage Engine Information

Overview

In this practice, you use the following statements to retrieve information about the use and availability of storage engines in the MySQL server:

- `SHOW CREATE TABLE`
- `SHOW TABLE STATUS`
- `SHOW ENGINES`

Duration

This practice takes approximately 10 minutes to complete.

Tasks

1. Display the statement that creates the `CountryLanguage` table and note which storage engine it uses.
2. Display the status of the `City` table and note which storage engine it uses.
3. Show all the storage engines that your MySQL server supports.

Note: Keep your `mysql` session open for the next practice.

Solutions 15-1: Displaying Storage Engine Information

Tasks

1. Display the statement that creates the `CountryLanguage` table and note which storage engine it uses.
   a. Log in to the MySQL server at the command prompt:

```
cmd> mysql -uroot -poracle
...
```

   b. Compare your statements and results to those shown below:

```
mysql> USE world_innodb
Database changed

mysql> SHOW CREATE TABLE CountryLanguage\G
*************************** 1. row ***************************
       Table: CountryLanguage
Create Table: CREATE TABLE `countrylanguage` (
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `Language` char(30) NOT NULL DEFAULT '',
  `IsOfficial` enum('T','F') NOT NULL DEFAULT 'F',
  `Percentage` float(4,1) NOT NULL DEFAULT '0.0',
  PRIMARY KEY (`CountryCode`,`Language`),
  KEY `CountryCode` (`CountryCode`),
  CONSTRAINT `countryLanguage_ibfk_1` FOREIGN KEY (`CountryCode`)
  REFERENCES `country` (`Code`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1
1 row in set (0.06 sec)
```

2. Display the status of the `City` table and note which storage engine it uses:
   Compare your statement and results to those shown below:

```
mysql> SHOW TABLE STATUS LIKE 'City'\G
*************************** 1. row ***************************
           Name: city
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 4188
 Avg_row_length: 97
    Data_length: 409600
Max_data_length: 0
   Index_length: 131072
      Data_free: 0
 Auto_increment: 4080
    Create_time: 2013-03-18 10:36:00
    Update_time: NULL
     Check_time: NULL
      Collation: latin1_swedish_ci
       Checksum: NULL
 Create_options:
        Comment:
1 row in set (0.08 sec)
```

3. Show all the storage engines that your MySQL server supports.

Compare your statement and results to those shown below:

```
mysql> SHOW ENGINES\G
*************************** 1. row ***************************
      Engine: FEDERATED
     Support: NO
     Comment: Federated MySQL storage engine
Transactions: NULL
          XA: NULL
  Savepoints: NULL
*************************** 2. row ***************************
      Engine: MRG_MYISAM
     Support: YES
     Comment: Collection of identical MyISAM tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 3. row ***************************
      Engine: MyISAM
     Support: YES
     Comment: MyISAM storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 4. row ***************************
      Engine: BLACKHOLE
     Support: YES
     Comment: /dev/null storage engine (anything you write to it disappears)
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 5. row ***************************
      Engine: CSV
     Support: YES
     Comment: CSV storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 6. row ***************************
      Engine: MEMORY
     Support: YES
     Comment: Hash based, stored in memory, useful for temporary tables
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 7. row ***************************
      Engine: ARCHIVE
     Support: YES
     Comment: Archive storage engine
Transactions: NO
          XA: NO
  Savepoints: NO
*************************** 8. row ***************************
      Engine: InnoDB
     Support: DEFAULT
     Comment: Supports transactions, row-level locking, and foreign keys
Transactions: YES
          XA: YES
  Savepoints: YES
*************************** 9. row ***************************
      Engine: PERFORMANCE_SCHEMA
     Support: YES
     Comment: Performance Schema
```

```
Transactions: NO
          XA: NO
  Savepoints: NO
9 rows in set (0.00 sec)
```

- The order of storage engines might vary on your system.

Note: Keep your `mysql` session open for the next practice.

Practice 15-2: Displaying and Creating Views

Overview

In this practice, you determine which tables are base tables and which are views, and create a view from an existing `world_innodb` table.

Duration

This practice takes approximately 10 minutes to complete.

Tasks

1. List the tables in the `world_innodb` database, including the table types.
2. Create a new view called `CityView`, which consists of the `ID` and `Name` columns from the `City` table. Confirm that the new view exists.
3. Display the structure of the `CityView` view.
4. Get a total count of the rows from the `CityView` view to confirm that it contains the correct data.

   Hint: The row count should be the same as for the `City` table.

Note: Keep your `mysql` session open for the next practice.

Solutions 15-2: Displaying and Creating Views

Tasks

1. List the tables in the `world_innodb` database, including the table types.

   Compare your statements and results to those shown below:

```
mysql> USE world_innodb
Database changed

mysql> SHOW FULL TABLES;
+------------------------+------------+
| Tables_in_world_innodb | Table_type |
+------------------------+------------+
| city                   | BASE TABLE |
| country                | BASE TABLE |
| country2               | BASE TABLE |
| countrylanguage        | BASE TABLE |
| countrylanguage2       | BASE TABLE |
| gelderlanddist         | BASE TABLE |
+------------------------+------------+
6 rows in set (0.00 sec)
```

   – Note that the current tables are all of type BASE TABLE.

2. Create a new view called `CityView`, which consists of the `ID` and `Name` columns from the `City` table.

   a. Compare your statement and results to those shown below:

```
mysql> CREATE VIEW CityView AS
    -> SELECT ID, Name FROM City;
Query OK, 0 rows affected (0.05 sec)
```

   b. Confirm that the new view exists:

```
mysql> SHOW FULL TABLES;
+------------------------+------------+
| Tables_in_world_innodb | Table_type |
+------------------------+------------+
| city                   | BASE TABLE |
| cityview               | VIEW       |
| country                | BASE TABLE |
| country2               | BASE TABLE |
| countrylanguage        | BASE TABLE |
| countrylanguage2       | BASE TABLE |
| gelderlanddist         | BASE TABLE |
+------------------------+------------+
7 rows in set (0.00 sec)
```

   – Returns a list of the tables in the `world_innodb` database, which now includes `CityView`

3. Display the structure of the `CityView` view.

   Compare your statement and results to those shown below:

```
mysql> DESCRIBE CityView;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| ID    | int(11)  | NO   |     | 0       |       |
| Name  | char(35) | NO   |     |         |       |
+-------+----------+------+-----+---------+-------+
2 rows in set (0.08 sec)
```

   – Displays the columns and their attributes in the newly created `CityView` view

4. Get a total count of the rows from the `CityView` view to confirm that it contains the correct data.

   Compare your statement and results to those shown below:

```
mysql> SELECT COUNT(*) FROM CityView;
+----------+
| COUNT(*) |
+----------+
|     4079 |
+----------+
```

   – The `CityView` view contains all the records from the `City` table.

   Note: Keep your `mysql` session open for the next practice.

Practice 15-3: Obtaining Metadata

Overview

In this practice, you retrieve metadata from the `INFORMATION_SCHEMA` database.

Duration

This practice takes approximately 20 minutes to complete.

Tasks

1. List all available databases and confirm that `INFORMATION_SCHEMA` is among them.
2. List all the tables in the `INFORMATION_SCHEMA` database.
3. Use the `INFORMATION_SCHEMA` database's `schemata` table to obtain information about the `world_innodb` database.

   Hint: Query the `schemata` table with the schema name set to `world_innodb`.
4. List the table name, type, and engine for all tables in the `world_innodb` database.

   Hint: First, determine which columns you need from the `tables` table in the `INFORMATION_SCHEMA` database. Then, query the `tables` table with the table schema set to `world_innodb`.
5. List the table name, creation time, and the current value for auto-increment table columns for all tables in the `Pets` database.

   Hint: Use the `table_name`, `create_time`, and `auto_increment` columns from the `tables` table in the `INFORMATION_SCHEMA` database, with the table schema set to `Pets`.
6. Exit the `mysql` client.

Solutions 15-3: Obtaining Metadata

Tasks

1. List all available databases and confirm that `INFORMATION_SCHEMA` is among them.

   Compare your statement and results to those shown below:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| pets               |
| sakila             |
| test               |
| world              |
| world_innodb       |
| world_innodb2      |
+--------------------+
9 rows in set (0.00 sec)
```

2. List all the tables in the `INFORMATION_SCHEMA` database.

   Compare your statement and results to those shown below:

```
mysql> SHOW TABLES FROM INFORMATION_SCHEMA;
+---------------------------------------+
| Tables_in_information_schema          |
+---------------------------------------+
| CHARACTER_SETS                        |
| COLLATIONS                            |
| COLLATION_CHARACTER_SET_APPLICABILITY |
| COLUMNS                               |
| COLUMN_PRIVILEGES                     |
| ENGINES                               |
| EVENTS                                |
| FILES                                 |
| GLOBAL_STATUS                         |
| GLOBAL_VARIABLES                      |
| KEY_COLUMN_USAGE                      |
| OPTIMIZER_TRACE                       |
| PARAMETERS                            |
| PARTITIONS                            |
| PLUGINS                               |
| PROCESSLIST                           |
| PROFILING                             |
| REFERENTIAL_CONSTRAINTS               |
| ROUTINES                              |
| SCHEMATA                              |
| SCHEMA_PRIVILEGES                     |
| SESSION_STATUS                        |
| SESSION_VARIABLES                     |
| STATISTICS                            |
| TABLES                                |
| TABLESPACES                           |
| TABLE_CONSTRAINTS                     |
```

```
| TABLE_PRIVILEGES                       |
| TRIGGERS                               |
| USER_PRIVILEGES                        |
| VIEWS                                  |
| . . .                                  |
| INNODB_METRICS                         |
| INNODB_SYS_FOREIGN_COLS                |
| INNODB_FT_CONFIG                       |
| INNODB_BUFFER_POOL_STATS               |
| INNODB_SYS_COLUMNS                     |
| INNODB_SYS_FOREIGN                     |
+----------------------------------------+
60 rows in set (0.00 sec)
```

3. Use the `INFORMATION_SCHEMA` database's `schemata` table to obtain information about the `world_innodb` database.

   Compare your statement and results to those shown below:

```
mysql> SELECT * FROM information_schema.schemata
    -> WHERE SCHEMA_NAME = 'world_innodb'\G
***************** 1. row *******************
              CATALOG_NAME: def
               SCHEMA_NAME: world_innodb
DEFAULT_CHARACTER_SET_NAME: latin1
    DEFAULT_COLLATION_NAME: latin1_swedish_ci
                  SQL_PATH: NULL
1 row in set (0.01 sec)
```

4. List the table name, type, and engine for all tables in the `world_innodb` database.

   a. First, list the available columns in the `INFORMATION_SCHEMA` database's `tables` table. Compare your statement and results to those shown below:

```
mysql> DESC INFORMATION_SCHEMA.TABLES;
+-----------------+---------------------+------+-----+---------+-------+
| Field           | Type                | Null | Key | Default | Extra |
+-----------------+---------------------+------+-----+---------+-------+
| TABLE_CATALOG   | varchar(512)        | NO   |     |         |       |
| TABLE_SCHEMA    | varchar(64)         | NO   |     |         |       |
| TABLE_NAME      | varchar(64)         | NO   |     |         |       |
| TABLE_TYPE      | varchar(64)         | NO   |     |         |       |
| ENGINE          | varchar(64)         | YES  |     | NULL    |       |
| VERSION         | bigint(21) unsigned | YES  |     | NULL    |       |
| ROW_FORMAT      | varchar(10)         | YES  |     | NULL    |       |
| TABLE_ROWS      | bigint(21) unsigned | YES  |     | NULL    |       |
| AVG_ROW_LENGTH  | bigint(21) unsigned | YES  |     | NULL    |       |
| DATA_LENGTH     | bigint(21) unsigned | YES  |     | NULL    |       |
| MAX_DATA_LENGTH | bigint(21) unsigned | YES  |     | NULL    |       |
| INDEX_LENGTH    | bigint(21) unsigned | YES  |     | NULL    |       |
| DATA_FREE       | bigint(21) unsigned | YES  |     | NULL    |       |
| AUTO_INCREMENT  | bigint(21) unsigned | YES  |     | NULL    |       |
| CREATE_TIME     | datetime            | YES  |     | NULL    |       |
| UPDATE_TIME     | datetime            | YES  |     | NULL    |       |
| CHECK_TIME      | datetime            | YES  |     | NULL    |       |
| TABLE_COLLATION | varchar(32)         | YES  |     | NULL    |       |
| CHECKSUM        | bigint(21) unsigned | YES  |     | NULL    |       |
| CREATE_OPTIONS  | varchar(255)        | YES  |     | NULL    |       |
| TABLE_COMMENT   | varchar(2048)       | NO   |     |         |       |
+-----------------+---------------------+------+-----+---------+-------+
21 rows in set (0.17 sec)
```

b. Then, create the query. Compare your statement and results to those shown below:

```
mysql> SELECT table_name, table_type, engine
    -> FROM information_schema.tables
    -> WHERE table_schema = 'world_innodb'
    -> ORDER BY table_name DESC;
+-----------------+------------+--------+
| table_name      | table_type | engine |
+-----------------+------------+--------+
| gelderlanddist  | BASE TABLE | InnoDB |
| countrylanguage2| BASE TABLE | InnoDB |
| countrylanguage | BASE TABLE | InnoDB |
| country2        | BASE TABLE | InnoDB |
| country         | BASE TABLE | InnoDB |
| cityview        | VIEW       | NULL   |
| city            | BASE TABLE | InnoDB |
+-----------------+------------+--------+
7 rows in set (0.00 sec)
```

5. List the table name, creation time, and the current value for auto-increment table columns for all tables in the `Pets` database.

   Compare your statement and results to those shown below:

```
mysql> SELECT table_name, create_time, auto_increment
    ->  FROM information_schema.tables
    ->  WHERE table_schema = 'pets'
    ->  ORDER BY table_name DESC;
+------------+---------------------+----------------+
| table_name | create_time         | auto_increment |
+------------+---------------------+----------------+
| pet_types  | 2011-11-15 10:01:39 |              9 |
| pet_info2  | 2011-12-30 15:11:02 |             14 |
| pet_info   | 2011-11-17 16:56:28 |             14 |
| owners     | 2011-11-15 10:01:38 |              7 |
+------------+---------------------+----------------+
4 rows in set (0.00 sec)
```

   – Creation times might vary on your system.

6. Exit the `mysql` client.

OPTIONAL Practice 15-4: Creating a Backup of MySQL Databases

Overview

In this practice, you use MySQL Enterprise Backup to create a backup of the MySQL databases on your system.

Duration

This practice takes approximately 5 minutes to complete.

Tasks

1. Create a folder called `backups` in the labs directory (`D:\labs`).
2. Use the `mysqlbackup` command to create a backup of all databases on your MySQL server. Connect to the server host on port 3306 as the root account. Include a time stamp and specify `D:\labs\backups` as the destination directory for backup files.
3. Review the contents of the `D:\labs\backups` folder.

OPTIONAL Solutions 15-4: Creating a Backup of MySQL Databases

Tasks

1. Create a folder called `backups` in the labs directory (`D:\labs`).

   Use Windows Explorer to navigate to the `D:\labs` directory and create the specified folder.

2. Use the `mysqlbackup` command to create a backup of all databases on your MySQL server. Connect to the server host on port 3306 as the root account. Include a time stamp and specify `D:\labs\backups` as the destination directory for backup files.

   Compare your command syntax and results to those shown below:

```
cmd> mysqlbackup --user=root --password=oracle --port=3306
          --with-timestamp --backup-dir=D:/labs/backups backup

MySQL Enterprise Backup version 3.8.1 [Mon 01/28/2013 ]
Copyright (c) 2003, 2012, Oracle and/or its affiliates. All Rights Reserved.

 mysqlbackup: INFO: Starting with following command line ...
 mysqlbackup --user=root --password=xxxxxx --port=3306 --with-timestamp
        --backup-dir=d:/labs/backups backup

 mysqlbackup: INFO: MySQL server version is '5.6.10-enterprise-commercial-advanced'.
 mysqlbackup: INFO: Got some server configuration information from running server.

IMPORTANT: Please check that mysqlbackup run completes successfully.
           At the end of a successful 'backup' run mysqlbackup
           prints "mysqlbackup completed OK!".

--------------------------------------------------------------------
                        Server Repository Options:
--------------------------------------------------------------------
  datadir = D:\ProgramData\MySQL\MySQL Server 5.6\data\
  innodb_data_home_dir =
  innodb_data_file_path = ibdata1:12M:autoextend
  innodb_log_group_home_dir = D:\ProgramData\MySQL\MySQL Server 5.6\data\
  innodb_log_files_in_group = 2
  innodb_log_file_size = 50331648
  innodb_page_size = 16384
  innodb_checksum_algorithm = crc32
  innodb_undo_directory = D:\ProgramData\MySQL\MySQL Server 5.6\data\
  innodb_undo_tablespaces = 0
  innodb_undo_logs = 128

--------------------------------------------------------------------
                        Backup Config Options:
--------------------------------------------------------------------
  datadir = d:\labs\backups\2013-04-19_07-29-42\datadir
  innodb_data_home_dir = d:\labs\backups\2013-04-19_07-29-42\datadir
  innodb_data_file_path = ibdata1:12M:autoextend
  innodb_log_group_home_dir = d:\labs\backups\2013-04-19_07-29-42\datadir
  innodb_log_files_in_group = 2
  innodb_log_file_size = 50331648
  innodb_page_size = 16384
  innodb_checksum_algorithm = crc32
  innodb_undo_directory = d:\labs\backups\2013-04-19_07-29-42\datadir
  innodb_undo_tablespaces = 0
  innodb_undo_logs = 128

 mysqlbackup: INFO: Unique generated backup id for this is 13663565820627785

 mysqlbackup: INFO: Creating 14 buffers each of size 16777216.
130419 07:29:44 mysqlbackup: INFO: Full Backup operation starts with following threads
                1 read-threads    6 process-threads    1 write-threads
130419 07:29:44 mysqlbackup: INFO: System tablespace file format is Antelope.
130419 07:29:44 mysqlbackup: INFO: Starting to copy all innodb files...
```

```
130419 07:29:44 mysqlbackup: INFO: Copying D:\ProgramData\MySQL\MySQL Server
5.6\data\ibdata1 (Antelope file format).
130419 07:29:44 mysqlbackup: INFO: Found checkpoint at lsn 8532413.
130419 07:29:44 mysqlbackup: INFO: Starting log scan from lsn 8531968.
130419 07:29:44 mysqlbackup: INFO: Copying log...
130419 07:29:44 mysqlbackup: INFO: Log copied, lsn 8532413.
 mysqlbackup: Progress in MB: 200 400 600 800 1000
. . .
130419 07:30:17 mysqlbackup: INFO: Completing the copy of innodb files.
130419 07:30:17 mysqlbackup: INFO: Preparing to lock tables: Connected to mysqld server.
130419 07:30:17 mysqlbackup: INFO: Starting to lock all the tables...
130419 07:30:18 mysqlbackup: INFO: All tables are locked and flushed to disk
130419 07:30:18 mysqlbackup: INFO: Opening backup source directory
'D:\ProgramData\MySQL\MySQL Server 5.6\data\'
130419 07:30:18 mysqlbackup: INFO: Starting to backup all non-innodb files in
        subdirectories of 'D:\ProgramData\MySQL\MySQL Server 5.6\data\'
130419 07:30:18 mysqlbackup: INFO: Copying the database directory 'mysql'
130419 07:30:18 mysqlbackup: INFO: Copying the database directory 'performance_schema'
130419 07:30:19 mysqlbackup: INFO: Copying the database directory 'sakila'
130419 07:30:19 mysqlbackup: INFO: Copying the database directory 'test'
130419 07:30:19 mysqlbackup: INFO: Copying the database directory 'world'
130419 07:30:19 mysqlbackup: INFO: Completing the copy of all non-innodb files.
130419 07:30:20 mysqlbackup: INFO: A copied database page was modified at 8532413.
           (This is the highest lsn found on page)
           Scanned log up to lsn 8535769.
           Was able to parse the log up to lsn 8535769.
           Maximum page number for a log record 349
130419 07:30:20 mysqlbackup: INFO: All tables unlocked
130419 07:30:20 mysqlbackup: INFO: All MySQL tables were locked for 2.188 seconds.
130419 07:30:20 mysqlbackup: INFO: Full Backup operation completed successfully.
130419 07:30:20 mysqlbackup: INFO: Backup created in directory 'd:\labs\backups\2013-04-
19_07-29-42'

------------------------------------------------------------
   Parameters Summary
------------------------------------------------------------
   Start LSN                  : 8531968
   End LSN                    : 8535769
------------------------------------------------------------

mysqlbackup completed OK!
```

- The exact output on your machine might vary from that shown here.

Note: When using the `--with-timestamp` option, you need to create the backup folder before running the `mysqlbackup` command.

3. Review the contents of the `D:\labs\backups` folder.
   a. Use Windows Explorer to navigate to the `D:\labs\backups` folder.
   b. A new folder exists in this directory with the date and time as the name, for example: `2013-04-19_07-29-42`,

Note: In the Oracle classroom environment, the MySQL server stores the original database data in `D:\ProgramData\MySQL\MySQL Server 5.6\data`. Windows hides the `ProgramData` folder by default. When using Windows Explorer, you must turn on the option to show hidden files and folders (Tools > Folder Options > View tab) to view this folder.

Practices for Lesson 16: Conclusion

Chapter 16

Practices for Lesson 16

Practices Overview

There are no practices for this lesson.

Appendix A: Glossary of MySQL Terms

Chapter 17

Glossary of MySQL Terms: Overview

Overview

This glossary includes explanations of some of the most important terms that you may come across while using MySQL.

-A-

ACID

The acronym for the following features of transactional databases: **Atomic**, **Consistent**, **Isolated**, and **Durable**. Transactional systems are often described as being ACID compliant, where "ACID" stands for the following properties:

- **Atomic:** All the statements execute successfully or are canceled as a unit.
- **Consistent:** A database is in a consistent state when a transaction begins and is left in a consistent state by the transaction.
- **Isolated:** One transaction does not affect another.
- **Durable:** All the changes that are committed are guaranteed to be stored persistently in the database and cannot be lost (even if RDBMS crashes). All the changes that are rolled back are guaranteed not to exist anymore.

-B-

Backup

Generally refers to the "backup" of table data and/or records of database transactions to a location other than system memory and/or the primary data location. *See also* **Dump**.

BIT

A **BIT** is a data type that represents bit-field values. The `BIT` column specifications take a width indicating the number of bits per value, from 1 through 64 bits. A bit is either 1 or 0, on or off. This data type is useful for tracking a collection of attributes.

-C-

Certification, MySQL

The Oracle Certification Program (OCP) validates various levels of MySQL expertise, for MySQL Developers and DBAs with the credentials to prove they have the knowledge, experience, and skills to use and manage MySQL products. A certificate is granted for those that pass an exam covering topics associated with various roles and levels of proficiency with the MySQL server. MySQL currently offers four certificates for Developer and DBA tracks.

Client

A **client** is an application that sends requests to the database server. Most commands are sent via a client, whether it is a command-line client, a PHP script, and so on.

Column

In the context of a relational database table, a column is a set of data values of a particular simple type, one for each row of the table. The columns provide the structure according to which rows are composed.

The term “field” is often used interchangeably with column, although many consider it more correct to use field (or field value) to refer specifically to the single item that exists at the intersection between one row and one column *(per Wikipedia)*.

Connectors

MySQL database **connectors** *(also called software **drivers**)* provide database client connectivity for a wide range of programming languages.

Constraints, Table

A constraint is simply a restriction placed on one or more column values of a table to actively enforce integrity rules. Constraints are implemented by using indexes.

-D-

Database

A database is a loose collection of database objects (such as tables). In MySQL, a database is synonymous with a schema.

Data Type

In a database, each table column value can be one of several different data types (or datatypes), depending on the form of that data. The data might be text, a whole number, a number with decimals, a date, or the time. MySQL data type categories include Numeric, Temporal, and Character String.

DDL

The acronym for **Data Definition Language**; SQL statements that are specifically designed for creation and modification of table data

DML

The acronym for **Data Manipulation Language**; SQL statements that are specifically designed for manipulation of table data

Dump, Database

A database dump contains a record of the table structure and/or the data from a database and is usually in the form of a list of SQL statements. A database dump is most often used for backing up a database so that its contents can be restored in the event of data loss. Corrupted databases can often be recovered by analysis of the dump *(per Wikipedia*. *See also* **Backup***.*

-E-

Engine

See **Storage Engine**.

Enterprise Backup, MySQL

The **MySQL Enterprise Backup** (**MEB**) is a command-line tool that performs “hot backup” operations for MySQL InnoDB databases, and “warm backups” for non-InnoDB storage engine tables.

Enterprise Monitor, MySQL

The **MySQL Enterprise Monitor** (**MEM**) is a web-based GUI tool with a monitoring and advising system, provided only to Enterprise customers. It enables you to monitor multiple databases, monitor replication, and receive notification of failures and/or resource issues.

Entity

In database theory, the term "entity" is used to denote a distinct item (such as a table or column), and the term "relationship" is used to denote that two entities have something to do with each other.

ERD/EER

The acronym for **Entity Relationship Diagram**/**Extended Entity Relationship**. An entity-relationship model is a method of diagramming database objects and the relationships between them.

-F-

File System

A file system (or ***filesystem***) is a method of storing and organizing computer files and the data they contain.

Function

A **function** is a stored operation that transforms a given input into a corresponding output. Functions can be invoked within expressions and return a value that is used in place of the function call when the expression is evaluated.

-G-

GPL

The acronym for **General Public License**; open source license for general use

GUI

The acronym for **Graphical User Interface**. The MySQL GUI tools form a comprehensive graphical user interface to your MySQL database. These easy-to-use graphical tools enable database developers and database administrators (DBAs) to be more productive.

-H-

Help, MySQL client

A user can get help on **`mysql`** client commands or statements by typing the **`mysql`** command with the **`--help`** option at a shell prompt. The **`help;`** or **`\h`** entered at the **`mysql>`** command-line prompt can also be used for help.

Host Name

A **host name** is a label that is assigned to a device connected to a computer network and that is used to identify the device in various forms of electronic communication *(per Wikipedia)*.

With MySQL, it refers to the name of the client or system used to connect to the MySQL server.

-I-

Index

An **index** in MySQL serves to assist in finding table rows more quickly and easily, much like an index of a technical manual. Database indexes are used to locate rows in a table. Instead of containing all of the row data, an index contains only the columns, and a pointer of some sort, used to locate the rows. It also contains information describing where the rows are physically located.

INFORMATION_SCHEMA

INFORMATION_SCHEMA is a virtual database that provides access to database metadata, such as schema, schema objects, and server statistics (such as status variables, settings, and connections).

InnoDB

InnoDB is the default storage engine for MySQL server. It allows ACID-compliant transactions.

-J-

Join

A **join** is an operation that produces a result by combining (joining) information in one table with information in another.

-K-

Keys

In a relational database, a **key** is used to uniquely identify each row in a table. There are different types of keys available for distinct purposes.

-L-

LAMP Stack

The acronym for **Linux**, **Apache**, **MySQL**, **PHP**/**Pearl**/**Python**. This is a combination of hardware and software that represents a solution “stack” of technologies that support application servers. Other derivatives are WAMP (Windows) and SAMP (Solaris).

-M-

Modes, SQL

SQL modes control aspects of server operation such as which SQL syntax MySQL supports and what kind of data validation checks it performs.

-N-

Normalization/Normal Forms

Normalization is the process of refining a database design to ensure that each independent piece of information is in only one place. Normalizing tables removes redundant data. This makes it possible to access data more flexibly, and eliminates the possibility that inappropriate modifications can take place that make the data inconsistent.

Normalization of a complex table often amounts to taking it through a process (using **normal forms**) of decomposition into a set of smaller tables.

NULL

`NULL` is a SQL keyword used to define data types as allowing a missing (or absent) value. It is also a query result.

The concept of NULL can actually have several meanings, such as "no value," "unknown value," "missing value," "out of range," "not applicable," "undefined," and "none of the above."

-O-

Operating System

An **operating system** (OS) [*also called **platform***] is an interface between hardware and users, which is responsible for the management and coordination of activities and the sharing of the resources of a computer, that acts as a host for computing applications run on the machine *(per Wikipedia)*.

-P-

PERFORMANCE_SCHEMA

PERFORMANCE_SCHEMA is a feature for monitoring MySQL server execution at a low level, with minimal impact on server performance.

It inspects the internal execution of the server (at run time) by using the **PERFORMANCE_SCHEMA** storage engine and the `PERFORMANCE_SCHEMA` database.

Permissions

Permissions *(also called **privileges**)* are access levels that are generally assigned to specific users for distinct data and/or task performance.

Planet MySQL

A MySQL-sponsored website for blogs, news, and opinions pertaining to MySQL: http://planet.mysql.com/

-Q-

Query

A **query** is a form of questioning, in a line of inquiry. In MySQL, the SQL language is used to put questions into a form that is understood by the MySQL server and can provide a result based on information stored in a database.

-R-

RDBMS

The acronym for **Relational Database Management System**. An RDBMS organizes and stores data in the form of tables. An RDBMS manages data according to the relational model. In the relational model, the fundamental structure to organize data is the "relation," which is where it gets its name.

Register

A **register** acts as a communication between the different components of the computer hardware. An example of a register is the memory data register that acts as a buffer between the computer memory and the central processing unit. However, there are multiple registers that perform communication functions within the computer hardware to handle the passing of information from one component to another. Most registers are multi-directional and can read and write in any direction (for example, from CPU to memory or from memory to CPU).

Row

A **row** *(also called a **record** or **tuple**)* represents a single, implicitly structured data item in a table. In simple terms, a database table can be thought of as consisting of rows and columns or fields. Each row in a table represents a set of related data, and every row in the table has the same structure *(per Wikipedia)*.

The rows of a table consist of a collection of values that describe an entity (for example, bank account).

-S-

SELECT statement

The **`SELECT`** SQL statement returns a result set of records from one or more tables.

Server, Database

A **database server** is a computer program that provides database services to other computer programs or computers, as defined by the client/server model. The term may also refer to a computer dedicated to running such a program.

Database management systems frequently provide database server functionality, and some DBMSs (for example, MySQL) rely exclusively on the client/server model for database access *(per Wikipedia)*.

SQL

The acronym for Structured Query Language. SQL is a database computer language designed for managing data in relational database management systems (RDBMS), and originally based on Relational Algebra. Its scope includes data query and update, schema creation and modification, and data access control *(per Wikipedia)*.

Storage Engine

A **storage engine** *(also called a **database engine**)* is the underlying software component that a database management system uses to create, retrieve, update, and delete data from a database.

Subquery
A **subquery** is a query nested within another SQL statement (query).

-T-

Tee file
A **tee file** is created by using the **`tee`** command, to record the input and output of SQL statements for a specified **`mysql`** session.

Transaction, Database
A **database transaction** comprises a unit of work performed against a database, and treated in a coherent and reliable way independent of other transactions. A transaction is a means to execute one or more SQL statements as a single unit of work, such that either all or none of the statements succeed.

Table, Database
A **database table** is a set of data elements (values) that are organized using a model of vertical columns (which are identified by their names) and horizontal rows. A table has a specified number of columns, but can have any number of rows *(per Wikipedia)*.

-U-

UNIX
UNIX is a multitasking, multi-user computer operating system.

-V-

View
A **view** *(also called **virtual tables**)* is a database object that is defined in terms of a `SELECT` statement that retrieves the data you want the view to produce.

-W-

Workbench, MySQL
MySQL Workbench is a cross-platform GUI tool for the MySQL server, for database architects, developers, and DBAs. MySQL Workbench provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration, and much more. MySQL Workbench is available on Windows, Linux, and Mac OS.

-XYZ-
--

Appendix B: Practice Solution Scripts

Chapter 18

Lesson 2: MySQL Server and Client

Practice 2-1: Install and Start the MySQL Server

```
-- Step (1)
-- Using Windows Explorer go to D:\stage\MySQL

-- Step (2)
-- Using Windows Explorer
-- Double click on the "mysql-installer-commercial-5.6.10.0.msi"
file.

-- Steps (3-19)
-- Follow MySQL Installer prompts

-- Step (20)
-- At command prompt:
mysql

-- Step (21)
-- Add mysql client program to PATH environment variable:
;D:\Program Files\MySQL\MySQL Server 5.6\bin;

-- Step (22)
-- At command prompt:
mysql -u root -poracle

-- Step (23)
-- At mysql prompt:
EXIT
```

Practice 2-2: Use the Keyboard Editing and Tee Commands

```
-- Step (3)
-- At command prompt:
mysql -u root -poracle

-- Step (4)
-- At mysql prompt:
\h

-- Step (6)
tee D:\labs\Lesson2_tee.txt

-- Step (11)
tee D:\labs\Test.txt

-- Step (12)
notee

-- Step (13)
tee D:\labs\Lesson2_tee.txt
```

```
-- Step (14)
notee \c

-- Step (15)
-- Using Windows Explorer go to D:\labs
-- Double click on the Lesson2_tee.txt file
```

Practice 2-3: Install the world_innodb Database

```
-- Step (1)
-- At mysql prompt:
CREATE DATABASE world_innodb CHARACTER SET latin1;

-- Step (2)
USE world_innodb

-- Step (3)
SOURCE D:\labs\world_innodb.sql

-- Step (4)
EXIT
```

Lesson 3: Database Basics

No scripts for this lesson

Lesson 4: Database Design

Practice 4-1: Quiz

No scripts for this practice

Practice 4-2: Evaluate a Database

```
-- Step (1)
-- At command prompt:
mysql -u root -poracle

-- Step (2)
-- At mysql prompt:
SHOW DATABASES;

-- Step (3)
USE world_innodb

-- Step (4)
SHOW TABLES;

-- Step (5)
DESCRIBE City;

-- Step (6)
SELECT * FROM City;

-- Step (7)
DESCRIBE Country;

-- Step (8)
SELECT * FROM Country\G

-- Step (9)
DESCRIBE CountryLanguage;

-- Step (10)
SELECT * FROM CountryLanguage;

-- Step (11)
EXIT
```

Practice 4-3: Create a Structure Diagram

No scripts for this practice

Lesson 5: Data Types

No scripts for this lesson

Lesson 6: Database and Table Creation

Practice 6-1: Display Table Creation Information

```
-- Step (1)
-- At command prompt:
mysql -u root -poracle

-- At mysql prompt:
USE world_innodb

-- Step (2)
SHOW CREATE TABLE Country\G

-- Step (3)
CREATE TABLE `Country2` (
`Code` char(3) NOT NULL DEFAULT '',
`Name` char(52) NOT NULL DEFAULT '',
`Continent` enum('Asia','Europe','North America','Africa',
'Oceania','Antarctica','South America')
NOT NULL DEFAULT 'Asia',
`Region` char(26) NOT NULL DEFAULT '',
`SurfaceArea` float(10,2) NOT NULL DEFAULT '0.00',
`IndepYear` smallint(6) DEFAULT NULL,
`Population` int(11) NOT NULL DEFAULT '0',
`LifeExpectancy` float(3,1) DEFAULT NULL,
`GNP` float(10,2) DEFAULT NULL,
`GNPOld` float(10,2) DEFAULT NULL,
`LocalName` char(45) NOT NULL DEFAULT '',
`GovernmentForm` char(45) NOT NULL DEFAULT '',
`HeadOfState` char(60) DEFAULT NULL,
`Capital` int(11) DEFAULT NULL,
`Code2` char(2) NOT NULL DEFAULT '',
PRIMARY KEY (`Code`)
);

-- Step (4)
SHOW TABLES;

-- Step (5)
SHOW INDEX FROM Country2\G

-- Step (6)
SHOW INDEX FROM City\G

-- Step (7)
EXIT
```

Practice 6-2: Create a Database

```
-- Step (1-4)
-- no scripts
```

```
-- Step (5)
-- At command prompt:
mysql -u root -poracle

-- At mysql prompt:
CREATE DATABASE Pets;

-- Step (6)
SHOW DATABASES;

-- Step (7)
USE Pets

-- Step (8)
CREATE TABLE pet_info (
    pID INT NOT NULL AUTO_INCREMENT,
    pName VARCHAR(20) NOT NULL,
    pGender ENUM('M', 'F') DEFAULT NULL,
    pBday DATE DEFAULT NULL,
    pDday DATE DEFAULT NULL,
    oID INT NOT NULL,
    tID INT NOT NULL,
    PRIMARY KEY (pID)
    );

CREATE TABLE owners (
    oID INT NOT NULL AUTO_INCREMENT,
    oName VARCHAR(20) NOT NULL,
    oPhone CHAR(11) NOT NULL,
    PRIMARY KEY (oID)
    );

CREATE TABLE pet_types (
    tID INT NOT NULL AUTO_INCREMENT,
    pType VARCHAR(20) NOT NULL,
    pCategory VARCHAR(20) NOT NULL,
    PRIMARY KEY (tID)
    );

-- Step (9)
SHOW TABLES;

-- Step (10)
DESC pet_info;

DESC owners;

DESC pet_types;

-- Step (11)
EXIT
```

Lesson 7: Basic Queries

Practice 7-1: Perform Basic Queries

```
-- Step (1)
-- At command prompt:
mysql -u root -poracle

-- At mysql prompt:
USE world_innodb

-- Step (2)
DESC Country;

-- Step (3)
SELECT Continent FROM Country;

-- Step (4)
SELECT Continent, Name FROM Country;

-- Step (5)
SELECT Region FROM Country;

-- Step (6)
SELECT DISTINCT Region FROM Country;

-- Step (7)
SELECT * FROM City WHERE ID = 3875;

-- Step (8)
SELECT Name, Population FROM Country WHERE Population < 1000;

-- Step (9)
SELECT Name FROM City ORDER BY Name DESC;

-- Step (10)
DESC CountryLanguage;

-- Step (11)
SELECT CountryCode, Language
FROM CountryLanguage
WHERE Language = 'Swedish'
ORDER BY CountryCode DESC;

-- Step (12)
SELECT Name
FROM City
ORDER BY Name ASC
LIMIT 10;

-- Step (13)
SELECT CountryCode, Language
    FROM CountryLanguage
```

```
    WHERE Language = 'Chinese'
    ORDER BY CountryCode DESC
    LIMIT 2;

-- Step (14)
SELECT *
    FROM Country
    WHERE GNP > GNPOld
    ORDER BY Name
    LIMIT 3\G

-- Step (15)
EXIT
```

Practice 7-2: Perform Basic Queries Using MySQL Workbench

No scripts for this practice

Practice 7-3: Perform Basic Queries on the Pets Database

```
-- Step (1)
-- At command prompt:
mysql -u root -poracle

-- From mysql client prompt:
USE Pets

-- Step (2)
DESC pet_info;

-- Step (3)
INSERT INTO pet_info (pName, pGender, pBday, pDday, oID, tID)
    VALUES ('Fluffy', 'F', '2003-02-04', NULL, 1, 1),
    ('Claws', 'M', '2004-03-17', NULL, 2, 1),
    ('Buffy', 'F', '1999-05-13', NULL, 1, 2);

-- Step (4)
SHOW TABLES;

-- Step (5)
SELECT * FROM pet_info;

-- Step (6)
SELECT * FROM pet_info LIMIT 1;

-- Step (7)
SELECT oID FROM pet_info WHERE pName = 'Fluffy';

-- Step (8)
SELECT pName FROM pet_info
WHERE tID = 1 AND pBday > '2003-01-01';

-- Step (9)
SELECT DISTINCT pGender FROM pet_info;
```

```
-- Step (10)
SELECT pName, tID FROM pet_info WHERE tID != 1;

-- Step (11)
SELECT pID, pName FROM pet_info
WHERE pName IN ('Claws', 'Buffy');

-- Step (12)
SELECT pID, pName from pet_info
WHERE oID = 1
AND (tID = 2 OR tID=3);

-- Step (13)
SELECT pName, pBday FROM pet_info
ORDER BY pBday ASC;

-- Step (14)
SELECT pID, pName, pBday FROM pet_info
ORDER BY pBday DESC;

-- Step (15)
EXIT
```

Lesson 8: Database and Table Maintenance

Practice 8-1: Remove a Database

```
-- Step (1)
-- At command prompt:
mysql -uroot -poracle

-- At mysql prompt:
CREATE DATABASE db1;

-- Step (2)
SHOW DATABASES;

-- Step (3)
DROP DATABASE db1;

-- Step (4)
SHOW DATABASES;
```

Practice 8-2: Create a New Table and Remove a Table

```
-- Step (1)
USE world_innodb

SHOW CREATE TABLE City\G

-- Step (2)
CREATE TABLE GelderlandDist AS
SELECT Name, District, CountryCode
FROM City
WHERE District = 'Gelderland';

-- Step (3)
SHOW TABLES;

-- Step (4)
SELECT * FROM GelderlandDist;

-- Step (5)
CREATE TABLE GelderlandDist2 LIKE GelderlandDist;

DESC GelderlandDist2;

-- Step (6)
SHOW TABLES;

-- Step (7)
DROP TABLE IF EXISTS GelderlandDist2;

-- Step (8)
SHOW TABLES;
```

Practice 8-3: Alter Table Columns

```
-- Step (1)
DESCRIBE GelderlandDist;

-- Step (2)
ALTER TABLE GelderlandDist MODIFY Name char(20);

-- Step (3)
DESCRIBE GelderlandDist;

-- Step (4)
ALTER TABLE GelderlandDist ADD Inauguration DATE NOT NULL;

-- Step (5)
DESCRIBE GelderlandDist;

-- Step (6)
SELECT * FROM GelderlandDist;
```

Practice 8-4: Modify Table Indexes and Constraints

```
-- Step (1)
SHOW CREATE TABLE City\G

-- Step (2)
ALTER TABLE City ADD INDEX CityName(Name);

-- Step (3)
SHOW CREATE TABLE City\G

-- Step (4)
ALTER TABLE City DROP INDEX CityName;

-- Step (5)
SHOW CREATE TABLE City\G

-- Step (6)
DESCRIBE GelderlandDist;

-- Step (7)
ALTER TABLE GelderlandDist ADD PRIMARY KEY(Name);

-- Step (8)
DESC GelderlandDist;
```

Practice 8-5: Further Practice

```
-- Step (1)
SHOW CREATE TABLE City\G;

-- Step (2)
CREATE TABLE Big_Cities
SELECT id, name, population from City
```

```
WHERE population > 8000000;

SHOW TABLES;

-- Step (3)
DESC Big_Cities;

-- Step (4)
SELECT * FROM Big_Cities;

-- Step (5)
ALTER TABLE Big_Cities ADD COLUMN Founded DATE NULL;

DESC Big_Cities;

-- Step (6)
SELECT * FROM Big_Cities;

-- Step (7)
ALTER TABLE Big_Cities DROP Founded;

DESC Big_Cities;

-- Step (8)
ALTER TABLE Big_Cities MODIFY ID INT(11) NULL;

DESC Big_Cities;

-- Step (9)
ALTER TABLE Big_Cities ADD PRIMARY KEY (ID);

DESC Big_Cities;

-- Step (10)
ALTER TABLE Big_Cities ADD INDEX Pop (Population);

DESC Big_Cities;

SHOW CREATE TABLE Big_Cities\G

-- Step (11)
ALTER TABLE Big_Cities DROP INDEX Pop;

DESC Big_Cities;

SHOW CREATE TABLE Big_Cities\G

-- Step (12)
DROP TABLE Big_Cities;

SHOW TABLES;
```

```
-- Step (13)
EXIT
```

Lesson 9: Table Data Manipulation

Practice 9-1: Insert and Replace Table Row Data

```
-- Step (1)
-- At command prompt:
mysql -uroot -poracle

-- At mysql prompt:
USE world_innodb

INSERT INTO GelderlandDist (Name, District, CountryCode,
Inauguration)
VALUES ('Sakila', 'Gelderland', 'SQL', '2001-07-01');

-- Step (2)
SELECT * FROM GelderlandDist;

-- Step (3)
INSERT INTO GelderlandDist (Name, District, CountryCode,
Inauguration)
VALUES ('MySQLland', 'Gelderland', 'MYS', '1984-08-04'),
('Fantasia', 'Gelderland', 'FNT', '1950-01-01');

-- Step (4)
SELECT * FROM GelderlandDist;

-- Step (5)
REPLACE INTO GelderlandDist (Name, District, CountryCode,
Inauguration)
VALUES ('MySQLland', 'Gelderland', 'SQL', '1984-08-04');

-- Step (6)
SELECT * FROM GelderlandDist;
```

Practice 9-2: Modify and Delete Table Row Data

```
-- Step (1)
UPDATE GelderlandDist
SET Inauguration = '1880-05-17'
WHERE Name = 'Ede';

-- Step (2)
SELECT * FROM GelderlandDist;

-- Step (3)
UPDATE GelderlandDist
SET CountryCode = 'FOO'
WHERE CountryCode = 'NLD'
ORDER BY Name
LIMIT 2;

-- Step (4)
```

```
SELECT * FROM GelderlandDist;

-- Step (5)
DELETE FROM GelderlandDist
WHERE CountryCode = 'FOO'
LIMIT 1;

-- Step (6)
SELECT * FROM GelderlandDist;

-- Step (7)
SELECT * FROM City WHERE CountryCode = 'FOO';

-- Step (8)
DELETE FROM GelderlandDist
WHERE CountryCode = 'FOO'
LIMIT 1;
```

Practice 9-3: Manipulate Table Row Data in the Pets Database

```
-- Step (1)
USE Pets

-- Step (2)
SELECT * FROM pet_info;

-- Step (3)
INSERT INTO pet_info (pName, pGender, pBday, pDday, oID, tID)
VALUES
('Fang','M','2000-08-27',NULL,3,2),
('Bowser','M','1989-08-31','2009-07-29',4,2),
('Chirpy','F','2008-09-11',NULL,2,3),
('Whistler',NULL,'2007-12-09',NULL,2,4),
('Slim','M','2006-04-29',NULL,3,5),
('Puffball','F','2009-03-30',NULL,4,1),
('Opus','F',NULL,NULL,5,1),
('Rocky','M','1998-04-04','2013-02-11',6,1),
('Koko','M','1997-02-08',NULL,3,1),
('Scruffy','M','2008-04-17',NULL,2,1);

SELECT * FROM pet_info;

-- Step (4)
INSERT INTO owners (oName, oPhone) VALUES
('Harold', '15554159855'),
('Gwen', '15551234567'),
('Benny', '15553456789'),
('Diane', '15554567890'),
('Caryn', '15554444444'),
('Chris', '15556666666');

SELECT * FROM owners;
```

```
-- Step (5)
INSERT INTO pet_types (pType, pCategory) VALUES
('Cat', 'Mammal'),
('Dog', 'Mammal'),
('Parrot', 'Bird'),
('Canary', 'Bird'),
('Snake', 'Reptile'),
('Hamster', 'Mammal'),
('Ferret', 'Mammal');

SELECT * FROM pet_types;

-- Step (6)

UPDATE pet_info
SET pGender = 'M'
WHERE pName = 'Whistler';

SELECT * FROM pet_info;

-- Step (7)
INSERT INTO pet_types (pType, pCategory)
VALUES ('Iguana', 'Reptile');

SELECT * FROM pet_types;

UPDATE pet_info
SET tID = 8
WHERE pName = 'Slim';

SELECT * FROM pet_info;

-- Step (8)
SELECT * FROM owners;

UPDATE pet_info SET oID = 3
WHERE oID = 1 AND pName = 'Buffy';

SELECT * FROM pet_info;

-- Step (9)
UPDATE owners
SET oPhone = '16163429988'
WHERE oName IN ('Caryn', 'Chris');

SELECT * FROM owners;

-- Step (10)
DELETE FROM pet_info
WHERE pBday < '2000-01-01';

SELECT * FROM pet_info;
```

```
-- Step (11)
SELECT DISTINCT oID FROM pet_info;

SELECT * FROM owners;

DELETE FROM owners WHERE oID = 6;

SELECT * FROM owners;

-- Step (12)
REPLACE INTO pet_info (pID, pName, pGender, pBday, pDday, oID,
tID)
VALUES (9,'Chewy','F', NULL, NULL,6,6);

SELECT * FROM pet_info;

-- Step (13)
INSERT INTO owners (oID, oName, oPhone)
VALUES (6,'Olga','18563330000');

SELECT * FROM owners;

-- Step (14)
EXIT
```

Lesson 10: Functions

Practice 10-1: Quiz

No scripts for this practice

Practice 10-2: Use Simple, String, and Temporal Functions

```
-- Step (1)
-- At command prompt:
mysql -uroot -poracle

-- At mysql prompt:
SELECT VERSION();

-- Step (2)
SELECT strcmp('awake','asleep'),
strcmp('awake','awake'),
strcmp('asleep','awake');

-- Step (3)
SELECT CONCAT('I ','am ','mostly ', 'awake!');

-- Step (4)
SELECT SUBSTRING('HarryMonkey', 6);

-- Step (5)
-- no script

-- Step (6)
SELECT DATE_FORMAT(NOW(), '%W the %D of %M in the year %Y');

-- Step (7)
SELECT DAYNAME(NOW() + INTERVAL 500 DAY);
```

Practice 10-3: Use Numeric and Control Flow Functions

```
-- Step (1)
SELECT FLOOR(-8.6), FLOOR(8.6);

-- Step (2)
USE world_innodb

SELECT IndepYear, Name,
CASE
WHEN IndepYear < 1300 then 'Ancient'
WHEN IndepYear < 1800 then 'Really Old'
WHEN IndepYear < 1900 then 'Not Old'
WHEN IndepYear < 2000 then 'New'
ELSE 'Brand New'
END
FROM Country
ORDER BY IndepYear DESC;
```

Practice 10-4: Use Aggregate Functions

```
-- Step (1)
USE world_innodb

SELECT Continent, SUM(Population)
FROM Country
GROUP BY Continent;

-- Step (2)
SELECT Continent, ROUND(AVG(LifeExpectancy))
FROM Country
GROUP BY Continent;

-- Step (3)
SELECT CountryCode, AVG(Population) AS AvgPop
FROM City
GROUP BY CountryCode
HAVING AVG(Population) > 500000;

-- Step (4)
SELECT GovernmentForm, COUNT(GovernmentForm) AS Governments
FROM Country
GROUP BY GovernmentForm
ORDER BY Governments DESC
LIMIT 5;

-- Step (5)
SELECT Continent, AVG(SurfaceArea) AS AverageSurfaceArea
FROM Country
GROUP BY Continent;

-- Step (6)
SELECT Continent, AVG(SurfaceArea) AS AverageSurfaceArea
FROM Country
GROUP BY Continent
WITH ROLLUP;

-- Step (7)
EXIT
```

Lesson 11: Exporting and Importing Data

Practice 11-1: Quiz

No scripts for this practice

Practice 11-2: Export Files Using a Query

```
-- Step (1)
-- At command prompt:
mysql -uroot -poracle

-- At mysql prompt:
USE Pets

-- Step (2)
SELECT *
INTO OUTFILE 'D:/labs/pet_info.txt'
FROM pet_info;

SELECT * FROM pet_info;

-- Step (3)
SELECT *
INTO OUTFILE 'D:/labs/owners.txt'
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\r'
FROM owners;

-- Step (4)
USE world_innodb

SELECT * INTO OUTFILE 'D:/labs/CountryLanguage.txt'
FROM CountryLanguage;
```

Practice 11-3: Import Files from a Data File

```
-- Step (1)
USE Pets

CREATE TABLE pet_info2 LIKE pet_info;

SHOW TABLES LIKE 'pet_info%';

-- Step (2)
LOAD DATA INFILE 'D:/labs/pet_info.txt'
INTO TABLE pet_info2;

-- Step (3)
SELECT * FROM pet_info2;

-- Step (4)
USE world_innodb
```

```
CREATE TABLE CountryLanguage2 LIKE CountryLanguage;

-- Step (5)
LOAD DATA INFILE 'D:/labs/CountryLanguage.txt'
INTO TABLE CountryLanguage2;

SELECT COUNT(*) FROM CountryLanguage2;
```

Practice 11-4: Backup Database Files with a Utility

```
-- Step (1)
-- At command prompt:
mysqldump -uroot -poracle world_innodb >
D:/labs/world_innodb_backup.sql

-- Step (2)
-- At command prompt:
mysqldump -uroot -poracle --skip-opt world_innodb >
D:/labs/world_innodb_backup2.sql

-- Step (3)
-- At mysql prompt:
CREATE DATABASE world_innodb2;

SHOW DATABASES;

EXIT

-- Step (4)
-- At command prompt:
mysql -uroot -poracle world_innodb2 <
D:/labs/world_innodb_backup2.sql

 mysql -uroot -poracle

-- At mysql prompt:
SHOW DATABASES;

USE world_innodb2

SHOW TABLES;

-- Step (5)
EXIT

--Step (6)
-- At command prompt:
mysqldump -uroot -poracle --tab=D:/labs world_innodb Country
```

Lesson 12: Joining Tables

Practice 12-1: Perform Inner and Outer Joins

```
-- Step (1)
-- At command prompt:
mysql -uroot -poracle

-- At mysql prompt:
USE world_innodb

DESC City;

DESC Country;

DESC CountryLanguage;

SELECT Name FROM City WHERE Name = 'San Antonio';

SELECT Country.Name, City.District
FROM Country
INNER JOIN City
ON CountryCode = Code
WHERE City.Name = 'San Antonio';

-- Step (2)
SELECT City.Name AS CapitalName, Country.Name AS CountryName
FROM City
INNER JOIN Country
ON City.ID = Country.Capital;

-- Step (3)
SELECT co.Name AS CountryName, ci.Name AS CityName
FROM Country AS co
LEFT JOIN City AS ci
ON co.Capital = ci.ID
WHERE co.Code IN ('CHE', 'ATA');

-- Step (4)
SELECT co.Name AS CountryName, ci.Name AS CityName
FROM Country AS co
RIGHT JOIN city AS ci
ON co.Capital = ci.ID
WHERE co.Code IN ('CHE', 'ATA');
```

Practice 12-2: Create Queries Requiring Joins

```
-- Step (1)
SELECT Name, Language
FROM CountryLanguage, Country
WHERE CountryCode = Code
AND Name = 'Sweden';
```

```
-- Step (2)
SELECT Name, Language
FROM CountryLanguage
INNER JOIN Country
ON CountryCode = Code;

-- Step (3)
SELECT Name, Language
FROM Country
LEFT JOIN CountryLanguage
ON CountryCode = Code;

-- Step (4)
SELECT Country.Name, COUNT(City.Name) AS Cities
FROM Country
INNER JOIN City
ON City.CountryCode = Country.Code
GROUP BY Country.Name
ORDER BY Cities DESC
LIMIT 1;

-- Step (5)
SELECT DISTINCT Country.Name FROM Country
INNER JOIN City ON Code = CountryCode
WHERE City.Population > 7000000;

-- Step (6)
USE Pets

DESC pet_info;

DESC owners;

DESC pet_types;

-- Step (7)
SELECT owners.oName
FROM owners
JOIN pet_info
ON owners.oID = pet_info.oID
WHERE oPhone LIKE '1555%' AND pGender = 'F';

-- Step (8)
SELECT oID, pName, pet_types.pType
FROM pet_info
RIGHT JOIN pet_types
ON pet_info.tID = pet_types.tID;

SELECT oID, pName, pet_types.pType
FROM pet_info
LEFT JOIN pet_types
ON pet_info.tID = pet_types.tID;
```

```
SELECT oID, pName, pet_types.pType
FROM pet_info
INNER JOIN pet_types
ON pet_info.tID = pet_types.tID;

-- Step (9)
-- If you are not doing the next/optional practice:
EXIT
```

Optional Practice 12-3: Additional Join Practice

```
-- Step (1)
USE world_innodb

SELECT DISTINCT Name, Language
FROM CountryLanguage, Country
WHERE CountryCode = Code
AND Name = 'Sweden';

-- Step (2)
SELECT Name, Language
FROM CountryLanguage, Country
WHERE CountryCode = Code
AND Name = 'Sweden'
ORDER BY Language DESC;

-- Step (3)
SELECT Name, Language
FROM CountryLanguage
LEFT JOIN Country
ON CountryCode = Code
WHERE Language = 'French';

-- Step (4)
SELECT Name, Language
FROM Country
LEFT JOIN CountryLanguage
ON CountryCode = Code
WHERE Language IS NULL;

-- Step (5)
-- no script

-- Step (6)
SELECT Name, Language
FROM Country
RIGHT JOIN CountryLanguage
ON CountryCode = Code
WHERE Language IS NULL;

-- Step (7)
SELECT Country.Name FROM Country
```

```
INNER JOIN City ON Code = CountryCode
WHERE City. Population > 7000000;

-- Step (8)
EXIT
```

Lesson 13: Table Subqueries

Practice 13-1: Perform Different Types of Subquery

```
-- Step (1)
-- At command prompt:
mysql -uroot -poracle

-- At mysql prompt:
USE world_innodb

SELECT Population
FROM City
WHERE Name = 'New York';

SELECT Name
FROM City
WHERE Population >
(SELECT Population
 FROM City
 WHERE Name = 'New York')
ORDER BY Population
LIMIT 3;

-- Step (2)
SELECT Country.Name,
(SELECT COUNT(*)
 FROM City
 WHERE CountryCode = Country.Code)
AS CityCount
FROM Country
WHERE Region = 'Nordic Countries';

-- Step (3)
SELECT Code
FROM Country
WHERE Name='Singapore';

SELECT Language
FROM CountryLanguage
WHERE CountryCode =
(SELECT Code
 FROM Country
 WHERE Name='Singapore')
ORDER BY Language DESC;
```

Practice 13-2: Perform Advanced Subqueries

```
-- Step (1)
SELECT CountryCode
FROM CountryLanguage
WHERE Language='English'
AND Percentage>50;
```

```
SELECT DISTINCT Continent
FROM Country
WHERE Code IN
(SELECT CountryCode
 FROM CountryLanguage
 WHERE Language='English'
AND Percentage>50);

-- Step (2)
SELECT CountryCode
FROM CountryLanguage
WHERE Language = 'Spanish';

SELECT Name
FROM Country
WHERE Continent = 'Europe'
AND Code IN
(SELECT CountryCode
FROM CountryLanguage
WHERE Language = 'Spanish')
ORDER BY Name;

-- Step (3)
SELECT MAX(Population) FROM City;

SELECT Country.Name
FROM Country JOIN City
ON Country.Code=City.CountryCode
WHERE City.Population =
(SELECT MAX(Population) FROM City);

-- Step (4)
SELECT * FROM Country c1
WHERE Continent = 'South America'
AND Population =
(SELECT MIN(Population)
 FROM Country c2
 WHERE c2.Continent = c1.Continent)\G

SELECT MIN(Population)
FROM Country c2
WHERE c2.Continent = c1.Continent;

-- Step (5)
SELECT MIN(Population) FROM Country
WHERE Continent = 'South America';

SELECT * FROM Country
WHERE Continent = 'South America'
AND Population =
(SELECT MIN(Population)
```

```
 FROM Country
 WHERE Continent = 'South America')\G

SELECT * FROM Country c1
WHERE Continent = 'South America'
AND Population =
(SELECT MIN(Population)
FROM Country c2
WHERE c2.Continent = c1.Continent)\G

-- Step (6)
EXIT;
```

Lesson 14: MySQL GUIs

No scripts for this lesson

Lesson 15: Supplemental Information

Practice 15-1: Displaying Storage Engine Information

```
-- Step (1)
-- At command prompt:
mysql -uroot -poracle

-- At mysql prompt:
USE world_innodb

SHOW CREATE TABLE CountryLanguage\G

-- Step (2)
SHOW TABLE STATUS LIKE 'City'\G

-- Step (3)
SHOW ENGINES\G
Practice 15-2: Display and Create Views
-- Step (1)
USE world_innodb

SHOW FULL TABLES;

-- Step (2)
CREATE VIEW CityView AS
SELECT ID, Name FROM City;

SHOW FULL TABLES;

-- Step (3)
DESCRIBE CityView;

-- Step (4)
SELECT COUNT(*) FROM CityView;
```

Practice 15-3: Obtain Metadata

```
-- Step (1)
SHOW DATABASES;

-- Step (2)
SHOW TABLES FROM INFORMATION_SCHEMA;

-- Step (3)
SELECT * FROM information_schema.schemata
WHERE SCHEMA_NAME = 'world_innodb'\G

-- Step (4)
DESC INFORMATION_SCHEMA.TABLES;

SELECT table_name, table_type, engine
FROM information_schema.tables
```

```
WHERE table_schema = 'world_innodb'
ORDER BY table_name DESC;

-- Step (5)
SELECT table_name, create_time, auto_increment
 FROM information_schema.tables
 WHERE table_schema = 'pets'
 ORDER BY table_name DESC;

-- Step (6)
EXIT
```

Optional Practice 15-4: Create a Backup of MySQL Databases

```
-- Step (1)
-- Using Windows Explorer go to D:\labs

-- Step (2)
-- At command prompt:
mysqlbackup --user=root --password=oracle --port=3306 --with-
timestamp --backup-

dir=D:/labs/backups backup

-- Step (3)
-- Using Windows Explorer go to D:\labs\backups
```