Practice 2: 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>
```

2.2: 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	

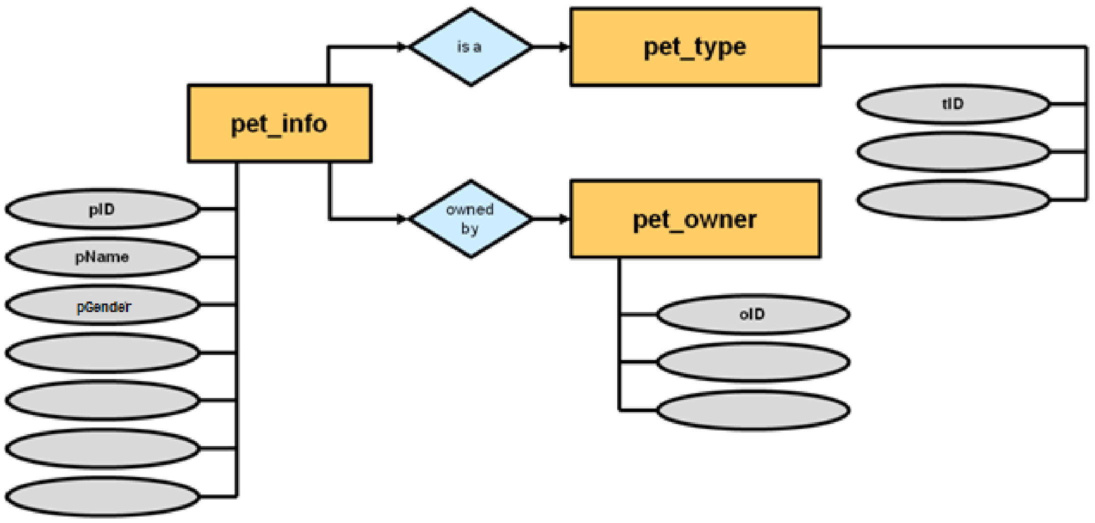
Note: Not all the information is available for every pet. Therefore, some of the columns must allow null values.

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:

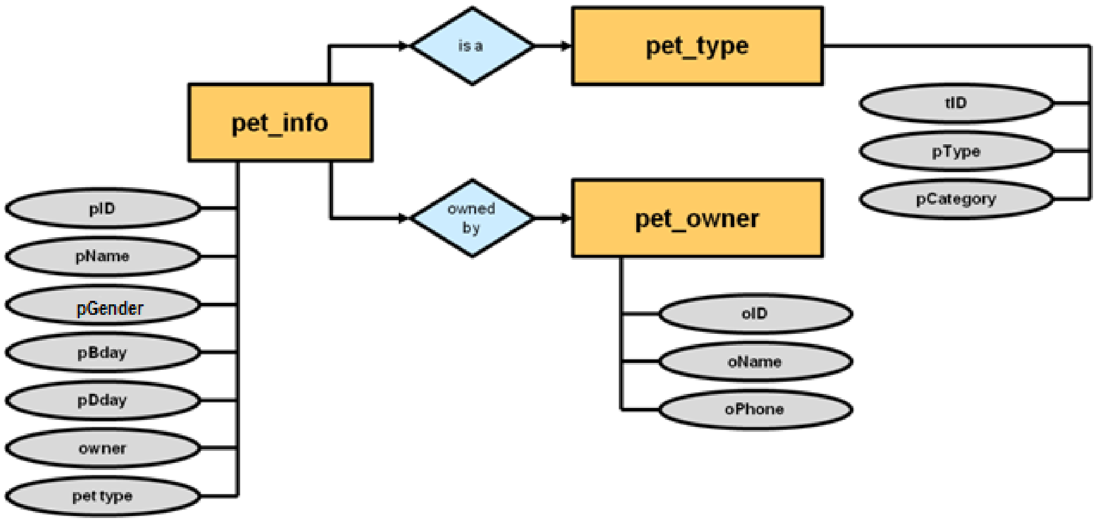
Attributes	**tID***	**pType**	**pCategory**
Unique?			
Options?			
Category?			
Data type?			

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 2-2: 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 3: 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 3-1: 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 3-1: Performing Basic Queries

Tasks

1. Start the `mysql` client and set the current database to `wo`
2.
3. `rld_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.

Practice 3-3: 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 3-3: 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.