CP4265 Assignment 4

Chapter 5 and 6

Exercises

To test whether a table has been modified correctly as you do these exercises, you can write and run an appropriate SELECT statement.

1. Write an INSERT statement that adds this row to the Categories table:

category_name:	Brass

   Code the INSERT statement so MySQL automatically generates the category_id column.

2. Write an UPDATE statement that modifies the row you just added to the Categories table. This statement should change the product_name column to “Woodwinds”, and it should use the category_id column to identify the row.

3. Write a DELETE statement that deletes the row you added to the Categories table in exercise 1. This statement should use the category_id column to identify the row.

4. Write an INSERT statement that adds this row to the Products table:

product_id:	The next automatically generated ID
category_id:	4
product_code:	dgx_640
product_name:	Yamaha DGX 640 88-Key Digital Piano
description:	Long description to come.
list_price:	799.99
discount_percent:	0
date_added:	Today’s date/time.

   Use a column list for this statement.

5. Write an UPDATE statement that modifies the product you added in exercise 4. This statement should change the discount_percent column from 0% to 35%.

6. Write a DELETE statement that deletes the Keyboards category. When you execute this statement, it will produce an error since the category has related rows in the Products table. To fix that, precede the DELETE statement with another DELETE statement that deletes all products in this category. (Remember that to code two or more statements in a script, you must end each statement with a semicolon.)

7. Write an INSERT statement that adds this row to the Customers table:

email_address:	rick@raven.com
password:	(empty string)
first_name:	Rick
last_name:	Raven

Use a column list for this statement.

8. Write an UPDATE statement that modifies the Customers table. Change the password column to “secret” for the customer with an email address of rick@raven.com.
9. Write an UPDATE statement that modifies the Customers table. Change the password column to “reset” for every customer in the table. If you get an error due to safe-update mode, you can add a LIMIT clause to update the first 100 rows of the table. (This should update all rows in the table.)
10. Open the script named create_my_guitar_shop.sql that’s in the mgs_ex_starts directory. Then, run this script. That should restore the data that’s in the database.

Chapter 6: How to code summary queries

1. Write a SELECT statement that returns these columns:

    The count of the number of orders in the Orders table

    The sum of the tax_amount columns in the Orders table

2. Write a SELECT statement that returns one row for each category that has products with these columns:

    The category_name column from the Categories table

    The count of the products in the Products table

    The list price of the most expensive product in the Products table

   Sort the result set so the category with the most products appears first.

3. Write a SELECT statement that returns one row for each customer that has orders with these columns:

    The email_address column from the Customers table

    The sum of the item price in the Order_Items table multiplied by the quantity in the Order_Items table

    The sum of the discount amount column in the Order_Items table multiplied by the quantity in the Order_Items table

   Sort the result set in descending sequence by the item price total for each customer.

4. Write a SELECT statement that returns one row for each customer that has orders with these columns:

    The email_address column from the Customers table

    A count of the number of orders

    The total amount for each order (*Hint: First, subtract the discount amount from the price. Then, multiply by the quantity.*)

   Return only those rows where the customer has more than 1 order.

   Sort the result set in descending sequence by the sum of the line item amounts.

5. Modify the solution to exercise 4 so it only counts and totals line items that have an item_price value that’s greater than 400.