Chapter 7
How to code subqueries

Exercises

1. Write a SELECT statement that returns the same result set as this SELECT statement, but don't use a join. Instead, use a subquery in a WHERE clause that uses the IN keyword.

```
SELECT DISTINCT category_name
FROM categories c JOIN products p
  ON c.category_id = p.category_id
ORDER BY category_name
```

2. Write a SELECT statement that answers this question: Which products have a list price that's greater than the average list price for all products?

   Return the product_name and list_price columns for each product.

   Sort the result set by the list_price column in descending sequence.

3. Write a SELECT statement that returns the category_name column from the Categories table.

   Return one row for each category that has never been assigned to any product in the Products table. To do that, use a subquery introduced with the NOT EXISTS operator.

4. Write a SELECT statement that returns three columns: email_address, order_id, and the order total for each order. To do this, you can group the result set by the email_address and order_id columns. In addition, you must calculate the order total from the columns in the Order_Items table.

   Write a second SELECT statement that uses the first SELECT statement in its FROM clause. The main query should return two columns: the customer's email address and the largest order for that customer. To do this, you can group the result set by the email_address. Sort the result set by the largest order in descending sequence.

   While I recommend that you do create the first SECLECT statement described above, you do not need to submit that first SELECT statement separately. Instead, create the second SELECT statement (which incorporates the first). Then, submit only this second (combined) statement as your solution to Exercise 4.

5. Write a SELECT statement that returns the name and discount percent of each product that has a unique discount percent. In other words, don't include products that have the same discount percent as another product.

   Sort the result set by the product_name column.

6. Use a correlated subquery to return one row per customer, representing the customer's oldest order (the one with the earliest date). Each row should include these three columns: email_address, order_id, and order_date.

   Sort the result set by the order_date and order_id columns.

PLEASE NOTE: Exercise 7 is based on features that are new to MySQL and material that is new to the Murach 3e text. So, this exercise is being offered here for a "test drive". While we will review both student and official solutions to this exercise in class, this exercise will not be counted toward your grade for this assignment.

7. Using the SELECT statement that you created in Exercise 4 as a starting point, re-write the query to use a *Common Table Expression* (CTE) rather than a subquery.