

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
```

1. 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 results by the list_price column in descending sequence.

2. 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.

3. Write a SELECT statement that returns three columns: email_address, order_id, and the order total for each customer. 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.

4. 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 results by the product_name column.

5. 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.

6. 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.

7. 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.
8. 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.)

Open the script named create_my_guitar_shop.sql that's in the mgs_ex_starts directory. Then, run this