Chapter 6
How to code summary queries

Exercises

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.

6. Write a SELECT statement that answers this question: What is the total amount ordered for each product? Return these columns:

   The product_name column from the Products table

   The total amount for each product in the Order_Items table (*Hint: You can calculate the total amount by subtracting the discount amount from the item price and then multiplying it by the quantity*)

   Use the WITH ROLLUP operator to include a row that gives the grand total.

   Note: Once you add the WITH ROLLUP operator, you may need to use MySQL Workbench's Execute SQL Script button instead of its Execute Current Statement button to execute this statement.

7. Write a SELECT statement that answers this question: Which customers have ordered more than one product? Return these columns:

   The email_address column from the Customers table

   The count of distinct products from the customer's orders

   Sort the result set in ascending sequence by the email_address column.

PLEASE NOTE: Exercises 8 through 10 are based in features that are new to MySQL and material that is new to the Murach 3e text. So, these exercises are being offered here for a "test drive". While we will review both student and official solutions to these exercises in class, these exercises will not be counted toward your grade for this assignment.

8. Write a SELECT statement that answers this question: What is the total quantity purchased for each product within each category? Return these columns:

   The category_name column from the category table

   The product_name column from the products table

   The total quantity purchased for each product with orders in the Order_Items table

   Use the WITH ROLLUP operator to include rows that give a summary for each category name as well as a row that gives the grand total.

   Use the IF and GROUPING functions to replace null values in the category_name and product_name columns with literal values if they're for summary rows.

9. Write a SELECT statement that uses an aggregate window function to get the total amount of each order. Return these columns:

   The order_id column from the Order_Items table

   The total amount for each order item in the Order_Items table (*Hint: You can calculate the total amount by subtracting the discount amount from the item price and then multiplying it by the quantity*)

   The total amount for each order

   Sort the result set in ascending sequence by the order_id column.

10. Modify the solution to exercise 9 so the column that contains the total amount for each order contains a cumulative total by item amount.

    Add another column to the SELECT statement that uses an aggregate window function to get the average item amount for each order.

    Modify the SELECT statement so it uses a named window for the two aggregate functions.