Assignment2- Chapter 5 and 6

-50 points

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

   The category_name column from the Categories table

   The count of the instruments in the Instruments table

   The list price of the most expensive instrument in the Instruments table

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

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

   The email_address column from the Musicians table

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

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

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

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

   The email_address column from the Musicians 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 musician has more than 1 order.

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

4. Modify the solution to exercise 3 so it only counts and totals line items that have an item_price value that's greater than 400.

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

   The instrument_name column from the Instruments table

   The total amount for each instrument 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.

6. Write a SELECT statement that answers this question: Which musicians have ordered more than one instrument? Return these columns:

   The email_address column from the Musicians table

   The count of distinct instruments from the musician's orders

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

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

   The category_name column from the category table

The instrument_name column from the instruments table

The total quantity purchased for each instrument 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 instrument_name columns with literal values if they're for summary rows.

8. 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.

9. Modify the solution to exercise 8 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.

10. Write a SELECT statement that uses aggregate window functions to calculate the order total for each musician and the order total for each musician by date. Return these columns:

    The musician_id column from the Orders table

    The order_date column from the Orders table

    The total amount for each order item in the Order_Items table

    The sum of the order totals for each musician

    The sum of the order totals for each musician by date *(Hint: You can create a peer group to get these values)*