**CCGC 5004 Database Systems**

**Lab Exercise 5 Summary Queries**

**Overview**

**To receive credit for this lab you must be present in today’s class. Late submissions are deducted 5% per day up to 5 days. Submissions received after 5 days will be given a grade of 0.**

**Since you may have made modifications to the tables in the previous exercise using DML commands you may wish to refresh your tables by executing the scrip again. Some of you indicated that you used the CTAS command to create a set of tables that would permit you to work on the tables without causing any changes to the original tables. You will not need not need to refresh the tables. The script is in Lesson 5. Make sure you are connected to your schema with the USE command.**

**You will write and execute queries as requested in the questions below.**

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

Screen Capture 1

1. 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

**Screen Capture** **2**.

1. 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. **Screen Capture 3**.

1. 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. **Screen Capture 4**

1. 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. **Screen Capture5**
2. 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.* **Screen Capture 6**

1. 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. **Screen Capture 7**

1. 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. Sc**reen Capture 8**