**MySQL: Writing Queries**

Stephan Peters

Colorado State University Global

23WC-ITS410-2: Database Management

Jamia Mills

18 February 2024

## MySQL: Writing Queries

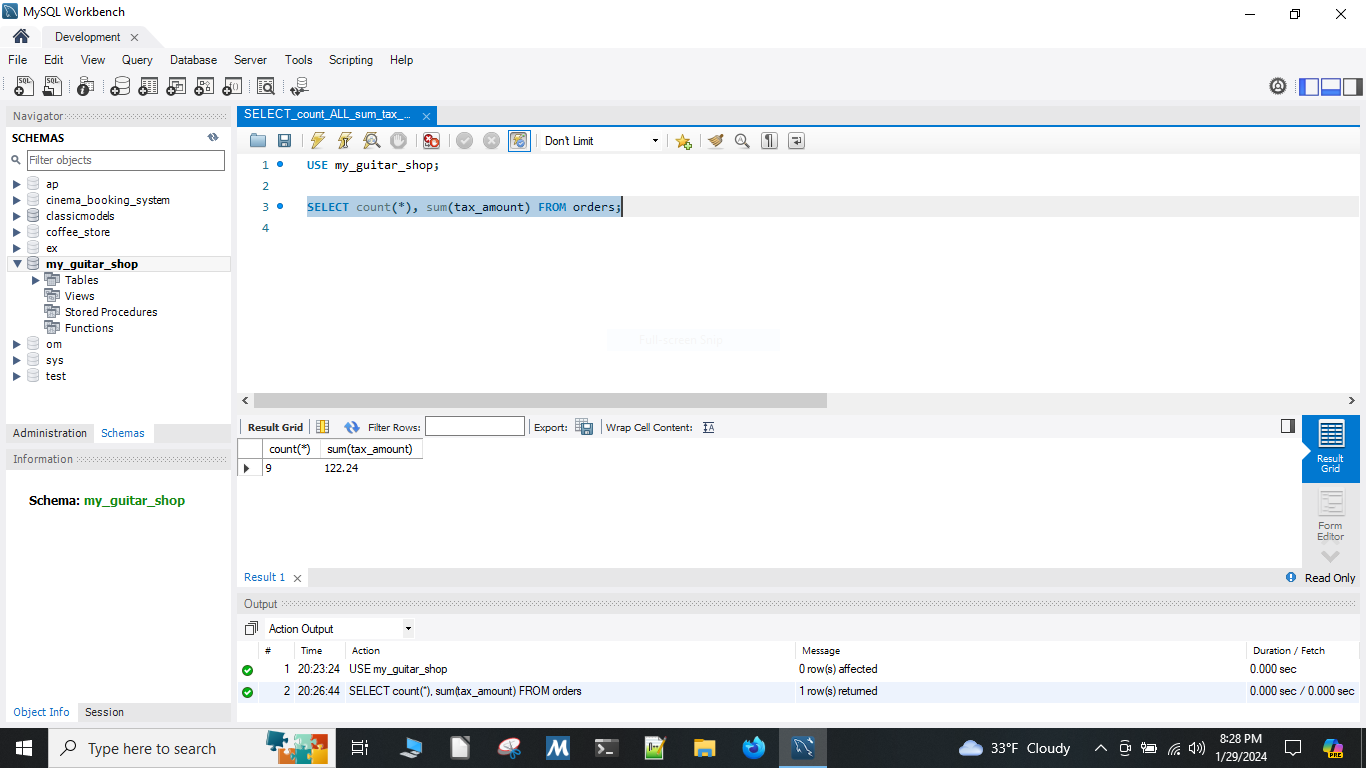
Full resolution images of the figures shown may be viewed on my public GitHub repository here: <https://github.com/speters33w/CSUGlobal_ITS410/blob/main/CriticalThinking5/Module_5_Critical_Thinking.md>, or with CSU Global Login via Google Drive here: <https://drive.google.com/drive/folders/1nKcaGnn44djHxhVxw4bNWfAykEXrzjV5?usp=sharing>

## Using Aggregate Functions

This query returns the count of the number of orders in the orders table and the sum of the tax amount columns in the orders table.

Figure 1

SELECT count(\*), sum(tax\_amount) FROM orders;

****

## Using GROUP BY in a MySQL query

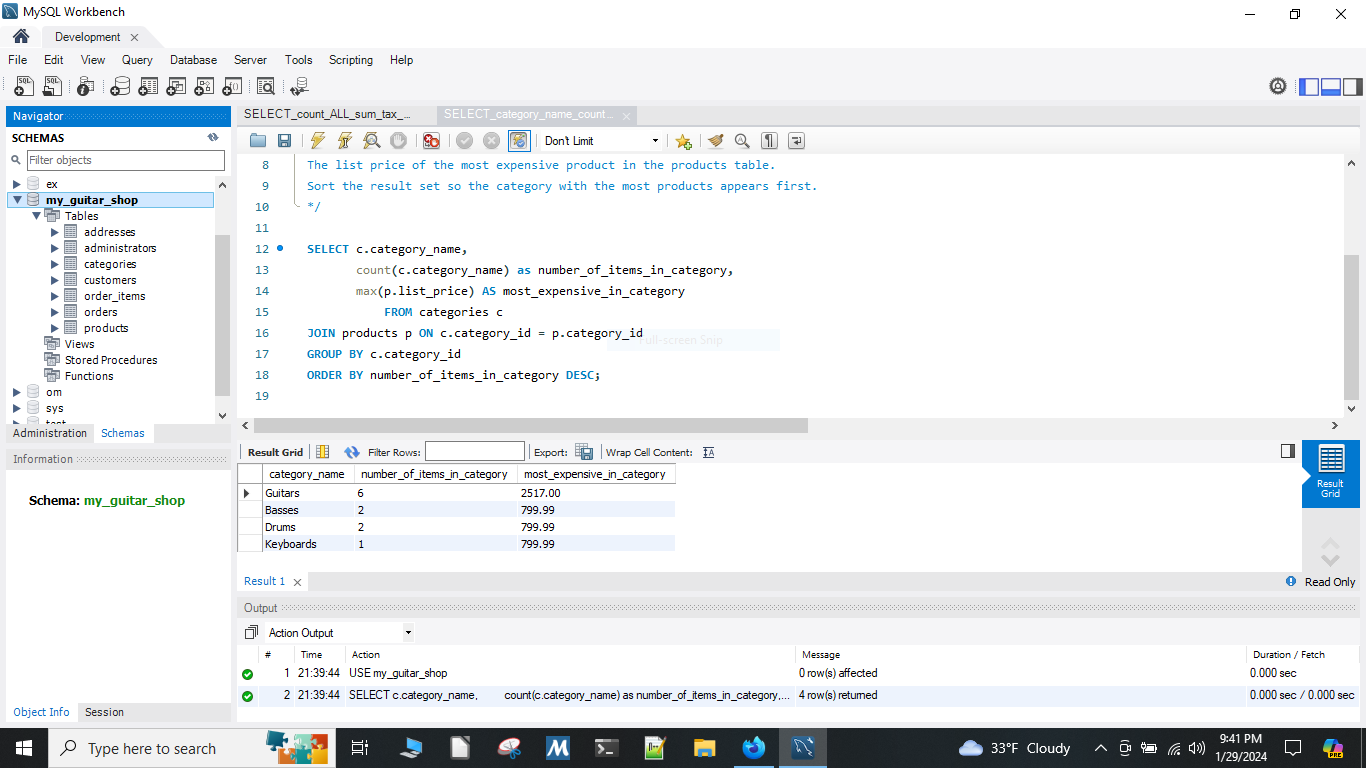
This query returns one row for each category that has products with these columns:

* 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.

Sorted so the category with the most products appears first.

Figure 2

SELECT c.category\_name, count(c.category\_name) as number\_of\_items\_in\_category, max(p.list\_price) AS most\_expensive\_in\_category FROM categories c JOIN products p ON c.category\_id = p.category\_id GROUP BY c.category\_id ORDER BY number\_of\_items\_in\_category DESC;

****

## Using sum() in a MySQL Query

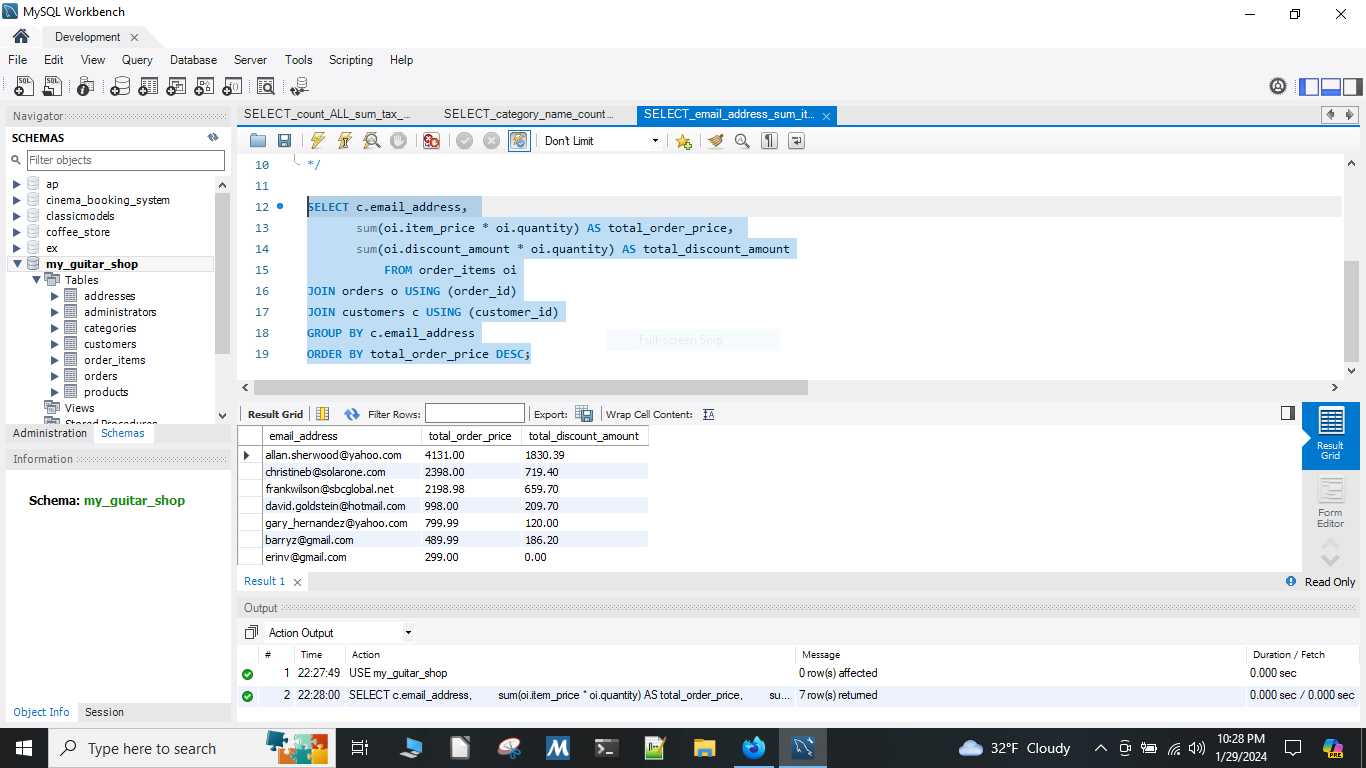
This query returns one row for each customer that has orders with these columns:

* 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

sorted in descending sequence by the item price total for each customer.

Figure 3

SELECT c.email\_address, sum(oi.item\_price \* oi.quantity) AS total\_order\_price, sum(oi.discount\_amount \* oi.quantity) AS total\_discount\_amount FROM order\_items oi JOIN orders o USING (order\_id) JOIN customers c USING (customer\_id) GROUP BY c.email\_address ORDER BY total\_order\_price DESC;

****

## Using sum() and count(DISTINCT) in a MySQL Query

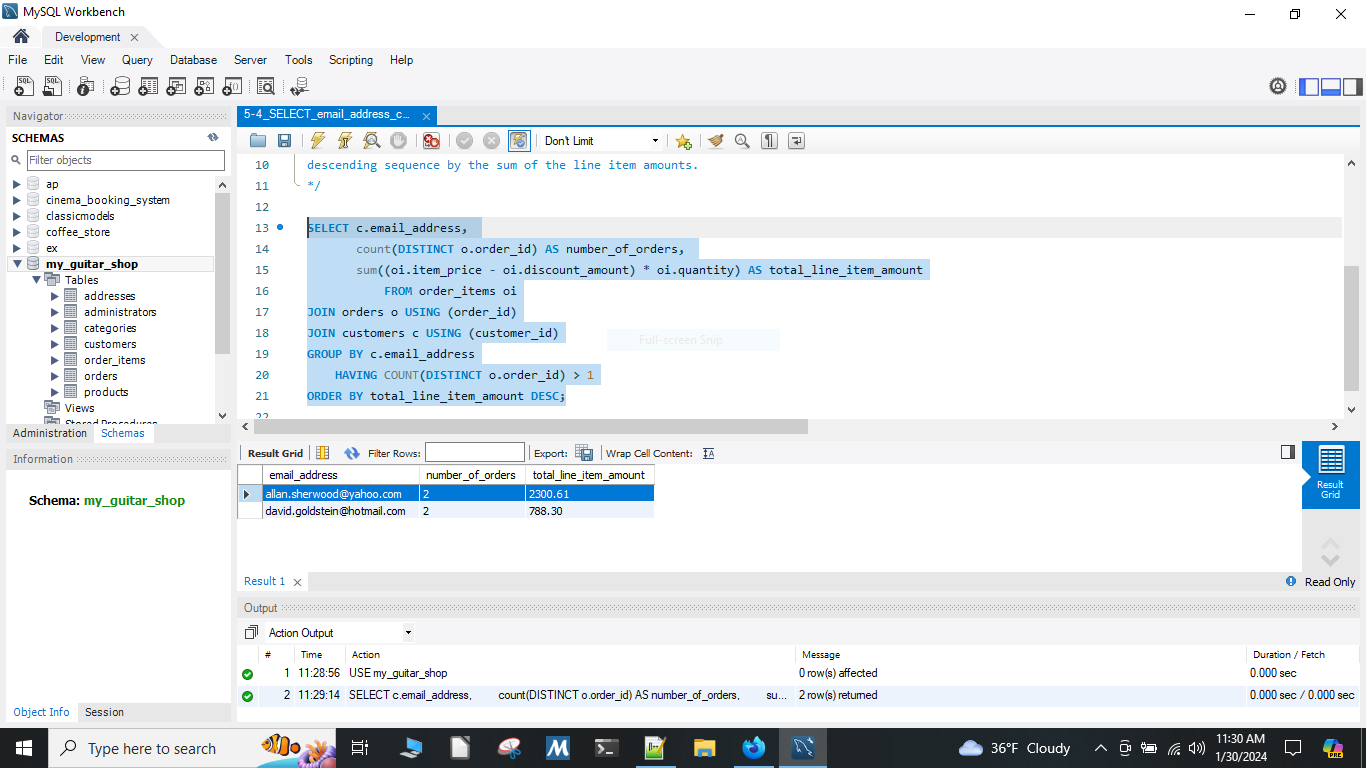
This query returns one row for each customer that has orders with these columns:

* The email address column from the customers table
* A count of the number of orders
* The total amount for each order

It returns only those rows where the customer has more than one order, sorted in descending sequence by the sum of the line-item amounts.

Figure 4

SELECT c.email\_address, count(DISTINCT o.order\_id) AS number\_of\_orders, sum((oi.item\_price - oi.discount\_amount) \* oi.quantity) AS total\_line\_item\_amount FROM order\_items oi JOIN orders o USING (order\_id) JOIN customers c USING (customer\_id) GROUP BY c.email\_address HAVING COUNT(DISTINCT o.order\_id) > 1 ORDER BY total\_line\_item\_amount DESC;

****