

Critical Thinking Assignment 5: Writing Queries

Alexander Ricciardi

Colorado State University Global

ITS410: Database Management

Dr. Murthy Rallapalli

May 18, 2025

Critical Thinking Assignment 5: Writing Queries

This documentation is part of the Critical Thinking 5 Assignment from ITS410: Database Management at Colorado State University Global. The documentation provides screenshots showcasing writing queries using MySQL and the My Guitar Shop database.

The Assignment Direction:

Writing Queries

Using the My Guitar Shop database you installed in Module 1, develop the following queries.

- 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

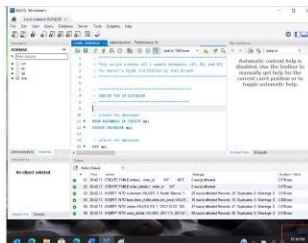
Execute the query and take a screenshot of the query and the results.
- Write a SELECT statement that returns one row for each category that has products with these columns:
 - The category_name column from the Categories table
 - The count of the products in the Products table
 - The list price of the most expensive product in the Products table.
 - Sort the result set so the category with the most products appears first.

Execute the query, and take a screenshot of the query and the results.
- 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.

Execute the query and take a screenshot of the query and the results.
- 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 one order.
 - Sort the result set in descending sequence by the sum of the line item amounts.

Execute the query and take a screenshot of the query and the results

All the screenshots should show current date. Example of screenshot.



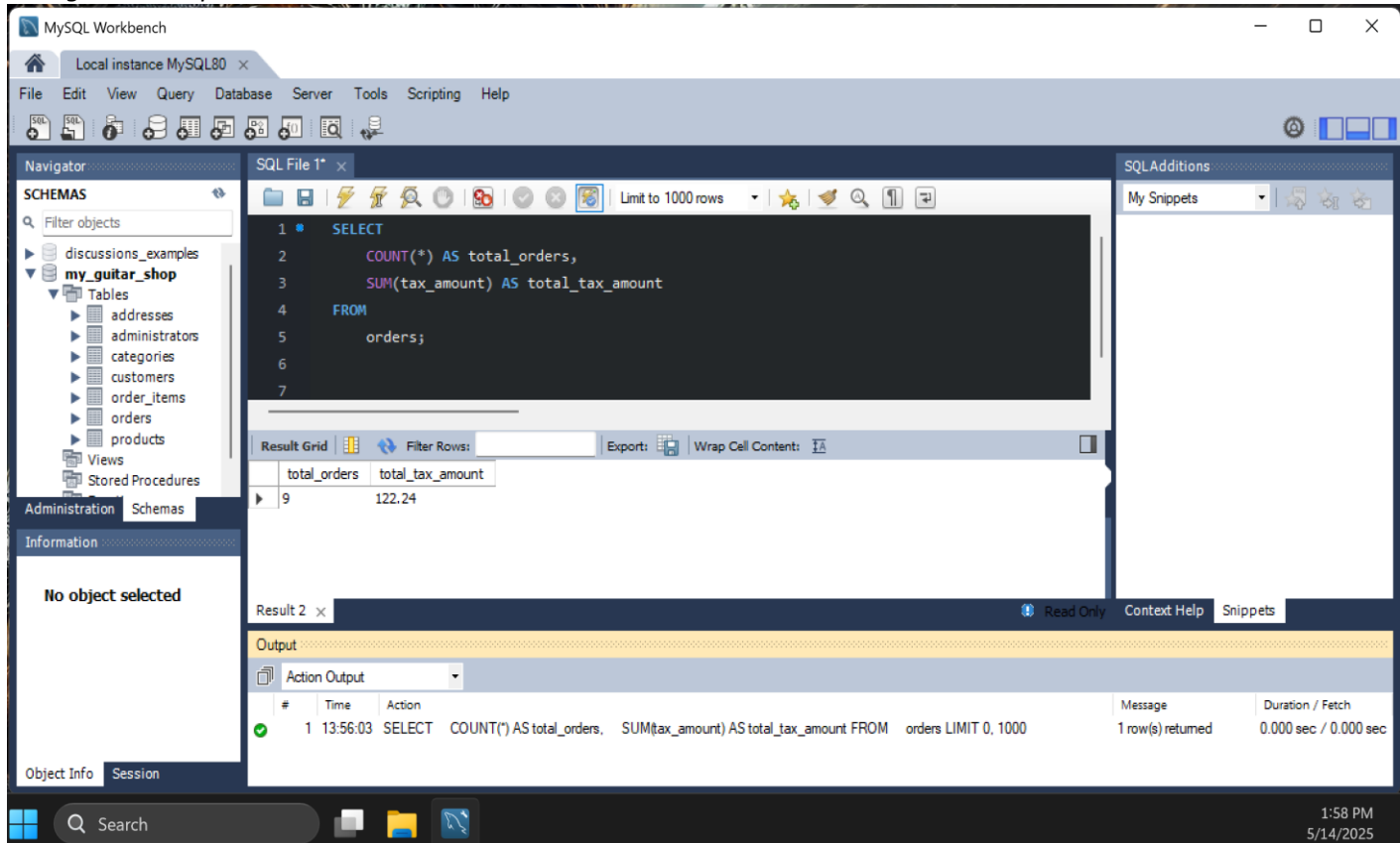
Submit your labeled results screenshots in a Word file.

Screenshots

Step 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

Figure 1
Assignment Steps 1



Note: The figure illustrates the MySQL Workbench result after performing steps 1.

Please see the next page.

Step 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.

Figure 2
Assignment Steps 2

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```

1  SELECT
2      c.category_name,
3      COUNT(p.product_id) AS number_of_products,
4      MAX(p.list_price) AS most_expensive_product
5  FROM
6      categories c
7  JOIN
8      products p ON c.category_id = p.category_id
9  GROUP BY
10     c.category_id, c.category_name
11  ORDER BY
12     number_of_products DESC;
13
14

```

The Results window shows the following data:

category_name	number_of_products	most_expensive_product
Guitars	6	2517.00
Basses	2	799.99
Drums	2	799.99
Keyboards	1	799.99

The Output window shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	14:18:35	SELECT	c.category_name, COUNT(p.product_id) AS number_of_products, MAX(p.li...	4 row(s) return... 0.000 sec / 0.000 sec

Note: The figure illustrates the MySQL Workbench result after performing steps 2.

Please see the next page.

Step 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.

Figure 3
Assignment Step 3

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```

1  SELECT
2      c.email_address,
3      SUM(oi.item_price * oi.quantity) AS total_item_price,
4      SUM(oi.discount_amount * oi.quantity) AS total_discount_amount
5  FROM
6      customers c
7  JOIN
8      orders o ON c.customer_id = o.customer_id
9  JOIN
10     order_items oi ON o.order_id = oi.order_id
11 GROUP BY
12     c.customer_id, c.email_address
13 ORDER BY
14     total_item_price DESC;
15

```

The Results window displays the following data:

email_address	total_item_price	total_discount_amount
allan.sherwood@yahoo.com	4131.00	1830.39
christineb@solarone.com	2398.00	719.40
frankwilson@sbcglobal.net	2198.98	659.70
david.goldstein@hotmail.com	998.00	209.70
gary_hernandez@yahoo.com	799.99	120.00
barryz@gmail.com	489.99	186.20
erinv@gmail.com	299.00	0.00

The Output window shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	15:04:53	SELECT	c.email_address, SUM(oi.item_price * o...	7 row(s) returned 0.000 sec / 0.000 sec

Note: The figure illustrates the MySQL Workbench result after performing steps 3.

Step 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 one order.
- Sort the result set in descending sequence by the sum of the line item amounts.

Note that the directions do not ask to incorporate the taxes into the total.

Figure 4
Assignment Step 4

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```

1 SELECT
2     c.email_address,
3     COUNT(DISTINCT o.order_id) AS number_of_orders,
4     SUM((oi.item_price - oi.discount_amount) * oi.quantity) AS total_order_customer
5 FROM
6     customers c
7 JOIN
8     orders o ON c.customer_id = o.customer_id
9 JOIN
10    order_items oi ON o.order_id = oi.order_id
11 GROUP BY
12     c.customer_id, -- In case 2 different customer ids have the same email address
13     c.email_address
14 HAVING
15     COUNT(DISTINCT o.order_id) > 1
16 ORDER BY
17     total_order_customer DESC;
18

```

The Results window shows the following data:

email_address	number_of_orders	total_order_customer
allan.sherwood@yahoo.com	2	2300.61
david.goldstein@hotmail.com	2	788.30

The Output window shows the following message:

```

#    Time    Action
1  15:59:03  SELECT  c.email_address,  COUNT(DISTINCT o.order_id) AS num...  2 row(s) returned

```

Note: The figure illustrates the MySQL Workbench result after performing steps 4.