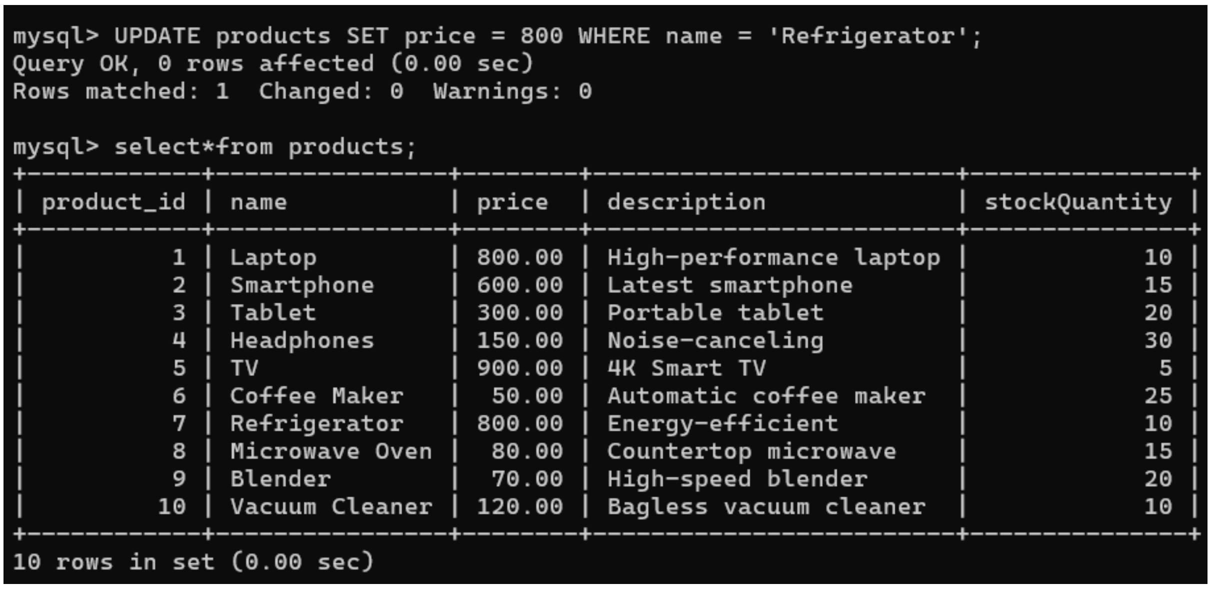
SQL Coding Challenge – 25/03/2025

Name: Monish Coumar S

Roll number: J208

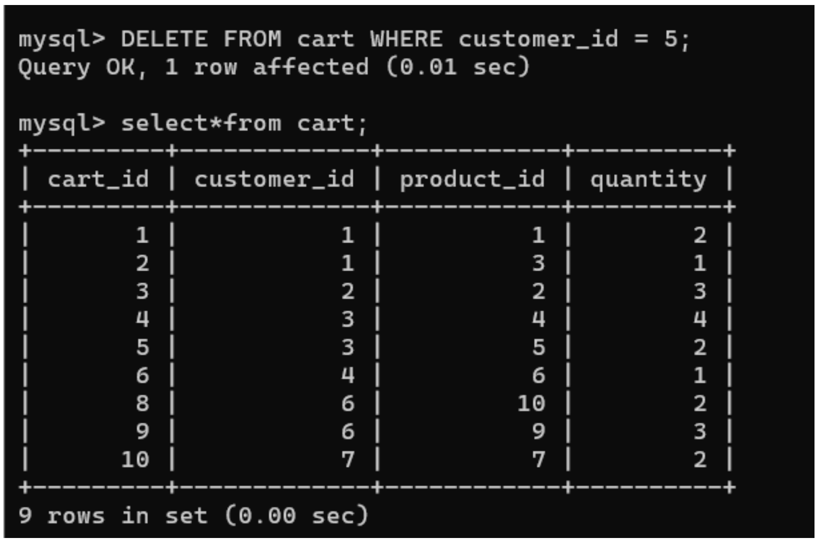
SQL QUERIES:

1.Update refrigerator product price to 800



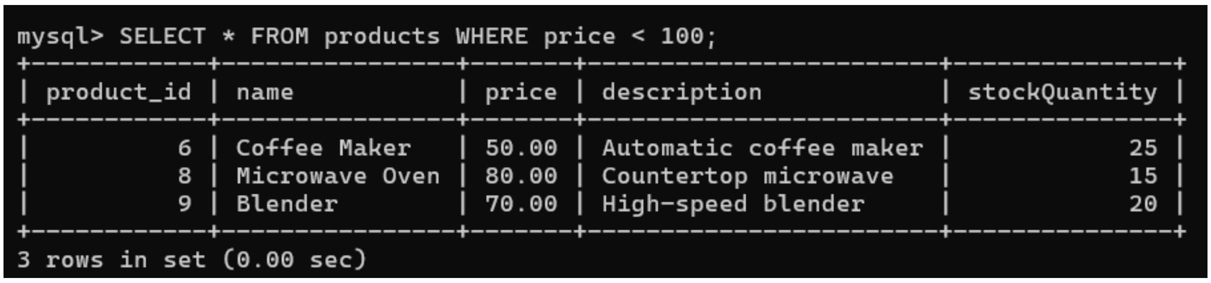
```
mysql> UPDATE products SET price = 800 WHERE name = 'Refrigerator';
Query OK, 0 rows affected (0.00 sec)
Rows matched: 1  Changed: 0  Warnings: 0

mysql> select*from products;
+------------+----------------+--------+-------------------------+---------------+
| product_id | name           | price  | description             | stockQuantity |
+------------+----------------+--------+-------------------------+---------------+
|          1 | Laptop         | 800.00 | High-performance laptop |            10 |
|          2 | Smartphone     | 600.00 | Latest smartphone       |            15 |
|          3 | Tablet         | 300.00 | Portable tablet         |            20 |
|          4 | Headphones     | 150.00 | Noise-canceling         |            30 |
|          5 | TV             | 900.00 | 4K Smart TV             |             5 |
|          6 | Coffee Maker   |  50.00 | Automatic coffee maker  |            25 |
|          7 | Refrigerator   | 800.00 | Energy-efficient        |            10 |
|          8 | Microwave Oven |  80.00 | Countertop microwave    |            15 |
|          9 | Blender        |  70.00 | High-speed blender      |            20 |
|         10 | Vacuum Cleaner | 120.00 | Bagless vacuum cleaner  |            10 |
+------------+----------------+--------+-------------------------+---------------+
10 rows in set (0.00 sec)
```

2. Remove all cart items for a specific customer



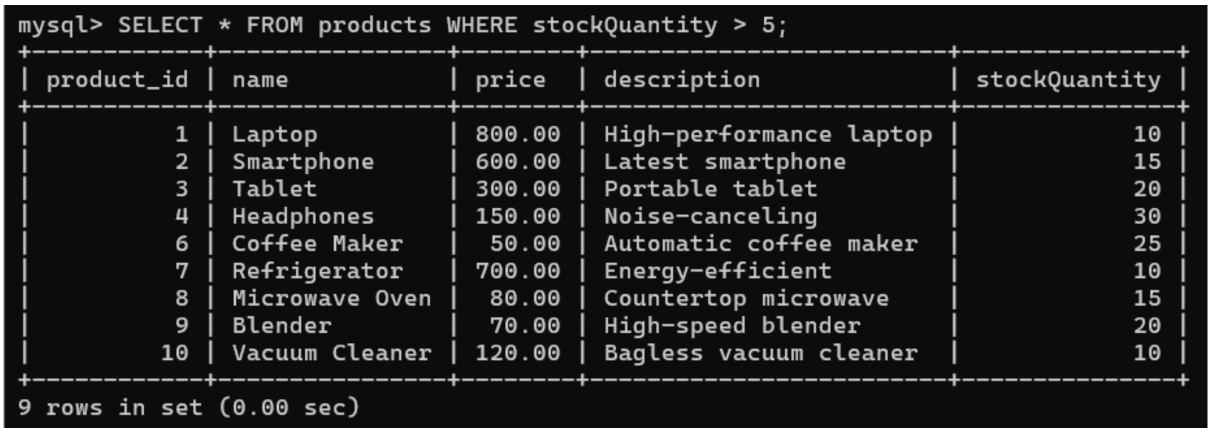
```
mysql> DELETE FROM cart WHERE customer_id = 5;
Query OK, 1 row affected (0.01 sec)

mysql> select*from cart;
+---------+-------------+------------+----------+
| cart_id | customer_id | product_id | quantity |
+---------+-------------+------------+----------+
|       1 |           1 |          1 |        2 |
|       2 |           1 |          3 |        1 |
|       3 |           2 |          2 |        3 |
|       4 |           3 |          4 |        4 |
|       5 |           3 |          5 |        2 |
|       6 |           4 |          6 |        1 |
|       8 |           6 |         10 |        2 |
|       9 |           6 |          9 |        3 |
|      10 |           7 |          7 |        2 |
+---------+-------------+------------+----------+
9 rows in set (0.00 sec)
```

3. Retrieve Products Priced Below 100



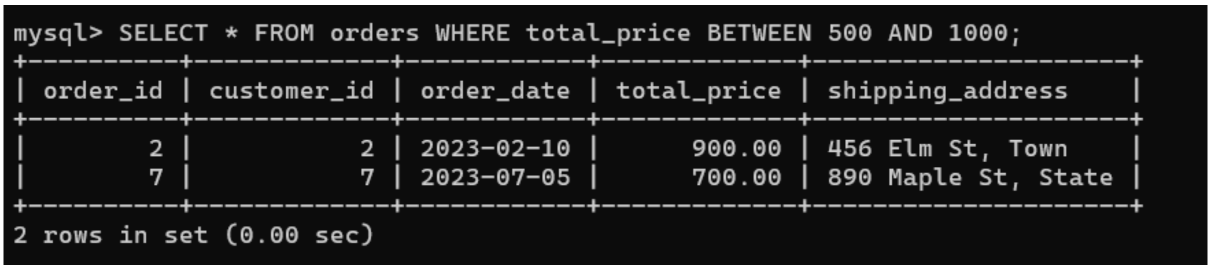
```
mysql> SELECT * FROM products WHERE price < 100;
+------------+----------------+-------+------------------------+---------------+
| product_id | name           | price | description            | stockQuantity |
+------------+----------------+-------+------------------------+---------------+
|          6 | Coffee Maker   | 50.00 | Automatic coffee maker |            25 |
|          8 | Microwave Oven | 80.00 | Countertop microwave   |            15 |
|          9 | Blender        | 70.00 | High-speed blender     |            20 |
+------------+----------------+-------+------------------------+---------------+
3 rows in set (0.00 sec)
```

4. Find Products with Stock Quantity Greater Than 5



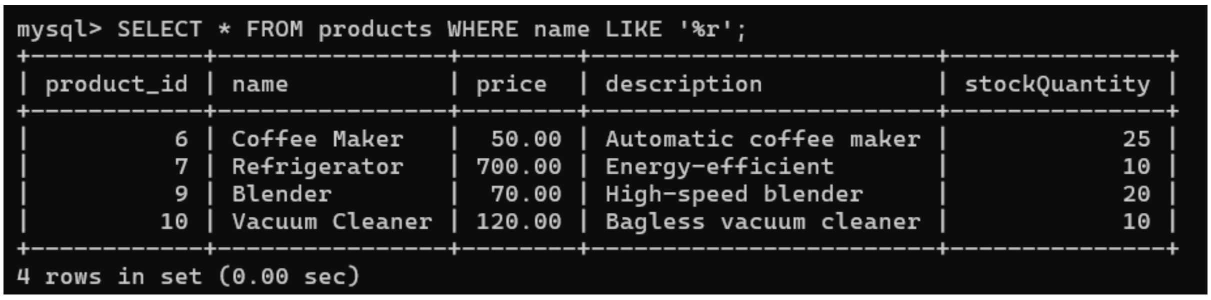
```
mysql> SELECT * FROM products WHERE stockQuantity > 5;
+------------+----------------+--------+-------------------------+---------------+
| product_id | name           | price  | description             | stockQuantity |
+------------+----------------+--------+-------------------------+---------------+
|          1 | Laptop         | 800.00 | High-performance laptop |            10 |
|          2 | Smartphone     | 600.00 | Latest smartphone       |            15 |
|          3 | Tablet         | 300.00 | Portable tablet         |            20 |
|          4 | Headphones     | 150.00 | Noise-canceling         |            30 |
|          6 | Coffee Maker   |  50.00 | Automatic coffee maker  |            25 |
|          7 | Refrigerator   | 700.00 | Energy-efficient        |            10 |
|          8 | Microwave Oven |  80.00 | Countertop microwave    |            15 |
|          9 | Blender        |  70.00 | High-speed blender      |            20 |
|         10 | Vacuum Cleaner | 120.00 | Bagless vacuum cleaner  |            10 |
+------------+----------------+--------+-------------------------+---------------+
9 rows in set (0.00 sec)
```

5. Retrieve Orders with Total Amount Between $500 and $1000



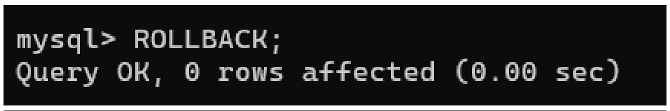
```
mysql> SELECT * FROM orders WHERE total_price BETWEEN 500 AND 1000;
+----------+-------------+------------+-------------+---------------------+
| order_id | customer_id | order_date | total_price | shipping_address    |
+----------+-------------+------------+-------------+---------------------+
|        2 |           2 | 2023-02-10 |      900.00 | 456 Elm St, Town    |
|        7 |           7 | 2023-07-05 |      700.00 | 890 Maple St, State |
+----------+-------------+------------+-------------+---------------------+
2 rows in set (0.00 sec)
```

6. Find Products which name end with letter 'r'

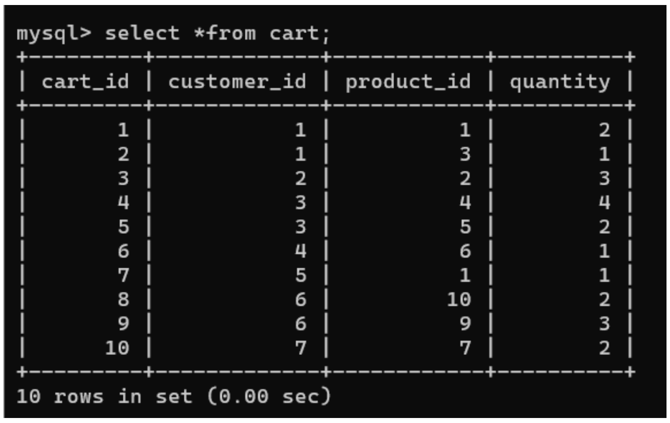


```
mysql> SELECT * FROM products WHERE name LIKE '%r';
+------------+----------------+--------+------------------------+---------------+
| product_id | name           | price  | description            | stockQuantity |
+------------+----------------+--------+------------------------+---------------+
|          6 | Coffee Maker   |  50.00 | Automatic coffee maker |            25 |
|          7 | Refrigerator   | 700.00 | Energy-efficient       |            10 |
|          9 | Blender        |  70.00 | High-speed blender     |            20 |
|         10 | Vacuum Cleaner | 120.00 | Bagless vacuum cleaner |            10 |
+------------+----------------+--------+------------------------+---------------+
4 rows in set (0.00 sec)
```

7. Retrieve Cart Items for Customer 5

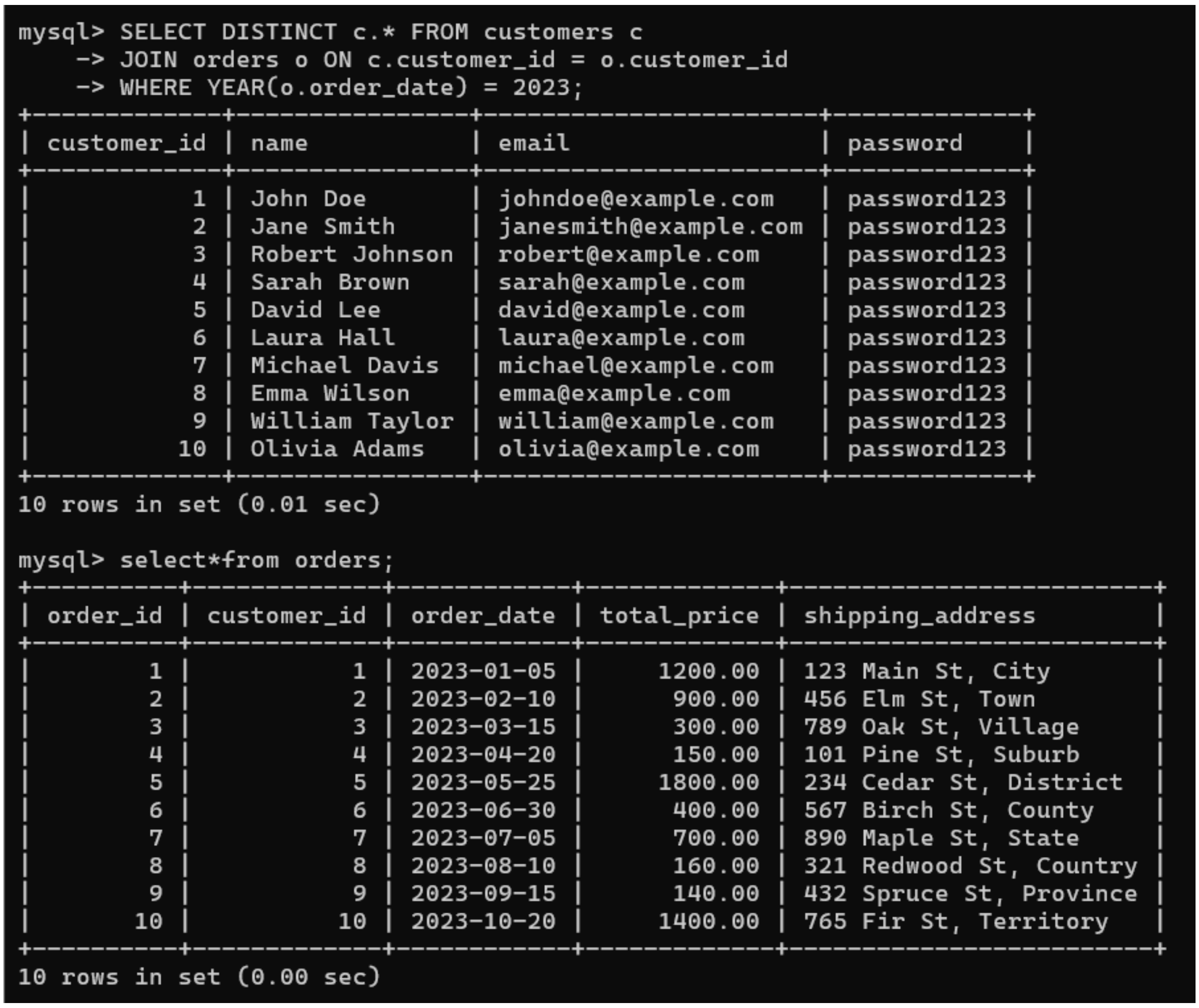


```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
```



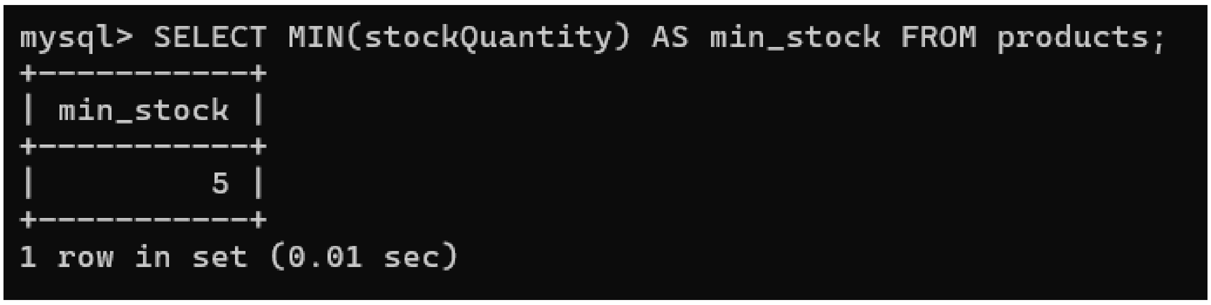
```
mysql> select *from cart;
+---------+-------------+------------+----------+
| cart_id | customer_id | product_id | quantity |
+---------+-------------+------------+----------+
|       1 |           1 |          1 |        2 |
|       2 |           1 |          3 |        1 |
|       3 |           2 |          2 |        3 |
|       4 |           3 |          4 |        4 |
|       5 |           3 |          5 |        2 |
|       6 |           4 |          6 |        1 |
|       7 |           5 |          1 |        1 |
|       8 |           6 |         10 |        2 |
|       9 |           6 |          9 |        3 |
|      10 |           7 |          7 |        2 |
+---------+-------------+------------+----------+
10 rows in set (0.00 sec)
```

8. Find Customers Who Placed Orders in 2023



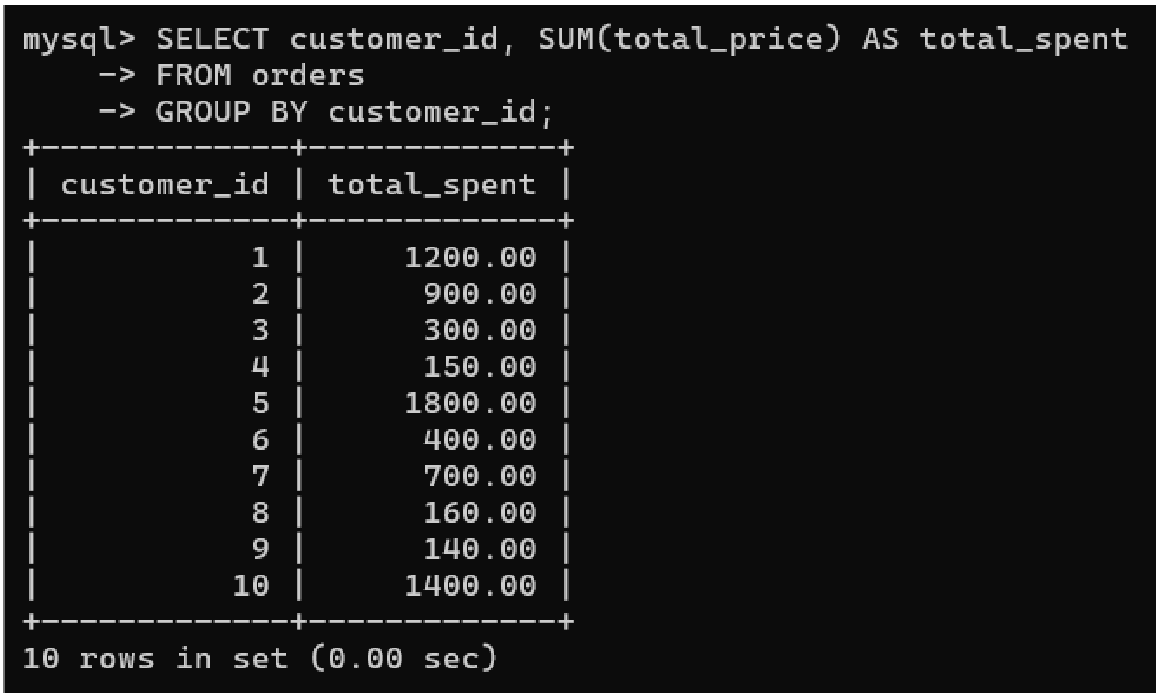
```
mysql> SELECT DISTINCT c.* FROM customers c
    -> JOIN orders o ON c.customer_id = o.customer_id
    -> WHERE YEAR(o.order_date) = 2023;
+-------------+----------------+-----------------------+-------------+
| customer_id | name           | email                 | password    |
+-------------+----------------+-----------------------+-------------+
|           1 | John Doe       | johndoe@example.com   | password123 |
|           2 | Jane Smith     | janesmith@example.com | password123 |
|           3 | Robert Johnson | robert@example.com    | password123 |
|           4 | Sarah Brown    | sarah@example.com     | password123 |
|           5 | David Lee      | david@example.com     | password123 |
|           6 | Laura Hall     | laura@example.com     | password123 |
|           7 | Michael Davis  | michael@example.com   | password123 |
|           8 | Emma Wilson    | emma@example.com      | password123 |
|           9 | William Taylor | william@example.com   | password123 |
|          10 | Olivia Adams   | olivia@example.com    | password123 |
+-------------+----------------+-----------------------+-------------+
10 rows in set (0.01 sec)

mysql> select*from orders;
+----------+-------------+------------+-------------+-------------------------+
| order_id | customer_id | order_date | total_price | shipping_address        |
+----------+-------------+------------+-------------+-------------------------+
|        1 |           1 | 2023-01-05 |     1200.00 | 123 Main St, City       |
|        2 |           2 | 2023-02-10 |      900.00 | 456 Elm St, Town        |
|        3 |           3 | 2023-03-15 |      300.00 | 789 Oak St, Village     |
|        4 |           4 | 2023-04-20 |      150.00 | 101 Pine St, Suburb     |
|        5 |           5 | 2023-05-25 |     1800.00 | 234 Cedar St, District  |
|        6 |           6 | 2023-06-30 |      400.00 | 567 Birch St, County    |
|        7 |           7 | 2023-07-05 |      700.00 | 890 Maple St, State     |
|        8 |           8 | 2023-08-10 |      160.00 | 321 Redwood St, Country |
|        9 |           9 | 2023-09-15 |      140.00 | 432 Spruce St, Province |
|       10 |          10 | 2023-10-20 |     1400.00 | 765 Fir St, Territory   |
+----------+-------------+------------+-------------+-------------------------+
10 rows in set (0.00 sec)
```

9. Determine the Minimum Stock Quantity for Each Product Category



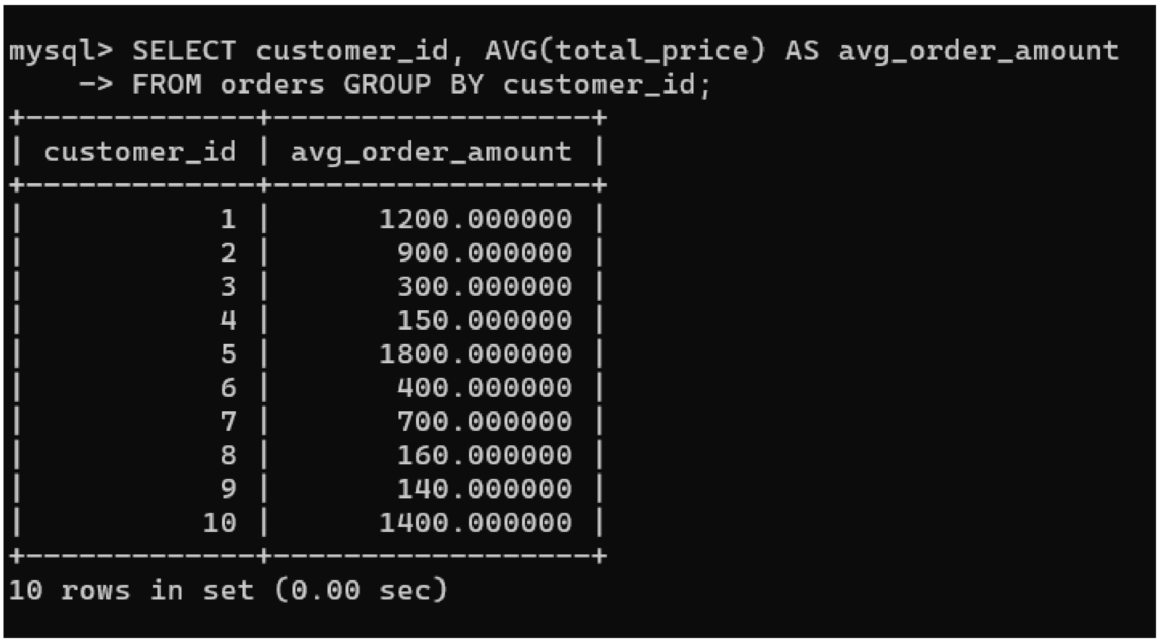
```
mysql> SELECT MIN(stockQuantity) AS min_stock FROM products;
+-----------+
| min_stock |
+-----------+
|         5 |
+-----------+
1 row in set (0.01 sec)
```

10. Calculate the Total Amount Spent by Each Customer



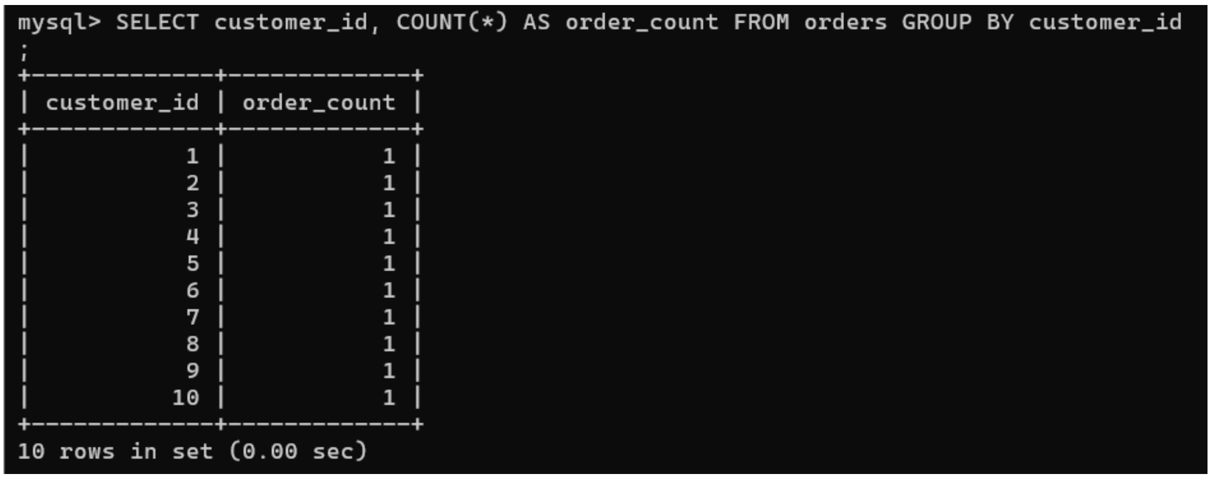
```
mysql> SELECT customer_id, SUM(total_price) AS total_spent
    -> FROM orders
    -> GROUP BY customer_id;
+-------------+-------------+
| customer_id | total_spent |
+-------------+-------------+
|           1 |     1200.00 |
|           2 |      900.00 |
|           3 |      300.00 |
|           4 |      150.00 |
|           5 |     1800.00 |
|           6 |      400.00 |
|           7 |      700.00 |
|           8 |      160.00 |
|           9 |      140.00 |
|          10 |     1400.00 |
+-------------+-------------+
10 rows in set (0.00 sec)
```

11. Find the Average Order Amount for Each Customer



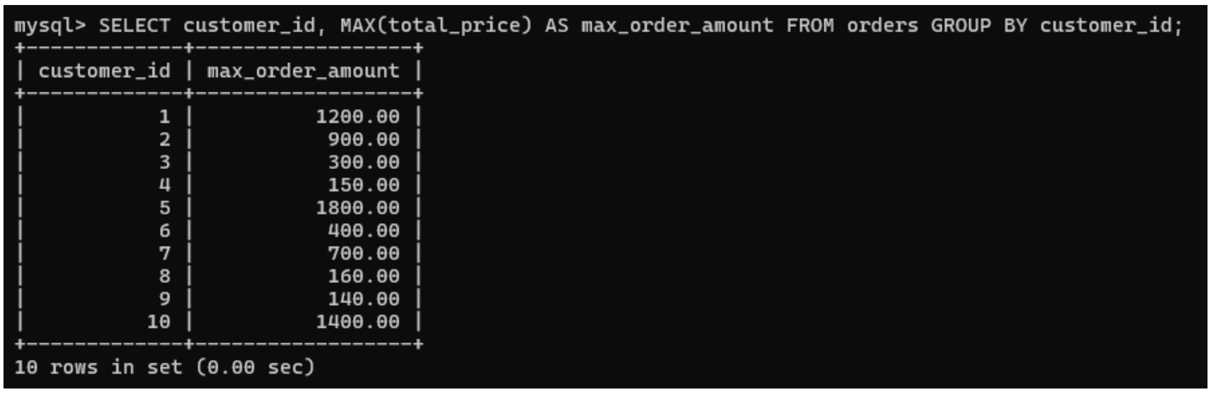
```
mysql> SELECT customer_id, AVG(total_price) AS avg_order_amount
    -> FROM orders GROUP BY customer_id;
+-------------+------------------+
| customer_id | avg_order_amount |
+-------------+------------------+
|           1 |      1200.000000 |
|           2 |       900.000000 |
|           3 |       300.000000 |
|           4 |       150.000000 |
|           5 |      1800.000000 |
|           6 |       400.000000 |
|           7 |       700.000000 |
|           8 |       160.000000 |
|           9 |       140.000000 |
|          10 |      1400.000000 |
+-------------+------------------+
10 rows in set (0.00 sec)
```

12. Count the Number of Orders Placed by Each Customer



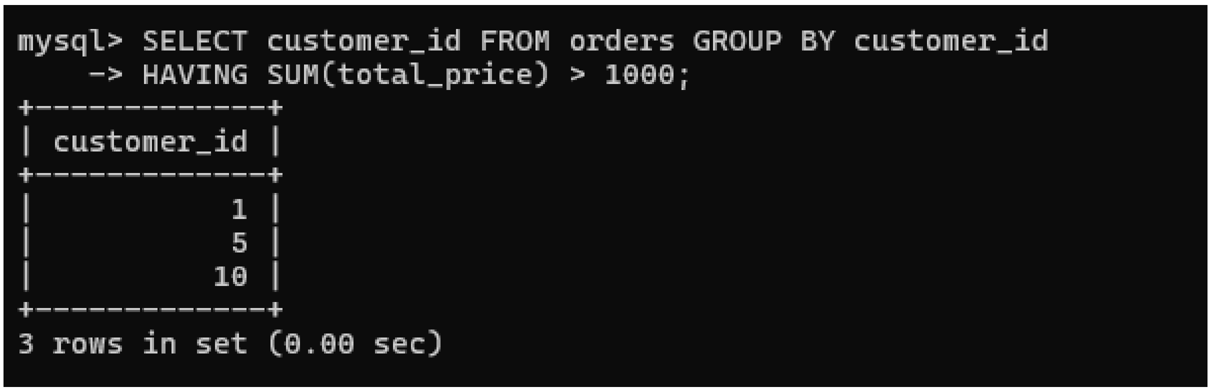
```
mysql> SELECT customer_id, COUNT(*) AS order_count FROM orders GROUP BY customer_id
;
+-------------+-------------+
| customer_id | order_count |
+-------------+-------------+
|           1 |           1 |
|           2 |           1 |
|           3 |           1 |
|           4 |           1 |
|           5 |           1 |
|           6 |           1 |
|           7 |           1 |
|           8 |           1 |
|           9 |           1 |
|          10 |           1 |
+-------------+-------------+
10 rows in set (0.00 sec)
```

13. Find the Maximum Order Amount for Each Customer



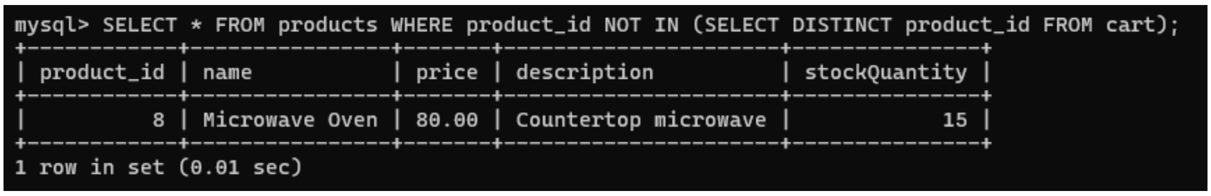
```
mysql> SELECT customer_id, MAX(total_price) AS max_order_amount FROM orders GROUP BY customer_id;
+-------------+------------------+
| customer_id | max_order_amount |
+-------------+------------------+
|           1 |          1200.00 |
|           2 |           900.00 |
|           3 |           300.00 |
|           4 |           150.00 |
|           5 |          1800.00 |
|           6 |           400.00 |
|           7 |           700.00 |
|           8 |           160.00 |
|           9 |           140.00 |
|          10 |          1400.00 |
+-------------+------------------+
10 rows in set (0.00 sec)
```

14. Get Customers Who Placed Orders Totaling Over 1000



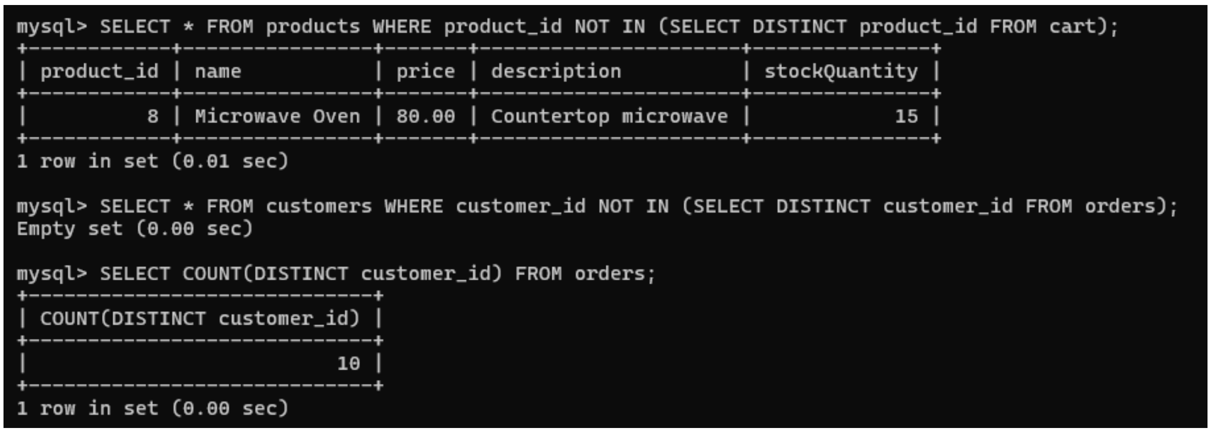
```
mysql> SELECT customer_id FROM orders GROUP BY customer_id
    -> HAVING SUM(total_price) > 1000;
+-------------+
| customer_id |
+-------------+
|           1 |
|           5 |
|          10 |
+-------------+
3 rows in set (0.00 sec)
```

15. Subquery to Find Products Not in the Cart



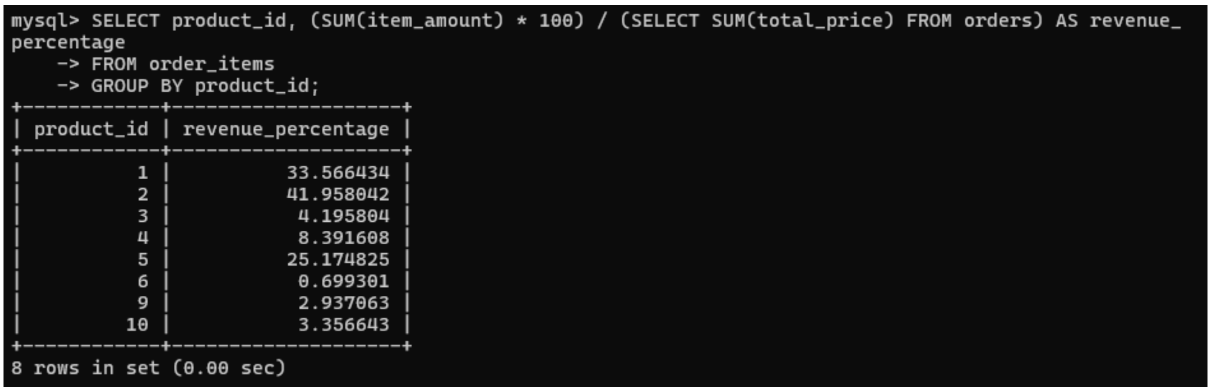
```
mysql> SELECT * FROM products WHERE product_id NOT IN (SELECT DISTINCT product_id FROM cart);
+------------+----------------+-------+----------------------+---------------+
| product_id | name           | price | description          | stockQuantity |
+------------+----------------+-------+----------------------+---------------+
|          8 | Microwave Oven | 80.00 | Countertop microwave |            15 |
+------------+----------------+-------+----------------------+---------------+
1 row in set (0.01 sec)
```

16. Subquery to Find Customers Who Haven't Placed Orders



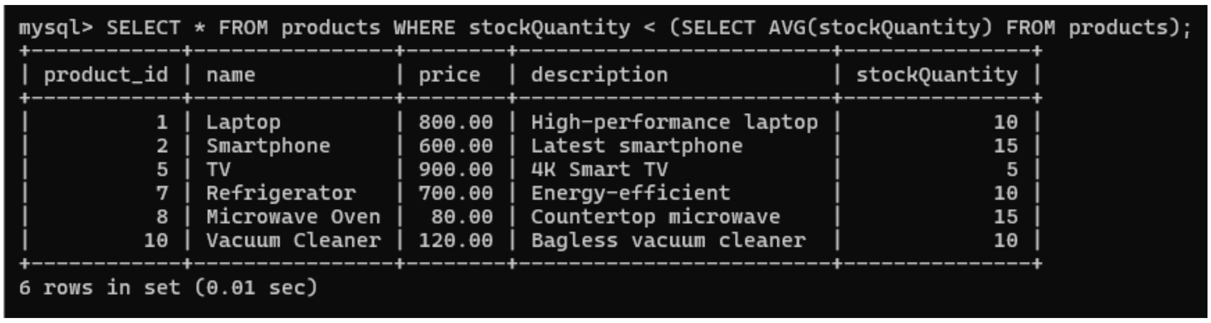
```
mysql> SELECT * FROM products WHERE product_id NOT IN (SELECT DISTINCT product_id FROM cart);
+------------+----------------+-------+----------------------+---------------+
| product_id | name           | price | description          | stockQuantity |
+------------+----------------+-------+----------------------+---------------+
|          8 | Microwave Oven | 80.00 | Countertop microwave |            15 |
+------------+----------------+-------+----------------------+---------------+
1 row in set (0.01 sec)

mysql> SELECT * FROM customers WHERE customer_id NOT IN (SELECT DISTINCT customer_id FROM orders);
Empty set (0.00 sec)

mysql> SELECT COUNT(DISTINCT customer_id) FROM orders;
+-----------------------------+
| COUNT(DISTINCT customer_id) |
+-----------------------------+
|                          10 |
+-----------------------------+
1 row in set (0.00 sec)
```

17. Subquery to Calculate the Percentage of Total Revenue for a Product



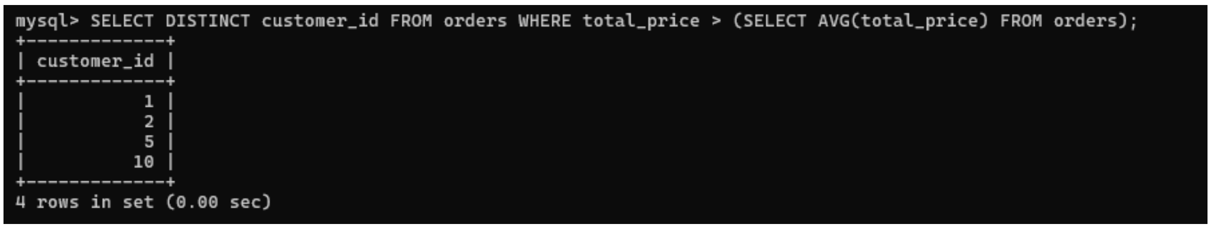
```
mysql> SELECT product_id, (SUM(item_amount) * 100) / (SELECT SUM(total_price) FROM orders) AS revenue_
percentage
    -> FROM order_items
    -> GROUP BY product_id;
+------------+--------------------+
| product_id | revenue_percentage |
+------------+--------------------+
|          1 |          33.566434 |
|          2 |          41.958042 |
|          3 |           4.195804 |
|          4 |           8.391608 |
|          5 |          25.174825 |
|          6 |           0.699301 |
|          9 |           2.937063 |
|         10 |           3.356643 |
+------------+--------------------+
8 rows in set (0.00 sec)
```

18. Subquery to Find Products with Low Stock



```
mysql> SELECT * FROM products WHERE stockQuantity < (SELECT AVG(stockQuantity) FROM products);
+------------+-----------------+--------+-------------------------+---------------+
| product_id | name            | price  | description             | stockQuantity |
+------------+-----------------+--------+-------------------------+---------------+
|          1 | Laptop          | 800.00 | High-performance laptop |            10 |
|          2 | Smartphone      | 600.00 | Latest smartphone       |            15 |
|          5 | TV              | 900.00 | 4K Smart TV             |             5 |
|          7 | Refrigerator    | 700.00 | Energy-efficient        |            10 |
|          8 | Microwave Oven  |  80.00 | Countertop microwave    |            15 |
|         10 | Vacuum Cleaner  | 120.00 | Bagless vacuum cleaner  |            10 |
+------------+-----------------+--------+-------------------------+---------------+
6 rows in set (0.01 sec)
```

19. Subquery to Find Customers Who Placed High-Value Orders



```
mysql> SELECT DISTINCT customer_id FROM orders WHERE total_price > (SELECT AVG(total_price) FROM orders);
+-------------+
| customer_id |
+-------------+
|           1 |
|           2 |
|           5 |
|          10 |
+-------------+
4 rows in set (0.00 sec)
```