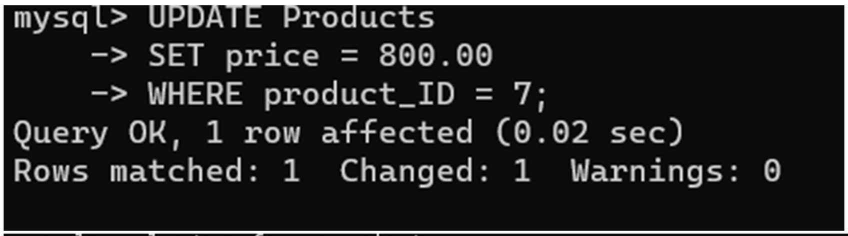
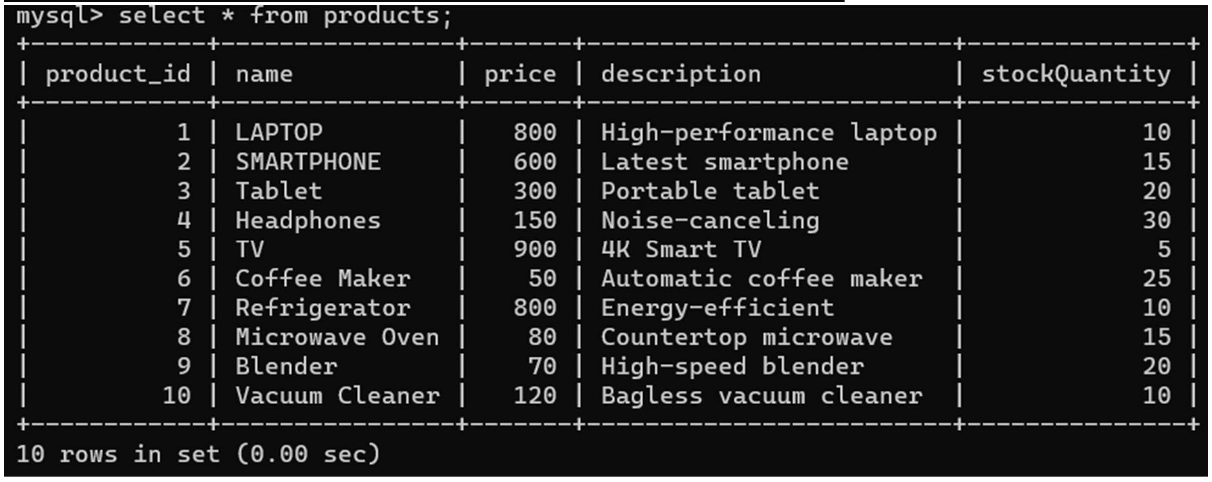
ASSESSMENT Ecommerce – SQL

1. Update refrigerator product price to 800.

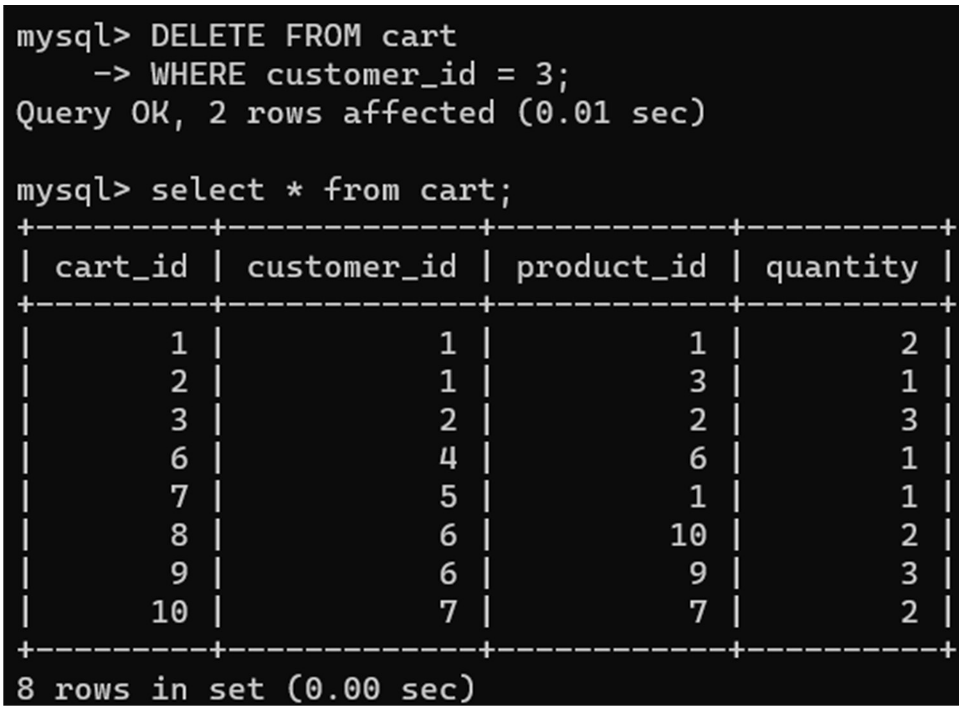


```
mysql> UPDATE Products
    -> SET price = 800.00
    -> WHERE product_ID = 7;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```



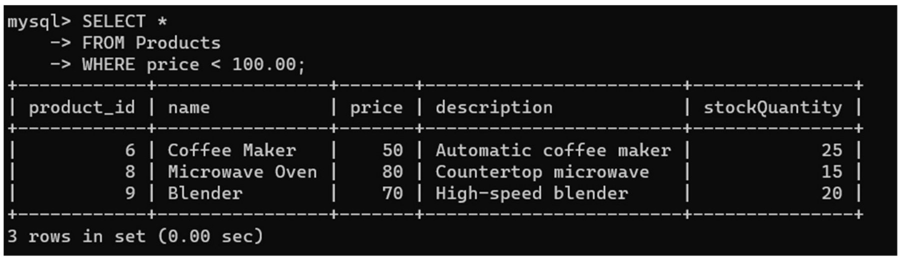
```
mysql> select * from products;
+------------+----------------+-------+------------------------+---------------+
| product_id | name           | price | description            | stockQuantity |
+------------+----------------+-------+------------------------+---------------+
|          1 | LAPTOP         |   800 | High-performance laptop |           10 |
|          2 | SMARTPHONE     |   600 | Latest smartphone      |            15 |
|          3 | Tablet         |   300 | Portable tablet        |            20 |
|          4 | Headphones     |   150 | Noise-canceling        |            30 |
|          5 | TV             |   900 | 4K Smart TV            |             5 |
|          6 | Coffee Maker   |    50 | Automatic coffee maker |            25 |
|          7 | Refrigerator   |   800 | Energy-efficient       |            10 |
|          8 | Microwave Oven |    80 | Countertop microwave   |            15 |
|          9 | Blender        |    70 | High-speed blender     |            20 |
|         10 | Vacuum Cleaner |   120 | Bagless vacuum cleaner |            10 |
+------------+----------------+-------+------------------------+---------------+
10 rows in set (0.00 sec)
```

2. Remove all cart items for a specific customer.



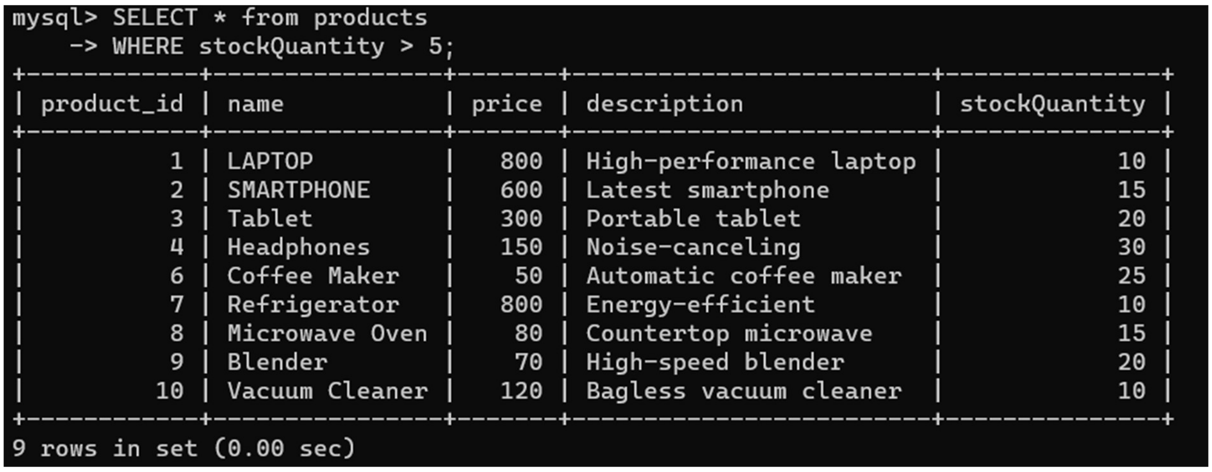
```
mysql> DELETE FROM cart
    -> WHERE customer_id = 3;
Query OK, 2 rows affected (0.01 sec)

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

3. Retrieve Products Priced Below $100.



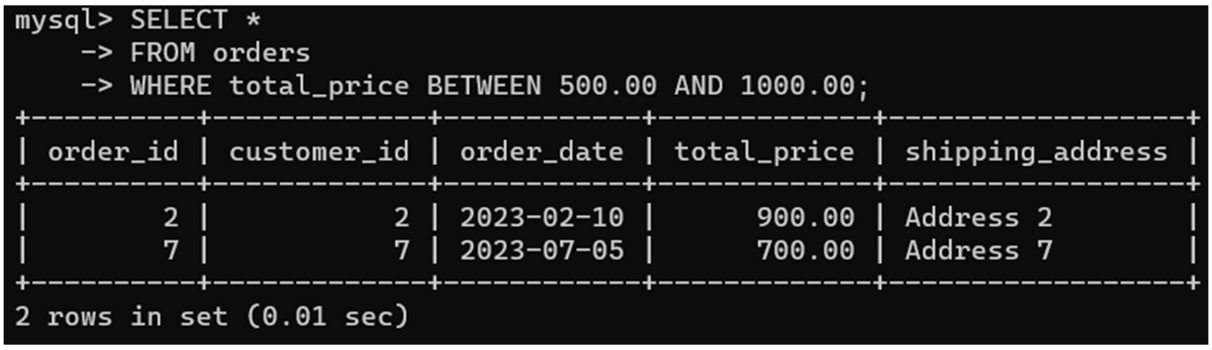
```
mysql> SELECT *
    -> FROM Products
    -> WHERE price < 100.00;
+------------+----------------+-------+------------------------+---------------+
| product_id | name           | price | description            | stockQuantity |
+------------+----------------+-------+------------------------+---------------+
|          6 | Coffee Maker   |    50 | Automatic coffee maker |            25 |
|          8 | Microwave Oven |    80 | Countertop microwave   |            15 |
|          9 | Blender        |    70 | High-speed blender     |            20 |
+------------+----------------+-------+------------------------+---------------+
3 rows in set (0.00 sec)
```

4. Find Products with Stock Quantity Greater Than 5.



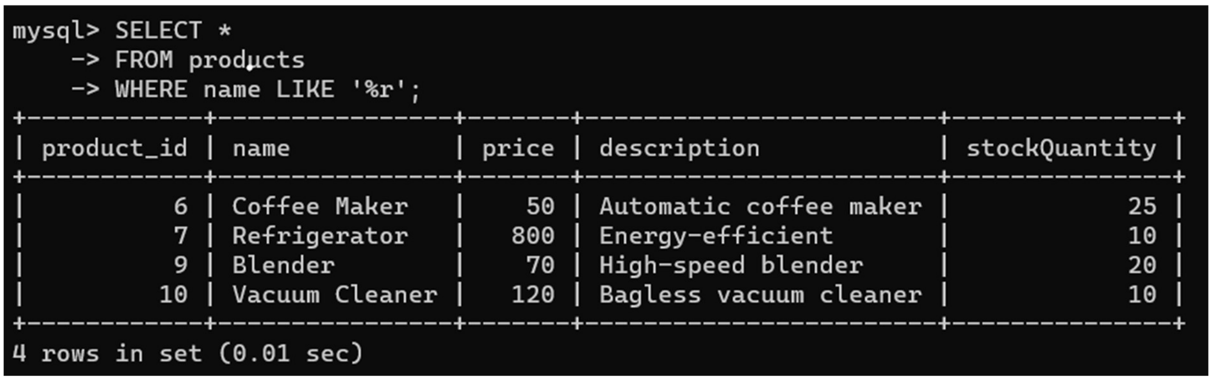
```
mysql> SELECT * from products
    -> WHERE stockQuantity > 5;
+------------+----------------+-------+-------------------------+---------------+
| product_id | name           | price | description             | stockQuantity |
+------------+----------------+-------+-------------------------+---------------+
|          1 | LAPTOP         |   800 | High-performance laptop |            10 |
|          2 | SMARTPHONE     |   600 | Latest smartphone       |            15 |
|          3 | Tablet         |   300 | Portable tablet         |            20 |
|          4 | Headphones     |   150 | Noise-canceling         |            30 |
|          6 | Coffee Maker   |    50 | Automatic coffee maker  |            25 |
|          7 | Refrigerator   |   800 | Energy-efficient        |            10 |
|          8 | Microwave Oven |    80 | Countertop microwave    |            15 |
|          9 | Blender        |    70 | High-speed blender      |            20 |
|         10 | Vacuum Cleaner |   120 | Bagless vacuum cleaner  |            10 |
+------------+----------------+-------+-------------------------+---------------+
9 rows in set (0.00 sec)
```

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



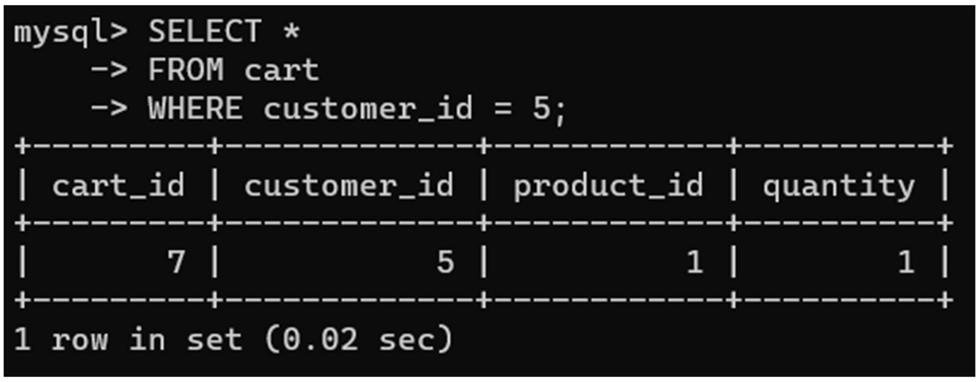
```
mysql> SELECT *
    -> FROM orders
    -> WHERE total_price BETWEEN 500.00 AND 1000.00;
+----------+-------------+------------+-------------+------------------+
| order_id | customer_id | order_date | total_price | shipping_address |
+----------+-------------+------------+-------------+------------------+
|        2 |           2 | 2023-02-10 |      900.00 | Address 2        |
|        7 |           7 | 2023-07-05 |      700.00 | Address 7        |
+----------+-------------+------------+-------------+------------------+
2 rows in set (0.01 sec)
```

6. Find Products which name end with letter ‘r’.



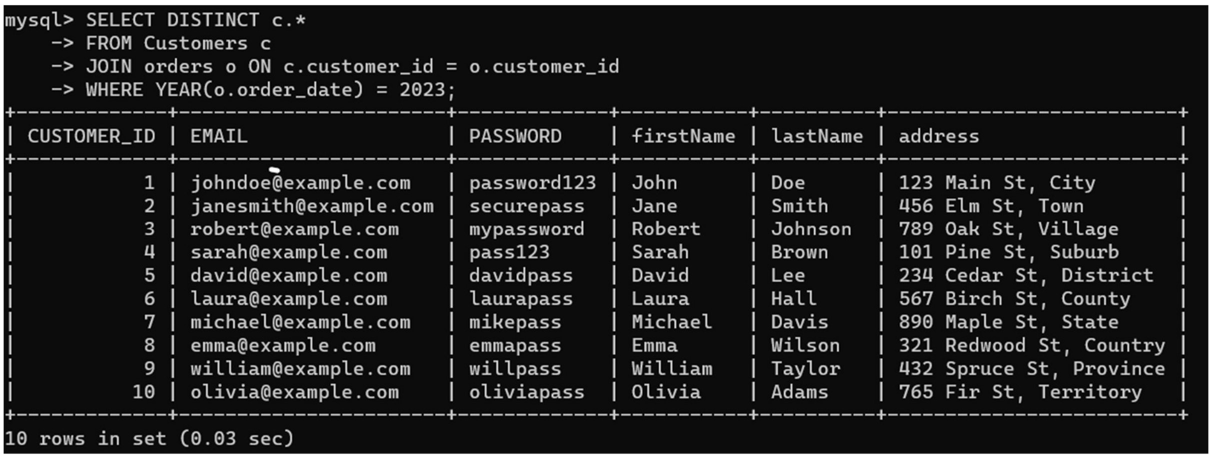
```
mysql> SELECT *
    -> FROM products
    -> WHERE name LIKE '%r';
+------------+----------------+-------+------------------------+---------------+
| product_id | name           | price | description            | stockQuantity |
+------------+----------------+-------+------------------------+---------------+
|          6 | Coffee Maker   |    50 | Automatic coffee maker |            25 |
|          7 | Refrigerator   |   800 | Energy-efficient       |            10 |
|          9 | Blender        |    70 | High-speed blender     |            20 |
|         10 | Vacuum Cleaner |   120 | Bagless vacuum cleaner |            10 |
+------------+----------------+-------+------------------------+---------------+
4 rows in set (0.01 sec)
```

7. Retrieve Cart Items for Customer 5.



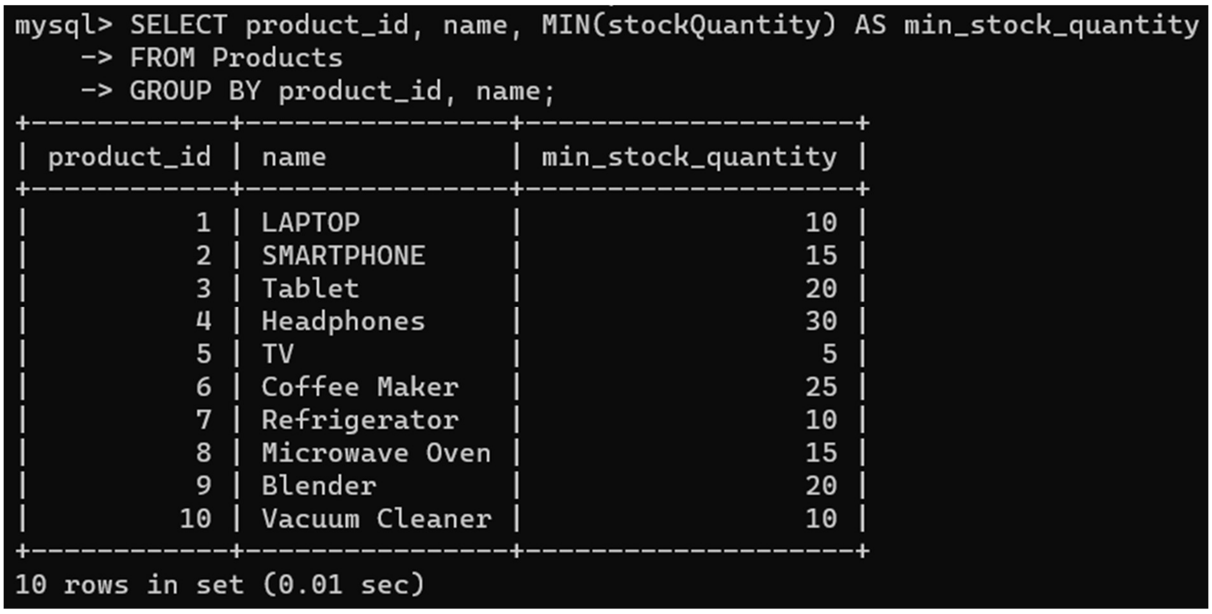
```
mysql> SELECT *
    -> FROM cart
    -> WHERE customer_id = 5;
+---------+-------------+------------+----------+
| cart_id | customer_id | product_id | quantity |
+---------+-------------+------------+----------+
|       7 |           5 |          1 |        1 |
+---------+-------------+------------+----------+
1 row in set (0.02 sec)
```

8. Find Customers Who Placed Orders in 2023.



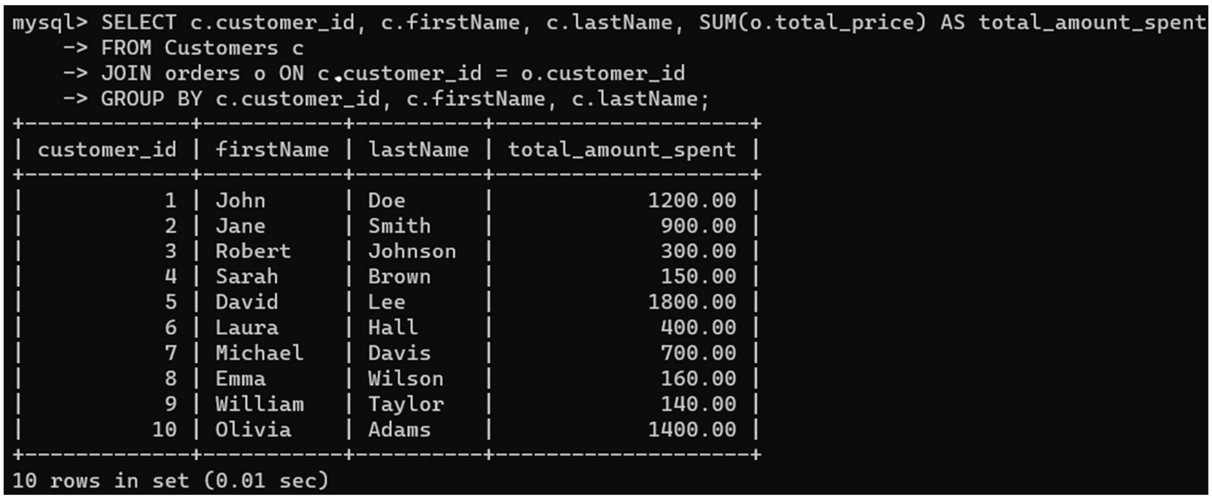
```
mysql> SELECT DISTINCT c.*
    -> FROM Customers c
    -> JOIN orders o ON c.customer_id = o.customer_id
    -> WHERE YEAR(o.order_date) = 2023;
+-------------+----------------------+-------------+-----------+----------+------------------------+
| CUSTOMER_ID | EMAIL                | PASSWORD    | firstName | lastName | address                |
+-------------+----------------------+-------------+-----------+----------+------------------------+
|           1 | johndoe@example.com  | password123 | John      | Doe      | 123 Main St, City      |
|           2 | janesmith@example.com | securepass  | Jane      | Smith    | 456 Elm St, Town       |
|           3 | robert@example.com   | mypassword  | Robert    | Johnson  | 789 Oak St, Village    |
|           4 | sarah@example.com    | pass123     | Sarah     | Brown    | 101 Pine St, Suburb    |
|           5 | david@example.com    | davidpass   | David     | Lee      | 234 Cedar St, District |
|           6 | laura@example.com    | laurapass   | Laura     | Hall     | 567 Birch St, County   |
|           7 | michael@example.com  | mikepass    | Michael   | Davis    | 890 Maple St, State    |
|           8 | emma@example.com     | emmapass    | Emma      | Wilson   | 321 Redwood St, Country |
|           9 | william@example.com  | willpass    | William   | Taylor   | 432 Spruce St, Province |
|          10 | olivia@example.com   | oliviapass  | Olivia    | Adams    | 765 Fir St, Territory  |
+-------------+----------------------+-------------+-----------+----------+------------------------+
10 rows in set (0.03 sec)
```

9. Determine the Minimum Stock Quantity for Each Product Category.



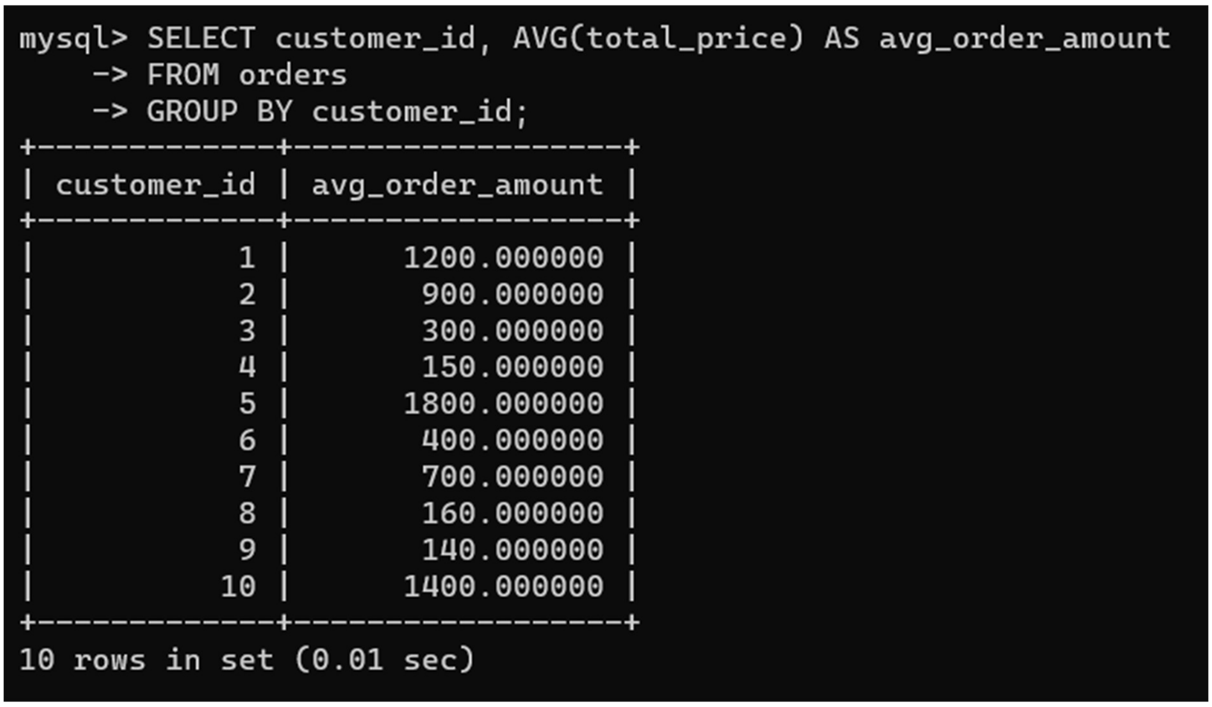
```
mysql> SELECT product_id, name, MIN(stockQuantity) AS min_stock_quantity
    -> FROM Products
    -> GROUP BY product_id, name;
+------------+----------------+--------------------+
| product_id | name           | min_stock_quantity |
+------------+----------------+--------------------+
|          1 | LAPTOP         |                 10 |
|          2 | SMARTPHONE     |                 15 |
|          3 | Tablet         |                 20 |
|          4 | Headphones     |                 30 |
|          5 | TV             |                  5 |
|          6 | Coffee Maker   |                 25 |
|          7 | Refrigerator   |                 10 |
|          8 | Microwave Oven |                 15 |
|          9 | Blender        |                 20 |
|         10 | Vacuum Cleaner |                 10 |
+------------+----------------+--------------------+
10 rows in set (0.01 sec)
```

10. Calculate the Total Amount Spent by Each Customer.



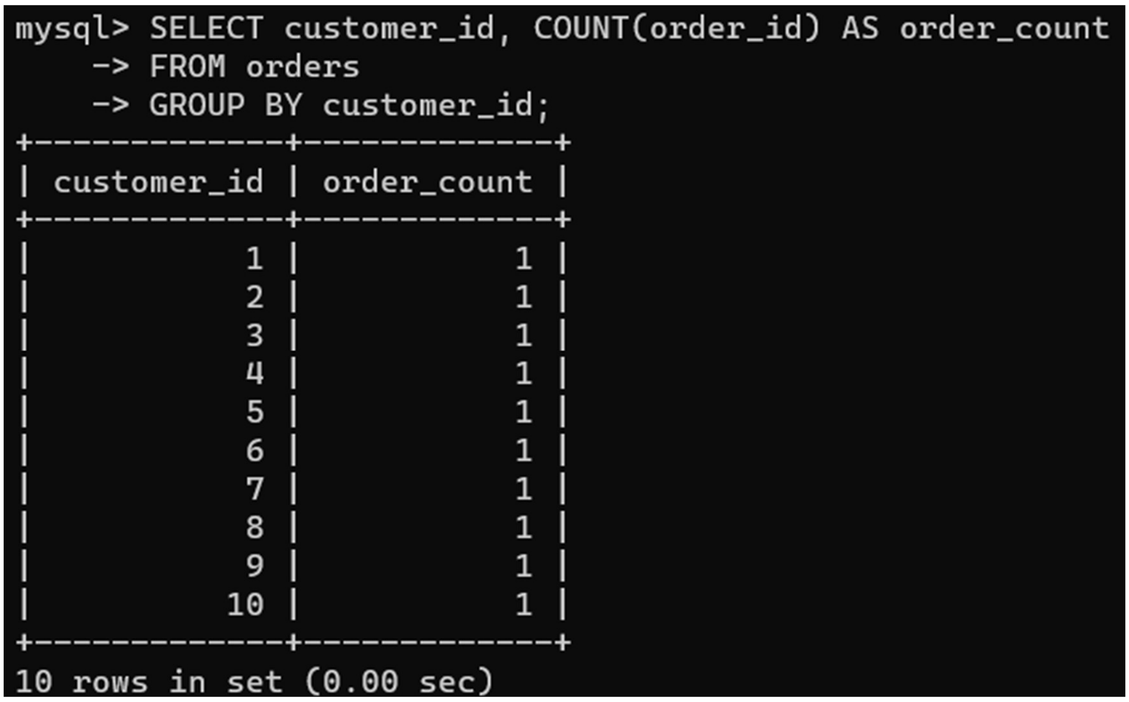
```
mysql> SELECT c.customer_id, c.firstName, c.lastName, SUM(o.total_price) AS total_amount_spent
    -> FROM Customers c
    -> JOIN orders o ON c.customer_id = o.customer_id
    -> GROUP BY c.customer_id, c.firstName, c.lastName;
+-------------+-----------+----------+--------------------+
| customer_id | firstName | lastName | total_amount_spent |
+-------------+-----------+----------+--------------------+
|           1 | John      | Doe      |            1200.00 |
|           2 | Jane      | Smith    |             900.00 |
|           3 | Robert    | Johnson  |             300.00 |
|           4 | Sarah     | Brown    |             150.00 |
|           5 | David     | Lee      |            1800.00 |
|           6 | Laura     | Hall     |             400.00 |
|           7 | Michael   | Davis    |             700.00 |
|           8 | Emma      | Wilson   |             160.00 |
|           9 | William   | Taylor   |             140.00 |
|          10 | Olivia    | Adams    |            1400.00 |
+-------------+-----------+----------+--------------------+
10 rows in set (0.01 sec)
```

11. Find the Average Order Amount for Each Customer.



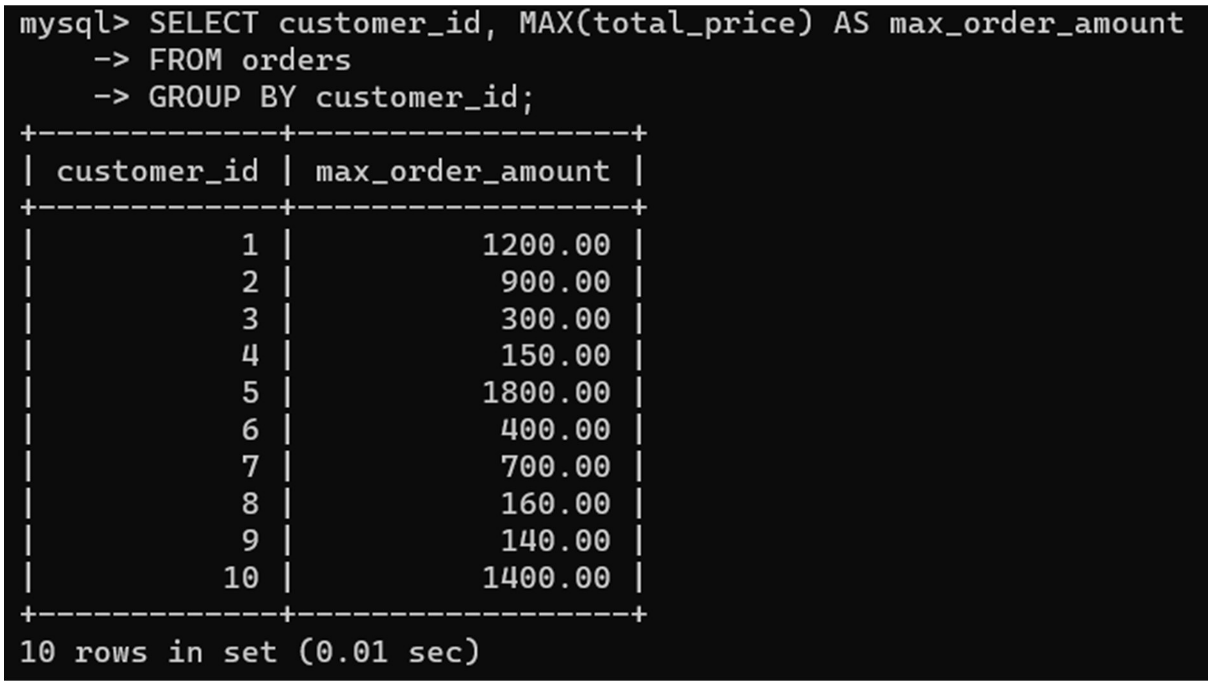
```
mysql> SELECT customer_id, AVG(total_price) AS avg_order_amount
    -> FROM orders
    -> GROUP BY customer_id;
+-------------+------------------+
| customer_id | avg_order_amount |
+-------------+------------------+
|           1 |      1200.000000 |
|           2 |       900.000000 |
|           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 (0.01 sec)
```

12. Count the Number of Orders Placed by Each Customer.



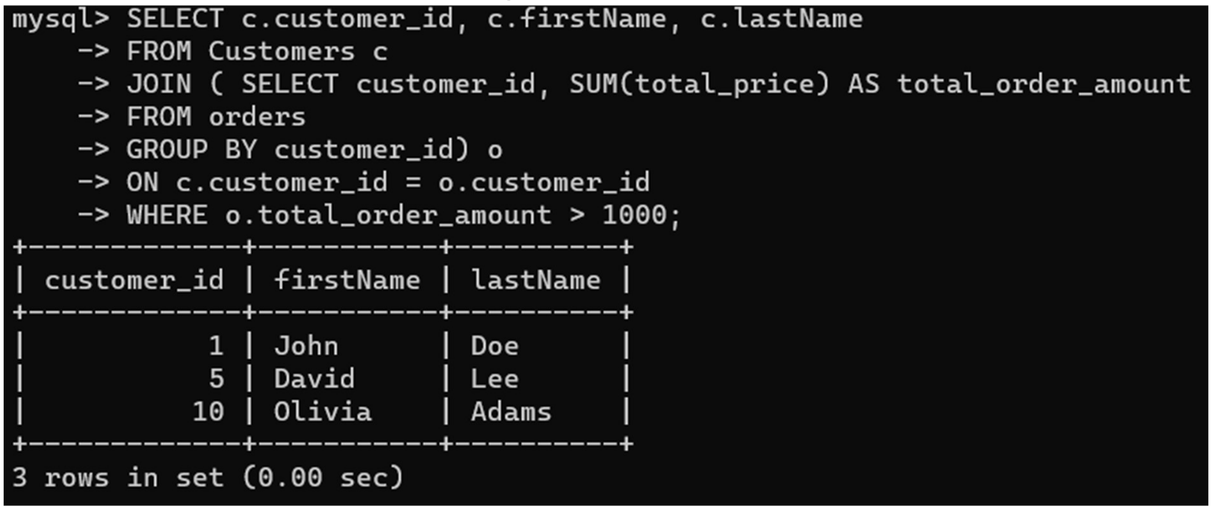
```
mysql> SELECT customer_id, COUNT(order_id) AS order_count
    -> FROM orders
    -> GROUP BY customer_id;
+-------------+-------------+
| customer_id | order_count |
+-------------+-------------+
|           1 |           1 |
|           2 |           1 |
|           3 |           1 |
|           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.



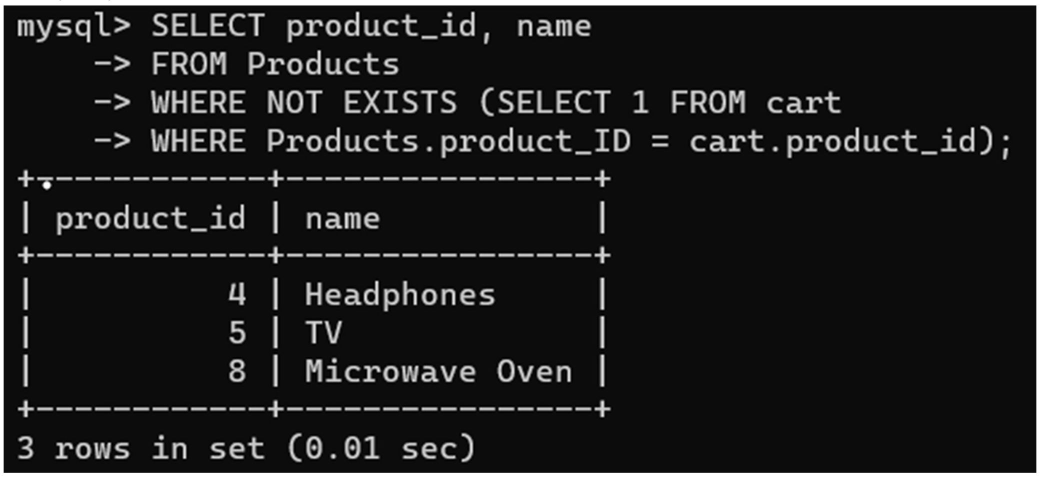
```
mysql> SELECT customer_id, MAX(total_price) AS max_order_amount
    -> FROM orders
    -> GROUP BY customer_id;
+-------------+------------------+
| customer_id | max_order_amount |
+-------------+------------------+
|           1 |          1200.00 |
|           2 |           900.00 |
|           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 (0.01 sec)
```

14. Get Customers Who Placed Orders Totaling Over $1000.



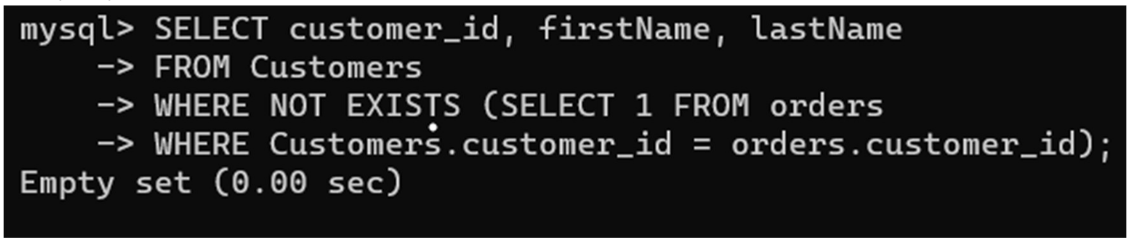
```
mysql> SELECT c.customer_id, c.firstName, c.lastName
    -> FROM Customers c
    -> JOIN ( SELECT customer_id, SUM(total_price) AS total_order_amount
    -> FROM orders
    -> GROUP BY customer_id) o
    -> ON c.customer_id = o.customer_id
    -> WHERE o.total_order_amount > 1000;
+-------------+-----------+----------+
| customer_id | firstName | lastName |
+-------------+-----------+----------+
|           1 | John      | Doe      |
|           5 | David     | Lee      |
|          10 | Olivia    | Adams    |
+-------------+-----------+----------+
3 rows in set (0.00 sec)
```

15. Subquery to Find Products Not in the Cart.



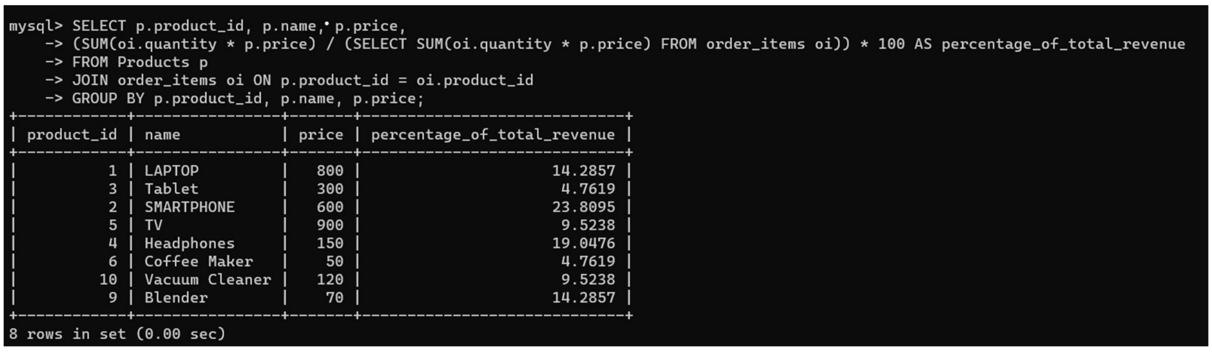
```
mysql> SELECT product_id, name
    -> FROM Products
    -> WHERE NOT EXISTS (SELECT 1 FROM cart
    -> WHERE Products.product_ID = cart.product_id);
+------------+----------------+
| product_id | name           |
+------------+----------------+
|          4 | Headphones     |
|          5 | TV             |
|          8 | Microwave Oven |
+------------+----------------+
3 rows in set (0.01 sec)
```

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



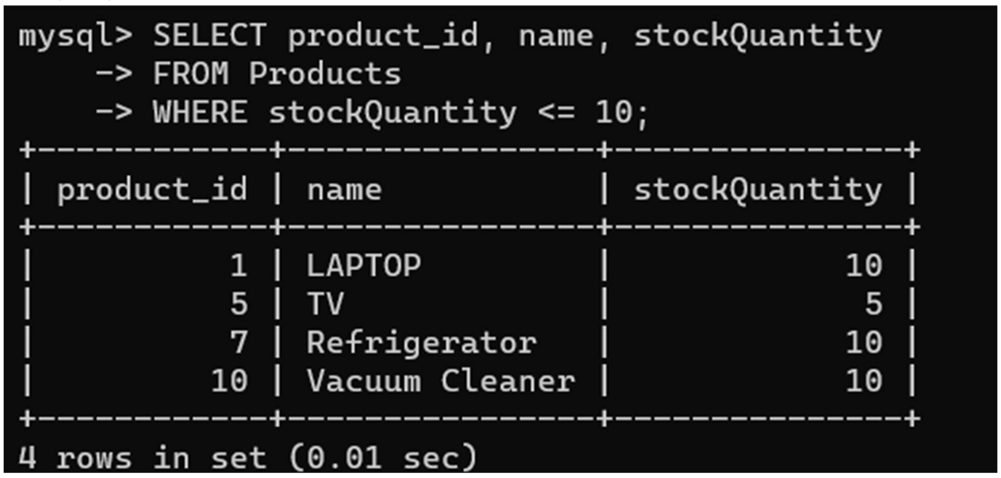
```
mysql> SELECT customer_id, firstName, lastName
    -> FROM Customers
    -> WHERE NOT EXISTS (SELECT 1 FROM orders
    -> WHERE Customers.customer_id = orders.customer_id);
Empty set (0.00 sec)
```

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



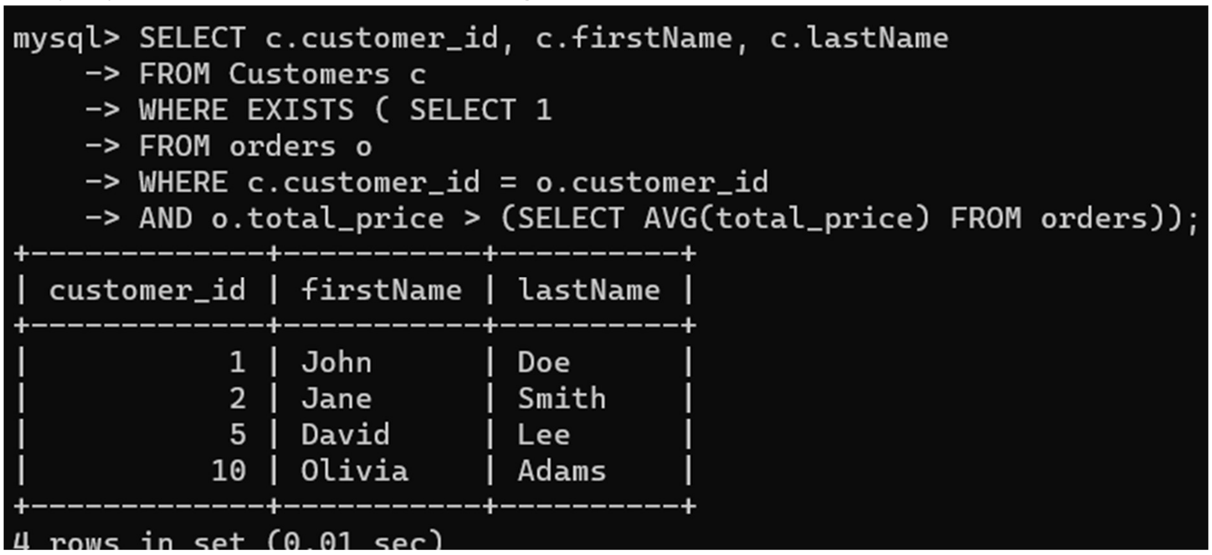
```
mysql> SELECT p.product_id, p.name, p.price,
    -> (SUM(oi.quantity * p.price) / (SELECT SUM(oi.quantity * p.price) FROM order_items oi)) * 100 AS percentage_of_total_revenue
    -> FROM Products p
    -> JOIN order_items oi ON p.product_id = oi.product_id
    -> GROUP BY p.product_id, p.name, p.price;
+------------+----------------+-------+-----------------------------+
| product_id | name           | price | percentage_of_total_revenue |
+------------+----------------+-------+-----------------------------+
|          1 | LAPTOP         |   800 |                     14.2857 |
|          3 | Tablet         |   300 |                      4.7619 |
|          2 | SMARTPHONE     |   600 |                     23.8095 |
|          5 | TV             |   900 |                      9.5238 |
|          4 | Headphones     |   150 |                     19.0476 |
|          6 | Coffee Maker   |    50 |                      4.7619 |
|         10 | Vacuum Cleaner |   120 |                      9.5238 |
|          9 | Blender        |    70 |                     14.2857 |
+------------+----------------+-------+-----------------------------+
8 rows in set (0.00 sec)
```

18. Subquery to Find Products with Low Stock.



```
mysql> SELECT product_id, name, stockQuantity
    -> FROM Products
    -> WHERE stockQuantity <= 10;
+------------+----------------+---------------+
| product_id | name           | stockQuantity |
+------------+----------------+---------------+
|          1 | LAPTOP         |            10 |
|          5 | TV             |             5 |
|          7 | Refrigerator   |            10 |
|         10 | Vacuum Cleaner |            10 |
+------------+----------------+---------------+
4 rows in set (0.01 sec)
```

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



```
mysql> SELECT c.customer_id, c.firstName, c.lastName
    -> FROM Customers c
    -> WHERE EXISTS ( SELECT 1
    -> FROM orders o
    -> WHERE c.customer_id = o.customer_id
    -> AND o.total_price > (SELECT AVG(total_price) FROM orders));
+-------------+-----------+----------+
| customer_id | firstName | lastName |
+-------------+-----------+----------+
|           1 | John      | Doe      |
|           2 | Jane      | Smith    |
|           5 | David     | Lee      |
|          10 | Olivia    | Adams    |
+-------------+-----------+----------+
4 rows in set (0.01 sec)
```