CODING CHALLENGE 1 – MYSQL [ECOMMERCE]

SWATHI BASKARAN

SQL Tables:

1. **Customers table:**
   •customer_id (Primary Key)
   • name
   • email
   • address

2. **Products table:**
   • product_id (Primary Key)
   • name
   • description
   • price
   • stockQuantity

3. **Cart table:**
   • cart_id (Primary Key)
   • customer_id (Foreign Key)
   • product_id (Foreign Key)
   • quantity

4. **Orders table:**
   • order_id (Primary Key)
   • customer_id (Foreign Key)
   • order_date
   • totalAmount

5. **OrderItems table:**
   • order_item_id (Primary Key)
   • order_id (Foreign Key)
   • product_id (Foreign Key)
   • quantity
   • itemAmount

1. Update refrigerator product price to 800.

   Query:

   UPDATE Products SET Price = 800 WHERE Name = 'Refrigerator';

   Output:

```
mysql> SELECT * FROM Products;
+------------+----------------+--------------------------+-------+---------------+
| Product_id | Name           | Description              | Price | StockQuantity |
+------------+----------------+--------------------------+-------+---------------+
|          1 | Laptop         | High-performance laptop  |   800 |            10 |
|          2 | Smartphone     | Latest smartphone        |   600 |            15 |
|          3 | Tablet         | Portable tablet          |   300 |            20 |
|          4 | Headphones     | Noise-canceling          |   150 |            30 |
|          5 | TV             | 4K Smart TV              |   900 |             5 |
|          6 | Coffee Maker   | Automatic coffee maker   |    50 |            25 |
|          7 | Refrigerator   | Energy-efficient         |   700 |            10 |
|          8 | Microwave Oven | Countertop microwave     |    80 |            15 |
|          9 | Blender        | High-speed blender       |    70 |            20 |
|         10 | Vacuum Cleaner | Bagless vacuum cleaner   |   120 |            10 |
+------------+----------------+--------------------------+-------+---------------+
10 rows in set (0.00 sec)

mysql> UPDATE Products SET Price = 800 WHERE Name = 'Refrigerator';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM Products;
+------------+----------------+--------------------------+-------+---------------+
| Product_id | Name           | Description              | Price | StockQuantity |
+------------+----------------+--------------------------+-------+---------------+
|          1 | Laptop         | High-performance laptop  |   800 |            10 |
|          2 | Smartphone     | Latest smartphone        |   600 |            15 |
|          3 | Tablet         | Portable tablet          |   300 |            20 |
|          4 | Headphones     | Noise-canceling          |   150 |            30 |
|          5 | TV             | 4K Smart TV              |   900 |             5 |
|          6 | Coffee Maker   | Automatic coffee maker   |    50 |            25 |
|          7 | Refrigerator   | Energy-efficient         |   800 |            10 |
|          8 | Microwave Oven | Countertop microwave     |    80 |            15 |
|          9 | Blender        | High-speed blender       |    70 |            20 |
|         10 | Vacuum Cleaner | Bagless vacuum cleaner   |   120 |            10 |
+------------+----------------+--------------------------+-------+---------------+
10 rows in set (0.00 sec)
```

2. Remove all cart items for a specific customer.

   Query:

   SET @user_customerID = 1;

   DELETE FROM Cart WHERE customer_ID = @user_customerID;

   Output:

```
mysql> SELECT * FROM Cart;
+--------+-------------+------------+----------+
| CartID | 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)

mysql> SET @user_customerID = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM Cart WHERE customer_ID = @user_customerID;
Query OK, 2 rows affected (0.01 sec)

mysql> SELECT * FROM Cart;
+--------+-------------+------------+----------+
| CartID | Customer_ID | Product_ID | Quantity |
+--------+-------------+------------+----------+
|      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 |
+--------+-------------+------------+----------+
8 rows in set (0.00 sec)
```

3. Retrieve products priced below $100

Query:

SELECT * FROM Products WHERE Price < 100;

Output:

```
mysql> SELECT * FROM Products WHERE Price < 100;
+------------+----------------+-----------------------+-------+---------------+
| Product_id | Name           | Description           | Price | StockQuantity |
+------------+----------------+-----------------------+-------+---------------+
|          6 | Coffee Maker   | Automatic coffee maker |    50 |            25 |
|          8 | Microwave Oven | Countertop microwave   |    80 |            15 |
|          9 | Blender        | High-speed blender     |    70 |            20 |
+------------+----------------+-----------------------+-------+---------------+
3 rows in set (0.01 sec)
```

4. Find Products with Stock Quantity Greater Than 5.

Query:

SELECT * FROM Products WHERE StockQuantity > 5;

Output:

```
mysql> SELECT * FROM Products WHERE StockQuantity > 5;
+------------+---------------+-------------------------+-------+---------------+
| Product_id | Name          | Description             | Price | StockQuantity |
+------------+---------------+-------------------------+-------+---------------+
|          1 | Laptop        | High-performance laptop |   800 |            10 |
|          2 | Smartphone    | Latest smartphone       |   600 |            15 |
|          3 | Tablet        | Portable tablet         |   300 |            20 |
|          4 | Headphones    | Noise-canceling         |   150 |            30 |
|          6 | Coffee Maker  | Automatic coffee maker  |    50 |            25 |
|          7 | Refrigerator  | Energy-efficient        |   800 |            10 |
|          8 | Microwave Oven| Countertop microwave    |    80 |            15 |
|          9 | Blender       | High-speed blender      |    70 |            20 |
|         10 | Vacuum Cleaner| Bagless vacuum cleaner  |   120 |            10 |
+------------+---------------+-------------------------+-------+---------------+
9 rows in set (0.00 sec)
```

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

   Query:

   SELECT * FROM Orders WHERE TotalAmount BETWEEN 500 AND 1000;

   Output:

```
mysql> SELECT * FROM Orders WHERE TotalAmount BETWEEN 500 AND 1000;
+----------+-------------+------------+-------------+
| Order_ID | Customer_ID | Order_date | TotalAmount |
+----------+-------------+------------+-------------+
|        2 |           2 | 2023-02-10 |         900 |
|        7 |           7 | 2023-07-05 |         700 |
+----------+-------------+------------+-------------+
2 rows in set (0.01 sec)
```

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

   Query:

   SELECT * FROM Products WHERE Name LIKE '%r';

   Output:

```
mysql> SELECT * FROM Products WHERE Name LIKE '%r';
+------------+---------------+-------------------------+-------+---------------+
| Product_id | Name          | Description             | Price | StockQuantity |
+------------+---------------+-------------------------+-------+---------------+
|          6 | Coffee Maker  | Automatic coffee maker |    50 |            25 |
|          7 | Refrigerator  | Energy-efficient        |   800 |            10 |
|          9 | Blender       | High-speed blender      |    70 |            20 |
|         10 | Vacuum Cleaner| Bagless vacuum cleaner  |   120 |            10 |
+------------+---------------+-------------------------+-------+---------------+
4 rows in set (0.00 sec)
```

7. Retrieve Cart Items for Customer 5.

   Query:

   SELECT * FROM Cart WHERE Customer_ID = 5;

   Output:

```
mysql> SELECT * FROM Cart WHERE Customer_ID = 5;
+--------+-------------+------------+----------+
| CartID | Customer_ID | Product_ID | Quantity |
+--------+-------------+------------+----------+
|      7 |           5 |          1 |        1 |
+--------+-------------+------------+----------+
1 row in set (0.01 sec)
```

8. Find Customers Who Placed Orders in 2023.

   Query:

   SELECT

   Cus.*

   FROM Customers Cus

   RIGHT JOIN Orders Ord

   ON Cus.Customer_ID = Ord.Customer_ID

   WHERE YEAR(Ord.Order_Date) = 2023;

   Output:

```
mysql> -- Finding customers who placed orders in 2023
mysql> SELECT
    -> Cus.*
    -> FROM Customers Cus
    -> RIGHT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
    -> WHERE YEAR(Ord.Order_Date) = 2023;
+-------------+---------------+---------------------+-------------------------+
| customer_id | name          | email               | Address                 |
+-------------+---------------+---------------------+-------------------------+
|           1 | John Doe      | johndoe@example.com | 123 Main St, City       |
|           2 | Jane Smith    | janesmith@example.com | 456 Elm St, Town      |
|           3 | Robert Johnson | robert@example.com | 789 Oak St, Village     |
|           4 | Sarah Brown   | sarah@example.com   | 101 Pine St, Suburb     |
|           5 | David Lee     | david@example.com   | 234 Cedar St, District  |
|           6 | Laura Hall    | laura@example.com   | 567 Birch St, County    |
|           7 | Michael Davis | michael@example.com | 890 Maple St, State     |
|           8 | Emma Wilson   | emma@example.com    | 321 Redwood St, Country |
|           9 | William Taylor | william@example.com | 432 Spruce St, Province |
|          10 | Olivia Adams  | olivia@example.com  | 765 Fir St, Territory   |
+-------------+---------------+---------------------+-------------------------+
10 rows in set (0.01 sec)
```

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

   Query:

   -- Initially, adding a column in Products Table called category and categorizing the products as "Electronics" and "Appliances" accordingly.

   ALTER TABLE Products ADD COLUMN Category VARCHAR(50);

UPDATE products SET category = 'Electronics' WHERE product_ID IN (1, 2, 3, 4, 5);

UPDATE products SET category = 'Appliances' WHERE product_ID IN (6, 7, 8, 9, 10);

--Now finding the minimum stock quantity for each product category

SELECT Category, MIN(StockQuantity) AS "Mininum Stock Quantity" FROM Products GROUP BY Category;

Output:

```
mysql> ALTER TABLE Products ADD COLUMN Category VARCHAR(50);
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> UPDATE products SET category = 'Electronics' WHERE product_ID IN (1, 2, 3, 4, 5);
Query OK, 5 rows affected (0.01 sec)
Rows matched: 5  Changed: 5  Warnings: 0

mysql> UPDATE products SET category = 'Appliances' WHERE product_ID IN (6, 7, 8, 9, 10);
Query OK, 5 rows affected (0.01 sec)
Rows matched: 5  Changed: 5  Warnings: 0
```

```
mysql> SELECT * FROM Products;
+------------+----------------+--------------------------+-------+--------------+-------------+
| Product_id | Name           | Description              | Price | StockQuantity | Category    |
+------------+----------------+--------------------------+-------+--------------+-------------+
|          1 | Laptop         | High-performance laptop  |   800 |            10 | Electronics |
|          2 | Smartphone     | Latest smartphone        |   600 |            15 | Electronics |
|          3 | Tablet         | Portable tablet          |   300 |            20 | Electronics |
|          4 | Headphones     | Noise-canceling          |   150 |            30 | Electronics |
|          5 | TV             | 4K Smart TV              |   900 |             5 | Electronics |
|          6 | Coffee Maker   | Automatic coffee maker   |    50 |            25 | Appliances  |
|          7 | Refrigerator   | Energy-efficient         |   800 |            10 | Appliances  |
|          8 | Microwave Oven | Countertop microwave     |    80 |            15 | Appliances  |
|          9 | Blender        | High-speed blender       |    70 |            20 | Appliances  |
|         10 | Vacuum Cleaner | Bagless vacuum cleaner   |   120 |            10 | Appliances  |
+------------+----------------+--------------------------+-------+--------------+-------------+
10 rows in set (0.00 sec)

mysql> SELECT Category, MIN(StockQuantity) AS "Mininum Stock Quantity" FROM Products GROUP BY Category;
+-------------+-----------------------+
| Category    | Mininum Stock Quantity |
+-------------+-----------------------+
| Electronics |                     5 |
| Appliances  |                    10 |
+-------------+-----------------------+
2 rows in set (0.01 sec)
```

10. Calculate the Total Amount Spent by Each Customer.

Query:

SELECT

Cus.Name,

Cus.Customer_ID,

IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) AS "Total Amount Spent"

FROM Customers Cus

LEFT JOIN Orders Ord

ON Cus.Customer_ID = Ord.Customer_ID

GROUP BY Customer_ID;

Output:

```
mysql> -- Calculating the total amount spent by each customer
mysql> SELECT
    -> Cus.Name,
    -> Cus.Customer_ID,
    -> IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) AS "Total Amount Spent"
    -> FROM Customers Cus
    -> LEFT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
    -> GROUP BY Customer_ID;
+----------------+-------------+--------------------+
| Name           | Customer_ID | Total Amount Spent |
+----------------+-------------+--------------------+
| John Doe       |           1 |               1200 |
| Jane Smith     |           2 |                900 |
| Robert Johnson |           3 |                300 |
| Sarah Brown    |           4 |                150 |
| David Lee      |           5 |               1800 |
| Laura Hall     |           6 |                400 |
| Michael Davis  |           7 |                700 |
| Emma Wilson    |           8 |                160 |
| William Taylor |           9 |                140 |
| Olivia Adams   |          10 |               1400 |
+----------------+-------------+--------------------+
10 rows in set (0.00 sec)
```

11. Find the Average Order Amount for Each Customer.

Query:

SELECT

Cus.Name,

Cus.Customer_ID,

IFNULL(ROUND(AVG(Ord.TotalAmount),2),0) AS "Total Amount Spent"

FROM Customers Cus

LEFT JOIN Orders Ord

ON Cus.Customer_ID = Ord.Customer_ID

GROUP BY Customer_ID;

Output:

```
mysql> SELECT
    -> Cus.Name,
    -> Cus.Customer_ID,
    -> IFNULL(ROUND(AVG(Ord.TotalAmount),2),0) AS "Total Amount Spent"
    -> FROM Customers Cus
    -> LEFT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
    -> GROUP BY Customer_ID;
+---------------+-------------+--------------------+
| Name          | Customer_ID | Total Amount Spent |
+---------------+-------------+--------------------+
| John Doe      |           1 |               1200 |
| Jane Smith    |           2 |                900 |
| Robert Johnson |           3 |                300 |
| Sarah Brown   |           4 |                150 |
| David Lee     |           5 |               1800 |
| Laura Hall    |           6 |                400 |
| Michael Davis |           7 |                700 |
| Emma Wilson   |           8 |                160 |
| William Taylor |           9 |                140 |
| Olivia Adams  |          10 |               1400 |
+---------------+-------------+--------------------+
10 rows in set (0.00 sec)
```

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

Query:

```
SELECT
Cus.Name,
Cus.Customer_ID,
COUNT(Order_ID) AS "Number of Orders Placed"
FROM Customers Cus
LEFT JOIN Orders Ord
ON Cus.Customer_ID = Ord.Customer_ID
WHERE Cus.Customer_ID = Ord.Customer_ID
GROUP BY Customer_ID;
```

Output:

```
mysql> SELECT
    -> Cus.Name,
    -> Cus.Customer_ID,
    -> COUNT(Order_ID) AS "Number of Orders Placed"
    -> FROM Customers Cus
    -> LEFT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
    -> WHERE Cus.Customer_ID = Ord.Customer_ID
    -> GROUP BY Customer_ID;
+---------------+-------------+-------------------------+
| Name          | Customer_ID | Number of Orders Placed |
+---------------+-------------+-------------------------+
| John Doe      |           1 |                       1 |
| Jane Smith    |           2 |                       1 |
| Robert Johnson |           3 |                       1 |
| Sarah Brown   |           4 |                       1 |
| David Lee     |           5 |                       1 |
| Laura Hall    |           6 |                       1 |
| Michael Davis |           7 |                       1 |
| Emma Wilson   |           8 |                       1 |
| William Taylor |           9 |                       1 |
| Olivia Adams  |          10 |                       1 |
+---------------+-------------+-------------------------+
10 rows in set (0.00 sec)
```

13. Find the Maximum Order Amount for Each Customer.

Query:

SELECT

Cus.Name,

Cus.Customer_ID,

IFNULL(MAX(Ord.TotalAmount),0) AS "Maximum Order Amount"

FROM Customers Cus

LEFT JOIN Orders Ord

ON Cus.Customer_ID = Ord.Customer_ID

GROUP BY Customer_ID;

Output:

```
mysql> -- Maximum order amount for each customer
mysql> SELECT
    -> Cus.Name,
    -> Cus.Customer_ID,
    -> IFNULL(MAX(Ord.TotalAmount),0) AS "Maximum Order Amount"
    -> FROM Customers Cus
    -> LEFT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
    -> GROUP BY Customer_ID;
+---------------+-------------+----------------------+
| Name          | Customer_ID | Maximum Order Amount |
+---------------+-------------+----------------------+
| John Doe      |           1 |                 1200 |
| Jane Smith    |           2 |                  900 |
| Robert Johnson |           3 |                  300 |
| Sarah Brown   |           4 |                  150 |
| David Lee     |           5 |                 1800 |
| Laura Hall    |           6 |                  400 |
| Michael Davis |           7 |                  700 |
| Emma Wilson   |           8 |                  160 |
| William Taylor |           9 |                  140 |
| Olivia Adams  |          10 |                 1400 |
+---------------+-------------+----------------------+
10 rows in set (0.00 sec)
```

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

Query:

SELECT

Cus.Name,

Cus.Customer_ID,

IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) AS "Total Order Amount"

FROM Customers Cus

LEFT JOIN Orders Ord

ON Cus.Customer_ID = Ord.Customer_ID

GROUP BY Cus.Customer_ID

HAVING IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) > 1000;

Output:

```
mysql> SELECT
    -> Cus.Name,
    -> Cus.Customer_ID,
    -> IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) AS "Total Order Amount"
    -> FROM Customers Cus
    -> LEFT JOIN Orders Ord
    -> ON Cus.Customer_ID = Ord.Customer_ID
    -> GROUP BY Cus.Customer_ID
    -> HAVING IFNULL(ROUND(SUM(Ord.TotalAmount),2),0) > 1000
    -> ;
+--------------+-------------+--------------------+
| Name         | Customer_ID | Total Order Amount |
+--------------+-------------+--------------------+
| John Doe     |           1 |               1200 |
| David Lee    |           5 |               1800 |
| Olivia Adams |          10 |               1400 |
+--------------+-------------+--------------------+
3 rows in set (0.00 sec)
```

15. Subquery to Find Products Not in the Cart.

Query:

```
SELECT *
FROM Products
WHERE Product_ID NOT IN
(
   SELECT Product_ID
   FROM CART
);
```

Output:

```
mysql> -- Finding products not in the cart
mysql> SELECT *
    -> FROM Products
    -> WHERE Product_ID NOT IN
    -> (
    ->     SELECT Product_ID
    ->     FROM CART
    -> );
+------------+----------------+----------------------+-------+---------------+-------------+
| Product_id | Name           | Description          | Price | StockQuantity | Category    |
+------------+----------------+----------------------+-------+---------------+-------------+
|          3 | Tablet         | Portable tablet      |   300 |            20 | Electronics |
|          8 | Microwave Oven | Countertop microwave |    80 |            15 | Appliances  |
+------------+----------------+----------------------+-------+---------------+-------------+
2 rows in set (0.00 sec)
```

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

Query:

```
SELECT *
FROM Customers
WHERE Customer_ID NOT IN
```

(

SELECT Customer_ID

FROM Orders

);

Output:

```
mysql> SELECT *
    -> FROM Customers
    -> WHERE Customer_ID NOT IN
    -> (
    ->     SELECT Customer_ID
    ->     FROM Orders
    -> );
+-------------+----------+-------------------+-------------------------+
| customer_id | name     | email             | Address                 |
+-------------+----------+-------------------+-------------------------+
|          11 | Rajmohan | rajmo@example.com | 236 Apple St, Woodlands |
+-------------+----------+-------------------+-------------------------+
1 row in set (0.00 sec)
```

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

Query:

SELECT

P.Product_ID,

P.Name,

IFNULL(ROUND(SUM(Oi.ItemAmount),2),0) AS "Product Revenue",

ROUND(IFNULL(SUM(Oi.ItemAmount),0) /

(SELECT SUM(ItemAmount) FROM OrderItems) * 100,2)

AS "Revenue Percentage"

FROM Products P

LEFT JOIN OrderItems Oi

ON P.Product_ID = Oi.Product_ID

GROUP BY P.Product_ID, P.Name;

Output:

```
mysql> SELECT
    -> P.Product_ID,
    -> P.Name,
    -> IFNULL(ROUND(SUM(Oi.ItemAmount),2),0) AS "Product Revenue",
    -> ROUND(IFNULL(SUM(Oi.ItemAmount),0) /
    -> (SELECT SUM(ItemAmount) FROM OrderItems) * 100,2)
    -> AS "Revenue Percentage"
    -> FROM Products P
    -> LEFT JOIN OrderItems Oi
    -> ON P.Product_ID = Oi.Product_ID
    -> GROUP BY P.Product_ID, P.Name;
+------------+----------------+-----------------+--------------------+
| Product_ID | Name           | Product Revenue | Revenue Percentage |
+------------+----------------+-----------------+--------------------+
|          1 | Laptop         |            2400 |              27.91 |
|          2 | Smartphone     |            3000 |              34.88 |
|          3 | Tablet         |             300 |               3.49 |
|          4 | Headphones     |             600 |               6.98 |
|          5 | TV             |            1800 |              20.93 |
|          6 | Coffee Maker   |              50 |               0.58 |
|          7 | Refrigerator   |               0 |               0.00 |
|          8 | Microwave Oven |               0 |               0.00 |
|          9 | Blender        |             210 |               2.44 |
|         10 | Vacuum Cleaner |             240 |               2.79 |
+------------+----------------+-----------------+--------------------+
10 rows in set (0.00 sec)
```

18. Subquery to Find Products with Low Stock.

Query:

```
SELECT * FROM Products
WHERE StockQuantity <
(
   SELECT AVG(StockQuantity)
   FROM Products
);
```

Output:

```
mysql> SELECT * FROM Products WHERE StockQuantity < (SELECT AVG(StockQuantity) FROM Products);
+------------+----------------+--------------------------+-------+---------------+-------------+
| Product_id | Name           | Description              | Price | StockQuantity | Category    |
+------------+----------------+--------------------------+-------+---------------+-------------+
|          1 | Laptop         | High-performance laptop  |   800 |            10 | Electronics |
|          2 | Smartphone     | Latest smartphone        |   600 |            15 | Electronics |
|          5 | TV             | 4K Smart TV              |   900 |             5 | Electronics |
|          7 | Refrigerator   | Energy-efficient         |   800 |            10 | Appliances  |
|          8 | Microwave Oven | Countertop microwave     |    80 |            15 | Appliances  |
|         10 | Vacuum Cleaner | Bagless vacuum cleaner   |   120 |            10 | Appliances  |
+------------+----------------+--------------------------+-------+---------------+-------------+
6 rows in set (0.00 sec)
```

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

Query:

```
SELECT * FROM Customers
WHERE Customer_ID IN
(
   SELECT Customer_ID
   FROM Orders
   WHERE TotalAmount =
   (
      SELECT MAX(TotalAmount)
      FROM Orders
   )
);
```

Output:

```
mysql> SELECT * FROM Customers
    -> WHERE Customer_ID IN
    -> (
    ->     SELECT Customer_ID
    ->     FROM Orders
    ->     WHERE TotalAmount =
    ->     (
    ->         SELECT MAX(TotalAmount)
    ->         FROM Orders
    ->     )
    -> );
+-------------+-----------+-------------------+-----------------------+
| customer_id | name      | email             | Address               |
+-------------+-----------+-------------------+-----------------------+
|           5 | David Lee | david@example.com | 234 Cedar St, District |
+-------------+-----------+-------------------+-----------------------+
1 row in set (0.01 sec)
```