Task 3: SQL for Data Analysis - Internship Documentation

Objective

To demonstrate SQL skills by extracting insights, creating views, using advanced queries, and optimizing performance using an ecommerce demo database.

📚 Index of Queries

📄 Step-by-Step Task Completion Guide

✅ Step 1: Environment Setup

- **Tool Used**: MySQL 8.0 Command Line Client
- **Dataset**: Ecommerce SQL Demo Dataset

✅ Step 2: Dataset Acquisition

1. Go to the GitHub repo: https://github.com/rohankale/ecommerce-sql-database
2. Download the ecommerce_demo.sql file
3. Save it to a known local path (e.g., C:\Users\SONY\Documents\ecommerce_demo.sql)

✅ Step 3: Import Dataset into MySQL

1. Open CMD
2. Navigate to MySQL installation directory:

```
cd C:\Program Files\MySQL\MySQL Server 8.0\bin
```

3. Import the SQL file:

```
mysql -u root -p < "C:\Users\SONY\Documents\ecommerce_demo.sql"
```

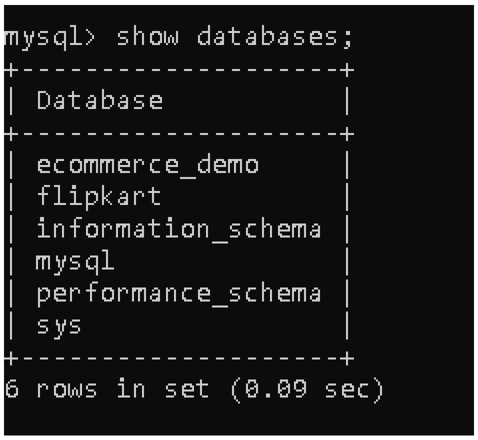
4. Enter your password when prompted.
5. Database will be imported successfully.

✅ Step 4: Connect to MySQL

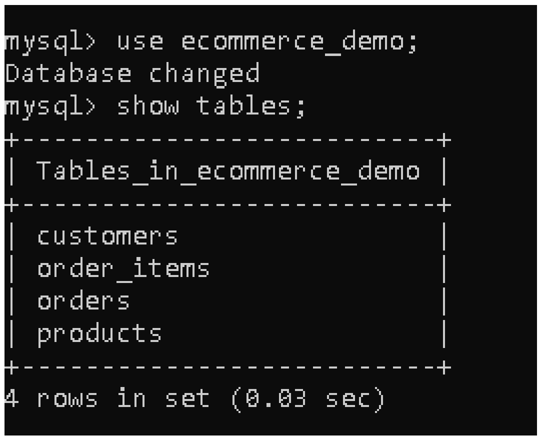
```
mysql -u root -p

SHOW DATABASES;

USE ecommerce_demo;
```

📊 Analysis & Query Execution

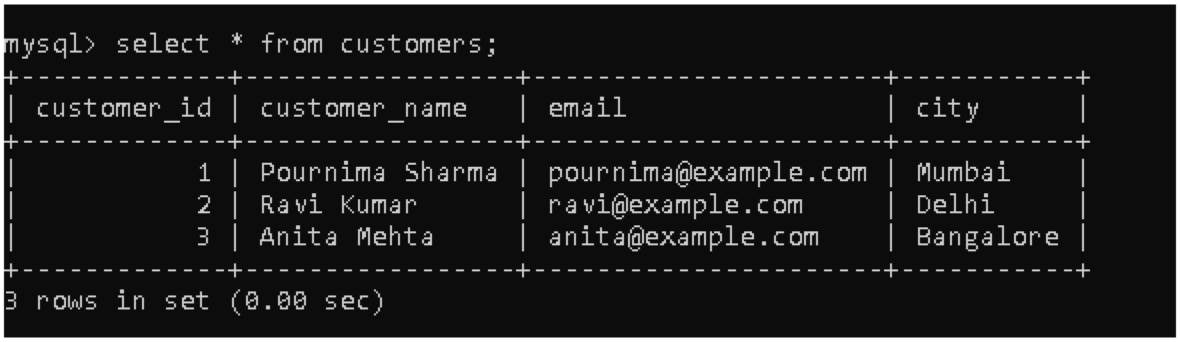
All queries are included in task3_sql_analysis.sql. Key queries include:



```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| ecommerce_demo     |
| flipkart           |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.09 sec)
```



```
mysql> use ecommerce_demo;
Database changed
mysql> show tables;
+--------------------------+
| Tables_in_ecommerce_demo |
+--------------------------+
| customers                |
| order_items              |
| orders                   |
| products                 |
+--------------------------+
4 rows in set (0.03 sec)
```



```
mysql> select * from customers;
+-------------+-----------------+----------------------+-----------+
| customer_id | customer_name   | email                | city      |
+-------------+-----------------+----------------------+-----------+
|           1 | Pournima Sharma | pournima@example.com | Mumbai    |
|           2 | Ravi Kumar      | ravi@example.com     | Delhi     |
|           3 | Anita Mehta     | anita@example.com    | Bangalore |
+-------------+-----------------+----------------------+-----------+
3 rows in set (0.00 sec)
```

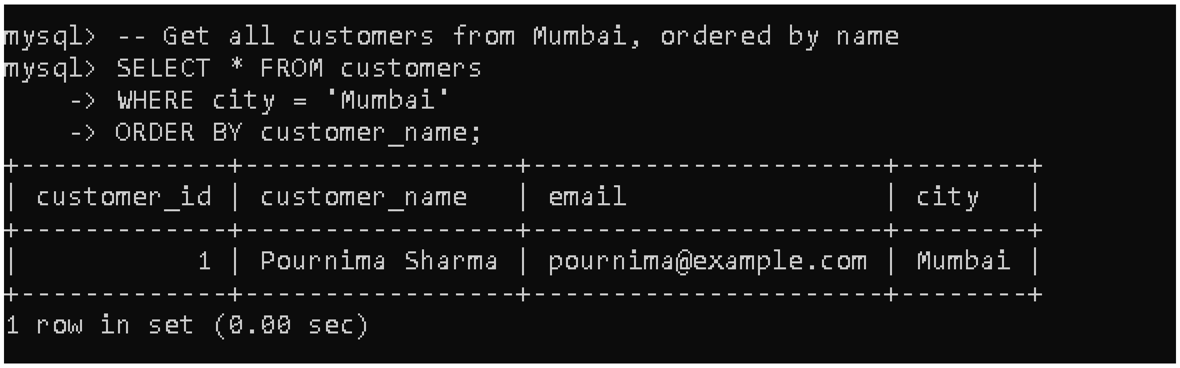
🔍 Basic Queries

1. Customers from Mumbai

SELECT * FROM customers

WHERE city = 'Mumbai'

ORDER BY customer_name;



```
mysql> -- Get all customers from Mumbai, ordered by name
mysql> SELECT * FROM customers
    -> WHERE city = 'Mumbai'
    -> ORDER BY customer_name;
+-------------+-----------------+----------------------+--------+
| customer_id | customer_name   | email                | city   |
+-------------+-----------------+----------------------+--------+
|           1 | Pournima Sharma | pournima@example.com | Mumbai |
+-------------+-----------------+----------------------+--------+
1 row in set (0.00 sec)
```

📈 Aggregate Functions & Grouping

2. Total quantity ordered for each product

SELECT

    p.product_name,

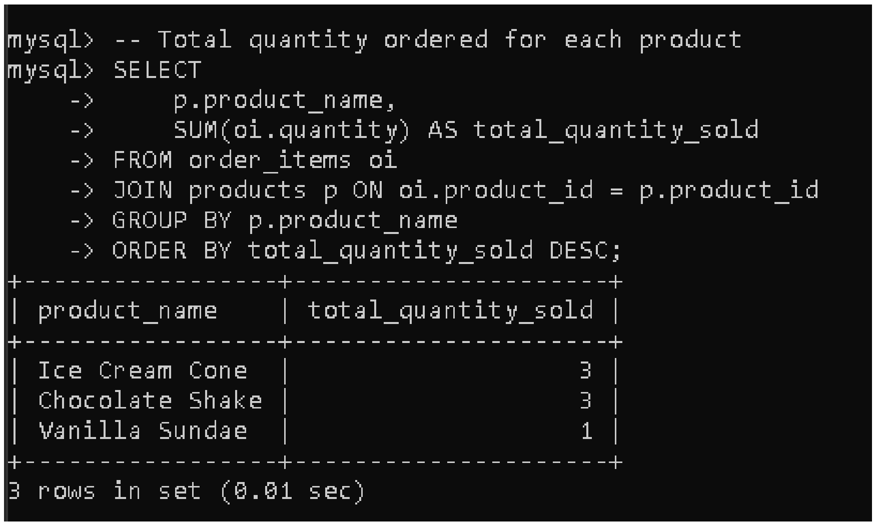
    SUM(oi.quantity) AS total_quantity_sold

FROM order_items oi

JOIN products p ON oi.product_id = p.product_id

GROUP BY p.product_name

ORDER BY total_quantity_sold DESC;



```
mysql> -- Total quantity ordered for each product
mysql> SELECT
    ->     p.product_name,
    ->     SUM(oi.quantity) AS total_quantity_sold
    -> FROM order_items oi
    -> JOIN products p ON oi.product_id = p.product_id
    -> GROUP BY p.product_name
    -> ORDER BY total_quantity_sold DESC;
+-----------------+---------------------+
| product_name    | total_quantity_sold |
+-----------------+---------------------+
| Ice Cream Cone  |                   3 |
| Chocolate Shake |                   3 |
| Vanilla Sundae  |                   1 |
+-----------------+---------------------+
3 rows in set (0.01 sec)
```

🔍 Basic Queries

3. Customer Orders Summary

SELECT

    c.customer_name,

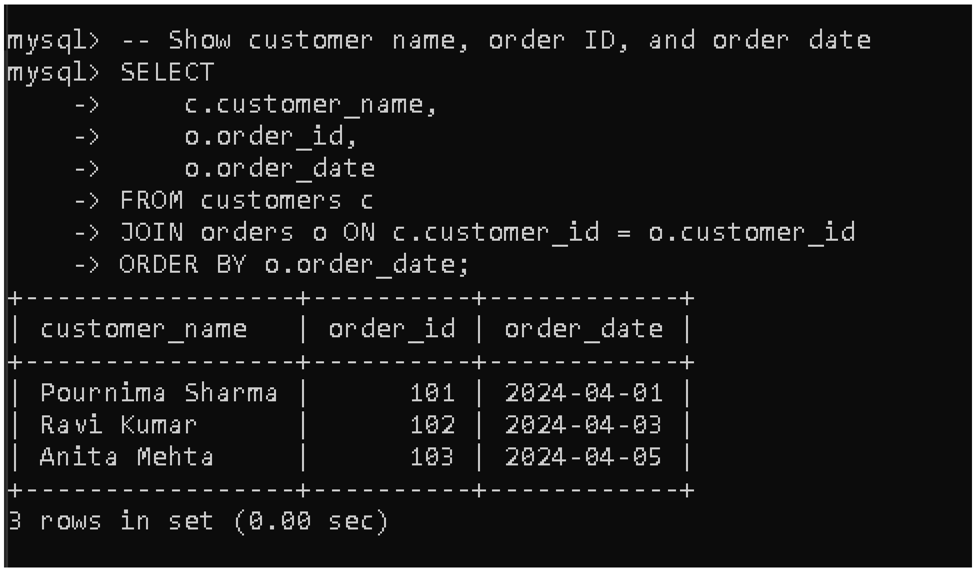
    o.order_id,

    o.order_date

FROM customers c

JOIN orders o ON c.customer_id = o.customer_id

ORDER BY o.order_date;

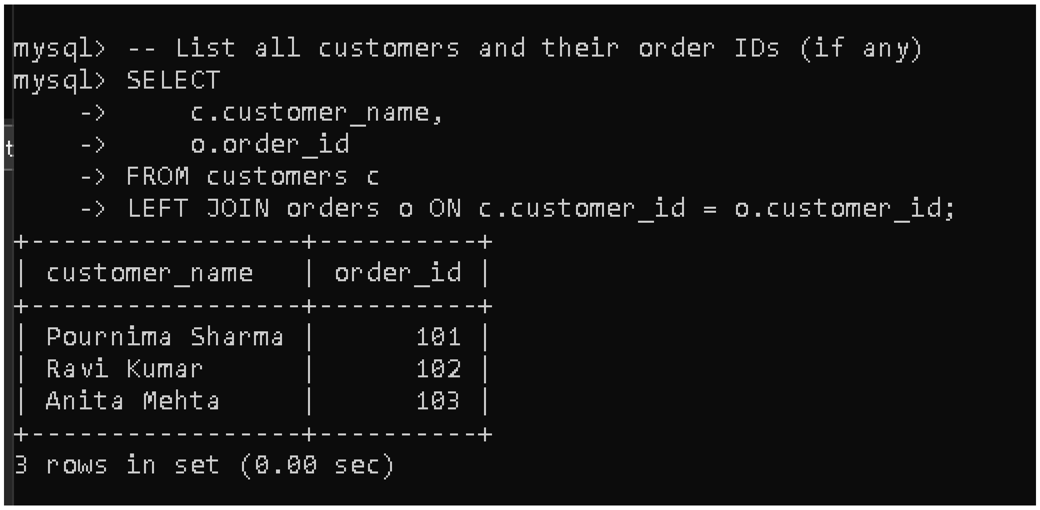


```
mysql> -- Show customer name, order ID, and order date
mysql> SELECT
    ->     c.customer_name,
    ->     o.order_id,
    ->     o.order_date
    -> FROM customers c
    -> JOIN orders o ON c.customer_id = o.customer_id
    -> ORDER BY o.order_date;
+-----------------+----------+------------+
| customer_name   | order_id | order_date |
+-----------------+----------+------------+
| Pournima Sharma |      101 | 2024-04-01 |
| Ravi Kumar      |      102 | 2024-04-03 |
| Anita Mehta     |      103 | 2024-04-05 |
+-----------------+----------+------------+
3 rows in set (0.00 sec)
```

🔍 Basic Queries

4. Customers and their order IDs (including those with no orders)

```
SELECT
    c.customer_name,
    o.order_id
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id;
```

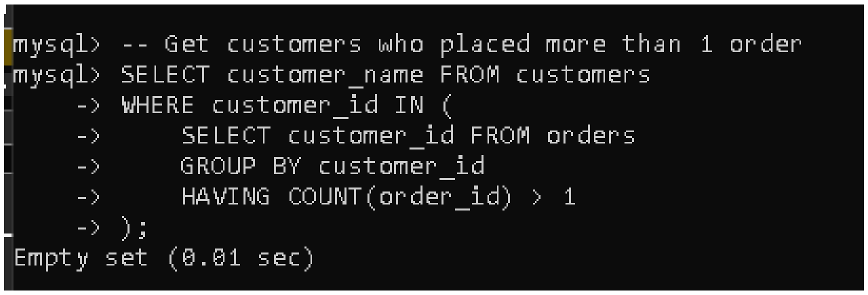


```
mysql> -- List all customers and their order IDs (if any)
mysql> SELECT
    ->     c.customer_name,
    ->     o.order_id
    -> FROM customers c
    -> LEFT JOIN orders o ON c.customer_id = o.customer_id;
+-----------------+----------+
| customer_name   | order_id |
+-----------------+----------+
| Pournima Sharma |      101 |
| Ravi Kumar      |      102 |
| Anita Mehta     |      103 |
+-----------------+----------+
3 rows in set (0.00 sec)
```

🧷 Joins & Subqueries

5. Customers with more than one order

```
SELECT customer_name FROM customers

WHERE customer_id IN (

    SELECT customer_id FROM orders

    GROUP BY customer_id

    HAVING COUNT(order_id) > 1

);
```

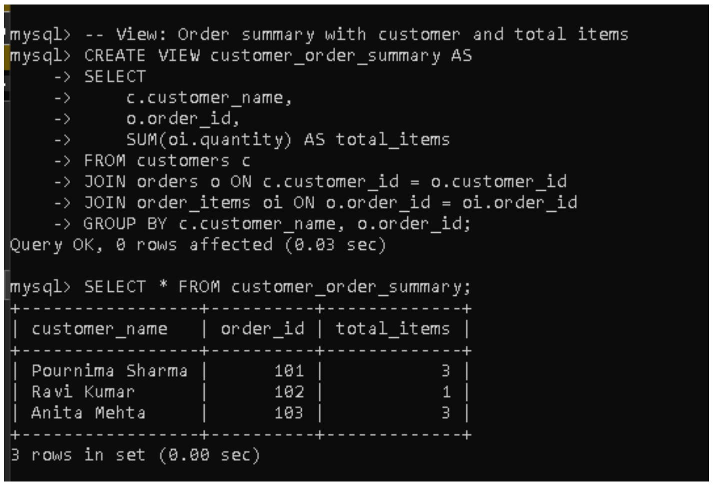


```
mysql> -- Get customers who placed more than 1 order
mysql> SELECT customer_name FROM customers
    -> WHERE customer_id IN (
    ->     SELECT customer_id FROM orders
    ->     GROUP BY customer_id
    ->     HAVING COUNT(order_id) > 1
    -> );
Empty set (0.01 sec)
```

🧠 Views and Indexing

6. Create view: Customer Order Summary

```
CREATE OR REPLACE VIEW customer_order_summary AS

SELECT

    c.customer_name,

    o.order_id,

    SUM(oi.quantity) AS total_items

FROM customers c

JOIN orders o ON c.customer_id = o.customer_id

JOIN order_items oi ON o.order_id = oi.order_id

GROUP BY c.customer_name, o.order_id;
```

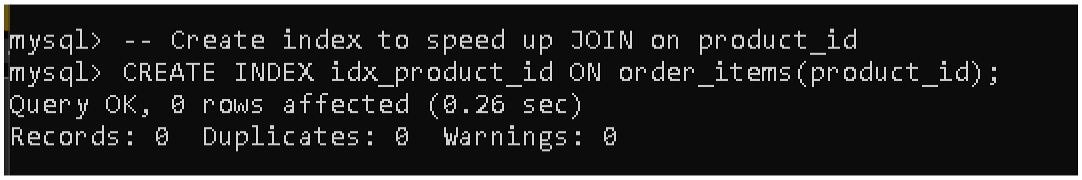


```
mysql> -- View: Order summary with customer and total items
mysql> CREATE VIEW customer_order_summary AS
    -> SELECT
    ->     c.customer_name,
    ->     o.order_id,
    ->     SUM(oi.quantity) AS total_items
    -> FROM customers c
    -> JOIN orders o ON c.customer_id = o.customer_id
    -> JOIN order_items oi ON o.order_id = oi.order_id
    -> GROUP BY c.customer_name, o.order_id;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM customer_order_summary;
+-----------------+----------+-------------+
| customer_name   | order_id | total_items |
+-----------------+----------+-------------+
| Pournima Sharma |      101 |           3 |
| Ravi Kumar      |      102 |           1 |
| Anita Mehta     |      103 |           3 |
+-----------------+----------+-------------+
3 rows in set (0.00 sec)
```

🧠 Views and Indexing

7. Create index on product_id

CREATE INDEX idx_product_id ON order_items(product_id);

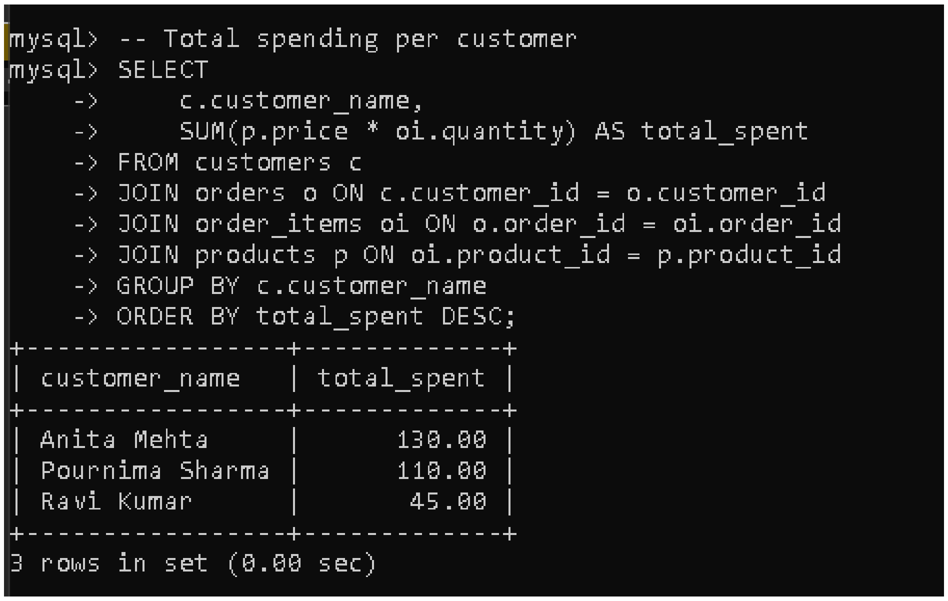


```
mysql> -- Create index to speed up JOIN on product_id
mysql> CREATE INDEX idx_product_id ON order_items(product_id);
Query OK, 0 rows affected (0.26 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

📈 Aggregate Functions & Grouping

8. Total spending per customer

```
SELECT
    c.customer_name,
    SUM(p.price * oi.quantity) AS total_spent
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
GROUP BY c.customer_name
ORDER BY total_spent DESC;
```

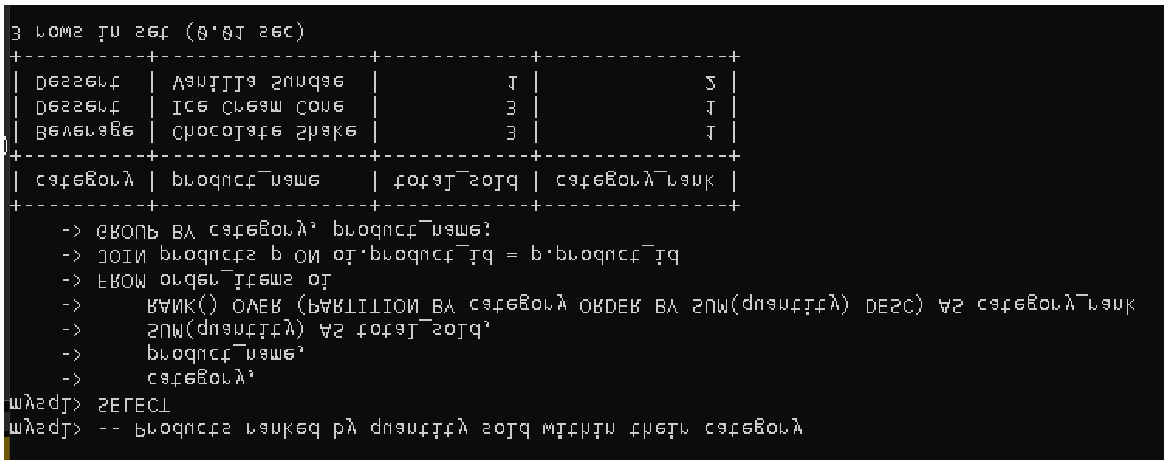


```
mysql> -- Total spending per customer
mysql> SELECT
    ->     c.customer_name,
    ->     SUM(p.price * oi.quantity) AS total_spent
    -> FROM customers c
    -> JOIN orders o ON c.customer_id = o.customer_id
    -> JOIN order_items oi ON o.order_id = oi.order_id
    -> JOIN products p ON oi.product_id = p.product_id
    -> GROUP BY c.customer_name
    -> ORDER BY total_spent DESC;
+----------------+-------------+
| customer_name  | total_spent |
+----------------+-------------+
| Anita Mehta    |      130.00 |
| Pournima Sharma |     110.00 |
| Ravi Kumar     |       45.00 |
+----------------+-------------+
3 rows in set (0.00 sec)
```

Window Functions & CTEs

9. Best-selling products by category rank

```
SELECT
    category, product_name,
    SUM(quantity) AS total_sold,
    RANK() OVER (PARTITION BY category ORDER BY SUM(quantity) DESC) AS category_rank
FROM order_items oi
JOIN products p ON oi.product_id = p.product_id
GROUP BY category, product_name;
```



```
mysql> -- Products ranked by quantity sold within their category
mysql> SELECT
    ->     category,
    ->     product_name,
    ->     SUM(quantity) AS total_sold,
    ->     RANK() OVER (PARTITION BY category ORDER BY SUM(quantity) DESC) AS category_rank
    -> FROM order_items oi
    -> JOIN products p ON oi.product_id = p.product_id
    -> GROUP BY category, product_name;
+----------+-----------------+------------+---------------+
| category | product_name    | total_sold | category_rank |
+----------+-----------------+------------+---------------+
| Beverage | Chocolate Shake |          3 |             1 |
| Dessert  | Ice Cream Cone  |          3 |             1 |
| Dessert  | Vanilla Sundae  |          1 |             2 |
+----------+-----------------+------------+---------------+
3 rows in set (0.01 sec)
```

🔍 Basic Queries

10. Recent orders in last 7 days

SELECT

o.order_id,

c.customer_name,

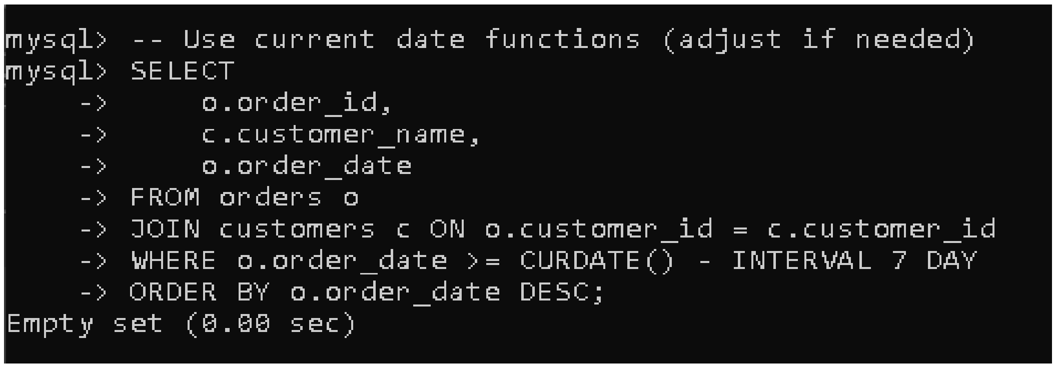
o.order_date

FROM orders o

JOIN customers c ON o.customer_id = c.customer_id

WHERE o.order_date >= CURDATE() - INTERVAL 7 DAY

ORDER BY o.order_date DESC;



```
mysql> -- Use current date functions (adjust if needed)
mysql> SELECT
    ->     o.order_id,
    ->     c.customer_name,
    ->     o.order_date
    -> FROM orders o
    -> JOIN customers c ON o.customer_id = c.customer_id
    -> WHERE o.order_date >= CURDATE() - INTERVAL 7 DAY
    -> ORDER BY o.order_date DESC;
Empty set (0.00 sec)
```

📈 Aggregate Functions & Grouping

11. Customer tier classification

SELECT

c.customer_name,

SUM(p.price * oi.quantity) AS total_spent,

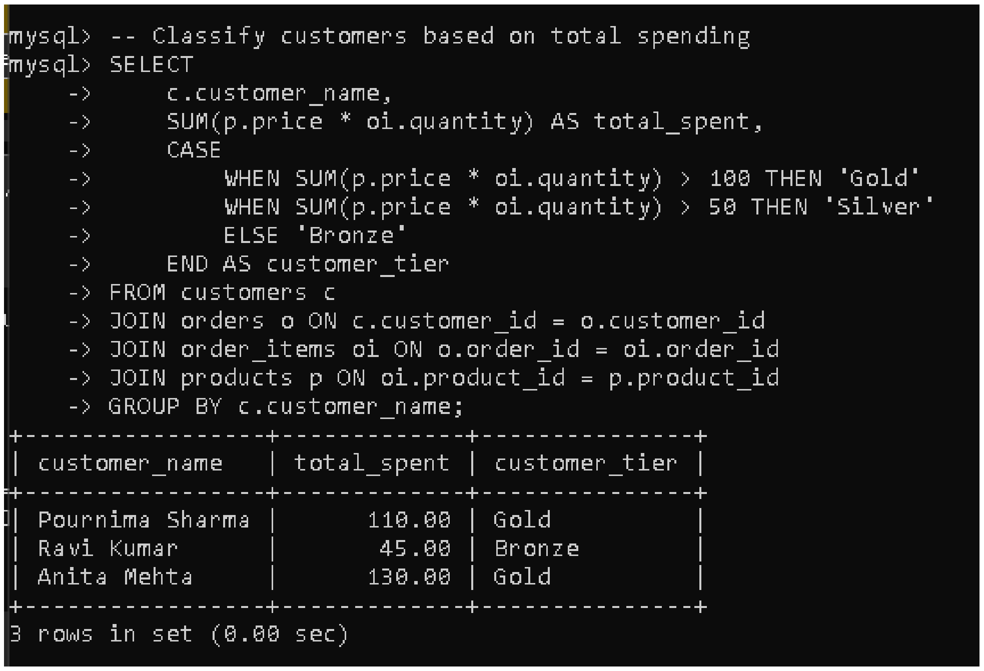
CASE

WHEN SUM(p.price * oi.quantity) > 100 THEN 'Gold'

WHEN SUM(p.price * oi.quantity) > 50 THEN 'Silver'

ELSE 'Bronze'

```
    END AS customer_tier

FROM customers c

JOIN orders o ON c.customer_id = o.customer_id

JOIN order_items oi ON o.order_id = oi.order_id

JOIN products p ON oi.product_id = p.product_id

GROUP BY c.customer_name;
```

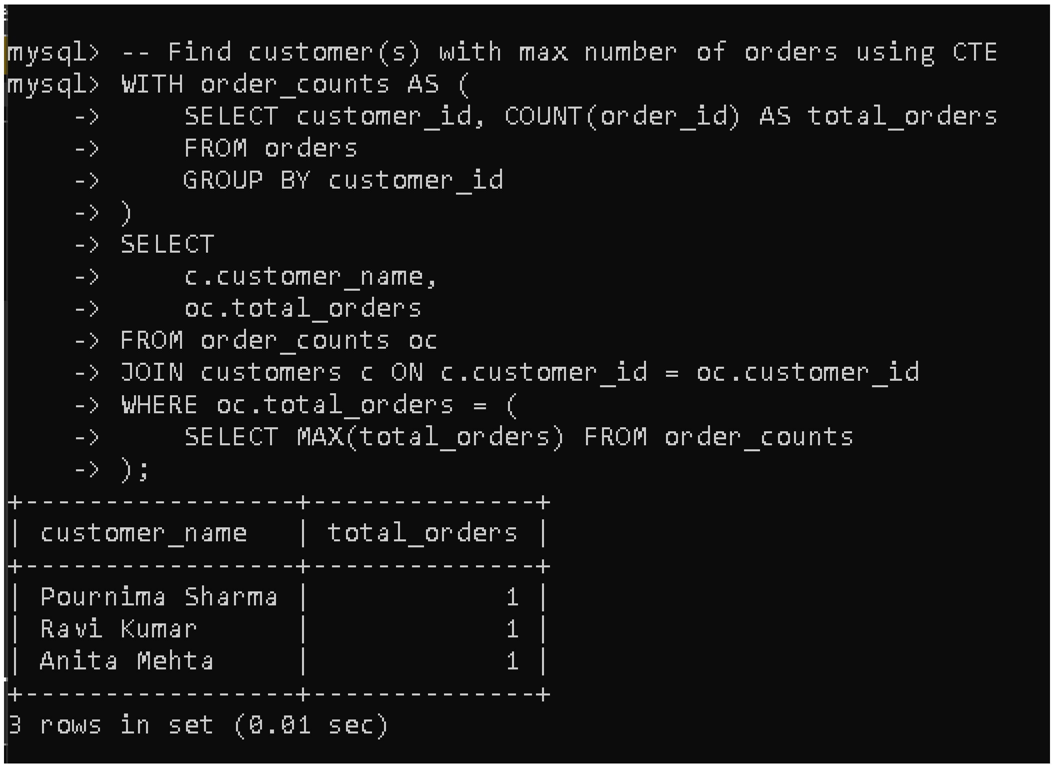


```
mysql> -- Classify customers based on total spending
mysql> SELECT
    ->     c.customer_name,
    ->     SUM(p.price * oi.quantity) AS total_spent,
    ->     CASE
    ->         WHEN SUM(p.price * oi.quantity) > 100 THEN 'Gold'
    ->         WHEN SUM(p.price * oi.quantity) > 50 THEN 'Silver'
    ->         ELSE 'Bronze'
    ->     END AS customer_tier
    -> FROM customers c
    -> JOIN orders o ON c.customer_id = o.customer_id
    -> JOIN order_items oi ON o.order_id = oi.order_id
    -> JOIN products p ON oi.product_id = p.product_id
    -> GROUP BY c.customer_name;
+----------------+-------------+---------------+
| customer_name  | total_spent | customer_tier |
+----------------+-------------+---------------+
| Pournima Sharma |     110.00 | Gold          |
| Ravi Kumar     |       45.00 | Bronze        |
| Anita Mehta    |      130.00 | Gold          |
+----------------+-------------+---------------+
3 rows in set (0.00 sec)
```

? Window Functions & CTEs

12. Most active customers (CTE)

```
WITH order_counts AS (

    SELECT customer_id, COUNT(order_id) AS total_orders

    FROM orders

    GROUP BY customer_id

)

SELECT

    c.customer_name,
    oc.total_orders

FROM order_counts oc

JOIN customers c ON c.customer_id = oc.customer_id

WHERE oc.total_orders = (

    SELECT MAX(total_orders) FROM order_counts

);
```



```
mysql> -- Find customer(s) with max number of orders using CTE
mysql> WITH order_counts AS (
    ->     SELECT customer_id, COUNT(order_id) AS total_orders
    ->     FROM orders
    ->     GROUP BY customer_id
    -> )
    -> SELECT
    ->     c.customer_name,
    ->     oc.total_orders
    -> FROM order_counts oc
    -> JOIN customers c ON c.customer_id = oc.customer_id
    -> WHERE oc.total_orders = (
    ->     SELECT MAX(total_orders) FROM order_counts
    -> );
+-----------------+--------------+
| customer_name   | total_orders |
+-----------------+--------------+
| Pournima Sharma |            1 |
| Ravi Kumar      |            1 |
| Anita Mehta     |            1 |
+-----------------+--------------+
3 rows in set (0.01 sec)
```

🧠 Views and Indexing

13. Enhanced view for dashboard

```
CREATE OR REPLACE VIEW detailed_order_summary AS

SELECT

    c.customer_name,

    p.category,

    p.product_name,
```

oi.quantity,

(p.price * oi.quantity) AS revenue,

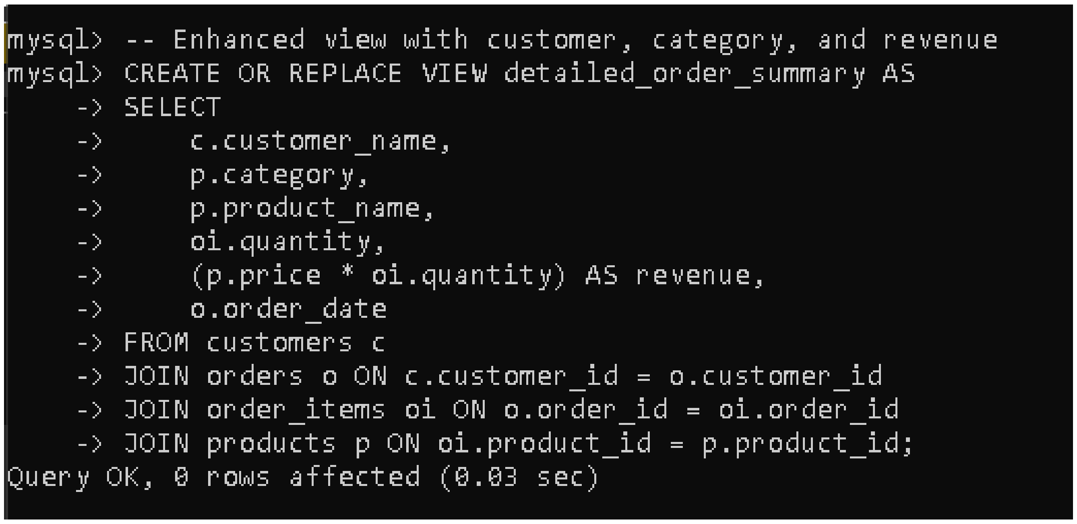
o.order_date

FROM customers c

JOIN orders o ON c.customer_id = o.customer_id

JOIN order_items oi ON o.order_id = oi.order_id

JOIN products p ON oi.product_id = p.product_id;



```
mysql> -- Enhanced view with customer, category, and revenue
mysql> CREATE OR REPLACE VIEW detailed_order_summary AS
    -> SELECT
    ->     c.customer_name,
    ->     p.category,
    ->     p.product_name,
    ->     oi.quantity,
    ->     (p.price * oi.quantity) AS revenue,
    ->     o.order_date
    -> FROM customers c
    -> JOIN orders o ON c.customer_id = o.customer_id
    -> JOIN order_items oi ON o.order_id = oi.order_id
    -> JOIN products p ON oi.product_id = p.product_id;
Query OK, 0 rows affected (0.03 sec)
```

⚙ Stored Procedures

14. Stored procedure to get top N customers

DELIMITER //

CREATE PROCEDURE top_customers(IN n INT)

BEGIN

SELECT

c.customer_name,

SUM(p.price * oi.quantity) AS total_spent

FROM customers c

JOIN orders o ON c.customer_id = o.customer_id

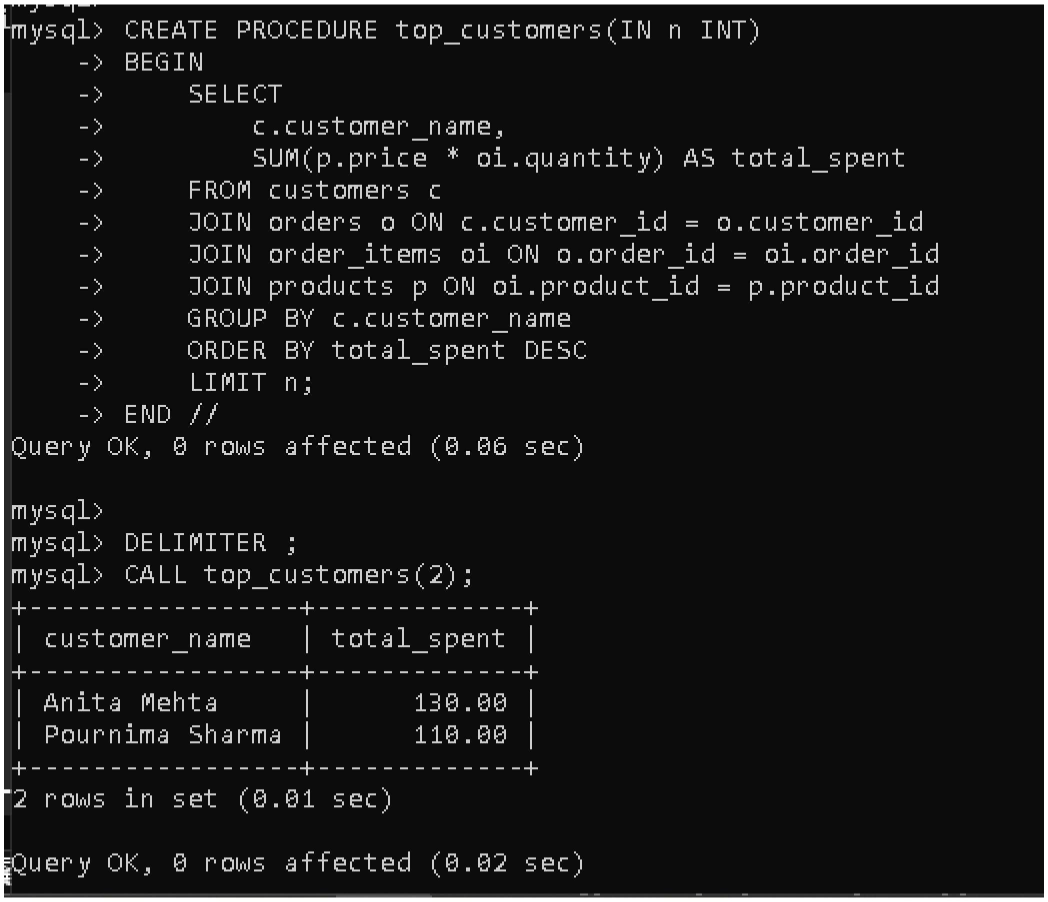
JOIN order_items oi ON o.order_id = oi.order_id

JOIN products p ON oi.product_id = p.product_id

GROUP BY c.customer_name

ORDER BY total_spent DESC

LIMIT n;



```
mysql> CREATE PROCEDURE top_customers(IN n INT)
    -> BEGIN
    ->     SELECT
    ->         c.customer_name,
    ->         SUM(p.price * oi.quantity) AS total_spent
    ->     FROM customers c
    ->     JOIN orders o ON c.customer_id = o.customer_id
    ->     JOIN order_items oi ON o.order_id = oi.order_id
    ->     JOIN products p ON oi.product_id = p.product_id
    ->     GROUP BY c.customer_name
    ->     ORDER BY total_spent DESC
    ->     LIMIT n;
    -> END //
Query OK, 0 rows affected (0.06 sec)

mysql>
mysql> DELIMITER ;
mysql> CALL top_customers(2);
+-----------------+-------------+
| customer_name   | total_spent |
+-----------------+-------------+
| Anita Mehta     |      130.00 |
| Pournima Sharma |      110.00 |
+-----------------+-------------+
2 rows in set (0.01 sec)

Query OK, 0 rows affected (0.02 sec)
```