SQL_Myntra_Assignment

Nandini

September 2025

1 Introduction

This report presents the results of fifteen SQL queries executed on a simulated Myntra E-Commerce database. The database includes tables for products, customers, orders, and order details. Each query demonstrates SQL clauses such as `WHERE`, `LIKE`, and aggregate functions. Screenshots of the outputs are included for documentation.

2 Database Setup

The following SQL statements were used to create the tables and insert sample data into the database.

```
-- Create Product table
CREATE TABLE Product (
  product_id INT PRIMARY KEY,
  name VARCHAR(50),
  brand VARCHAR(50),
  category VARCHAR(50),
  price DECIMAL(10,2),
  stock INT
);

-- Insert sample products
INSERT INTO Product (product_id, name, brand, category, price, stock)
VALUES
(1, 'T-shirt', 'Roadster', 'Topwear', 1299.00, 50),
(2, 'Jeans', 'Levis', 'Bottomwear', 2999.00, 30),
(3, 'Sneakers', 'Nike', 'Footwear', 1899.00, 20);

-- Create Customer table
CREATE TABLE Customer (
  customer_id INT PRIMARY KEY,
  name VARCHAR(50),
  city VARCHAR(50),
  gender VARCHAR(10)
);

-- Insert sample customers
INSERT INTO Customer (customer_id, name, city, gender)
VALUES
(1, 'Nayeon', 'Mumbai', 'Female'),
(2, 'Jihyo', 'Delhi', 'Female'),
(3, 'Sana', 'Banglore', 'Female'),
(4, 'Oliver', 'Mumbai', 'Male'),
```

```
(5, 'Sana', 'Banglore', 'Female'),
(6, 'Mina', 'Surat', 'Female'),
(7, 'Elio', 'Delhi', 'Male'),
(8, 'Peter', 'Ahmedabad', 'Male');

-- Create Orders table
CREATE TABLE Orders (
  order_id INT PRIMARY KEY,
  customer_id INT,
  order_date DATE,
  total_amount DECIMAL(10,2),
  FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
);

-- Insert sample orders
INSERT INTO Orders (order_id, customer_id, order_date, total_amount)
VALUES
(1, 1, '2025-09-01', 1299.00),
(2, 2, '2025-09-02', 2999.00),
(3, 3, '2025-09-03', 1899.00);

-- Create OrderDetails table
CREATE TABLE OrderDetails (
  detail_id INT AUTO_INCREMENT PRIMARY KEY,
  order_id INT,
  product_id INT,
  quantity INT,
  FOREIGN KEY (order_id) REFERENCES Orders(order_id),
  FOREIGN KEY (product_id) REFERENCES Product(product_id)
);

-- Insert sample order details
INSERT INTO OrderDetails (order_id, product_id, quantity)
VALUES
(1, 1, 2),
(2, 2, 3),
(3, 3, 3);
```

3 Query Results

Each task below includes a brief description and a screenshot of the query output.

Task 1:Output of Task 1: Products priced above Rs.1000

```
mysql> -- Products priced above 1000
mysql> SELECT
    ->
    ->
    ->
    -> * FROM Product WHERE price > 1000;
+------------+---------------+----------+----------+---------+-------+
| product_id | name          | brand    | category | price   | stock |
+------------+---------------+----------+----------+---------+-------+
|          1 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
|          2 | Denim Jacket  | GAP      | Outer    | 2999.00 |    50 |
|          3 | Cargo Pants   | H&M      | Bottom   | 1899.00 |    35 |
|          4 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
+------------+---------------+----------+----------+---------+-------+
4 rows in set (0.01 sec)
```

Task 2: Customers in cities starting with 'M'

```
mysql> -- Customers in cities starting with 'M'
mysql> SELECT * FROM Product WHERE city LIKE 'M%';
ERROR 1054 (42S22): Unknown column 'city' in 'where clause'
mysql> -- Customers in cities starting with 'M'
mysql> SELECT * FROM CUSTOMER
    -> WHERE CITY LIKE 'M%';
+-------------+--------+--------+--------+
| customer_id | name   | city   | gender |
+-------------+--------+--------+--------+
|           1 | Nayeon | Mumbai | Female |
|           4 | Oliver | Mumbai | Male   |
+-------------+--------+--------+--------+
2 rows in set (0.00 sec)
```

Retrieves customers whose city names begin with the letter 'M'.

Task 3: Products with brand containing 'Roadster'

```
mysql> Products with brand containing 'Roadster'
    -> SELECT * FROM Product
    -> WHERE brand LIKE '%Roadster%';
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Products with b
rand containing 'Roadster'
SELECT * FROM Product
WHERE brand LIKE' at line 1
mysql> -- Products with brand containing 'roadster'
mysql> SELECT * FROM Product
    -> WHERE BRAND LIKE '%Roadster%;'
    '> ;
Empty set (0.00 sec)

mysql> -- Products with brand containing 'Roadster'
mysql> SELECT * FROM PRODUCT
    -> WHERE BRAND LIKE '%Roadster%';
+------------+-----------------+-----------+----------+-------+-------+
| product_id | name            | brand     | category | price | stock |
+------------+-----------------+-----------+----------+-------+-------+
|          1 | Oversized Tee   | Roadster  | Tops     |  1299.00 |    50 |
|          4 | Oversized Tee   | Roadster  | Tops     |  1299.00 |    50 |
+------------+-----------------+-----------+----------+-------+-------+
2 rows in set (0.00 sec)

mysql>
```

Displays products where the brand name includes 'Roadster'.

Task 4: Female customers from Mumbai or Delhi

```
mysql> -- Female customers from Mumbai or Delhi
mysql> SELECT * FROM Customer
    -> WHERE gender = 'Female' AND city IN ('Mumbai', 'Delhi');
+-------------+---------+--------+--------+
| customer_id | name    | city   | gender |
+-------------+---------+--------+--------+
|           1 | Nayeon  | Mumbai | Female |
|           2 | Jihyo   | Delhi  | Female |
+-------------+---------+--------+--------+
2 rows in set (0.01 sec)

mysql>
```

Filters female customers located in either Mumbai or Delhi.

Task 5: Total products in stock

```
mysql> Total products in stock
    -> SELECT SUM(stock) AS total_stock
    -> FROM Product;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Total products
in stock
SELECT SUM(stock) AS total_stock
FROM Product' at line 1
mysql> -- Total products in stock
mysql> SELECT SUM (stock) AS total_stock
    -> FROM Product;
ERROR 1630 (42000): FUNCTION myntraschemadb.SUM does not exist. Check the 'Function Name Parsing and Resolution' section in the Reference Manual
mysql> -- Total products in stock
mysql> SELECT SUM(stock) AS total_stock
    -> FROM Product;
+-------------+
| total_stock |
+-------------+
|         185 |
+-------------+
1 row in set (0.04 sec)

mysql> -- Average order amount
mysql> SELECT AVG(total_amount) AS average_order
    -> FROM Orders;
+---------------+
| average_order |
+---------------+
|   2065.666667 |
+---------------+
1 row in set (0.00 sec)
```

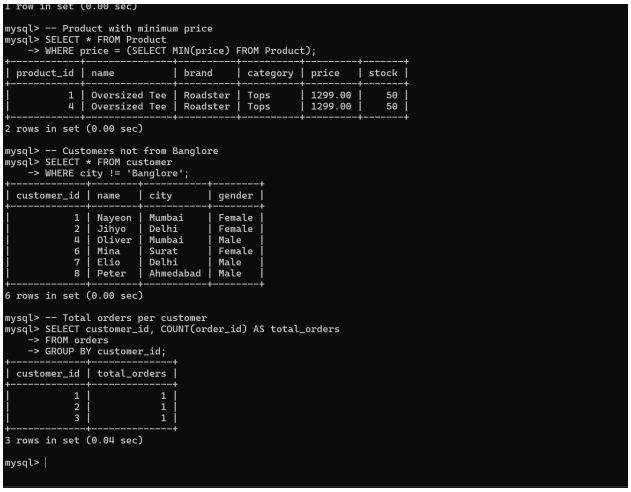
Calculates the total number of products available across all inventory.

Task 6: Average order amount

```
mysql> Total products in stock
    -> SELECT SUM(stock) AS total_stock
    -> FROM Product;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'Total products
in stock
SELECT SUM(stock) AS total_stock
FROM Product' at line 1
mysql> -- Total products in stock
mysql> SELECT SUM (stock) AS total_stock
    -> FROM Product;
ERROR 1630 (42000): FUNCTION myntraschemadb.SUM does not exist. Check the 'Function Name Parsing and Resolution' section in the Reference Manual
mysql> -- Total products in stock
mysql> SELECT SUM(stock) AS total_stock
    -> FROM Product;
+-------------+
| total_stock |
+-------------+
|         185 |
+-------------+
1 row in set (0.04 sec)

mysql> -- Average order amount
mysql> SELECT AVG(total_amount) AS average_order
    -> FROM Orders;
+---------------+
| average_order |
+---------------+
|   2065.666667 |
+---------------+
1 row in set (0.00 sec)
```

Returns the average value of all orders placed.

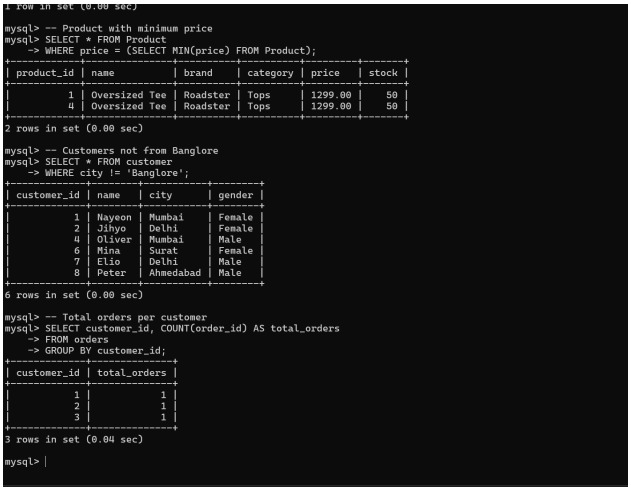
Task 7: Product with minimum price



```
1 row in set (0.00 sec)

mysql> -- Product with minimum price
mysql> SELECT * FROM Product
    -> WHERE price = (SELECT MIN(price) FROM Product);
+------------+----------------+----------+----------+---------+-------+
| product_id | name           | brand    | category | price   | stock |
+------------+----------------+----------+----------+---------+-------+
|          1 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
|          4 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
+------------+----------------+----------+----------+---------+-------+
2 rows in set (0.00 sec)

mysql> -- Customers not from Banglore
mysql> SELECT * FROM customer
    -> WHERE city != 'Banglore';
+-------------+--------+-----------+--------+
| customer_id | name   | city      | gender |
+-------------+--------+-----------+--------+
|           1 | Nayeon | Mumbai    | Female |
|           2 | Jihyo  | Delhi     | Female |
|           4 | Oliver | Mumbai    | Male   |
|           6 | Mina   | Surat     | Female |
|           7 | Elio   | Delhi     | Male   |
|           8 | Peter  | Ahmedabad | Male   |
+-------------+--------+-----------+--------+
6 rows in set (0.00 sec)

mysql> -- Total orders per customer
mysql> SELECT customer_id, COUNT(order_id) AS total_orders
    -> FROM orders
    -> GROUP BY customer_id;
+-------------+--------------+
| customer_id | total_orders |
+-------------+--------------+
|           1 |            1 |
|           2 |            1 |
|           3 |            1 |
+-------------+--------------+
3 rows in set (0.04 sec)

mysql>
```

Identifies the product with the lowest price in the catalog.

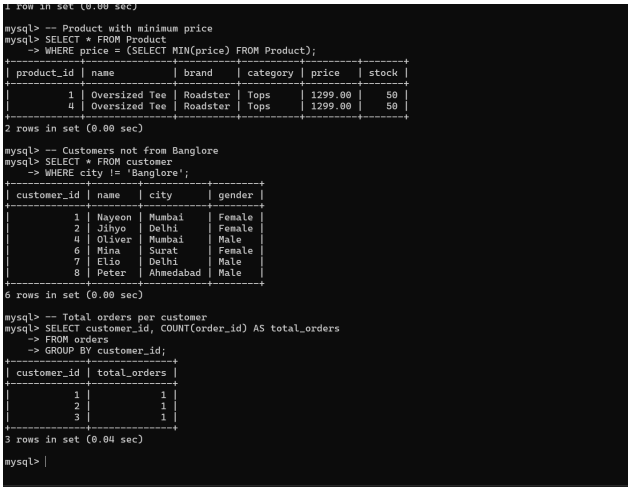
Task 8: Customers not from Bangalore



```
1 row in set (0.00 sec)

mysql> -- Product with minimum price
mysql> SELECT * FROM Product
    -> WHERE price = (SELECT MIN(price) FROM Product);
+------------+----------------+----------+----------+---------+-------+
| product_id | name           | brand    | category | price   | stock |
+------------+----------------+----------+----------+---------+-------+
|          1 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
|          4 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
+------------+----------------+----------+----------+---------+-------+
2 rows in set (0.00 sec)

mysql> -- Customers not from Banglore
mysql> SELECT * FROM customer
    -> WHERE city != 'Banglore';
+-------------+--------+-----------+--------+
| customer_id | name   | city      | gender |
+-------------+--------+-----------+--------+
|           1 | Nayeon | Mumbai    | Female |
|           2 | Jihyo  | Delhi     | Female |
|           4 | Oliver | Mumbai    | Male   |
|           6 | Mina   | Surat     | Female |
|           7 | Elio   | Delhi     | Male   |
|           8 | Peter  | Ahmedabad | Male   |
+-------------+--------+-----------+--------+
6 rows in set (0.00 sec)

mysql> -- Total orders per customer
mysql> SELECT customer_id, COUNT(order_id) AS total_orders
    -> FROM orders
    -> GROUP BY customer_id;
+-------------+--------------+
| customer_id | total_orders |
+-------------+--------------+
|           1 |            1 |
|           2 |            1 |
|           3 |            1 |
+-------------+--------------+
3 rows in set (0.04 sec)

mysql>
```

Lists all customers whose city is not Bangalore.

Task 9: Total orders placed by each customer



```
1 row in set (0.00 sec)

mysql> -- Product with minimum price
mysql> SELECT * FROM Product
    -> WHERE price = (SELECT MIN(price) FROM Product);
+------------+----------------+----------+----------+---------+-------+
| product_id | name           | brand    | category | price   | stock |
+------------+----------------+----------+----------+---------+-------+
|          1 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
|          4 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
+------------+----------------+----------+----------+---------+-------+
2 rows in set (0.00 sec)

mysql> -- Customers not from Banglore
mysql> SELECT * FROM customer
    -> WHERE city != 'Banglore';
+-------------+--------+-----------+--------+
| customer_id | name   | city      | gender |
+-------------+--------+-----------+--------+
|           1 | Nayeon | Mumbai    | Female |
|           2 | Jihyo  | Delhi     | Female |
|           4 | Oliver | Mumbai    | Male   |
|           6 | Mina   | Surat     | Female |
|           7 | Elio   | Delhi     | Male   |
|           8 | Peter  | Ahmedabad | Male   |
+-------------+--------+-----------+--------+
6 rows in set (0.00 sec)

mysql> -- Total orders per customer
mysql> SELECT customer_id, COUNT(order_id) AS total_orders
    -> FROM orders
    -> GROUP BY customer_id;
+-------------+--------------+
| customer_id | total_orders |
+-------------+--------------+
|           1 |            1 |
|           2 |            1 |
|           3 |            1 |
+-------------+--------------+
3 rows in set (0.04 sec)

mysql>
```

Shows how many orders each customer has placed.

Task 10: Customers whose name starts with 'A' and ends with 'a'

```
mysql> -- Customers whose name starts with 'A' and ends with 'a'
mysql> SELECT * FROM Customer
    -> WHERE NAME LIKE 'A%a';
+-------------+--------+--------+--------+
| customer_id | name   | city   | gender |
+-------------+--------+--------+--------+
|           9 | Ananya | Indore | Female |
+-------------+--------+--------+--------+
1 row in set (0.00 sec)
```

Filters customers whose names begin with 'A' and end with 'a'.

Task 11: Products where category contains exactly 5 letters

```
mysql> -- Products with category of exact 5 letters
mysql> SELECT * FROM PRODUCT
    -> WHERE Length(category) = 5;
+------------+--------------+-------+----------+---------+-------+
| product_id | name         | brand | category | price   | stock |
+------------+--------------+-------+----------+---------+-------+
|          2 | Denim Jacket | GAP   | Outer    | 2999.00 |    50 |
+------------+--------------+-------+----------+---------+-------+
1 row in set (0.04 sec)
```

Returns products whose category name is exactly five characters long.

Task 12: Total quantity sold for each product ID

```
ysql> -- Total quantity sold per product ID
ysql> SELECT PRODUCT_ID, SUM(QUANTITY) AS TOTAL_QUANTITY
   -> FROM ORDERDETAILS
   -> GROUP BY PRODUCT_ID;
+------------+----------------+
| PRODUCT_ID | TOTAL_QUANTITY |
+------------+----------------+
|          1 |              2 |
|          2 |              3 |
|          3 |              3 |
+------------+----------------+
 rows in set (0.01 sec)

ysql> -- Products priced between 1000 and 3000
ysql> SELECT * FROM PRODUCT
   -> WHERE price > 1000 AND price < 3000;
+------------+--------------+----------+----------+---------+-------+
| product_id | name         | brand    | category | price   | stock |
+------------+--------------+----------+----------+---------+-------+
|          1 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
|          2 | Denim Jacket  | GAP      | Outer    | 2999.00 |    50 |
|          3 | Cargo Pants   | H&M      | Bottom   | 1899.00 |    35 |
|          4 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
+------------+--------------+----------+----------+---------+-------+
 rows in set (0.00 sec)
```

Aggregates the total quantity sold per product across all orders.

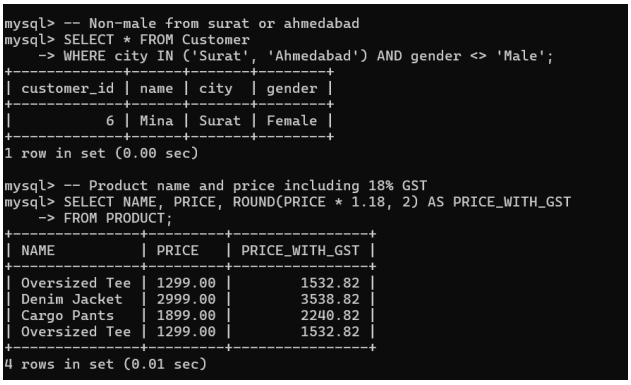
Task 13: Products priced between Rs.1000 and Rs.3000

```
ysql> -- Total quantity sold per product ID
ysql> SELECT PRODUCT_ID, SUM(QUANTITY) AS TOTAL_QUANTITY
   -> FROM ORDERDETAILS
   -> GROUP BY PRODUCT_ID;
+------------+----------------+
| PRODUCT_ID | TOTAL_QUANTITY |
+------------+----------------+
|          1 |              2 |
|          2 |              3 |
|          3 |              3 |
+------------+----------------+
 rows in set (0.01 sec)

ysql> -- Products priced between 1000 and 3000
ysql> SELECT * FROM PRODUCT
   -> WHERE price > 1000 AND price < 3000;
+------------+--------------+----------+----------+---------+-------+
| product_id | name         | brand    | category | price   | stock |
+------------+--------------+----------+----------+---------+-------+
|          1 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
|          2 | Denim Jacket  | GAP      | Outer    | 2999.00 |    50 |
|          3 | Cargo Pants   | H&M      | Bottom   | 1899.00 |    35 |
|          4 | Oversized Tee | Roadster | Tops     | 1299.00 |    50 |
+------------+--------------+----------+----------+---------+-------+
 rows in set (0.00 sec)
```

Displays products with prices strictly between Rs.1000 and Rs.3000.

Task 14: Customers from Surat or Ahmedabad but not male

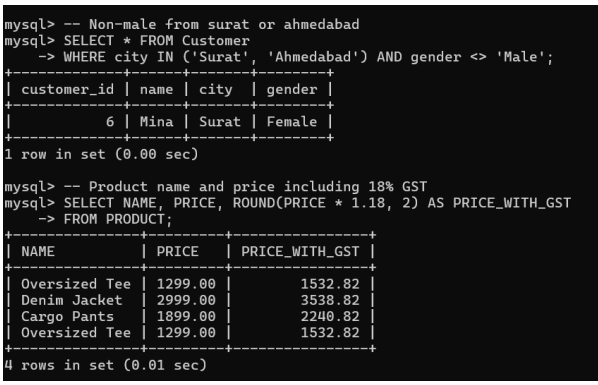
Filters customers from Surat or Ahmedabad who are not male.



```
mysql> -- Non-male from surat or ahmedabad
mysql> SELECT * FROM Customer
    -> WHERE city IN ('Surat', 'Ahmedabad') AND gender <> 'Male';
+-------------+------+-------+--------+
| customer_id | name | city  | gender |
+-------------+------+-------+--------+
|           6 | Mina | Surat | Female |
+-------------+------+-------+--------+
1 row in set (0.00 sec)

mysql> -- Product name and price including 18% GST
mysql> SELECT NAME, PRICE, ROUND(PRICE * 1.18, 2) AS PRICE_WITH_GST
    -> FROM PRODUCT;
+---------------+---------+----------------+
| NAME          | PRICE   | PRICE_WITH_GST |
+---------------+---------+----------------+
| Oversized Tee | 1299.00 |        1532.82 |
| Denim Jacket  | 2999.00 |        3538.82 |
| Cargo Pants   | 1899.00 |        2240.82 |
| Oversized Tee | 1299.00 |        1532.82 |
+---------------+---------+----------------+
4 rows in set (0.01 sec)
```

Task 15: Product name and price including 18% GST

Calculates and displays the price of each product after applying 18% GST.



```
mysql> -- Non-male from surat or ahmedabad
mysql> SELECT * FROM Customer
    -> WHERE city IN ('Surat', 'Ahmedabad') AND gender <> 'Male';
+-------------+------+-------+--------+
| customer_id | name | city  | gender |
+-------------+------+-------+--------+
|           6 | Mina | Surat | Female |
+-------------+------+-------+--------+
1 row in set (0.00 sec)

mysql> -- Product name and price including 18% GST
mysql> SELECT NAME, PRICE, ROUND(PRICE * 1.18, 2) AS PRICE_WITH_GST
    -> FROM PRODUCT;
+---------------+---------+----------------+
| NAME          | PRICE   | PRICE_WITH_GST |
+---------------+---------+----------------+
| Oversized Tee | 1299.00 |        1532.82 |
| Denim Jacket  | 2999.00 |        3538.82 |
| Cargo Pants   | 1899.00 |        2240.82 |
| Oversized Tee | 1299.00 |        1532.82 |
+---------------+---------+----------------+
4 rows in set (0.01 sec)
```