Task 3

Project: Online Store Order Management System (PostgreSQL)

Objective: Create a system to manage orders, customers, and products for an online store

Database Creation: Create a database named OnlineStore.

Create tables: Customers (CUSTOMER_ID, NAME, EMAIL, PHONE, ADDRESS)

Products (PRODUCT_ID, PRODUCT_NAME, CATEGORY, PRICE, STOCK)

Orders (ORDER_ID, CUSTOMER_ID, PRODUCT_ID, QUANTITY, ORDER_DATE) Set up foreign keys linking Orders to Customers and Products.

Data Creation: Insert sample data for customers, products, and orders.

```
INSERT 0 4
onlinestore=# SELECT * FROM customers;
 customer_id |  name   |        email        |   phone    |  address
-------------+---------+---------------------+------------+-----------
           1 | Alice   | alice@example.com   | 9876543210 | Bangalore
           2 | Bob     | bob@example.com     | 9876543211 | Chennai
           3 | Charlie | charlie@example.com | 9876543212 | Hyderabad
(3 rows)


onlinestore=# SELECT * FROM products;
 product_id | product_name |  category   |  price   | stock
------------+--------------+-------------+----------+-------
          1 | Laptop       | Electronics | 60000.00 |    10
          2 | Mobile       | Electronics | 20000.00 |     0
          3 | Shoes        | Fashion     |  3000.00 |    25
          4 | Watch        | Fashion     |  5000.00 |     5
(4 rows)


onlinestore=# SELECT * FROM orders;
 order_id | customer_id | product_id | quantity | order_date
----------+-------------+------------+----------+------------
        1 |           1 |          1 |        1 | 2025-08-01
        2 |           2 |          2 |        2 | 2025-08-02
        3 |           1 |          3 |        1 | 2025-08-03
        4 |           3 |          4 |        2 | 2025-08-05
(4 rows)
```

Order Management:

a) Retrieve all orders placed by a specific customer.

```
onlinestore=# SELECT o.order_id, c.name AS customer, p.product_name, o.quantity, o.order_d
onlinestore-# FROM orders o
onlinestore-# JOIN products p ON o.product_id = p.product_id
onlinestore-# JOIN customers c ON o.customer_id = c.customer_id
onlinestore-# WHERE c.name = 'Alice'
onlinestore-# ORDER BY o.order_date;
 order_id | customer | product_name | quantity | order_date
----------+----------+--------------+----------+------------
        1 | Alice    | Laptop       |        1 | 2025-08-01
        3 | Alice    | Shoes        |        1 | 2025-08-03
(2 rows)
```

b) Find products that are out of stock.

```
onlinestore=# SELECT product_id, product_name, stock
onlinestore-# FROM products
onlinestore-# WHERE stock = 0;
 product_id | product_name | stock
------------+--------------+-------
          2 | Mobile       |     0
(1 row)
```

c) Calculate the total revenue generated per product.

```
onlinestore=# SELECT p.product_id, p.product_name, SUM(o.quantity * p.price) AS total_reve
onlinestore-# FROM orders o
onlinestore-# JOIN products p ON o.product_id = p.product_id
onlinestore-# GROUP BY p.product_id, p.product_name
onlinestore-# ORDER BY total_revenue DESC;
 product_id | product_name | total_revenue
------------+--------------+---------------
          1 | Laptop       |      60000.00
          2 | Mobile       |      40000.00
          4 | Watch        |      10000.00
          3 | Shoes        |       3000.00
(4 rows)
```

d) Retrieve the top 5 customers by total purchase amount.

```
onlinestore=# SELECT c.customer_id, c.name, SUM(o.quantity * p.price) AS total_spent
onlinestore-# FROM orders o
onlinestore-# JOIN customers c ON o.customer_id = c.customer_id
onlinestore-# JOIN products p ON o.product_id = p.product_id
onlinestore-# GROUP BY c.customer_id, c.name
onlinestore-# ORDER BY total_spent DESC
onlinestore-# LIMIT 5;
 customer_id |  name   | total_spent
-------------+---------+-------------
           1 | Alice   |    63000.00
           2 | Bob     |    40000.00
           3 | Charlie |    10000.00
(3 rows)
```

e) Find customers who placed orders in at least two different product categories.

```
onlinestore=# SELECT c.customer_id, c.name, COUNT(DISTINCT p.category) AS categories_orde
onlinestore-# FROM orders o
onlinestore-# JOIN customers c ON o.customer_id = c.customer_id
onlinestore-# JOIN products p ON o.product_id = p.product_id
onlinestore-# GROUP BY c.customer_id, c.name
onlinestore-# HAVING COUNT(DISTINCT p.category) >= 2;
 customer_id | name  | categories_ordered
-------------+-------+--------------------
           1 | Alice |                  2
(1 row)
```

Analytics:

a) Find the month with the highest total sales.

```
onlinestore=# SELECT TO_CHAR(order_date, 'YYYY-MM') AS month, SUM(o.quantity * p.price) AS
onlinestore-# FROM orders o
onlinestore-# JOIN products p ON o.product_id = p.product_id
onlinestore-# GROUP BY month
onlinestore-# ORDER BY total_sales DESC
onlinestore-# LIMIT 1;
  month  | total_sales
---------+-------------
 2025-08 |   113000.00
(1 row)
```

b) Identify products with no orders in the last 6 months

```
onlinestore=# SELECT p.product_id, p.product_name
onlinestore-# FROM products p
onlinestore-# WHERE p.product_id NOT IN (
onlinestore(#   SELECT DISTINCT o.product_id FROM orders o
onlinestore(#   WHERE o.order_date >= CURRENT_DATE - INTERVAL '6 months'
onlinestore(# );
 product_id | product_name
------------+--------------
(0 rows)
```

c) Retrieve customers who have never placed an order.

```
onlinestore=# SELECT c.customer_id, c.name, c.email
onlinestore-# FROM customers c
onlinestore-# LEFT JOIN orders o ON c.customer_id = o.customer_id
onlinestore-# WHERE o.order_id IS NULL;
 customer_id | name | email
-------------+------+-------
(0 rows)
```

d) Calculate the average order value across all orders.

```
onlinestore=# SELECT ROUND(AVG(o.quantity * p.price),2) AS avg_order_value
onlinestore-# FROM orders o
onlinestore-# JOIN products p ON o.product_id = p.product_id;
 avg_order_value
-----------------
        28250.00
(1 row)
```