SQL Tasks

Each task is organized by topics such as joins, distinct values, ordering results, limits and offsets, aggregate functions, set operations, and subqueries. You need to refer to the tables provided in the accompanying Excel CSV format while completing these tasks.

---

1. JOINS

Task 1:
Retrieve the customer_name, city, and order_date for each customer who placed an order in 2023 by joining the customers and orders tables.

Task 2:
Get a list of all products (with product_name, category, and total_price) ordered by customers living in Mumbai, by joining the customers, orders, order_items, and products tables.

Task 3:
Find all orders where customers paid using 'Credit Card' and display the customer_name, order_date, and total_price by joining the customers, orders, and order_items tables.

Task 4:
Display the product_name, category, and the total_price for all products ordered in the first half of 2023 (January - June) by joining the orders, order_items, and products tables.

Task 5:
Show the total number of products ordered by each customer, displaying customer_name and total products ordered, using joins between customers, orders, and order_items.

---

2. DISTINCT

Task 1:
Get a distinct list of cities where customers are located.

Task 2:
Retrieve distinct supplier_name from the products table.

Task 3:
Find distinct payment methods used in the orders table.

Task 4:
List all distinct product categories that have been ordered.

Task 5:
Find distinct cities from which suppliers supply products by querying the products table.

---

3. ORDER BY

Task 1:
List all customers sorted by customer_name in ascending order.

Task 2:
Display all orders sorted by total_price in descending order.

Task 3:
Retrieve a list of products sorted by price in ascending order and then by category in descending order.

Task 4:
Sort all orders by order_date in descending order and display the order_id, customer_id, and order_date.

Task 5:
Get the list of cities where orders were placed, sorted in alphabetical order, and display the total number of orders placed in each city.

---

4. LIMIT & OFFSET

Task 1:
Retrieve the first 10 rows from the customers table ordered by customer_name.

Task 2:
Display the top 5 most expensive products (sorted by price in descending order).

Task 3:
Get the orders for the 11th to 20th customers (using OFFSET and LIMIT), sorted by customer_id.

Task 4:
List the first 5 orders placed in 2023, displaying order_id, order_date, and customer_id.

Task 5:
Fetch the next 10 distinct cities where orders were placed, using LIMIT and OFFSET.

---

5. AGGREGATE FUNCTIONS

Task 1:
Calculate the total number of orders placed by all customers.

Task 2:
Find the total revenue generated from orders paid via 'UPI' from the orders table.

Task 3:
Get the average price of all products in the products table.

Task 4:
Find the maximum and minimum total price of orders placed in 2023.

Task 5:
Calculate the total quantity of products ordered for each product_id using the order_items table.

---

6. SET OPERATIONS

Task 1:
Get the list of customers who have placed orders in both 2022 and 2023 (use INTERSECT).

Task 2:
Find the products that were ordered in 2022 but not in 2023 (use EXCEPT).

Task 3:
Display the list of supplier_city from the products table that do not match any customer_city in the customers table (use EXCEPT).

Task 4:
Show a combined list of supplier_city from products and city from customers (use UNION).

Task 5:
Find the list of product_name from the products table that were ordered in 2023 (use INTERSECT with the orders and order_items tables).

---

7. SUBQUERIES

Task 1:
Find the names of customers who placed orders with a total price greater than the average total price of all orders.

Task 2:
Get a list of products that have been ordered more than once by any customer.

Task 3:
Retrieve the product names that were ordered by customers from Pune using a subquery.

Task 4:
Find the top 3 most expensive orders using a subquery.

Task 5:
Get the customer names who placed orders for a product that costs more than ₹30,000 using a subquery.