Project Title: E-Commerce Sales Analysis

Project Overview:
This project involves working with a relational database system to manage customers, orders, products, and payments. The goal is to develop SQL queries to extract insights, manage data efficiently, and understand the relationships between different tables.

Objectives

- Implement database design principles.
- Retrieve and manipulate data using SQL queries.
- Utilize aggregate functions for analytical insights.
- Apply join operations to combine data from multiple tables.
- Use subqueries and advanced filters for efficient data processing.

Database Schema
The project involves the following tables:

1. **Customers**: Stores customer details.
2. **Orders**: Contains order information.
3. **Products**: Lists products available in the store.
4. **Order_Items**: Stores items included in each order.
5. **Payments**: Maintains records of payments made for orders.

Project Deliverables:

1. **Database Setup:** Create the required tables and insert sample data.
2. **Basic Queries:** Retrieve and filter data efficiently.
3. **Aggregations:** Generate summary reports using GROUP BY and HAVING.
4. **Joins & Relationships:** Combine data from multiple tables.
5. **Subqueries & Advanced SQL:** Optimize and refine queries for real-world scenarios.

Conclusion
This project successfully demonstrates SQL database management skills by creating and executing queries to extract meaningful insights. The structured approach to managing data enhances the efficiency and scalability of a relational database system.

Analysis Questions-

1. Basic Data Retrieval (SELECT, WHERE, ORDER BY, LIMIT)

1. Retrieve all records from the **customers** table, displaying all available columns.
2. Fetch only the **customer ID, first name, and email** from the **customers** table.
3. List all **products** that belong to the **Clothing** category.
4. Retrieve all **orders** where the total purchase amount is **greater than $500**.
5. Find all **customers** who joined the platform **after January 1, 2023**.
6. Display the **top 5 most expensive products** available in the database.
7. List the **latest 10 orders** placed, sorted by order date in **descending order**.
8. Retrieve all **orders** that have a status of **"Completed"**.
9. Find all **orders** that were placed between **February 1, 2023, and February 28, 2023**.
10. List all **products** that have a **price between $50 and $100**.

---

2. Aggregate Functions (COUNT, SUM, AVG, MIN, MAX, GROUP BY, HAVING)

1. Count the **total number of customers** in the database.
2. Find the **average order amount** from the orders table.
3. Retrieve the **highest and lowest** priced products from the product list.
4. Count the **number of products in each category**, grouping by category.
5. Calculate the **total revenue** generated from all orders.
6. Find the **total number of orders placed by each customer**, sorted by highest to lowest.
7. Calculate the **total revenue generated for each month** in 2023.
8. List all **customers who have placed more than 5 orders**.
9. Identify the **most frequently used payment method** based on the number of transactions.
10. Find the **average product price** for each category.

---

3. Joins (INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN)

1. Retrieve all **order details along with the customer's first and last name**.
2. Fetch **order items with product names, quantities, and subtotal values**
3. List all **payment transactions along with the corresponding order details**
4. Identify **customers who have never placed an order**
5. Find all **products that have never been purchased** (i.e., do not appear in any order).
6. Retrieve **customers and their total spending** by summing up all their orders.
7. Get the **total number of products ordered by each customer**.

8. Display all **orders along with the names of the products included in each order**.
9. Find **orders that do not have any associated payments** recorded.
10. Retrieve **customers along with the last date they placed an order**.

---

4. Subqueries & Advanced Filters

1. Find the **most expensive product** in the store using a subquery.
2. Retrieve the list of **customers who have placed at least one order**.
3. Display **orders where the total amount is greater than the average order amount**.
4. Find the **cheapest product in each category** using a correlated subquery.
5. Identify **the customer who has placed the highest number of orders**.
6. Fetch the **second most expensive product** using an alternative ranking method.
7. List all **customers who have never made a payment for any order**.
8. Retrieve all **products with stock levels below the average stock quantity**.
9. Find **customers who have spent more than $2000 in total on orders**.
10. Identify employees who **earn the same salary as at least one other employee**.
