PROJECT 1

Select Statement

1. Retrieve all columns from the `products` table.
2. Retrieve only the `product_id` and `Product` columns from the `products` table.
3. Retrieve the `Customer name` and `city` from the `customers` table.

From Statement

4. Write a query to display all columns from the `orders` table.
5. Display the `order_id`, `order_date`, and `total_amount` from the `orders` table.

Where Statement

6. Retrieve all products from the `products` table where the `price` is greater than 500.
7. Find all customers from the `customers` table who live in "Houston".
8. Retrieve all orders from the `orders` table where the `total_amount` is less than 1000.
9. Find all products in the `products` table that belong to the "Electronics" category.

Group By and Order By

10. Count the number of products in each category from the `products` table and group the results by `Category`.
11. Retrieve the total number of orders placed by each customer from the `orders` table, grouped by `customer_id`.
12. Display the average `price` of products in each category, sorted by the average price in descending order.
13. Find the total `quantity` of each product sold from the `order_items` table, grouped by `product_id`.

Using Having vs. Where Statement

14. Retrieve categories from the `products` table where the average price is greater than 500 (use `GROUP BY` and `HAVING`).
15. Find customers from the `customers` table who are in the "Young" age bracket (use `WHERE`).
16. Retrieve products from the `products` table where the price is greater than 300 and the category is "Accessories" (use `WHERE`).
17. Display categories from the `products` table that have more than 5 products (use `GROUP BY` and `HAVING`)

PROJECT 1

Limit and Aliasing

18. Retrieve the top 5 most expensive products from the `products` table (use `LIMIT`).
19. Display the first 10 orders from the `orders` table, sorted by `order_date` in ascending order.
20. Retrieve the `product_id` and `Product` from the `products` table, and alias them as `ID` and `Product Name`.
21. Find the top 3 customers with the highest `total_amount` spent from the `orders` table.

Joins in MySQL

22. Retrieve the `order_id`, `order_date`, and `Customer name` by joining the `orders` and `customers` tables.
23. Display the `product_id`, `Product`, and `quantity` sold by joining the `products` and `order_items` tables.
24. Find the total revenue generated by each product by joining the `products` and `order_items` tables.
25. Retrieve the `Customer name`, `order_date`, and `total_amount` by joining the `customers` and `orders` tables.

Unions in MySQL

26. Retrieve a list of all unique cities from the `customers` table and combine it with a list of all unique categories from the `products` table (use `UNION`).
27. Combine the `product_id` from the `products` table with the `order_id` from the `orders` table (use `UNION`).

Case Statements

28. Create a new column in the `products` table called `Price Range` that categorizes products as "Low" (price < 300), "Medium" (price between 300 and 700), and "High" (price > 700) using a `CASE` statement.
29. Use a `CASE` statement to categorize customers in the `customers` table as "Young" (age <= 33), "Working Class" (age between 34 and 49), and "Retired" (age >= 50).
30. Retrieve the `order_id` and a new column called `Order Size` that categorizes orders as "Small" (total_amount < 500), "Medium" (total_amount between 500 and 1000), and "Large" (total_amount > 1000) using a `CASE` statement.

PROJECT 1

Intermediate Challenges

31. Find the top 5 customers who have spent the most money in total (join `customers` and `orders` tables).

32. Retrieve the `Product` name, `Category`, and total revenue generated by each product (join `products` and `order_items` tables).

33. Display the `Customer name`, `order_date`, and `total_amount` for orders placed in January 2023 (use `WHERE` with date filtering).

34. Find the average `total_amount` of orders for each customer, and display only those customers whose average order amount is greater than 1000 (use `GROUP BY` and `HAVING`).

35. Retrieve the `Customer name`, `Product`, and `quantity` for all orders placed by customers in "New York" (join `customers`, `orders`, and `order_items` tables).

36. Find customers who placed orders in both January and February 2023.