Case Study:-

Given a schema with below table details and schema relations, Design the schema, table and find the answers to the below questions set with queries.

Tables:

Customers: Contains customer information (customer\_id, name, email, country)

Orders: Contains order details (order\_id, customer\_id, order\_date, total\_amount)

Order\_Items: Contains items in each order (order\_id, product\_id, quantity, unit\_price)

Products: Contains product information (product\_id, product\_name, category\_id)

Categories: Contains product categories (category\_id, category\_name)

Customers - customer\_id(primary key)

Orders - order\_id(primary key) customer\_id(foreign key to customer table)

Order\_Items - order\_id(foreign key to Orders table), product\_id(primary key)

Products - product\_id(foreign key to order\_items table), category\_id(primary key)

Categories - category\_id(foreign key to Products table)

Questions Set 1:

Basic Retrieval:

● Retrieve the names and countries of all customers.

Order Aggregation:

● Find the total number of orders made.

Join and Aggregation:

● List the top 5 customers who made the most orders along with their total order count.

Order by and Limit:

● Retrieve the top 10 orders by the total order amount.

Group by and Aggregate Function:

● Calculate the average order amount per customer.

Join and Subquery:

● Get the list of products in the 'Electronics' category.

Subquery and Aggregation:

● Find the customer(s) who spent the most in a single order.

Group by with Having:

● List countries with more than 5 customers.

Window Functions:

● Rank customers based on their total order amount.

Window Functions with Partitioning:

● Calculate the running total order amount for each customer.

Nested Subqueries:

● Retrieve the orders made by customers from 'Germany'.

Multiple Joins:

● List order details with customer name, order date, product name, and quantity.

Using Window Function with Joins:

● Display the average order amount in each category.

Combining Aggregate Functions with Window Functions:

● Find the customer with the highest total order amount, along with their average order amount.

Subqueries with Multiple Conditions:

● Retrieve customers who have made orders greater than the average order amount.

Questions Set 2:

Using ROW\_NUMBER with Joins:

● Apply ROW\_NUMBER() within each category to enumerate products, displaying product details alongside their row numbers.

Ranking and Joining:

● Rank customers based on their total order amount within their respective countries, displaying customer details alongside their rankings.

Using LAG/LEAD with Joins:

● Retrieve the details of products and their previous and next products within the same category, ordered by product\_id.

Partitioning with Aggregation and Joins:

● Calculate the average order amount for each customer, comparing it to the average order amount within their respective countries.

Using NTILE with Joins:

● Partition customers into quartiles based on their total order amount within each country, displaying customer details along with their quartile rankings.

Preceding and Following with Joins:

● Calculate the sum of order amounts for each customer within a specified time frame preceding and following each order, alongside order details.

Using RANK with Different Joins:

● Rank products within each category based on their sales quantities, displaying product details and their rankings.

Comparing Aggregate Values with Joins:

● Compare the total order amount of each customer with the average total order amount within their country, displaying customer details and their comparison results.

Analyzing Trends with Window Functions and Joins:

● Display the difference in order amounts for each customer between their first and last orders, including customer details.

Combining Multiple Window Functions with Joins:

● Calculate the cumulative sum of order amounts for each customer, alongside their average order amount within their country, in separate columns.