**--Task**

**--1. Filter and Aggregate on Join Results using SQL\*\***

**--Task: Join the `Orders` and `Customers` tables to find the total order amount per customer and filter out customers who have spent less than $1,000.**

SELECT c.CustomerID,c.FirstName,c.LastName,SUM(o.Amount) AS TotalSpent

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName

HAVING SUM(o.Amount) >= 1000;

**--2. Cumulative Aggregations and Ranking in SQL Queries\*\***

**--Task: Create a cumulative sum of the `OrderAmount` for each customer to track the running total of how much each customer has spent.**

SELECT c.CustomerID,c.FirstName,c.LastName,o.OrderID,o.Amount,

SUM(o.Amount) OVER (PARTITION BY c.CustomerID ORDER BY o.OrderID) AS RunningTotal

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

ORDER BY c.CustomerID, o.OrderID;

**--3. OVER and PARTITION BY Clause in SQL Queries\*\***

**--Task: Rank the customers based on the total amount they have spent, partitioned by city.**

ALTER TABLE Customers

ADD City VARCHAR(100);

UPDATE Customers

SET City = 'Chennai'

WHERE CustomerID IN (1, 3);

UPDATE Customers

SET City = 'Bangalore'

WHERE CustomerID IN (2, 4);

UPDATE Customers

SET City = 'Coimbatore'

WHERE CustomerID = 5;

UPDATE Customers

SET City = 'Pondicherry'

WHERE CustomerID = 6;

UPDATE Customers

SET City = 'Mumbai'

WHERE CustomerID = 6;

SELECT c.CustomerID,c.FirstName,c.LastName,c.City,

SUM(o.Amount) AS TotalSpent,

RANK() OVER (PARTITION BY c.City ORDER BY SUM(o.Amount) DESC) AS SpendingRank

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName, c.City

ORDER BY c.City, SpendingRank;

**--4. Total Aggregation using OVER and PARTITION BY in SQL Queries\*\***

**--Task: Calculate the total amount of all orders (overall total) and the percentage each customer's total spending contributes to the overall total.**

WITH CustomerTotalSpending AS (

SELECT c.CustomerID,c.FirstName,c.LastName,c.City,

SUM(o.Amount) AS TotalSpent

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName, c.City),

OverallTotal AS (SELECT

SUM(TotalSpent) AS OverallTotalSpent

FROM CustomerTotalSpending

)

SELECT c.CustomerID,c.FirstName,c.LastName,c.City,c.TotalSpent,o.OverallTotalSpent,

(c.TotalSpent \* 100.0 / o.OverallTotalSpent) AS SpendingPercentage

FROM CustomerTotalSpending c

CROSS JOIN OverallTotal o

ORDER BY SpendingPercentage DESC;

**--5. Ranking in SQL\*\***

**--Task: Rank all customers based on the total amount they have spent, without partitioning.**

SELECT c.CustomerID,c.FirstName,c.LastName,c.City,

SUM(o.Amount) AS TotalSpent,

RANK() OVER (ORDER BY SUM(o.Amount) DESC) AS SpendingRank

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName, c.City

ORDER BY SpendingRank;

--additional tasks that build on the concepts of filtering, aggregating, ranking, and window functions in SQL:

**--6. Task: Calculate the Average Order Amount per City\*\***

**--Task: Write a query that joins the `Orders` and `Customers` tables, calculates the average order amount for each city, and orders the results by the average amount in descending order.**

SELECT c.City,AVG(o.Amount) AS AverageOrderAmount

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.City

ORDER BY AverageOrderAmount DESC;

**--7. Task: Find Top N Customers by Total Spending\*\***

**--Task: Write a query to find the top 3 customers who have spent the most, using `ORDER BY` and `LIMIT`.**

SELECT TOP 3 c.CustomerID,c.FirstName,c.LastName,c.City,

SUM(o.Amount) AS TotalSpent

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName, c.City

ORDER BY TotalSpent DESC;

**--8. Task: Calculate Yearly Order Totals\*\***

**--Task: Write a query that groups orders by year (using `OrderDate`), calculates the total amount of orders for each year, and orders the results by year.**

SELECT YEAR(o.OrderDate) AS OrderYear,SUM(o.Amount) AS TotalAmount

FROM Orders o

GROUP BY YEAR(o.OrderDate)

ORDER BY OrderYear;

**--9. Task: Calculate the Rank of Customers by Total Order Amount\*\***

**--Task: Write a query that ranks customers by their total spending, but only for customers located in "Mumbai". The rank should reset for each customer in "Mumbai".**

SELECT c.CustomerID, c.FirstName, SUM(o.Amount) AS TotalSpent,

RANK() OVER (ORDER BY SUM(o.Amount) DESC) AS CustomerRank

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

WHERE c.City = 'Mumbai'

GROUP BY c.CustomerID, c.FirstName;

**--10. Task: Compare Each Customer's Total Order to the Average Order Amount\*\***

**--Task: Write a query that calculates each customer's total order amount and compares it to the average order amount for all customers.**

SELECT c.CustomerID, c.FirstName, c.LastName,

SUM(o.Amount) AS TotalSpent,

SUM(o.Amount) - AVG(SUM(o.Amount)) OVER () AS Comparision

FROM Customers c

INNER JOIN Orders o ON c.CustomerID = o.CustomerID

GROUP BY c.CustomerID, c.FirstName, c.LastName

ORDER BY TotalSpent DESC;