**HANDS ON EXERCISES (24.08.2024)**

Filter and Aggregate on Join Results using SQL

1.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, SUM(O.TotalAmount) AS TotalSpent

FROM CustomerTable C

JOIN Orders O ON C.CustomerId = O.CustomerId

GROUP BY C.CustomerId

HAVING SUM(O.TotalAmount) >= 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

CustomerID, OrderID, OrderDate, TotalAmount,

SUM(TotalAmount) OVER (PARTITION BY CustomerID ORDER BY OrderDate) AS TotalCumulative

FROM Orders

ORDER BY CustomerID, OrderDate;

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

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

SELECT

c.CustomerId,

Firstname,

City,

SUM(TotalAmount) AS TotalSpent,

RANK() OVER (PARTITION BY City ORDER BY SUM(TotalAmount) DESC) AS CustomerRank

FROM CustomerTable C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY c.CustomerId, Firstname, City

ORDER BY City, CustomerRank;

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.

SELECT

C.CustomerId,Firstname,

SUM(O.TotalAmount) AS CustomerTotal,

SUM(SUM(O.TotalAmount)) OVER () AS OverallTotal,

(SUM(O.TotalAmount) \* 100.0) / SUM(SUM(O.TotalAmount)) OVER () AS PercentageOfOverall

FROM CustomerTable C

JOIN Orders O ON C.CustomerID = O.CustomerID

GROUP BY C.CustomerID, Firstname;

5. Ranking in SQL\*\*

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

SELECT

CustomerID,

SUM(TotalAmount) AS CustomerTotal,

RANK() OVER (ORDER BY SUM(TotalAmount) DESC) AS CustomerRank

FROM Orders

GROUP BY CustomerID

ORDER BY CustomerRank;

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

City,

AVG(O.TotalAmount) AS AverageOrderAmount

FROM Orders O

JOIN CustomerTable C ON O.CustomerID = C.CustomerID

GROUP BY 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

CustomerID,

SUM(TotalAmount) AS CustomerTotal

FROM Orders

GROUP BY CustomerID

ORDER BY CustomerTotal 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(OrderDate) AS Year,

SUM(TotalAmount) AS TotalAmount

FROM Orders

GROUP BY YEAR(OrderDate)

ORDER BY Year;

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,

SUM(TotalAmount) AS CustomerTotal,

RANK() OVER (ORDER BY SUM(TotalAmount) DESC) AS CustomerRank

FROM Orders o

JOIN CustomerTable c ON o.CustomerID = c.CustomerId

WHERE City = 'Mumbai'

GROUP BY c.CustomerId

ORDER BY CustomerRank;

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.

WITH CustomerTotals AS (

SELECT

CustomerID,

SUM(TotalAmount) AS CustomerTotal

FROM Orders

GROUP BY CustomerID ),

AverageOrder AS (

SELECT

AVG(CustomerTotal) AS AvgOrderAmount

FROM CustomerTotals

)

SELECT

CT.CustomerID,

CT.CustomerTotal,

AO.AvgOrderAmount,

CT.CustomerTotal - AO.AvgOrderAmount AS DifferenceFromAverage

FROM CustomerTotals CT

CROSS JOIN AverageOrder AO

ORDER BY CT.CustomerTotal DESC;