EXERCISE - 2

1. 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.

Query:
```
SELECT c.CustomerID, c.FirstName, c.LastName, SUM(o.TotalAmount) AS
TotalSpent
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.CustomerID, c.FirstName, c.LastName
HAVING SUM(o.TotalAmount) >= 1000;
```

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

Query:
```
SELECT o.CustomerID, c.FirstName, c.LastName, o.OrderDate, o.TotalAmount,
SUM(o.TotalAmount) OVER (PARTITION BY o.CustomerID ORDER BY
o.OrderDate) AS RunningTotal
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID;
```

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

Query:
```
SELECT c.CustomerID, c.City, SUM(o.TotalAmount) AS TotalSpent,
RANK() OVER (PARTITION BY c.City ORDER BY SUM(o.TotalAmount) DESC) AS
CustomerRank
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
GROUP BY c.CustomerID, c.City;
```

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

Query: WITH CustomerTotals AS (

SELECT c.CustomerID, SUM(o.TotalAmount) AS TotalSpent

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

GROUP BY c.CustomerID,

)

SELECT CustomerID, TotalSpent,

TotalSpent * 100.0 / SUM(TotalSpent) OVER () AS PercentageOfTotal

FROM CustomerTotals;

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

Query: SELECT c.CustomerID, c.FirstName, SUM(o.TotalAmount) AS TotalSpent,

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

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

GROUP BY c.CustomerID, c.FirstName;

6. 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.

Query: SELECT c.City, AVG(o.TotalAmount) AS AvgOrderAmount

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

GROUP BY c.City

ORDER BY AvgOrderAmount DESC;

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

Query: SELECT c.CustomerID, c.FirstName, SUM(o.TotalAmount) AS TotalSpent

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

GROUP BY c.CustomerID, c.FirstName,

ORDER BY TotalSpent DESC

LIMIT 3;

8. 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.

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

FROM Orders o

GROUP BY YEAR(o.OrderDate)

ORDER BY OrderYear;

9. 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".

Query: SELECT c.CustomerID, c.FirstName, SUM(o.TotalAmount) AS TotalSpent,

RANK() OVER (ORDER BY SUM(o.TotalAmount) 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. Write a query that calculates each customer's total order amount and compares it to the average order amount for all customers.

Query: SELECT c.CustomerID, c.FirstName, c.LastName,

SUM(o.TotalAmount) AS TotalSpent,

SUM(o.TotalAmount) - AVG(SUM(o.TotalAmount)) OVER () AS DifferenceFromAvg

FROM Orders o

JOIN Customers c ON o.CustomerID = c.CustomerID

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