**EXERCISE**

**Day4 - SQL**

**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.**

select c.customerid, sum(o.orderamount) as totalorderamount

from customers c

join orders o on c.customerid = o.customerid

group by c.customerid

having sum(o.orderamount) >= 1000;

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

select c.customer\_id, c.firstname, c.lastname, o.orderdate, o.orderamount,

sum(o.orderamount) over (partition by c.customer\_id order by o.orderdate) as runningtotal

from customers c

join orders o on c.customer\_id = o.customer\_id

order by c.customer\_id, o.orderdate;

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

select c.customerid, c.city, sum(o.orderamount) as totalspending,

rank() over (partition by c.city order by sum(o.orderamount) desc) as cityrank

from customers c

join orders o on c.customerid = o.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.**

select c.customerid, sum(o.orderamount) as totalspending,

(sum(o.orderamount) / sum(sum(o.orderamount)) over ()) \* 100 as percentageoftotal

from customers c

join orders o on c.customerid = o.customerid

group by c.customerid;

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

select c.customer\_id, c.firstname, c.lastname,

sum(o.orderamount) as totalspending,

rank() over (order by sum(o.orderamount) desc) as overallrank

from customers c

join orders o on c.customer\_id = o.customer\_id

group by c.customer\_id, c.firstname, c.lastname;

**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.**

select c.city, avg(o.orderamount) as averageorderamount

from customers c

join orders o on c.customerid = o.customerid

group by c.city

order by avg(o.orderamount) desc;

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

select top 3 c.customer\_id, c.firstname, c.lastname,

sum(o.orderamount) as totalspent

from customers c

join orders o on c.customer\_id = o.customer\_id

group by c.customer\_id, c.firstname, c.lastname

order by totalspent desc;

**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.**

select year(o.orderdate) as orderyear, sum(o.orderamount) as totalorderamount

from orders o

group by year(o.orderdate)

order by year(o.orderdate);

**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".**

select c.customer\_id, c.firstname, c.lastname,

sum(o.orderamount) as totalspent,

rank() over (order by sum(o.orderamount) desc) as mumbairank

from customers c

join orders o on c.customer\_id = o.customer\_id

where c.city = 'mumbai'

group by c.customer\_id, c.firstname, c.lastname;

**10. 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, sum(o.orderamount) as totalspending,

avg(o.orderamount) over () as averageorderamount,

case when sum(o.orderamount) > avg(o.orderamount) over () then 'above average'

when sum(o.orderamount) = avg(o.orderamount) over () then 'average'

else 'below average' end as spendingcomparison

from customers c

join orders o on c.customerid = o.customerid

group by c.customerid;