DNA ASSIGNEMENT	<u>S DHAANESH</u>
SQL QUERY & LOGIC	MYSQL

Q1. Customers who don't order frequently

LOGIC

STEP: 1 (Identify which AGGREGATION functions to be performed)

- COUNT Total no of orders placed by each customer
- AVG Compare the order count to avg of order count, ie order count < avg count, then the person is tagged as the one who orders less frequently.

STEP: 2 (Which Columns are required)

- CUSTOMERID, CUSTOMERNAME from customer table
- ORDERID from orderinfo table

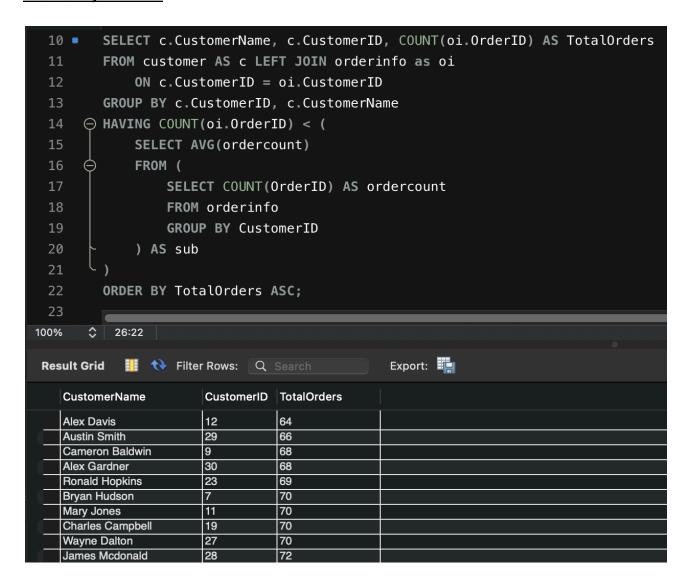
STEP: 3 (Which tables to JOIN)

• JOIN CUSTOMER and ORDERINFO on customerID

STEP : 4 (GROUP BY and ORDER BY to be used ? where ?)

- GROUP BY CUSTOMERID, CUSTOMERNAME to get order counts per customer
- ORDER BY TOTALORDERS to show the least frequent first

SQL Query & Result



Q2. Frequency of Ordering

LOGIC

STEP: 1 (Identify which AGGREGATION functions to be performed)

• COUNT – Total number of orders placed by each customer per year.

STEP: 2 (Which Columns are required)

• CUSTOMERID, CUSTOMERNAME from customer table

• ORDERDATE, ORDERID from orderinfo table

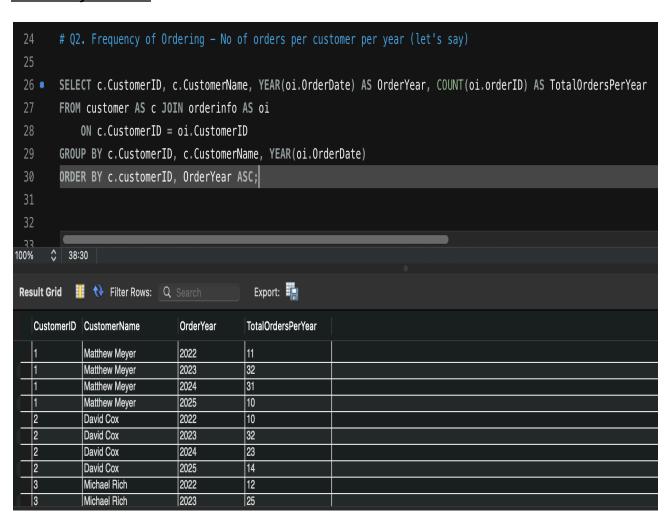
STEP: 3 (Which tables to JOIN)

• JOIN customer and orderinfo ON CustomerID

STEP: 4 (GROUP BY and ORDER BY to be used? where?)

- GROUP BY CUSTOMERID, CUSTOMERNAME, YEAR(ORDERDATE) → to get yearly frequency per customer
- ORDER BY CUSTOMERID, ORDERYEAR → so orders are shown chronologically for each customer

SQL Query & Result



Q3. Correlation between Discount and Sales

LOGIC

<u>STEP: 1</u> (Identify which AGGREGATION functions to be performed)

- SUM Total sales (LineTotal) for each discount level.
- Calculate correlation coefficient between discount and sales in a statistical tool.

STEP: 2 (Which Columns are required)

• DISCOUNT, LINETOTAL from **orderdetails** table

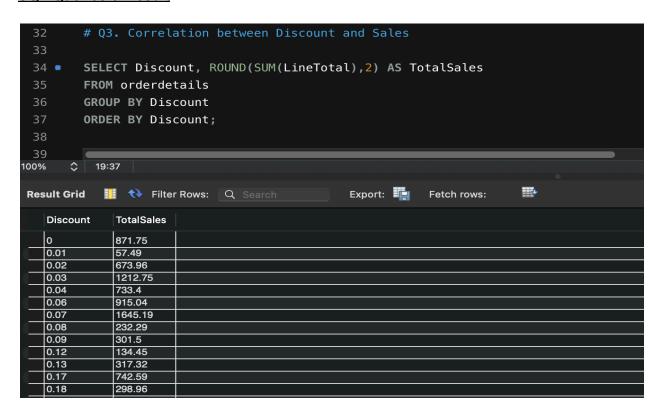
STEP: 3 (Which tables to JOIN)

• No join required – data is entirely available in **orderdetails** table.

STEP: 4 (GROUP BY and ORDER BY to be used? where?)

- GROUP BY DISCOUNT → to get sales total per discount value
- ORDER BY DISCOUNT → so results are shown in ascending discount order

SQL Queries & Result



Q4. Region Analysis

LOGIC

<u>STEP: 1</u> (Identify which AGGREGATION functions to be performed)

- COUNT Number of orders from each region
- SUM Total sales (TotalAmount) from each region

STEP: 2 (Which Columns are required)

- PRIMARYADDRESS from **customer** table (to extract region/state)
- ORDERID, TOTALAMOUNT from orderinfo table

STEP: 3 (Which tables to JOIN)

• JOIN customer and orderinfo ON CustomerID

STEP: 4 (GROUP BY and ORDER BY to be used? where?)

- GROUP BY REGION (extracted from PRIMARYADDRESS) → to analyze orders and sales per region
- ORDER BY TOTALSALES (descending) → to list highest performing regions first

SQL Query & Result

```
SUBSTRING_INDEX(SUBSTRING_INDEX(c.PrimaryAddress, ',', -1), ' ', 2) AS Region,
            COUNT(o.OrderID) AS TotalOrders,
            ROUND(SUM(o.TotalAmount),2) AS TotalSales
        FROM customer AS c
        JOIN orderinfo AS o
             ON c.CustomerID = o.CustomerID
        GROUP BY Region
        ORDER BY TotalSales DESC;
                                                  Export:
Result Grid
           III 🛟 Filter Rows: Q Search
  Region
                TotalOrders
                              TotalSales
   GA
                230
                              330201.17
   DPO
                224
                              310170.95
                              236708.58
  н
                170
   APO
                166
                              233575.33
   WI
                154
                              222229.03
   МТ
                146
                              214806.11
   NE
                148
                              214289.57
                146
                              207909.14
   WY
                86
                              124356.45
                              123133.86
                              117937.35
                75
```

5. Product based on review

LOGIC

STEP: 1 (Identify which AGGREGATION functions to be performed)

- AVG Average review rating for each product
- COUNT Number of reviews per product (to measure popularity)

STEP: 2 (Which Columns are required)

- PRODUCTID, PRODUCTNAME from product table
- RATING from review table

STEP: 3 (Which tables to JOIN)

JOIN product and review ON ProductID

STEP: 4 (GROUP BY and ORDER BY to be used? where?)

- GROUP BY PRODUCTID, PRODUCTNAME → to get metrics per product
- ORDER BY AVERAGERATING (descending) → to show best-rated products first
- Then ORDER BY REVIEWCOUNT to highlight most-reviewed products

SQL Query and Result

```
# Q5. Product based on review - need to add a column of rating to the table

SELECT

p.ProductID,
p.ProductName,
ROUND(AVG(r.Rating), 2) AS AverageRating,
COUNT(r.Rating) AS ReviewCount

FROM
product p

JOIN

review r ON p.ProductID = r.ProductID

GROUP BY
p.ProductID, p.ProductName

ORDER BY

AverageRating DESC, ReviewCount DESC;
```