DNA ASSIGNEMENT	**S DHAANESH**
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

```sql
SELECT c.CustomerName, c.CustomerID, COUNT(oi.OrderID) AS TotalOrders
FROM customer AS c LEFT JOIN orderinfo as oi
    ON c.CustomerID = oi.CustomerID
GROUP BY c.CustomerID, c.CustomerName
HAVING COUNT(oi.OrderID) < (
    SELECT AVG(ordercount)
    FROM (
        SELECT COUNT(OrderID) AS ordercount
        FROM orderinfo
        GROUP BY CustomerID
    ) AS sub
)
ORDER BY TotalOrders ASC;
```

100% 26:22

Result Grid Filter Rows: Search Export:

CustomerName	CustomerID	TotalOrders
Alex Davis	12	64
Austin Smith	29	66
Cameron Baldwin	9	68
Alex Gardner	30	68
Ronald Hopkins	23	69
Bryan Hudson	7	70
Mary Jones	11	70
Charles Campbell	19	70
Wayne Dalton	27	70
James Mcdonald	28	72

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

```
24  # Q2. Frequency of Ordering - No of orders per customer per year (let's say)
25
26  SELECT c.CustomerID, c.CustomerName, YEAR(oi.OrderDate) AS OrderYear, COUNT(oi.orderID) AS TotalOrdersPerYear
27  FROM customer AS c JOIN orderinfo AS oi
28      ON c.CustomerID = oi.CustomerID
29  GROUP BY c.CustomerID, c.CustomerName, YEAR(oi.OrderDate)
30  ORDER BY c.customerID, OrderYear ASC;
31
32
33
```

100% 38:30

Result Grid | Filter Rows: Search | Export:

CustomerID	CustomerName	OrderYear	TotalOrdersPerYear
1	Matthew Meyer	2022	11
1	Matthew Meyer	2023	32
1	Matthew Meyer	2024	31
1	Matthew Meyer	2025	10
2	David Cox	2022	10
2	David Cox	2023	32
2	David Cox	2024	23
2	David Cox	2025	14
3	Michael Rich	2022	12
3	Michael Rich	2023	25

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.

<u>STEP : 2</u> (Which Columns are required)

- DISCOUNT, LINETOTAL from **orderdetails** table

<u>STEP : 3</u> (Which tables to JOIN)

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

<u>STEP : 4</u> (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

<u>SQL Queries & Result</u>

```
32    # Q3. Correlation between Discount and Sales
33
34    SELECT Discount, ROUND(SUM(LineTotal),2) AS TotalSales
35    FROM orderdetails
36    GROUP BY Discount
37    ORDER BY Discount;
38
39
```

100% 19:37

Result Grid | Filter Rows: Search | Export: | Fetch rows:

Discount	TotalSales
0	871.75
0.01	57.49
0.02	673.96
0.03	1212.75
0.04	733.4
0.06	915.04
0.07	1645.19
0.08	232.29
0.09	301.5
0.12	134.45
0.13	317.32
0.17	742.59
0.18	298.96

Q4. Region Analysis

LOGIC

STEP : 1 (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

```
39    # Q4. Region Analysis
40
41 •  SELECT
42        SUBSTRING_INDEX(SUBSTRING_INDEX(c.PrimaryAddress, ',', -1), ' ', 2) AS Region,
43        COUNT(o.OrderID) AS TotalOrders,
44        ROUND(SUM(o.TotalAmount),2) AS TotalSales
45    FROM customer AS c
46    JOIN orderinfo AS o
47        ON c.CustomerID = o.CustomerID
48    GROUP BY Region
49    ORDER BY TotalSales DESC;
50
```

100% 1:50

Result Grid Filter Rows: Search Export:

Region	TotalOrders	TotalSales
GA	230	330201.17
DPO	224	310170.95
HI	170	236708.58
APO	166	233575.33
WI	154	222229.03
MT	146	214806.11
NE	148	214289.57
CT	146	207909.14
WY	86	124356.45
WV	86	123133.86
LA	79	117937.35
TX	75	116193.04
NV	70	112876.16

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

```sql
# 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;
```