**SQL Query used;**

**1.Average Delivery Time**

This query calculates the average delivery time (in days) by comparing the Order Date and Delivery Date.

SELECT Customer Key, AVG(DATEDIFF(S.Delivery Date, S.OrderDate)) AS AverageDeliveryTime

FROM Cleaned\_Sales S

GROUP BY CustomerKey;

DATEDIFF(S.DeliveryDate, S.OrderDate):

This calculates the difference in days between when an order was placed and when it was delivered.

AVG(...):

Averages the delivery time for each customer.

**2.Order Rate by Product Category**

This query calculates the number of orders for each product category.

SELECT P.Category, COUNT(S.OrderNumber) AS OrderCount

FROM Cleaned\_Sales S

JOIN Cleaned\_Products P ON S.ProductKey = P.ProductKey

GROUP BY P.Category;

COUNT(S.OrderNumber):

Counts the number of orders made in each product category.

JOIN:

Joins Cleaned\_Sales and Cleaned\_Products on ProductKey.

**3. Total Revenue by Store Size (Square Meters)**

This query calculates total revenue based on the store size in square meters.

SELECT Store.SquareMeters, SUM(S.Quantity \* P.UnitPriceUSD) AS TotalRevenue

FROM Cleaned\_Sales S

JOIN Cleaned\_Stores Store ON S.StoreKey = Store.StoreKey

JOIN Cleaned\_Products P ON S.ProductKey = P.ProductKey

GROUP BY Store.SquareMeters;

SUM(S.Quantity \* P.UnitPriceUSD):

Multiplies the quantity sold by the unit price to get total revenue.

GROUP BY Store.SquareMeters:

Aggregates by store size in square meters.

**4. Acquisition Month**

This query retrieves the acquisition month based on the first purchase date of each customer.

SELECT CustomerKey, DATE\_FORMAT(FirstPurchaseDate, '%Y-%m') AS AcquisitionMonth

FROM Cleaned\_Customers;

DATE\_FORMAT(FirstPurchaseDate, '%Y-%m'):

Formats the acquisition month as YYYY-MM.

**5. Churned Status**

This query retrieves the churn status for each customer.

SELECT CustomerKey, Churned

FROM Cleaned\_Customers;

Churned:

1 means the customer is churned, and 0 means the customer is retained.

**6. First Purchase Date**

This query retrieves the first purchase date of each customer.

SELECT CustomerKey, FirstPurchaseDate

FROM Cleaned\_Customers;

FirstPurchaseDate:

The date when the customer made their first purchase.

**7. Frequency (Number of Orders)**

This query calculates the frequency of orders made by each customer.

SELECT CustomerKey, COUNT(S.OrderNumber) AS Frequency

FROM Cleaned\_Sales S

GROUP BY CustomerKey;

COUNT(S.OrderNumber):

Counts the total number of orders for each customer.

GROUP BY CustomerKey:

Groups by the customer to get the total number of orders.

**8. Months Since Acquisition**

This query calculates the number of months that have passed since the customer’s first purchase.

SELECT CustomerKey, TIMESTAMPDIFF(MONTH, FirstPurchaseDate, CURDATE()) AS MonthsSinceAcquisition

FROM Cleaned\_Customers;

TIMESTAMPDIFF(MONTH, FirstPurchaseDate, CURDATE()):

Calculates the difference in months from the first purchase date to today.

**9. Spending Category**

This query segments customers into spending categories based on their total spend.

SELECT CustomerKey,

CASE

WHEN SUM(S.Quantity \* P.UnitPriceUSD) < 100 THEN 'Low Spend'

WHEN SUM(S.Quantity \* P.UnitPriceUSD) BETWEEN 100 AND 499 THEN 'Medium Spend'

ELSE 'High Spend'

END AS SpendingCategory

FROM Cleaned\_Sales S

JOIN Cleaned\_Products P ON S.ProductKey = P.ProductKey

GROUP BY CustomerKey;

CASE:

Categorizes customers as Low Spend, Medium Spend, or High Spend based on their total spend.

SUM(S.Quantity \* P.UnitPriceUSD):

Calculates total spend per customer.

**10. Total Loss**

This query calculates the total loss, where the product cost exceeds the selling price.

SELECT P.Category,

SUM(CASE

WHEN (S.Quantity \* P.UnitCostUSD) > (S.Quantity \* P.UnitPriceUSD)

THEN (S.Quantity \* P.UnitCostUSD) - (S.Quantity \* P.UnitPriceUSD)

ELSE 0

END) AS TotalLoss

FROM Cleaned\_Sales S

JOIN Cleaned\_Products P ON S.ProductKey = P.ProductKey

GROUP BY P.Category;

SUM(...):

Calculates the total loss by summing the differences where the cost is greater than the selling price.

GROUP BY P.Category:

Groups the total loss by product category.

**11. Acquisition Month (Calculated Column)**

The Acquisition Month is based on the First Purchase Date.

SELECT CustomerKey,

DATE\_FORMAT(FirstPurchaseDate, '%Y-%m') AS AcquisitionMonth

FROM Cleaned\_Customers;

DATE\_FORMAT(FirstPurchaseDate, '%Y-%m'):

Formats the date into year-month format (YYYY-MM).

**12. Churned (Calculated Column)**

If not already existing, a calculated Churned column can be derived based on your criteria for determining churn (e.g., no activity for a certain period).

SELECT CustomerKey,

CASE

WHEN TIMESTAMPDIFF(MONTH, MAX(S.OrderDate), CURDATE()) > 12 THEN 1 -- Assume churned if no purchase in last 12 months

ELSE 0

END AS Churned

FROM Cleaned\_Customers C

LEFT JOIN Cleaned\_Sales S ON C.CustomerKey = S.CustomerKey

GROUP BY CustomerKey;

TIMESTAMPDIFF:

Measures how many months have passed since the last order to mark churned status.

**13. First Purchase Date (Calculated Column)**

The First Purchase Date can be calculated as the earliest order date for each customer.

SELECT CustomerKey,

MIN(S.OrderDate) AS FirstP urchaseDate

FROM Cleaned\_Sales S

GROUP BY CustomerKey;

MIN(S.OrderDate):

Retrieves the first purchase (order) date for each customer.

**14. Frequency (Order Count, Calculated Column)**

This calculated column counts the number of orders (frequency) for each customer.

SELECT CustomerKey,

COUNT(S.OrderNumber) AS Frequency -- Number of orders

FROM Cleaned\_Sales S

GROUP BY CustomerKey;

COUNT(S.OrderNumber):

Counts the number of orders a customer has made.

**15. Months Since Acquisition (Calculated Column)**

This column calculates how many months have passed since the First Purchase Date.

SELECT CustomerKey,

TIMESTAMPDIFF(MONTH, FirstPurchaseDate, CURDATE()) AS MonthsSinceAcquisition

FROM Cleaned\_Customers;

TIMESTAMPDIFF(MONTH, FirstPurchaseDate, CURDATE()):

Returns the difference in months between the first purchase date and today.

**16. Spending Category (Calculated Column)**

This calculated column segments customers into Low Spend, Medium Spend, and High Spend based on their total spend.

SELECT CustomerKey,

CASE

WHEN SUM(S.Quantity \* P.UnitPriceUSD) < 100 THEN 'Low Spend'

WHEN SUM(S.Quantity \* P.UnitPriceUSD) BETWEEN 100 AND 499 THEN 'Medium Spend'

ELSE 'High Spend'

END AS SpendingCategory

FROM Cleaned\_Sales S

JOIN Cleaned\_Products P ON S.ProductKey = P.ProductKey

GROUP BY CustomerKey;

CASE:

Assigns customers into spend categories based on their total spend.

**17. Total Sales (Calculated Column)**

If you need a Total Sales column for each order or customer, you can create it like this:

SELECT CustomerKey,

SUM(S.Quantity \* P.UnitPriceUSD) AS TotalSales

FROM Cleaned\_Sales S

JOIN Cleaned\_Products P ON S.ProductKey = P.ProductKey

GROUP BY CustomerKey;

SUM(S.Quantity \* P.UnitPriceUSD):

Sums up the total sales per customer.

**18. Profit Contribution (Calculated Column)**

You can also create a column for Profit Contribution to calculate the profit made from each sale.

SELECT CustomerKey,

SUM((S.Quantity \* P.UnitPriceUSD) - (S.Quantity \* P.UnitCostUSD)) AS ProfitContribution

FROM Cleaned\_Sales S

JOIN Cleaned\_Products P ON S.ProductKey = P.ProductKey

GROUP BY CustomerKey;

SUM((S.Quantity \* P.UnitPriceUSD) - (S.Quantity \* P.UnitCostUSD)):

Subtracts the unit cost from the unit price to calculate profit.

**19. Churn Rate (Calculated Column)**

If you’re calculating Churn Rate, it would be a measure that calculates the percentage of churned customers, not a column, but here’s a SQL query for it:

SELECT

(SELECT COUNT(\*) FROM Cleaned\_Customers WHERE Churned = 1) \* 100.0 /

(SELECT COUNT(\*) FROM Cleaned\_Customers) AS ChurnRate;

ChurnRate:

The percentage of customers who are churned.