RFM ANALYSIS ON MARKETING CAMPAIGN DATASET

Skills used: CREATE DATABASE, CREATE TABLE, SELECT, LIMIT, COUNT, INFORMATION_SCHEMA.COLUMNS, COALESCE, ALTER TABLE, ADD COLUMN, UPDATE, CREATE VIEW, WITH, ADD INDEX, CASE, SELECT, GROUP BY, COUNT, DROP VIEW.

1. Database Creation and Table Setup

First, Let's check if there's an old workspace named RFM_analysis and, if there is, it gets rid of it. Then, it creates a fresh workspace with the same name.

```
DROP DATABASE IF EXISTS `RFM_analysis`;
CREATE DATABASE `RFM_analysis`;
USE rfm_analysis;
```

In the SQL editor, the dataset named marketing_campaign has been imported into our database workspace (RFM_analysis).

2. Data Exploration

```
-- Head of the DataFrame
SELECT * FROM marketing_campaign LIMIT 5;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: | Fetch rows:

ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	MntFruits	MntMea
5524	1957	Graduation	Single	58138	0	0	04-09-2012	58	635	88	546
2174	1954	Graduation	Single	46344	1	1	08-03-2014	38	11	1	6
4141	1965	Graduation	Together	71613	0	0	21-08-2013	26	426	49	127
6182	1984	Graduation	Together	26646	1	0	10-02-2014	26	11	4	20
5324	1981	PhD	Married	58293	1	0	19-01-2014	94	173	43	118

Result Grid | Form Editor

Let's look into the number of rows and columns in the DataFrame

```
-- Shape of the DataFrame
SELECT COUNT(*) AS num_rows
FROM marketing_campaign;
```

num_rows
2216

```
SELECT COUNT(*) AS num_columns
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'marketing_campaign';
```

num_columns
29

Let's extract information about the data types of columns in the DataFrame named **marketing_campaign**

```
-- Data Types
SELECT COLUMN_NAME, DATA_TYPE
FROM INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = 'marketing_campaign';
```

	COLUMN_NAME	DATA_TYPE
▶	ID	int
	Year_Birth	int
	Education	text
	Marital_Status	text
	Income	int
	Kidhome	int

Let's check for missing values in the **marketing_campaign** DataFrame across various columns.

```
-- Missing Values
SELECT *
FROM marketing_campaign
WHERE COALESCE(ID, Year_Birth, Education, Marital_Status, Income, Kidhome,
Teenhome, Dt_Customer, Recency, Complain, MntWines, MntFruits,
MntMeatProducts, MntFishProducts, MntSweetProducts, MntGoldProds,
NumDealsPurchases, AcceptedCmp1, AcceptedCmp2, AcceptedCmp3, AcceptedCmp4,
AcceptedCmp5, Response, NumWebPurchases, NumCatalogPurchases,
NumStorePurchases, NumWebVisitsMonth
) IS NULL;
```

The query returned no results, it means that there are no missing values in the specified columns of the **marketing_campaign** DataFrame.

Let's check for duplicated values in the **marketing_campaign** DataFrame based on multiple columns.

```
-- Duplicated Values
SELECT COUNT(*) AS Duplicated_Values
FROM marketing_campaign
GROUP BY
  ID, Year_Birth, Education, Marital_Status, Income, Kidhome, Teenhome,
Dt_Customer, Recency, Complain, MntWines, MntFruits, MntMeatProducts,
MntFishProducts, MntSweetProducts, MntGoldProds, NumDealsPurchases,
AcceptedCmp1, AcceptedCmp2, AcceptedCmp3, AcceptedCmp4, AcceptedCmp5,
Response, NumWebPurchases, NumCatalogPurchases, NumStorePurchases,
NumWebVisitsMonth
HAVING COUNT(*) > 1
LIMIT 0, 1000;
```

The query did not return any results, it indicates that there are no duplicated values in the specified columns of the **marketing_campaign** DataFrame.

3. Data Cleaning and Transformation

Let's rename specific columns in the **marketing_campaign** table.

```
Select * from marketing_campaign;
-- Rename specific columns using ALTER TABLE
ALTER TABLE marketing_campaign
CHANGE COLUMN MntWines Wines INT,
CHANGE COLUMN MntFruits Fruits INT,
CHANGE COLUMN MntMeatProducts Meat INT,
CHANGE COLUMN MntFishProducts Fish INT,
CHANGE COLUMN MntSweetProducts Sweets INT,
CHANGE COLUMN MntGoldProds Gold INT;
Select * from marketing_campaign;
```

Adding two new columns, **Frequency** and **Monetary**, and Let's update their values

```
-- Add the 'Frequency' and 'Monetary' columns
ALTER TABLE marketing_campaign
ADD COLUMN Frequency INT,
ADD COLUMN Monetary INT;

-- Update the values for the new columns based on your calculations
UPDATE marketing_campaign
SET Frequency = NumDealsPurchases + NumWebPurchases + NumCatalogPurchases +
NumStorePurchases,
    Monetary = Wines + Fruits + Meat + Fish + Sweets + Gold;
Select * from marketing_campaign;
```

edCmp4	AcceptedCmp5	AcceptedCmp1	AcceptedCmp2	Complain	Z_CostContact	Z_Revenue	Response	Frequency	Monetary
0	0	0	0	3	11	1	25	1617
0	0	0	0	3	11	0	6	27
0	0	0	0	3	11	0	21	776
0	0	0	0	3	11	0	8	53
0	0	0	0	3	11	0	19	422

4. Creating RFM Dataset

Creating a new table named **rfm_data** derived from the **marketing_campaign** table, with ID as the index

```
-- Create the 'rfm_data' table with 'ID' as the index
CREATE TABLE rfm_data AS
SELECT
    ID,
    Recency,
    NumDealsPurchases + NumWebPurchases + NumCatalogPurchases +
NumStorePurchases AS Frequency,
    Wines + Fruits + Meat + Fish + Sweets + Gold AS Monetary
FROM marketing_campaign;

-- Add an index on the 'ID' column
ALTER TABLE rfm_data
ADD INDEX idx_ID (ID);

-- Retrieve the data from the new table
SELECT * FROM rfm_data LIMIT 5;
```

ID	Recency	Frequency	Monetary
5524	58	25	1617
2174	38	6	27
4141	26	21	776
6182	26	8	53
5324	94	19	422

5. Calculating RFM Scores

Let's Calculate RFM scores for the **rfm_data** table based on the Recency, Frequency, and Monetary values. The **NTILE(5)** function is used to assign scores in quintiles (5 groups) for each of these dimensions.

```
SELECT
ID
,Recency
,Frequency
```

```
,Monetary
,NTILE(5) OVER(ORDER BY Recency DESC) AS Recency_Score
,NTILE(5) OVER(ORDER BY Frequency ASC) AS Frequency_Score
,NTILE(5) OVER(ORDER BY Monetary ASC) AS Monetary_Score
FROM
rfm_data
ORDER BY
ID
```

ID	Recency	Frequency	Monetary	Recency_Score	Frequency_Score	Monetary_Score
0	66	17	1198	2	3	5
1	0	18	577	5	4	3
9	86	11	120	1	2	2
13	57	6	32	3	1	1
17	81	28	1028	1	5	4
20	91	8	183	1	2	2

6. Creating RFM View

Creates a view named **RFM_View** by combining calculations for RFM values and scores.

```
-- DropView if exixted
DROP VIEW IF EXISTS RFM_View

CREATE VIEW RFM_View AS
WITH
-- Calculate RFM Values
RFM_CALC AS (
    SELECT
        ID,
        Recency,
        NumDealsPurchases + NumWebPurchases + NumCatalogPurchases +
NumStorePurchases AS Frequency,
        Wines + Fruits + Meat + Fish + Sweets + Gold AS Monetary
    FROM marketing_campaign
    GROUP BY ID  -- Assuming 'ID' is the correct column for grouping
),
-- Calculate RMF Scores
RFM_SCORES AS (
    SELECT
        ID,
        Recency,
        Frequency,
        Monetary,
        NTILE(5) OVER (ORDER BY Recency DESC) AS Recency_Score,
        NTILE(5) OVER (ORDER BY Frequency ASC) AS Frequency_Score,
        NTILE(5) OVER (ORDER BY Monetary ASC) AS Monetary_Score
    FROM RFM_CALC
),
-- Calculate Avg RFM Score
RFM_AVG_SCORE AS (
    SELECT
        ID,
        CONCAT_WS('-', Recency_Score, Frequency_Score, Monetary_Score) AS
R_F_M,
        CAST((CAST(Recency_Score AS Float) + Frequency_Score +
Monetary_Score) / 3 AS DECIMAL(16, 2)) AS Avg_RFM_Score
    FROM RFM_SCORES
)
```

```sql
SELECT
    T1.ID,
    Recency,
    Frequency,
    Monetary,
    Recency_Score,
    Frequency_Score,
    Monetary_Score,
    R_F_M,
    Avg_RFM_Score
FROM RFM_SCORES T1
JOIN RFM_AVG_SCORE T2 ON T1.ID = T2.ID;

SELECT * FROM RFM_View LIMIT 10;
```

ID	Recency	Frequency	Monetary	Recency_Score	Frequency_Score	Monetary_Score	R_F_M	Avg_RFM_Score
11110	56	0	5	3	1	1	3-1-1	1.67
5555	81	0	6	1	1	1	1-1-1	1.00
3955	20	0	6	4	1	1	4-1-1	2.00
11181	85	0	8	1	1	1	1-1-1	1.00
10104	65	4	8	2	1	1	2-1-1	1.33
5824	1	4	8	5	1	1	5-1-1	2.33

7. Customer Segmentation

This new view, **Customer_Segmentation**, includes additional columns for **Value Segmentation** and **Customer Segmentation** based on the calculated RFM scores.

```sql
-- Drop View if already exists
DROP VIEW IF EXISTS Customer_Segmentaion;
---- Create a View for the Customer Segments & Value Segments using the
View "RFM_View"
CREATE VIEW Customer_Segmentation AS
SELECT *,
  CASE
    WHEN Avg_RFM_Score >= 4 THEN 'High Value'
    WHEN Avg_RFM_Score >= 2.5 AND Avg_RFM_Score < 4 THEN 'Mid Value'
    WHEN Avg_RFM_Score > 0 AND Avg_RFM_Score < 2.5 THEN 'Low Value'
  END AS Value_Seg, -- Value Segment
  CASE
    WHEN Frequency_Score >= 4 AND Recency_Score >= 4 AND Monetary_Score >=
4 THEN 'VIP'
    WHEN Frequency_Score >= 3 AND Monetary_Score < 4 THEN 'Regular'
    WHEN Recency_Score <= 3 AND Recency_Score > 1 THEN 'Inactive'
    WHEN Recency_Score = 1 THEN 'Churned'
    WHEN Recency_Score >= 4 AND Frequency_Score <= 4 THEN 'New Customer'
  END AS Cust_Seg -- Customer Segment
FROM RFM_View;

SELECT * FROM Customer_Segmentation ORDER BY Avg_RFM_Score LIMIT 10;
```

ency	Frequency	Monetary	Recency_Score	Frequency_Score	Monetary_Score	R_F_M	Avg_RFM_Score	Value_Seg	Cust_Seg
0	6	1	1	1	1-1-1	1.00	Low Value	Churned
0	8	1	1	1	1-1-1	1.00	Low Value	Churned
4	10	1	1	1	1-1-1	1.00	Low Value	Churned
4	10	1	1	1	1-1-1	1.00	Low Value	Churned
4	11	1	1	1	1-1-1	1.00	Low Value	Churned

8. Analyzing Customer Segments

Let's aggregate customer counts based on the "**Value_Seg**" column from the "**Customer_Segmentation**" view.

```
SELECT
Value_Seg,
COUNT(ID) AS Customer_Count
FROM Customer_Segmentation
GROUP BY Value_Seg
ORDER BY Customer_Count
```

Value_Seg	Customer_Count
High Value	513
Low Value	756
Mid Value	947

Let's aggregate customer counts based on the "**Cust_Seg**" column from the "**Customer_Segmentation**" view

```
SELECT
Cust_Seg,
COUNT(ID) AS Customer_Count
FROM Customer_Segmentation
GROUP BY Cust_Seg
ORDER BY Customer_Count;
```

Cust_Seg	Customer_Count
VIP	282
Churned	361
New Customer	420
Regular	449
Inactive	704

Insights:

- **VIP Customers (282):**

  There are 282 customers classified as VIP. These customers likely represent a high-value and highly engaged segment, often associated with loyal and valuable customers.

- **Churned Customers (361):**

  There are 361 customers classified as Churned. Churned customers are those who have stopped interacting or transacting with the business. This segment requires attention and strategies for re-engagement.

- **New Customers (420):**

  There are 420 customers classified as New. These are recently acquired customers who may need nurturing and targeted marketing efforts to maximize their retention and conversion into loyal customers.

- **Regular Customers (449):**

  There are 449 customers classified as Regular. Regular customers are likely those who make consistent but not necessarily frequent purchases. Maintaining their engagement is essential for sustained revenue.

- **Inactive Customers (704):**

  There are 704 customers classified as Inactive. This group may include both dormant and inactive customers. Strategies for re-engagement and understanding the reasons for inactivity are crucial for this segment.

Conclusion of the Project:

- The project has successfully segmented customers based on RFM analysis, allowing for targeted strategies tailored to different customer behaviors and needs.
- Specific attention is needed for the Churned and Inactive customer segments to implement re-engagement strategies.
- The VIP and New Customer segments represent opportunities for continued growth and loyalty-building initiatives.
- Regular customers form a stable customer base that requires consistent efforts to maintain their loyalty.
- Ongoing monitoring and analysis will be crucial to adapt strategies based on changing customer behaviors and market dynamics.