EXPLORING RFM ANALYSIS: METHODOLOGIES AND INNOVATIVE APPROACHES

Ahmet YÜCE, December 09,2023

Abstract:

In the ever-evolving landscape of business, understanding customer behavior stands paramount for sustained success. Recency, Frequency, and Monetary (RFM) analysis have long served as pillars in dissecting customer engagement. This article navigates through established RFM practices, exploring innovative methodologies to redefine customer segmentation and behavior analysis. It delves into the discrepancies within traditional RFM metrics, highlighting the diminishing significance of recency with prolonged durations and unfair different durations for calculations of frequency and monetary values. The study introduces a novel monthly based approach, unveiling nuanced customer categorizations that diverge from conventional perceptions. Monthly analysis emerges as a new approach to customer segmentation. This paradigm shift in RFM analysis underscores the need for businesses to adapt to evolving customer-centric strategies.

By showcasing both traditional RFM practices and innovative methodologies, this article aims to offer a holistic view of how RFM analysis evolves to meet the complex demands of modern business landscapes. Through this exploration, readers will gain a comprehensive understanding of RFM analysis, its adaptability, and its potential for driving impactful business strategies.

Keywords*: RFM Analysis, Customer Segmentation, Recency, Frequency, Monetary, Customer Behavior, Alternative to RFM Analysis, Monthly Analysis, Business Strategies*

Introduction

In today's rapidly evolving business landscape, understanding and effectively catering to customer needs are paramount for sustainable success. One of the pivotal tools aiding businesses in this quest for customer-centricity is RFM analysis. This article aims to provide an overview of RFM analysis, while also delving into innovative methodologies.

Before diving into innovative methodologies, we will examine established practices and explore common examples illustrating how businesses implement RFM analysis to categorize customers. Building on established methodologies, I will shed light on innovative approaches and personalized adaptations of RFM analysis. I will introduce you to some novel techniques to provide nuanced insights into customer segmentation and behavior analysis. Let’s begin with what RFM Analysis is.

What is RFM Analysis?

RFM, which stands for Recency, Frequency, and Monetary, is a data-driven technique utilized by businesses to categorize and segment their customer base. At its core, RFM analysis quantifies customer behavior based on three key parameters:

1. Recency (R): This metric evaluates the time elapsed since a customer's last interaction or transaction with the business. It emphasizes the principle that more recent interactions may indicate higher engagement or responsiveness.

2. Frequency (F): Frequency assesses the number of interactions or transactions conducted by a customer within a defined period. It sheds light on how often customers engage with the business, indicating loyalty or buying habits.

3. Monetary (M): Monetary value represents the total spending or monetary contribution of a customer over a specific timeframe. It signifies the customer's profitability or the potential revenue they bring to the business.

The primary goal of RFM analysis is to segment customers into distinct groups based on their behaviors and transaction histories. By categorizing customers according to these three metrics, businesses gain valuable insights into varying customer segments:

- Identification of High-Value Customers: RFM analysis helps identify and prioritize high-value customers who contribute significantly to the business's revenue streams. These customers often exhibit recent activity, frequent transactions, and substantial monetary contributions.

- Personalized Marketing Strategies: Segmentation through RFM analysis enables businesses to tailor marketing strategies and campaigns based on the distinct needs and behaviors of different customer segments. This personalization enhances customer engagement and satisfaction.

- Retention and Loyalty Programs: Understanding customer behavior through RFM analysis aids in developing targeted retention and loyalty programs. By recognizing and nurturing loyal customers or re-engaging dormant ones, businesses can enhance customer retention rates.

PART I – *TRADITIONAL RFM PRACTICES*

Now, let's delve into the common practices of conducting RFM analysis. In this study, Python programming language, Jupyter notebook as the IDE, and the "Online Retail" dataset from the UCI Machine Learning Repository as the data set cleaned have been utilized. The essence of the necessary code segments will be covered in this article, but the complete set of codes won't be presented here. You can access the detailed notebook on my GitHub account.

"Online Retail is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers."

In the original dataset, there are eight attributes: InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country. To derive the monetary value (total sales amount for each customer), we introduced a new attribute labeled "Revenue" into the dataset. This was achieved by implementing the following code: df["Revenue"] = df["UnitPrice"] * df["Quantity"].

Before constructing the RFM Table, the details regarding our cleaned and prepared dataset are outlined below:

```
df.info();
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 397884 entries, 0 to 541908
Data columns (total 10 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Unnamed: 0   397884 non-null  int64
 1   InvoiceNo    397884 non-null  object
 2   StockCode    397884 non-null  object
 3   Description  397884 non-null  object
 4   Quantity     397884 non-null  float64
 5   InvoiceDate  397884 non-null  datetime64[ns]
 6   UnitPrice    397884 non-null  float64
 7   CustomerID   397884 non-null  float64
 8   Country      397884 non-null  object
 9   Revenue      397884 non-null  float64
dtypes: datetime64[ns](1), float64(4), int64(1), object(4)
memory usage: 33.4+ MB
```

The initial phase and foundation of RFM analysis involve the preparation of the RFM Table, which combines the calculation of each RFM metric. An example of the RFM Table is demonstrated below. The RFM Table, although being the most basic table obtained at the outset of our analysis, holds immense value by allowing us to conduct valuable analyses such as sorting columns to list lost customers, the most loyal, or the most valuable clients. It bestows us with the opportunity to derive valuable insights from its columns.

```
RFM_Table.head()
```

CustomerID	recency	frequency	monetary
12346.0	325	1	77183.60
12347.0	1	182	4310.00
12348.0	74	31	1797.24
12349.0	18	73	1757.55
12350.0	309	17	334.40

```
RFM_Table.sort_values(by="monetary", ascending=False)
```

CustomerID	recency	frequency	monetary
14646.0	1	2076	280206.02
18102.0	1	431	259657.30
17450.0	7	337	194550.79
16446.0	1	3	168472.50
14911.0	1	5675	143825.06

We can interpret the first customer in the left-hand table as follows: This customer has made a single purchase(frequency=1) in total, and it has been 325 days(recency=325) since that purchase without any subsequent transactions, indicating that he/she is a lost customer.

In the right-hand table, we observe the top 5 customers who have made the highest total purchases. Except for customer 16446 in the fourth row, these customers appear to be loyal due to their high frequency and very low recency values. Customer 16446, who has made a total of 3 purchases and made his/her last purchase yesterday, seems to be a potentially valuable new customer. However, it wouldn't be possible to definitively confirm his/her status as a new customer solely based on this RFM Table without looking at his/her prior purchase dates.

```
RFM_Table.describe()
```

	recency	frequency	monetary
count	4338.000000	4338.000000	4338.000000
mean	91.536422	91.720609	2054.266460
std	100.014169	228.785094	8989.230441
min	0.000000	1.000000	3.750000
25%	17.000000	17.000000	307.415000
50%	50.000000	41.000000	674.485000
75%	141.000000	100.000000	1661.740000
max	373.000000	7847.000000	280206.020000

We can also observe the statistics of RFM attributes leveraging the describe() function. We will use 25%, 50%, and 75% quantile values of these attributes shortly.

Let's now take a step-by-step look at how this table is constructed.

1. Find the Recency, Frequency, and Monetary values of each Customer:

```
import datetime as dt

NOW = df['InvoiceDate'].max()              # NOW = dt.datetime(2011,12,10)
```

```
RFM_Table = df.groupby('CustomerID').agg({'InvoiceDate': lambda x: (NOW - x.max()).days,
                                          'InvoiceNo': lambda x: len(x),
                                           'Revenue': lambda x: x.sum()})

RFM_Table['InvoiceDate'] = RFM_Table['InvoiceDate'].astype(int)

RFM_Table.rename(columns={'InvoiceDate': 'recency',
                          'InvoiceNo': 'frequency',
                           'Revenue': 'monetary'}, inplace=True)
```

The code above is used to generate an RFM (Recency, Frequency, Monetary) Table for customer segmentation. The variable **NOW** is set to the maximum date in the 'InvoiceDate' column of the dataset, essentially representing the current date or reference point for recency calculations.

The dataset is grouped by 'CustomerID' to perform aggregation operations on each customer's data. Three aggregation metrics are computed for each customer:

- **Recency (InvoiceDate)**: Calculating the number of days between the reference date (NOW) and the last invoice date for each customer.
- **Frequency (InvoiceNo)**: Counting the number of invoices for each customer, indicating their transaction frequency.
- **Monetary (Revenue)**: Summing up the revenue or total amount spent by each customer across all transactions.

The **lambda** functions here are applied within the **agg()** method to perform specific calculations on the grouped data for each customer, resulting in the aggregation of Recency, Frequency, and Monetary metrics, forming the basis of the RFM Table.

- **lambda x: (NOW - x.max()).days**:

This lambda function calculates the 'Recency' metric for each customer. It computes the number of days between the reference date (**NOW**) and the maximum date in the 'InvoiceDate' column for each customer. This value represents how recently each customer made their last purchase.

- **lambda x: len(x)**:

This lambda function determines the 'Frequency' metric for each customer. It counts the number of invoices (transactions) associated with each customer, providing insights into how frequently they conduct transactions with the business.

- **lambda x: x.sum()**:

This lambda function calculates the 'Monetary' metric for each customer. It sums up the total revenue generated by each customer across all their transactions, indicating their monetary contribution or total spending.

The 'InvoiceDate' values in the RFM_Table are converted to integer type for clearer representation of recency as days.

The resulting aggregated metrics are assigned as columns in the RFM_Table, labeled as 'recency', 'frequency', and 'monetary' respectively for better clarity and understanding.

Our RFM Table is now prepared. In the second phase, we will add three more columns to this table.

2. Add 'r_quartile', 'f_quartile', 'm_quartile' columns to the RFM_Table:

Let's assume that we have arranged each customer in order based on either recency, frequency, or monetary values. We will divide our sorted lists into four segments and determine which segment each customer falls into. Consequently, instead of examining individual RFM values for customers, we will look into which segment they belong to. Customers in the fourth segment/quartile will be the best, while those in the first quartile will be the lowest-ranked customers for the respective RFM metric.

We will now expand our RFM Table by adding columns that display RFM quartiles, as shown below.

	recency	frequency	monetary	r_quartile	f_quartile	m_quartile
CustomerID						
12346.0	325	1	77183.60	1	1	4
12347.0	1	182	4310.00	4	4	4
12348.0	74	31	1797.24	2	2	4
12349.0	18	73	1757.55	3	3	4
12350.0	309	17	334.40	1	1	2

We can employ different code blocks to add the 'r_quartile', 'f_quartile', and 'm_quartile' columns to the RFM Table. Different code samples are shown below. We will proceed with the second one.

Code sample - I

```
quantiles = RFM_Table.quantile(q=[0.25, 0.5, 0.75])
quantiles = quantiles.to_dict()

def R_Segmenting(x,p,d):
    if   x <= d[p][0.25]:   return 4
    elif x <= d[p][0.50]:   return 3
    elif x <= d[p][0.75]:   return 2
    else                :   return 1

def FM_Segmenting(x,p,d):
    if   x <= d[p][0.25]:   return 1
    elif x <= d[p][0.50]:   return 2
    elif x <= d[p][0.75]:   return 3
    else                :   return 4

rfm_q = RFM_Table.copy()
rfm_q['r_quartile'] = rfm_q['recency'].  apply(R_Segmenting,  args=('recency',   quantiles,))
rfm_q['f_quartile'] = rfm_q['frequency'].apply(FM_Segmenting, args=('frequency', quantiles,))
rfm_q['m_quartile'] = rfm_q['monetary'].  apply(FM_Segmenting, args=('monetary',  quantiles,))
```

Code sample - II

```
# Cut RFM metrics into 4 groups and label them:
rfm_q["r_quartile"] = pd.qcut(rfm_q["recency"], 4, labels=[4, 3, 2, 1])
rfm_q["f_quartile"] = pd.qcut(rfm_q["frequency"].rank(method="first"), 4, labels=[1, 2, 3, 4])
rfm_q["m_quartile"] = pd.qcut(rfm_q["monetary"], 4, labels=[1, 2, 3, 4])
```

Both codes perform RFM (Recency, Frequency, Monetary) segmentation by dividing the customers into quartiles based on these metrics.

Our primary goal is to arrange customers based on RFM metrics and identify the thresholds that divide them into four segments. To illustrate, cutting a string into four parts requires making three cuts. Similarly, we'll identify these three positions/values where the divisions will occur. For this purpose, the first code sample

utilizes the `quantile()` function. Once these division thresholds are identified, it stores them in a dictionary. Following this, by implementing a custom segmenting function, it applies these thresholds to each RFM metric / column, categorizing the segments as 1-2-3-4.

The second code sample uses qcut() function to identify the cutting thresholds (quartiles). Let's look at the second code:

rfm_q["r_quartile"] = pd.qcut(rfm_q["recency"], 4, labels=[4, 3, 2, 1]): Divides the "recency" column into 4 quartiles using `pd.qcut()`. Assigns labels 4, 3, 2, 1 to these quartiles, where 4 represents the most recent customers and 1 represents the least recent.

rfm_q["f_quartile"] = pd.qcut(rfm_q["frequency"].rank(method="first"), 4, labels=[1, 2, 3, 4]): Calculates the rank of the "frequency" column and divides it into 4 quartiles using `pd.qcut()`. Assigns labels 1, 2, 3, 4 to these quartiles, likely ordering customers by their frequency, where 1 represents the least frequent and 4 represents the most frequent.

rfm_q["m_quartile"] = pd.qcut(rfm_q["monetary"], 4, labels=[1, 2, 3, 4]): Divides the "monetary" column into 4 quartiles using `pd.qcut()`. Assigns labels 1, 2, 3, 4 to these quartiles, where 1 represents the lowest-spending customers and 4 represents the highest-spending.

This approach enables the segmentation of customers into four distinct groups based on their RFM metrics, allowing for easier analysis and identification of customer behavior patterns.

As an alternative, we can divide our customers into 5 quantiles but creating 5x5x5=125 different segments would make analysis quite challenging, so it's not a highly preferable approach unless you exclude one of the RFM metrics.

```
# Cutting RFM metrics into 5 groups and labeling them:
rfm_q["r_quartile"] = pd.qcut(rfm_q["recency"], 5, labels=[5, 4, 3, 2, 1])
rfm_q["f_quartile"] = pd.qcut(rfm_q["frequency"].rank(method="first"), 5, labels=[1, 2, 3, 4, 5])
rfm_q["m_quartile"] = pd.qcut(rfm_q["monetary"], 5, labels=[1, 2, 3, 4, 5])
```

3. Combine RFM quartiles and add a new column to the RFM Table:

In the third step, we'll incorporate the merged "RFM_quartiles" as a new column into our RFM Table.

	recency	frequency	monetary	r_quartile	f_quartile	m_quartile	RFM_quartiles
CustomerID							
12350.0	309	17	334.40	1	1	2	112
12349.0	18	73	1757.55	3	3	4	334
12348.0	74	31	1797.24	2	2	4	224
12347.0	1	182	4310.00	4	4	4	444
12346.0	325	1	77183.60	1	1	4	114

We will generate a new column named 'RFM_quartiles' by writing the code below. This new column consolidates the individual quartile values of 'r_quartile', 'f_quartile', and 'm_quartile' as strings concatenated together, enabling the representation of merged quartile segments for each customer in a single column.

```
# Combine RFM quartiles and add as a new column
RFM["RFM_quartiles"] =  RFM['r_quartile'].astype(str) + RFM['f_quartile'].astype(str) + RFM['m_quartile'].astype(str)
```

Segmenting customers into RFM quartiles, particularly through the 'RFM_quartiles' column, offers businesses a comprehensive view of customer behavior. Grouping customers based on RFM metrics and consolidating them into quartiles simplifies analysis. It condenses multidimensional data into manageable segments, easing comparative assessments.

Each quartile represents a distinct customer behavior pattern. For instance, the '444' segment denotes the most recent, frequent, and high-spending customers, indicating significant potential for revenue and loyalty.

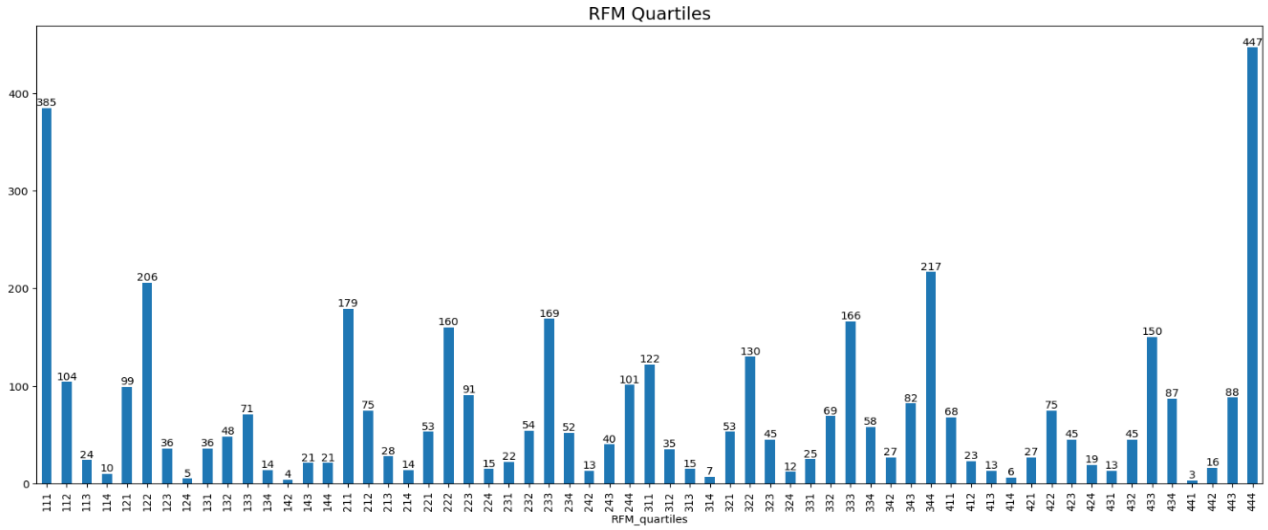
Segmentation allows the identification of different segments such as high-value customers. Understanding these segments enables tailored strategies, like targeted marketing or retention initiatives, specific to each group's preferences. This might involve exclusive offers, loyalty programs, or personalized experiences to further engage and retain these valuable customers.

Observing the patterns of each segment over time helps in predicting future behavior. Tracking changes in their engagement or spending habits might indicate shifts in preferences or potential churn risks, allowing proactive measures to be taken.

4. Define Customer Segments by grouping RFM quartiles:

When each RFM attribute—recency, frequency, and monetary— is segmented into four levels (ranging from 1 to 4), combining these three attributes results in a total segment combination of 4 x 4 x 4 = 64 (such as '111' to '444'). However, individually tracking, and devising strategies for each of these 64 segments can become challenging. Hence, there arises a need to rationalize and group these segments effectively to streamline analysis and strategy development in a more manageable manner.

The chart below illustrates the distribution of total customers based on their RFM quartiles.



Now, in the fourth step, we'll proceed with the grouping process of these (maximum 64) segments.

	recency	frequency	monetary	r_quartile	f_quartile	m_quartile	RFM_quartiles	Segments
CustomerID								
17600.0	17	13	161.67	4	1	1	411	Recent Customers
13651.0	34	6	94.20	3	1	1	311	Still Got Hope
15592.0	46	24	388.58	3	2	2	322	Potential Loyalist
15802.0	142	1	451.42	1	1	2	112	Lost
18075.0	22	142	2611.75	3	4	4	344	Champions
13919.0	58	28	384.81	2	2	2	222	Potential Loyalist
16423.0	24	18	346.12	3	2	2	322	Potential Loyalist
16451.0	87	14	266.97	2	1	1	211	Still Got Hope

We will add a final column named “Segments” to RFM_Table using the code below. This code segments customers by evaluating their RFM quartile patterns against predefined criteria, assigning them to specific segments that represent distinct customer behavior groups. This code defines a segmentation classifier function called 'classifier'. It aims to assign segment names ('Champions', 'Loyal Customers', 'Potential Loyalist', etc.) based on the RFM quartiles.

- A dictionary called 'Segments' contains segment names and their corresponding RFM quartile patterns.

```
Segments = {
            'Segments':
                            ['Champions',
                             'Loyal Customers',
                             'Potential Loyalist',
                             'Recent Customers',
                             'Customers Needing Attention',
                             'Still Got Hope',
                             'Need to Get Them Back',
                             'Give it a Try',
                             'Lost'],\
                        'RFM_quartiles':
                            ['(3|4)-(3|4)-(3|4)',
                             '(3|4)-(3|4)-(1|2|3|4)',
                             '(2|3|4)-(2|3)-(1|2|3|4)',
                             '(4)-(1)-(1|2|3|4)',
                             '(2|3)-(2|3)-(2|3)',
                             '(2|3)-(1|2)-(1|2|3|4)',
                             '(1|2)-(3|4)-(2|3|4)',
                             '(1|2)-(1|2|3|4)-(1|2|3|4)',
                             '(1)-(1)-(1|2|3|4)']
            }
pd.DataFrame(Segments)
```

	Segments	RFM_quartiles
0	Champions	(3|4)-(3|4)-(3|4)
4	Customers Needing Attention	(2|3)-(2|3)-(2|3)
7	Give it a Try	(1|2)-(1|2|3|4)-(1|2|3|4)
8	Lost	(1)-(1)-(1|2|3|4)
1	Loyal Customers	(3|4)-(3|4)-(1|2|3|4)
6	Need to Get Them Back	(1|2)-(3|4)-(2|3|4)
2	Potential Loyalist	(2|3|4)-(2|3)-(1|2|3|4)
3	Recent Customers	(4)-(1)-(1|2|3|4)
5	Still Got Hope	(2|3)-(1|2)-(1|2|3|4)

- The 'classifier' function maps each RFM quartile pattern to a specific segment according to defined conditions using if-elif statements.

```
def classifier(rfm):
    if (rfm[0] in ['3', '4'])   & (rfm[1] in ['3', '4']) & (rfm[2] in ['3', '4']):            rfm = 'Champions'

    elif (rfm[0] in ['3', '4']) & (rfm[1] in ['3', '4']) & (rfm[2] in ['1', '2', '3', '4']):  rfm = 'Loyal Customers'

    elif (rfm[0] in ['2','3','4']) & (rfm[1] in ['2','3']) & (rfm[2] in ['1','2','3', '4']):  rfm = 'Potential Loyalist'

    elif (rfm[0] in ['4'])      & (rfm[1] in ['1'])      & (rfm[2] in ['1', '2', '3', '4']):  rfm = 'Recent Customers'

    elif (rfm[0] in ['2', '3']) & (rfm[1] in ['2', '3']) & (rfm[2] in ['2', '3']):        rfm = 'Customers Needing Attention'

    elif (rfm[0] in ['2', '3']) & (rfm[1] in ['1', '2']) & (rfm[2] in ['1', '2', '3', '4']):  rfm = 'Still Got Hope'

    elif (rfm[0] in ['1', '2']) & (rfm[1] in ['3', '4']) & (rfm[2] in ['2', '3', '4']):       rfm = 'Need to Get Them Back'

    elif (rfm[0] in ['1'])      & (rfm[1] in ['1'])      & (rfm[2] in ['1', '2','3','4']):    rfm = 'Lost'

    elif (rfm[0] in ['1', '2']) & (rfm[1] in ['1','2','3','4']) & (rfm[2] in ['1','2','3','4']):rfm = 'Give it a Try'

    return rfm
```

- The 'segmented_rfm' DataFrame, created by copying the 'rfm_quartiles' DataFrame, applies the 'classifier' function to the 'RFM_quartiles' column. This assigns a segment name to each entry based on its respective quartile pattern.

```
segmented_rfm = rfm_quartiles.copy()
segmented_rfm['Segments'] = segmented_rfm["RFM_quartiles"].apply(classifier)
```

So, by segmenting our customers, we've finalized our RFM Table. With this table in hand, we can conduct detailed analyses and gain various insights.

In practice, we can employ different codes serving the same purpose. Below is an example using a regex function for customer segmentation based solely on recency and frequency metrics using 5 quartiles.

```
# Creating 5 quartiles based on recency and frequency metrics, excluding monetary.

[r][f]_quartiles_of_segments =

  {
    r'[1-2][1-2]': 'Hibernating',
    r'[1-2][3-4]': 'At_Risk',
    r'[1-2]5'    : 'Cant_Loose',
    r'3[1-2]'    : 'About_to_Sleep',
    r'33'        : 'Need_Attention',
    r'[3-4][4-5]': 'Loyal_Customers',
    r'41'        : 'Promising',
    r'51'        : 'New_Customers',
    r'[4-5][2-3]': 'Potential_Loyalists',
    r'5[4-5]'    : 'Champions'
   }

rfm_q['Segment'] = rfm_q['r_quartile'].astype(str) + rfm_q['f_quartile'].astype(str)
rfm_q['Segment'] = rfm_q['Segment'].replace([r][f]_quartiles_of_segments, regex=True)
```

Once we've segmented our customers, we can see how many customers fall into each segment, develop specific strategies for each segment, and focus on a particular group.

```
segmented_rfm["Segments"].value_counts(dropna=False)
```

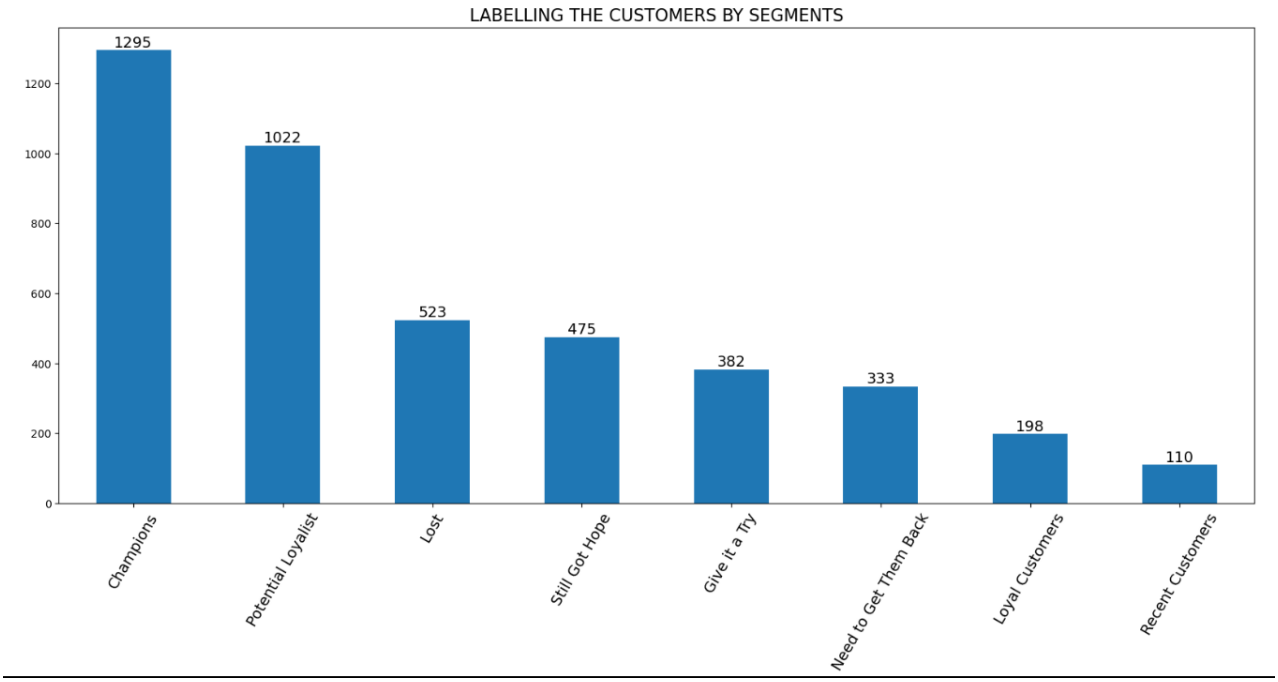
```
Champions               1295
Potential Loyalist      1022
Lost                     523
Still Got Hope           475
Give it a Try            382
Need to Get Them Back    333
Loyal Customers          198
Recent Customers         110
```

Besides checking whether we've made a sound grouping by looking at the averages of each group's RFM attributes, we can also gather information about each group's shopping behavior.

```
segmented_rfm.groupby('Segments').agg({'recency':'mean','frequency':'mean',
                                       'monetary':['mean','count']}).round(1)
```

	recency	frequency	monetary	
	mean	mean	mean	count
Segments				
Champions	16.3	216.1	4949.1	1295
Give it a Try	243.3	29.8	448.7	382
Lost	257.7	8.7	580.9	523
Loyal Customers	21.8	80.2	442.1	198
Need to Get Them Back	152.1	122.9	1865.4	333
Potential Loyalist	59.9	38.9	866.8	1022
Recent Customers	9.0	9.3	2137.8	110
Still Got Hope	65.5	9.4	415.6	475

Using various libraries' plotting techniques, we can quickly gain insights into our customer portfolio.



PART II – *AN ALTERNATIVE APPROACH TO RFM ANALYSIS: MONTHLY ANALYSIS*

In the previous section, we provided examples of how RFM analysis is commonly conducted. In the following section, I will introduce a new method I developed.

In standard RFM analysis practices, there's no specific limitation regarding the period analyzed, whether it's six months, one year, or two years. RFM metrics are calculated for the entire period in the dataset. When dealing with a dataset that spans many years, including earlier years might not always be an appropriate choice, and focusing on the most recent year or two could be more meaningful. There's no significant difference between a customer with a Recency value of one year and another with ten years; both would fall into the “lost customer” category.

Similarly, while emphasizing loyalty for a customer who has been consistently loyal for a long period, say 10 years of shopping, it's essential to note that, for an online retail company in a rapidly changing market, a customer shopping regularly for 1-2 years is also considered loyal. In loyalty calculations, as the number of years increases, the loyalty of recent customers within the last 1-2 years might diminish in significance, despite their actual loyalty, which should be considered important. So, including very early transactions in the Frequency calculation might distort the interpretation of loyalty understanding of today’s business.

Additionally, determining a customer's shopping frequency based on a particular timeframe is a matter worth considering. Comparing customer shopping patterns and spending solely based on their loyalty duration, like two years versus five, may lead to biased assessments. Performing monthly calculations enables averaging frequencies and expenditures, facilitating fairer customer comparisons. This highlights the need to recalibrate Frequency and Monetary metrics, emphasizing the significance of monthly assessments for a more balanced evaluation of customer behaviors.

In the standard RFM analysis, calculations are based on days. The fundamental unit of time is the day. The adaptation to a monthly tracking system for customer analysis appears to offer greater logical coherence. The ambiguity in the analysis time frame and the reliance on daily calculations in standard RFM analysis led me to explore a different approach.

According to this approach, I segmented customers based on monthly shopping frequency and amounts according to the following criteria. These criteria/conditions can be customized considering the typical average shopping time frames in the type of market where the analysis is conducted.

- Customers who haven't shopped in the last seven months will be labeled as "Lost".
- Customers who haven't shopped in the last three months will be labeled as "About to Lost".
- Customers who have shopped only in the last one or two months will be labeled as "Recent Customer".
- Customers who have shopped in 84% of the available months in the dataset and whose total shopping amount is in the highest 20% will be labeled as "Golden Customer".
- Customers who have shopped in the last three months and have shopped in 60% or higher of the available months in the dataset will be labeled as "Loyal Customer".
- Customers who have shopped in the last three months and have shopped in 40% or higher of the available months in the dataset will be labeled as "Valuable Irregular Customer".
- The remaining customers will be labeled as "Irregular Customer."

This analysis included all 13 months covered by our dataset. For datasets spanning longer periods, analyzing on a 12-month basis, aligned with the calendar year, followed by comparing each year, would be a more suitable approach.

At the end of the analysis, I categorized customers in line with my approach as shown in the table below:

```
df_monthly_spending
```

	CustomerID	2010-12	2011-01	2011-02	...	TotalSales	CountMonthsShopped	ShoppingPercentage	CustomerSegment
0	12346.00	0.00	77183.60	0.00	...	77183.60	1	7.69	Lost
1	12347.00	711.79	475.39	0.00	...	4310.00	7	53.85	Valuable Irregular Customer
2	12348.00	892.80	227.44	0.00	...	1797.24	4	30.77	About to Lost
3	12349.00	0.00	0.00	0.00	...	1757.55	1	7.69	Recent Customer
4	12350.00	0.00	0.00	334.40	...	334.40	1	7.69	Lost
...	...	...	...	...	...	...	...	...	...
4333	18280.00	0.00	0.00	0.00	...	180.60	1	7.69	Lost
4334	18281.00	0.00	0.00	0.00	...	80.82	1	7.69	Irregular Customer
4335	18282.00	0.00	0.00	0.00	...	178.05	2	15.38	Irregular Customer
4336	18283.00	0.00	213.75	100.95	...	2045.53	10	76.92	Loyal Customer
4337	18287.00	0.00	0.00	0.00	...	1837.28	2	15.38	Irregular Customer

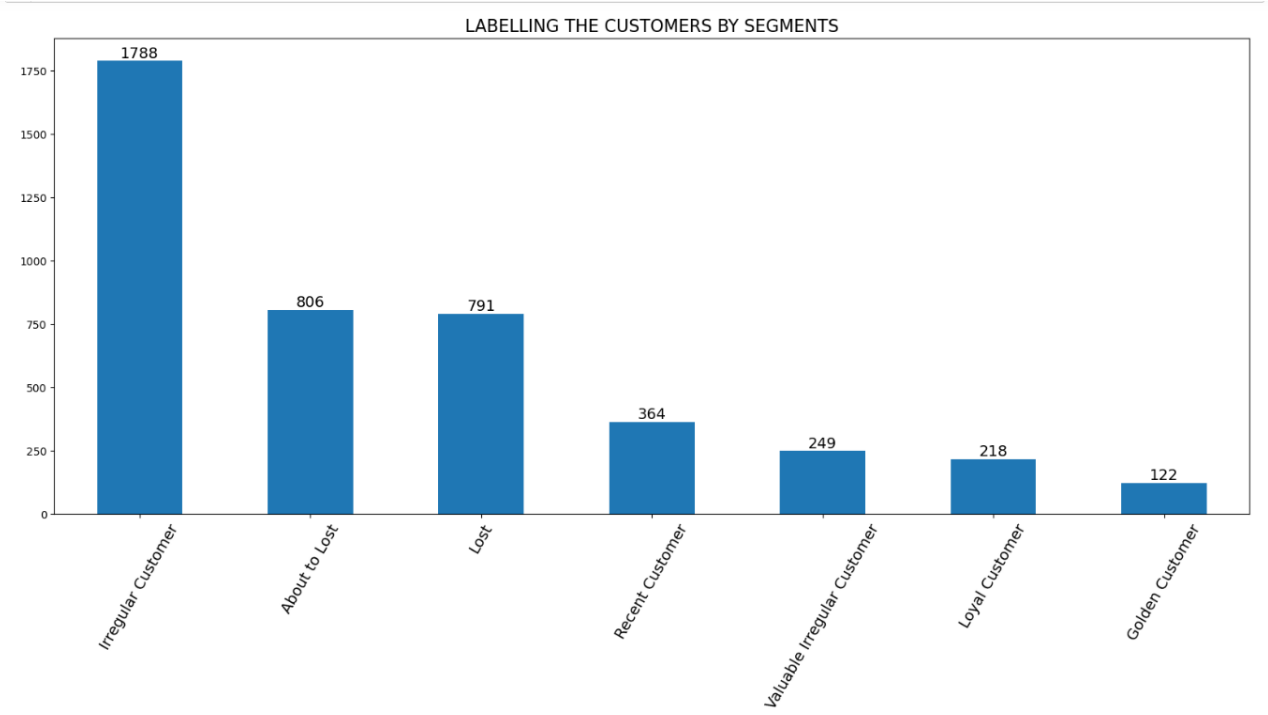
4338 rows × 18 columns

The distribution of customers into segments based on monthly analysis is as illustrated below:

```
df_monthly_spending.CustomerSegment.value_counts(ascending=True)
```

```
Golden Customer                122
Loyal Customer                 218
Valuable Irregular Customer    249
Recent Customer                364
Lost                           791
About to Lost                  806
Irregular Customer            1788
```

https://medium.com/@yuce.ahmet.tr/exploring-rfm-analysis-methodologies-and-innovative-approaches-360f82fc536b



Ultimately, when comparing the segments derived from the standard RFM analysis with those generated by the new monthly approach, we encounter a significantly different categorization and notice substantial differences in the number of customers falling into similar segments. While the standard analysis typically identifies 'Golden/Champion' customers as the primary group—a notion that might not seem very reasonable—the new approach reveals that the primary group consists of Irregular customers. It appears that in the analysis conducted using Recency, Frequency, and Monetary metrics, the significance of numerous low-value daily transactions diminishes, as does the consideration of monthly shopping and transaction amounts in the new calculation method. Consequently, in the new approach, which emphasizes regular monthly and higher-value transactions, the number of champions decreases. It's evident that the new approach provides a more balanced evaluation of loyalty and valuable customer status, revealing that the larger group doesn't actually comprise champions.

Monthly Analysis:

Irregular Customer	1788
About to Lost	806
Lost	791
Recent Customer	364
Valuable Irregular Customer	249
Loyal Customer	218
Golden Customer	122

Standard RFM Analysis:

Champions	1295
Potential Loyalist	1022
Lost	523
Still Got Hope	475
Give it a Try	382
Need to Get Them Back	333
Loyal Customers	198
Recent Customers	110

Conducting analyses monthly offers superiority over the standard RFM analysis due to its alignment with datasets that provide monthly snapshots, coherence with cohort analyses, and the preparation of company financial statements on a monthly basis. Additionally, it enables administrative and financial oversight and control on a monthly cadence, which enhances its effectiveness compared to the standard RFM analysis.

Now, let's walk through step by step how we conducted this monthly analysis.

1. Compose the Monthly Spending Table:

Initially, we will create a new table named “monthly_spending” that shows the monthly shopping totals for each customer and whether they shopped in each month. Then, using this table, we will segment the customers.

```
monthly_spending
```

	CustomerID	YearMonth	Revenue	CountMonthsShopped	ShoppingPercentage
0	12346.00	2011-01	77183.60	1	7.69
1	12347.00	2010-12	711.79	7	53.85
2	12347.00	2011-01	475.39	7	53.85
3	12347.00	2011-04	636.25	7	53.85
4	12347.00	2011-06	382.52	7	53.85
...	...	...	...	...	...
13049	18283.00	2011-10	112.99	10	76.92
13050	18283.00	2011-11	637.71	10	76.92
13051	18283.00	2011-12	208.00	10	76.92
13052	18287.00	2011-05	765.28	2	15.38
13053	18287.00	2011-10	1072.00	2	15.38

13054 rows × 5 columns

In this table, notice that while the total monthly spending for a customer appears in separate rows for each month they shopped, the total number of months shopped, and its percentage is repeated as a unique/single value across each row representing the customer's monthly shopping. The purchases of 4338 customers are displayed month by month, resulting in 13054 rows. Soon, we'll perform a pivot operation to transform each month in the dataset into a separate column, creating a table where customers aren't repeated.

To create the monthly spending table above, we utilized the code below.

```
# Calculate monthly spending for each customer
monthly_spending = df.groupby(['CustomerID', 'YearMonth'])['Revenue'].sum().reset_index()

# Find how many months each customer shopped
monthly_spending['CountMonthsShopped'] = monthly_spending.groupby('CustomerID')['YearMonth'].transform('count')

# Calculate the monthly shopping percentages for customers (divide by the total number of months)
total_months = monthly_spending['YearMonth'].nunique()
monthly_spending['ShoppingPercentage'] = (monthly_spending['CountMonthsShopped'] / total_months) * 100
```

This code snippet performs the following steps:

Calculate Monthly Spending: Groups the data by 'CustomerID' and 'YearMonth', then sums the 'Revenue' for each group, representing the monthly spending per customer.

Count Months Shopped: Creates a new column 'CountMonthsShopped' to determine how many months each customer has shopped. It counts the unique 'YearMonth' values for each 'CustomerID'.

Calculate Monthly Shopping Percentages: Calculates the percentage of months shopped for each customer concerning the total number of unique months available in the dataset ('total_months').

2. Pivot the monthly spending table by the "YearMonth" column and "Revenue" values.

```
df_monthly_spending
```

YearMonth	2010-12	2011-01	2011-02	2011-03	2011-04	2011-05	2011-06	2011-07	2011-08	2011-09	2011-10	2011-11	2011-12
CustomerID													
12346.00	0.00	77183.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12347.00	711.79	475.39	0.00	0.00	636.25	0.00	382.52	0.00	584.91	0.00	1294.32	0.00	224.82
12348.00	892.80	227.44	0.00	0.00	367.00	0.00	0.00	0.00	0.00	310.00	0.00	0.00	0.00
12349.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1757.55	0.00
12350.00	0.00	0.00	334.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...
18280.00	0.00	0.00	0.00	180.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
18281.00	0.00	0.00	0.00	0.00	0.00	0.00	80.82	0.00	0.00	0.00	0.00	0.00	0.00
18282.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.21	0.00	0.00	0.00	77.84
18283.00	0.00	213.75	100.95	0.00	115.60	85.22	296.52	139.89	0.00	134.90	112.99	637.71	208.00
18287.00	0.00	0.00	0.00	0.00	0.00	765.28	0.00	0.00	0.00	0.00	1072.00	0.00	0.00

4338 rows × 13 columns

To pivot the monthly spending table by expanding the unique values of YearMonth as new columns and filling them with Revenue values, we utilized the code below.

```
df_monthly_spending = monthly_spending.pivot(index='CustomerID', columns='YearMonth', values='Revenue').fillna(0)
```

This line of code uses the `pivot()` function in Pandas to restructure the DataFrame `monthly_spending` in a way that reorganizes the data into a pivot table format. Here's what each parameter does:

- `index='CustomerID'`: The 'CustomerID' column will be used as the index for the new DataFrame.
- `columns='YearMonth'`: The unique values in the 'YearMonth' column will be used to create new columns in the pivot table.
- `values='Revenue'`: The values in the 'Revenue' column will populate the cells of the pivot table.

This operation essentially transforms the DataFrame by rearranging the data. Each unique 'CustomerID' becomes a row index, each unique 'YearMonth' value becomes a column, and the 'Revenue' values corresponding to each customer and month pair fill in the cells of the pivot table. Any missing values (if a customer didn't spend in a particular month) are filled with zeros.

3. Calculate and add the total spending of each Customer to the pivot table:

```
df_monthly_spending
```

YearMonth	2010-12	2011-01	2011-02	2011-03	2011-04	2011-05	2011-06	2011-07	2011-08	2011-09	2011-10	2011-11	2011-12	TotalSales
CustomerID														
12346.00	0.00	77183.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	77183.60
12347.00	711.79	475.39	0.00	0.00	636.25	0.00	382.52	0.00	584.91	0.00	1294.32	0.00	224.82	4310.00
12348.00	892.80	227.44	0.00	0.00	367.00	0.00	0.00	0.00	0.00	310.00	0.00	0.00	0.00	1797.24
12349.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1757.55	0.00	1757.55
12350.00	0.00	0.00	334.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	334.40
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
18280.00	0.00	0.00	0.00	180.60	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	180.60
18281.00	0.00	0.00	0.00	0.00	0.00	0.00	80.82	0.00	0.00	0.00	0.00	0.00	0.00	80.82
18282.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	100.21	0.00	0.00	0.00	77.84	178.05
18283.00	0.00	213.75	100.95	0.00	115.60	85.22	296.52	139.89	0.00	134.90	112.99	637.71	208.00	2045.53
18287.00	0.00	0.00	0.00	0.00	0.00	765.28	0.00	0.00	0.00	0.00	1072.00	0.00	0.00	1837.28

4338 rows × 14 columns

To add the “TotalSales” feature to the table, we utilized the code below.

```
# Add a new column by calculating the total sales for each customer
df_monthly_spending['TotalSales'] = df_monthly_spending.sum(axis=1)
```

4. Add "CountMonthsShopped" and "ShoppingPercentage" features to the pivot table:

```
df_monthly_spending
```

	CustomerID	2010-12	2011-01	2011-02	...	TotalSales	CountMonthsShopped	ShoppingPercentage
0	12346.00	0.00	77183.60	0.00	...	77183.60	1	7.69
1	12347.00	711.79	475.39	0.00	...	4310.00	7	53.85
2	12348.00	892.80	227.44	0.00	...	1797.24	4	30.77
3	12349.00	0.00	0.00	0.00	...	1757.55	1	7.69
4	12350.00	0.00	0.00	334.40	...	334.40	1	7.69
...	...	...	...	...	...	...	...	...
4333	18280.00	0.00	0.00	0.00	...	180.60	1	7.69
4334	18281.00	0.00	0.00	0.00	...	80.82	1	7.69
4335	18282.00	0.00	0.00	0.00	...	178.05	2	15.38
4336	18283.00	0.00	213.75	100.95	...	2045.53	10	76.92
4337	18287.00	0.00	0.00	0.00	...	1837.28	2	15.38

4338 rows × 18 columns

To create the table above, first, we will create the shopping_counts_percentage_df below and then merge it with our pivot table df_monthly_spending.

```
shopping_counts_percentage_df
```

	CustomerID	CountMonthsShopped	ShoppingPercentage
0	12346.00	1	7.69
1	12347.00	7	53.85
8	12348.00	4	30.77
12	12349.00	1	7.69
13	12350.00	1	7.69
...	...	...	...
13038	18280.00	1	7.69
13039	18281.00	1	7.69
13040	18282.00	2	15.38
13042	18283.00	10	76.92
13052	18287.00	2	15.38

4338 rows × 3 columns

To create the table above, we use the code line below.

```
# Create a new DataFrame containing 'CustomerID', 'CountMonthsShopped', and 'ShoppingPercentage' columns
shopping_counts_percentage_df = monthly_spending[['CustomerID','CountMonthsShopped','ShoppingPercentage']].drop_duplicates()
```

Finally, we merge these tables using the code below.

```
# Merge with df_monthly_spending using 'CustomerID' as the index
df_monthly_spending = df_monthly_spending.merge(shopping_counts_percentage_df, on='CustomerID', how='left')
```

5. Segment Customers by pre-defined criteria/conditions:

We will create this final table showing the customer segments by using the code below.

```
df_monthly_spending
```

	CustomerID	2010-12	2011-01	2011-02	...	TotalSales	CountMonthsShopped	ShoppingPercentage	CustomerSegment
0	12346.00	0.00	77183.60	0.00	...	77183.60	1	7.69	Lost
1	12347.00	711.79	475.39	0.00	...	4310.00	7	53.85	Valuable Irregular Customer
2	12348.00	892.80	227.44	0.00	...	1797.24	4	30.77	About to Lost
3	12349.00	0.00	0.00	0.00	...	1757.55	1	7.69	Recent Customer
4	12350.00	0.00	0.00	334.40	...	334.40	1	7.69	Lost
...	...	...	...	...	...	...	...	...	...
4333	18280.00	0.00	0.00	0.00	...	180.60	1	7.69	Lost
4334	18281.00	0.00	0.00	0.00	...	80.82	1	7.69	Irregular Customer
4335	18282.00	0.00	0.00	0.00	...	178.05	2	15.38	Irregular Customer
4336	18283.00	0.00	213.75	100.95	...	2045.53	10	76.92	Loyal Customer
4337	18287.00	0.00	0.00	0.00	...	1837.28	2	15.38	Irregular Customer

4338 rows × 18 columns

```python
# Define the conditions for customer segmentation
def customer_segment(row):
    last_three_months = row.iloc[-6:-3]   # the last three columns(-3) are not Date(Month) columns.
    last_seven_months = row.iloc[-10:-3]

    if last_seven_months.eq(0).all():
        return "Lost"
    elif last_three_months.eq(0).all() and row.iloc[-9:-6].gt(0).any():
        return "About to Lost"
    elif row.iloc[1:12].eq(0).all() and row.iloc[12:14].gt(0).any():
        return "Recent Customer"
    elif (row['ShoppingPercentage'] >= 84) & (row['TotalSales'] >= df_monthly_spending['TotalSales'].quantile(0.80)):
        return "Golden Customer"
    elif last_three_months.gt(0).any() and row['ShoppingPercentage'] >= 60:
        return "Loyal Customer"
    elif last_three_months.gt(0).any() and row['ShoppingPercentage'] >= 40:
        return "Valuable Irregular Customer"
    else:
        return "Irregular Customer"

# Apply the customer_segment function to create the "CustomerSegment" column
df_monthly_spending['CustomerSegment'] = df_monthly_spending.apply(customer_segment, axis=1)
```

This code defines a function `customer_segment()` that takes a row from a DataFrame as input and applies various conditions to classify customers into segments based on their shopping behavior over specific periods. According to the conditions written in the customer_segment function, customers are categorized as follows:

Lost Customer: A customer who has made zero purchases in the last seven months.

About to Be Lost: Customers with zero purchases in the last three months but with some purchases in the previous 3 months of last three months.

Recent Customer: A customer who has had purchases in the last 2 months but zero purchases before.

Golden Customer: Customers with a high ShoppingPercentage (above the 84th percentile) and high TotalSales (above the 80th percentile of the entire dataset).

Loyal Customer: Customers with purchases in the last 3 months and a ShoppingPercentage above 60%.

Valuable Irregular Customer: Customers with purchases in the last three months and a ShoppingPercentage above 40% but less than 60%.

Irregular Customer: Customers who do not fit into any of the previous categories.

Finally, these conditions are applied to the DataFrame `df_monthly_spending` using the `apply()` function along the rows (axis=1), creating a new column called "CustomerSegment" based on the segmentation results for each customer.

So, we've created our result table that segments our customers.

Note that we used **eq(0).all()** and **gt(0).any()** functions of the pandas library in our conditions:

- **eq(0).all()**: This checks if all values in the rows/DataFrame are equal to zero. If all values are zero, this expression returns **True**.
- **gt(0).any()**: This checks if any value in the rows/DataFrame is greater than zero. If any value is greater than zero, this expression returns **True**.

CONCLUSION:

RFM analysis serves as a powerful tool for businesses seeking to understand, segment, and strategize around their customer base. By harnessing the insights derived from Recency, Frequency, and Monetary metrics, businesses can drive effective marketing initiatives, bolster customer relationships, and ultimately foster long-term success.

In the exploration of RFM Analysis and its methodologies, we've navigated through established practices and innovative approaches, unraveling the diverse dimensions of customer segmentation and behavior analysis. RFM, standing for Recency, Frequency, and Monetary, serves as a potent tool to dissect customer behaviors and categorize them effectively.

The standard RFM analysis has been the cornerstone, but our journey revealed the limitations within this framework. The conventional approach might overshadow recent loyal customers' significance by frequency, undervaluing their importance amidst evolving market dynamics. For instance, while a decade-long loyal customer showcases significant loyalty, acknowledging the loyalty of those who've engaged consistently for 1-2 years in a dynamic online retail landscape is equally crucial.

Classical RFM analysis encounters challenges not only with Frequency but also with Recency and Monetary calculations. An important problem in the standard RFM analysis is that as the recency duration extends, its significance by magnitude diminishes. There's little distinction between a customer lost for one year and another lost for ten years. This underscores the diminishing relevance of recency duration in assessing customer value.

Comparing the shopping frequencies and expenditure totals of customers solely based on the duration of their loyalty which is not fixed for each customer, such as between a customer for two years versus one for five years, may not yield a fair assessment. Therefore, conducting calculations on a monthly basis allows for the computation of the average monthly frequencies and expenditures, enabling a more equitable comparison among customers. This observation underscores the necessity of recalibrating Frequency and Monetary metrics, highlighting the importance of monthly-based assessments for a more equitable evaluation of customer behaviors.

Our departure from daily calculations to a monthly tracking system reflected an evolved understanding. This innovative method highlighted a redefined customer landscape, reshaping the narrative. It shifted the focus from one segment, in this case, apparent champions, to a more nuanced view, spotlighting another segment,

irregular yet valuable customers. This recalibration emphasized the potency of regular, higher-value transactions, ushering in a new wave of loyalty interpretation.

The advantage of monthly analyses transcends mere alignment with datasets; it offers coherence for other types of analysis such as cohort analysis, and an enhanced grip on financial and administrative control. By elevating analyses to a monthly cadence, it solidifies its supremacy in steering businesses toward a more informed, strategic customer-centric approach.

In essence, this journey from established practices to innovative methodologies underlines the evolution necessary in the realm of RFM analysis. It champions a shift in perspective, urging businesses to recalibrate their understanding of customer loyalty and value, ultimately propelling them toward more informed and agile customer-centric strategies in today's ever-evolving market landscape.

Ahmet Yüce
Data Scientist

You can find the jupyter notebooks in the link below:

https://github.com/yuceahmet/CUSTOMER-SEGMENTATION