A
TERM PROJECT REPORT ON

"RFM ANALYSIS OF CUSTOMER SEGMENTATION, USING PYTHON & POWER BI"

UNDER
PROF. DR. PRADIP KUMAR BALA

BY
RACHIT MAHESHWARI (BA025-21)

M.B.A – BUSINESS ANALYTICS

2021-2023

INDIAN INSTITUTE OF MANAGEMENT
RANCHI

Table of Contents

RFM Analysis for Customer Segmentation, using Python and Power BI

Executive Summary

This report outlines the process and findings of an RFM analysis for customer segmentation using Python and Power BI. The primary objective of this project was to segment customers based on their purchasing behavior and identify the most valuable customers for targeted marketing campaigns.

The analysis was conducted using Python, where we explored the dataset, cleaned the data, and calculated the RFM scores for each customer. We then segmented the customers based on their RFM scores using a predefined set of segments. We also visualized the results using a scatter plot to understand the distribution of customers based on their recency, frequency, and monetary values.

The results showed that the majority of the customers fell into the 'hibernating', 'lost', and 'at risk' segments, indicating that there is a significant number of customers who have not made purchases in a while and need to be reactivated. On the other hand, the 'champions' and 'loyal customers' segments represent the most valuable customers who make frequent purchases and spend the most.

To further understand the data and create interactive visualizations, we used Power BI. We imported the cleaned data into Power BI and created several visualizations, including a heatmap and bar charts, to provide a comprehensive overview of the customer segments.

In conclusion, the RFM analysis and customer segmentation project showed that there are several customer segments that can be targeted for marketing campaigns to improve customer engagement and retention.

Using Python and Power BI, we were able to efficiently process and analyze the data, as well as create interactive visualizations for a more in-depth understanding of the results.

Aim

The aim of this project is to create a useful tool for sales managers that can aid in increasing sales and customer retention by identifying high-priority customers for outreach.

To accomplish this goal, I utilized RFM analysis, a commonly used technique in direct marketing and database marketing, especially in the retail industry.

The study focuses on customer segmentation through data visualization using Power BI, where RFM analysis was performed on the sales data of the company.

Requirements

- Jupyter notebook environment, with Python 3.7 or higher or Google Colaboratory for the execution of Python code.
- Power BI application for preparing the dashboard to visualize the customer segments after processing the CSV file.

Packages used in Python

- numpy
- pandas
- math
- datetime
- dataprep
- matplotlib

Data Acquisition

This data has been obtained from real sales orders. It is acquired as a CSV file.
Dataset Link -
https://docs.google.com/spreadsheets/d/1y6qRpNMzBzOYLlz19Fhw6lW1DWQKeY3bWj3xQztaEQs/edit#gid=1602865175

Total Rows:- 235574
Total Columns:- 5

Dataset Description

1. **csv format**

	A	B	C	D
1	country;id;week.year;revenue;units			
2	KR;702234;03.2019;808,08;1			
3	KR;702234;06.2019;1606,80;2			
4	KR;3618438;08.2019;803,40;1			
5	KR;3618438;09.2019;803,40;1			
6	KR;3618438;09.2019;803,40;1			
7	KR;3618438;13.2019;2376,42;3			
8	KR;3618438;12.2019;1198,74;1			
9	KR;702234;16.2019;797,82;1			
10	KR;3618438;18.2019;399,54;1			
11	KR;3618438;16.2019;1596,00;2			

Fig-1 (Screenshot of the dataset used in csv format)

2. **xlsx format (Tabular format)**

	A	B	C	D	E
1	country	id	week.year	revenue	units
2	KR	702234	3.2019	808,08	1
3	KR	702234	6.2019	1606,80	2
4	KR	3618438	8.2019	803,40	1
5	KR	3618438	9.2019	803,40	1
6	KR	3618438	9.2019	803,40	1
7	KR	3618438	13.2019	2376,42	3
8	KR	3618438	12.2019	1198,74	1
9	KR	702234	16.2019	797,82	1
10	KR	3618438	18.2019	399,54	1
11	KR	3618438	16.2019	1596,00	2
12	KR	3618438	17.2019	1206,78	1
13	KR	3618438	18.2019	399,54	1
14	KR	3618438	21.2019	1594,98	2
15	KR	3618438	19.2019	1206,78	1

Fig-2 (Screenshot of the dataset used in xlsx format)

Meta Data:

1. **country –** Country name codes. (Nominal)
2. **id –** Customer id (Numeric - int)
3. **week.year –** Transaction date (Date)
4. **revenue –** Revenue from a particular order (Numeric – float)
5. **units –** Number of units bought (Numeric – int)

Methodology (Process Flow)

Following is the series of processes or steps that are being taken in order to determine the frequency, recency, and monetary values over the last 365 days for each customer using a dataset of sales orders over a given period of time using Python.

1. Firstly, certain libraries will be imported to work on the dataset.
2. In the beginning, the dataset will go through the EDA analysis, that is, exploratory data analysis for data pre-processing which will involve –

    a. Data Preparation - Importing the CSV file and put it into a dataframe with pandas
    b. Data Cleaning – Removing the unnecessary data and null values, processing it in a way that will be useful later in the project. It will help in checking the quality of the data.
    c. Data Exploration – Exploring the data with basic visualizations to get insights about the data which will help in identifying patterns and relationship within the dataset.

3. After this, data modelling will be performed to transform the data to obtain RFM values. For this, RFM scores will be calculated

4. After this, customer will be segregated based on the above RFM scores into various different categories

5. Finally, all the above analysis done in Python will be executed in Power BI dashboard that processes the CSV result for better visualization.

Code Results & Observations

Import of desired libraries

```
import numpy as np
import pandas as pd
import math
from datetime import timedelta, datetime
from dataprep.clean import clean_country
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use("seaborn")
plt.rcParams["figure.figsize"] = (20, 5)
```

Above code will import libraries such as,

a) **Numpy -** provides support for large, multi-dimensional arrays and matrices, along with a large collection of mathematical functions.
b) **Pandas -** provides data structures and tools for data manipulation and analysis.
c) **Math -** provides mathematical functions defined by the C standard.
d) from datetime import timedelta, datetime imports two classes from the built-in datetime module.
    - **timedelta -** used to represent a duration or difference between two dates or times.
    - **datetime -** used to represent a specific date and time.
e) 'from **dataprep.clean** import **clean_country'** imports a function called clean_country from the dataprep.clean module. This function is used to clean and standardize country names in a dataset.
f) **Matplotlib.pyplot -** provides a variety of tools for data visualization.
    - pyplot is a sub-library of Matplotlib that provides a convenient interface for creating plots and charts.

Steps for EDA Analysis

Now we will be doing the EDA analysis on the above dataset and various steps of which are described below: -

Step 1: Data Preparation

a) We utilize pandas to import the CSV and store it as a dataframe called df1. Our understanding of the domain tells us that each row in the dataset corresponds to a unique order.

```
df1 = pd.read_csv('C:/Users/rachi/Downloads/sales_asia.csv',
                  dtype={'week.year': str},
                  sep=';',
                  decimal=',')
```

```
df1.head()
```

	country	id	week.year	revenue	units
0	KR	702234	03.2019	808.08	1
1	KR	702234	06.2019	1606.80	2
2	KR	3618438	08.2019	803.40	1
3	KR	3618438	09.2019	803.40	1
4	KR	3618438	09.2019	803.40	1

b) Checking number of rows and columns in the dataset

```
df1.shape
```

```
(235574, 5)
```

c) We split the 'week.year' column into two columns, namely 'week' and 'year'.

```
# Splitting 'week.year' column on '.' and creating 'week' and 'year' columns

df1['week'] = df1['week.year'].astype(str).str.split('.').str[0]
df1['year'] = df1['week.year'].astype(str).str.split('.').str[1]
```

d) To facilitate analysis, we use the datetime package to convert the date, which is represented as the week of the year, into the year-month-day format.

```
# Converting year and week into date, using Monday as first day of the week

df1['date'] = pd.to_datetime(df1['year'].map(str) + df1['week'].map(str) + '-1', format='%Y%W-%w')
```

```
df1.head()
```

	country	id	week.year	revenue	units	week	year	date
0	KR	702234	03.2019	808.08	1	03	2019	2019-01-21
1	KR	702234	06.2019	1606.80	2	06	2019	2019-02-11
2	KR	3618438	08.2019	803.40	1	08	2019	2019-02-25
3	KR	3618438	09.2019	803.40	1	09	2019	2019-03-04
4	KR	3618438	09.2019	803.40	1	09	2019	2019-03-04

```
df1.columns
```

```
Index(['country', 'id', 'week.year', 'revenue', 'units', 'week', 'year',
       'date'],
      dtype='object')
```

e) Eliminating unnecessary columns from the dataframe df1.

```
# Removing unnecesary columns

df2 = df1.drop(['week.year', 'week', 'year'], axis=1)
```

```
df2.head()
```

	country	id	revenue	units	date
0	KR	702234	808.08	1	2019-01-21
1	KR	702234	1606.80	2	2019-02-11
2	KR	3618438	803.40	1	2019-02-25
3	KR	3618438	803.40	1	2019-03-04
4	KR	3618438	803.40	1	2019-03-04

f) We rename the column 'revenue' as 'monetary' in accordance with RFM analysis conventions.

```
#Rename columns

df2.rename({'revenue': 'monetary'}, axis="columns", inplace=True)
```

```
df2.head()
```

	country	id	monetary	units	date
0	KR	702234	808.08	1	2019-01-21
1	KR	702234	1606.80	2	2019-02-11
2	KR	3618438	803.40	1	2019-02-25
3	KR	3618438	803.40	1	2019-03-04
4	KR	3618438	803.40	1	2019-03-04

Step 2: Raw Data Description

a) Basic details/information about the dataframe df2

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235574 entries, 0 to 235573
Data columns (total 5 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   country   235574 non-null  object
 1   id        235574 non-null  int64
 2   monetary  235574 non-null  float64
 3   units     235574 non-null  int64
 4   date      235574 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(2), object(1)
memory usage: 9.0+ MB
```

b) Basic statistical description of the dataframe df2

```
df2.describe()
```

	id	monetary	units
count	2.355740e+05	2.355740e+05	235574.000000
mean	3.193118e+06	2.840211e+03	8.599642
std	7.371744e+06	2.247532e+04	602.939290
min	6.000180e+05	-1.061539e+05	-150000.000000
25%	2.214396e+06	3.994800e+02	1.000000
50%	3.140856e+06	1.150320e+03	1.000000
75%	3.892650e+06	2.216160e+03	2.000000
max	2.419308e+08	2.415857e+06	150000.000000

- The dataset covers a period of time and consists of **235,574 transactions and 5 columns.**
- The **largest transaction** in terms of units was **150,000.**
- However, it appears that there was also a **return** of the same amount, resulting in a **negative 150,000 units.**
- The **costliest purchase** made during this time period was **2.41 million**.

c) Checking if there are null values in the dataframe

```
df2.isnull().sum()
```

```
country     0
id          0
monetary    0
units       0
date        0
dtype: int64
```

d) Now, let's take a look at the time frame that is covered by the dataset:

```
# Let's view the period of time included in the dataset

df2['date'].min()
```

```
Timestamp('2019-01-07 00:00:00')
```

```
df2['date'].max()
```

```
Timestamp('2020-11-30 00:00:00')
```

e) Next, we'll examine the number of countries in which sales were made during this period:

```
# Let's explore in how many different countries we have sales in that period

df2['country'].unique()
```

```
array(['KR', 'PK', 'MM', 'VN', 'IN', 'SA', 'PH', 'AF', 'CN', 'BD', 'ID',
       'TH', 'IQ', 'MY', 'JP', 'IR', 'TR', 'UZ'], dtype=object)
```

```
df2['country'].nunique()
```

```
18
```

f) With the dataprep.clean package we can get the full country names after transforming the country codes**:**

```
# Transforming country codes into full country names with clean_country function
# from dataprep library

clean_country(df2, "country")['country_clean'].unique()
```

```
Country Cleaning Report:
        235574 values cleaned (100.0%)
Result contains 235574 (100.0%) values in the correct format and 0 null values (0.0%)

array(['South Korea', 'Pakistan', 'Myanmar', 'Vietnam', 'India',
       'Saudi Arabia', 'Philippines', 'Afghanistan', 'China',
       'Bangladesh', 'Indonesia', 'Thailand', 'Iraq', 'Malaysia', 'Japan',
       'Iran', 'Turkey', 'Uzbekistan'], dtype=object)
```

g) The total count of customers across all countries:

```
df2['id'].nunique()
```

```
21837
```

h) We set the date as the index in order to plot the time series.

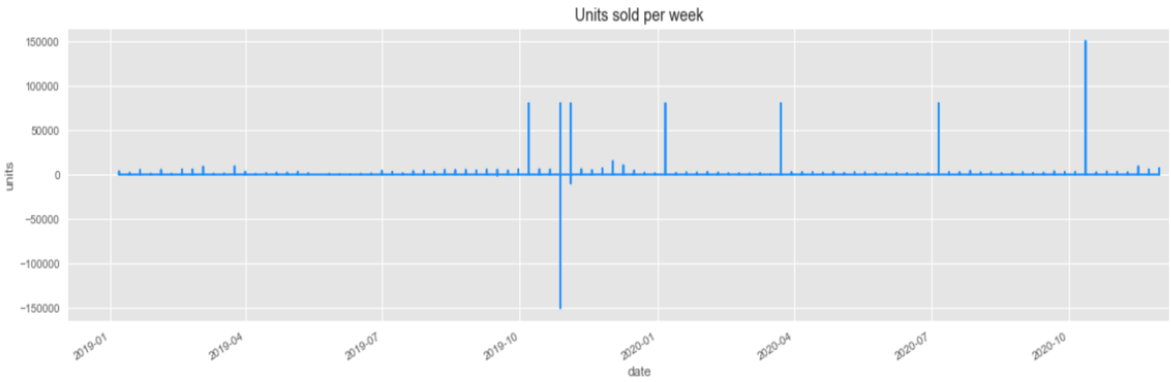
```
df2b = df2.set_index("date")
df2b.head()
```

date	country	id	monetary	units
2019-01-21	KR	702234	808.08	1
2019-02-11	KR	702234	1606.80	2
2019-02-25	KR	3618438	803.40	1
2019-03-04	KR	3618438	803.40	1
2019-03-04	KR	3618438	803.40	1

Step 3: Data Exploration

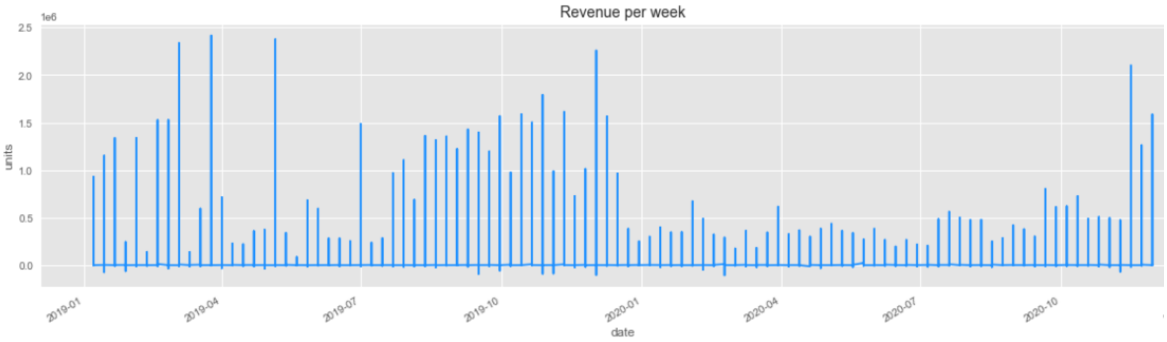
a) Creating a line plot that represents the weekly sales of a product **‘Units sold per week’**, using a dataset. The graph includes descriptive labels and formatting, which improve its visual clarity and legibility.

```
plt.style.use('ggplot')
plt.title('Units sold per week')
plt.ylabel('units')
plt.xlabel('date');
df2b['units'].plot(figsize=(20,5), c='dodgerblue');
```



b) Creating a line plot that represents the weekly revenue of a product **‘Revenue per week’**, using a dataset. The graph includes descriptive labels and formatting, which improve its visual clarity and legibility.

```python
plt.style.use('ggplot')
plt.title('Revenue per week')
plt.ylabel('units')
plt.xlabel('date');
df2b['monetary'].plot(figsize=(20,5), c='dodgerblue');
```



c) In order to enhance the clarity of the plots, we transform the dates to monthly periods.

```python
df2c = df2b.to_period("M")
```

```python
df2c.head()
```

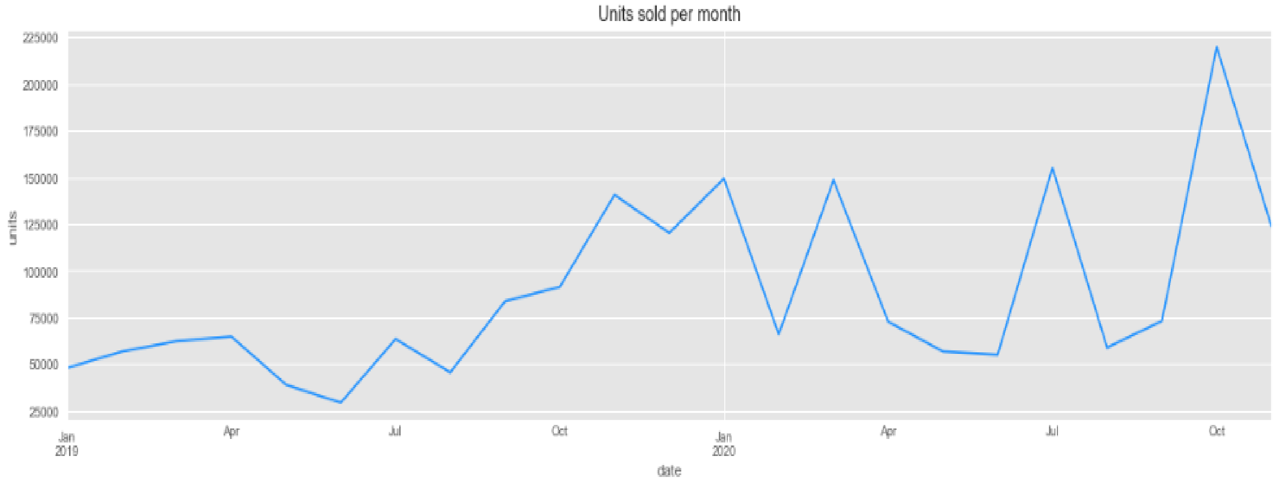
date	country	id	monetary	units
2019-01	KR	702234	808.08	1
2019-02	KR	702234	1606.80	2
2019-02	KR	3618438	803.40	1
2019-03	KR	3618438	803.40	1
2019-03	KR	3618438	803.40	1

d) Now, we group the units and revenue by the same time period and aggregate the values.

Units chart:

Generating a line plot that displays the total number of **'units sold per month'**, using a dataset. The graph includes descriptive labels and formatting to make it easy to read. By using the **'groupby' function to aggregate and summarize the data by month**, the graph provides a more comprehensive overview of sales trends at a higher level.

```python
plt.style.use('ggplot')
df2c['units'].groupby('date').agg(sum).plot(figsize=(20,5), c='dodgerblue')
plt.title('Units sold per month')
plt.ylabel('units')
plt.xlabel('date');
```



Revenue chart:

Generating a line plot that displays the total **'revenue per month'**, using a dataset. The graph includes descriptive labels and formatting to make it easy to read. By using the **'groupby' function to aggregate and summarize the data by month**, the graph provides a more comprehensive overview of sales trends at a higher level.

```python
plt.style.use('ggplot')
df2c['monetary'].groupby('date').agg(sum).plot(figsize=(20,5), c='dodgerblue')
plt.title('Revenue per month')
plt.ylabel('revenue')
plt.xlabel('date');
```

Modelling

Step 4: Altering the data to derive RFM values.

To obtain RFM values, we will transform the data by assigning scores to each customer based on their purchase behavior. Before doing so, we will create new features, namely 'recency', 'frequency', and 'monetary', based on the customer's purchasing history.

The **'recency' feature** will be determined by finding the minimum value of 'days_since_last_purchase' for each customer.

The **'frequency' feature** will be calculated by counting the total number of orders made by each customer during a specific period.

The **'monetary' feature** will be calculated by summing up the total value of all purchases made by each customer during the same period.

Once these features are created, we can assign scores to each customer based on their recency, frequency, and monetary behavior. This will allow us to gain a better understanding of customer behavior and create targeted marketing strategies.

a) We will narrow our focus to sales made within the past 365 days, starting from the most recent date.

```
period = 365
date_N_days_ago = df2['date'].max() - timedelta(days=period)
```


b) We eliminate the rows with dates that precede 365 days ago.

```
df2 = df2[df2['date']> date_N_days_ago]
```

```
df2.reset_index(drop=True, inplace=True)
```

```
df2.head()
```

	country	id	monetary	units	date
0	KR	4375152	773.58	1	2019-12-16
1	KR	705462	337.26	1	2019-12-09
2	KR	705462	337.26	1	2019-12-23
3	KR	705462	421.56	2	2019-12-16
4	KR	706854	391.50	1	2019-12-09

c) Basic information about the dataframe df2

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 124640 entries, 0 to 124639
Data columns (total 5 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   country   124640 non-null  object
 1   id        124640 non-null  int64
 2   monetary  124640 non-null  float64
 3   units     124640 non-null  int64
 4   date      124640 non-null  datetime64[ns]
dtypes: datetime64[ns](1), float64(1), int64(2), object(1)
memory usage: 4.8+ MB
```

d) There are customers with the same 'id' in several countries. This causes errors in the monetary values. We will solve this by creating a new feature: a unique 'id+' identifier that combines country code and customer id.

```
df3 = df2.copy()

df3['id+'] = df3['country'].map(str) + df3['id'].map(str)
```

```
df3.head()
```

	country	id	monetary	units	date	id+
0	KR	4375152	773.58	1	2019-12-16	KR4375152
1	KR	705462	337.26	1	2019-12-09	KR705462
2	KR	705462	337.26	1	2019-12-23	KR705462
3	KR	705462	421.56	2	2019-12-16	KR705462
4	KR	706854	391.50	1	2019-12-09	KR706854

e) We set the NOW date as one day after the date of the last sale.

```
NOW = df3['date'].max() + timedelta(days=1)
NOW
```

```
Timestamp('2020-12-01 00:00:00')
```

f) We create a new column called 'days_since_last_purchase' in the dataset, which calculates the number of days between the purchase date and the latest date.

```
df3['days_since_purchase'] = df3['date'].apply(lambda x:(NOW - x).days)
```

```
df3.head()
```

	country	id	monetary	units	date	id+	days_since_purchase
0	KR	4375152	773.58	1	2019-12-16	KR4375152	351
1	KR	705462	337.26	1	2019-12-09	KR705462	358
2	KR	705462	337.26	1	2019-12-23	KR705462	344
3	KR	705462	421.56	2	2019-12-16	KR705462	351
4	KR	706854	391.50	1	2019-12-09	KR706854	358

g) We will determine the recency score for each customer based on their 'days_since_last_purchase' column, where the recency score will be the minimum value for each customer.

```
aggr = {
    'days_since_purchase': lambda x:x.min(),
    'date': lambda x: len([d for d in x if d >= NOW - timedelta(days=period)])
}
```

h) We will calculate the frequency for each customer, which is the total number of orders they made during the given period.

```
rfm = df3.groupby(['id', 'id+', 'country']).agg(aggr).reset_index()
rfm.rename(columns={'days_since_purchase': 'recency',
                    'date': 'frequency'},
           inplace=True)
```

i) Getting the number of data values in the 'frm' dataframe

Rows: 16569
Columns: 5

```
rfm
```

	id	id+	country	recency	frequency
0	600018	CN600018	CN	29	7
1	600060	CN600060	CN	155	1
2	600462	CN600462	CN	211	2
3	600888	CN600888	CN	8	3
4	601014	CN601014	CN	225	1
...	...	...	...	...	...
16564	241575552	IQ241575552	IQ	15	1
16565	241794972	IQ241794972	IQ	351	1
16566	241888554	IQ241888554	IQ	43	1
16567	241900254	IQ241900254	IQ	8	62
16568	241930824	IQ241930824	IQ	36	2

16569 rows × 5 columns

j) We calculate the revenue generated by each customer within the last 365 days.

```
df3[df3['date'] >= NOW - timedelta(days=period)]\
    .groupby('id+')['monetary'].sum()
```

k) Retrieving only the monetary value for a particular customer with id 3790218.

```
df3[ (df3['id'] == 3790218) & (df3['date'] >= NOW - timedelta(days=period))]\
    .groupby('id+')['monetary'].sum()
```

```
id+
AF3790218       9706.08
BD3790218       7267.38
CN3790218     716199.60
ID3790218      49154.22
IQ3790218       1243.08
MM3790218       7110.60
PH3790218       1013.58
PK3790218     211108.20
TH3790218       1245.48
TR3790218      16072.02
VN3790218       3377.34
Name: monetary, dtype: float64
```

l) To ensure the accuracy of the monetary value, we will verify it by checking our biggest customer's transaction history.

```
# Checking monetary value is correct by checking on our biggest customer

rfm[rfm['monetary']==rfm['monetary'].max()]
```

	id	id+	country	recency	frequency	monetary
173	638544	CN638544	CN	1	217	21482332.56

m) We verify if customers belonging to different countries have distinct monetary values by examining the data of customer with id 3790218.

```
# We check that customers with id 3790218 get a different monetary value per country

rfm[rfm['id']==3790218]
```

	id	id+	country	recency	frequency	monetary
11057	3790218	AF3790218	AF	309	1	9706.08
11058	3790218	BD3790218	BD	176	4	7267.38
11059	3790218	CN3790218	CN	1	60	716199.60
11060	3790218	ID3790218	ID	260	9	49154.22
11061	3790218	IQ3790218	IQ	176	1	1243.08
11062	3790218	MM3790218	MM	183	3	7110.60
11063	3790218	PH3790218	PH	127	3	1013.58
11064	3790218	PK3790218	PK	43	5	211108.20
11065	3790218	TH3790218	TH	295	1	1245.48
11066	3790218	TR3790218	TR	29	10	16072.02
11067	3790218	VN3790218	VN	302	1	3377.34

n) We are adding the revenue generated by each customer in the previous period, which is stored in dataframe df3, to the rfm dataframe.

```
rfm['monetary'] = rfm['id+']\
    .apply(lambda x: df3[ (df3['id+'] == x) & (df3['date'] >= NOW - timedelta(days=period))]\
    .groupby(['id', 'country']).sum().iloc[0,0])

rfm.head()
```

	id	id+	country	recency	frequency	monetary
0	600018	CN600018	CN	29	7	21402.78
1	600060	CN600060	CN	155	1	1201.14
2	600462	CN600462	CN	211	2	2033.64
3	600888	CN600888	CN	8	3	2335.80
4	601014	CN601014	CN	225	1	230.52

Step 5: Calculating RFM scores

a) To calculate RFM scores, we will rate the customers' recency, frequency, and monetary value factors on a scale of 1 to 5. We'll split each characteristic into groups with 20% of the samples using the quintiles method.

Recency scores will be lower numbers, while frequency and monetary value scores will be higher. These scores will be given to each customer as their R, F, and M scores.

```
quintiles = rfm[['recency', 'frequency', 'monetary']].quantile([.2, .4, .6, .8]).to_dict()
quintiles
```

```
{'recency': {0.2: 15.0, 0.4: 50.0, 0.6: 120.0, 0.8: 239.0},
 'frequency': {0.2: 1.0, 0.4: 2.0, 0.6: 4.0, 0.8: 9.0},
 'monetary': {0.2: 967.5,
  0.4: 2212.2,
  0.6: 4852.548000000001,
  0.8: 13957.500000000005}}
```

```
def r_score(x):
    if x <= quintiles['recency'][.2]:
        return 5
    elif x <= quintiles['recency'][.4]:
        return 4
    elif x <= quintiles['recency'][.6]:
        return 3
    elif x <= quintiles['recency'][.8]:
        return 2
    else:
        return 1
```

```python
def fm_score(x, c):
    if x <= quintiles[c][.2]:
        return 1
    elif x <= quintiles[c][.4]:
        return 2
    elif x <= quintiles[c][.6]:
        return 3
    elif x <= quintiles[c][.8]:
        return 4
    else:
        return 5
```

```python
rfm['r'] = rfm['recency'].apply(lambda x: r_score(x))
rfm['f'] = rfm['frequency'].apply(lambda x: fm_score(x, 'frequency'))
rfm['m'] = rfm['monetary'].apply(lambda x: fm_score(x, 'monetary'))
```

b) We will now combine the R, F, and M data to generate an overall RFM score for each customer through aggregation.

```python
rfm['rfm_score'] = rfm['r'].map(str) + rfm['f'].map(str) + rfm['m'].map(str)

rfm.head()
```

	id	country	recency	frequency	monetary	r	f	m	rfm_score
0	600018	CN	29	7	21402.78	4	4	5	445
1	600060	CN	155	1	1201.14	2	1	2	212
2	600462	CN	211	2	2033.64	2	2	2	222
3	600888	CN	8	3	2335.80	5	3	3	533
4	601014	CN	225	1	230.52	2	1	1	211

c) We can use the R, F, and M scores to create 125 customer segments with these values. However, we can reduce the number of segments by combining F and M scores, resulting in 11 segments.

fm = (f+m)/2

```python
def truncate(x):
    return math.trunc(x)
```

```python
rfm['fm'] = ((rfm['f'] + rfm['m'])/2).apply(lambda x: truncate(x))
```

```python
rfm.head()
```

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm
0	600018	CN	29	7	21402.78	4	4	5	445	4
1	600060	CN	155	1	1201.14	2	1	2	212	1
2	600462	CN	211	2	2033.64	2	2	2	222	2
3	600888	CN	8	3	2335.80	5	3	3	533	3
4	601014	CN	225	1	230.52	2	1	1	211	1

Step 6: The RFM analysis is used to segment customers into the following 11 categories:

- **Champions:** Customers who have recently made frequent and high-value purchases.
- **Loyal Customers:** Customers who make regular purchases and respond well to promotions.
- **Potential Loyalists:** Customers who are new but have made an average frequency of purchases.
- **Recent Customers:** Customers who have made a recent purchase but not often.
- **Promising:** Customers who have made recent purchases but haven't spent much yet.
- **Customers Needing Attention:** Customers who have above average recency, frequency, and monetary values but may not have made a purchase recently.
- **About To Sleep:** Customers who have below average recency and frequency and may be lost if not reactivated.
- **At Risk:** Customers who purchased often but a long time ago and need to be brought back.
- **Can't Lose Them:** Customers who used to purchase frequently but haven't returned for a long time.
- **Hibernating:** Customers whose last purchase was a long time ago and they have a low number of orders.
- **Lost:** Customers who made a purchase a long time ago and never returned.

a) We create a segment map of only **11 segments** based on only two scores: 'r' and 'fm'
This code block is mapping the RFM scores to customer segments using regular expressions. It creates a dictionary called segment_map that defines the segment names based on the combination of R, F, and M scores.

For example, customers with an R score of 5, an F score of 5, and an M score of 5 will have an RFM score of "555". This value will match the regular expression "55" in the segment_map dictionary, and the corresponding segment name "champions" will be assigned to these customers in the rfm['segment'] column.

```
segment_map = {
    r'22': 'hibernating',
    r'[1-2][1-2]': 'lost',
    r'15': 'can\'t lose',
    r'[1-2][3-5]': 'at risk',
    r'3[1-2]': 'about to sleep',
    r'33': 'need attention',
    r'55': 'champions',
    r'[3-5][4-5]': 'loyal customers',
    r'41': 'promising',
    r'51': 'new customers',
    r'[4-5][2-3]': 'potential loyalists'
}

rfm['segment'] = rfm['r'].map(str) + rfm['fm'].map(str)
rfm['segment'] = rfm['segment'].replace(segment_map, regex=True)
rfm.head()
```

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
0	600018	CN	29	7	21402.78	4	4	5	445	4	loyal customers
1	600060	CN	155	1	1201.14	2	1	2	212	1	lost
2	600462	CN	211	2	2033.64	2	2	2	222	2	hibernating
3	600888	CN	8	3	2335.80	5	3	3	533	3	potential loyalists
4	601014	CN	225	1	230.52	2	1	1	211	1	lost

b) Checking if there are null values in rfm

```
rfm.isnull().sum()
```

```
id           0
country      0
recency      0
frequency    0
monetary     0
r            0
f            0
m            0
rfm_score    0
fm           0
segment      0
dtype: int64
```

Step 7: Examining certain customer segments within the dataframe.

a) Can't Lose

```
rfm[rfm['segment']=="can't lose"].sort_values(by='monetary', ascending=False)
```

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
13028	4096386	JP	260	105	220267.86	1	5	5	155	5	can't lose
3502	2443284	IN	246	10	102208.02	1	5	5	155	5	can't lose
14174	4262646	IN	316	10	91909.44	1	5	5	155	5	can't lose
2435	1803672	IN	267	12	70506.96	1	5	5	155	5	can't lose
13254	4132968	VN	253	26	42535.14	1	5	5	155	5	can't lose
11222	3815274	IN	267	11	37968.72	1	5	5	155	5	can't lose
1458	1031454	PH	267	23	31833.30	1	5	5	155	5	can't lose
5437	2809158	IN	274	12	27150.12	1	5	5	155	5	can't lose
14644	4326906	IN	337	11	22351.68	1	5	5	155	5	can't lose
259	668070	MM	267	11	21886.92	1	5	5	155	5	can't lose
15331	4418268	SA	302	10	14295.54	1	5	5	155	5	can't lose

b) Need Attention

```
rfm[rfm['segment']=="need attention"].sort_values(by='monetary', ascending=False).head(10)
```

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
8245	3242664	TR	64	1	73823.58	3	1	5	315	3	need attention
13065	4107798	JP	120	2	67257.48	3	2	5	325	3	need attention
9847	3561900	ID	120	1	59700.00	3	1	5	315	3	need attention
6626	2921070	ID	71	2	34730.22	3	2	5	325	3	need attention
10009	3587772	CN	92	1	29961.00	3	1	5	315	3	need attention
3087	2131194	JP	57	1	28543.74	3	1	5	315	3	need attention
13463	4160490	JP	99	1	24842.22	3	1	5	315	3	need attention
1251	993414	KR	71	2	22018.32	3	2	5	325	3	need attention
3936	2544588	BD	71	2	19043.82	3	2	5	325	3	need attention
3616	2468010	TH	85	2	18599.58	3	2	5	325	3	need attention

c) Loyal Customers

```
rfm[rfm['segment']=='loyal customers'].sort_values(by='monetary', ascending=False).head(10)
```

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
15420	4422780	TR	92	13	2315341.14	3	5	5	355	5	loyal customers
2882	2030526	JP	22	50	1519339.86	4	5	5	455	5	loyal customers
3220	2182446	JP	29	18	1492057.68	4	5	5	455	5	loyal customers
12660	4041366	PK	50	9	736626.96	4	4	5	445	4	loyal customers
5612	2853774	VN	8	6	712230.00	5	4	5	545	4	loyal customers
10343	3649728	PH	29	81	579167.52	4	5	5	455	5	loyal customers
8284	3248568	TR	64	3	573792.72	3	3	5	335	4	loyal customers
15450	4427148	IN	29	14	502843.32	4	5	5	455	5	loyal customers
14678	4332210	ID	43	21	474773.40	4	5	5	455	5	loyal customers
2802	1985592	IQ	78	4	460390.86	3	3	5	335	4	loyal customers

d) Champions

```
rfm[rfm['segment']=='champions'].sort_values(by='monetary', ascending=False).head(10)
```

	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
173	638544	CN	1	217	21482332.56	5	5	5	555	5	champions
15436	4424580	CN	1	104	16912322.46	5	5	5	555	5	champions
14754	4341960	TR	1	200	16550997.90	5	5	5	555	5	champions
11942	3929094	ID	1	470	8748884.64	5	5	5	555	5	champions
9626	3520734	JP	1	198	6207519.96	5	5	5	555	5	champions
15915	4494150	TR	1	57	4874668.14	5	5	5	555	5	champions
10168	3618438	KR	8	1020	4615660.08	5	5	5	555	5	champions
14027	4245048	PH	1	993	4358515.98	5	5	5	555	5	champions
3050	2111100	IN	1	876	4270717.80	5	5	5	555	5	champions
11742	3894492	PH	8	63	4106366.22	5	5	5	555	5	champions

e) Customers with above-average monetary value who require attention.

```
rfm[(rfm['monetary']>rfm['monetary'].mean()) & (rfm['segment']=='need attention')]\
    .sort_values(by='monetary', ascending=False)
```

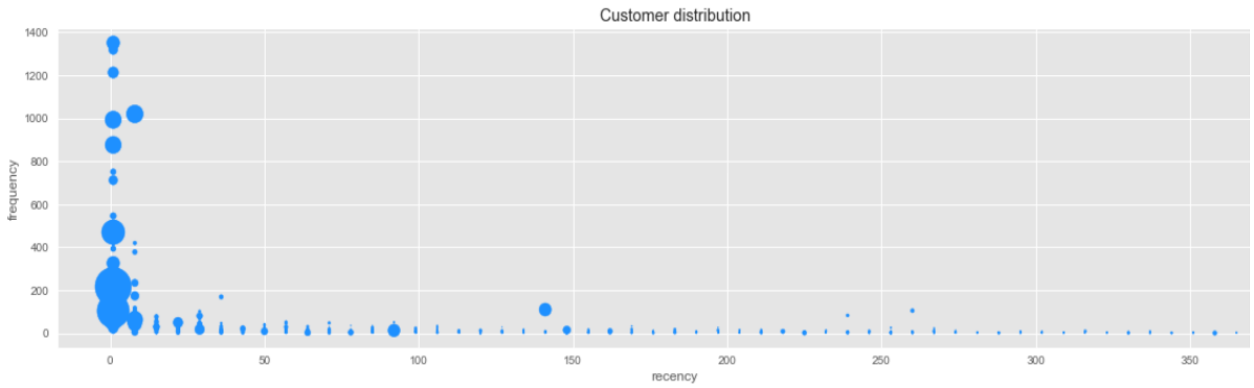
	id	country	recency	frequency	monetary	r	f	m	rfm_score	fm	segment
8245	3242664	TR	64	1	73823.58	3	1	5	315	3	need attention
13065	4107798	JP	120	2	67257.48	3	2	5	325	3	need attention
9847	3561900	ID	120	1	59700.00	3	1	5	315	3	need attention
6626	2921070	ID	71	2	34730.22	3	2	5	325	3	need attention
10009	3587772	CN	92	1	29961.00	3	1	5	315	3	need attention
3087	2131194	JP	57	1	28543.74	3	1	5	315	3	need attention
13463	4160490	JP	99	1	24842.22	3	1	5	315	3	need attention
1251	993414	KR	71	2	22018.32	3	2	5	325	3	need attention

Step 8: Scatter plot is used to examine how customers are distributed.

When plotting the data, the x-axis represents 'recency' while the y-axis represents 'frequency'. The size of the points on the graph is determined by 'monetary' value.

It can be observed that customers who spend the most also tend to purchase more frequently.

```
plt.style.use('ggplot')
rfm.plot.scatter(x='recency', y='frequency', s=rfm['monetary']*5e-5, figsize=(20,5), c='dodgerblue')
plt.gca().set(xlabel='recency', ylabel='frequency', title='Customer distribution');
```



Step 9: Saving (export) the dataframe as a CSV file to be able to process it in Power BI.

```python
# We export the dataframe to a CSV file for later processing it in Power BI
# (We added the parameter float_format='%.2f' for setting numbers to two decimals)

rfm.to_csv('rfm_asia.csv', encoding='utf-8', index=False, float_format='%.2f')
```

Step 10: Designing a dashboard on Power BI

a) After processing the CSV file resulting from executing the above Python code, the CSV file is transformed into a xlsx format.
b) Then this file that is 'rfm_asia.xlsx' is imported in Power BI to design the suitable dashboard

Customer Segmentation

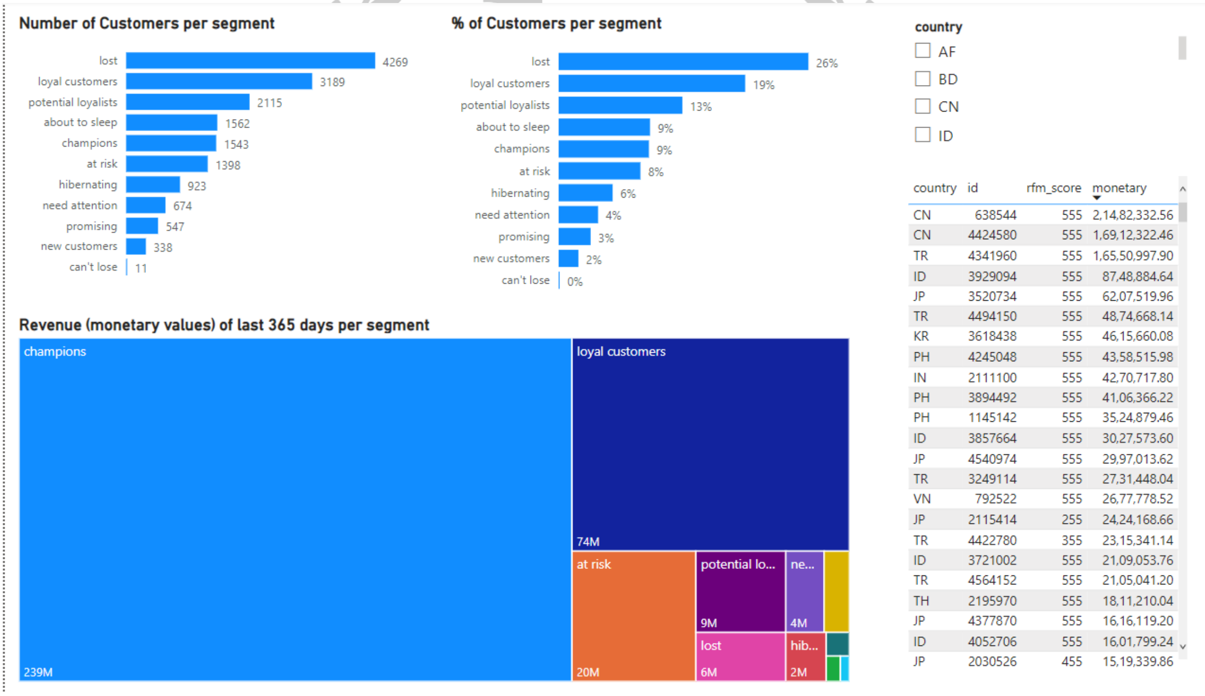


Fig-3 (Screenshot of Customer Segmentation in Power BI)

RFM Analysis

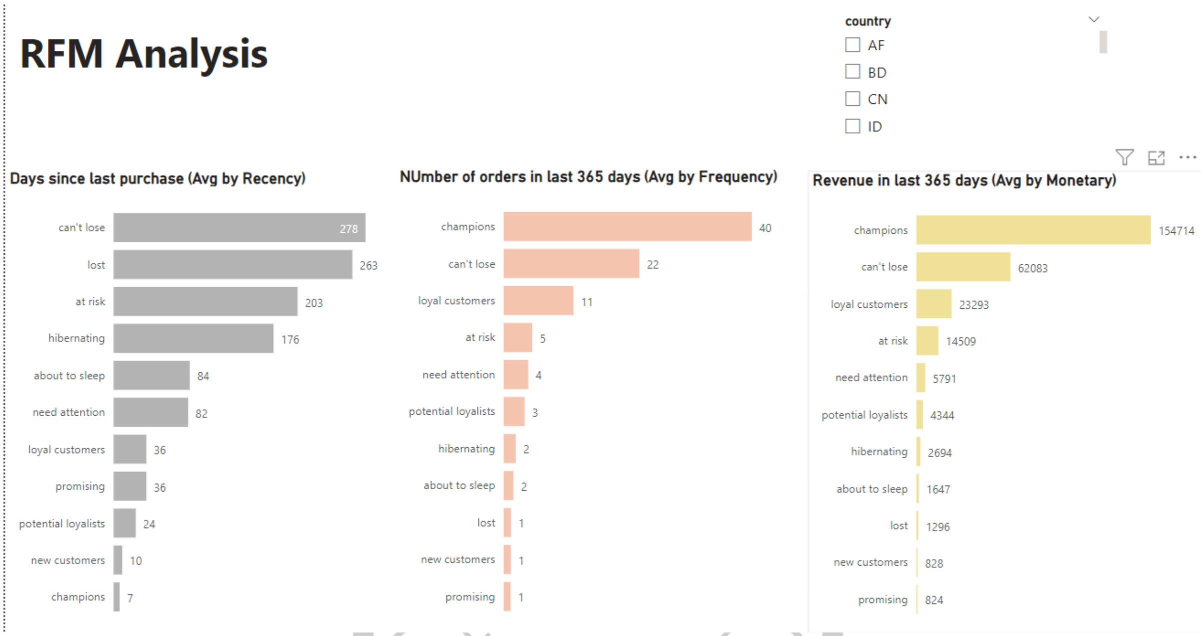


Fig-4 (Screenshot of RFM Analysis in Power BI)

Key Observations and Suggestions

- 32% of the customers are **'lost' or 'hibernating'** (meaning they have a few orders from long ago) which comprises almost 1/3rd of the total customers
  - For these customers, the marketing team has to design campaign to revive interest
  - Can offer other relevant products and special discounts
  - Recreate brand value

- 28% of the customers are either **'champions' or 'loyal customers'** (meaning they visit frequently and spend the most)
  - Almost 87% of the revenue is generated by only these two segments of customers
  - For these customers, the company should reward them as they will promote the brand
  - They can be early adopters of the new products
  - They can be targeted to upsell higher value products
  - The marketing team should ask for reviews and suggestions from these customers as it will increase their engagement

- **'Can't Lose'** customers frequently made the biggest purchases, but they have not returned for a long time. Though the percentage of these customers is very low, less than 1%, but we can not afford to lose them.
  - Marketing team should try to talk to them directly and win them back via renewals or new products

- 16% of the customers are either **'potential loyalists' or 'promising'** (recent customers who have purchased from us)
    - Since this is a huge chunk of customer, the marketing team should try to convert these customers to 'loyal customers' and then in a long term they can become 'champions'
    - They should be offered with a membership or be enrolled in a loyalty program
    - They should be recommended other products while shopping as they have spent a good amount and bought more than once
    - They should be offered free trials so as to create brand awareness

- 8% of the customers are **'at risk'** customers who are 3rd among all the segment in terms of revenue generated by them which is around 6% of the total revenue
    - These customers used to spent big money and also purchased often, but long time ago and we need to bring them back.
    - Marketing team should send personalized emails to these customers to reconnect with them.
    - Should also offer renewals and provide helpful resources.

Recommendations & Managerial Responsibility for each Customer Segment

Customer Segment	Activity	Actionable Tip
ABOUT TO SLEEP	Below average recency, frequency and monetary values. Will lose them if not reactivated.	Share valuable resources, recommend popular products / renewals at discount, reconnect with them.
AT RISK	Spent big money and purchased often. But long time ago. Need to bring them back!	Send personalized emails to reconnect, offer renewals, provide helpful resources.
CAN'T LOSE	Made biggest purchases, and often. But haven't returned for a long time.	Win them back via renewals or newer products, don't lose them to competition, talk to them.
CHAMPIONS	Bought recently, buy often and spend the most!	Reward them. Can be early adopters for new products. Will promote your brand.
HIBERNATING	Last purchase was long back, low spenders and low number of orders.	Offer other relevant products and special discounts. Recreate brand value.
LOST	Lowest recency, frequency and monetary scores.	Revive interest with reach out campaign, ignore otherwise.
LOYAL CUSTOMERS	Spend good money with us often. Responsive to promotions.	Upsell higher value products. Ask for reviews. Engage them.
NEED ATTENTION	Above average recency, frequency and monetary values. May not have bought very recently though.	Make limited time offers. Recommend based on past purchases. Reactivate them.
NEW CUSTOMERS	Bought most recently, but not often.	Provide on-boarding support, give them early success, start building relationship.
POTENTIAL LOYALISTS	Recent customers, but spent a good amount and bought more than once.	Offer membership / loyalty program, recommend other products.
PROMISING	Recent shoppers, but haven't spent much.	Create brand awareness, offer free trials.

Fig-5 (Screenshot of Actionable Tip for each customer segment)

Conclusion

RFM analysis is a powerful tool for segmenting customers based on their transactional behaviour. In this analysis, we divided customers into segments based on their **Recency, Frequency, and Monetary Value scores**, and then aggregated those scores to derive an overall RFM score for each customer.

The resulting segments allowed us to gain insights into customer behaviour and identify opportunities for customer retention and growth.

Our analysis revealed that the majority of customers who spend the most also purchase more frequently, indicating that there is a strong relationship between frequency and monetary value.

Additionally, we **identified several customer segments that require specific attention, including 'About To Sleep', 'At Risk', 'Can't Lose Them', and 'Hibernating' segments**. By focusing on these segments, businesses can create targeted marketing campaigns and initiatives to re-engage these customers and prevent churn.

Overall, RFM analysis is an effective way to segment customers and gain insights into their transactional behaviour, enabling businesses to create tailored strategies to improve customer retention and drive growth.

Code Link – https://colab.research.google.com/drive/1roEakth833cvPwDVvELi6sLlS0RDkHLn?usp=sharing

Learnings from the Project

1. **Importance of Customer Segmentation:**
   This project demonstrated the importance of segmenting customers based on their purchasing behaviour using RFM analysis. By segmenting customers into different groups, businesses can better understand their customers' needs and preferences, tailor marketing efforts, and improve customer retention.

2. **RFM Analysis:**
   RFM analysis is a powerful tool that helps businesses identify their most valuable customers based on their purchasing behaviour. By analysing recency, frequency, and monetary value, businesses can segment their customers into different categories and tailor their marketing efforts to each group.

3. **Data Pre-processing:**
   Data pre-processing is a critical step in any data analysis project. In this project, we cleaned the data and transformed it by normalizing and scaling it to make it suitable for analysis.

4. **Data Visualization:**
   Data visualization is an essential tool for exploring and presenting data. In this project, we used various visualizations line plots, scatter plots to analyse and explore the data. These visualizations helped us identify patterns and insights that were not immediately apparent from the raw data.

5. **Business Insights:**
   After transforming the data to get RFM scores, we analysed the customer data, and gained several insights that could be useful for businesses.
   For example, we identified the most valuable customers (Champions), the customers who are at risk of leaving (At Risk), and the customers who need attention (Customers Needing Attention), and other such segments of customers. This information can be used to develop targeted marketing campaigns to retain valuable customers and re-engage customers who are at risk of leaving.

6. **Usage of Tools:**
   - **Python –**
     Used this versatile programming language that can be used for various data analysis tasks, including data manipulation, data cleaning, and data visualization.
   - **Power BI –**
     Used this powerful business analytics tool that can be used to create interactive visualizations and reports

Overall, this project helped to reinforce the importance of data analysis and visualization in making informed business decisions. This project demonstrated the power of data analysis and visualization in understanding customer behaviour and developing effective marketing strategies.

By applying RFM analysis to customer data, businesses can gain valuable insights that can help them improve customer retention and grow their business.

References

1. https://www.analyticsvidhya.com/blog/2021/07/customer-segmentation-using-rfm-analysis/
2. https://medium.com/@ugursavci/customer-segmentation-using-rfm-analysis-in-python-218a3255f714
3. https://towardsdatascience.com/implementing-customer-segmentation-using-rfm-analysis-with-pyspark-3aed363f1d53
4. https://ploiitubsamon.medium.com/rfm-analysis-for-customer-segmentation-with-power-bi-5d2f5bd62038#:~:text=To%20determine%20the%20customer%20segmentation,is%20the%20latest%20purchase%20date.