CUSTOMERS SEGMENTATION IN MARKETING USING RFM ANALYSIS Recency, frequency, monetary (RFM segmentation)

RFM is a method used to identify and analyze customers based on the Recency of their last purchase, the Frequency of purchases, and the Monetary amount spent

important of customer segmentation:

Identify the best customers and focus on offering them the best treatment Improve product assortment based on your top customers' preferences Create better communication,offers,loyalty programs,experiences,and products for your power customers Increase customer engagement by leveraging the best-performing channels Improve and prioritize customer service team's activity Create lookalike audiences based on your top customers and improve the results of acquisition

In [4]:
```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
```

In [2]:
```
# load the data
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving RETAIL.csv to RETAIL.csv
```

In [5]:
```
# store and show the data
df=pd.read_csv('RETAIL.csv')
df
```

Out[5]:

	Invoice	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Total_Price
0	489434	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01	6.95	13085.0	United Kingdom	83.40
1	489434	PINK CHERRY LIGHTS	12	2009-12-01	6.75	13085.0	United Kingdom	81.00
2	489434	WHITE CHERRY LIGHTS	12	2009-12-01	6.75	13085.0	United Kingdom	81.00
3	489434	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01	2.10	13085.0	United Kingdom	100.80
4	489434	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01	1.25	13085.0	United Kingdom	30.00
...	...	...	...	...	...	...	...	...
797878	581587	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09	2.10	12680.0	France	12.60
797879	581587	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09	4.15	12680.0	France	16.60
797880	581587	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09	4.15	12680.0	France	16.60
797881	581587	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09	4.95	12680.0	France	14.85
797882	581587	POSTAGE	1	2011-12-09	18.00	12680.0	France	18.00

797883 rows × 8 columns

In [6]:
```
# Drop negative values
df=df[df['Total_Price']>0]
```

In [7]:
```
df
```

Out[7]:

	Invoice	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Total_Price
0	489434	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01	6.95	13085.0	United Kingdom	83.40
1	489434	PINK CHERRY LIGHTS	12	2009-12-01	6.75	13085.0	United Kingdom	81.00
2	489434	WHITE CHERRY LIGHTS	12	2009-12-01	6.75	13085.0	United Kingdom	81.00
3	489434	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01	2.10	13085.0	United Kingdom	100.80
4	489434	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01	1.25	13085.0	United Kingdom	30.00
...	...	...	...	...	...	...	...	...
797878	581587	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09	2.10	12680.0	France	12.60
797879	581587	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09	4.15	12680.0	France	16.60
797880	581587	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09	4.15	12680.0	France	16.60
797881	581587	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09	4.95	12680.0	France	14.85
797882	581587	POSTAGE	1	2011-12-09	18.00	12680.0	France	18.00

779423 rows × 8 columns

In [8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 779423 entries, 0 to 797882
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   Invoice      779423 non-null  object 
 1   Description  779423 non-null  object 
 2   Quantity     779423 non-null  int64  
 3   InvoiceDate  779423 non-null  object 
 4   Price        779423 non-null  float64
 5   Customer_ID  779423 non-null  float64
 6   Country      779423 non-null  object 
 7   Total_Price  779423 non-null  float64
dtypes: float64(3), int64(1), object(4)
memory usage: 53.5+ MB
```

In [9]:
```
# Convert invoice date column to datetime
df['InvoiceDate']=pd.to_datetime(df['InvoiceDate'])
```

```
<ipython-input-9-ca351bcae1de>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['InvoiceDate']=pd.to_datetime(df['InvoiceDate'])
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 779423 entries, 0 to 797882
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype         
---  ------       --------------   -----         
 0   Invoice      779423 non-null  object        
 1   Description  779423 non-null  object        
 2   Quantity     779423 non-null  int64         
 3   InvoiceDate  779423 non-null  datetime64[ns]
 4   Price        779423 non-null  float64       
 5   Customer_ID  779423 non-null  float64       
 6   Country      779423 non-null  object        
 7   Total_Price  779423 non-null  float64       
dtypes: datetime64[ns](1), float64(3), int64(1), object(3)
memory usage: 53.5+ MB
```

In [11]:
```
import datetime

today = datetime.date.today()
today_date = today.strftime('%d-%m-%Y')
print(today_date)
```

```
25-06-2023
```

In [12]:
```
df['InvoiceDate'].max()
```

Out[12]:
```
Timestamp('2011-12-09 00:00:00')
```

In [13]:
```
# create today_date column by adding 2 days on max date
max_date = df['InvoiceDate'].max()
today_date = max_date + pd.Timedelta(days=2)
```

```
# create today_date column
df['today_date'] = today_date
```

```
<ipython-input-13-a49d43ec2136>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['today_date'] = today_date
```

In [14]:
```
df
```

Out[14]:

	Invoice	Description	Quantity	InvoiceDate	Price	Customer_ID	Country	Total_Price	toda
0	489434	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	2009-12-01	6.95	13085.0	United Kingdom	83.40	2011
1	489434	PINK CHERRY LIGHTS	12	2009-12-01	6.75	13085.0	United Kingdom	81.00	2011
2	489434	WHITE CHERRY LIGHTS	12	2009-12-01	6.75	13085.0	United Kingdom	81.00	2011
3	489434	RECORD FRAME 7" SINGLE SIZE	48	2009-12-01	2.10	13085.0	United Kingdom	100.80	2011
4	489434	STRAWBERRY CERAMIC TRINKET BOX	24	2009-12-01	1.25	13085.0	United Kingdom	30.00	2011
...	...	...	...	...	...	...	...	...	
797878	581587	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09	2.10	12680.0	France	12.60	2011
797879	581587	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09	4.15	12680.0	France	16.60	2011
797880	581587	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09	4.15	12680.0	France	16.60	2011
797881	581587	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09	4.95	12680.0	France	14.85	2011
797882	581587	POSTAGE	1	2011-12-09	18.00	12680.0	France	18.00	2011

779423 rows × 9 columns

making RFM(Recency&Frequency and monetary)

set values from 1_5 for every customer in Recency&Frequency and Monetary

the closer date has 5 and the farest date has 1

the higher frequency has 5 and the lowest frequency has 1

the highest money entery has 5 and the lowest money entery has 1

In [15]:
```
# Create Recency , Frequency , Monetary for data by Customer ID
rfm = df.groupby('Customer_ID').agg({
    'InvoiceDate': lambda x: (today_date - x.max()).days,
    'Customer_ID': 'count',
    'Total_Price': 'sum'
})
rfm.rename(columns={'InvoiceDate': 'Recency',
                    'Customer_ID': 'Frequency',
                    'Total_Price': 'Monetary'}, inplace=True)

# Assign scores to each customer based on their Recency, Frequency, and Monetary
rfm['recency_score']=pd.qcut(rfm['Recency'],5,labels=[5,4,3,2,1])
rfm['frequency_score']=pd.qcut(rfm['Frequency'].rank(method='first'),5,labels=[1,2,3,4
rfm['monetary_score']=pd.qcut(rfm['Monetary'].rank(method='first'),5,labels=[1,2,3,4,5

# Combine scores to create an overall RFM score for each customer
rfm['RFM_SCORE'] = (rfm['recency_score'].astype(str) + rfm['frequency_score'].astype(s
# making segment map for customers to cluster them
seg_map={r'[1-3][1-5][1-3]':'At Risk Customers',r'[1-3][3-5][3-5]':'need_attention',r'
rfm['segment']=rfm['RFM_SCORE'].replace(seg_map,regex=True)

rfm
```

Out[15]:

Customer_ID	Recency	Frequency	Monetary	recency_score	frequency_score	monetary_score	RFM_S
12346.0	327	34	77556.46	2	2	5	
12347.0	4	222	4921.53	5	5	5	
12348.0	77	51	2019.40	3	3	4	
12349.0	20	175	4428.69	5	5	5	
12350.0	312	17	334.40	2	2	2	
...	...	...	...	...	...	...	
18283.0	5	938	2664.90	5	5	4	
18284.0	433	28	461.68	1	2	2	
18285.0	662	12	427.00	1	1	2	
18286.0	478	67	1296.43	1	3	4	
18287.0	44	155	4182.99	4	4	5	

5878 rows × 8 columns

we making RFM ANALYSIS to Identify the best customers and focus on offering them the best treatment and Improving and prioritize customer service team's activity and campaigns

In [16]:
```
#Count RFM values
rfm['segment'].value_counts()
```

Out[16]:
```
At Risk Customers      2604
Best Customers         1171
need_attention          813
new_customers           319
promissing              297
Big Spenders            274
loyal_customers         168
Low Spenders            127
Churning Customers       64
potential_loyalists      41
Name: segment, dtype: int64
```

In [17]:
```
# segment description
rfm[['segment','Recency','Frequency','Monetary']].groupby('segment').agg(['mean','cour
```

Out[17]:

	Recency			Frequency			Monetary		
	mean	count	max	mean	count	max	mean	count	max
segment									
At Risk Customers	359.47	2604	740	33.29	2604	329	450.27	2604	1220.90
Best Customers	17.78	1171	60	412.18	1171	12435	9618.47	1171	580987.04
Big Spenders	37.67	274	60	108.66	274	173	2740.50	274	16833.17
Churning Customers	391.97	64	693	18.67	64	35	4800.88	64	77556.46
Low Spenders	28.00	127	59	73.74	127	180	409.62	127	601.98
loyal_customers	39.53	168	60	73.46	168	171	901.50	168	1213.16
need_attention	209.49	813	682	171.89	813	2499	3541.32	813	65500.07
new_customers	38.77	319	184	14.94	319	37	1683.02	319	168472.50
potential_loyalists	11.34	41	21	59.07	41	74	1573.91	41	2812.77
promissing	22.91	297	60	35.23	297	73	663.04	297	1214.72

In [19]:

```
# Create a clustering chart of the RFM data.
import plotly.express as px
fig = px.scatter(rfm, x='Recency', y='Frequency', color='segment',
                 title='K-means Clustering of RFM Data')
# Set the axis labels and title
fig.update_layout(xaxis_title='Recency', yaxis_title='Frequency')
fig.update_layout(yaxis_range=[100, 2000])
fig.show()
```

THE FINALS RESULTS

we see that the most effective customers is (Best Customers&loyal customers) and there is some customers in ther way to be loyal customers and promissing and new customer so we need to care about them and some customers need more attention.

there is customers(Churning Customers) don't waste your effort and money on them.

In [ ]: