Task - 7 to 9 RFM Customer Segmentation

Customer segmentation is usally performed to understand customers behaviors. It leverages aqcuired customer data like the one we have in our case, **transactions data** in order to divide customers into groups.

Our goal in this Notebook is to cluster our customers to get insights in:

- Increasing revenue (Knowing customers who present most of our revenue)
- Increasing customer retention
- Discovering Trends and patterns
- Defining customers at risk

RFM Analysis answers these questions:

- Who are our best customers?
- Who has the potential to be converted in more profitable customers?
- Which customers we must retain?
- Which group of customers is most likely to respond to our current campaign?

Prepare the Data for RFM Analysis

In [1]:
```
# import all the required modules
import pandas as pd
import numpy as np

import time, warnings
import datetime as dt

#visualizations
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
%matplotlib inline
import seaborn as sns

warnings.filterwarnings("ignore")
```

In [2]:
```
#load the dataset
retail_df = pd.read_excel("../Online Retail.xlsx")
retail_df.head()
```

Out[2]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	Un Kingc
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	Un Kingc
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	Un Kingc

As customer clusters may vary by geography and as we idenitified from exploratory data analysis that most of our customer base and most of our customer historical data is from United Kingdom, we will restrict the data to only United Kingdom customers for this task.

In [3]:
```
retail_uk = retail_df[retail_df['Country']=='United Kingdom']
#check the shape
retail_uk.shape
```

Out[3]: (495478, 8)

Since we know there are a significant amount of cancelled orders, let'e remove them as well

In [4]:
```
#remove canceled orders
#retail_uk = retail_uk[retail_uk['Quantity']>0]
#retail_uk.shape
retail_uk = retail_uk[~retail_uk['InvoiceNo'].astype(str).str.startswith('C')]
retail_uk.shape
```

Out[4]: (487622, 8)

Let's remove anonymous/guest users as we won't be able to figure out which customer to target in those cases

In [5]:
```
#remove rows where customerID are NA
retail_uk.dropna(subset=['CustomerID'],how='all',inplace=True)
retail_uk.shape
```

Out[5]:
```
(354345, 8)
```

In [6]:
```
#restrict the data to one full year because it's better to use a metric per Months
retail_uk = retail_uk[retail_uk['InvoiceDate']>= "2010-12-09"]
retail_uk.shape
```

Out[6]:
```
(342478, 8)
```

In [7]:
```
print("Summary..")
#exploring the unique values of each attribute
print("Number of transactions: ", retail_uk['InvoiceNo'].nunique())
print("Number of products bought: ",retail_uk['StockCode'].nunique())
print("Number of customers:", retail_uk['CustomerID'].nunique() )
print("Percentage of anonymous/guest customers: ", round(retail_uk['CustomerID'].is
```

```
Summary..
Number of transactions:  16017
Number of products bought:  3611
Number of customers: 3863
Percentage of anonymous/guest customers:  0.0 %
```

Task 7: RFM Analysis Implementation

RFM (**Recency, Frequency, Monetary**) analysis is a customer segmentation technique that uses past purchase behavior to divide customers into groups.
RFM helps divide customers into various categories or clusters to identify customers who are more likely to respond to promotions and also for future personalization services.

- RECENCY (R): Days since last purchase
- FREQUENCY (F): Total number of purchases
- MONETARY VALUE (M): Total money this customer spent.

We will create those 3 customer attributes for each customer.

Recency

To calculate recency, we need to choose a date point from which we evaluate **how many days ago was the customer's last purchase**.

In [8]:
```
#last date available in our dataset
retail_uk['InvoiceDate'].max()
```

Out[8]:
```
Timestamp('2011-12-09 12:49:00')
```

The last date we have is 2011-12-09 so we will use it as reference.

In [9]:
```
now = dt.date(2011,12,9)
print(now)
```

```
2011-12-09
```

In [10]:
```
#create a new column called date which contains the date of invoice only
retail_uk['date'] = retail_uk['InvoiceDate'].dt.date
```

In [11]:
```
retail_uk.head()
```

Out[11]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
19632	537879	22114	HOT WATER BOTTLE TEA AND SYMPATHY	12	2010-12-09 08:34:00	3.95	14243.0
19633	537879	22835	HOT WATER BOTTLE I AM SO POORLY	8	2010-12-09 08:34:00	4.65	14243.0
19634	537879	85150	LADIES & GENTLEMEN METAL SIGN	6	2010-12-09 08:34:00	2.55	14243.0
19635	537879	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	4	2010-12-09 08:34:00	7.95	14243.0
19636	537879	21524	DOORMAT SPOTTY HOME SWEET HOME	2	2010-12-09 08:34:00	7.95	14243.0

In [12]:
```
#group by customers and check last date of purshace
recency_df = retail_uk.groupby(by='CustomerID', as_index=False)['date'].max()
recency_df.columns = ['CustomerID','LastPurshaceDate']
recency_df.head()
```

Out[12]:

	CustomerID	LastPurshaceDate
0	12346.0	2011-01-18
1	12747.0	2011-12-07
2	12748.0	2011-12-09
3	12749.0	2011-12-06
4	12820.0	2011-12-06

In [13]:
```
#calculate recency
recency_df['Recency'] = recency_df['LastPurshaceDate'].apply(lambda x: (now - x).da
```

In [14]:
```
recency_df.head()
```

Out[14]:

	CustomerID	LastPurshaceDate	Recency
0	12346.0	2011-01-18	325
1	12747.0	2011-12-07	2
2	12748.0	2011-12-09	0
3	12749.0	2011-12-06	3
4	12820.0	2011-12-06	3

In [15]:
```
#drop LastPurchaseDate as we don't need it anymore
recency_df.drop('LastPurshaceDate',axis=1,inplace=True)
```

Now we have the recency attribute created. e.g: Customer with ID = 12346 did his/her last purshace 325 days ago.

Frequency

Frequency helps us to know **how many times a customer purchased from us**. To do that we need to check how many invoices are registered by the same customer.

In [16]:
```
# drop duplicates
retail_uk_copy = retail_uk
retail_uk_copy.drop_duplicates(subset=['InvoiceNo', 'CustomerID'], keep="first", in
#calculate frequency of purchases
frequency_df = retail_uk_copy.groupby(by=['CustomerID'], as_index=False)['InvoiceNo
frequency_df.columns = ['CustomerID','Frequency']
frequency_df.head()
```

Out[16]:

	CustomerID	Frequency
0	12346.0	1
1	12747.0	10
2	12748.0	196
3	12749.0	5
4	12820.0	4

Monetary

Monetary attribute answers the question: **How much money did the customer spent over time?**

To do that, first, we will create a new column total cost to have the total price per invoice.

In [17]:
```
#create column total cost
retail_uk['TotalCost'] = retail_uk['Quantity'] * retail_uk['UnitPrice']
```

In [18]:
```
monetary_df = retail_uk.groupby(by='CustomerID',as_index=False).agg({'TotalCost': '
monetary_df.columns = ['CustomerID','Monetary']
monetary_df.head()
```

Out[18]:

	CustomerID	Monetary
0	12346.0	77183.60
1	12747.0	658.89
2	12748.0	3739.23
3	12749.0	98.35
4	12820.0	58.20

Create RFM Table

In [19]:
```
#merge recency dataframe with frequency dataframe
temp_df = recency_df.merge(frequency_df,on='CustomerID')
temp_df.head()
```

Out[19]:

	CustomerID	Recency	Frequency
0	12346.0	325	1
1	12747.0	2	10
2	12748.0	0	196
3	12749.0	3	5
4	12820.0	3	4

In [20]:
```
#merge with monetary dataframe to get a table with the 3 columns
rfm_df = temp_df.merge(monetary_df,on='CustomerID')
#use CustomerID as index
rfm_df.set_index('CustomerID',inplace=True)
#check the head
rfm_df.head()
```

Out[20]:

CustomerID	Recency	Frequency	Monetary
12346.0	325	1	77183.60
12747.0	2	10	658.89
12748.0	0	196	3739.23
12749.0	3	5	98.35
12820.0	3	4	58.20

Customer with ID = 12346 has recency: 325 days, frequency:1, and monetary: 77183,60 £.

RFM Table Correctness verification

In [21]:
```
retail_uk[retail_uk['CustomerID']==12346.0]
```

Out[21]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
61619	541431	23166	MEDIUM CERAMIC TOP STORAGE JAR	74215	2011-01-18 10:01:00	1.04	12346.0

In [22]:
```
(now - dt.date(2011,1,18)).days == 325
```

Out[22]:
```
True
```

As we can see our RFM table is correct. The first customer bought only once, and only one product with huge amount.

Task 8: Customer Segmentation Insights

Before moving to customer segments, Let's see the application of Pareto Principle – commonly referred to as the 80-20 rule on our dataset by applying it to our RFM variables.

Pareto's rule says **80% of the results come from 20% of the causes.**

Similarly, **20% customers contribute to 80% of your total revenue**. Let's verify that because that will help us know which customers to focus on when marketing new products.

Applying 80-20 rule

In [23]:
```
#get the 80% of the revenue
pareto_cutoff = rfm_df['Monetary'].sum() * 0.8
print("The 80% of total revenue is: ",round(pareto_cutoff,2))
```

```
The 80% of total revenue is:  890679.54
```

In [24]:
```
customers_rank = rfm_df
# Create a new column that is the rank of the value of coverage in ascending order
customers_rank['Rank'] = customers_rank['Monetary'].rank(ascending=0)
#customers_rank.drop('RevenueRank',axis=1,inplace=True)
customers_rank.head()
```

Out[24]:

CustomerID	Recency	Frequency	Monetary	Rank
12346.0	325	1	77183.60	2.0
12747.0	2	10	658.89	183.0
12748.0	0	196	3739.23	34.0
12749.0	3	5	98.35	1140.0
12820.0	3	4	58.20	1665.5

Top Customers

In [25]:
```
customers_rank.sort_values('Rank',ascending=True)
```

Out[25]:

CustomerID	Recency	Frequency	Monetary	Rank
16446.0	0	2	168471.25	1.0
12346.0	325	1	77183.60	2.0
15098.0	182	3	39916.50	3.0
18102.0	0	57	39646.45	4.0
17949.0	1	44	28685.69	5.0
...	...	...	...	...
15645.0	18	1	0.42	3859.5
15503.0	362	1	0.42	3859.5
17914.0	3	1	0.39	3861.5
13271.0	37	1	0.39	3861.5
13256.0	14	1	0.00	3863.0

3863 rows × 4 columns

In [26]:
```
#get top 20% of the customers
top_20_cutoff = 3863 *20 /100
top_20_cutoff
```

Out[26]: 772.6

In [27]:
```
#sum the monetary values over the customer with rank <=773
revenueByTop20 = customers_rank[customers_rank['Rank'] <= 772]['Monetary'].sum()
round(revenueByTop20,2)
```

Out[27]: np.float64(976683.35)

In [28]:
```
print("Is 80% of total revenue achieved by 20% of the top customers: ", "Yes" if re
```

```
Is 80% of total revenue achieved by 20% of the top customers:  Yes
```

In our case, the 80% of total revenue is achieved by the 20% of TOP customers. It would be interesting to study this group of customers because they are those who make our most revenue.

Applying RFM score formula

The simplest way to create customers segments from RFM Model is to use **Quartiles**. We assign a score from 1 to 4 to Recency, Frequency and Monetary. Four is the best/highest value, and one is the lowest/worst value. A final RFM score is calculated simply by combining individual RFM score numbers.

Note: Quintiles (score from 1-5) offer better granularity, in case the business needs that but it will be more challenging to create segments since we will have 555 possible combinations. So, we will use quartiles.

RFM Quartiles

In [29]:
```
quantiles = rfm_df.quantile(q=[0.25,0.5,0.75])
quantiles
```

Out[29]:

	Recency	Frequency	Monetary	Rank
0.25	17.0	1.0	17.4	966.5
0.50	49.0	2.0	45.0	1930.5
0.75	134.0	5.0	121.6	2898.0

In [30]:
```
quantiles.to_dict()
```

Out[30]:
```
{'Recency': {0.25: 17.0, 0.5: 49.0, 0.75: 134.0},
 'Frequency': {0.25: 1.0, 0.5: 2.0, 0.75: 5.0},
 'Monetary': {0.25: 17.4, 0.5: 45.0, 0.75: 121.60000000000001},
 'Rank': {0.25: 966.5, 0.5: 1930.5, 0.75: 2898.0}}
```

Creation of RFM segmentation table

We will create two segmentation classes since, high recency is bad, while high frequency and monetary value is good.

In [31]:
```
# Arguments (x = value, p = recency, monetary_value, frequency, d = quartiles dict)
def RScore(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
# Arguments (x = value, p = recency, monetary_value, frequency, k = quartiles dict)
def FMScore(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
```

In [32]:
```
#create rfm segmentation table
rfm_segmentation = rfm_df
```

```
rfm_segmentation['R_Quartile'] = rfm_segmentation['Recency'].apply(RScore, args=('R
rfm_segmentation['F_Quartile'] = rfm_segmentation['Frequency'].apply(FMScore, args=
rfm_segmentation['M_Quartile'] = rfm_segmentation['Monetary'].apply(FMScore, args=(
```

In [33]:
```
rfm_segmentation.head()
```

Out[33]:

CustomerID	Recency	Frequency	Monetary	Rank	R_Quartile	F_Quartile	M_Quartile
12346.0	325	1	77183.60	2.0	1	1	4
12747.0	2	10	658.89	183.0	4	4	4
12748.0	0	196	3739.23	34.0	4	4	4
12749.0	3	5	98.35	1140.0	4	3	3
12820.0	3	4	58.20	1665.5	4	3	3

Now that we have the score of each customer, we can represent our customer segmentation. First, we need to combine the scores (R_Quartile, F_Quartile,M_Quartile) together.

In [34]:
```
rfm_segmentation['RFMScore'] = rfm_segmentation.R_Quartile.map(str) \
                            + rfm_segmentation.F_Quartile.map(str) \
                            + rfm_segmentation.M_Quartile.map(str)
rfm_segmentation.head()
```

Out[34]:

CustomerID	Recency	Frequency	Monetary	Rank	R_Quartile	F_Quartile	M_Quartile	RI
12346.0	325	1	77183.60	2.0	1	1	4	
12747.0	2	10	658.89	183.0	4	4	4	
12748.0	0	196	3739.23	34.0	4	4	4	
12749.0	3	5	98.35	1140.0	4	3	3	
12820.0	3	4	58.20	1665.5	4	3	3	

Best Recency score = 4: most recently purchase. Best Frequency score = 4: most quantity purchase. Best Monetary score = 4: spent the most.

Let's see who are our **Champions** (best customers).

In [35]:
```
rfm_segmentation[rfm_segmentation['RFMScore']=='444'].sort_values('Monetary', ascen
```

Out[35]:

CustomerID	Recency	Frequency	Monetary	Rank	R_Quartile	F_Quartile	M_Quartile	RFI
18102.0	0	57	39646.45	4.0	4	4	4	
17949.0	1	44	28685.69	5.0	4	4	4	
17450.0	8	44	25953.51	6.0	4	4	4	
16013.0	3	45	17163.28	8.0	4	4	4	
16333.0	7	22	14418.96	9.0	4	4	4	
15769.0	7	25	11660.84	11.0	4	4	4	
12901.0	8	28	9230.45	12.0	4	4	4	
13798.0	1	56	7786.26	13.0	4	4	4	
16684.0	4	28	7700.08	14.0	4	4	4	
17857.0	4	23	7655.18	15.0	4	4	4	

We can find here a suggestion of key segments and then we can decide which segment to consider for further study.

Note: the suggested link use the opposite valuation: 1 as highest/best score and 4 is the lowest.

How many customers do we have in each segment?

In [36]:
```
print("Best Customers: ",len(rfm_segmentation[rfm_segmentation['RFMScore']=='444']))
print('Loyal Customers: ',len(rfm_segmentation[rfm_segmentation['F_Quartile']==4]))
print("Big Spenders: ",len(rfm_segmentation[rfm_segmentation['M_Quartile']==4]))
print('Almost Lost: ', len(rfm_segmentation[rfm_segmentation['RFMScore']=='244']))
print('Lost Customers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='144']))
print('Lost Cheap Customers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='
```

```
Best Customers:  356
Loyal Customers:  752
Big Spenders:  966
Almost Lost:  64
Lost Customers:  9
Lost Cheap Customers:  353
```

Now that we knew our customers segments we can choose how to target or deal with each segment.

For example:

Best Customers - Champions: Reward them. They can be early adopters to new products. Suggest them "Refer a friend".

At Risk: Send them personalized emails to encourage them to shop.

Task 9: Visualization and Reporting

In [ ]: