Retail

It is a business critical requirement to understand the value derived from a customer. RFM is a method used for analyzing customer value.

Perform customer segmentation using RFM analysis. The resulting segments can be ordered from most valuable (highest recency, frequency, and value) to least valuable (lowest recency, frequency, and value). Identifying the most valuable RFM segments can capitalize on chance relationships in the data used for this analysis

In [1]:
```python
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
```

In [2]:
```python
df = pd.read_excel('T:\Masters In Data Science\Capstone Project\Project 3\\Online Retail.xlsx')
```

In [3]:
```python
df.head()
```

Out[3]:

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

In [4]:
```python
df.shape
```

Out[4]:
```
(541909, 8)
```

Descriptive Analysis

In [5]:
```python
df.describe()
```

Out[5]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

Unit Price : Average Unit price sold id 4.6 Also we have to note that Min unit price is negative which means store had to return some amount for returned products during the period of our analysis

Quantity : Average quantity bought is 9.55 Also some products were returned to the store by customers during the period of our analysis

In [6]:
```python
df.describe(include='O')
```

Out[6]:

	InvoiceNo	StockCode	Description	Country
count	541909	541909	540455	541909
unique	25900	4070	4223	38
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	United Kingdom
freq	1114	2313	2369	495478

Invoice No : The total number of invoices created is 25900

Country : Store is functional in 38 countries

Stock Code : There are total 4070 types of items in stock

In [7]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype         
---  ------       --------------   -----         
 0   InvoiceNo    541909 non-null  object        
 1   StockCode    541909 non-null  object        
 2   Description  540455 non-null  object        
 3   Quantity     541909 non-null  int64         
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64       
 6   CustomerID   406829 non-null  float64       
 7   Country      541909 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

Dropping the duplicates

In [8]:
```
df = df.drop_duplicates()
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 536641 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype         
---  ------       --------------   -----         
 0   InvoiceNo    536641 non-null  object        
 1   StockCode    536641 non-null  object        
 2   Description  535187 non-null  object        
 3   Quantity     536641 non-null  int64         
 4   InvoiceDate  536641 non-null  datetime64[ns]
 5   UnitPrice    536641 non-null  float64       
 6   CustomerID   401604 non-null  float64       
 7   Country      536641 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 36.8+ MB
```

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135037
Country             0
dtype: int64
```

In [11]:
```
round((df.isnull().sum()/len(df))*100,2)
```

Out[11]:
```
InvoiceNo       0.00
StockCode       0.00
Description     0.27
Quantity        0.00
InvoiceDate     0.00
UnitPrice       0.00
CustomerID     25.16
Country         0.00
dtype: float64
```

There are 25.16% null values in the column Customer Id we will see if we can find any customer Ids in the invoices otherwise we will drop them from the dataset

In [12]:
```
null_cust_id_inv = set(df[df['CustomerID'].isnull()]['InvoiceNo'])
```

In [13]:
```
df[df['InvoiceNo'].isin(null_cust_id_inv) & (~df['CustomerID'].isnull())]
```

Out[13]:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country

We could not find any customer Ids using invoice numbers so we will drop the null rows from the dataset

In [14]:
```
df = df.dropna()
df.shape
```

Out[14]:
```
(401604, 8)
```

Cohort Analysis

a. Create month cohorts and analyse active customers for each cohort

In [15]:
```
from datetime import timedelta
df['Month_of_year'] = df['InvoiceDate'].dt.to_period('M')
df['Month_of_year'].nunique()
```

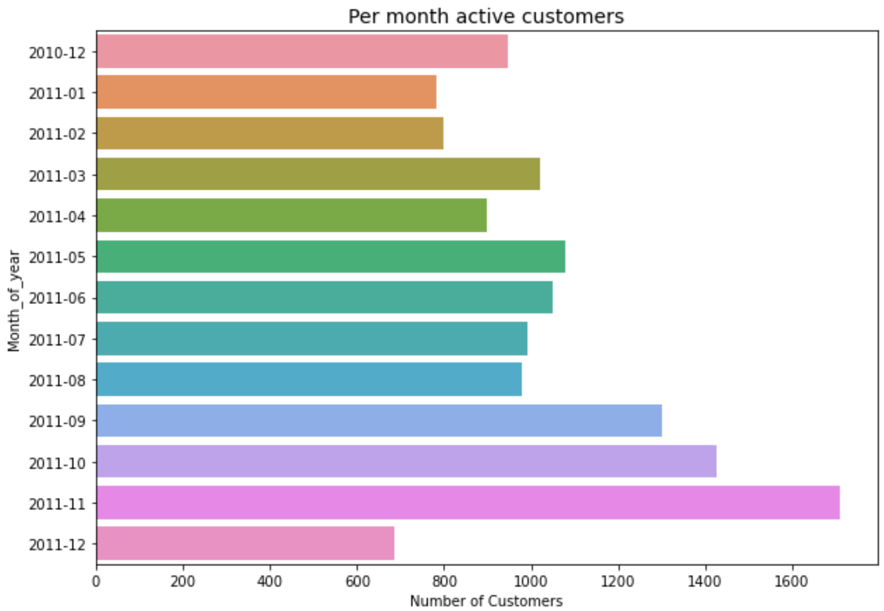
Out[15]: 13

In [16]:
```
Month_Cohort = df.groupby('Month_of_year')['CustomerID'].nunique()
Month_Cohort
```

Out[16]:
```
Month_of_year
2010-12     948
2011-01     783
2011-02     798
2011-03    1020
2011-04     899
2011-05    1079
2011-06    1051
2011-07     993
2011-08     980
2011-09    1302
2011-10    1425
2011-11    1711
2011-12     686
Freq: M, Name: CustomerID, dtype: int64
```

In [17]:
```
import seaborn as sns
```

In [18]:
```
plt.figure(figsize=(10,7))
sns.barplot(x = Month_Cohort.values, y = Month_Cohort.index)
plt.xlabel('Number of Customers')
plt.title('Per month active customers',fontsize=14)
plt.show()
```



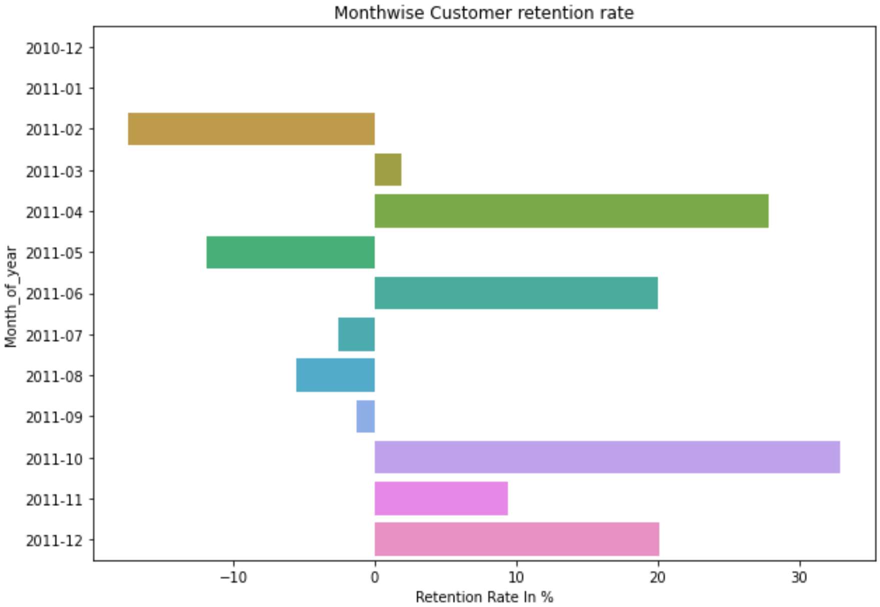
In [19]:
```
Month_Cohort = Month_Cohort.shift(1)
```

b) Analyse the retention rate of customers.

In [20]:
```
Retention_rate = round(Month_Cohort.pct_change(periods=1)*100,2)
Retention_rate
```

Out[20]:
```
Month_of_year
2010-12       NaN
2011-01       NaN
2011-02    -17.41
2011-03      1.92
2011-04     27.82
2011-05    -11.86
2011-06     20.02
2011-07     -2.59
2011-08     -5.52
2011-09     -1.31
2011-10     32.86
2011-11      9.45
2011-12     20.07
Freq: M, Name: CustomerID, dtype: float64
```

In [21]:
```
plt.figure(figsize=(10,7))
sns.barplot(x=Retention_rate.values,y=Retention_rate.index)
plt.xlabel('Retention Rate In %')
plt.title('Monthwise Customer retention rate')
plt.show()
```



Monetary analysis

In [22]:
```
df['amount'] = df['Quantity'] * df['UnitPrice']
```

In [23]:
```
monetary = df.groupby('CustomerID')['amount'].sum().reset_index()
```

In [24]:
```
monetary
```

Out[24]:

	CustomerID	amount
0	12346.0	0.00
1	12347.0	4310.00
2	12348.0	1797.24
3	12349.0	1757.55
4	12350.0	334.40
...	...	...
4367	18280.0	180.60
4368	18281.0	80.82
4369	18282.0	176.60
4370	18283.0	2045.53
4371	18287.0	1837.28

4372 rows × 2 columns

Frequency analysis

In [25]:
```
df.head()
```

Out[25]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Month_of_year	amount
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010-12	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010-12	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12	20.34

In [26]:
```
Frequency = df.groupby('CustomerID').nunique()['InvoiceNo'].reset_index()
Frequency
```

Out[26]:

	CustomerID	InvoiceNo
0	12346.0	2
1	12347.0	7
2	12348.0	4
3	12349.0	1
4	12350.0	1
...	...	...
4367	18280.0	1
4368	18281.0	1
4369	18282.0	3
4370	18283.0	16
4371	18287.0	3

4372 rows × 2 columns

Recency analysis

In [27]:
```
ref_day = max(df['InvoiceDate']) + timedelta(days=1)
ref_day
```

Out[27]:
```
Timestamp('2011-12-10 12:50:00')
```

In [28]:
```
df['days_since_last_order'] = (ref_day - df['InvoiceDate'])
```

In [29]:
```
Recency = df.groupby('CustomerID').nunique()['days_since_last_order'].reset_index()
Recency
```

Out[29]:

	CustomerID	days_since_last_order
0	12346.0	2
1	12347.0	7
2	12348.0	4
3	12349.0	1
4	12350.0	1
...	...	...
4367	18280.0	1
4368	18281.0	1
4369	18282.0	3
4370	18283.0	16
4371	18287.0	3

4372 rows × 2 columns

In [30]:
```
RFM = pd.merge(Recency,Frequency,on='CustomerID',how='left')
RFM = pd.merge(RFM,monetary,on='CustomerID',how='left')
```

In [31]:
```
RFM
```

Out[31]:

	CustomerID	days_since_last_order	InvoiceNo	amount
0	12346.0	2	2	0.00
1	12347.0	7	7	4310.00
2	12348.0	4	4	1797.24
3	12349.0	1	1	1757.55
4	12350.0	1	1	334.40
...	...	...	...	...
4367	18280.0	1	1	180.60
4368	18281.0	1	1	80.82
4369	18282.0	3	3	176.60
4370	18283.0	16	16	2045.53
4371	18287.0	3	3	1837.28

4372 rows × 4 columns

In [32]:
```
RFM.columns = ['Customer_id','Recency','Frequency','Monetary']
```

In [33]:
```
RFM
```

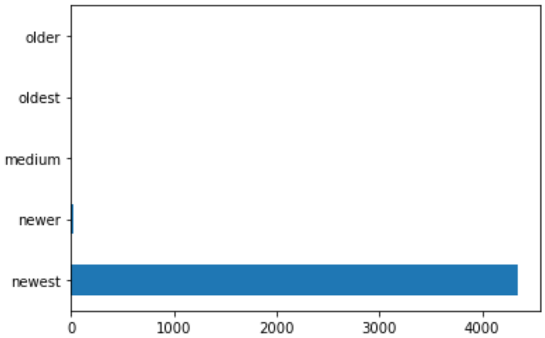
Out[33]:

	Customer_id	Recency	Frequency	Monetary
0	12346.0	2	2	0.00
1	12347.0	7	7	4310.00
2	12348.0	4	4	1797.24
3	12349.0	1	1	1757.55
4	12350.0	1	1	334.40
...	...	...	...	...
4367	18280.0	1	1	180.60
4368	18281.0	1	1	80.82
4369	18282.0	3	3	176.60
4370	18283.0	16	16	2045.53
4371	18287.0	3	3	1837.28

4372 rows × 4 columns

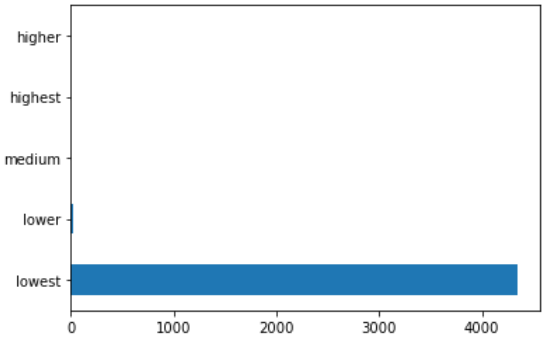
In [34]:
```
RFM['recency_labels'] = pd.cut(RFM['Recency'],bins=5,labels=['newest','newer','medium','older','oldest'])
RFM['recency_labels'].value_counts().plot(kind='barh')
RFM['recency_labels'].value_counts()
```

Out[34]:
```
newest     4348
newer        18
medium        3
oldest        2
older         1
Name: recency_labels, dtype: int64
```



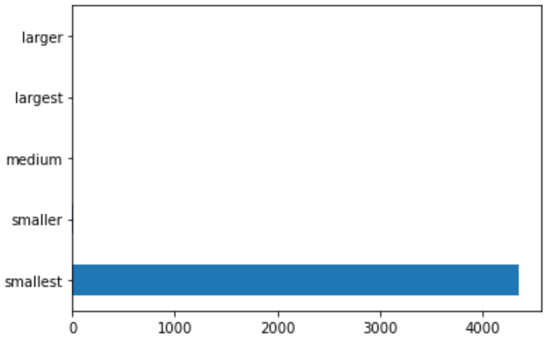
In [35]:
```
RFM['frequency_labels'] = pd.cut(RFM['Frequency'],bins=5,labels=['lowest','lower','medium','higher','highest'])
RFM['frequency_labels'].value_counts().plot(kind='barh')
RFM['frequency_labels'].value_counts()
```

Out[35]:
```
lowest      4348
lower         18
medium         3
highest        2
higher         1
Name: frequency_labels, dtype: int64
```



In [36]:
```
RFM['monetary_labels'] = pd.cut(RFM['Monetary'],bins=5,labels=['smallest','smaller','medium','larger','largest'])
RFM['monetary_labels'].value_counts().plot(kind='barh')
RFM['monetary_labels'].value_counts()
```

Out[36]:
```
smallest    4357
smaller        9
medium         3
largest        2
larger         1
Name: monetary_labels, dtype: int64
```



RFM Segment

In [37]:
```python
RFM['Segment'] = RFM[['recency_labels','frequency_labels','monetary_labels']].agg('-'.join,axis=1)
RFM.head()
```

Out[37]:

	Customer_id	Recency	Frequency	Monetary	recency_labels	frequency_labels	monetary_labels	Segment
0	12346.0	2	2	0.00	newest	lowest	smallest	newest-lowest-smallest
1	12347.0	7	7	4310.00	newest	lowest	smallest	newest-lowest-smallest
2	12348.0	4	4	1797.24	newest	lowest	smallest	newest-lowest-smallest
3	12349.0	1	1	1757.55	newest	lowest	smallest	newest-lowest-smallest
4	12350.0	1	1	334.40	newest	lowest	smallest	newest-lowest-smallest

RFM score

In [38]:
```python
recency_dict = {'newest':1,'newer':2,'medium':3,'older':4,'oldest':5}
frequency_dict = {'lowest':1,'lower':2,'medium':3,'higher':4,'highest':5}
monetary_dict = {'smallest':1,'smaller':2,'medium':3,'larger':4,'largest':5}
```

In [39]:
```python
RFM['rfm_score'] = RFM['recency_labels'].map(recency_dict).astype(int) + RFM['frequency_labels'].map(frequency_dict).astype(in
```

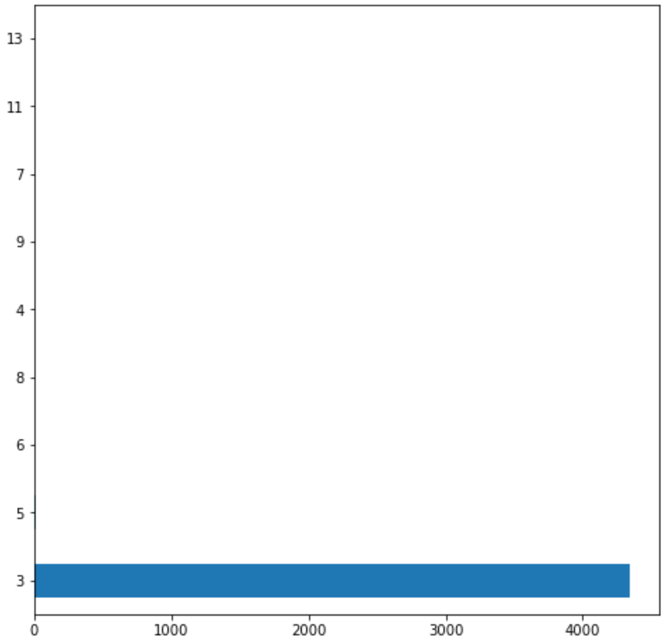
In [40]:
```python
RFM.head()
```

Out[40]:

	Customer_id	Recency	Frequency	Monetary	recency_labels	frequency_labels	monetary_labels	Segment	rfm_score
0	12346.0	2	2	0.00	newest	lowest	smallest	newest-lowest-smallest	3
1	12347.0	7	7	4310.00	newest	lowest	smallest	newest-lowest-smallest	3
2	12348.0	4	4	1797.24	newest	lowest	smallest	newest-lowest-smallest	3
3	12349.0	1	1	1757.55	newest	lowest	smallest	newest-lowest-smallest	3
4	12350.0	1	1	334.40	newest	lowest	smallest	newest-lowest-smallest	3

In [41]:
```python
plt.figure(figsize=(8,8))
RFM['rfm_score'].value_counts().plot(kind='barh')
RFM['rfm_score'].value_counts()
```

Out[41]:
```
3     4344
5       11
6        4
8        3
4        3
9        3
7        2
11       1
13       1
Name: rfm_score, dtype: int64
```



This shows that most of the customers are new in the store also they visit the store less frequently and they spend less amount of money in the store

Data Preprocessing

In [42]:
```
RFM.shape
```

Out[42]:
```
(4372, 9)
```

In [43]:
```
RFM.head()
```

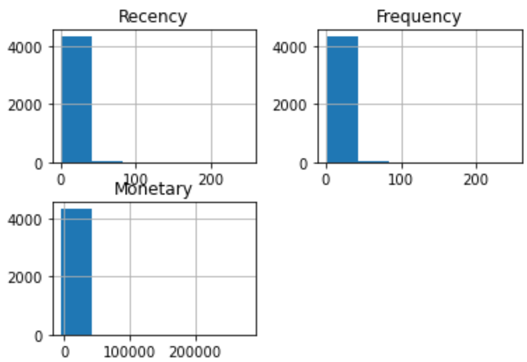
Out[43]:

	Customer_id	Recency	Frequency	Monetary	recency_labels	frequency_labels	monetary_labels	Segment	rfm_score
0	12346.0	2	2	0.00	newest	lowest	smallest	newest-lowest-smallest	3
1	12347.0	7	7	4310.00	newest	lowest	smallest	newest-lowest-smallest	3
2	12348.0	4	4	1797.24	newest	lowest	smallest	newest-lowest-smallest	3
3	12349.0	1	1	1757.55	newest	lowest	smallest	newest-lowest-smallest	3
4	12350.0	1	1	334.40	newest	lowest	smallest	newest-lowest-smallest	3

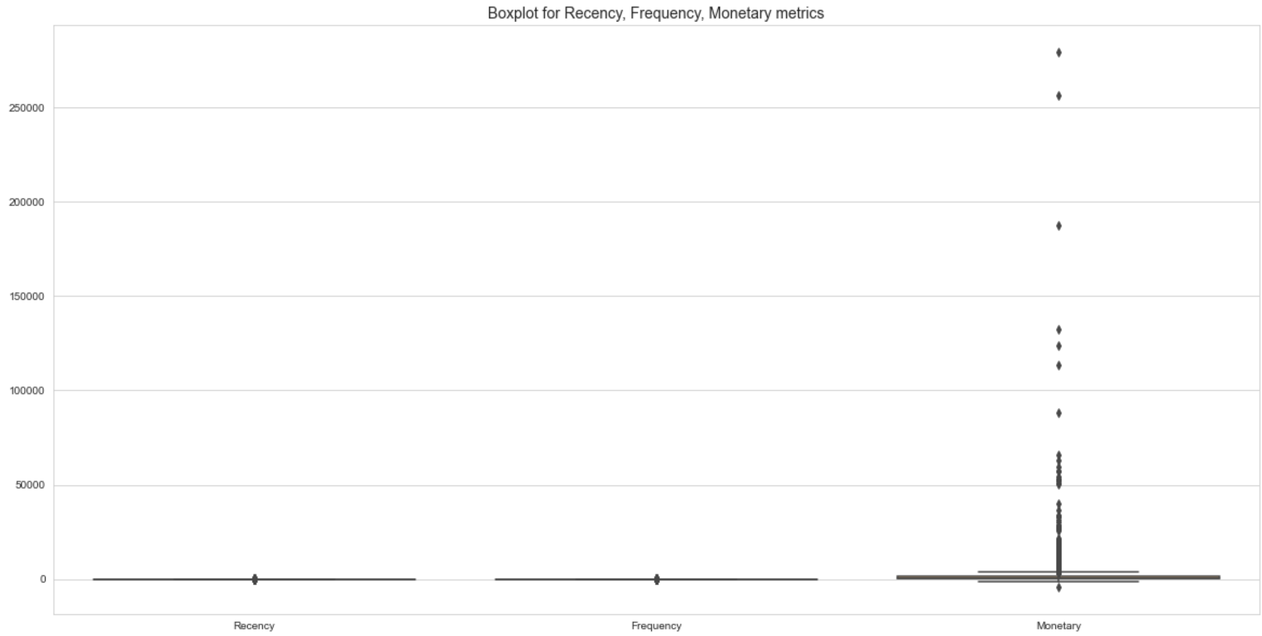
Let us check the distribution of the data

In [44]:
```
plt.figure(figsize=(20,10))
RFM[['Recency','Frequency','Monetary']].hist(bins=6)
plt.title('Histplot of Recency, Frequency, Monetary metrics',fontsize=14)
plt.show()
```

```
<Figure size 1440x720 with 0 Axes>
```



In [45]:
```
plt.figure(figsize = (20,10))
sns.set_style('whitegrid')
sns.boxplot(data=RFM[['Recency','Frequency','Monetary']],palette='rainbow')
plt.title('Boxplot for Recency, Frequency, Monetary metrics',fontsize=14)
plt.show()
```



Looks like monetary metrics have a lot of outliers

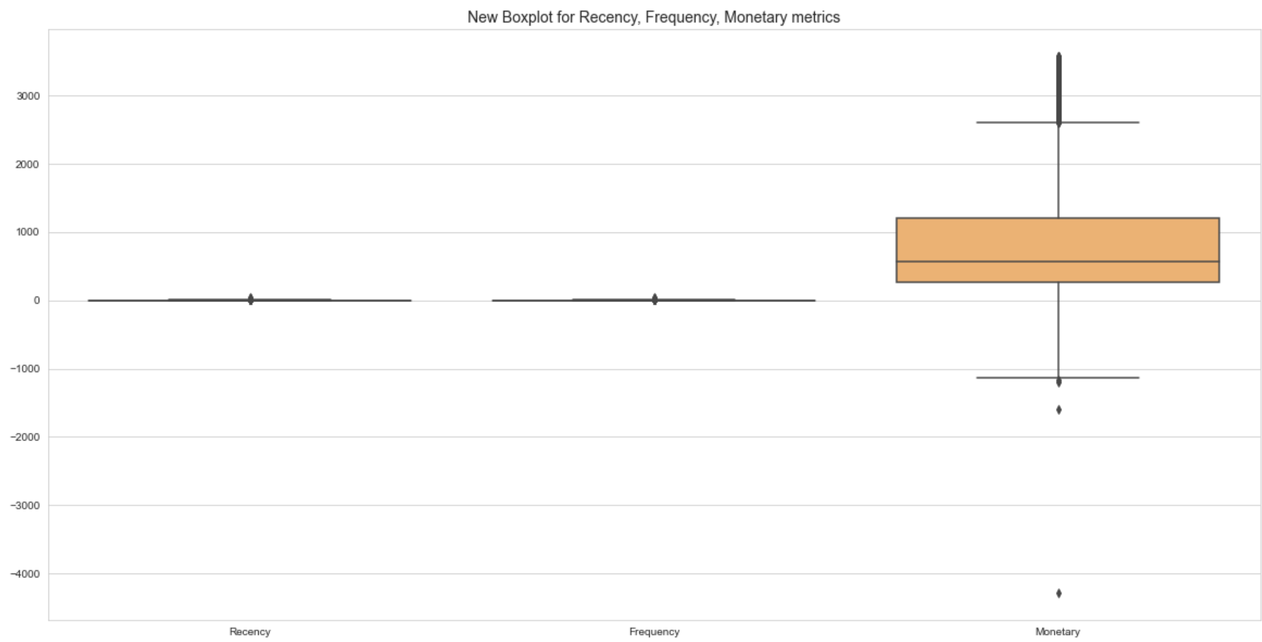
In [46]:
```
Q1 = RFM['Monetary'].quantile(0.25)
Q3 = RFM['Monetary'].quantile(0.75)
print('Q1=',Q1,'Q3=',Q3)

IQR = Q3-Q1
print(IQR)

Upper_whisker = Q3+1.5*IQR
Lower_whisker = Q1-1.5*IQR

print('Upper Whisker=',Upper_whisker)
print('Lower Whisker=',Lower_whisker)
```

```
Q1= 291.795 Q3= 1608.335
1316.54
Upper Whisker= 3583.145
Lower Whisker= -1683.0149999999999
```

In [47]:
```
RFM_new = RFM[RFM['Monetary']<Upper_whisker]
RFM_new.shape
```

Out[47]:
```
(3952, 9)
```

In [48]:
```
plt.figure(figsize = (20,10))
sns.set_style('whitegrid')
sns.boxplot(data=RFM_new[['Recency','Frequency','Monetary']],palette='rainbow')
plt.title('New Boxplot for Recency, Frequency, Monetary metrics',fontsize=14)
plt.show()
```



Now let us scale the data for better use

In [49]:
```
RFM_new_for_scaling = RFM_new[['Recency','Frequency','Monetary']]
```

In [50]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
RFM_scaled = scaler.fit_transform(RFM_new_for_scaling)
```

In [51]:
```
RFM_scaled = pd.DataFrame(RFM_scaled)
RFM_scaled
```

Out[51]:

	0	1	2
0	-0.448910	-0.449696	-1.041655
1	0.192483	0.187178	1.157895
2	-0.769606	-0.768133	1.109321
3	-0.769606	-0.768133	-0.632400
4	2.437357	2.416238	0.849693
...	...	...	...
3947	-0.769606	-0.768133	-0.820628
3948	-0.769606	-0.768133	-0.942744
3949	-0.128214	-0.131259	-0.825523
3950	4.040839	4.008423	1.461765
3951	-0.128214	-0.131259	1.206898

3952 rows × 3 columns

In [52]:
```
RFM_scaled.columns = ['Recency','Frequency','Monetary']
RFM_scaled
```

Out[52]:

	Recency	Frequency	Monetary
0	-0.448910	-0.449696	-1.041655
1	0.192483	0.187178	1.157895
2	-0.769606	-0.768133	1.109321
3	-0.769606	-0.768133	-0.632400
4	2.437357	2.416238	0.849693
...	...	...	...
3947	-0.769606	-0.768133	-0.820628
3948	-0.769606	-0.768133	-0.942744
3949	-0.128214	-0.131259	-0.825523
3950	4.040839	4.008423	1.461765
3951	-0.128214	-0.131259	1.206898

3952 rows × 3 columns

Build K means model

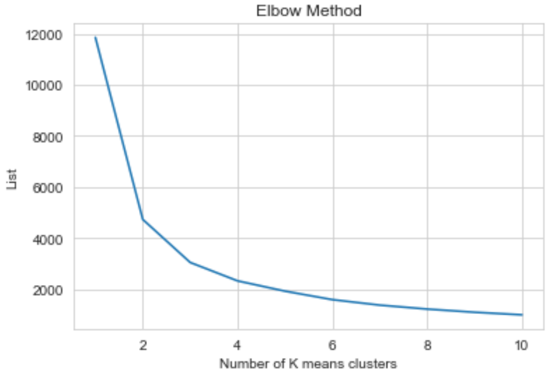
In [53]:
```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3,max_iter=50)
kmeans.fit(RFM_scaled)
```

Out[53]:
```
KMeans(max_iter=50, n_clusters=3)
```

In [54]:
```
kmeans.labels_
```

Out[54]:
```
array([0, 2, 0, ..., 0, 1, 2])
```

In [55]:
```
blank_list = []

for i in range(1,11):
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(RFM_scaled)
    blank_list.append(kmeans.inertia_)

plt.plot(range(1,11),blank_list)
plt.title('Elbow Method')
plt.xlabel('Number of K means clusters')
plt.ylabel('List')
plt.show()
```



From elbow plot we can see after 4 clusters line starts to stabalize so we can select optimum number of clusters as 4

In [56]:
```
df_inertia = pd.DataFrame(list(zip(range(1,11),blank_list)))
df_inertia.columns = ['clusters','inertia']
df_inertia
```

Out[56]:

	clusters	inertia
0	1	11856.000000
1	2	4734.056191
2	3	3050.533413
3	4	2329.370267
4	5	1927.391812
5	6	1594.692018
6	7	1380.349119
7	8	1225.933463
8	9	1101.309180
9	10	1000.936887

Building Kmeans model again for 4 numbers of clusters

In [57]:
```
kmeans = KMeans(n_clusters=4,max_iter=50)
```

In [58]:
```
kmeans.fit(RFM_scaled)
```

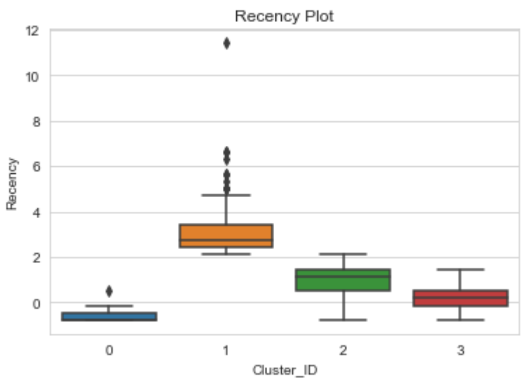
Out[58]: KMeans(max_iter=50, n_clusters=4)

In [59]:
```
RFM_scaled['Cluster_ID'] = kmeans.labels_
RFM_scaled.head()
```

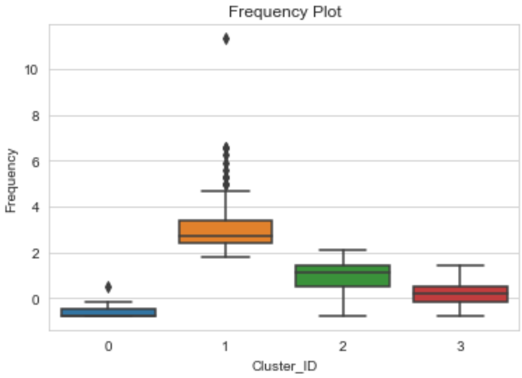
Out[59]:

	Recency	Frequency	Monetary	Cluster_ID
0	-0.448910	-0.449696	-1.041655	0
1	0.192483	0.187178	1.157895	3
2	-0.769606	-0.768133	1.109321	3
3	-0.769606	-0.768133	-0.632400	0
4	2.437357	2.416238	0.849693	1

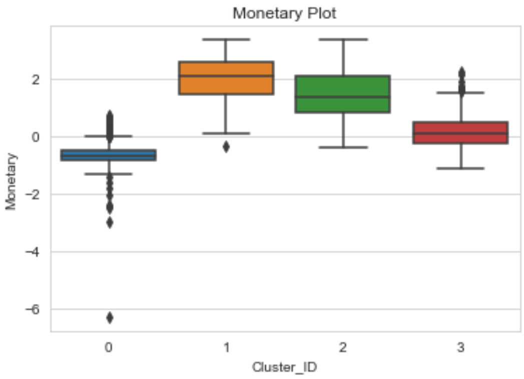
In [60]:
```
sns.boxplot(x = 'Cluster_ID',y = 'Recency',data=RFM_scaled)
plt.title('Recency Plot')
plt.show()
```



In [61]:
```
sns.boxplot(x = 'Cluster_ID',y = 'Frequency',data=RFM_scaled)
plt.title('Frequency Plot')
plt.show()
```



In [62]:
```
sns.boxplot(x = 'Cluster_ID',y = 'Monetary',data=RFM_scaled)
plt.title('Monetary Plot')
plt.show()
```



Cluster ID 0 : Cluster ID 0 contains customers who are having lowest frequency lowest recency and least money spent in our store these customers are least important from our perspective.

Cluster ID 1 : Cluster ID 1 contains customers who are having lower frequent, have spent medium amount of money and have not bought very recently there are more important than cluster 0 but not more important than other customers.

Cluster ID 2 : These customers are having medium frequency to visit store, they have spent some good amount after visiting the store so these are more important for the store.

Cluster ID 3 : These customers have bought most recently from the store, They have spent most money and also visit the store more frequently visiting the store so they are most important for the store.

Converting the processed files for tableau dashboarding purpose

In [63]:
```
df.to_csv('Master_data.csv')
RFM.to_csv('RFM_analysis.csv')
df_inertia.to_csv('Inertia.csv')
```

In [64]:
```
product_descr = df[['StockCode','Description']]
product_descr = product_descr.drop_duplicates()
product_descr.to_csv('Product_description.csv')
```

In [ ]: