Capstone Project: Retail

Problem Statement:

- It is a critical requirement for business to understand the value derived from a customer. RFM is a method used for analyzing customer value.
- Customer segmentation is the practice of segregating the customer base into groups of individuals based on some common characteristics such as age, gender, interests, and spending habits
- 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).

Dataset Description: This is a transnational data set which contains all the transactions that occurred between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique and all-occasion gifts.

- **InvoiceNo:** Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.
- **StockCode:** Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
- **Description:** Product (item) name. Nominal.
- **Quantity:** The quantities of each product (item) per transaction. Numeric.
- **InvoiceDate:** Invoice Date and time. Numeric, the day and time when each transaction was generated.
- **UnitPrice:** Unit price. Numeric, Product price per unit in sterling.
- **CustomerID:** Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
- **Country:** Country name. Nominal, the name of the country where each customer resides.

Project Task:

Data Cleaning:

1. Perform a preliminary data inspection and data cleaning.

   a. Check for missing data and formulate an apt strategy to treat them.

   b. Remove duplicate data records.

   c. Perform descriptive analytics on the given data.

Data Transformation:

1. Perform cohort analysis (a cohort is a group of subjects that share a defining characteristic). Observe how a cohort behaves across time and compare it to other cohorts.

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

   b. Analyze the retention rate of customers.

Project Task:

Data Modeling :

1. Build a RFM (Recency Frequency Monetary) model. Recency means the number of days since a customer made the last purchase. Frequency is the number of purchase in a given period. It could be 3 months, 6 months or 1 year. Monetary is the total amount of money a customer spent in that given period. Therefore, big spenders will be differentiated among other customers such as MVP (Minimum Viable Product) or VIP.

2. Calculate RFM metrics.

3. Build RFM Segments. Give recency, frequency, and monetary scores individually by dividing them into quartiles.

   b1. Combine three ratings to get a RFM segment (as strings).

   b2. Get the RFM score by adding up the three ratings.

   b3. Analyze the RFM segments by summarizing them and comment on the findings.

Note: Rate “recency" for customer who has been active more recently higher than the less recent customer, because each company wants its customers to be recent.

Note: Rate “frequency" and “monetary" higher, because the company wants the customer to visit more often and spend more money

Project Task:

Data Modeling :

1. Create clusters using k-means clustering algorithm.

   a. Prepare the data for the algorithm. If the data is asymmetrically distributed, manage the skewness with appropriate transformation. Standardize the data.

   b. Decide the optimum number of clusters to be formed.

   c. Analyze these clusters and comment on the results.

Project Task:

Data Reporting:

1. Create a dashboard in tableau by choosing appropriate chart types and metrics useful for the business. The dashboard must entail the following:

   a. Country-wise analysis to demonstrate average spend. Use a bar chart to show the monthly figures

   b. Bar graph of top 15 products which are mostly ordered by the users to show the number of products sold

   c. Bar graph to show the count of orders vs. hours throughout the day

   d. Plot the distribution of RFM values using histogram and frequency charts

   e. Plot error (cost) vs. number of clusters selected

   f. Visualize to compare the RFM values of the clusters using heatmap

>>>>>---

--->>>>>

SOLUTION:

(A) Data Cleaning

(1) Reading Data and Preliminary Data Inspection

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from datetime import timedelta
from pandas import ExcelWriter
```

In [2]:
```
df = pd.read_excel("Online Retail.xlsx")
df.head()
```

Out[2]:

	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 [3]:
```
# Check shape of data
df.shape
```

Out[3]:
```
(541909, 8)
```

In [4]:
```
# Check feature details of data
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
```

- **(a) Missing values treatment:**

In [5]:
```
# Check missing values in data
df.isnull().sum()
```

Out[5]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

In [6]:
```
# Calculating the Missing Values % contribution in DF
df_null = round(df.isnull().sum()/len(df)*100,2)
df_null
```

Out[6]:
```
InvoiceNo       0.00
StockCode       0.00
Description     0.27
Quantity        0.00
InvoiceDate     0.00
UnitPrice       0.00
CustomerID     24.93
Country         0.00
dtype: float64
```

As we can see two columns in data have missing values.

- Description - 0.27% (1454 nos.)
- CustomerID - 24.93% (135080)

CustomerID is important feature of our analysis since our analysis is centered around Customers only so we can not impute null values **CustomerID** with mean/ median/ mode in this case. We will check possibility to fill null values in **CustomerID** column by looking up for **InvoiceNo** of the row having null **CustomerID** in other rows where **CustomerID** is present. If there are still any null values in **CustomerID** after this process then we will drop complete row having missing **CustomerID**.

We can drop **Description** feature from our data since it is not not going to contribute in our model.

In [7]:
```
invoice_null_custid = set(df[df['CustomerID'].isnull()]['InvoiceNo'])
df[df['InvoiceNo'].isin(invoice_null_custid) & (~df['CustomerID'].isnull())]
```

Out[7]:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country

We could not find any value to impute null values in **CustomerID** column since all entries for a particular **InvoiceNo** have missing **CustomerID** if that particular **InvoiceNo** has null **CustomerID** in even one entry. So we will drop all rows having null values in **CustomerID**.

In [8]:
```
df = df.drop('Description', axis=1)
df = df.dropna()
df.shape
```
Out[8]:
```
(406829, 7)
```

- **(b) Remove duplicate data records:** Since our data is transactional data and it has duplicate entries for InvoiceNo and CustomerID, we will drop only those rows which are completely duplicated, not on the basis of any one particular column such as InvoiceNo or CustomerID etc.

In [9]:
```
df = df.drop_duplicates()
df.shape
```
Out[9]:
```
(401602, 7)
```

- **(c) Perform descriptive anaylysis on the given data:**

In [10]:
```
# CustomerID is 'float64', changing the datatype of CustomerId to string as Customer ID as

df['CustomerID'] = df['CustomerID'].astype(str)
```

In [11]:
```
df.describe(datetime_is_numeric=True)
```
Out[11]:

	Quantity	InvoiceDate	UnitPrice
count	401602.000000	401602	401602.000000
mean	12.182579	2011-07-10 12:08:08.129839872	3.474064
min	-80995.000000	2010-12-01 08:26:00	0.000000
25%	2.000000	2011-04-06 15:02:00	1.250000
50%	5.000000	2011-07-29 15:40:00	1.950000
75%	12.000000	2011-10-20 11:58:00	3.750000
max	80995.000000	2011-12-09 12:50:00	38970.000000
std	250.283248	NaN	69.764209

- **Quantity:** Average quantity of each product in transaction is 12.18. Also note that minimum value in **Quantity** column is negative. This implies that some customers had returned the product during our period of analysis.
- **InvoiceDate:** Our data has transaction between 01-12-2010 to 09-12-2011
- **UnitPrice:** Average price of each product in transactions is 3.47

In [12]:
```
df.describe(include=['O'])
```
Out[12]:

	InvoiceNo	StockCode	CustomerID	Country
count	401602	401602	401602	401602
unique	22190	3684	4372	37
top	576339	85123A	17841.0	United Kingdom

	InvoiceNo	StockCode	CustomerID	Country
freq	542	2065	7812	356726

- **InvoiceNo:** Total entries in preprocessed data are 4,01,602 but transactions are 22,190. Most number of entries (count of unique products) are in Invoice No. '576339' and is 542 nos.
- **StockCode:** There are total 3684 unique products in our data and product with stock code '85123A' appears most frequently (2065 times) in our data.
- **CustomerID:** There are 4372 unique customers in our final preprocessed data. Customer with ID '17841' appears most frequently in data (7812 times)
- **Country:** Company has customers across 37 countries. Most entries are from United Kingdom in our dataset (356726)

(B) Data Transformation

(2) Perform Cohort Analysis

- **(a) Create month cohort of customers and analyze active customers in each cohort:**

In [13]:
```
# Convert to InvoiceDate to Year-Month format
df['month_year'] = df['InvoiceDate'].dt.to_period('M')
df['month_year'].nunique()
```

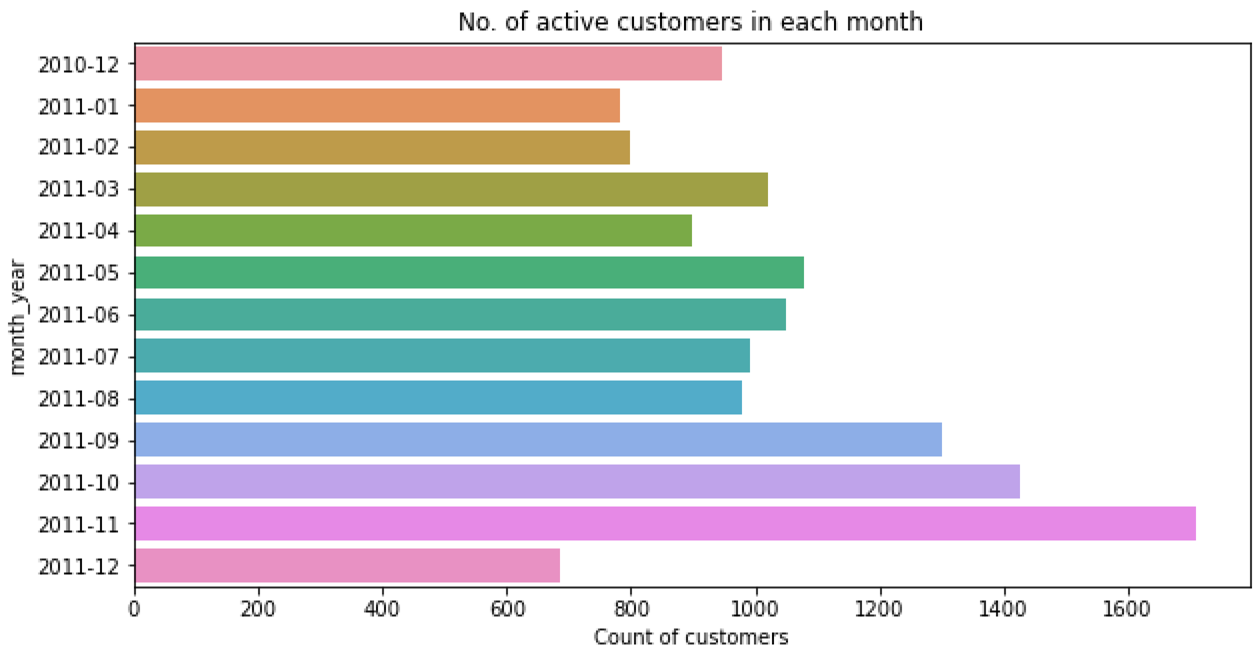
Out[13]:
```
13
```

In [14]:
```
month_cohort = df.groupby('month_year')['CustomerID'].nunique()
month_cohort
```

Out[14]:
```
month_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 [15]:
```
plt.figure(figsize=(10,5))
sns.barplot(y = month_cohort.index, x = month_cohort.values);
plt.xlabel("Count of customers")
plt.title("No. of active customers in each month")
```

Out[15]:
```
Text(0.5, 1.0, 'No. of active customers in each month')
```



- **(b) Analyze the retention rate of customers:**

In [16]:
```
month_cohort - month_cohort.shift(1)
```

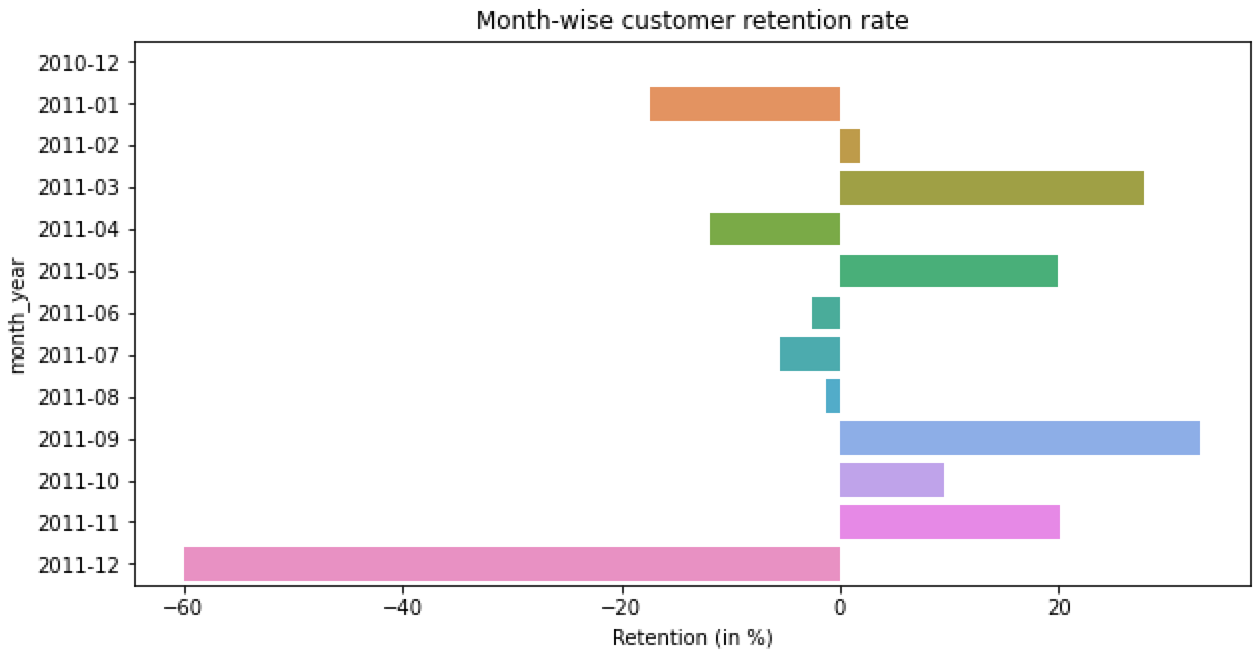
Out[16]:
```
month_year
2010-12       NaN
2011-01    -165.0
2011-02      15.0
2011-03     222.0
2011-04    -121.0
2011-05     180.0
2011-06     -28.0
2011-07     -58.0
2011-08     -13.0
2011-09     322.0
2011-10     123.0
2011-11     286.0
2011-12   -1025.0
Freq: M, Name: CustomerID, dtype: float64
```

In [17]:
```
retention_rate = round(month_cohort.pct_change(periods=1)*100,2)
retention_rate
```

Out[17]:
```
month_year
2010-12      NaN
2011-01   -17.41
2011-02     1.92
2011-03    27.82
2011-04   -11.86
2011-05    20.02
2011-06    -2.59
2011-07    -5.52
2011-08    -1.31
2011-09    32.86
2011-10     9.45
2011-11    20.07
2011-12   -59.91
Freq: M, Name: CustomerID, dtype: float64
```

In [18]:
```
plt.figure(figsize=(10,5))
sns.barplot(y = retention_rate.index, x = retention_rate.values);
```

```
plt.xlabel("Retention (in %)")
plt.title("Month-wise customer retention rate");
```



Monetary analysis:

In [19]:
```
df['amount'] = df['Quantity']*df['UnitPrice']
df.head()
```

Out[19]:

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	month_year	amount
0	536365	85123A	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010-12	15.30
1	536365	71053	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12	20.34
2	536365	84406B	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010-12	22.00
3	536365	84029G	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12	20.34
4	536365	84029E	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12	20.34

In [20]:
```
df_monetary = df.groupby('CustomerID').sum()['amount'].reset_index()
df_monetary
```

Out[20]:

	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

	CustomerID	amount
...	...	...
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 [21]:
df_frequency = df.groupby('CustomerID').nunique()['InvoiceNo'].reset_index()
# df_freqency = df.drop_duplicates('InvoiceNo').groupby('CustomerID').count()['InvoiceNo'].
df_frequency
```

Out[21]:

	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 [22]:
# We will fix reference date for calculating recency as last transaction day in data + 1 d
ref_day = max(df['InvoiceDate']) + timedelta(days=1)
df['days_to_last_order'] = (ref_day - df['InvoiceDate']).dt.days
df.head()
```

Out[22]:

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	month_year	amount	days_to_las
0	536365	85123A	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010-12	15.30	
1	536365	71053	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12	20.34	
2	536365	84406B	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	2010-12	22.00	

	InvoiceNo	StockCode	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	month_year	amount	days_to_las
3	536365	84029G	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12	20.34	
4	536365	84029E	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12	20.34	

In [23]:
```
df_recency = df.groupby('CustomerID')['days_to_last_order'].min().reset_index()
df_recency
```

Out[23]:

	CustomerID	days_to_last_order
0	12346.0	326
1	12347.0	2
2	12348.0	75
3	12349.0	19
4	12350.0	310
...	...	...
4367	18280.0	278
4368	18281.0	181
4369	18282.0	8
4370	18283.0	4
4371	18287.0	43

4372 rows × 2 columns

Calculate RFM metrics:

In [24]:
```
df_rf = pd.merge(df_recency, df_frequency,  on='CustomerID', how='inner')
df_rfm = pd.merge(df_rf, df_monetary, on='CustomerID', how='inner')
df_rfm.columns = ['CustomerID', 'Recency', 'Frequency', 'Monetary']
df_rfm.head()
```

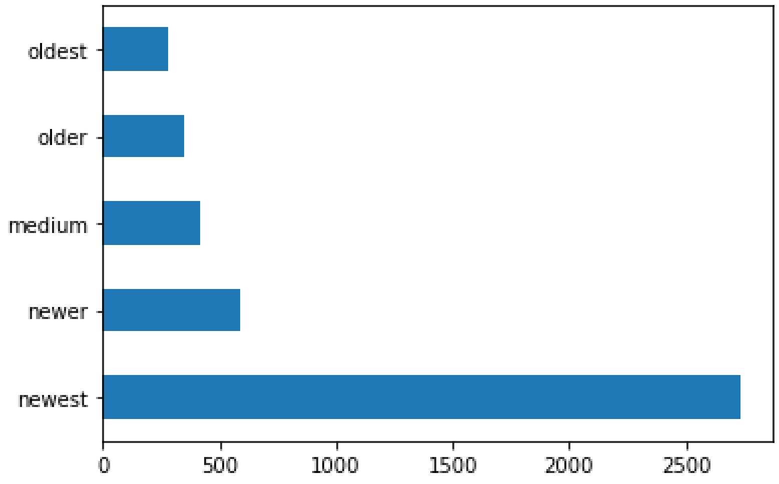
Out[24]:

	CustomerID	Recency	Frequency	Monetary
0	12346.0	326	2	0.00
1	12347.0	2	7	4310.00
2	12348.0	75	4	1797.24
3	12349.0	19	1	1757.55
4	12350.0	310	1	334.40

Build RFM Segments:

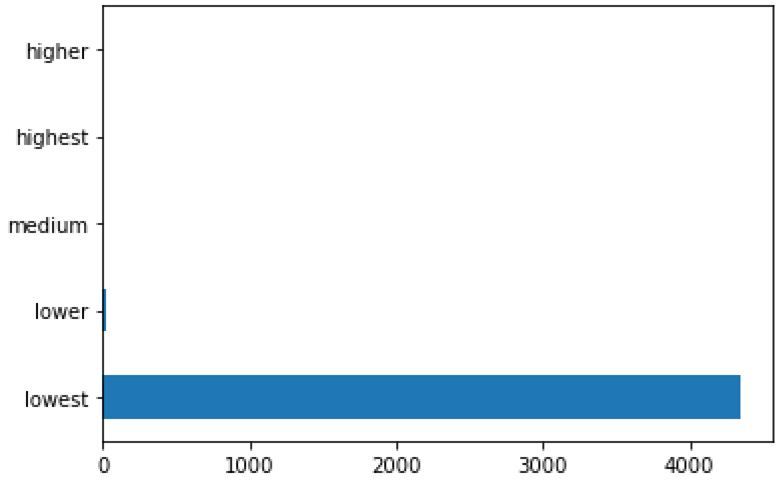
In [25]:
```
df_rfm['recency_labels'] = pd.cut(df_rfm['Recency'], bins=5,
                                  labels=['newest', 'newer', 'medium', 'older', 'oldest
df_rfm['recency_labels'].value_counts().plot(kind='barh');
df_rfm['recency_labels'].value_counts()
```

```
Out[25]: newest     2734
         newer       588
         medium      416
         older       353
         oldest      281
         Name: recency_labels, dtype: int64
```



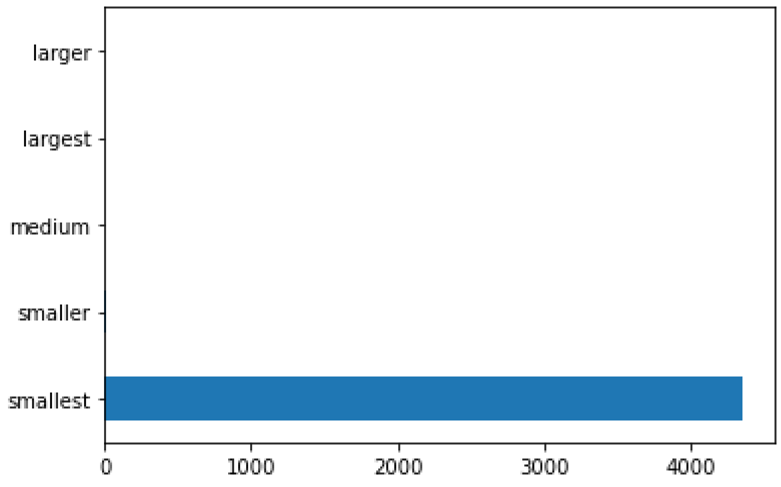
```
In [26]: df_rfm['frequency_labels'] = pd.cut(df_rfm['Frequency'], bins=5, labels=['lowest', 'lower'
         df_rfm['frequency_labels'].value_counts().plot(kind='barh');
         df_rfm['frequency_labels'].value_counts()
```

```
Out[26]: lowest      4348
         lower         18
         medium         3
         highest        2
         higher         1
         Name: frequency_labels, dtype: int64
```



```
In [27]: df_rfm['monetary_labels'] = pd.cut(df_rfm['Monetary'], bins=5, labels=['smallest', 'smalle
         df_rfm['monetary_labels'].value_counts().plot(kind='barh');
         df_rfm['monetary_labels'].value_counts()
```

```
Out[27]: smallest    4357
         smaller        9
         medium         3
         largest        2
         larger         1
         Name: monetary_labels, dtype: int64
```



In [28]:
```
df_rfm['rfm_segment'] = df_rfm[['recency_labels','frequency_labels','monetary_labels']].ag
df_rfm.head()
```

Out[28]:

	CustomerID	Recency	Frequency	Monetary	recency_labels	frequency_labels	monetary_labels	rfm_segment
0	12346.0	326	2	0.00	oldest	lowest	smallest	oldest-lowest-smallest
1	12347.0	2	7	4310.00	newest	lowest	smallest	newest-lowest-smallest
2	12348.0	75	4	1797.24	newest	lowest	smallest	newest-lowest-smallest
3	12349.0	19	1	1757.55	newest	lowest	smallest	newest-lowest-smallest
4	12350.0	310	1	334.40	oldest	lowest	smallest	oldest-lowest-smallest

RFM Score:

In [29]:
```
recency_dict = {'newest': 5, 'newer':4, 'medium': 3, 'older':2, 'oldest':1}
frequency_dict = {'lowest':1, 'lower':2, 'medium': 3, 'higher':4, 'highest':5}
monetary_dict = {'smallest':1, 'smaller':2, 'medium': 3, 'larger':4, 'largest':5}

df_rfm['rfm_score'] = df_rfm['recency_labels'].map(recency_dict).astype(int)+ df_rfm['freq
df_rfm.head(10)
```

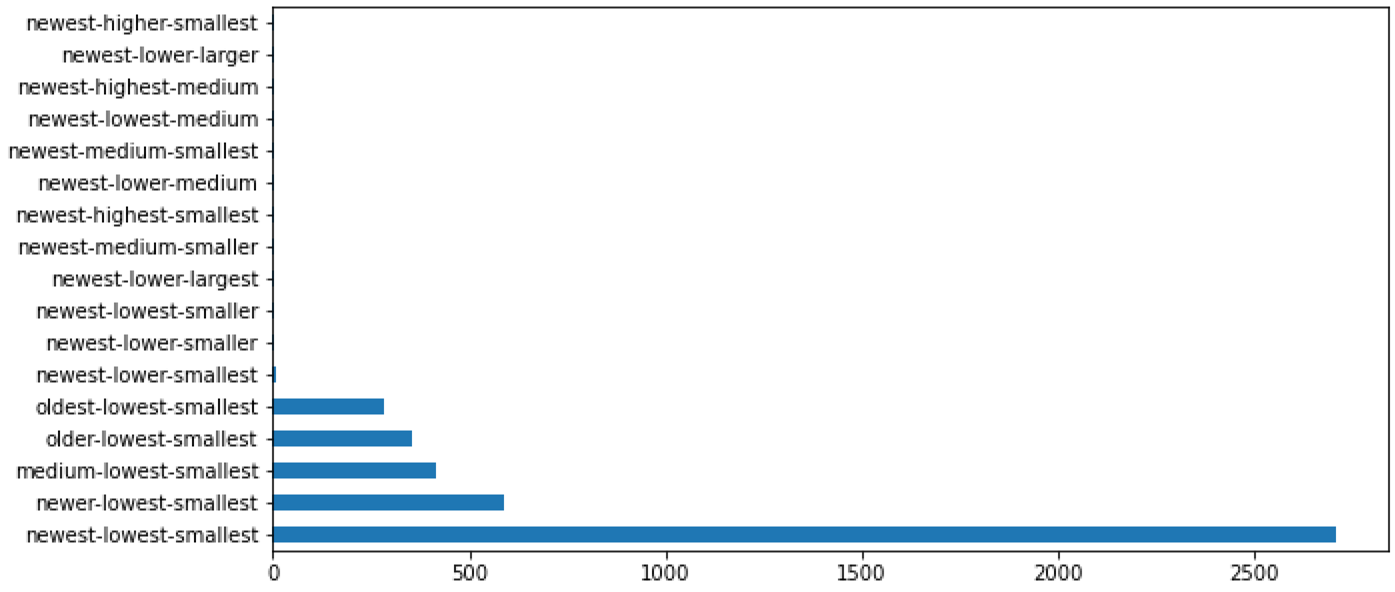
Out[29]:

	CustomerID	Recency	Frequency	Monetary	recency_labels	frequency_labels	monetary_labels	rfm_segment	rfm
0	12346.0	326	2	0.00	oldest	lowest	smallest	oldest-lowest-smallest	
1	12347.0	2	7	4310.00	newest	lowest	smallest	newest-lowest-smallest	
2	12348.0	75	4	1797.24	newest	lowest	smallest	newest-lowest-smallest	

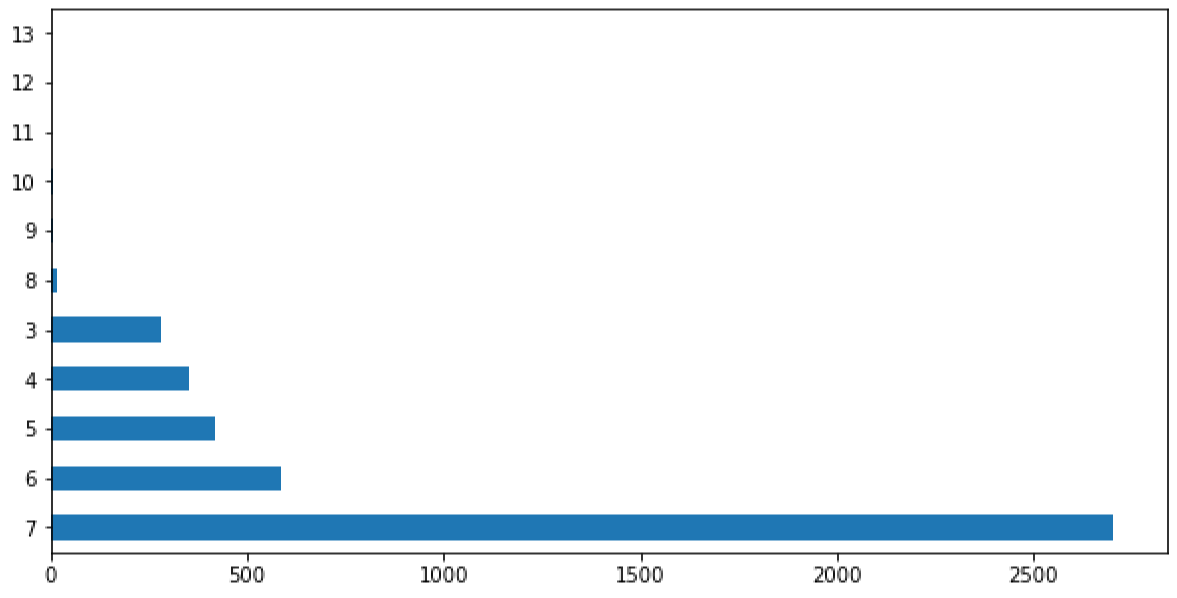
	CustomerID	Recency	Frequency	Monetary	recency_labels	frequency_labels	monetary_labels	rfm_segment	rfm
3	12349.0	19	1	1757.55	newest	lowest	smallest	newest-lowest-smallest	
4	12350.0	310	1	334.40	oldest	lowest	smallest	oldest-lowest-smallest	
5	12352.0	36	11	1545.41	newest	lowest	smallest	newest-lowest-smallest	
6	12353.0	204	1	89.00	medium	lowest	smallest	medium-lowest-smallest	
7	12354.0	232	1	1079.40	older	lowest	smallest	older-lowest-smallest	
8	12355.0	214	1	459.40	medium	lowest	smallest	medium-lowest-smallest	
9	12356.0	23	3	2811.43	newest	lowest	smallest	newest-lowest-smallest	

Analyze RFM Segment and Score:

In [30]:
```
df_rfm['rfm_segment'].value_counts().plot(kind='barh', figsize=(10, 5));
```



In [31]:
```
df_rfm['rfm_score'].value_counts().plot(kind='barh', figsize=(10, 5));
```



Data Modeling:

1. Create clusters using k-means clustering algorithm.

a. Prepare the data for the algorithm. If the data is asymmetrically distributed, manage the skewness with appropriate transformation. Standardize the data.

In [32]:

```
print(df_rfm.shape)
df_rfm.head()
```

```
(4372, 9)
```

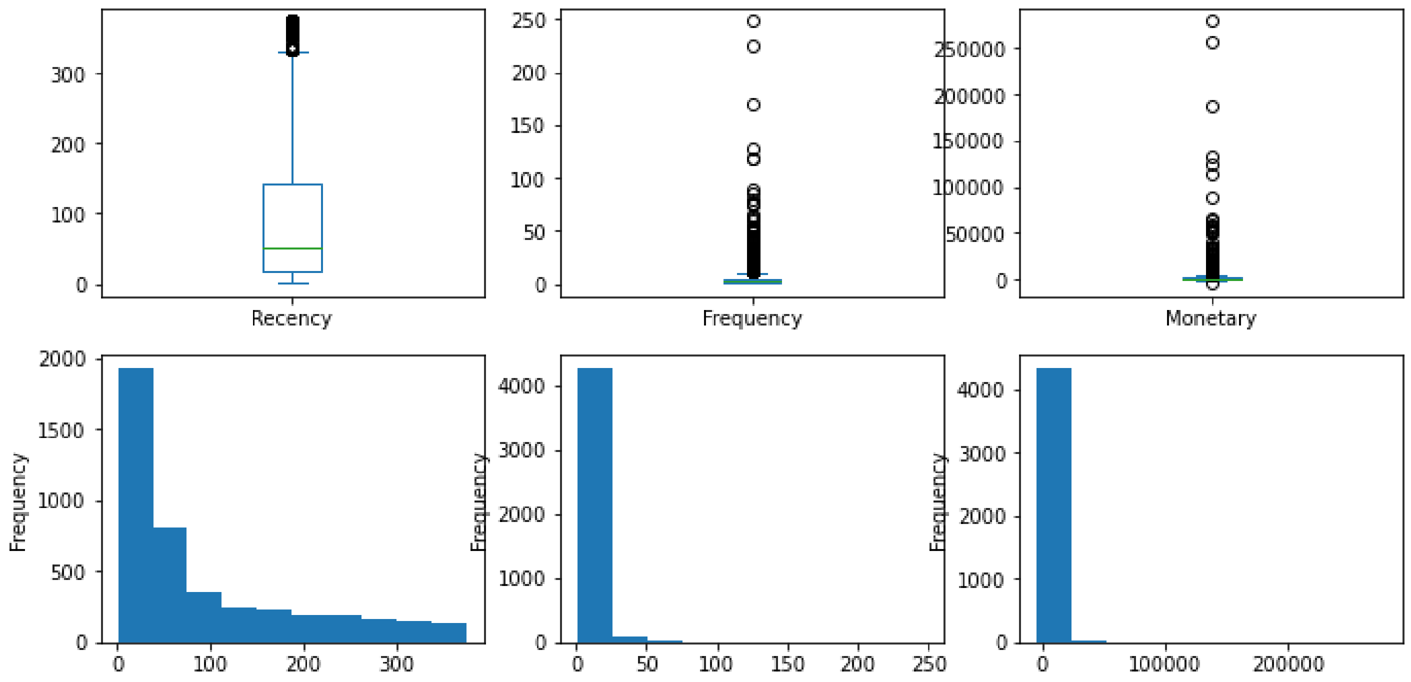
Out[32]:

	CustomerID	Recency	Frequency	Monetary	recency_labels	frequency_labels	monetary_labels	rfm_segment	rfm
0	12346.0	326	2	0.00	oldest	lowest	smallest	oldest-lowest-smallest	
1	12347.0	2	7	4310.00	newest	lowest	smallest	newest-lowest-smallest	
2	12348.0	75	4	1797.24	newest	lowest	smallest	newest-lowest-smallest	
3	12349.0	19	1	1757.55	newest	lowest	smallest	newest-lowest-smallest	
4	12350.0	310	1	334.40	oldest	lowest	smallest	oldest-lowest-smallest	

In [33]:

```
plt.figure(figsize=(12,6))

for i, feature in enumerate(['Recency', 'Frequency', 'Monetary']):
    plt.subplot(2,3,i+1)
    df_rfm[feature].plot(kind='box')
    plt.subplot(2,3,i+1+3)
    df_rfm[feature].plot(kind='hist')
```



Outliers: Frequency and Monetary features in above data seem to have lot of outliers. Lets drop them.

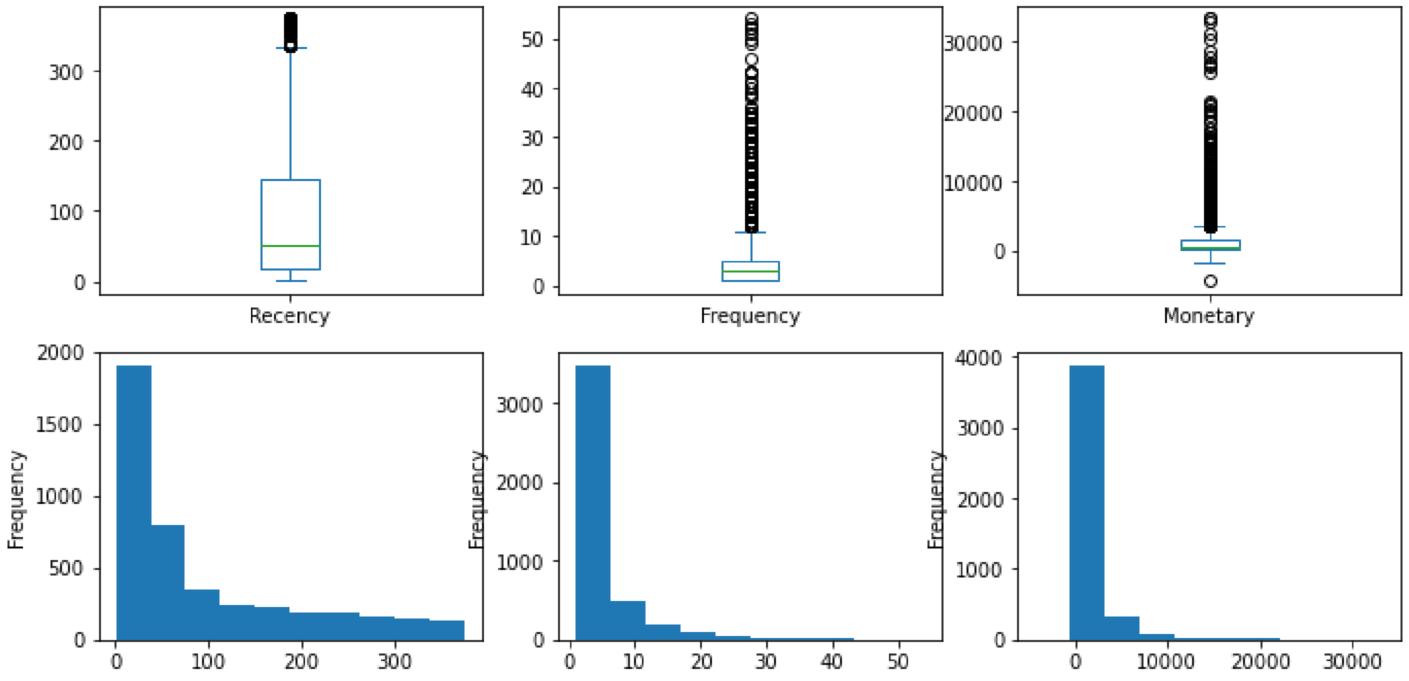
In [34]:
```
df_rfm = df_rfm[(df_rfm['Frequency']<60) & (df_rfm['Monetary']<40000)]
df_rfm.shape
```

Out[34]:
```
(4346, 9)
```

26 Customers removed as outlier from out data.

In [35]:
```
plt.figure(figsize=(12,6))

for i, feature in enumerate(['Recency', 'Frequency', 'Monetary']):
    plt.subplot(2,3,i+1)
    df_rfm[feature].plot(kind='box')
    plt.subplot(2,3,i+1+3)
    df_rfm[feature].plot(kind='hist')
```



Log Transformation: Now since all three features have right skewed data therefore we will use log transformation of these features in our model.

In [36]:
```
df_rfm_log_trans = pd.DataFrame()
df_rfm_log_trans['Recency'] = np.log(df_rfm['Recency'])
df_rfm_log_trans['Frequency'] = np.log(df_rfm['Frequency'])
df_rfm_log_trans['Monetary'] = np.log(df_rfm['Monetary']-df_rfm['Monetary'].min()+1)
```

Standard Scalar Transformation: It is extremely important to rescale the features so that they have a comparable scale.

In [37]:
```
scaler = StandardScaler()

df_rfm_scaled = scaler.fit_transform(df_rfm_log_trans[['Recency', 'Frequency', 'Monetary']
df_rfm_scaled

df_rfm_scaled = pd.DataFrame(df_rfm_scaled)
df_rfm_scaled.columns = ['Recency', 'Frequency', 'Monetary']
df_rfm_scaled.head()
```

Out[37]:

	Recency	Frequency	Monetary
0	1.402988	-0.388507	-0.770922
1	-2.100874	0.967301	1.485132
2	0.392218	0.361655	0.364190
3	-0.552268	-1.138669	0.342970
4	1.368370	-1.138669	-0.527416

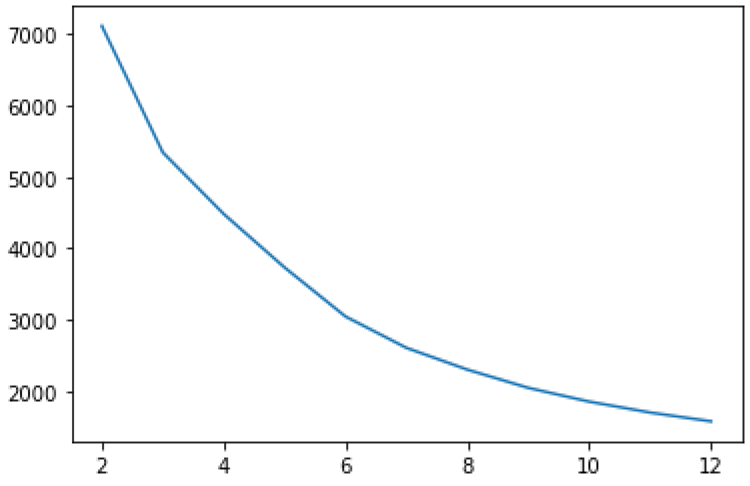
b. Build K-Means Clustering Model and Decide the optimum number of clusters to be formed.

In [38]:
```
# k-means with some arbitrary k
kmeans = KMeans(n_clusters=3, max_iter=50)
kmeans.fit(df_rfm_scaled)
```

Out[38]:
```
KMeans(max_iter=50, n_clusters=3)
```

In [39]:
```
kmeans.labels_
```
Out[39]:
```
array([1, 2, 0, ..., 0, 2, 0])
```
In [40]:
```
# Finding the Optimal Number of Clusters with the help of Elbow Curve/ SSD
ssd = []
range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters, max_iter=100)
    kmeans.fit(df_rfm_scaled)

    ssd.append(kmeans.inertia_)

# plot the SSDs for each n_clusters
plt.plot(range_n_clusters,ssd);
```


In [41]:
```
# Creating dataframe for exporting to create visualization in tableau later
df_inertia = pd.DataFrame(list(zip(range_n_clusters, ssd)), columns=['clusters', 'intertia
df_inertia
```
Out[41]:

	clusters	intertia
0	2	7113.097396
1	3	5343.115435
2	4	4481.024256
3	5	3730.838474
4	6	3044.793367
5	7	2605.826255
6	8	2301.172692
7	9	2045.838544
8	10	1852.943004
9	11	1700.397856
10	12	1577.081020

In [42]:
```
# Finding the Optimal Number of Clusters with the help of Silhouette Analysis
range_n_clusters = [2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(df_rfm_scaled)

    cluster_labels = kmeans.labels_

    silhouette_avg = silhouette_score(df_rfm_scaled, cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, silhouett
```

```
For n_clusters=2, the silhouette score is 0.44132753537785846
For n_clusters=3, the silhouette score is 0.3803019251906771
For n_clusters=4, the silhouette score is 0.3623606426972478
For n_clusters=5, the silhouette score is 0.3438837918281012
For n_clusters=6, the silhouette score is 0.3443915151384028
For n_clusters=7, the silhouette score is 0.3428617732216645
For n_clusters=8, the silhouette score is 0.3354671816479655
For n_clusters=9, the silhouette score is 0.3464234161259565
For n_clusters=10, the silhouette score is 0.35706878411373083
```

We can select optimum number of clusters as 3 in our final model

In [43]:

```
# Final model with k=3
kmeans = KMeans(n_clusters=3, max_iter=50)
kmeans.fit(df_rfm_scaled)
```

Out[43]:

```
KMeans(max_iter=50, n_clusters=3)
```

c. Analyze these clusters and comment on the results.

In [44]:

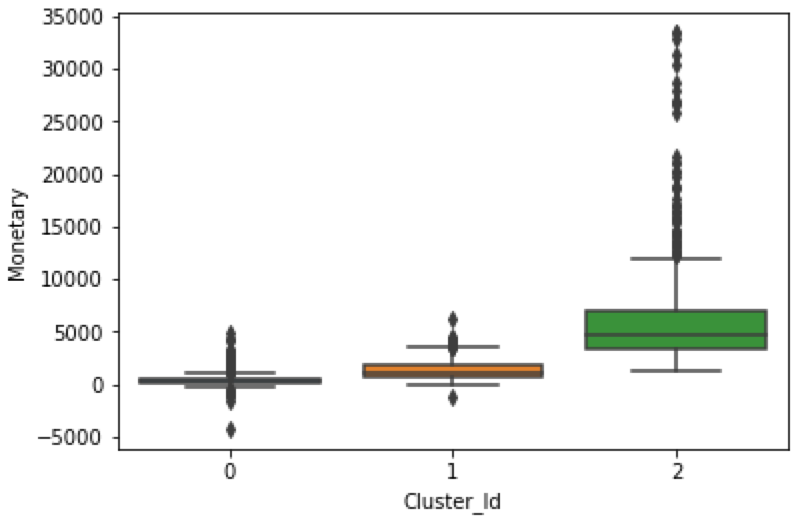
```
# assign the label
df_rfm['Cluster_Id'] = kmeans.labels_
df_rfm.head()
```

Out[44]:

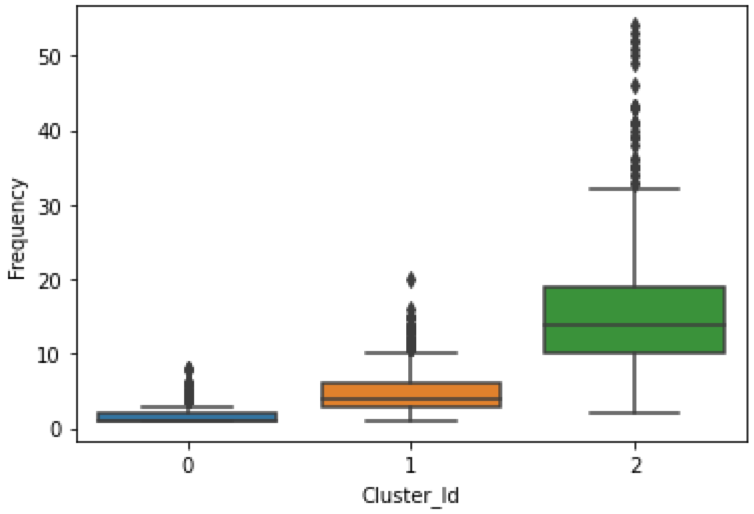
	CustomerID	Recency	Frequency	Monetary	recency_labels	frequency_labels	monetary_labels	rfm_segment	rfm
0	12346.0	326	2	0.00	oldest	lowest	smallest	oldest-lowest-smallest	
1	12347.0	2	7	4310.00	newest	lowest	smallest	newest-lowest-smallest	
2	12348.0	75	4	1797.24	newest	lowest	smallest	newest-lowest-smallest	
3	12349.0	19	1	1757.55	newest	lowest	smallest	newest-lowest-smallest	
4	12350.0	310	1	334.40	oldest	lowest	smallest	oldest-lowest-smallest	

In [45]:

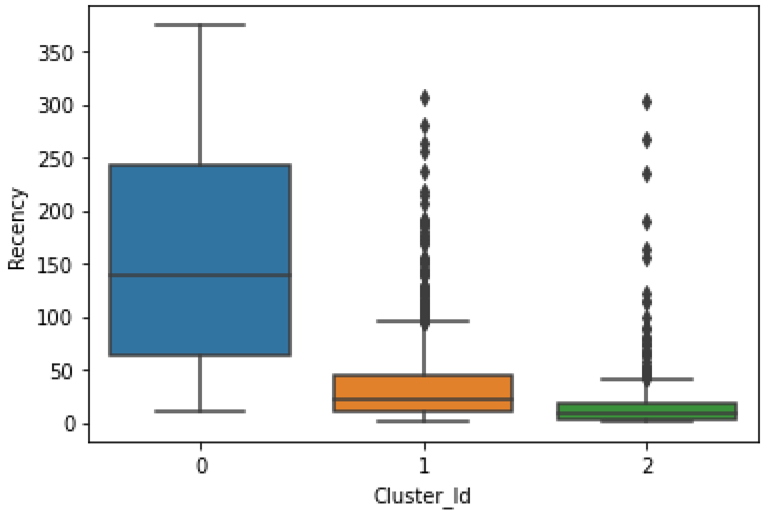
```
# Box plot to visualize Cluster Id vs Monetary
sns.boxplot(x='Cluster_Id', y='Monetary', data=df_rfm);
```



In [46]:
```
# Box plot to visualize Cluster Id vs Frequency
sns.boxplot(x='Cluster_Id', y='Frequency', data=df_rfm);
```



In [47]:
```
# Box plot to visualize Cluster Id vs Recency
sns.boxplot(x='Cluster_Id', y='Recency', data=df_rfm);
```



Inference:

As we can observe from above boxplots that our model has nicely created 3 segements of customer with the interpretation as below:

- Customers with Cluster Id 0 are less frequent buyers with low monetary expenditure and also they have not purchased anything in recent time and hence least important for business.
- Customers with Cluster Id 1 are the customers having Recency, Frequency and Monetary score in the medium range.
- Customers with Cluster Id 2 are the most frequent buyers, spending high amount and recently placing orders so they are the most important customers from business point of view.

Data Reporting:

1. Create a dashboard in tableau by choosing appropriate chart types and metrics useful for the business. The dashboard must entail the following:

```
 a. Country-wise analysis to demonstrate average spend. Use a bar chart to show the
monthly figures
 b. Bar graph of top 15 products which are mostly ordered by the users to show the
number of products sold
 c. Bar graph to show the count of orders vs. hours throughout the day
 d. Plot the distribution of RFM values using histogram and frequency charts
 e. Plot error (cost) vs. number of clusters selected
 f. Visualize to compare the RFM values of the clusters using heatmap
```

In [48]:
```python
# Writing dataframe to excel file for creating visualization in tableau
writer = pd.ExcelWriter('C:\\Users\\ishan\\mgpython\\Capstone Project\\Retail\\outpu

df.to_excel(writer, sheet_name='master_data', index=False)
df_rfm.to_excel(writer, sheet_name='rfm_data', index=False)
df_inertia.to_excel(writer, sheet_name='inertia', index=False)
writer.save()
```

In [49]:
```python
product_desc = pd.read_excel("Online Retail.xlsx")
product_desc = product_desc[['StockCode', 'Description']]
product_desc = product_desc.drop_duplicates()
product_desc.to_csv('product_desc.csv', index=False)
```

Please refer Dashboard created in Tableau for visualization and graphs