In [67]:
```
#importing essential libraries for building a machine learning model
#task 2

import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import scipy.cluster.hierarchy as shc
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,recall_score,precision_score,f1_score
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
import warnings
warnings.filterwarnings("ignore")
```

In [68]:
```
df= pd.read_csv("Online Retail.csv",encoding="ISO-8859-1")
```

In [69]:
```
df.head()
```

Out[69]:

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

In [70]:
```
df.head(24)
```

Out[70]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	12/1/2010 8:26	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	12/1/2010 8:26	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	12/1/2010 8:28	1.85	17850.0	United Kingdom
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	12/1/2010 8:34	1.69	13047.0	United Kingdom
10	536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	12/1/2010 8:34	2.10	13047.0	United Kingdom
11	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	12/1/2010 8:34	2.10	13047.0	United Kingdom
12	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	12/1/2010 8:34	3.75	13047.0	United Kingdom
13	536367	22310	IVORY KNITTED MUG COSY	6	12/1/2010 8:34	1.65	13047.0	United Kingdom
14	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	12/1/2010 8:34	4.25	13047.0	United Kingdom
15	536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	12/1/2010 8:34	4.95	13047.0	United Kingdom
16	536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	12/1/2010 8:34	9.95	13047.0	United Kingdom
17	536367	21754	HOME BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047.0	United Kingdom
18	536367	21755	LOVE BUILDING BLOCK WORD	3	12/1/2010 8:34	5.95	13047.0	United Kingdom
19	536367	21777	RECIPE BOX WITH METAL HEART	4	12/1/2010 8:34	7.95	13047.0	United Kingdom
20	536367	48187	DOORMAT NEW ENGLAND	4	12/1/2010 8:34	7.95	13047.0	United Kingdom
21	536368	22960	JAM MAKING SET WITH JARS	6	12/1/2010 8:34	4.25	13047.0	United Kingdom
22	536368	22913	RED COAT RACK PARIS FASHION	3	12/1/2010 8:34	4.95	13047.0	United Kingdom
23	536368	22912	YELLOW COAT RACK PARIS FASHION	3	12/1/2010 8:34	4.95	13047.0	United Kingdom

In [71]:
```
df.shape
## dataset rows and columns
```

Out[71]:
```
(541909, 8)
```

In [72]:
```
df.columns
```

Out[72]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [73]:
```
df.columns
```

Out[73]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [74]:
```
#gives all info about the dataset.
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  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

In [75]:
```
##MISSING VALUE TREATMENT
df.isna().sum()
```

Out[75]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

In [76]:
```
df['Description'].fillna(df['Description'].mode()[0],inplace=True)
```

In [77]:
```
df.isna().sum()
```

Out[77]:
```
InvoiceNo           0
StockCode           0
Description         0
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

In [78]:
```
## % of missing values
(df.isnull().sum()/len(df)*100).sort_values(ascending=False)  ## % of missing values
```

Out[78]:
```
CustomerID     24.926694
InvoiceNo       0.000000
StockCode       0.000000
Description     0.000000
Quantity        0.000000
InvoiceDate     0.000000
UnitPrice       0.000000
Country         0.000000
dtype: float64
```

In [79]:
```
print("Number of unique customers IDs:",len(df['CustomerID'].unique()))
```

```
Number of unique customers IDs: 4373
```

In [80]:
```
df = df.dropna()
```

In [81]:
```
df.shape
```

Out[81]:
```
(406829, 8)
```

In [82]:
```
##EXPLORATORY DATA ANALYSIS

#DATA CLEANING

print(df.info())
print(df.shape)
print(df.isnull().sum())
df = df.dropna()
print(df.info())
print(df.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    406829 non-null  object
 1   StockCode    406829 non-null  object
 2   Description  406829 non-null  object
 3   Quantity     406829 non-null  int64
 4   InvoiceDate  406829 non-null  object
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 27.9+ MB
None
(406829, 8)
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
Int64Index: 406829 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    406829 non-null  object
 1   StockCode    406829 non-null  object
 2   Description  406829 non-null  object
 3   Quantity     406829 non-null  int64
 4   InvoiceDate  406829 non-null  object
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 27.9+ MB
None
(406829, 8)
```

In [83]:
```
cat = []
con = []

for i in df.columns:
    if(df[i].dtypes=="object"):
        cat.append(i)
    else:
        con.append(i)
```

In [84]:
```
df_cat = cat
df_cat=df[cat]
df_cat
```

Out[84]:

	InvoiceNo	StockCode	Description	InvoiceDate	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	12/1/2010 8:26	United Kingdom
1	536365	71053	WHITE METAL LANTERN	12/1/2010 8:26	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	12/1/2010 8:26	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	12/1/2010 8:26	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	12/1/2010 8:26	United Kingdom
...	...	...	...	...	...
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12/9/2011 12:50	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	12/9/2011 12:50	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	12/9/2011 12:50	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	12/9/2011 12:50	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	12/9/2011 12:50	France

406829 rows × 5 columns

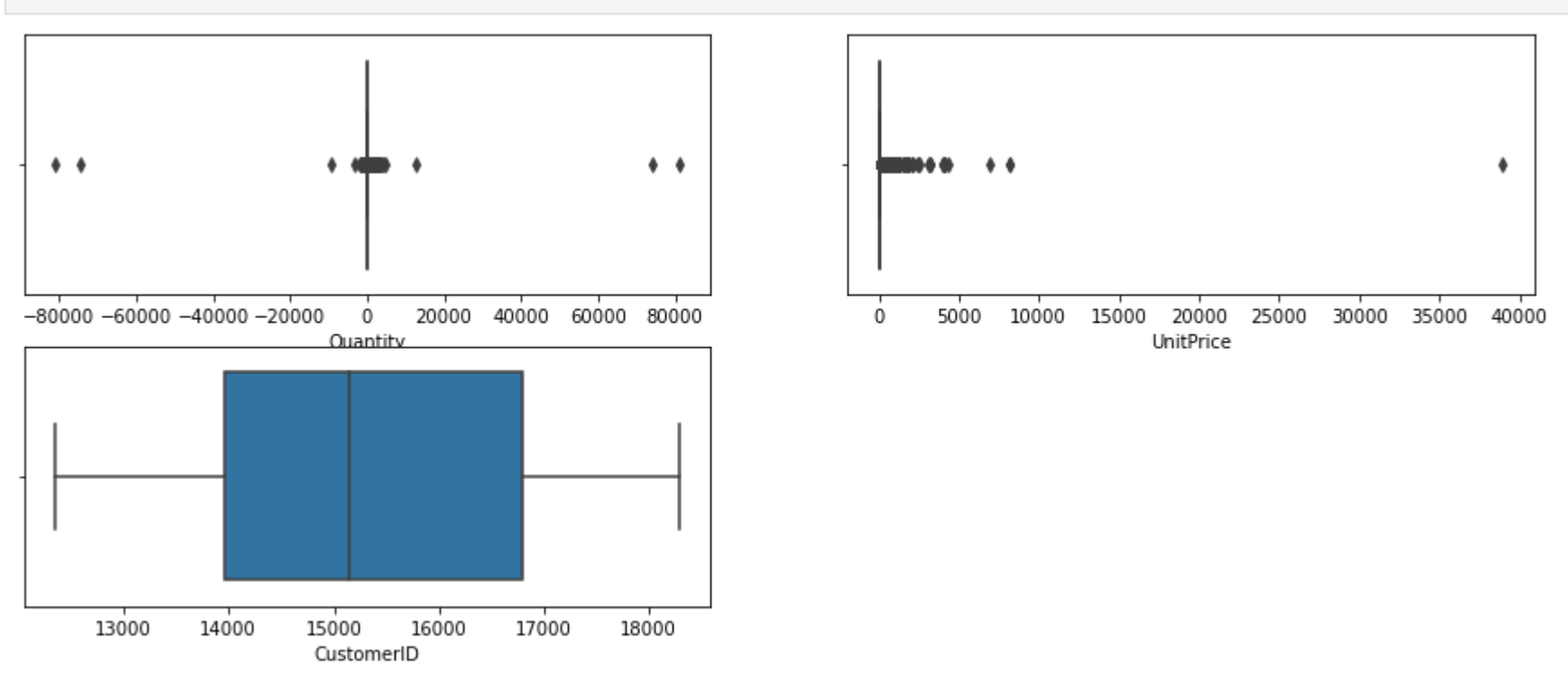
In [85]:
```
df_con = con
df_con=df[con]
df_con
```

Out[85]:

	Quantity	UnitPrice	CustomerID
0	6	2.55	17850.0
1	6	3.39	17850.0
2	8	2.75	17850.0
3	6	3.39	17850.0
4	6	3.39	17850.0
...	...	...	...
541904	12	0.85	12680.0
541905	6	2.10	12680.0
541906	4	4.15	12680.0
541907	4	4.15	12680.0
541908	3	4.95	12680.0

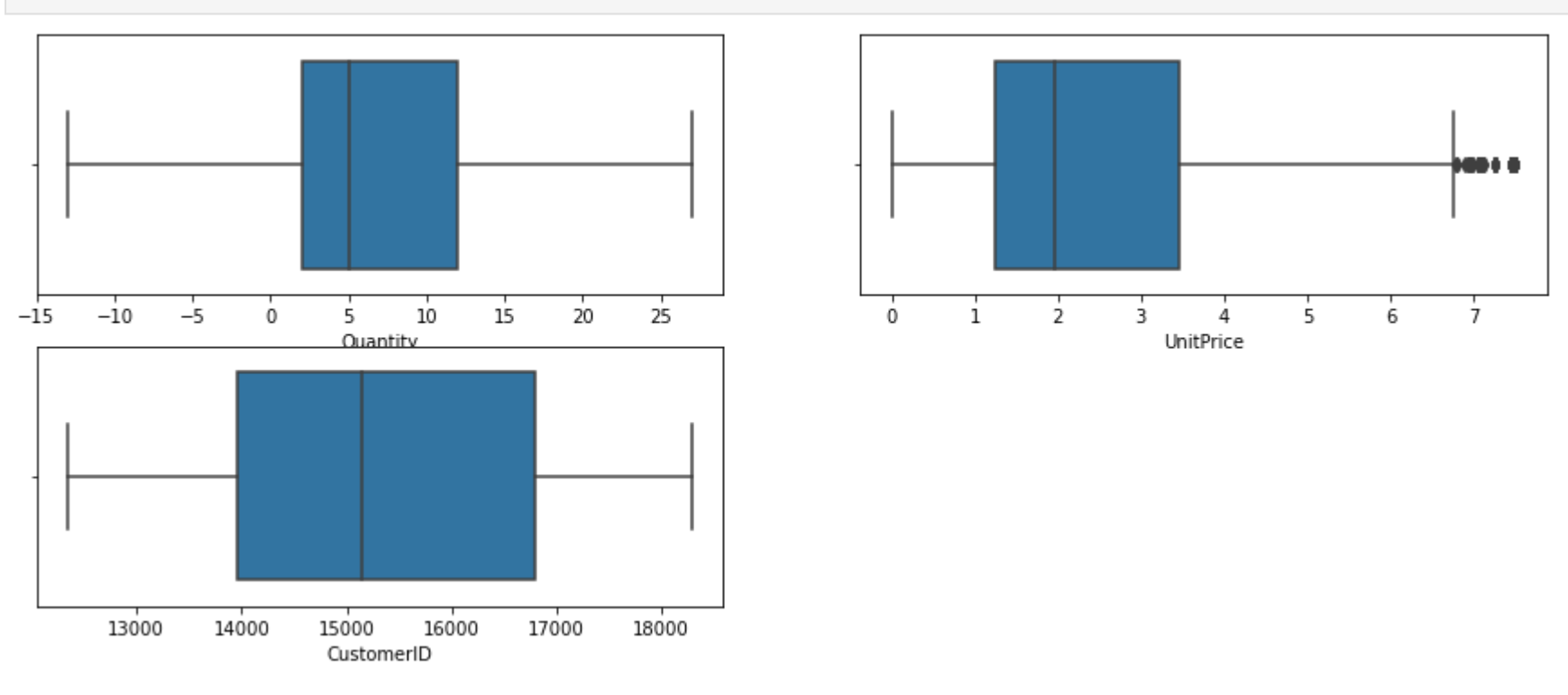
406829 rows × 3 columns

In [86]:
```
##Removing outliers
plt.figure(figsize=(15,9))
for x1,i in enumerate(df_con.columns):
    if df_con[i].dtypes=='int64' or df_con[i].dtypes=='float64':
        plt.subplot(3,2,x1+1)
        sns.boxplot(df_con[i])
```



In [87]:
```
for i in df_con.columns:
    q1 = df_con[i].quantile(0.25)
    q3 = df_con[i].quantile(0.75)
    IQR = q3-q1
    uppertail = q3+1.5*IQR
    lowertail = q1-1.5*IQR
    df_con.loc[(df_con[i]>uppertail)|(df_con[i]<lowertail)]
    mean_1= df_con[i].mean()
    df_con.loc[(df_con[i]>uppertail)| (df_con[i]<lowertail),i]=mean_1
```

In [88]:
```
plt.figure(figsize=(15,9))
for x1,i in enumerate(df_con.columns):
    if df_con[i].dtypes=='int64' or df_con[i].dtypes=='float64':
        plt.subplot(3,2,x1+1)
        sns.boxplot(df_con[i])
```



In [89]:
```
# Convert InvoiceDate to datetime
df["InvoiceDate"] = pd.to_datetime(df["InvoiceDate"])
```

In [90]:
```
# Calculate total purchase amount
df["TotalAmount"] = df["Quantity"] * df["UnitPrice"]
df["TotalAmount"]
```

Out[90]:
```
0         15.30
1         20.34
2         22.00
3         20.34
4         20.34
          ...
541904    10.20
541905    12.60
541906    16.60
541907    16.60
541908    14.85
Name: TotalAmount, Length: 406829, dtype: float64
```

In [91]:
```
Adding new attributes ;
RFM
R- Reference(Days since last purchase)
F- Frequency(Total number of purchases)
M-Monetory Value(Total money, customer spent.)
```

```
  Input In [91]
    Adding new attributes ;
           ^
SyntaxError: invalid syntax
```

In [ ]:
```
#data preprocessing
df['CustomerID'] = df['CustomerID'].astype(str)
df['Amount'] = df['Quantity']*df['UnitPrice']
rfm_df_m = df.groupby('CustomerID')['Amount'].sum()
rfm_df_m.reset_index()
rfm_df_m.columns = ['CustomerID', 'Amount']
print(rfm_df_m)
```

In [ ]:
```
rfm_df_f = df.groupby('CustomerID')['InvoiceNo'].count()
rfm_df_f = rfm_df_f.reset_index()
rfm_df_f.columns = ['CustomerID', 'Frequency']
print(rfm_df_f)
```

In [ ]:
```
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'],format='%d-%m-%Y %H:%M')
max_date = max(df['InvoiceDate'])
df['Diff'] = max_date - df['InvoiceDate']
rfm_df_p = df.groupby('CustomerID')['Diff'].min()
rfm_df_p = rfm_df_p.reset_index()
rfm_df_p.columns = ['CustomerID','Diff']
rfm_df_p['Diff'] = rfm_df_p['Diff'].dt.days
print(rfm_df_p)
```

In [ ]:
```
rfm_df_final = pd.merge(rfm_df_m,rfm_df_f,on='CustomerID',how='inner')
rfm_df_final = pd.merge(rfm_df_final,rfm_df_p,on='CustomerID',how='inner')
rfm_df_final.columns = ['CustomerID', 'Amount', 'Frequency', 'Recency']
print(rfm_df_final.head())
```

In [ ]:
```
Q1 = rfm_df_final.Amount.quantile(0.05)
Q3 = rfm_df_final.Amount.quantile(0.95)
IQR = Q3 - Q1
rfm_df_final = rfm_df_final[(rfm_df_final.Amount >= Q1 - 1.5*IQR) & (rfm_df_final.Amount <= Q3 + 1.5*IQR)]

Q1 = rfm_df_final.Recency.quantile(0.05)
Q3 = rfm_df_final.Recency.quantile(0.95)
IQR = Q3 - Q1
rfm_df_final = rfm_df_final[(rfm_df_final.Recency >= Q1 - 1.5*IQR) & (rfm_df_final.Recency <= Q3 + 1.5*IQR)]

Q1 = rfm_df_final.Frequency.quantile(0.05)
Q3 = rfm_df_final.Frequency.quantile(0.95)
IQR = Q3 - Q1
rfm_df_final = rfm_df_final[(rfm_df_final.Frequency >= Q1 - 1.5*IQR) & (rfm_df_final.Frequency <= Q3 + 1.5*IQR)]
```

In [92]:
```
from sklearn.preprocessing import MinMaxScaler
```

In [93]:
```
X = rfm_df_final[['Amount', 'Frequency', 'Recency']]
scaler = MinMaxScaler()
rfm_df_scaled = scaler.fit_transform(X)
```

In [94]:
```
rfm_df_scaled = pd.DataFrame(rfm_df_scaled)
rfm_df_scaled.columns = ['Amount', 'Frequency', 'Recency']
rfm_df_scaled.head()
```

Out[94]:

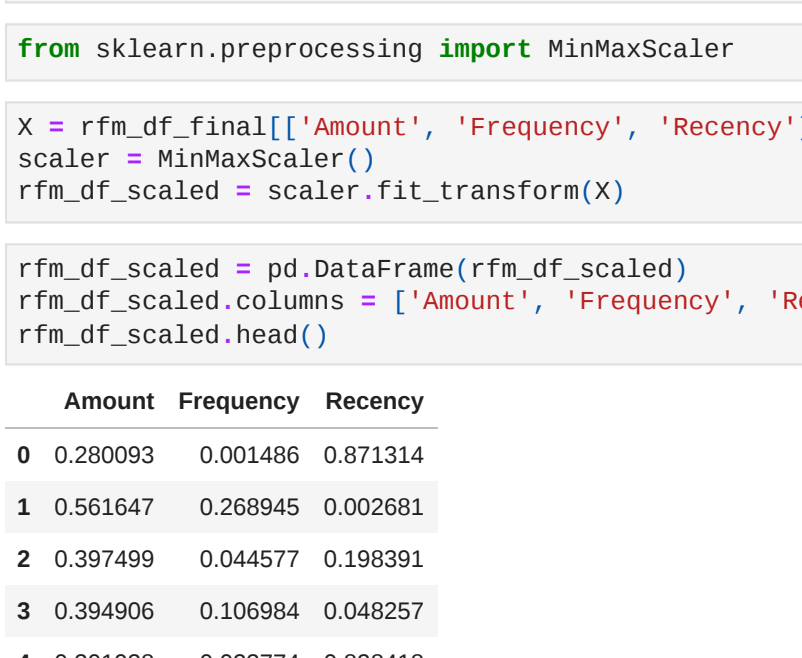
	Amount	Frequency	Recency
0	0.280093	0.001486	0.871314
1	0.561647	0.268945	0.002681
2	0.397499	0.044577	0.198391
3	0.394906	0.106984	0.048257
4	0.301938	0.023774	0.828418

In [95]:
```
##K- MEAN MODEL
kmeans = KMeans(n_clusters=3, max_iter=50)
kmeans.fit(rfm_df_scaled)
lbs = kmeans.labels_
print(kmeans.labels_)
```

```
[1 2 0 ... 1 0 0]
```

In [96]:
```
#wss
wss = []
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(rfm_df_scaled)
    wss.append(kmeans.inertia_)

plt.plot(wss)
```

Out[96]:
```
[<matplotlib.lines.Line2D at 0x2afe2aa6d90>]
```



In [97]:
```
from sklearn.metrics import silhouette_score
```

In [98]:
```
#silhouette score
range_n_clusters = [2, 3, 4, 5, 6, 7, 8]
for num_clusters in range_n_clusters:
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(rfm_df_scaled)
    cluster_labels = kmeans.labels_
    silhouette_avg = silhouette_score(rfm_df_scaled, cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, silhouette_avg))
```

```
For n_clusters=2, the silhouette score is 0.5810160466375435
For n_clusters=3, the silhouette score is 0.536645013227016
For n_clusters=4, the silhouette score is 0.4942205763829097
For n_clusters=5, the silhouette score is 0.4394396970513924
For n_clusters=6, the silhouette score is 0.39528278959209234
For n_clusters=7, the silhouette score is 0.3770609269833941
For n_clusters=8, the silhouette score is 0.3790147798294038
```

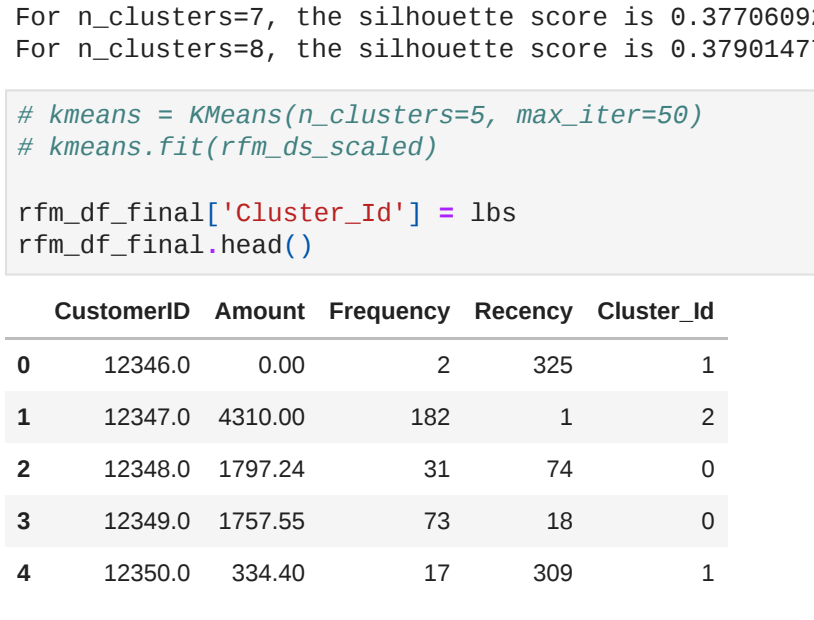
In [99]:
```
# kmeans = KMeans(n_clusters=5, max_iter=50)
# kmeans.fit(rfm_ds_scaled)

rfm_df_final['Cluster_Id'] = lbs
rfm_df_final.head()
```

Out[99]:

	CustomerID	Amount	Frequency	Recency	Cluster_Id
0	12346.0	0.00	2	325	1
1	12347.0	4310.00	182	1	2
2	12348.0	1797.24	31	74	0
3	12349.0	1757.55	73	18	0
4	12350.0	334.40	17	309	1

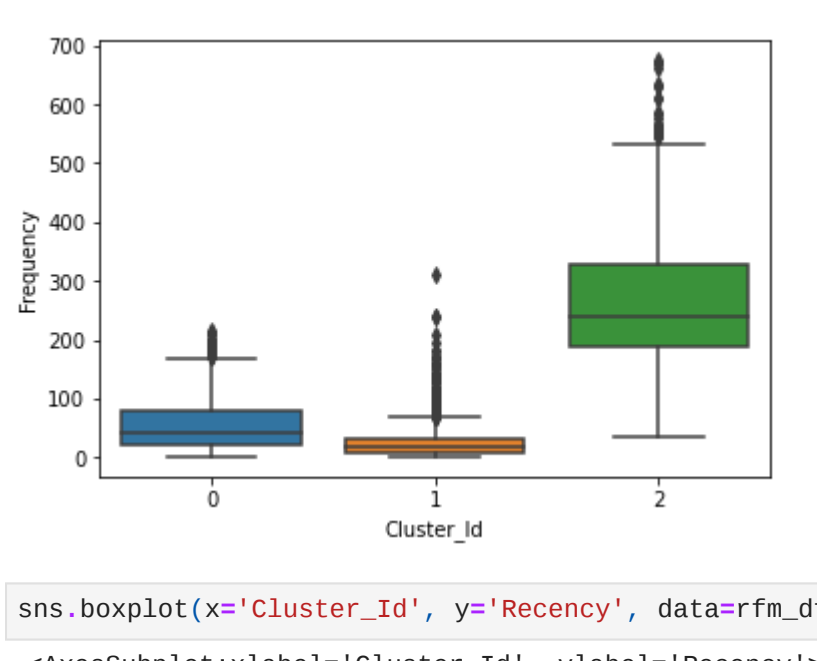
In [100…
```
sns.boxplot(x='Cluster_Id', y='Frequency', data=rfm_df_final)
```

Out[100]:
```
<AxesSubplot:xlabel='Cluster_Id', ylabel='Frequency'>
```



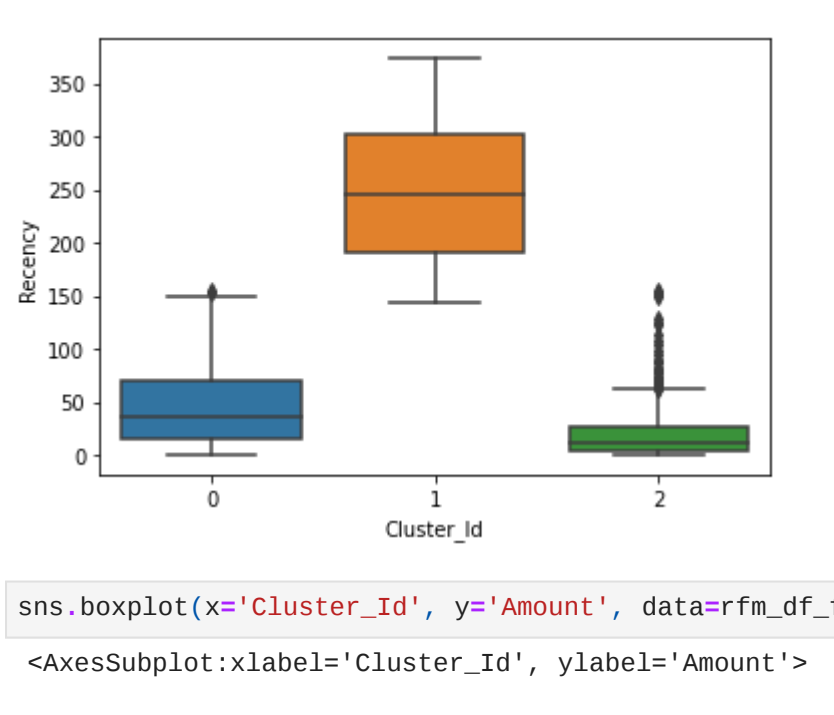
In [101…
```
sns.boxplot(x='Cluster_Id', y='Recency', data=rfm_df_final)
```

Out[101]:
```
<AxesSubplot:xlabel='Cluster_Id', ylabel='Recency'>
```



In [102…
```
sns.boxplot(x='Cluster_Id', y='Amount', data=rfm_df_final)
```

Out[102]:
```
<AxesSubplot:xlabel='Cluster_Id', ylabel='Amount'>
```



In [ ]:

In [ ]: