In [1]:
```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
dataset = pd.read_csv('E-com_Data.csv')
dataset.head()
```

Out[2]:

	CustomerID	Item Code	InvoiceNo	Date of purchase	Quantity	Time	price per Unit	Price	Shipping Location	Cancelled_status	Reason of return	Sold as set
0	4355.0	15704	396177.0	29-10-2017	6.0	3:36:00 PM	321.0	1926.0	Location 1	NaN	NaN	NaN
1	4352.0	14616	394422.0	05-10-2017	2.0	2:53:00 PM	870.0	1740.0	Location 1	NaN	NaN	NaN
2	4352.0	14614	394422.0	12-10-2017	2.0	2:53:00 PM	933.0	1866.0	Location 1	NaN	NaN	NaN
3	4352.0	85014B	388633.0	22-08-2017	3.0	2:47:00 PM	623.0	1869.0	Location 1	NaN	NaN	NaN
4	4352.0	15364	394422.0	10-10-2017	2.0	2:53:00 PM	944.0	1888.0	Location 1	NaN	NaN	NaN

In [3]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541118 entries, 0 to 541115
Data columns (total 12 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   CustomerID       404189 non-null  float64
 1   Item Code        537979 non-null  object
 2   InvoiceNo        537979 non-null  float64
 3   Date of purchase 537979 non-null  object
 4   Quantity         537979 non-null  float64
 5   Time             537979 non-null  object
 6   price per Unit   537979 non-null  float64
 7   Price            537979 non-null  float64
 8   Shipping Location 537979 non-null  object
 9   Cancelled_status  8345 non-null   object
 10  Reason of return  3 non-null      object
 11  Sold as set       0 non-null      float64
dtypes: float64(6), object(6)
memory usage: 49.5+ MB
```

In [4]:
```
# checking the duplicate rows

dataset.duplicated().sum()
```

Out[4]:
```
5145
```

In [5]:
```
# all duplicated rows from the dataset

dataset.loc[dataset.duplicated(),:]
```

Out[5]:

	CustomerID	Item Code	InvoiceNo	Date of purchase	Quantity	Time	price per Unit	Price	Shipping Location	Cancelled_status	Reason of return	Sold as set
61260	4043.0	15819	403353.0	02-12-2017	1.0	2:07:00 PM	447.0	447.0	Location 36	NaN	NaN	NaN
70587	3984.0	15422	405579.0	09-12-2017	24.0	2:09:00 PM	77.0	1848.0	Location 36	NaN	NaN	NaN
84823	3828.0	14519	398139.0	31-10-2017	1.0	2:04:00 PM	85.0	85.0	Location 36	NaN	NaN	NaN
120521	3384.0	15121	395068.0	11-06-2017	1.0	11:37:00 AM	298.0	298.0	Location 36	NaN	NaN	NaN
182786	2607.0	15660	393225.0	30-09-2017	1.0	12:31:00 PM	484.0	484.0	Location 36	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
541111	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
541112	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
541113	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
541114	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
541115	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

3145 rows × 12 columns

In [6]:
```
# dropping all the duplicates
dataset = dataset.drop_duplicates (ignore_index = True) # index starts from 0,1,2 etc
```

In [7]:
```
dataset.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
dataset.shape
```

Out[8]:
```
(537973, 12)
```

In [9]:
```
dataset.columns
```

Out[9]:
```
Index(['CustomerID', 'Item Code', 'InvoiceNo', 'Date of purchase', 'Quantity',
       'Time', 'price per Unit', 'Price', 'Shipping Location',
       'Cancelled_status', 'Reason of return', 'Sold as set'],
      dtype='object')
```

In [10]:
```
dataset = dataset[['CustomerID', 'InvoiceNo', 'Date of purchase', 'Price']]
dataset.head()
```

Out[10]:

	CustomerID	InvoiceNo	Date of purchase	Price
0	4355.0	396177.0	29-10-2017	1926.0
1	4352.0	394422.0	05-10-2017	1740.0
2	4352.0	394422.0	12-10-2017	1866.0
3	4352.0	388633.0	22-08-2017	1869.0
4	4352.0	394422.0	10-10-2017	1888.0

In [11]:
```
dataset.isnull().sum()/ len(dataset) * 100
```

Out[11]:
```
CustomerID          24.869370
InvoiceNo            0.000186
Date of purchase     0.000186
Price                0.000186
dtype: float64
```

In [12]:
```
dataset = dataset.dropna(subset=['CustomerID']) # removes all null value from specified column
```

In [13]:
```
dataset.isnull().sum()/ len(dataset) * 100
```

Out[13]:
```
CustomerID          0.0
InvoiceNo           0.0
Date of purchase    0.0
Price               0.0
dtype: float64
```

In [14]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 404181 entries, 0 to 537940
Data columns (total 4 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   CustomerID       404181 non-null  float64
 1   InvoiceNo        404181 non-null  float64
 2   Date of purchase 404181 non-null  object
 3   Price            404181 non-null  float64
dtypes: float64(3), object(1)
memory usage: 15.4 MB
```

In [15]:
```
dataset = dataset.rename( columns = {'InvoiceNo' : 'InvoiceNo', 'Date of purchase' : 'Date'})
```

In [16]:
```
dataset.head()
```

Out[16]:

	CustomerID	InvoiceNo	Date	Price
0	4355.0	396177.0	29-10-2017	1926.0
1	4352.0	394422.0	05-10-2017	1740.0
2	4352.0	394422.0	12-10-2017	1866.0
3	4352.0	388633.0	22-08-2017	1869.0
4	4352.0	394422.0	10-10-2017	1888.0

In [17]:
```
# to covert the Date datatype from object
dataset['Date'] = pd.to_datetime(dataset['Date'])
```

In [18]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 404181 entries, 0 to 537940
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   CustomerID  404181 non-null  float64
 1   InvoiceNo   404181 non-null  float64
 2   Date        404181 non-null  datetime64[ns]
 3   Price       404181 non-null  float64
dtypes: datetime64[ns](1), float64(3)
memory usage: 15.4 MB
```

In [19]:
```
dataset['CustomerID'].nunique()
```

Out[19]:
```
4349
```

In [20]:
```
dataset['Date'].describe()
```

Out[20]:
```
count                  404181
unique                    301
top       2017-11-24 00:00:00
freq                     2522
first     2016-02-12 00:00:00
last      2017-12-19 00:00:00
Name: Date, dtype: object
```

In [21]:
```
import datetime as dt
```

In [22]:
```
Latest_date = dt.datetime(2017,12,20)
Latest_date
```

Out[22]:
```
datetime.datetime(2017, 12, 20, 0, 0)
```

In [23]:
```
RFMScore = dataset.groupby('CustomerID').agg ({'Date' : lambda x : (Latest_date - x.max()).days,
                                               'InvoiceNo': lambda x : x.count(),
                                               'Price': lambda x: x.sum()})
```

In [24]:
```
# Recency is calculated as the number of days since the most recent transaction.
# Frequency is the count of transactions.
# Monetary is the sum of transaction prices.

RFMScore.rename (columns = {'Date' : 'Recency', 'InvoiceNo' : 'Frequency', 'Price' : 'Monetary'}, inplace = True)
```

In [25]:
```
RFMScore.reset_index()
```

Out[25]:

	CustomerID	Recency	Frequency	Monetary
0	2.0	4	182	553704.0
1	3.0	77	27	257404.0
2	4.0	20	72	176613.0
3	5.0	18	16	41976.0
4	6.0	9	84	151822.0
...	...	...	...	...
4344	4368.0	17	10	20480.0
4345	4369.0	181	7	10774.0
4346	4370.0	12	13	24962.0
4347	4371.0	4	754	280068.0
4348	4372.0	51	70	262820.0

4349 rows × 4 columns

In [26]:
```
RFMScore.Recency.describe()
```

Out[26]:
```
count    4349.000000
mean       61.445160
std        89.650941
min         1.000000
25%        16.000000
50%        39.000000
75%        73.000000
max       617.000000
Name: Recency, dtype: float64
```

In [27]:
```
RFMScore.Frequency.describe()
```

Out[27]:
```
count    4349.000000
mean       92.936537
std       232.080935
min         1.000000
25%        17.000000
50%        42.000000
75%       101.000000
max      7970.000000
Name: Frequency, dtype: float64
```

In [28]:
```
RFMScore.Monetary.describe()
```

Out[28]:
```
count    4.349000e+03
mean     2.299380e+05
std      8.572589e+05
min     -5.637200e+04
25%      2.814000e+04
50%      8.365500e+04
75%      2.056120e+05
max      1.153619e+07
Name: Monetary, dtype: float64
```

In [29]:
```
# splitting the data for quantile method

quantiles = RFMScore.quantile(q = [0.25, 0.50, 0.75])
quantiles = quantiles.to_dict()
quantiles
```

Out[29]:
```
{'Recency': {0.25: 16.0, 0.5: 39.0, 0.75: 73.0},
 'Frequency': {0.25: 17.0, 0.5: 42.0, 0.75: 101.0},
 'Monetary': {0.25: 28140.0, 0.5: 83655.0, 0.75: 205612.0}}
```

In [30]:
```
def RecencyScore (x,p,d):
    if x<= d[p][0.25]:
        return 1                    # 1 means recent customer under 10 days
    elif x<= d[p][0.50]:
        return 2                    # 2 means under 19 days
    elif x<= d[p][0.75]:
        return 3                    # 3 means under 73 days
    else:
        return 4
```

In [31]:
```
def FreqMonetaryScore (x,p,d):
    if x<= d[p][0.25]:
        return 4                    # silver Customer
    elif x<= d[p][0.50]:
        return 3                    # Gold Customer
    elif x<= d[p][0.75]:
        return 2                    # Diamond Customer
    else :
        return 1                    # Platinum Customer
```

In [32]:
```
RFMScore.columns
```

Out[32]:
```
Index(['Recency', 'Frequency', 'Monetary'], dtype='object')
```

In [33]:
```
RFMScore['R'] = RFMScore['Recency'].apply(RecencyScore, args = ('Recency', quantiles))
RFMScore['F'] = RFMScore['Frequency'].apply(FreqMonetaryScore, args = ('Frequency', quantiles))
RFMScore['M'] = RFMScore['Monetary'].apply(FreqMonetaryScore, args = ('Monetary', quantiles))
```

In [34]:
```
RFMScore.reset_index()
```

Out[34]:

	CustomerID	Recency	Frequency	Monetary	R	F	M
0	2.0	4	182	553704.0	1	1	1
1	3.0	77	27	257404.0	4	2	1
2	4.0	20	72	176613.0	3	2	2
3	5.0	18	16	41976.0	2	4	2
4	6.0	9	84	151822.0	1	2	2
...	...	...	...	...	...	...	...
4344	4368.0	17	10	20480.0	2	4	4
4345	4369.0	181	7	10774.0	4	4	4
4346	4370.0	12	13	24962.0	2	4	4
4347	4371.0	4	754	280068.0	1	1	1
4348	4372.0	51	70	262820.0	3	2	1

4349 rows × 7 columns

In [35]:
```
RFMScore['RFMvalue'] = RFMScore[['R','F','M']].sum(axis=1)
RFMScore['RFMGroup'] = RFMScore.R.map(str) + RFMScore.F.map(str) + RFMScore.M.map(str)
RFMScore.reset_index()
```

Out[35]:

	CustomerID	Recency	Frequency	Monetary	R	F	M	RFMvalue	RFMGroup
0	2.0	4	182	553704.0	1	1	1	3	111
1	3.0	77	27	257404.0	4	2	1	7	421
2	4.0	20	72	176613.0	3	2	2	7	322
3	5.0	18	16	41976.0	2	4	2	8	242
4	6.0	9	84	151822.0	1	2	2	5	122
...	...	...	...	...	...	...	...	...	...
4344	4368.0	17	10	20480.0	2	4	4	10	244
4345	4369.0	181	7	10774.0	4	4	4	12	444
4346	4370.0	12	13	24962.0	2	4	4	10	244
4347	4371.0	4	754	280068.0	1	1	1	3	111
4348	4372.0	51	70	262820.0	3	2	1	6	321

4349 rows × 9 columns

In [36]:
```
RFMScore['RFMvalue'].nunique()
```

Out[36]:
```
10
```

In [37]:
```
Loyalty_Level = ['Platinum','Diamond','Gold','Silver']
score_cuts = pd.qcut(RFMScore.RFMvalue, q=4, labels=Loyalty_Level)
RFMScore['Loyalty_Level'] = score_cuts.values
RFMScore.reset_index()
RFMScore
```

Out[37]:

	CustomerID	Recency	Frequency	Monetary	R	F	M	RFMvalue	RFMGroup	Loyalty_Level
0	2.0	4	182	553704.0	1	1	1	3	111	Platinum
1	3.0	77	27	257404.0	4	2	1	7	421	Diamond
2	4.0	20	72	176613.0	3	2	2	7	322	Diamond
3	5.0	18	16	41976.0	2	4	2	8	242	Gold
4	6.0	9	84	151822.0	1	2	2	5	122	Platinum
...	...	...	...	...	...	...	...	...	...	...
4344	4368.0	17	10	20480.0	2	4	4	10	244	Silver
4345	4369.0	181	7	10774.0	4	4	4	12	444	Silver
4346	4370.0	12	13	24962.0	2	4	4	10	244	Silver
4347	4371.0	4	754	280068.0	1	1	1	3	111	Platinum
4348	4372.0	51	70	262820.0	3	2	1	6	321	Diamond

4349 rows × 10 columns

In [38]:
```
RFMScore.to_csv('final_data.csv')
```

In [ ]:
```

```