

# Guided Project Report

## Association Rule Mining Market Basket Analysis

Name: Shruti Verma

Course: AI and ML

(Batch 4)

Duration: 10 months

Problem Statement: Build a relation between the products bought by the customers using store data and understand the loyalty of the customers.

### Prerequisites

What things you need to install the software and how to install them:

Python 3.8 or higher versions This setup requires that your machine has latest version of python. The following url <https://www.python.org/downloads/> can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: <https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/>. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url <https://www.anaconda.com/download/> You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.8 then run below commands in command prompt/terminal to install these packages pip install -U scikit-learn pip install numpy pip install scipy if you have chosen to install anaconda then run below commands in anaconda prompt to install these packages conda install -c scikit-learn conda install -c anaconda numpy conda install -c anaconda scipy . Importing matplotlib.pyplot, Seaborn and datetime modules. Also , installed pip install chart\_studio

### Video Link

[https://drive.google.com/file/d/1yF7Gt2BxeM40db4m\\_wHbI4vayuF9v7cH/view?usp=sharing](https://drive.google.com/file/d/1yF7Gt2BxeM40db4m_wHbI4vayuF9v7cH/view?usp=sharing)

### Dataset used

The data source is online retail data having close to 5.4 lakh entries and 8 column features. The link is (<https://archive.ics.uci.edu/ml/datasets/Online+Retail#>)

## Method used for detection

1. Loading and cleaning the data set.
2. Preprocessing the dataset
3. Applying RFM model and then clustering based on customer ids and RFM score
4. Using Elbow bend method finding the optimum number of clusters for dividing the customers into.

## Importing the libraries and capturing images:

### 1. Importing the libraries and loading the online retail dataset

The screenshot shows a Jupyter Notebook interface with the title "Capstone Project on Customer Segmentation". The notebook has several code cells and their corresponding outputs:

- In [1]:**

```
1 ## Import the necessary libraries
2
3 import numpy as np
4 import pandas as pd
5 import matplotlib.pyplot as plt
```
- In [2]:**

```
1 # reading the online retail data file
2 df = pd.read_excel('./Online Retail.xlsx')
```
- In [3]:**

```
1 # displaying the data
2 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]:**

```
1 # Data descriptive analysis
2 # there are more than 5 lakh entries and 8 feature columns
3 df.shape
```
- Out[4]:** (541909, 8)
- In [5]:**

```
1 # Names of the Columns details
2 df.columns
```
- Out[5]:** Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate', 'UnitPrice', 'CustomerID', 'Country'], dtype='object')
- In [6]:**

```
1 df.describe()
```

### 2. Checking for null values and number of items

The screenshot shows a Jupyter Notebook interface with the title "Capstone Project on Customer Segmentation - Jupyter Notebook". The notebook has several cells:

- In [6]:** `df.describe()`
 

	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
- In [7]:** `# check for null values`  
`df.isnull().sum()`
 

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	dtype
count	0	0	1454	0	0	0	135080	0	int64
- In [8]:** `# Unique Stockitems , there are 3661 item list`  
`len(df.StockCode.unique())`
 

Out[8]: 4070
- In [9]:** `# Visualizing the values in the bar graph`  
`import matplotlib.pyplot as plt`  
`# Figure Size`  
`fig = plt.figure(figsize =(10, 7))`
 

Item	Quantity
item[0:5]	6.0
item[1:5]	6.0
item[2:5]	8.0
item[3:5]	6.0
item[4:5]	6.0

## Visualising the data in bar graph

The screenshot shows a Jupyter Notebook interface with the title "Capstone Project on Customer Segmentation - Jupyter Notebook". The notebook has two cells:

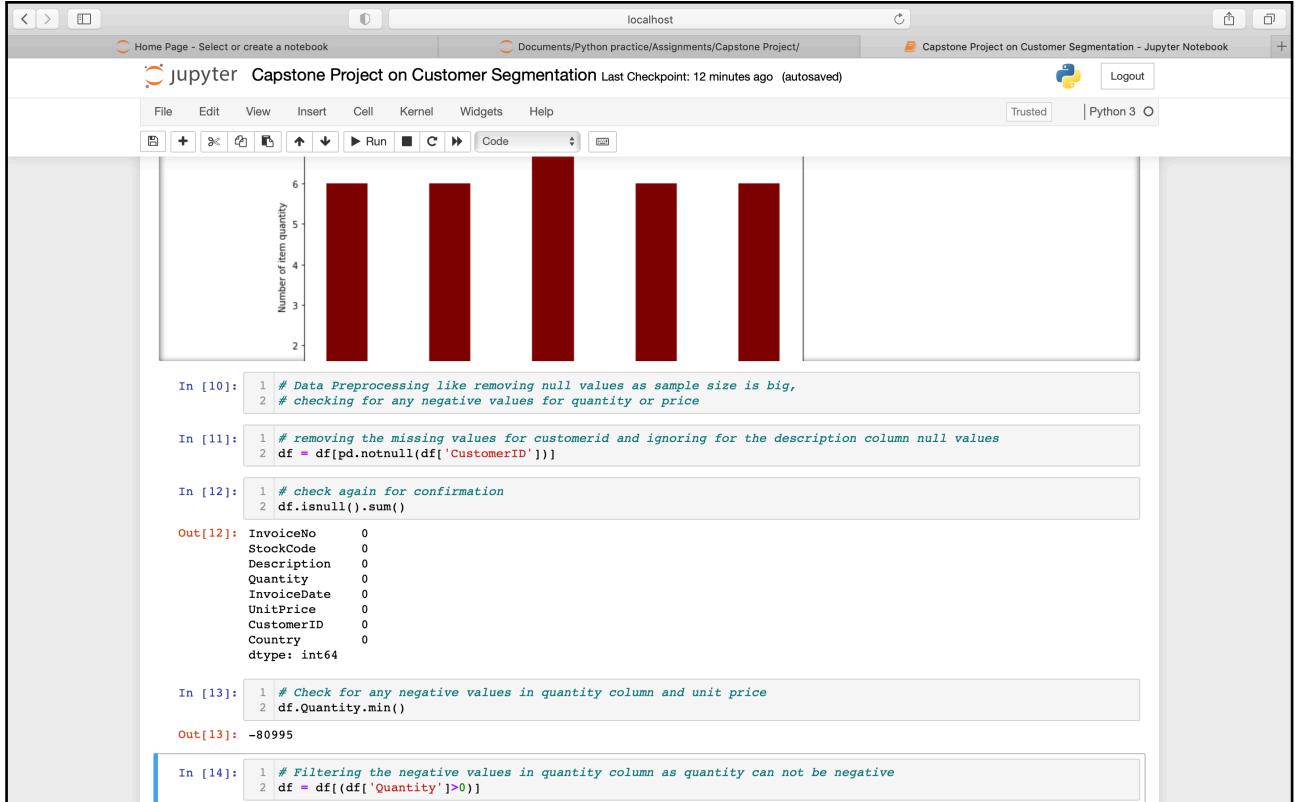
- In [8]:** `# Unique Stockitems , there are 3661 item list`  
`len(df.StockCode.unique())`
 

Out[8]: 4070
- In [9]:** `# Visualizing the values in the bar graph`  
`import matplotlib.pyplot as plt`  
`# Figure Size`  
`fig = plt.figure(figsize =(10, 7))`  
`item = df['Description'].head(5)`  
`quantity = df['Quantity'].head(5)`  
`# Horizontal Bar Plot`  
`plt.bar(item[0:5], quantity[0:5], color ='maroon', width = 0.4)`  
`plt.xlabel("Item Details")`  
`plt.ylabel("Number of item quantity")`  
`plt.title("Online Retail Data")`  
`plt.show()`
 

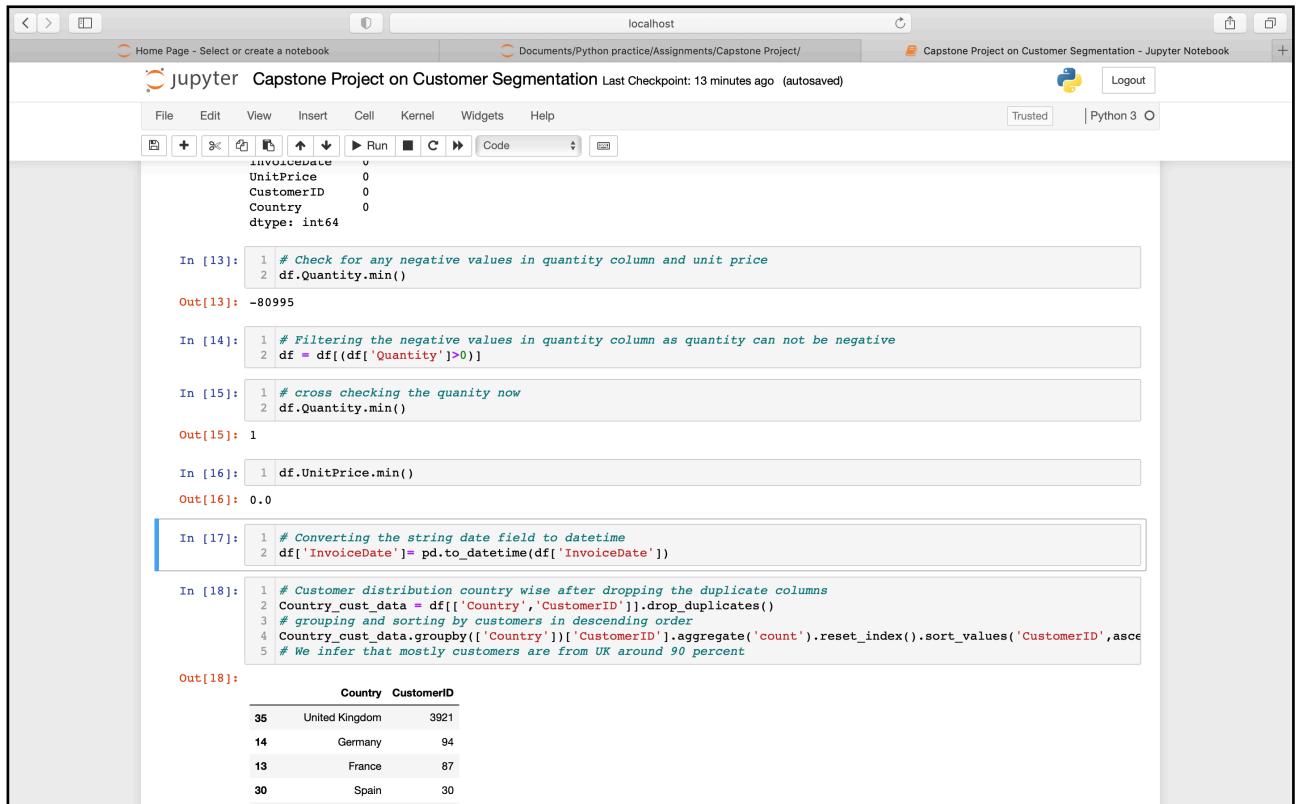
Item	Quantity
item[0:5]	6.0
item[1:5]	6.0
item[2:5]	8.0
item[3:5]	6.0
item[4:5]	6.0

## Data processing

Removing the nulls and negative values for non negative value columns in any



Fixing the datatype as date time and grouping by country



Adding Total Amount column and rechecking specifications of data

```

In [19]: 1 # Can use UK data for further calculations
2 df = df.query("Country=='United Kingdom").reset_index(drop = True)

In [20]: 1 # Unique Stockitems , there are 3661 item list
2 len(df.StockCode.unique())

Out[20]: 3645

In [21]: 1 # Adding a TotalAmt column
2 df['TotalAmt'] = df['Quantity']*df['UnitPrice']
3

In [22]: 1 # Checking for the added column TotalAmt
2 df.columns

Out[22]: Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate', 'UnitPrice', 'CustomerID', 'Country', 'TotalAmt'],
              dtype='object')

In [23]: 1 # Checking the shape post data cleaning, 9 count is due to added new column TotalAmt
2 df.shape

Out[23]: (354345, 9)

In [24]: 1 df.head()

Out[24]:
   InvoiceNo StockCode          Description  Quantity  InvoiceDate  UnitPrice  CustomerID  Country  TotalAmt
0    536365  85123A  WHITE HANGING HEART T-LIGHT HOLDER      6 2010-12-01 08:26:00     2.55    17850.0  United Kingdom    15.30
1    536365    71053        WHITE METAL LANTERN      6 2010-12-01 08:26:00     3.39    17850.0  United Kingdom    20.34
2    536365  84406B       CREAM CUPID HEARTS COAT HANGER      8 2010-12-01 08:26:00     2.75    17850.0  United Kingdom    22.00
3    536365  84029G  KNITTED UNION FLAG HOT WATER BOTTLE      6 2010-12-01 08:26:00     3.39    17850.0  United Kingdom    20.34
4    536365  84029E      RED WOOLLY HOTTIE WHITE HEART.      6 2010-12-01 08:26:00     3.39    17850.0  United Kingdom    20.34

```

## Applying RFM model

```

In [25]: 1 # Customer Segmentation using RFM Model
2 # Recency = Latest Date - Last Invoice Data, most recent purchase date
3 # Frequency = count of invoice number of transactions, no. of transactions within a period say 1 year
4 # Monetory = total or average sales attributed to a customer

In [26]: 1 import datetime as dt

In [27]: 1 # set latest date as 2011-12-10 as last purchase date was 2011-12-09.
2 #Helps to know no. of days from recent purchase
3 Latest_date = dt.datetime(2011,12,10)

In [28]: 1 # Calculating RFM modelling scores for each customer
2 RFM_score = df.groupby('CustomerID').agg({'InvoiceDate':lambda x:(Latest_date - x.max()).days,'InvoiceNo':lambda x:

In [29]: 1 RFM_score

Out[29]:
   InvoiceDate  InvoiceNo  TotalAmt
CustomerID
12346.0        325         1  77183.60
12747.0         2          103  4196.01
12748.0         0          4596  33719.73
click to unscroll output; double click to hide
12820.0        3           59  942.34
...
18280.0        277          10  180.60
18281.0        180           7  80.82
18282.0         7           12  178.05
18283.0        3           756 2094.88

```

Renaming the columns to RFM

localhost

Home Page - Select or create a notebook

Documents/Python practice/Assignments/Capstone Project/ Capstone Project on Customer Segmentation - Jupyter Notebook

Jupyter Capstone Project on Customer Segmentation Last Checkpoint: 19 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 Logout

Code

12346.0 325 1 77183.60  
12747.0 2 103 4196.01  
12748.0 0 4596 33719.73  
12749.0 3 199 4090.88  
12820.0 3 59 942.34  
... ... ... ...  
18280.0 277 10 180.60  
18281.0 180 7 80.82  
18282.0 7 12 178.05  
18283.0 3 756 2094.88

```
In [30]: 1 # Converting InvoiceDate into type int
2 RFM_score['InvoiceDate'] = RFM_score['InvoiceDate'].astype(int)

In [31]: 1 # replacing column names
2 RFM_score.rename(columns={'InvoiceDate': 'Recency',
3                           'InvoiceNo': 'Frequency',
4                           'TotalAmt': 'Monetary'}, inplace = True)

In [32]: 1 RFM_score.reset_index().head()
2 # Customer 12346 has bought almost 11 months ago maybe some sale was going on that time

Out[32]: CustomerID Recency Frequency Monetary
0 12346.0 325 1 77183.60
1 12747.0 2 103 4196.01
click to scroll output; double click to hide 4596 33719.73
3 12749.0 3 199 4090.88
4 12820.0 3 59 942.34

In [33]: 1 # Descriptive Stats about recency
2 RFM_score.Recency.describe()
3 # we can infer average is 92 days for recency and median is 50
```

## Visualization of distribution

localhost

Home Page - Select or create a notebook

Documents/Python practice/Assignments/Capstone Project/ Capstone Project on Customer Segmentation - Jupyter Notebook

Jupyter Capstone Project on Customer Segmentation Last Checkpoint: 20 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 Logout

Code

```
In [33]: 1 # Descriptive Stats about recency
2 RFM_score.Recency.describe()
3 # we can infer average is 92 days for recency and median is 50

Out[33]: count    3921.000000
mean     91.722265
std      99.528532
min      0.000000
25%     17.000000
50%     50.000000
75%     142.000000
max     373.000000
Name: Recency, dtype: float64
```

```
In [34]: 1 # Visualisation of distribution plot of data for recency
2
3 import seaborn as sns
4
5 x = RFM_score['Recency']
6
7 ax = sns.distplot(x)

/Users/ashokdayal/opt/anaconda3/envs/shru/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

localhost

Home Page - Select or create a notebook

Documents/Python practice/Assignments/Capstone Project/ Capstone Project on Customer Segmentation - Jupyter Notebook Logout

Jupyter Capstone Project on Customer Segmentation Last Checkpoint: 22 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
mean      90.371079
std       217.796155
min       1.000000
25%      17.000000
50%      41.000000
75%      99.000000
max      7847.000000
Name: Frequency, dtype: float64
```

```
In [36]: 1 x = RFM_score['Frequency']
2 ax = sns.distplot(x)
3
/Users/ashokdayal/opt/anaconda3/envs/shru/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `dis plot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
```

```
In [37]: 1 # for Monotory
2 RFM_score.Monotory.describe()
```

```
Out[37]: count    3921.000000
mean     1863.910113
std      7481.922217
min      0.000000
25%     300.040000
50%     651.620000
75%     1575.890000
max     259657.300000
Name: Monotory, dtype: float64
```

localhost

Home Page - Select or create a notebook

Documents/Python practice/Assignments/Capstone Project/ Capstone Project on Customer Segmentation - Jupyter Notebook Logout

Jupyter Capstone Project on Customer Segmentation Last Checkpoint: 24 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [37]: 1 # for Monotory
2 RFM_score.Monotory.describe()
```

```
Out[37]: count    3921.000000
mean     1863.910113
std      7481.922217
min      0.000000
25%     300.040000
50%     651.620000
75%     1575.890000
max     259657.300000
Name: Monotory, dtype: float64
```

```
In [38]: 1 x = RFM_score['Monotory']
2 ax = sns.distplot(x)
3
/Users/ashokdayal/opt/anaconda3/envs/shru/lib/python3.8/site-packages/seaborn/distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `dis plot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
warnings.warn(msg, FutureWarning)
```

```
In [39]: 1 # Splitting into four groups using quantile
2
3 quantiles = RFM_score.quantile(q=[0.25,0.5,0.75])
```

Splitting data for RFM into 4 groups



```

In [39]: 1 # Splitting into four groups using quantile
2
3 quantiles = RFM_score.quantile(q=[0.25,0.5,0.75])
4 quantiles = quantiles.to_dict()
5 quantiles
Out[39]: {'Recency': {0.25: 17.0, 0.5: 50.0, 0.75: 142.0},
'Frequency': {0.25: 17.0, 0.5: 41.0, 0.75: 99.0},
'Monetary': {0.25: 300.0399999999996, 0.5: 651.8199999999999, 0.75: 1575.89}}

```

```

In [40]: 1 # Function to create R,F,M segments
2
3 # 1 is returned for most recent
4 def Rscore(x,p,d):
5     if x <= d[p][0.25]:
6         return 1
7     elif x <= d[p][0.50]:
8         return 2
9     elif x <= d[p][0.75]:
10        return 3
11    else:
12        return 4

```

```

In [41]: 1 # Assigning 1 value to the highest frequency and monetary
2 def FandMscore(x,p,d):
3     if x <= d[p][0.25]:
4         return 4
5     elif x <= d[p][0.50]:
6         return 3
7     elif x <= d[p][0.75]:
8         return 2
9     else:
10        return 1

```

```

In [42]: 1 # Adding R,F,M values as columns to get better picture for each customer
2

```

Calculating the total score , one value indicates high score and 4 indicates low score

```

In [42]: 1 # Adding R,F,M values as columns to get better picture for each customer
2
3 RFM_score['R'] = RFM_score['Recency'].apply(Rscore,args = ('Recency',quantiles,))
4 RFM_score['F'] = RFM_score['Frequency'].apply(FandMscore,args = ('Frequency',quantiles,))
5 RFM_score['M'] = RFM_score['Monetary'].apply(FandMscore,args = ('Monetary',quantiles,))
6

```

```

In [43]: 1 RFM_score.head()
Out[43]:
   Recency  Frequency  Monetary  R  F  M
CustomerID
12346.0      325       1  77183.60  4  4  1
12747.0       2       103   4196.01  1  1  1
12748.0       0       4596   33719.73  1  1  1
12749.0       3       199   4090.88  1  1  1
12820.0       3       59    942.34  1  2  2

```

```

In [44]: 1 # get the total RFM score
2 RFM_score['RFM_score'] = RFM_score[['R','F','M']].sum(axis=1)

```

```

In [45]: 1 # Lower the RFM score, more loyal is the customer
2 RFM_score.head()
Out[45]:
   Recency  Frequency  Monetary  R  F  M  RFM_score
CustomerID
12346.0      325       1  77183.60  4  4  1       9
12747.0       2       103   4196.01  1  1  1       3
12748.0       0       4596   33719.73  1  1  1       3
12749.0       3       199   4090.88  1  1  1       3
12820.0       3       59    942.34  1  2  2       5

```

Assigning loyalty level

## Plot to check the distribution

The screenshot shows a Jupyter Notebook interface with the title "Capstone Project on Customer Segmentation - Jupyter Notebook". In the code cell (In [48]), the following Python code is displayed:

```
# plotting the graphs to see the distribution
import chart_studio as cs
import plotly.offline as po
import plotly.graph_objs as go
#Recency vs Frequency
# based on earlier graphs seen
graph = RFM_score.query("Monetary <50000 and Frequency <2000")
plot_data = [
    go.Scatter(
        x=graph.query("RFM_loyalty_level == 'Low'")['Recency'],
        y=graph.query("RFM_loyalty_level == 'Low'")['Frequency'],
        mode='markers',
        name='Low',
        marker= dict(size= 7,
                    line= dict(width=1),
                    color= 'blue',
                    opacity= 0.8
                )
    ),
    go.Scatter(
        x=graph.query("RFM_loyalty_level == 'Medium'")['Recency'],
        y=graph.query("RFM_loyalty_level == 'Medium'")['Frequency'],
        mode='markers',
        name='Medium',
        marker= dict(size= 9,
                    line= dict(width=1),
                    color= 'green',
                    opacity= 0.5
                )
    ),
    go.Scatter(
        x=graph.query("RFM_loyalty_level == 'High'" )['Recency'],
        y=graph.query("RFM_loyalty_level == 'High'" )['Frequency'],
        mode='markers',
        name='High',
        marker= dict(size= 11,
                    line= dict(width=1),
                    color= 'red',
                    opacity= 0.9
                )
),
]
```

The screenshot shows a Jupyter Notebook interface with the title "Capstone Project on Customer Segmentation - Jupyter Notebook". In the code cell (In [46]), the following Python code is displayed:

```
# Assigning Loyalty level to each customer
loyalty_level = ['High', 'Medium', 'Low']
score_cuts = pd.qcut(RFM_score.RFM_score, q=3, labels = loyalty_level)
RFM_score['RFM_loyalty_level'] = score_cuts.values
RFM_score.reset_index().head()
```

In the output cell (Out[46]), the resulting DataFrame is shown:

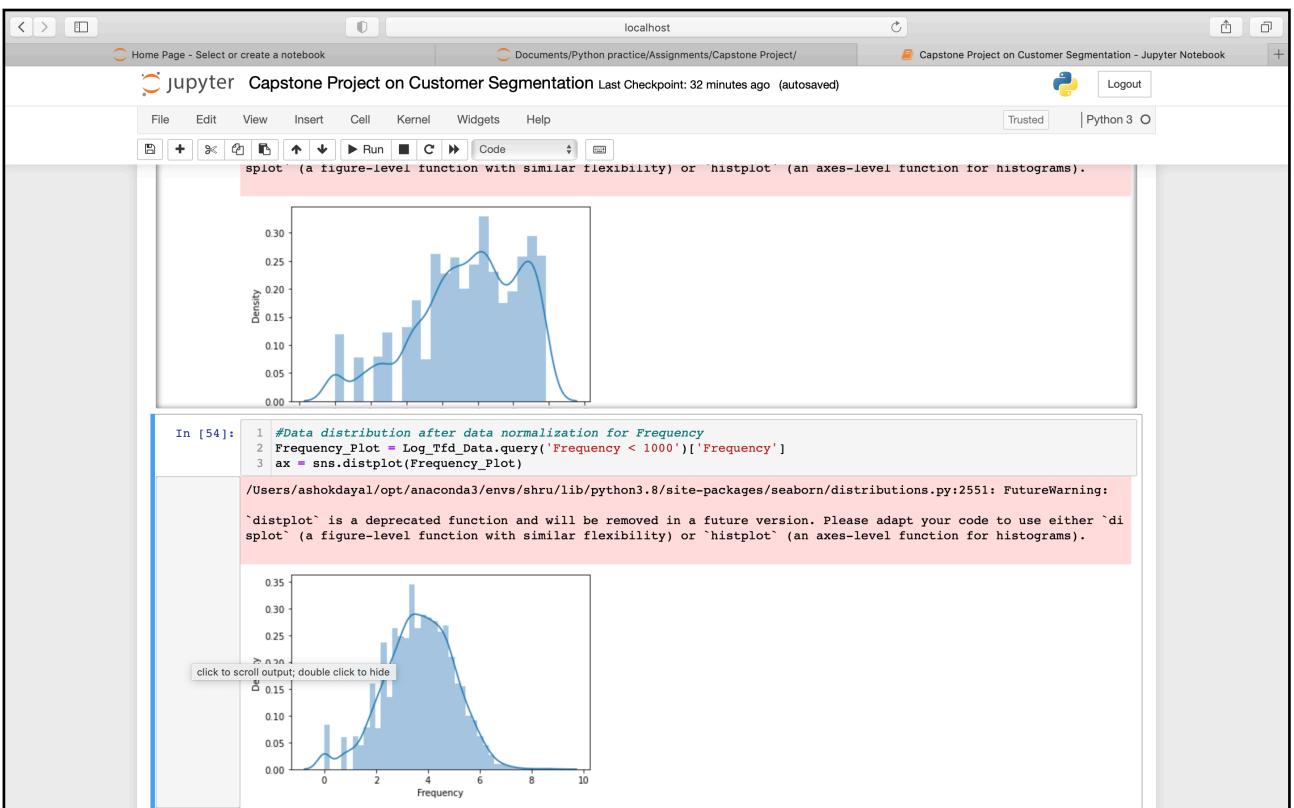
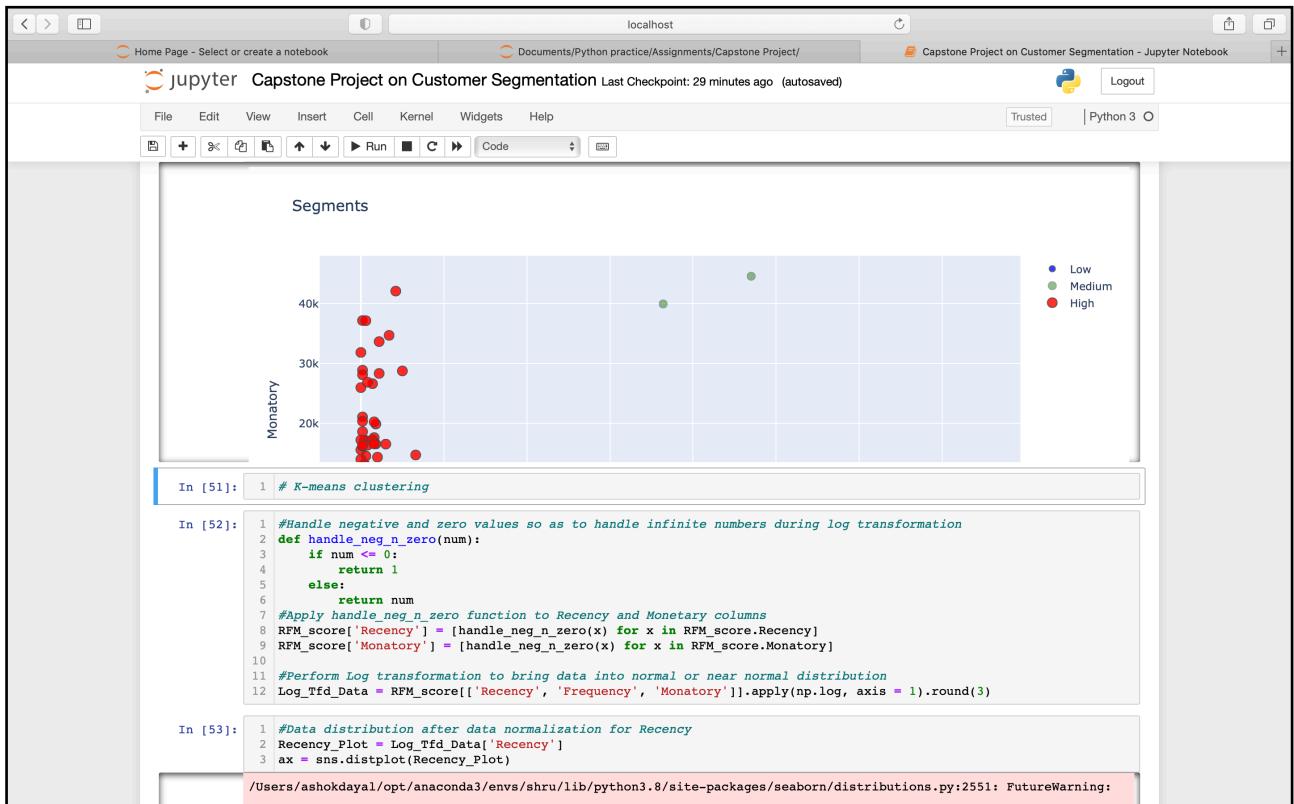
	CustomerID	Recency	Frequency	Monetary	R	F	M	RFM_score	RFM_loyalty_level
0	12346.0	325	1	77183.60	4	4	1	9	Medium
1	12747.0	2	103	4196.01	1	1	1	3	High
2	12748.0	0	4596	33719.73	1	1	1	3	High
3	12749.0	3	199	4090.88	1	1	1	3	High
4	12820.0	3	59	942.34	1	2	2	5	High

In the code cell (In [47]), the command `pip install chart_studio` is run, and the output shows that all dependencies are already satisfied.

In the code cell (In [48]), the following Python code is displayed:

```
# plotting the graphs to see the distribution
```

## Applying K means clustering



Jupyter Capstone Project on Customer Segmentation Last Checkpoint: 33 minutes ago (autosaved)

```

In [56]: 1 from sklearn.preprocessing import StandardScaler
2
3 #Bring the data on same scale
4 scaleobj = StandardScaler()
5 Scaled_Data = scaleobj.fit_transform(Log_Tfd_Data)

In [57]: 1 #Transform it back to dataframe
2 Scaled_Data = pd.DataFrame(Scaled_Data, index = RFM_score.index, columns = Log_Tfd_Data.columns)

In [58]: 1 # Applying K- means clustering
2
3 from sklearn.cluster import KMeans
4 # sum of square distance is distance of each point to the center of the cluster
5 # Checking for K ranging from 1 to 15, as K increase sum of sq distance tends to zero
6 sum_of_sq_dist = {}
7 for k in range(1,15):
8     km = KMeans(n_clusters= k, init= 'k-means++', max_iter= 1000)
9     km = km.fit(Scaled_Data)
10    sum_of_sq_dist[k] = km.inertia_
11
12 #Plot the graph for the sum of square distance values and Number of Clusters
13 sns.pointplot(x = list(sum_of_sq_dist.keys()), y = list(sum_of_sq_dist.values()))
14 plt.xlabel('Number of Clusters(k)')

```

Jupyter Capstone Project on Customer Segmentation Last Checkpoint: 35 minutes ago (autosaved)

```

In [59]: 1 # based on elbow bend method K - means cluster value is 3 in this case

In [61]: 1 #Perform K-Mean Clustering or build the K-Means clustering model
2 KMean_clust = KMeans(n_clusters= 3, init= 'k-means++', max_iter= 1000)
3 KMean_clust.fit(Scaled_Data)
4
5 #Find the clusters for the observation given in the dataset
6 RFM_score['Cluster'] = KMean_clust.labels_
7 RFM_score.head(10)

Out[61]:


| CustomerID | Recency | Frequency | Monetary | R | F | M | RFM_score | RFM_loyalty_level | Cluster |
|------------|---------|-----------|----------|---|---|---|-----------|-------------------|---------|
| 12346.0    | 325     | 1         | 77183.60 | 4 | 4 | 1 | 9         | Medium            | 2       |
| 12747.0    | 2       | 103       | 4196.01  | 1 | 1 | 1 | 3         | High              | 0       |
| 12748.0    | 1       | 4596      | 33719.73 | 1 | 1 | 1 | 3         | High              | 0       |
| 12749.0    | 3       | 199       | 4090.88  | 1 | 1 | 1 | 3         | High              | 0       |
| 12820.0    | 3       | 59        | 942.34   | 1 | 2 | 2 | 5         | High              | 0       |


```

Plot the graph to see the loyal customer distribution

Jupyter Capstone Project on Customer Segmentation Last Checkpoint: 36 minutes ago (autosaved)

```
In [62]: 1 # Visualize and plot the data
2
3 from matplotlib import pyplot as plt
4 plt.figure(figsize=(7,7))
5
6 ##Scatter Plot Frequency Vs Recency
7 Colors = ["red", "green", "blue"]
8 RFM_score['Color'] = RFM_score['Cluster'].map(lambda p: Colors[p])
9 ax = RFM_score.plot(
10     kind="scatter",
11     x="Recency", y="Frequency",
12     figsize=(10,8),
13     c = RFM_score['Color']
14 )
```

```
In [63]: 1 RFM_score.head(20)
2 # hence colors assigned are blue is for medium , green for high and red for low loyalty level
```

Jupyter Capstone Project on Customer Segmentation Last Checkpoint: 37 minutes ago (autosaved)

```
In [63]: 1 RFM_score.head(20)
2 # hence colors assigned are blue is for medium , green for high and red for low loyalty level
```

```
Out[63]:
CustomerID  Recency  Frequency  Monatory  R  F  M  RFM_score  RFM_loyalty_level  Cluster  Color
12346.0      325       1  77183.60  4  4  1          9    Medium   2  blue
12747.0      2        103  4196.01  1  1  1          3    High    0  red
12748.0      1        4596  33719.73  1  1  1          3    High    0  red
12749.0      3        199  4090.88  1  1  1          3    High    0  red
12820.0      3        59   942.34  1  2  2          5    High    0  red
12821.0     214       6    92.72  4  4  4         12    Low     1  green
12822.0      70        46  948.88  3  2  2          7  Medium   2  blue
12823.0      74        5   1759.50  3  4  1          8  Medium   1  green
12824.0      59        25   397.12  3  3  3          9  Medium   2  blue
12826.0      2        91   1474.72  1  2  2          5    High    0  red
12827.0      5        25   430.15  1  3  3          7  Medium   2  blue
12828.0      2        56   1018.71  1  2  2          5    High    0  red
12829.0     336       11   293.00  4  4  4         12    Low     1  green
12830.0      37        38   6814.64  2  3  1          6    High    2  blue
12831.0     262       9   215.05  4  4  4         12    Low     1  green
12832.0      32        27   383.03  2  3  3          8  Medium   2  blue
12833.0     145       24   417.38  4  3  3         10    Low     1  green
12834.0     282       18   312.38  4  3  3         10    Low     1  green
12836.0      59        175  2612.86  3  1  1          5    High    2  blue
12837.0     173       12   134.10  4  4  4         12    Low     1  green
```

```
In [64]: 1 KMean_clust.labels_
```

