Capstone Project

Customer Segmentation

Name : Dhivakar.R

Course: AI and ML(Aug2020)

Problem Statement: Grouping customers into sections based on their common characteristics is called Customer Segmentation. These clusters allow the companies to target the customers with the correct marketing message and tailor their offers for a specific group. This not only helps them boost their sales, but also helps them build customer relations and understand them in a better way.

In this project, our aim will be to perform customer segmentation on Online Retail Dataset (https://archive.ics.uci.edu/ml/datasets/Online+Retail#) to understand the customers. Given this dataset, our task is to:

a) Load the dataset and perform a descriptive analysis on it (Total number of entries, the column types, unique/non-null entries for each attribute, unique stock items, visualizing various attributes using bar charts/piecharts and so on).

b) Perform data cleaning. Specifically, given the dataset, handle the entries that either have missing information or have attribute values that are not feasible such as negative quantity.

c) Perform data pre-processing for the required attribute fields

d) Since this database has no additional attribute information for the customer, we will use RFM model (refer: https://clevertap.com/blog/rfmanalysis/) for segmentation. Modify the database to include RFM model attributes.

e) Now once you have your database ready, perform data clustering on this dataset by assuming a fixed number of clusters.

f) Find the optimal number of clusters that the customers can be divided into.

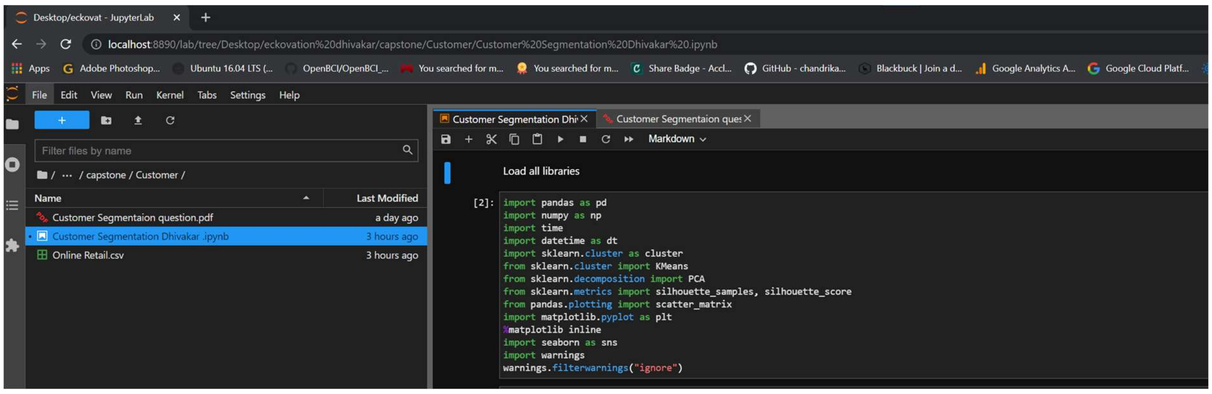
Prerequisites

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

Python 3.6 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-externalcommand/ . Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic.

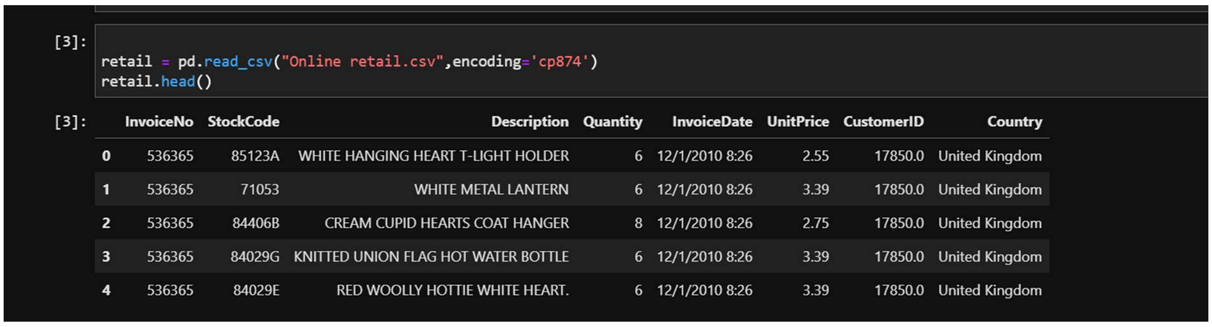
Second and easier option is to download anaconda and use its anaconda prompt to run the commands. To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 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 the libraries and loading dataset :



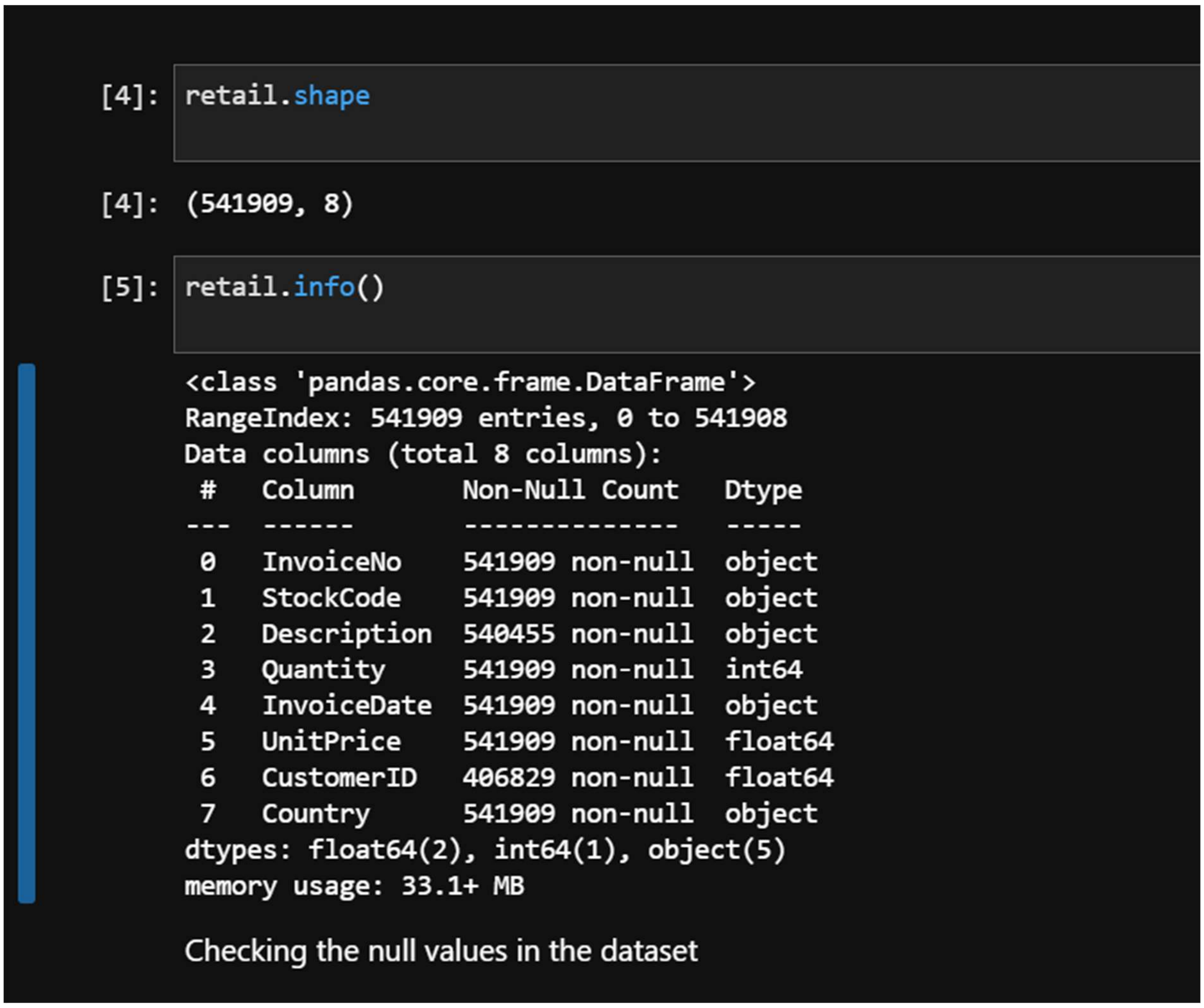
Load all libraries

```
[2]: import pandas as pd
     import numpy as np
     import time
     import datetime as dt
     import sklearn.cluster as cluster
     from sklearn.cluster import KMeans
     from sklearn.decomposition import PCA
     from sklearn.metrics import silhouette_samples, silhouette_score
     from pandas.plotting import scatter_matrix
     import matplotlib.pyplot as plt
     %matplotlib inline
     import seaborn as sns
     import warnings
     warnings.filterwarnings("ignore")
```



```
[3]: retail = pd.read_csv("Online retail.csv",encoding='cp874')
     retail.head()
```

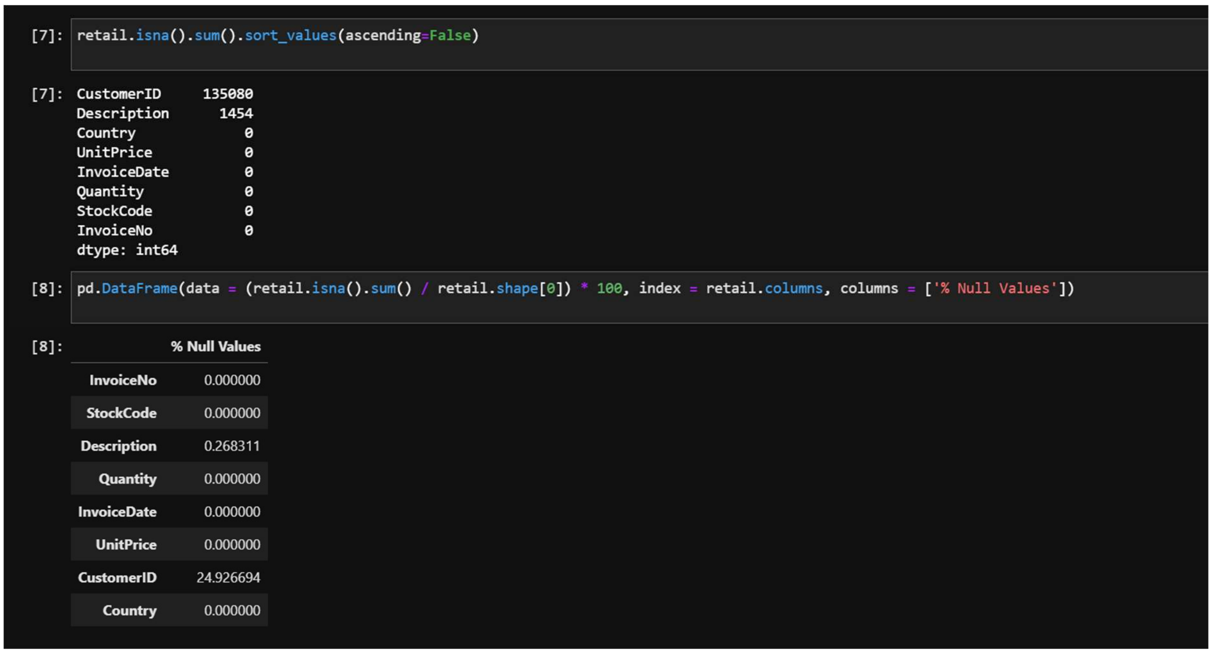
```
[3]:    InvoiceNo  StockCode                         Description  Quantity   InvoiceDate  UnitPrice  CustomerID         Country
     0     536365     85123A  WHITE HANGING HEART T-LIGHT HOLDER         6  12/1/2010 8:26       2.55     17850.0  United Kingdom
     1     536365      71053                 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
```



```
[4]: retail.shape

[4]: (541909, 8)

[5]: retail.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
```

Checking the null values in the dataset

Checking the null values :

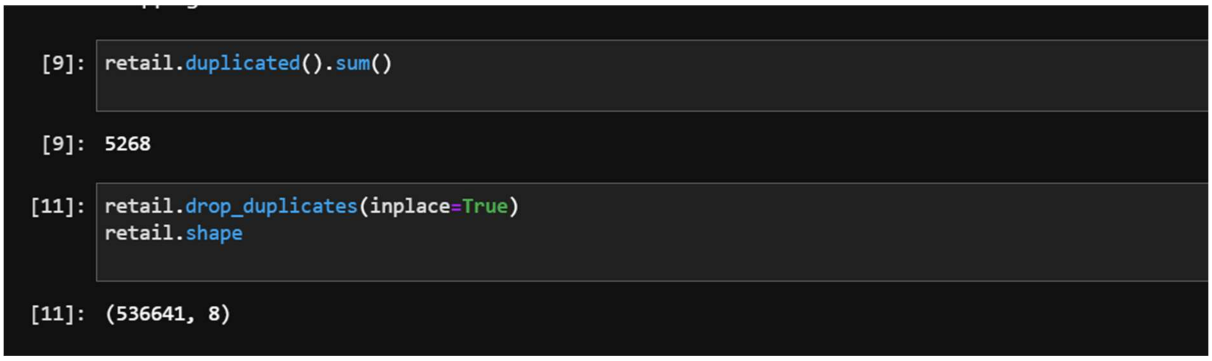


```
[7]: retail.isna().sum().sort_values(ascending=False)

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

[8]: pd.DataFrame(data = (retail.isna().sum() / retail.shape[0]) * 100, index = retail.columns, columns = ['% Null Values'])
```

[8]:

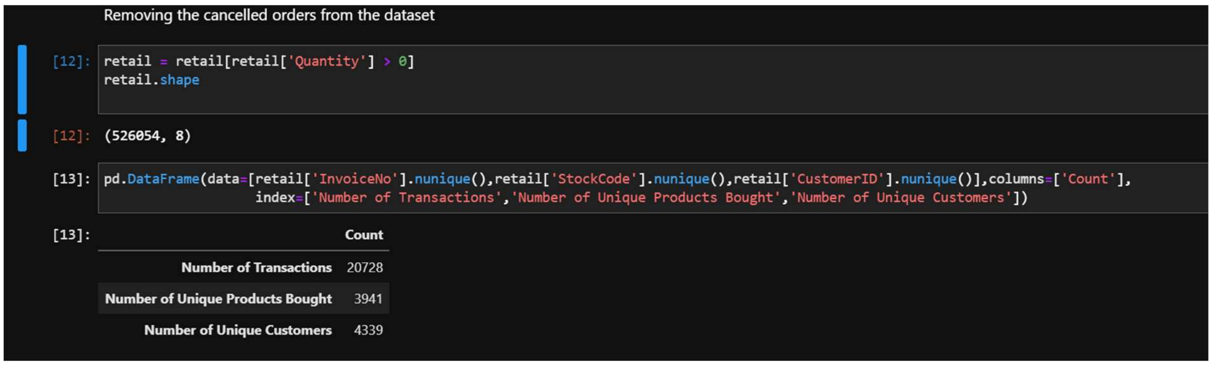
	% Null Values
InvoiceNo	0.000000
StockCode	0.000000
Description	0.268311
Quantity	0.000000
InvoiceDate	0.000000
UnitPrice	0.000000
CustomerID	24.926694
Country	0.000000

Dropping the rows with null values in customer id column



```
[9]:  retail.duplicated().sum()

[9]:  5268

[11]: retail.drop_duplicates(inplace=True)
      retail.shape

[11]: (536641, 8)
```

Removing the cancelled orders from the dataset



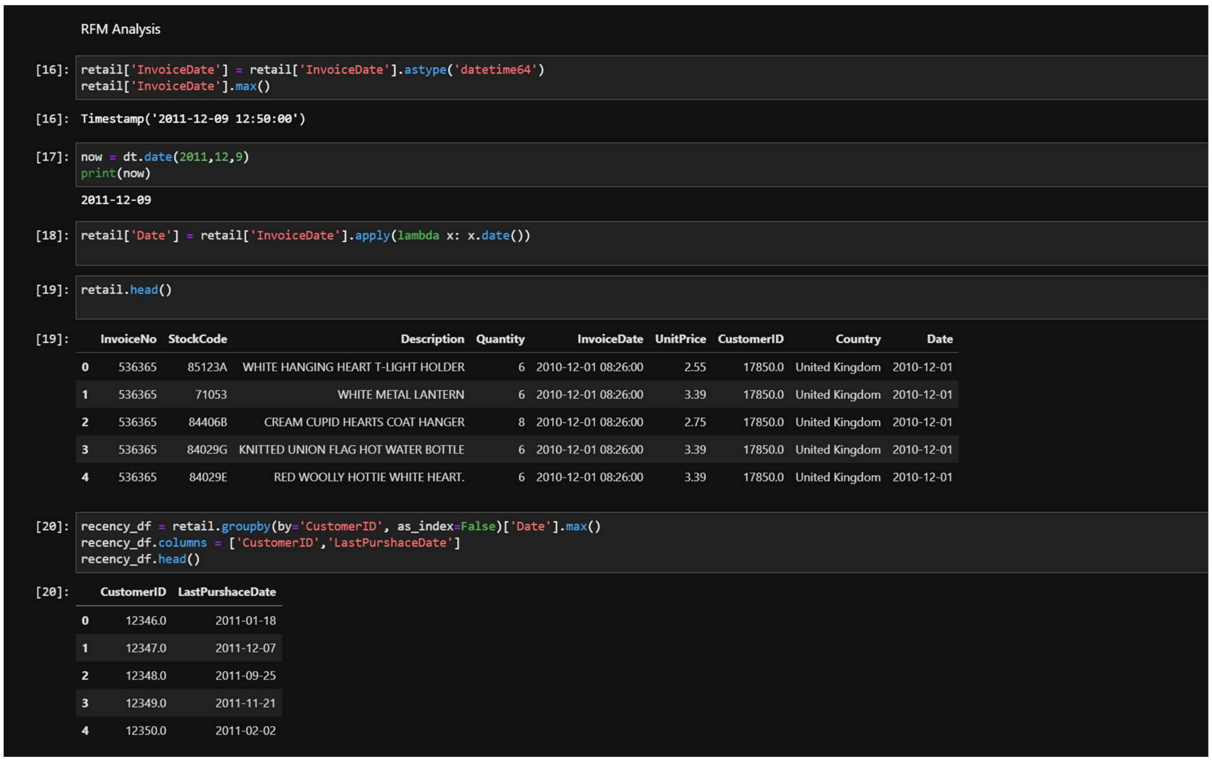
Removing the cancelled orders from the dataset

```
[12]: retail = retail[retail['Quantity'] > 0]
      retail.shape

[12]: (526054, 8)

[13]: pd.DataFrame(data=[retail['InvoiceNo'].nunique(),retail['StockCode'].nunique(),retail['CustomerID'].nunique()],columns=['Count'],
                   index=['Number of Transactions','Number of Unique Products Bought','Number of Unique Customers'])
```

[13]:

	Count
Number of Transactions	20728
Number of Unique Products Bought	3941
Number of Unique Customers	4339

RFM Analysis



RFM Analysis

```
[16]: retail['InvoiceDate'] = retail['InvoiceDate'].astype('datetime64')
      retail['InvoiceDate'].max()

[16]: Timestamp('2011-12-09 12:50:00')

[17]: now = dt.date(2011,12,9)
      print(now)

      2011-12-09

[18]: retail['Date'] = retail['InvoiceDate'].apply(lambda x: x.date())

[19]: retail.head()
```

[19]:

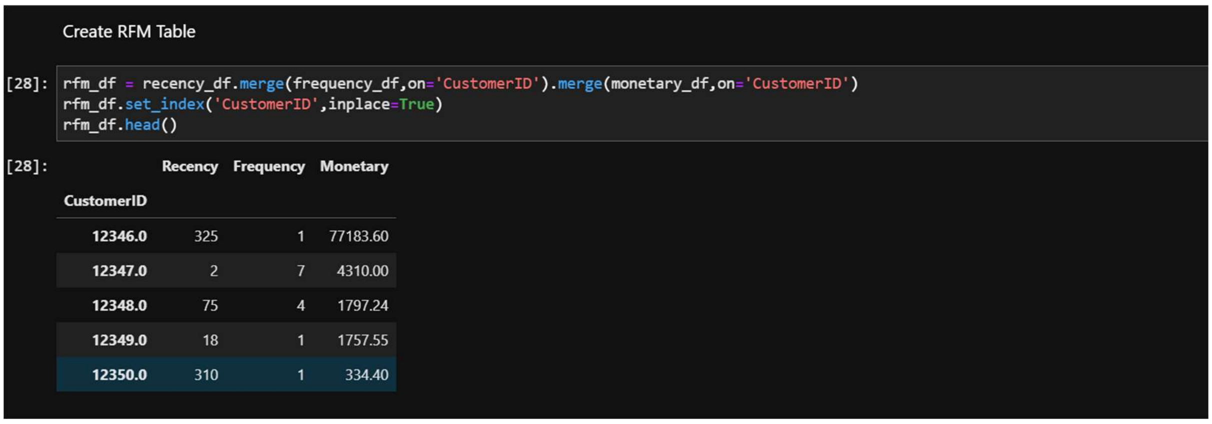
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Date
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	2010-12-01
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	2010-12-01
2	536365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

```
[20]: recency_df = retail.groupby(by='CustomerID', as_index=False)['Date'].max()
      recency_df.columns = ['CustomerID','LastPurshaceDate']
      recency_df.head()
```

[20]:

	CustomerID	LastPurshaceDate
0	12346.0	2011-01-18
1	12347.0	2011-12-07
2	12348.0	2011-09-25
3	12349.0	2011-11-21
4	12350.0	2011-02-02

RFM Table:



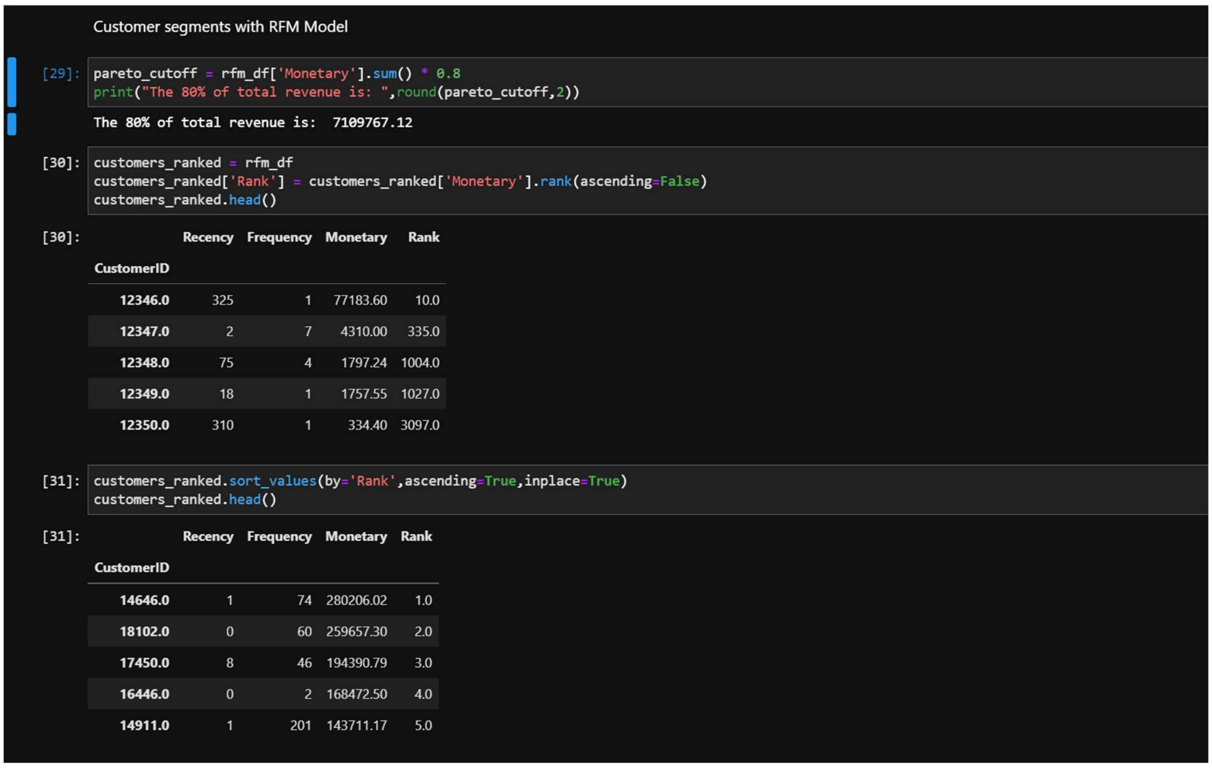
Create RFM Table

```
[28]: rfm_df = recency_df.merge(frequency_df,on='CustomerID').merge(monetary_df,on='CustomerID')
      rfm_df.set_index('CustomerID',inplace=True)
      rfm_df.head()
```

[28]:

CustomerID	Recency	Frequency	Monetary
12346.0	325	1	77183.60
12347.0	2	7	4310.00
12348.0	75	4	1797.24
12349.0	18	1	1757.55
12350.0	310	1	334.40

Customer segments with RFM Model



Customer segments with RFM Model

```
[29]: pareto_cutoff = rfm_df['Monetary'].sum() * 0.8
      print("The 80% of total revenue is: ",round(pareto_cutoff,2))
```

```
The 80% of total revenue is:  7109767.12
```

```
[30]: customers_ranked = rfm_df
      customers_ranked['Rank'] = customers_ranked['Monetary'].rank(ascending=False)
      customers_ranked.head()
```

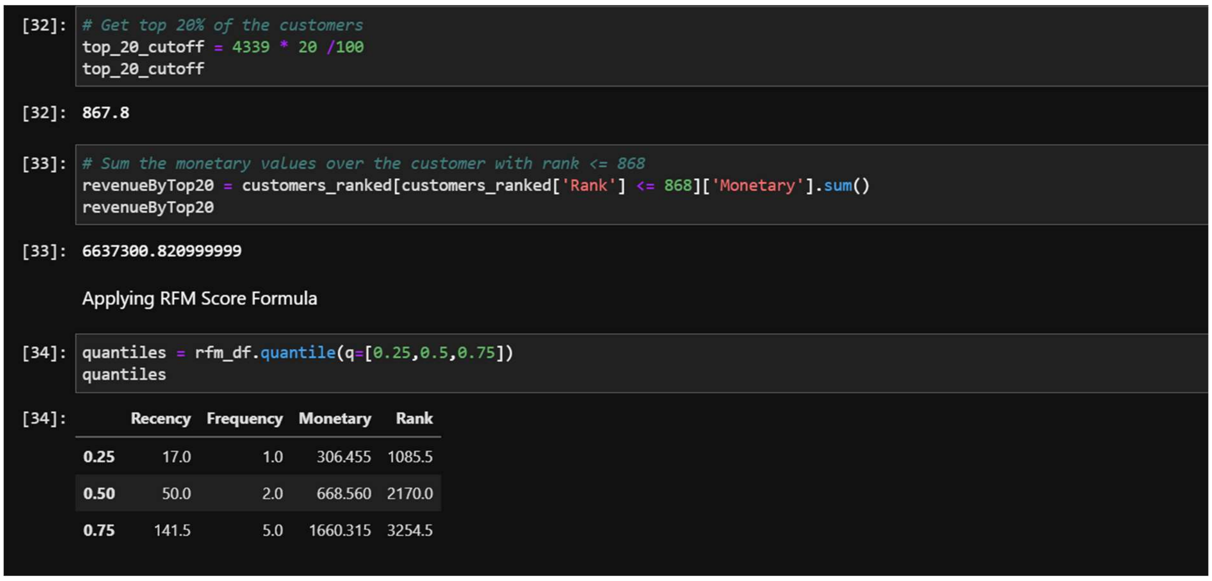
[30]:

CustomerID	Recency	Frequency	Monetary	Rank
12346.0	325	1	77183.60	10.0
12347.0	2	7	4310.00	335.0
12348.0	75	4	1797.24	1004.0
12349.0	18	1	1757.55	1027.0
12350.0	310	1	334.40	3097.0

```
[31]: customers_ranked.sort_values(by='Rank',ascending=True,inplace=True)
      customers_ranked.head()
```

[31]:

CustomerID	Recency	Frequency	Monetary	Rank
14646.0	1	74	280206.02	1.0
18102.0	0	60	259657.30	2.0
17450.0	8	46	194390.79	3.0
16446.0	0	2	168472.50	4.0
14911.0	1	201	143711.17	5.0



```
[32]: # Get top 20% of the customers
      top_20_cutoff = 4339 * 20 /100
      top_20_cutoff
```

```
[32]: 867.8
```

```
[33]: # Sum the monetary values over the customer with rank <= 868
      revenueByTop20 = customers_ranked[customers_ranked['Rank'] <= 868]['Monetary'].sum()
      revenueByTop20
```

```
[33]: 6637300.820999999
```

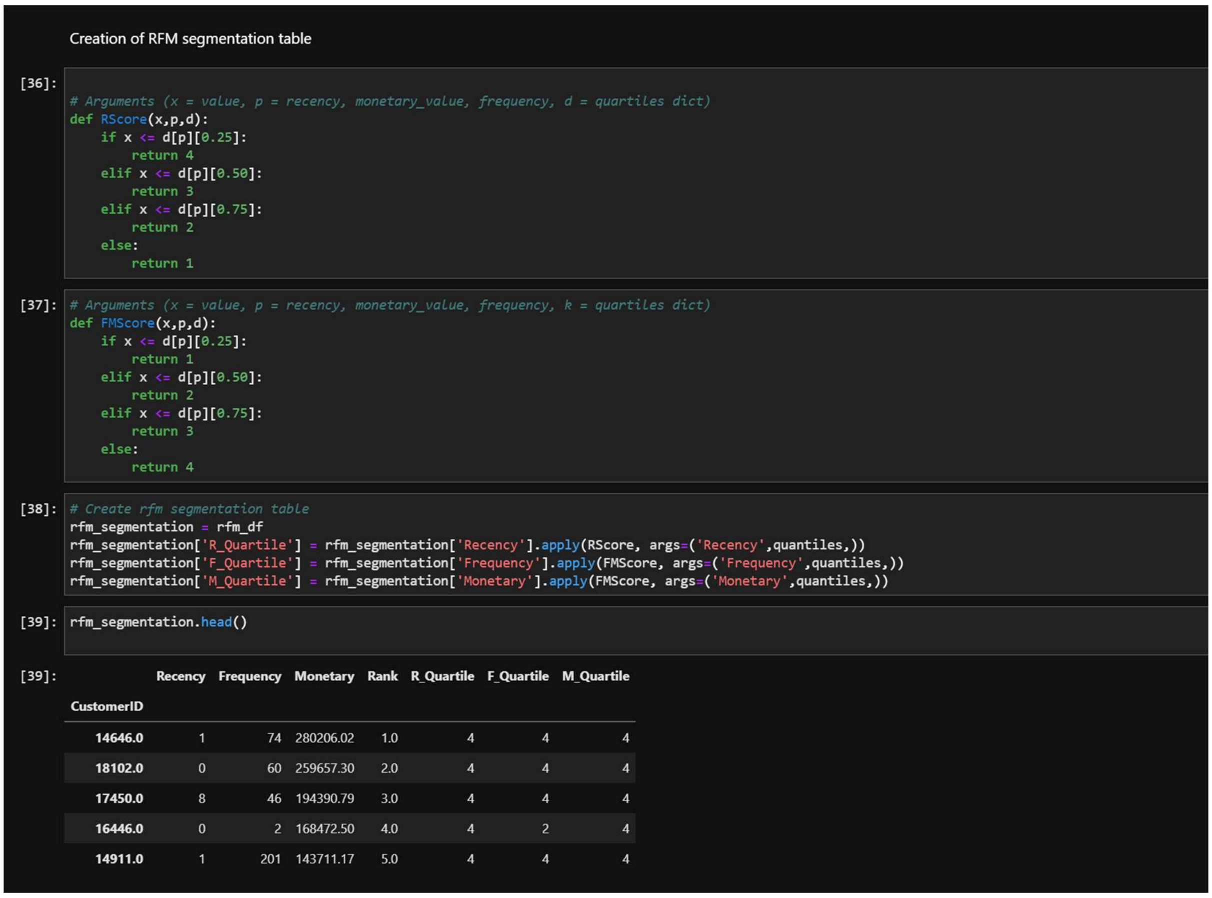
Applying RFM Score Formula

```
[34]: quantiles = rfm_df.quantile(q=[0.25,0.5,0.75])
      quantiles
```

[34]:

	Recency	Frequency	Monetary	Rank
0.25	17.0	1.0	306.455	1085.5
0.50	50.0	2.0	668.560	2170.0
0.75	141.5	5.0	1660.315	3254.5

Creation of RFM Segmentation Table:



Creation of RFM segmentation table

```
[36]:
# Arguments (x = value, p = recency, monetary_value, frequency, d = quartiles dict)
def RScore(x,p,d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1
```

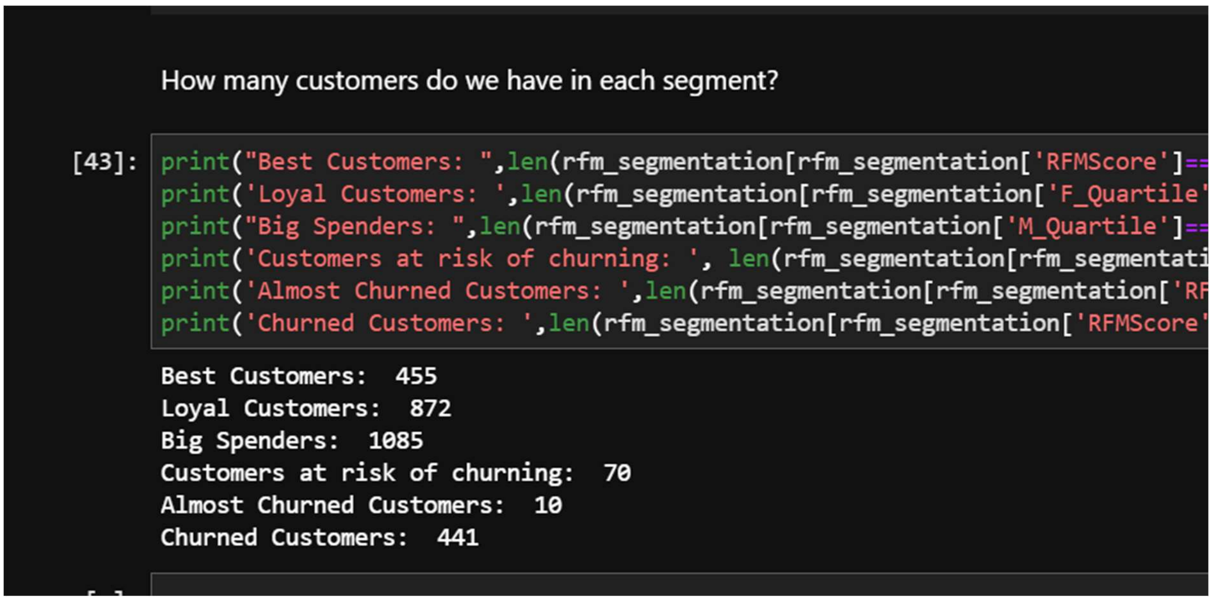
```
[37]: # Arguments (x = value, p = recency, monetary_value, frequency, k = quartiles dict)
def FMScore(x,p,d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4
```

```
[38]: # Create rfm segmentation table
rfm_segmentation = rfm_df
rfm_segmentation['R_Quartile'] = rfm_segmentation['Recency'].apply(RScore, args=('Recency',quantiles,))
rfm_segmentation['F_Quartile'] = rfm_segmentation['Frequency'].apply(FMScore, args=('Frequency',quantiles,))
rfm_segmentation['M_Quartile'] = rfm_segmentation['Monetary'].apply(FMScore, args=('Monetary',quantiles,))
```

```
[39]: rfm_segmentation.head()
```

[39]:

	Recency	Frequency	Monetary	Rank	R_Quartile	F_Quartile	M_Quartile
CustomerID							
14646.0	1	74	280206.02	1.0	4	4	4
18102.0	0	60	259657.30	2.0	4	4	4
17450.0	8	46	194390.79	3.0	4	4	4
16446.0	0	2	168472.50	4.0	4	2	4
14911.0	1	201	143711.17	5.0	4	4	4



How many customers do we have in each segment?

```
[43]: print("Best Customers: ",len(rfm_segmentation[rfm_segmentation['RFMScore']==
      print('Loyal Customers: ',len(rfm_segmentation[rfm_segmentation['F_Quartile'
      print("Big Spenders: ",len(rfm_segmentation[rfm_segmentation['M_Quartile']==
      print('Customers at risk of churning: ', len(rfm_segmentation[rfm_segmentati
      print('Almost Churned Customers: ',len(rfm_segmentation[rfm_segmentation['RF
      print('Churned Customers: ',len(rfm_segmentation[rfm_segmentation['RFMScore'
```

```
Best Customers:  455
Loyal Customers:  872
Big Spenders:  1085
Customers at risk of churning:  70
Almost Churned Customers:  10
Churned Customers:  441
```