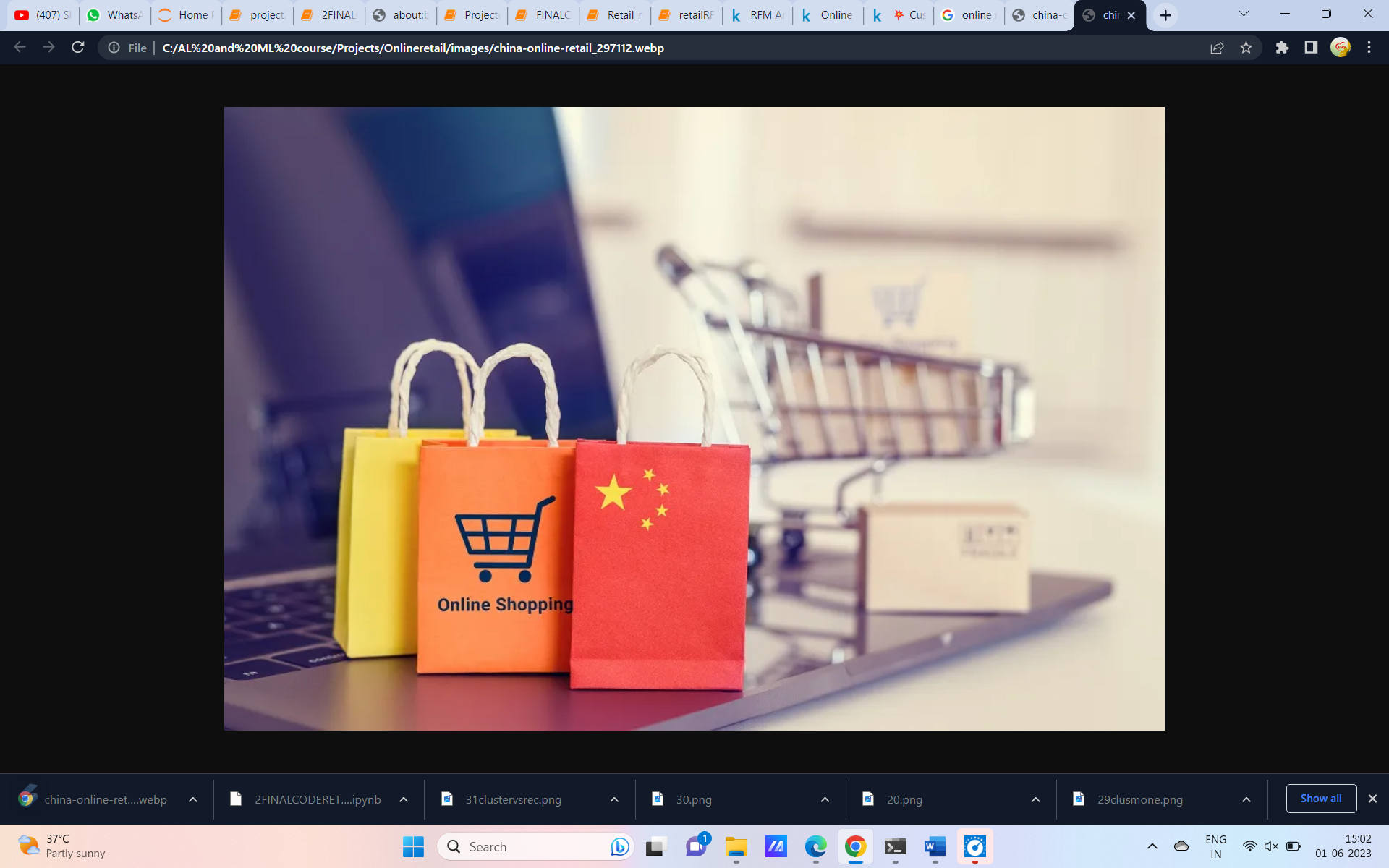
CAPSTONE PROJECT – II

ONLINE RETAIL SALES



Submitted by

SUBASREE.M.R

**Table of Contents**

1. Problem Statement

2. Project Objective

3. Data Description

4. Data Pre-processing Steps and Inspiration

5. Choosing the Algorithm for the Project

6. Motivation and Reasons For Choosing the Algorithm

7. Model Building

8. Model Evaluation and Techniques

9. Inferences from the Same

10. Conclusion

11. References

**Chapter 1 - Problem Statement**

  An online retail store is trying to understand the various customer purchase patterns for their firm, you are required to give enough evidence based insights to provide the same.

**The company wants to segment its customers and determine marketing strategies according to these segments. To this end, we will define the behavior of customers and create groups according to clusters in these behaviors. In other words, we will include those who exhibit common behaviors in the same groups and we will try to develop special sales and marketing techniques for these groups.**

**Chapter 2- Project Objective**

* Understand the dataset and features
* Use suitable Data Preprocessing and Feature Selection/Engineering Methods
* **we will define the behavior of customers and create groups according to clusters behaviors**

**Chapter 3- Data Description**

The dataset used in this project is online retail sales. The dataset contains the following columns

**InvoiceNo:** Invoice number. Nominal. A 6-digit integral number uniquely assigned to each transaction. If this code starts with the 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:** Invice date and time. Numeric. The day and time when a transaction was generated.

**UnitPrice:** Unit price. Numeric. Product price per unit in sterling (Â£).

**CustomerID**: Customer number. Nominal. A 5-digit integral number uniquely assigned to each customer.

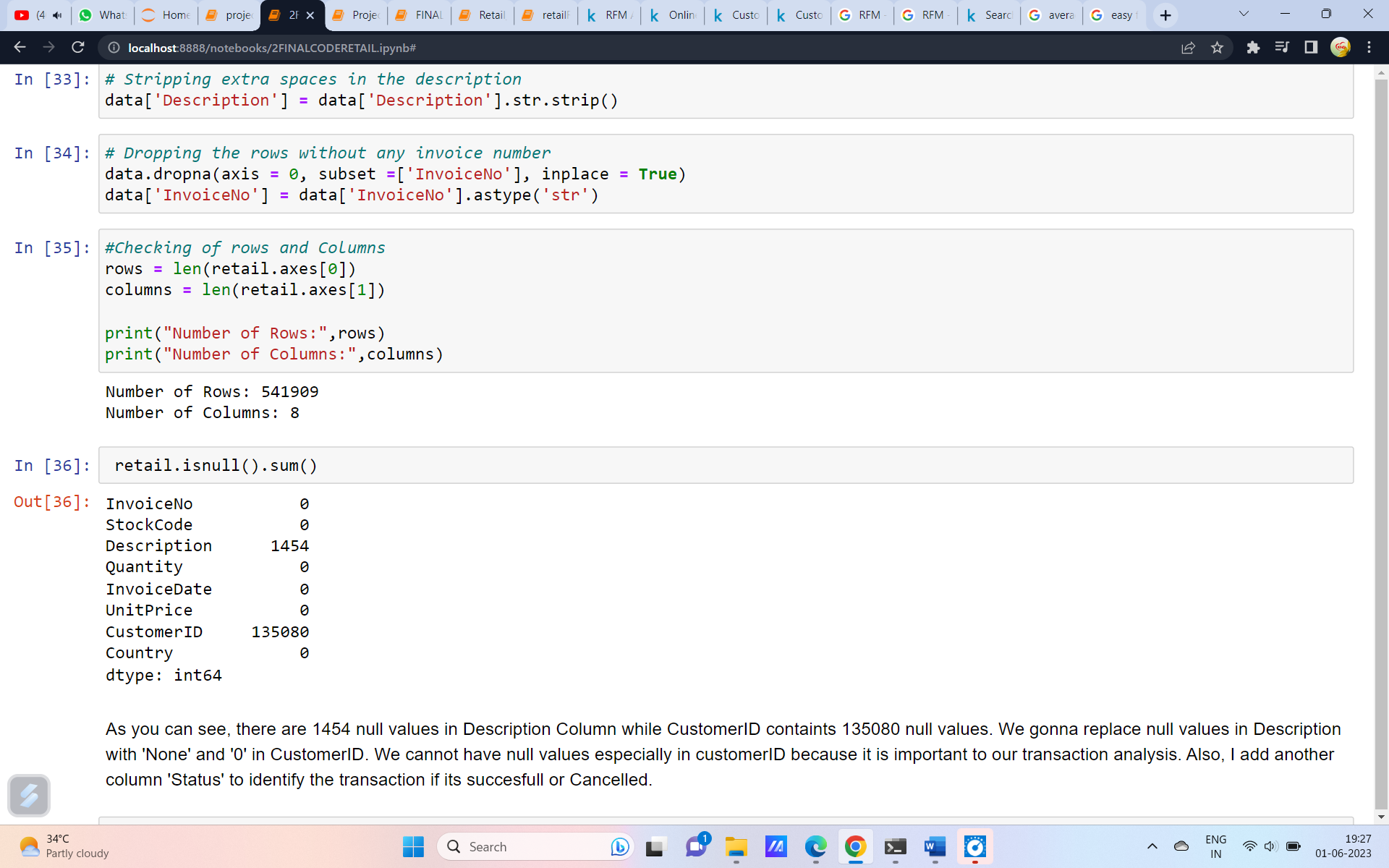
**Country:** Country name. Nominal. The name of the country where a customer resides.



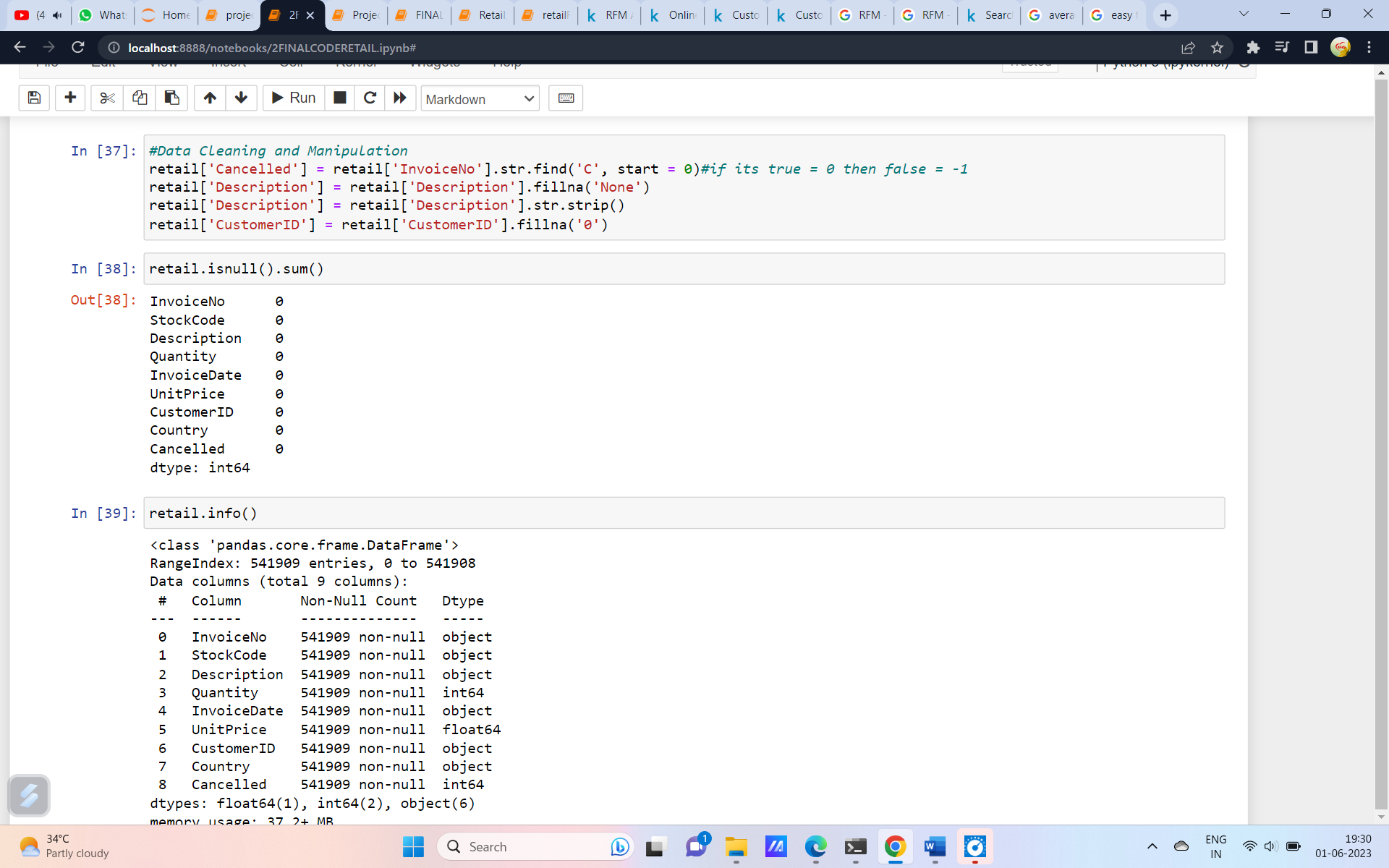
**This dataset contains ~5.4 lakh records and 8 attributes.**

**Chapter -4 Data Pre-processing Steps**

In this chapter we are going to check the null values and other data cleaning process.



In this the null values and NAN values are checked and the dataset seems to have null values in it. As you can see, there are 1454 null values in Description Column while CustomerID containts 135080 null values. We gonna replace null values in Description with 'None' and '0' in CustomerID. We cannot have null values especially in customerID because it is important to our transaction analysis.



Once the null values are treated the data is ready for the next process. The next step is to apply RFM analysis to the model. Before that lets see what is RFM in the next section

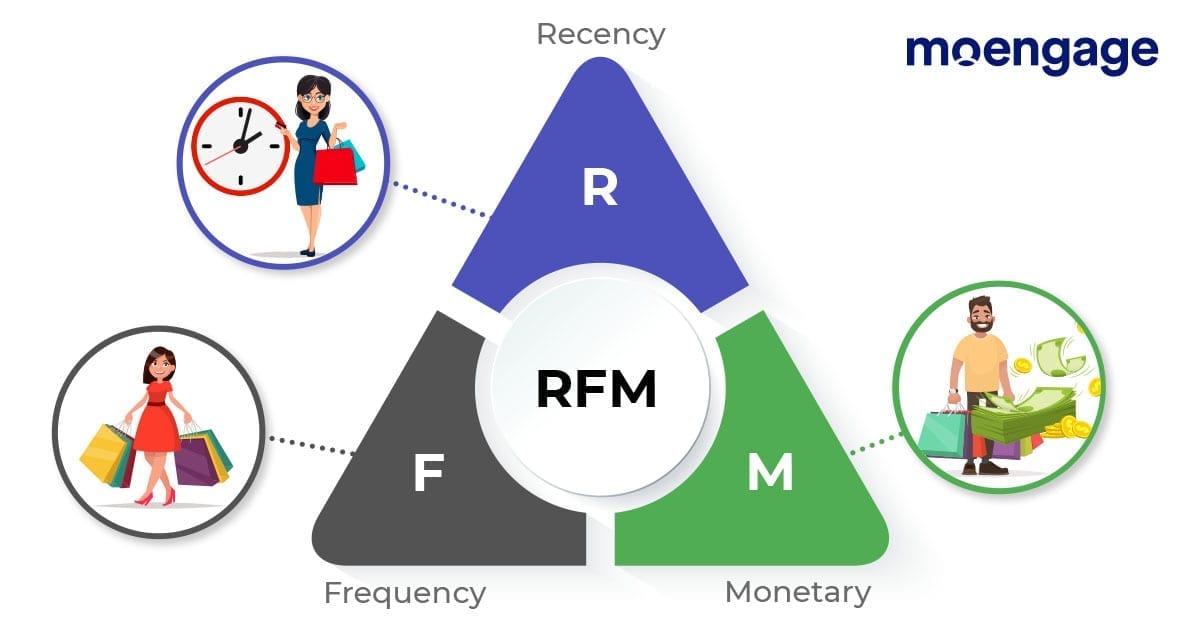
**Chapter 5. Choosing the Algorithm for the Project**

**Introduction**

**RFM Analysis**

RFM is a method used for analyzing customer value. It is commonly used in database marketing and direct marketing and has received particular attention in retail and professional services industries.

RFM stands for the three dimensions:  
**- R**ecency – How recently did the customer purchase?  
**- F**requency – How often do they purchase?  
**- M**onetary Value – How much do they spend?

****

Customer purchases may be represented by a table with columns for the customer name, date of purchase and purchase value. One approach to RFM is to assign a score for each dimension on a scale from 1 to 10. The maximum score represents the preferred behavior and a formula could be used to calculate the three scores for each customer. For example, a service-based business could use these calculations

* **Recency** = the maximum of "10 – the number of months that have passed since the customer last purchased" and 1
* **Frequency** = the maximum of "the number of purchases by the customer in the last 12 months (with a limit of 10)" and 1
* **Monetary** = the highest value of all purchases by the customer expressed as a multiple of some benchmark value

Alternatively, categories can be defined for each attribute. For instance, Recency might be broken into three categories: customers with purchases within the last 90 days; between 91 and 365 days; and longer than 365 days. Such categories may be derived from business rules or using data mining techniques to find meaningful breaks.

Alternatively, categories can be defined for each attribute. For instance, Recency might be broken into three categories: customers with purchases within the last 90 days; between 91 and 365 days; and longer than 365 days. Such categories may be derived from business rules or using data mining techniques to find meaningful breaks.

The resulting segments can be ordered from most valuable (highest recency, frequency, and value) to least valuable (lowest recency, frequency, and value). Identifying the most valuable RFM segments can capitalize on chance relationships in the data used for this analysis. For this reason, it is highly recommended that another set of data be used to validate the results of the RFM segmentation process. Advocates of this technique point out that it has the virtue of simplicity: no specialized statistical software is required, and the results are readily understood by business people. In the absence of other targeting techniques, it can provide a lift in response rates for promotions.

**Chapter 6. Motivation and Reasons For Choosing the Algorithm**

**WHY RFM?**

RFM analysis enables online retailers to increase revenue by targeting specific groups of existing customers with messages and offers that are more likely to be relevant based on data about a particular set of behaviors.

1. **Better understanding of customer behavior**: RFM analysis helps online retailers better understand their customers’ behavior based on their transaction history [**1**](https://www.omniconvert.com/blog/rfm-model/).
2. **Identification of customer segments**: RFM analysis allows online retailers to identify how customers are distributed among the RFM segments [**1**](https://www.omniconvert.com/blog/rfm-model/).
3. **Tailored marketing efforts**: By leveraging the RFM framework, online retailers can tailor their marketing efforts to specific customer groups, increasing response rates, customer retention, customer satisfaction, and customer lifetime value (CLTV) [**2**](https://www.smartinsights.com/online-brand-strategy/multichannel-strategies/underestimated-growth-pattern-ecommerce-companies-rfm-model/).
4. **Cost-effective**: RFM analysis is a cost-effective way of acquiring actionable customer behavior analysis and is relatively easy to quantify customer behavior.

**ADVANTAGES of RFM**

RFM analysis is a popular method for customer segmentation that stands for Recency, Frequency, and Monetary value. It has several advantages:

1. **Cost-effective**: RFM analysis is a cost-effective way of acquiring actionable customer behavior analysis and is relatively easy to quantify customer behavior .
2. **Predictive**: The RFM model is very valuable in predicting response and can help boost a company’s profits in the short term
3. **Simplicity**: Purchase behavior can be summarized utilizing a very small number of variables .
4. **Targeted**: RFM analysis is more meaningful for targeting particular customers .
5. **Boost remarketing strategy**: RFM analysis helps your customers to purchase more frequently with mail and other ad types .
6. **More loyal customers**: RFM analysis can help you identify customers who are not completely loyal to you and may need to feel special and show some attention .

**KMeans Algorithim**

K-means is a popular **unsupervised learning algorithm** used for clustering data points . The algorithm iteratively divides data points into K clusters by minimizing the variance in each cluster . It aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster .

The algorithm works by first selecting K initial centroids, where K is a predetermined number of clusters. Each data point is then assigned to its nearest centroid, forming K clusters. The centroids are then recomputed as the mean of all data points in the cluster, and the data points are reassigned to their nearest centroid. This process is repeated until the centroids no longer change or a maximum number of iterations is reached.

K-means is widely used in many applications due to its simplicity and efficiency. However, it has some limitations, such as sensitivity to the initial placement of centroids and difficulty in handling clusters of different sizes and densities.

It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

The algorithm takes the unlabeled dataset as input, divides the dataset into k-number of clusters, and repeats the process until it does not find the best clusters. The value of k should be predetermined in this algorithm.

The k-means clustering algorithm mainly performs two tasks:

* Determines the best value for K center points or centroids by an iterative process.
* Assigns each data point to its closest k-center. Those data points which are near to the particular k-center, create a cluster.

Hence each cluster has datapoints with some commonalities, and it is away from other clusters.

The below diagram explains the working of the K-means Clustering Algorithm:

## 

## How does the K-Means Algorithm Work?

The working of the K-Means algorithm is explained in the below steps:

**Step-1:** Select the number K to decide the number of clusters.

**Step-2:** Select random K points or centroids. (It can be other from the input dataset).

**Step-3:** Assign each data point to their closest centroid, which will form the predefined K clusters.

**Step-4:** Calculate the variance and place a new centroid of each cluster.

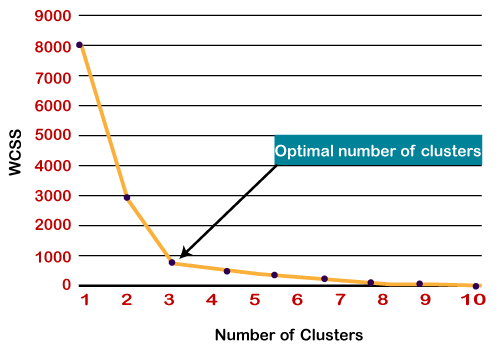
**Step-5:** Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.

**Step-6:** If any reassignment occurs, then go to step-4 else go to FINISH.

**Step-7**: The model is ready.

### **Elbow Method**

The Elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster.

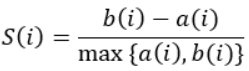


**Silhouette analysis:**

The silhouette coefficient or silhouette score kmeans is a measure of how similar a data point is within-cluster (cohesion) compared to other clusters (separation). The Silhouette score can be easily calculated in Python using the metrics module of the scikit-learn/sklearn library.

* Select a range of values of k (say 1 to 10).
* Plot Silhouette coefﬁcient for each value of K.

The equation for calculating the silhouette coefﬁcient for a particular data point:



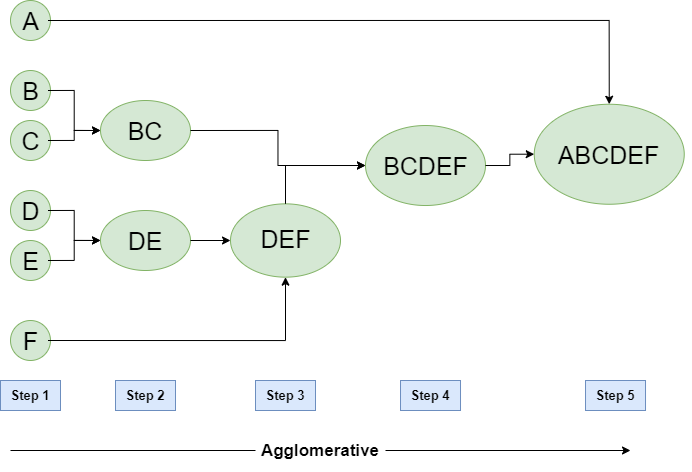
* S(i) is the silhouette coefficient of the data point i.
* a(i) is the average distance between i and all the other data points in the cluster to which i belongs.
* b(i) is the average distance from i to all clusters to which i does not belong.

**Hierarchical clustering**

A **Hierarchical clustering** method works via grouping data into a tree of clusters. Hierarchical clustering begins by treating every data point as a separate cluster. Then, it repeatedly executes the subsequent steps:

1. Identify the 2 clusters which can be closest together, and
2. Merge the 2 maximum comparable clusters. We need to continue these steps until all the clusters are merged together.

In Hierarchical Clustering, the aim is to produce a hierarchical series of nested clusters. A diagram called **Dendrogram**(A Dendrogram is a tree-like diagram that statistics the sequences of merges or splits) graphically represents this hierarchy and is an inverted tree that describes the order in which factors are merged (bottom-up view) or clusters are broken up (top-down view).

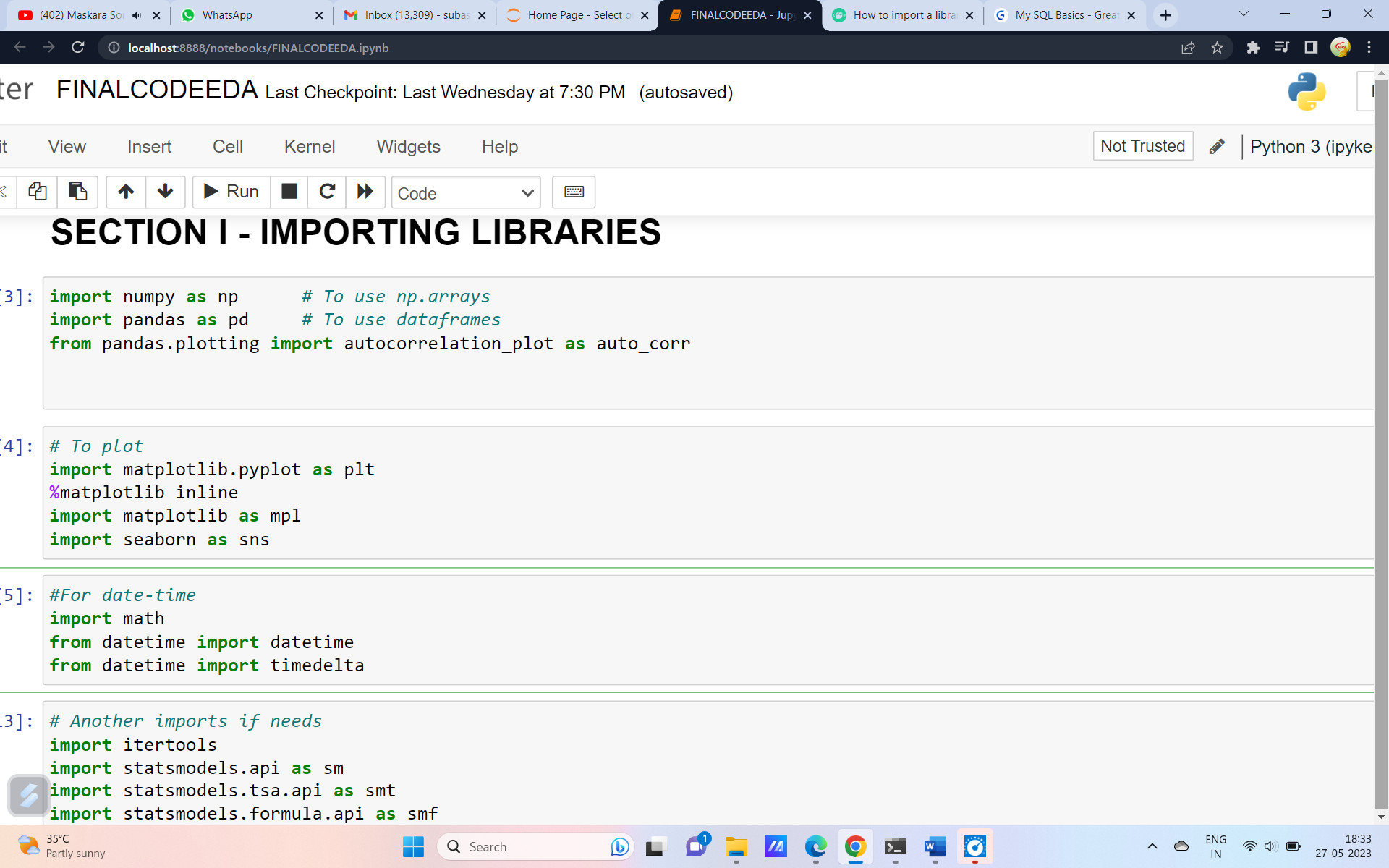


**Chapter -7 Model Building**

In this section we are going to start working on the dataset given. The flow of the coding is based on the flow given below.

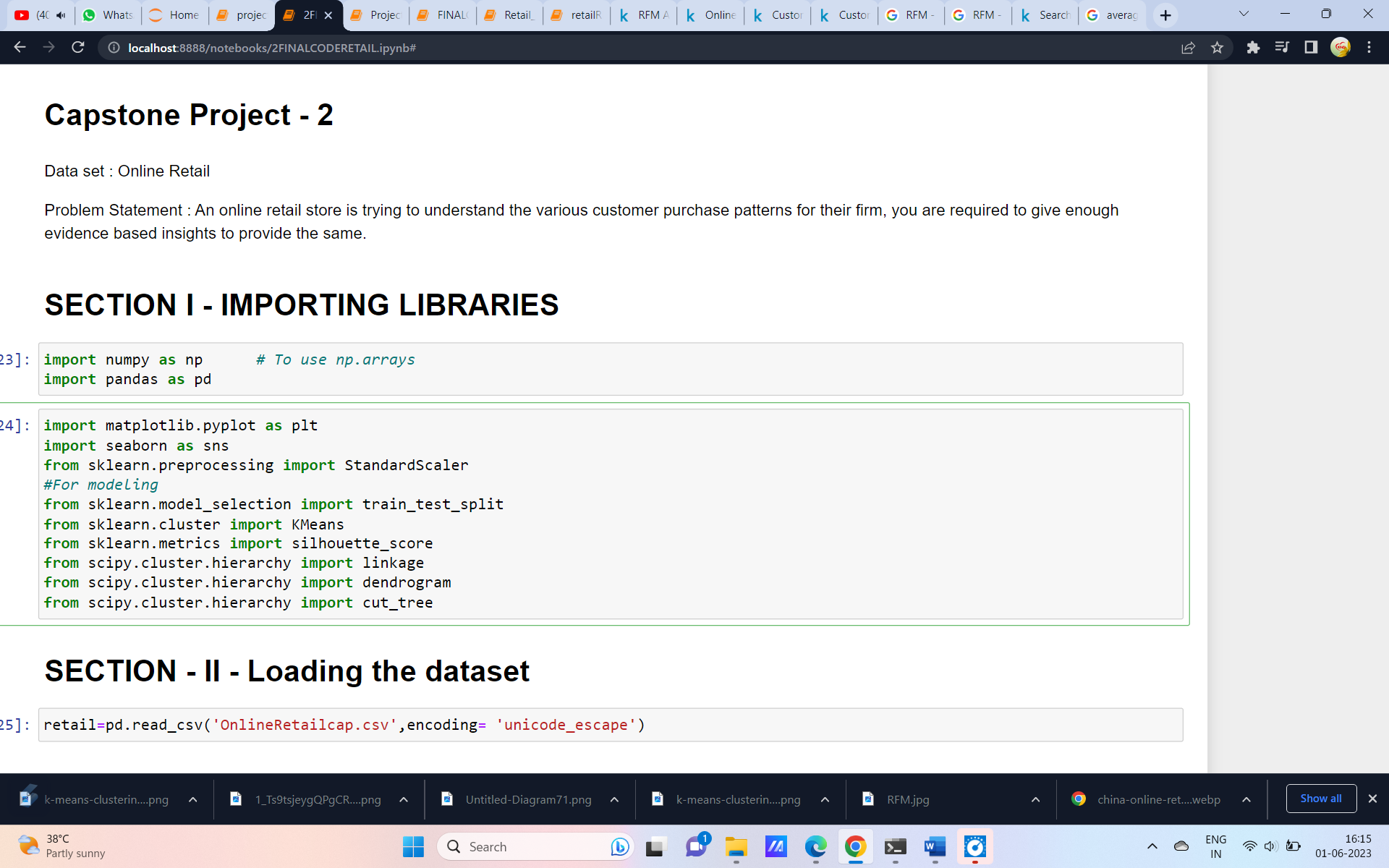
**Section – I -Importing Libraries**

In Python, libraries are used to refer to a collection of modules that are used repeatedly in various programs without the need of writing them from scratch.. In Python, you can import libraries using the import statement. After importing the library, you can use its functions and methods by calling them with the library’s alias.



**Section II Loading the dataset**

In Python, you can load a dataset from various sources such as a CSV file, an Excel file, or a database. One common way to load a dataset is to use the pandas library. For example, to load a CSV file into a pandas DataFrame, you can use the read\_csv() function.

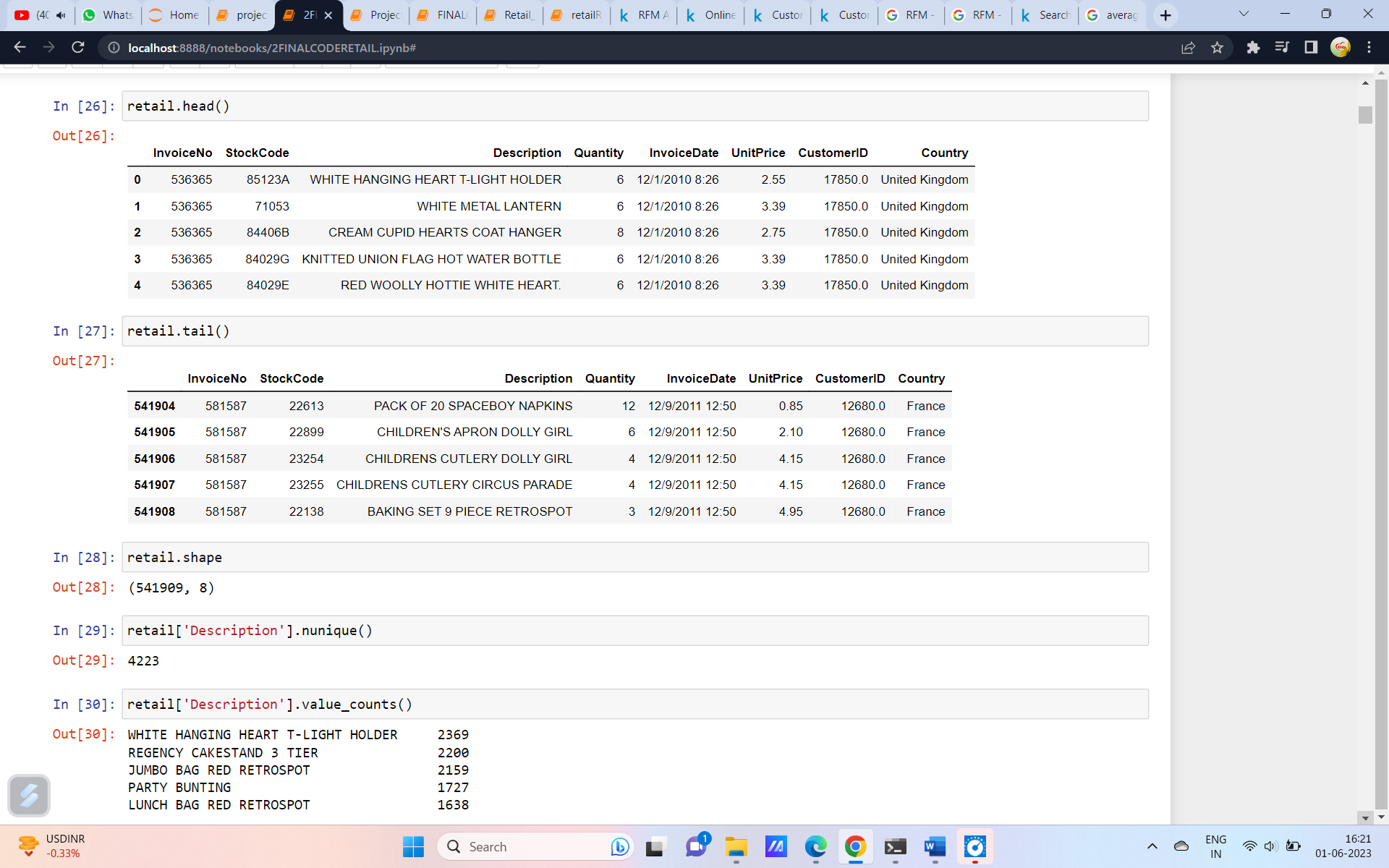


Section III - Data Exploration

Data exploration is the process of analyzing and visualizing a dataset to better understand its characteristics and uncover any underlying patterns or relationships. In Python, there are several libraries that can be used for data exploration, including pandas, matplotlib, and seaborn.

The pandas library provides several functions for exploring data, such as the describe() function which generates summary statistics for numerical columns, and the value\_counts() function which counts the frequency of unique values in a categorical column.

For visualizing data, you can use the matplotlib and seaborn libraries to create various types of plots such as histograms, scatter plots, and box plots.



# **SECTION - 4 - Data Visualization**

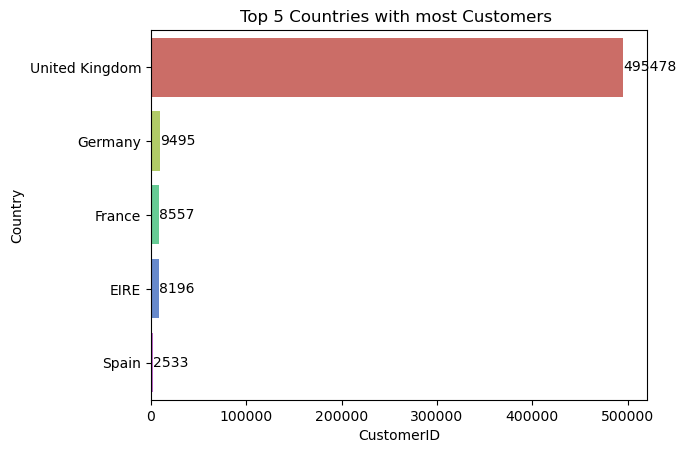


Fig.1

The Fig.1 plot shows the top 5 countries with most of the customers. United kingdom has the top most customers.

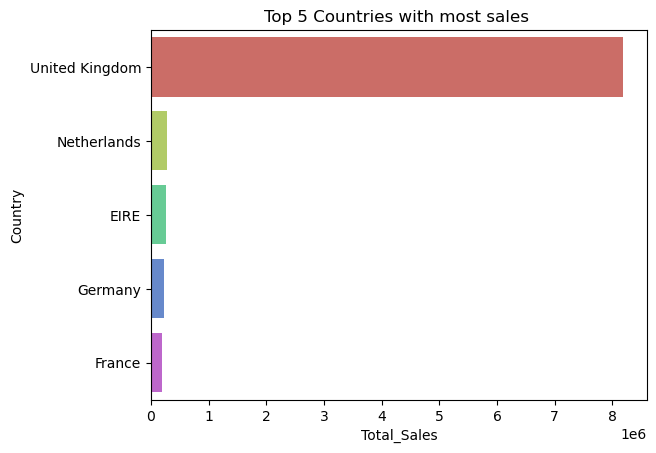


Fig.2

The above graph shows the country which has most of the sales. It is clear that the united kingdom has the higest number of sales compared to others. Netherlands does not come in top 5 countries which have more customer but it has high sales. Though spain has more number of customers the sales is not in top 5.

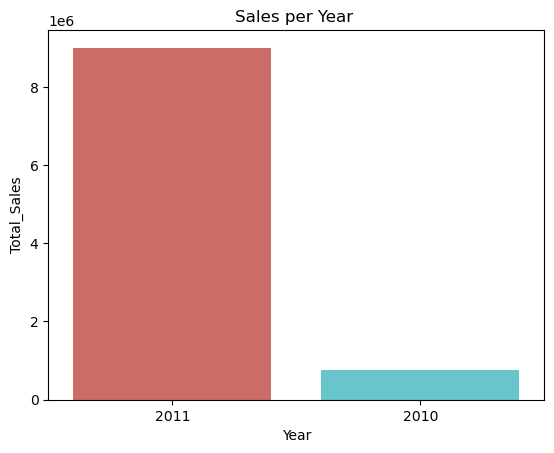


Fig.3

The Fig.3 graph shows the sales per year. It is clear that the year 2011 has the most of the sales that is because only one month data is present for 2010.

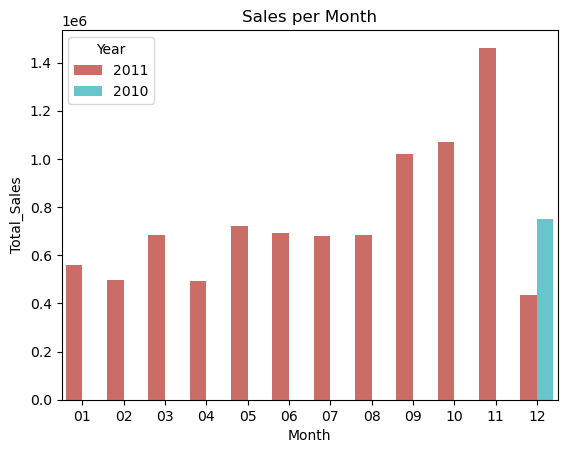


Fig.4

The above graph shows the total sales every month. As mentioned earlier it is clear that the data set contains only 2011 data and 2010 data is present only for the December month.

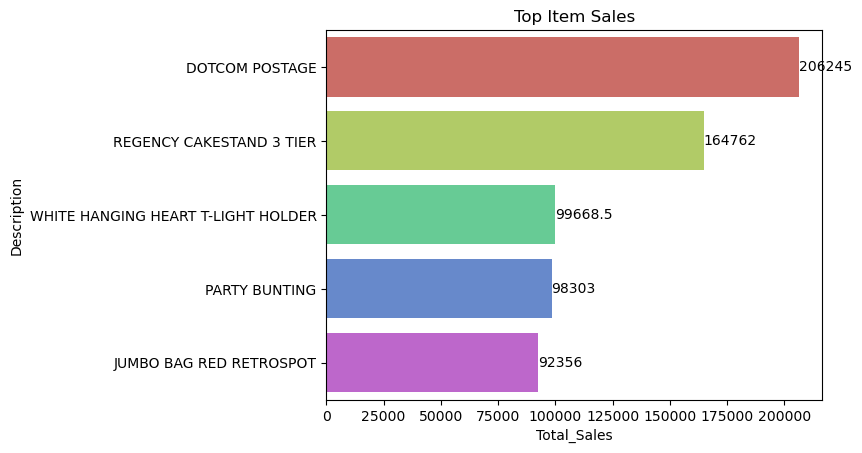


Fig.5

The Fig.5 graph shows the top items according to the total sales.

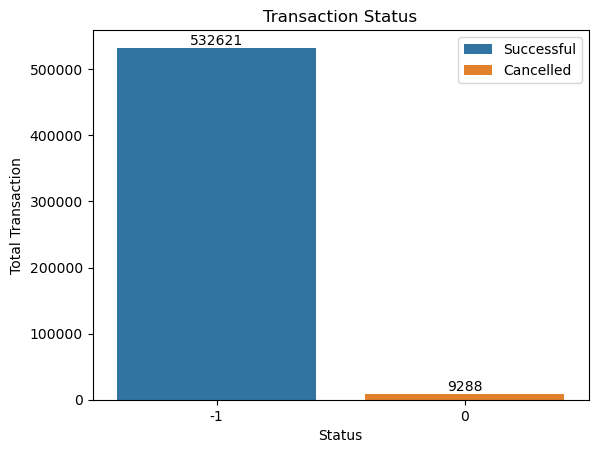


Fig.6

Fig.6 shows the total number of successful and cancelled transactions. 98.29% of the transactions are successful and only 1.71% are cancelled ones.

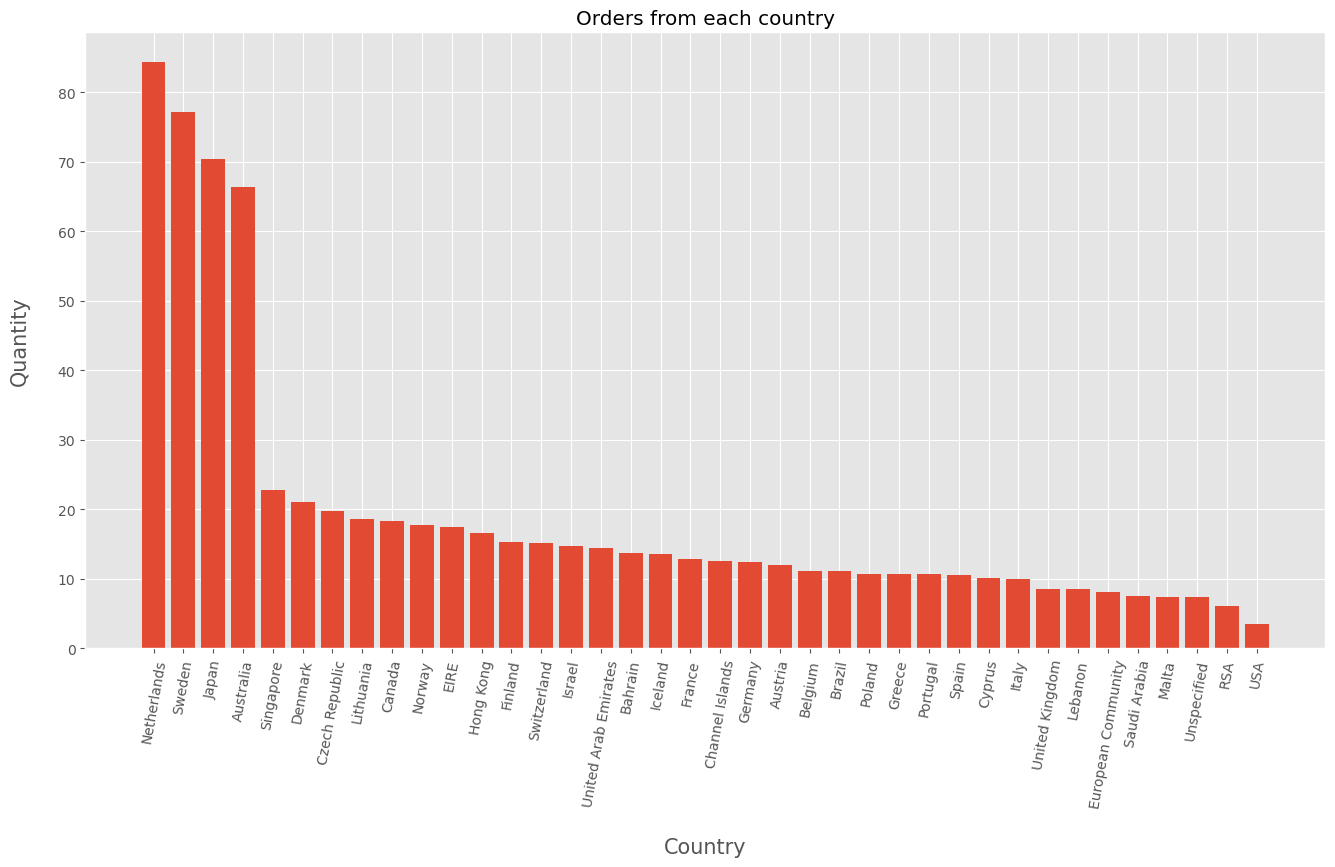


Fig.7

Fig.7 shows the country which has more number of orders with quantity wise. United kingdom has the most number of quantity.

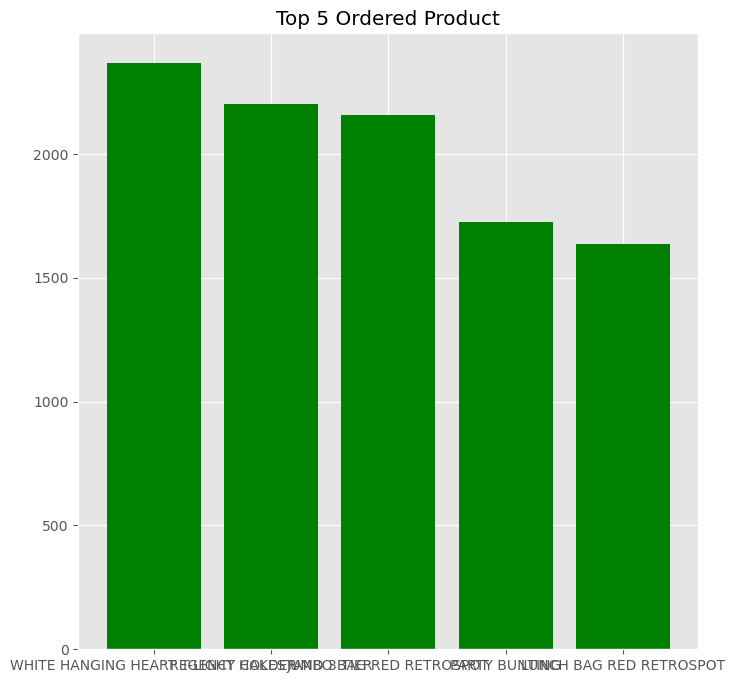


Fig.8

Fig.8 shows the products which are ordered most. White hanging heart is the most ordered product.

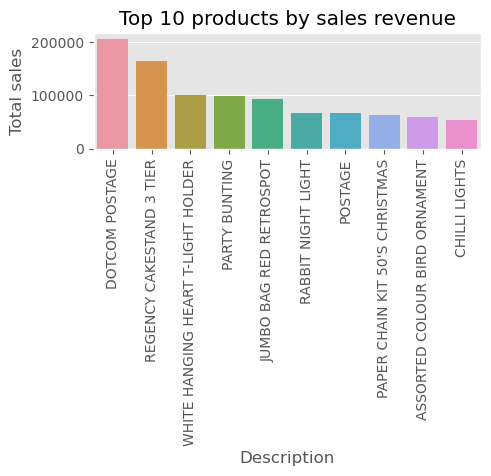


Fig.9

Fig.9 shows the top 10 products based on the total sales. The Dotcom Postage is the product which was sold most.

Chapter 5. Choosing the Algorithm for the Project

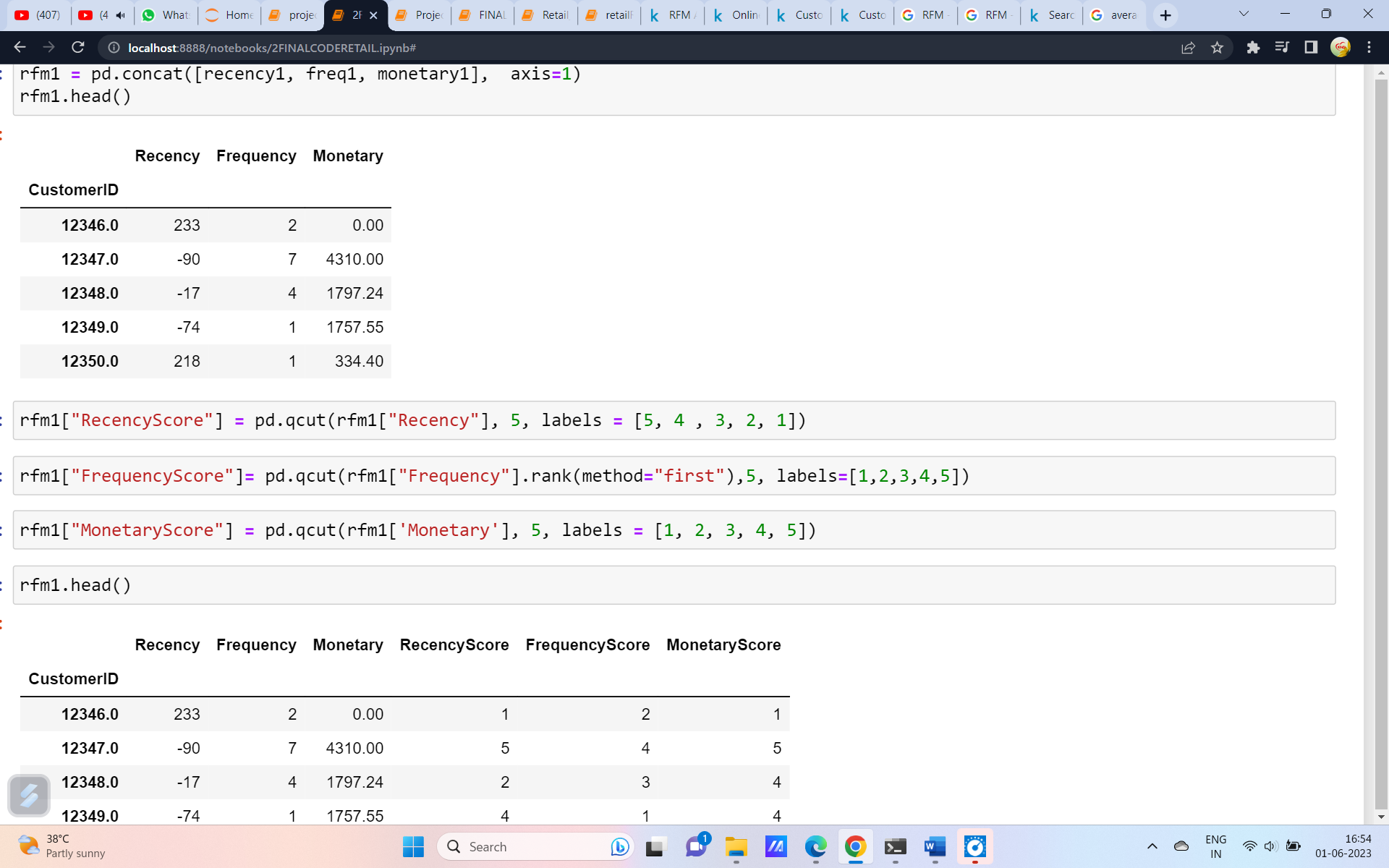
**RFM Analysis**

Recency: How recently a customer has made a purchase

# Frequency: How often a customer makes a purchase.

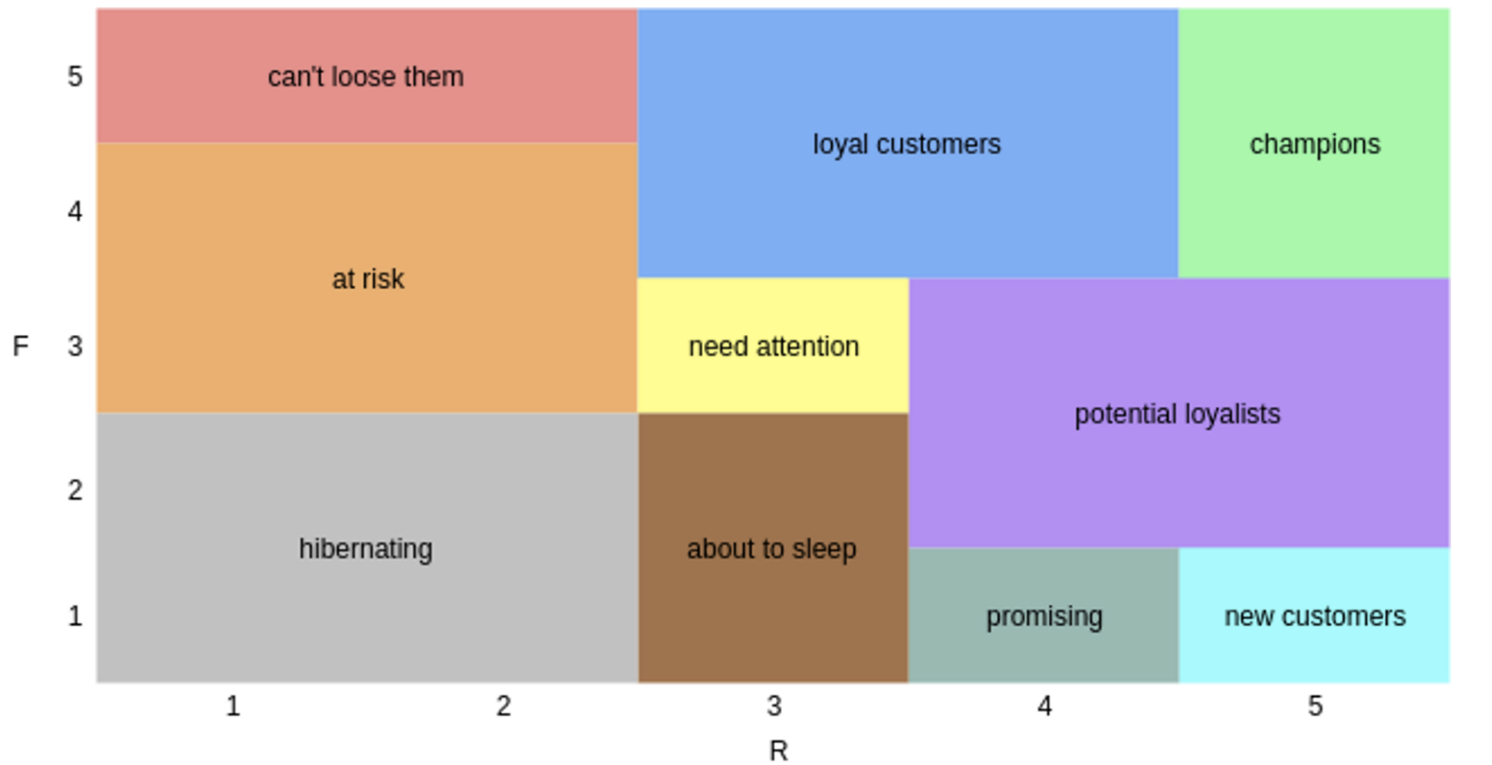
# Monetary Value: How much money a customer spends on purchases.

# so it is based on amount so multipluing price and quantity



**Customer Segmentation – Part 1**

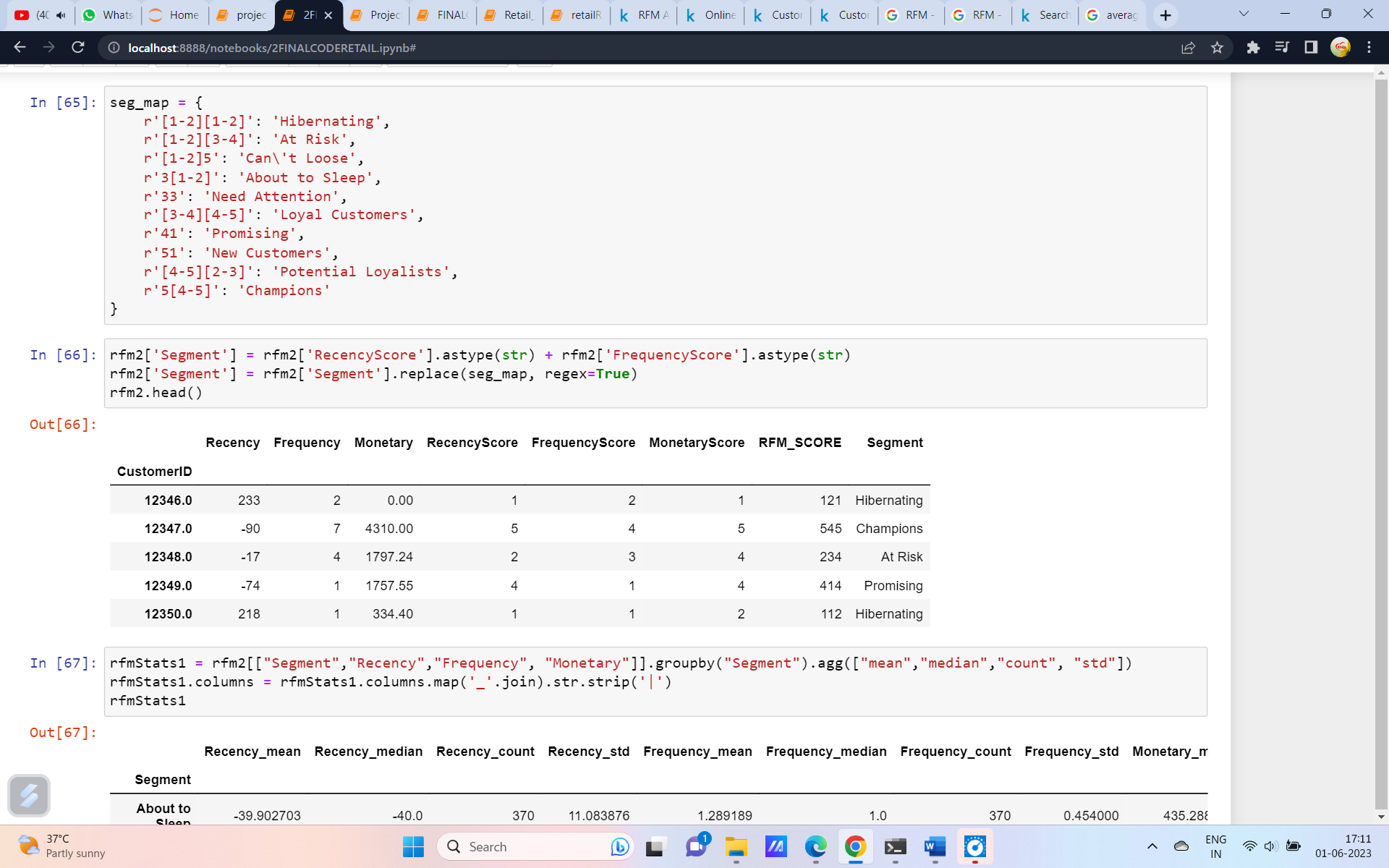
In this the customers are segmented based on the segments.



Based on the image above the customers are segmented into

* Hibernating
* At risk
* Cant loose them
* About to sleep
* Need attension
* Loyal customers
* Champions
* Potential loyalists
* Promising
* New customers

The RFM analysis is carried out based on the above one and the customers are segmented.



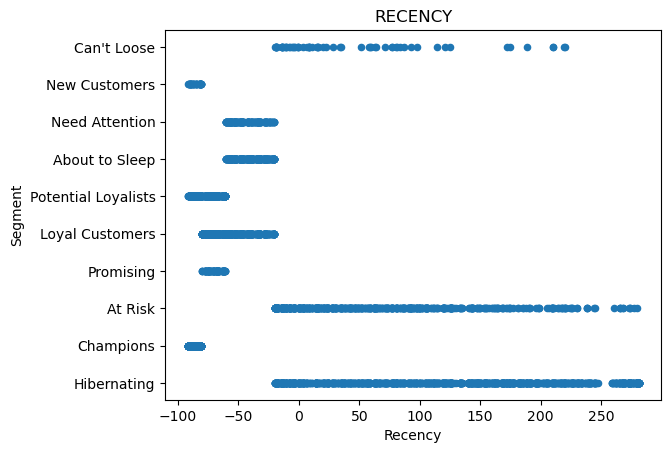


Fig.10

The fig.10 shows the recency vs the segment of the customers.

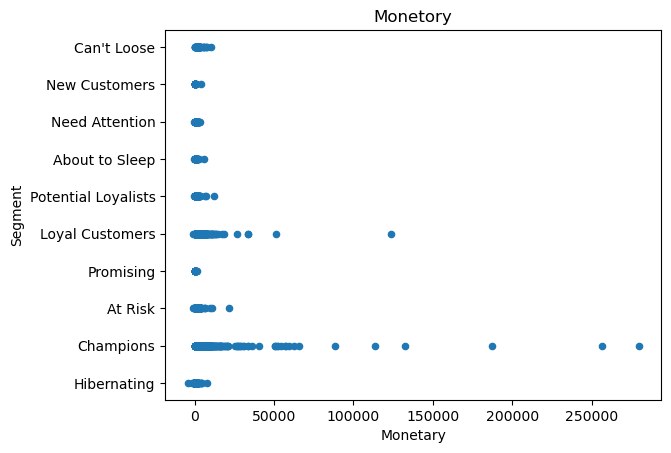


Fig.11

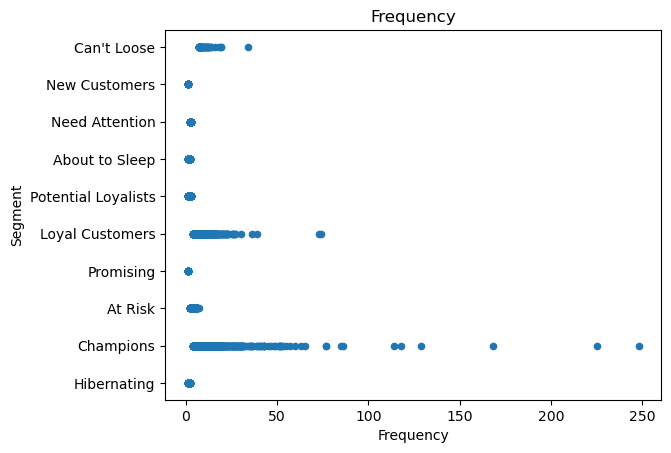


Fig.12

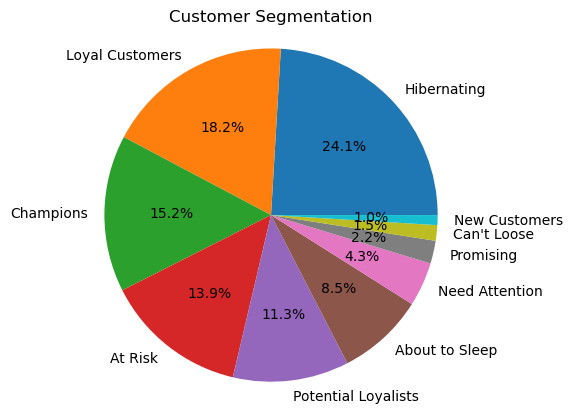


Fig.13

Fig.13 shows the segmentation of the customers. It shows there are 18.2% loyal customers.24.1% are hibernating.15.2% are champions.1.5% customers are very important and cant be loosed.

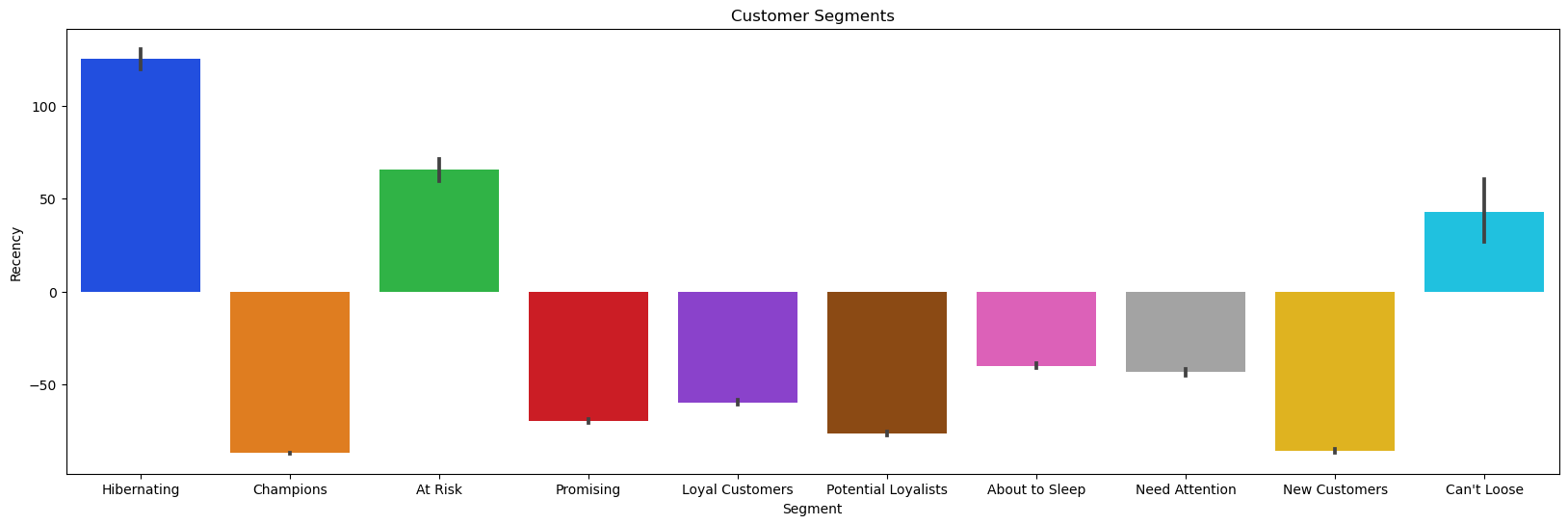


Fig.14

Fig.14 shows  Customer whose frequency is high is often spending more money in purchasing the products Customer whose frequency is less is spending less money in purchasing the products.

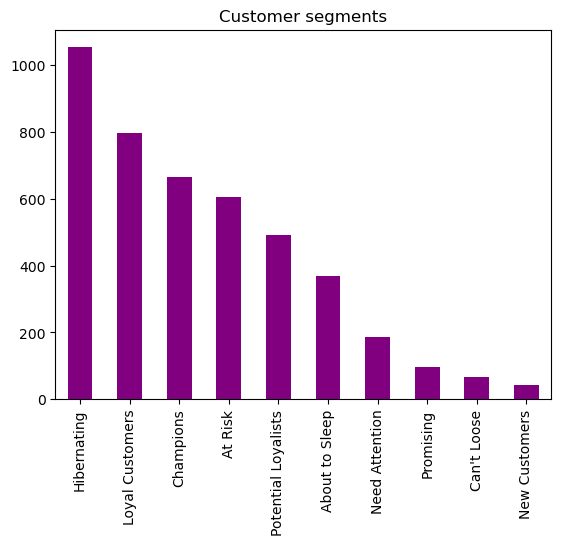


Fig.15

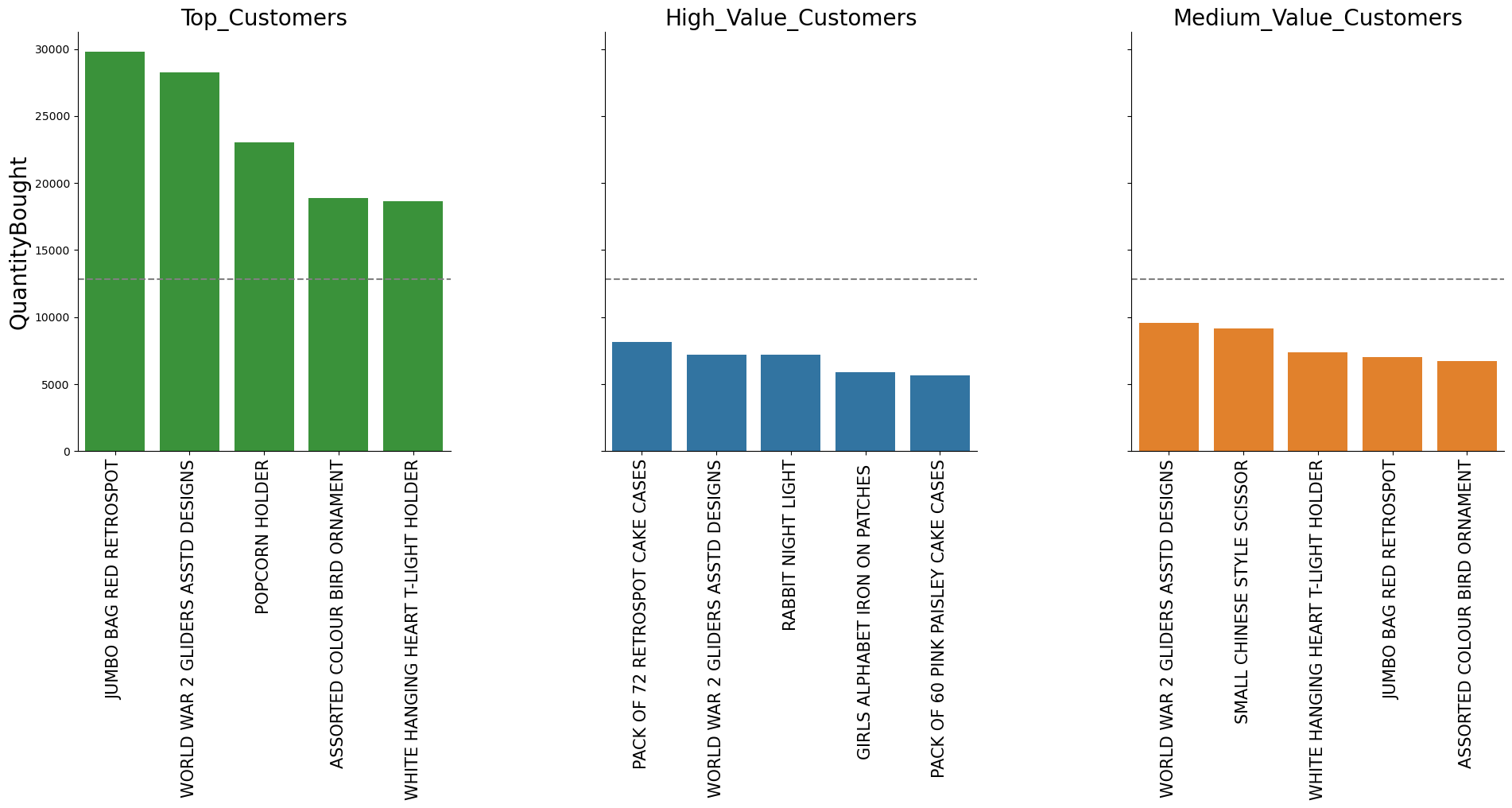
# **RFM CUSTOMER SEGMENT - TYPE II**

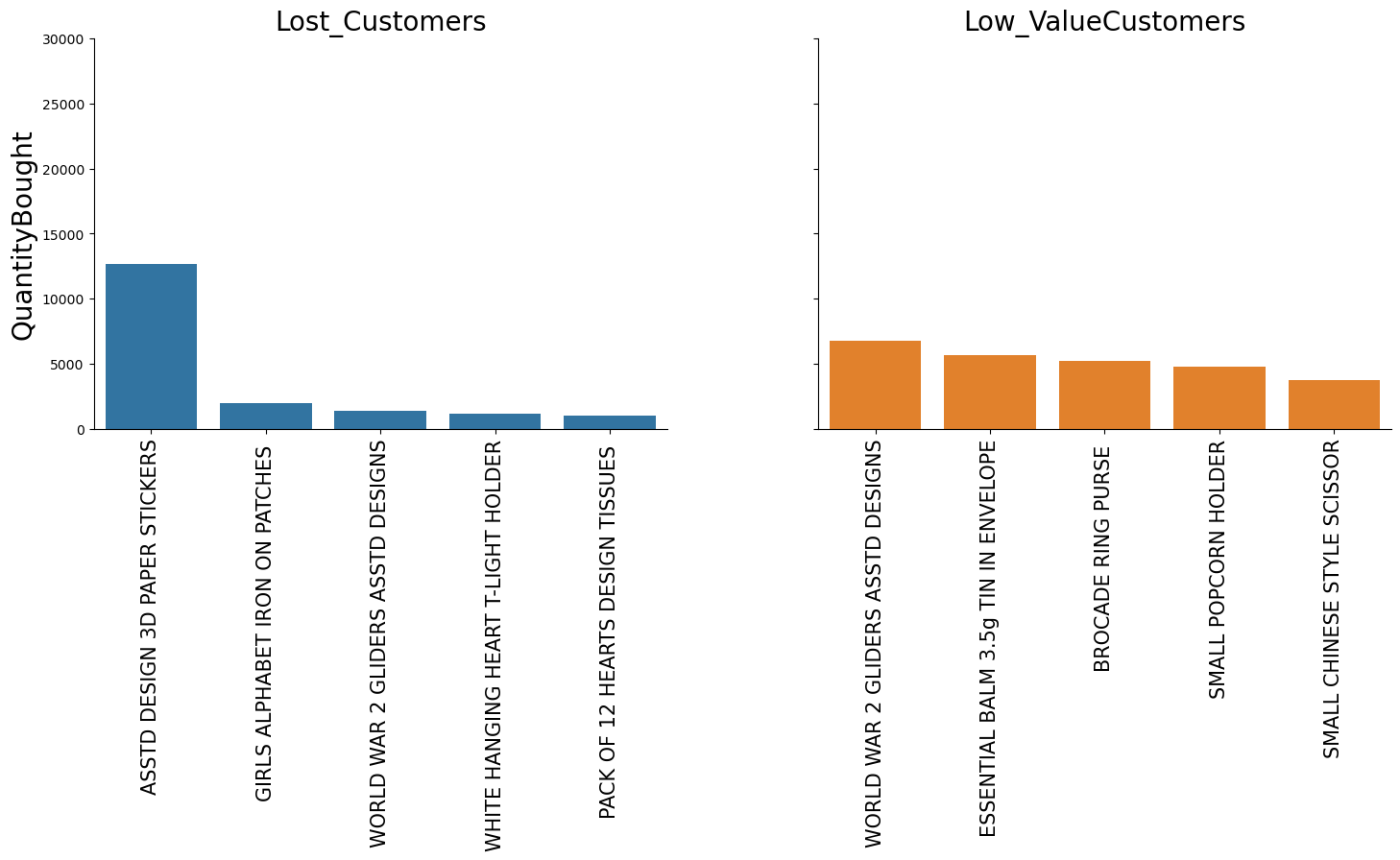
In this method the customer segmentation is carried based on the RFM score.

The customers are segmented into following

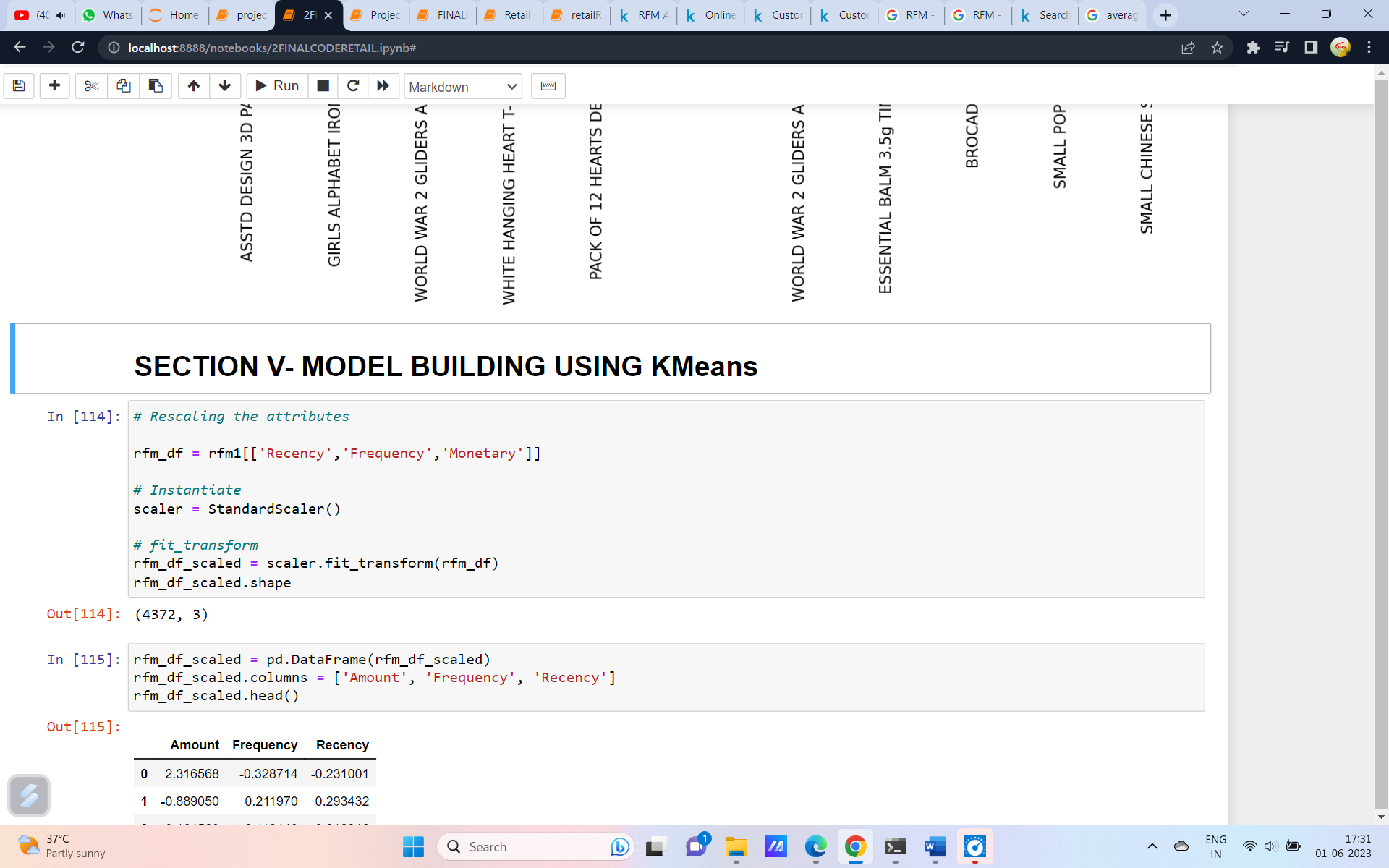
|  |  |
| --- | --- |
| 'RFM\_SCORE' >4.5 | 'Top\_Customers' |
| 'RFM\_SCORE' >4 | 'High\_Value\_Customers' |
| 'RFM\_SCORE' >3 | 'Medium\_Value\_Customers' |
| 'RFM\_SCORE' >1.6 | 'Low\_ValueCustomers' |

The below figure shows the product bought by the customer segments.



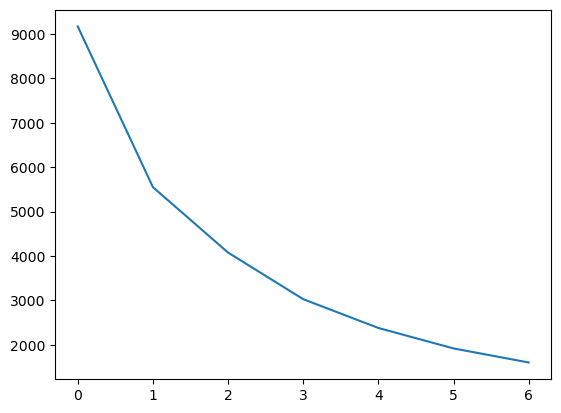


**Chapter 7-MODEL BUILDING USING KMeans**



The model is built using Kmeans algorithm. Before that it is important to scale the data. So using standard scaler the data is scaled.

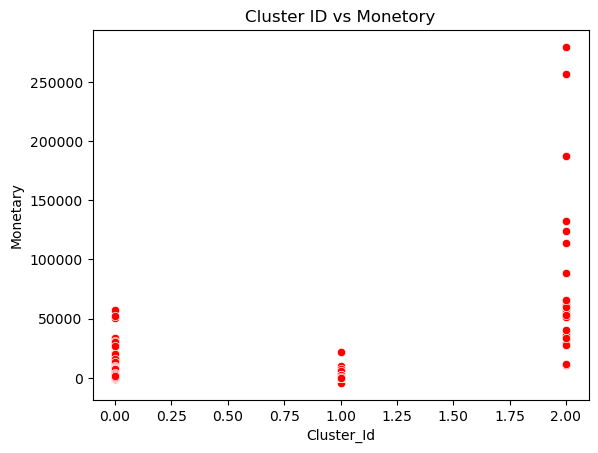
To find the optimal number of clusters elbow mentod is used. From the elbow methos the number of cluster value is K=3



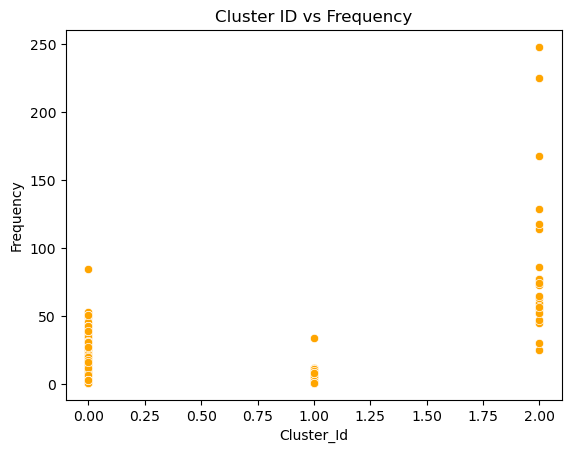
Still the sillohutte analayisi is used to find the K value.

**Chapter 8. Model Evaluation and Techniques**

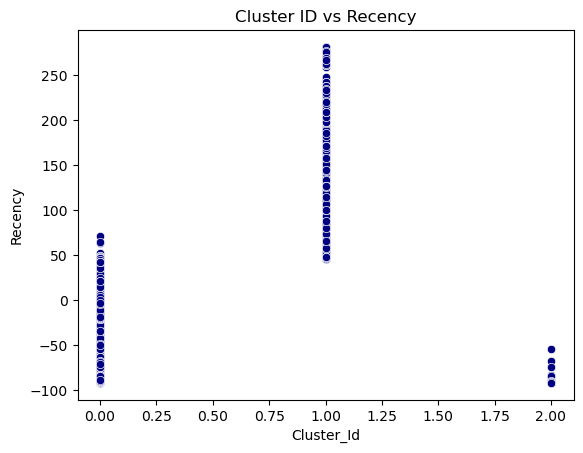
The model is built and evaluated. From this the below results are got.



In the above graph of Monetory vs Cluster ID it shows the cluster 2 customers tend to spend more money.

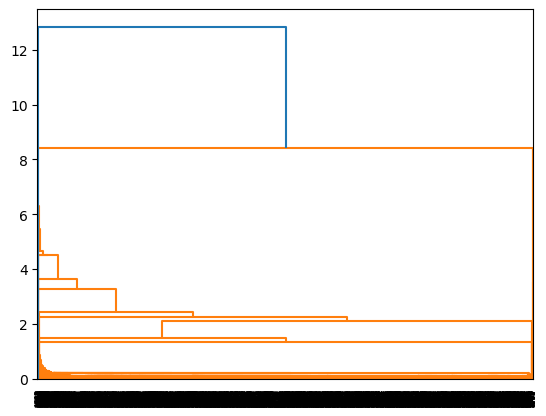


The above graph shows the Recency vs Cluster Id. It is observed that the customers from the cluster 2 tend to visit more times than the other.

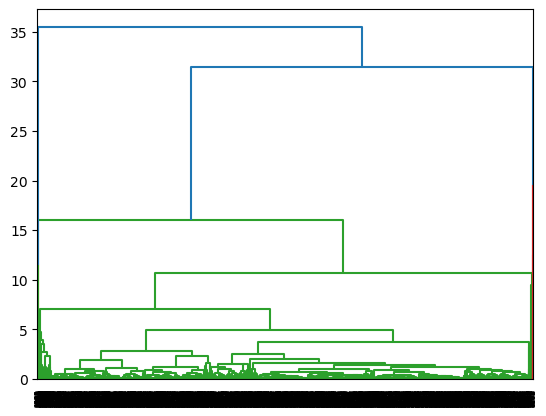


The above graphs shows the scatterplots Recency with respect to Cluster ID, it is very clear that the customers belong to cluster 1 have visited more recency.

Based on this Hierarchical clustering is applied and the dendograpm is plotted.



Dendogram for single linkage is not good. So we go for complete.



The dendogram for complate linkage is given above.

**Chapter -9. Inferences**

Inference: K-Means Clustering with 3 Cluster Ids

--Customers with Cluster Id 1 are the customers with high amount of transactions as compared to other customers.

--Customers with Cluster Id 1 are frequent buyers.

--Customers with Cluster Id 2 are not recent buyers and hence least of importance from business point of view.

--Hierarchical Clustering with 3 Cluster Labels

--Customers with Cluster\_Labels 2 are the customers with high amount of transactions as compared to other customers.

--Customers with Cluster\_Labels 2 are frequent buyers.

--Customers with Cluster\_Labels 0 are not recent buyers and hence least of importance from business point of view

--25% Customers are Potential Customers and Champions.

-- ~1% are New customers. Will have to plan some new interesting schems to attract more of them.

--~40 % are Hibernating and At Risk Customers.Will have to plan some discounts keeping into consideration the price plasticity.

**Chapter 11. Conclusion**

Overall, RFM analysis enables marketers to increase revenue by targeting specific groups of existing customers with messages and offers that are more likely to be relevant based on data about a particular set of behaviors. This leads to increased response rates, customer retention, customer satisfaction, and customer lifetime value (CLTV) .

**Chapter 12- REFERENCES**

<https://crunchingthedata.com/when-to-use-k-means-clustering/#:~:text=One%20of%20the%20main%20advantages,clustering%20model%20to%20be%20available>

<https://towardsdatascience.com/clustering-evaluation-strategies-98a4006fcfc>

<https://www.geeksforgeeks.org/rfm-analysis-analysis-using-python/>

CODE

import numpy as np # To use np.arrays

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

#For modeling

from sklearn.model\_selection import train\_test\_split

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

from scipy.cluster.hierarchy import linkage

from scipy.cluster.hierarchy import dendrogram

from scipy.cluster.hierarchy import cut\_tree

retail=pd.read\_csv('OnlineRetailcap.csv',encoding= 'unicode\_escape')

retail.head()

retail.tail()

retail['Description'].nunique()

retail.shape

# Stripping extra spaces in the description

data['Description'] = data['Description'].str.strip()

# Dropping the rows without any invoice number

data.dropna(axis = 0, subset =['InvoiceNo'], inplace = True)

data['InvoiceNo'] = data['InvoiceNo'].astype('str')

#Data Cleaning and Manipulation

retail['Cancelled'] = retail['InvoiceNo'].str.find('C', start = 0)#if its true = 0 then false = -1

retail['Description'] = retail['Description'].fillna('None')

retail['Description'] = retail['Description'].str.strip()

retail['CustomerID'] = retail['CustomerID'].fillna('0')

#Data Cleaning and Manipulaton

retail['InvoiceDate'] = pd.to\_datetime(retail['InvoiceDate'])

retail['Year'] = pd.to\_datetime(retail['InvoiceDate']).dt.strftime('%Y')

retail['Month'] = pd.to\_datetime(retail['InvoiceDate']).dt.strftime('%m')

retail['CustomerID'] = retail['CustomerID'].astype(int)

retail['Total\_Sales'] = retail['Quantity']\*retail['UnitPrice']

#Analysis and Visualization

total\_sales\_per\_country = retail.groupby('Country')['Total\_Sales'].sum().reset\_index().sort\_values('Total\_Sales',ascending=False)

z = sns.barplot(data=total\_sales\_per\_country.head(5), x='Total\_Sales',y='Country',palette = 'hls')

plt.title("Top 5 Countries with most sales ")

total\_customer\_per\_country = retail.groupby('Country')['CustomerID'].count().reset\_index().sort\_values('CustomerID',ascending=False)

z = sns.barplot(data=total\_customer\_per\_country.head(5), x='CustomerID',y='Country',palette = 'hls')

plt.title("Top 5 Countries with most Customers ")

for i in z.containers:

z.bar\_label(i,)

total\_sales\_per\_year = retail.groupby('Year')['Total\_Sales'].sum().reset\_index().sort\_values('Total\_Sales',ascending=False)

z = sns.barplot(data=total\_sales\_per\_year, x='Year',y='Total\_Sales',palette = 'hls')

plt.title("Sales per Year ")

total\_sales\_per\_month = retail.groupby(['Month','Year'])['Total\_Sales'].sum().reset\_index()

z = sns.barplot(data=total\_sales\_per\_month, y='Total\_Sales',x='Month',hue='Year',palette = 'hls')

plt.title("Sales per Month")

total\_sales\_per\_category = retail.groupby('Description')['Total\_Sales'].sum().reset\_index().sort\_values('Total\_Sales',ascending=False)

z = sns.barplot(data=total\_sales\_per\_category.head(5), y='Description',x='Total\_Sales',palette = 'hls')

plt.title("Top Item Sales")

for i in z.containers:

z.bar\_label(i,)

#total\_cancelled = retail.groupby(['Cancelled']).count().reset\_index()

z = sns.countplot(x=retail['Cancelled'], hue = retail['Cancelled'], dodge=False )

plt.ylabel("Total Transaction")

plt.xlabel("Status")

plt.title("Transaction Status")

labels = ["Successful","Cancelled"]

z.legend(labels)

for i in z.containers:

z.bar\_label(i,)

from datetime import datetime

import datetime as dt

date\_string = '2022-09-19 13:55:26'

date\_format = '%Y-%m-%d %H:%M:%S'

datetime\_object = datetime.strptime(date\_string, date\_format)

freq1 = df1.groupby("CustomerID").agg({"InvoiceDate":"nunique"}).rename(columns={"InvoiceDate": "Frequency"})

freq1

#Monetory

df1["TotalPrice"] = df1["Quantity"] \* df1["UnitPrice"]

monetary1 = df1.groupby("CustomerID").agg({"TotalPrice":"sum"}).rename(columns={"TotalPrice":"Monetary"})

monetary1.head()

print("2009-2010: Min Date", df1["InvoiceDate"].min(), "Max Date", df1["InvoiceDate"].max())

# Convert the 'InvoiceDate' column to datetime objects

df1['InvoiceDate'] = pd.to\_datetime(df1['InvoiceDate'])

# Perform the subtraction

recency1 = (dt.datetime(2011, 9, 9) - df1.groupby("CustomerID").agg({"InvoiceDate":"max"})).rename(columns = {"InvoiceDate":"Recency"})

recency1["Recency"] = recency1["Recency"].apply(lambda x: x.days)

rfm1 = pd.concat([recency1, freq1, monetary1], axis=1)

rfm1.head()

rfm1["RecencyScore"] = pd.qcut(rfm1["Recency"], 5, labels = [5, 4 , 3, 2, 1])

rfm1["FrequencyScore"]= pd.qcut(rfm1["Frequency"].rank(method="first"),5, labels=[1,2,3,4,5])

rfm1["MonetaryScore"] = pd.qcut(rfm1['Monetary'], 5, labels = [1, 2, 3, 4, 5])

# RFM Scores: Category

rfm1["RFM\_SCORE"] = (rfm1['RecencyScore'].astype(str) +

rfm1['FrequencyScore'].astype(str) +

rfm1['MonetaryScore'].astype(str))

seg\_map = {

r'[1-2][1-2]': 'Hibernating',

r'[1-2][3-4]': 'At Risk',

r'[1-2]5': 'Can\'t Loose',

r'3[1-2]': 'About to Sleep',

r'33': 'Need Attention',

r'[3-4][4-5]': 'Loyal Customers',

r'41': 'Promising',

r'51': 'New Customers',

r'[4-5][2-3]': 'Potential Loyalists',

r'5[4-5]': 'Champions'

}

rfm2['Segment'] = rfm2['RecencyScore'].astype(str) + rfm2['FrequencyScore'].astype(str)

rfm2['Segment'] = rfm2['Segment'].replace(seg\_map, regex=True)

rfm2.head()

rfmStats1 = rfm2[["Segment","Recency","Frequency", "Monetary"]].groupby("Segment").agg(["mean","median","count", "std"])

rfmStats1.columns = rfmStats1.columns.map('\_'.join).str.strip('|')

rfmStats1

segment\_counts = rfm2['Segment'].value\_counts()

# Create a pie chart

plt.pie(segment\_counts, labels=segment\_counts.index, autopct='%1.1f%%')

plt.title('Customer Segmentation')

plt.axis('equal')

plt.show()

# Create a scatter plot of 'Recency' vs 'Frequency'

plt.scatter(rfm2['Recency'], rfm2['Frequency'])

plt.xlabel('Recency')

plt.ylabel('Frequency')

plt.title('Recency vs frequency')

plt.show()

# Create a scatter plot of 'Recency' vs 'Monetary'

plt.scatter(rfm1['Recency'], rfm1['Monetary'])

plt.xlabel('Recency')

plt.ylabel('Monetary')

plt.title('Recency vs Monetory')

plt.show()

# Create a scatter plot of 'Frequency' vs 'Monetary'

plt.scatter(rfm1['Frequency'], rfm1['Monetary'])

plt.xlabel('Frequency')

plt.ylabel('Monetary')

plt.title('monetory vs frequency')

plt.show()

# set figure size

plt.figure(figsize=(20, 6))

# create bar plot

sns.barplot(x=rfm1["Segment"], y=rfm1["Recency"], palette="bright")

plt.title('Customer Segments')

plt.show()

# create bar plot with color

rfm1['Segment'].value\_counts().plot(kind='bar', color='purple')

plt.title('Customer segments')

plt.show()

RFM=rfm1[['Recency','Frequency','Monetary']]

RFM

RFM['R\_Rank']= RFM['Recency'].rank(ascending=False)

RFM['F\_Rank']= RFM['Frequency'].rank(ascending=True)

RFM['M\_Rank']= RFM['Monetary'].rank(ascending=True)

RFM['R\_rank\_norm']=(RFM['R\_Rank']/RFM['R\_Rank'].max())\*100

RFM['F\_rank\_norm']=(RFM['F\_Rank']/RFM['F\_Rank'].max())\*100

RFM['M\_rank\_norm']=(RFM['M\_Rank']/RFM['M\_Rank'].max())\*100

RFM['RFM\_SCORE']=0.15\*RFM['R\_rank\_norm'] +0.28\* RFM['F\_rank\_norm']+0.57\*RFM['M\_rank\_norm']

RFM['RFM\_SCORE'] \*= 0.05

RFM[['RFM\_SCORE']].sort\_values(by='RFM\_SCORE' ,ascending=False)

RFM['Customer\_Segments']= np.where(RFM['RFM\_SCORE']>4.5,'Top\_Customers',

np.where(RFM['RFM\_SCORE']>4,'High\_Value\_Customers',

np.where(RFM['RFM\_SCORE']>3,'Medium\_Value\_Customers',

np.where(RFM['RFM\_SCORE']>1.6,'Low\_ValueCustomers', 'Lost\_Customers'))))

# Means of RFM Segments

RFM\_Means=RFM\_.groupby('Customer\_Segments').mean()

RFM\_Means

# Size of Segments

segment\_size=RFM\_.groupby('Customer\_Segments').agg(SegmentSize=('Recency','count')).eval('SegmentProp= SegmentSize/SegmentSize.sum()')

segment\_size.sort\_values(by='SegmentSize',ascending=False)

segments\_table=df1.merge(RFM,left\_on='CustomerID',right\_index=True)

segments\_table.head()

segments\_table1= segments\_table.drop(columns=['Recency','Frequency','Monetary','R\_rank\_norm','F\_rank\_norm','M\_rank\_norm'])

segments\_table1.head()

table1=segments\_table1.groupby(['Customer\_Segments','Description'],as\_index=False).agg(QuantityBought=('Quantity','sum'))

table1

Top5Purchases=table1.sort\_values(["Customer\_Segments", "QuantityBought"],ascending=[True, False]).groupby("Customer\_Segments").head(5).set\_index("Customer\_Segments")

Top5Purchases.reset\_index(inplace=True)

temp =Top5Purchases[Top5Purchases['Customer\_Segments'].isin(['Top\_Customers','High\_Value\_Customers','Medium\_Value\_Customers'])]

g=sns.FacetGrid(temp,col='Customer\_Segments',sharex=False,aspect=1.1,height=6,col\_wrap=3,hue='Customer\_Segments',col\_order=['Top\_Customers','High\_Value\_Customers','Medium\_Value\_Customers'])

g.map\_dataframe(sns.barplot,x='Description',y='QuantityBought')

g.set\_axis\_labels('')

g.refline(y=temp['QuantityBought'].mean()) # Creating the mean line

g.set\_titles(col\_template='{col\_name}',size=20)

g.set\_xticklabels(rotation=90,fontsize=15)

g.set\_ylabels(size=20);

plt.show()

temp1 =Top5Purchases[Top5Purchases['Customer\_Segments'].isin(['Lost\_Customers', 'Low\_ValueCustomers'])]

g=sns.FacetGrid(temp1,col='Customer\_Segments',sharex=False,ylim=(0,30000),aspect=1.5,height=5,col\_wrap=2,hue='Customer\_Segments',col\_order=['Lost\_Customers', 'Low\_ValueCustomers'])

g.map\_dataframe(sns.barplot,x='Description',y='QuantityBought')

g.set\_axis\_labels('')

g.set\_titles(col\_template='{col\_name}',size=20)

g.set\_xticklabels(rotation=90,fontsize=15)

g.set\_ylabels(size=20);

plt.show()

rfm\_df = rfm1[['Recency','Frequency','Monetary']]

# Instantiate

scaler = StandardScaler()

# fit\_transform

rfm\_df\_scaled = scaler.fit\_transform(rfm\_df)

rfm\_df\_scaled.shape

rfm\_df\_scaled = pd.DataFrame(rfm\_df\_scaled)

rfm\_df\_scaled.columns = ['Amount', 'Frequency', 'Recency']

rfm\_df\_scaled.head()

# k-means with some arbitrary k

kmeans = KMeans(n\_clusters=4, max\_iter=50)

kmeans.fit(rfm\_df\_scaled)

# Elbow-curve/SSD

ssd = []

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)

ssd.append(kmeans.inertia\_)

# plot the SSDs for each n\_clusters

plt.plot(ssd)

# Silhouette analysis

range\_n\_clusters = [2, 3, 4, 5, 6, 7, 8]

for num\_clusters in range\_n\_clusters:

# intialise kmeans

kmeans = KMeans(n\_clusters=num\_clusters, max\_iter=50)

kmeans.fit(rfm\_df\_scaled)

cluster\_labels = kmeans.labels\_

# silhouette score

silhouette\_avg = silhouette\_score(rfm\_df\_scaled, cluster\_labels)

print("For n\_clusters={0}, the silhouette score is {1}".format(num\_clusters, silhouette\_avg))

# Final model with k=3

kmeans = KMeans(n\_clusters=3, max\_iter=50)

kmeans.fit(rfm\_df\_scaled)

# assign the label

rfm1['Cluster\_Id'] = kmeans.labels\_

rfm1.head()

sns.scatterplot(x='Cluster\_Id', y='Monetary', data=rfm1,color='red')

plt.title('Cluster ID vs Monetory')

plt.show()

from scipy.cluster.hierarchy import linkage, dendrogram,cophenet

from scipy.spatial.distance import pdist

mergings = linkage(rfm\_df\_scaled, method="single", metric='euclidean')

dendrogram(mergings)

plt.title('Single Linkage')

plt.show()

mergings = linkage(rfm\_df\_scaled, method="complete", metric='euclidean')

dendrogram(mergings)

plt.title('Complete')

plt.show()

# 3 clusters

cluster\_labels = cut\_tree(mergings, n\_clusters=3).reshape(-1, )

cluster\_labels

rfm1['Cluster\_Labels'] = cluster\_labels

rfm1.head()

sns.boxplot(x='Cluster\_Labels', y='Monetary', data=rfm1)

plt.title('Cluster label vs Monetory')

sns.boxplot(x='Cluster\_Labels', y='Frequency', data=rfm1)

plt.title('Cluster label vs Frequency')

sns.boxplot(x='Cluster\_Labels', y='Recency', data=rfm1)

plt.title('Cluster label vs Recency')