Data Science Capstone Project

Name of the project:

**Retail Store Data Analysis**

**Done by:**Mrinmay Saha

DESCRIPTION

* It is a critical requirement for business to understand the value derived from a customer. RFM is a method used for analyzing customer value.
* Customer segmentation is the practice of segregating the customer base into groups of individuals based on some common characteristics such as age, gender, interests, and spending habits
* Perform customer segmentation using RFM analysis. The resulting segments can be ordered from most valuable (highest recency, frequency, and value) to least valuable (lowest recency, frequency, and value).

**Dataset Description**

This is a transnational data set which contains all the transactions that occurred between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique and all-occasion gifts.

|  |  |
| --- | --- |
| **Variables** | **Description** |
| InvoiceNo | Invoice number. Nominal, a six digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation |
| StockCode | Product (item) code. Nominal, a five 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 | Invoice Date and time. Numeric, the day and time when each transaction was generated |
| UnitPrice | Unit price. Numeric, product price per unit in sterling |
| CustomerID | Customer number. Nominal, a six digit integral number uniquely assigned to each customer |
| Country | Country name. Nominal, the name of the country where each customer resides |

**Project Task: Week 1**

**Data Cleaning:**

1. Perform a preliminary data inspection and data cleaning.

* Check for missing data and formulate an apt strategy to treat them.
* Remove duplicate data records.
* Perform descriptive analytics on the given data.

**Data Transformation:**

1. Perform cohort analysis (a cohort is a group of subjects that share a defining characteristic). Observe how a cohort behaves across time and compare it to other cohorts.

* Create month cohorts and analyze active customers for each cohort.
* Analyze the retention rate of customers.

**Data Modeling :**

1. Build a RFM (Recency Frequency Monetary) model. Recency means the number of days since a customer made the last purchase. Frequency is the number of purchase in a given period. It could be 3 months, 6 months or 1 year. Monetary is the total amount of money a customer spent in that given period. Therefore, big spenders will be differentiated among other customers such as MVP (Minimum Viable Product) or VIP.
2. Calculate RFM metrics.
3. Build RFM Segments. Give recency, frequency, and monetary scores individually by dividing them into quartiles.

* Combine three ratings to get a RFM segment (as strings).
* Get the RFM score by adding up the three ratings.
* Analyze the RFM segments by summarizing them and comment on the findings.

**Note:**

* Rate “recency" for customer who has been active more recently higher than the less recent customer, because each company wants its customers to be recent.
* Rate “frequency" and “monetary" higher, because the company wants the customer to visit more often and spend more money.

**Source Code with graphs an insights:**

#importing required libaries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

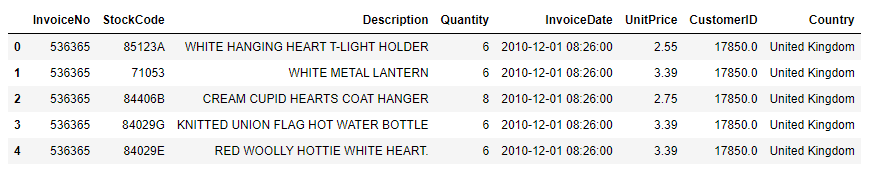
%matplotlib inline

#loading the data

df=pd.read\_excel('Online Retail.xlsx')

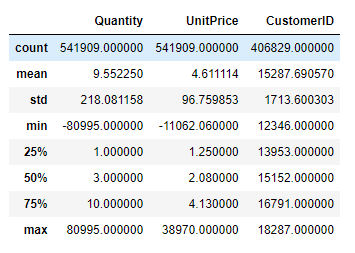
#checking the first 5 data

df.head()

Ouput:

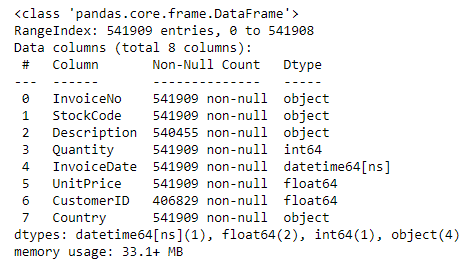
df.shape

Ouput: (541909, 8)

df.describe()

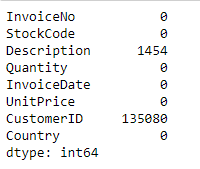
Ouput :

df.info()

Ouput:

# Data Cleaning

# Checking For missing Data and formulate an apt strategy to treat them.

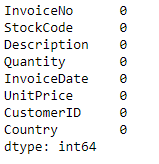
df.isnull().sum()

Output:

#removing the null values fro CustomerID columns

df.dropna(subset=["CustomerID"],inplace=True)

#checking if the null values are removed or not

df.isnull().sum()

Output:

Insight: We can confirm from the above observation that all the null values are dropped.

b) Removing duplicate data records.

#checking number of duplicate entries

df.duplicated().sum()

Output: 5225

#removing the duplicate entires

df.drop\_duplicates(inplace=True)

#verifying if the dupliacte entires are removed

df.duplicated().sum()

Output: 0

# Removing records that have quantity value in negative

df=df[df['Quantity']>0]

df.describe()

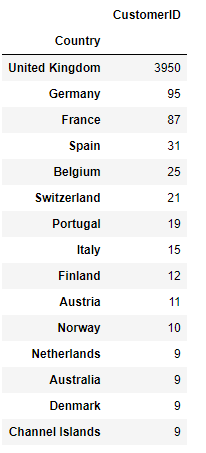
Output:

# 

# Performing descriptive analytics on the given data.

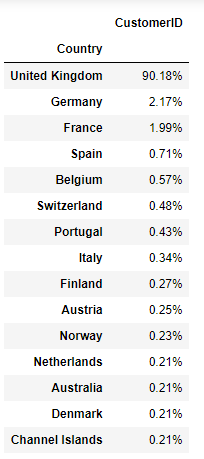
#lets observe the first 15 distrubution of customers per country

df.groupby('Country')['CustomerID'].nunique().sort\_values(ascending=False).to\_frame().head(15)

Ouput:

#checking the percentage of the customer from different Countries

percentage=(df.groupby('Country')['CustomerID'].nunique()/df.groupby('Country')['CustomerID'].nunique().sum()).mul(100).round(2).sort\_values(ascending=False).to\_frame().head(15).astype(str)+'%'

percentage

Ouput:

#Visualize the Items contributing to maximum Price Value

df['TotalPrice'] = df.Quantity \* df.UnitPrice

data =df.TotalPrice.sort\_values(ascending=False).head(10).to\_frame().style.hide\_index()

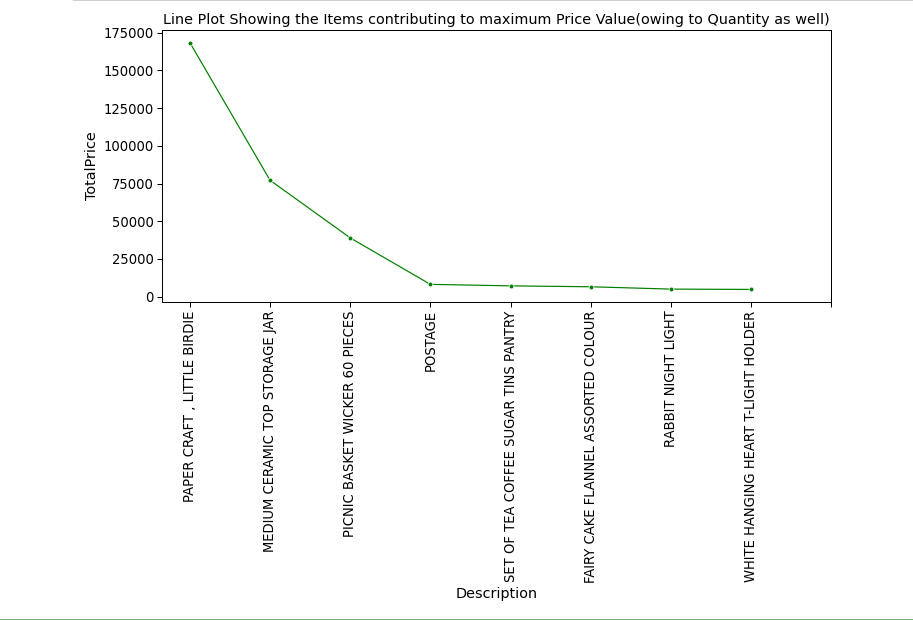
desc = df.sort\_values(by='TotalPrice', ascending=False)['Description'].head(10)

price = df.sort\_values(by='TotalPrice', ascending=False)['TotalPrice'].head(10)

plt.figure(figsize=(12,5))

sns.lineplot(y=price,x=desc, marker='o', color='g',).set\_title('Line Plot Showing the Items contributing to maximum Price Value(owing to Quantity as well)')

plt.xticks(range(0,9), rotation=90)

plt.show();

Output:

#Let's visualize some top products from the whole range.

top\_products = df['Description'].value\_counts()[:20]

plt.figure(figsize=(12,6))

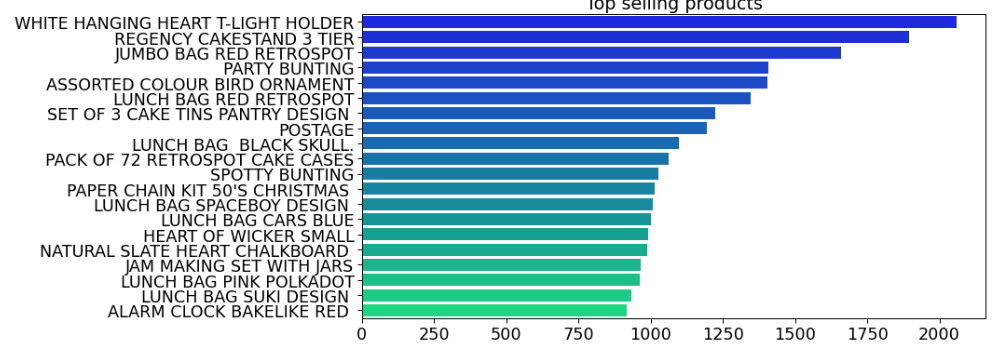
sns.set\_context("paper", font\_scale=2)

sns.barplot(y = top\_products.index,

x = top\_products.values,palette='winter')

plt.title("Top selling products")

plt.show();

Output:

# Data Transformation

# Perform cohort analysis (a cohort is a group of subjects that share a defining characteristic). Observe how a cohort behaves across time and compare it to other cohorts.

# Define a function that will parse the date

import datetime

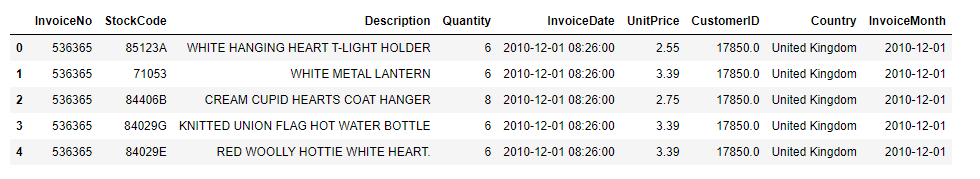
def get\_month(x):

return datetime.datetime(x.year,x.month,1)

# Create InvoiceMonth column

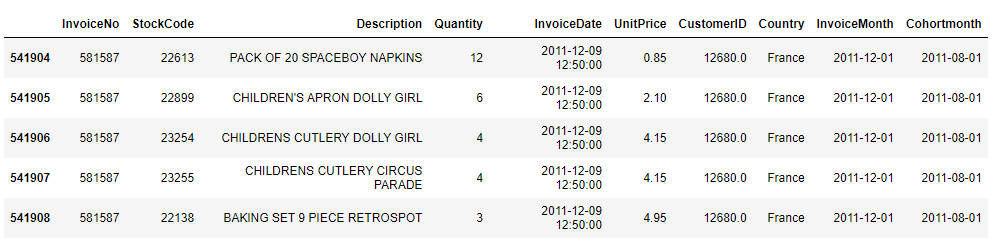
df['InvoiceMonth'] = df['InvoiceDate'].apply(get\_month)

df.head()

Output:

df['Cohortmonth']=df.groupby(['CustomerID'])['InvoiceMonth'].transform('min')

df.tail()

Output:

def get\_date\_int(df, column):

year = df[column].dt.year

month = df[column].dt.month

return year, month

# Get the integers for date parts from the `InvoiceMonth` column

invoice\_year, invoice\_month = get\_date\_int(df,'InvoiceMonth')

# Get the integers for date parts from the `CohortMonth` column

cohort\_year, cohort\_month = get\_date\_int(df,'Cohortmonth')

# Calculate difference in years

years\_diff = invoice\_year - cohort\_year

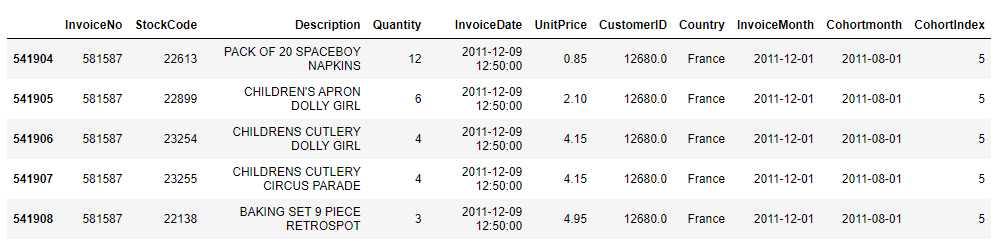
# Calculate difference in months

months\_diff = invoice\_month - cohort\_month

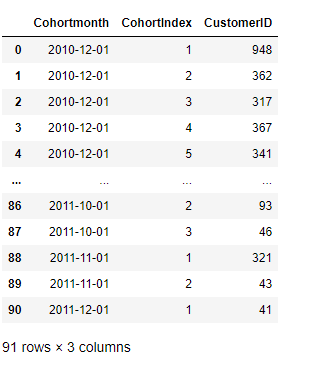
# Extract the difference in months from all previous values

df['CohortIndex'] = years\_diff \* 12 + months\_diff + 1

df.tail()

Output:

cohort\_data=df.groupby(['Cohortmonth','CohortIndex'])['CustomerID'].nunique().reset\_index()

cohort\_data

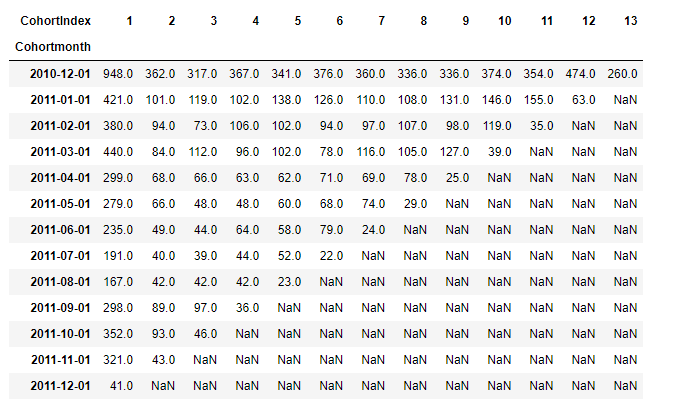
# Analyze the retention rate of customers.

#creating pivot table

cohort\_table=cohort\_data.pivot\_table(index='Cohortmonth',columns=['CohortIndex'],values='CustomerID')

cohort\_table

Output:



#representing the cohort table in percentage form

cohot\_table\_perc=cohort\_table.divide(cohort\_table.iloc[:,0],axis=0)

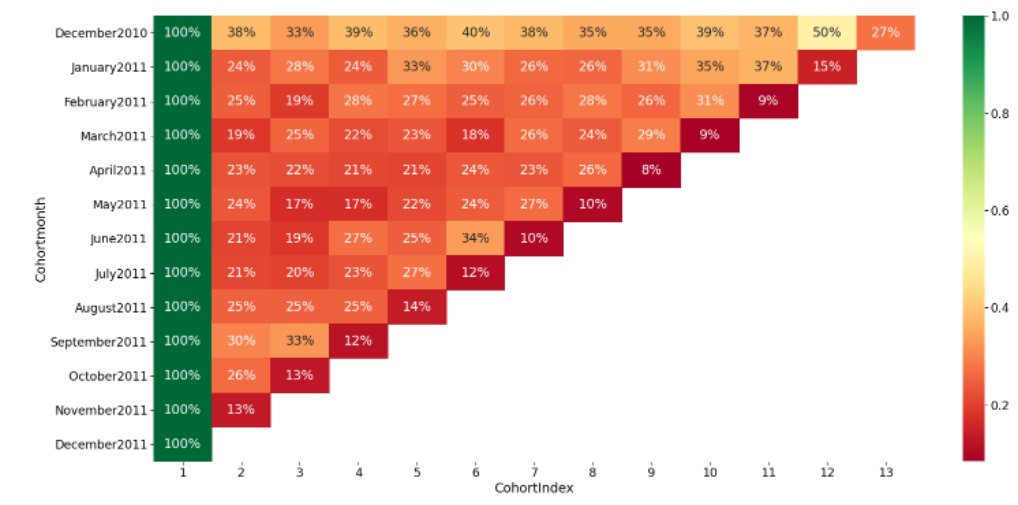
#changing index

cohort\_table.index=cohort\_table.index.strftime('%B''%Y')

#visualize the result as heatmap

plt.figure(figsize=(25,12))

sns.heatmap(cohot\_table\_perc,annot=True,cmap='RdYlGn',fmt='.0%')

Output:

Insight:From the above heatmap we can see the percentage of the customers that are retained every month.

# Data Modeling :

# Build a RFM (Recency Frequency Monetary) model.

#Importing the required libaries

import datetime as dt

df['InvoiceDate'].max()

Output: Timestamp('2011-12-09 12:50:00')

#Setting the latest date as 2011-12-10 as the max date of dataset is 2011-12-09.

current\_date=dt.datetime(2011,12,10)

df['Total\_price']=df.Quantity\*df.UnitPrice

RFM\_table=df.groupby('CustomerID').agg({'InvoiceDate': lambda x: (current\_date - x.max()).days,

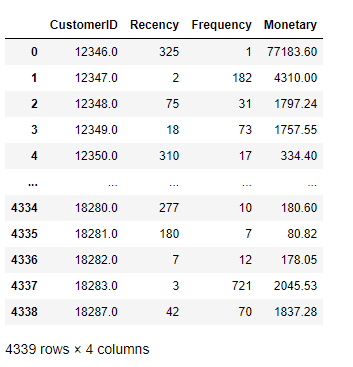
'InvoiceNo': lambda x: len(x),'Total\_price': lambda x: x.sum()})

RFM\_table=RFM\_table.rename(columns={'InvoiceDate': 'Recency',

'InvoiceNo': 'Frequency',

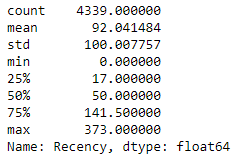
'Total\_price': 'Monetary'})

RFM\_table.reset\_index()

Output:

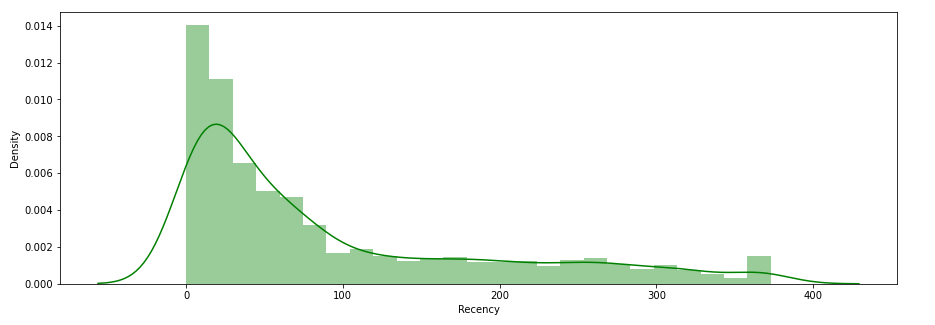
#Perforing the descripe analysis on date to understand it better

RFM\_table.Recency.describe()



#frequency distribution plot

plt.figure(figsize=(15,5))

ax=sns.distplot(RFM\_table['Recency'],color='g')

#Perforing the descripe analysis on date to understand it better

RFM\_table.Frequency.describe()

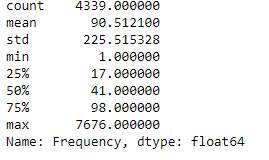
#frequency distribution plot

plt.figure(figsize=(15,5))

ax=sns.distplot(RFM\_table['Frequency'],color='g')

#Perforing the descripe analysis on date to understand it better

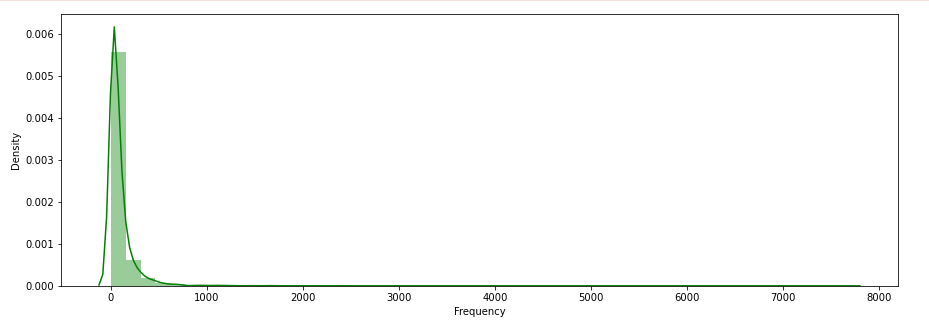
RFM\_table.Frequency.describe()



#frequency distribution plot

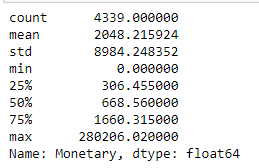
plt.figure(figsize=(15,5))

ax=sns.distplot(RFM\_table['Frequency'],color='g')



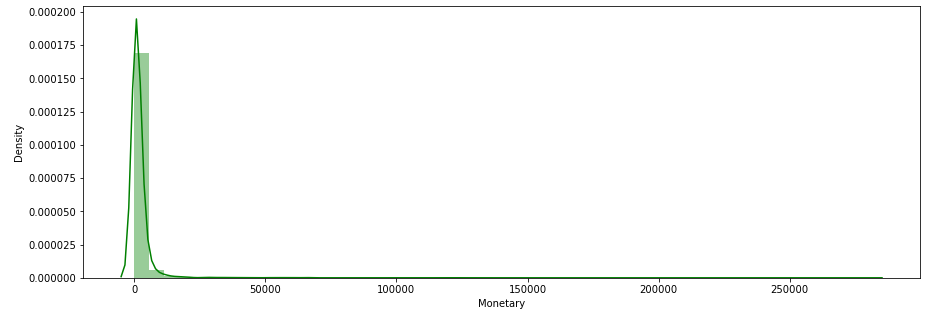
#Perforing the descripe analysis on date to understand it better

RFM\_table.Monetary.describe()



#frequency distribution plot

plt.figure(figsize=(15,5))

ax=sns.distplot(RFM\_table['Monetary'],color='g')

#Dividing the RFM in quantiles

quant=RFM\_table.quantile(q=[0.25,0.5,0.75])

quant.to\_dict()

Output:{'Recency': {0.25: 17.0, 0.5: 50.0, 0.75: 141.5},

'Frequency': {0.25: 17.0, 0.5: 41.0, 0.75: 98.0},

'Monetary': {0.25: 306.45500000000004,

0.5: 668.5600000000002,

0.75: 1660.315}}

Assiging RFM score

def recencyscore(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

def fmscore(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:

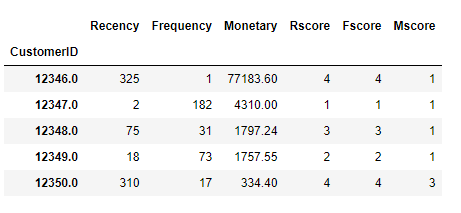
return 1

RFM\_table['Rscore'] = RFM\_table['Recency'].apply(recencyscore, args=('Recency',quant,))

RFM\_table['Fscore'] = RFM\_table['Frequency'].apply(fmscore, args=('Frequency',quant,))

RFM\_table['Mscore'] = RFM\_table['Monetary'].apply(fmscore, args=('Monetary',quant,))

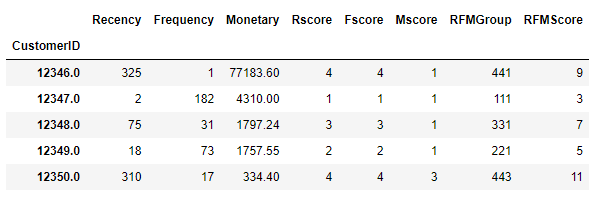
RFM\_table.head()



RFM\_table['RFMGroup'] = RFM\_table.Rscore.map(str)+RFM\_table.Fscore.map(str)+RFM\_table.Mscore.map(str)

RFM\_table['RFMScore']=RFM\_table[['Rscore','Fscore','Mscore']].sum(axis=1)

RFM\_table.head()

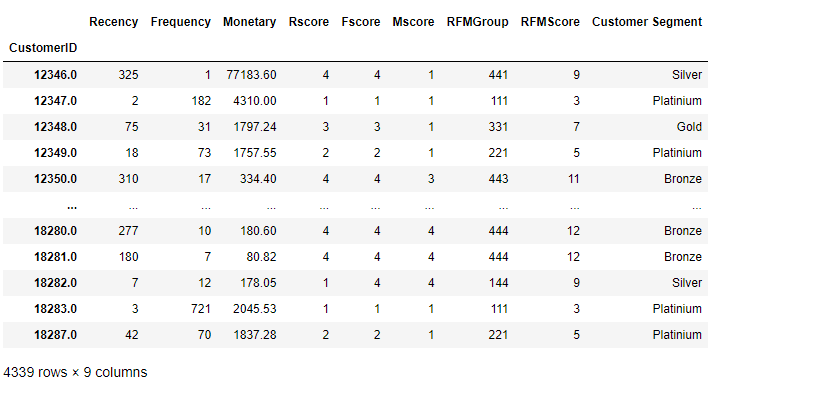


#Assiging customers to each sectionsas per the loyality

Segmentaion=['Platinium','Gold','Silver','Bronze']

cuts=pd.qcut(RFM\_table.RFMScore,q=4,labels=Segmentaion)

RFM\_table['Customer Segment']=cuts.values

RFM\_table

#Describing the RFMscore as per the customer behavior

pd.set\_option("display.max\_colwidth", 10000)

data = {'Customer Segement':['Best Customers', 'Loyal Customers', 'Big Spender', 'Almost Lost','Lost Customers','Lost Cheap Customers'],

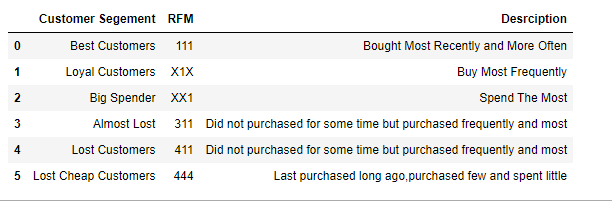
'RFM':['111', 'X1X', 'XX1', '311','411','444'],

'Desrciption':['Bought Most Recently and More Often', 'Buy Most Frequently', 'Spend The Most',

'Did not purchased for some time but purchased frequently and most',

'Did not purchased for some time but purchased frequently and most','Last purchased long ago,purchased few and spent little']}

pd.DataFrame(data)



**Insight:** In the above chart we have sorted the customer as per the different RFM score and given the details about which are the best and loyal customers and which customers are in the verge of churning so we now the company can take the strategies as per the date so that the customers can be retained.

**Data Modeling :**

1. Create clusters using k-means clustering algorithm.

* Prepare the data for the algorithm. If the data is asymmetrically distributed, manage the ske
* wness with appropriate transformation. Standardize the data.
* Decide the optimum number of clusters to be formed.
* Analyze these clusters and comment on the results.

1. Create a dashboard in tableau by choosing appropriate chart types and metrics useful for the business. The dashboard must entail the following:

* Country-wise analysis to demonstrate average spend. Use a bar chart to show the monthly figures
* Bar graph of top 15 products which are mostly ordered by the users to show the number of products sold
* Bar graph to show the count of orders vs. hours throughout the day
* Plot the distribution of RFM values using histogram and frequency charts
* Plot error (cost) vs. number of clusters selected
* Visualize to compare the RFM values of the clusters using heatmap

# Week 2

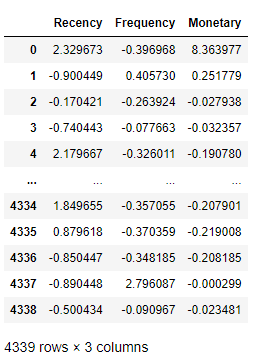
# Create clusters using k-means clustering algorithm.

#importing required libaries

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

cluster=RFM\_table

cluster=cluster.reset\_index(level=0).iloc[:,[1,2,3]]

cluster

#Standarding the data

sc=StandardScaler()

standard=sc.fit\_transform(cluster)

standard= pd.DataFrame(data=standard, index=cluster.index, columns=cluster.columns)

standard

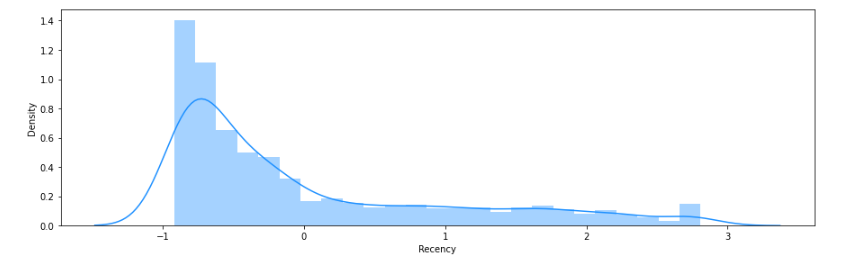
# Distribution plot

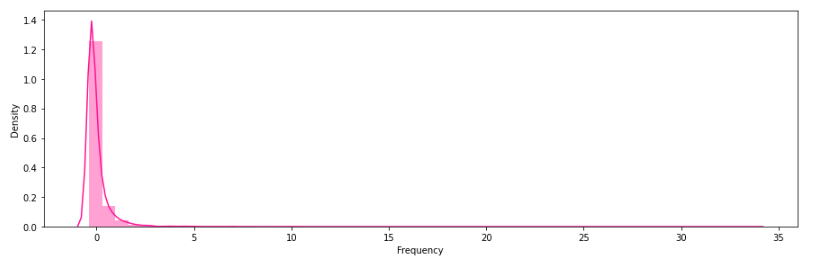
fig, axes = plt.subplots(3, 1, figsize=(15, 15))

sns.distplot(standard.Recency , color="dodgerblue", ax=axes[0], axlabel='Recency')

sns.distplot(standard.Frequency , color="deeppink", ax=axes[1], axlabel='Frequency')

sns.distplot(standard.Monetary , color="gold", ax=axes[2], axlabel='Monetary')

plt.show();



# 

# Insight: We can observe that the means & averages are approximately uniformed now in each distribution. Now the data can be used for unsupervised algorithm i.e. K-Means

# Decide the optimum number of clusters to be formed.

#using elobow method to fing the number of cluster

wcss = []

for i in range(1, 11):

kmeans = KMeans(n\_clusters = i, init = 'k-means++')

kmeans.fit(standard)

wcss.append(kmeans.inertia\_)

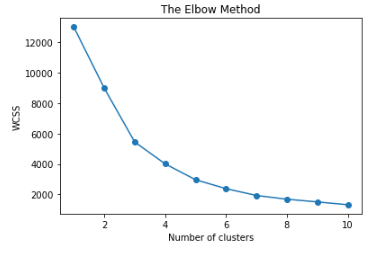
plt.plot(range(1, 11), wcss,marker='o')

plt.title('The Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel('WCSS')

plt.show()



# Insight: From the above graph we can see the optimum value for the number of cluster will be n\_cluster=4.

# Analyze these clusters and comment on the results.

#intizating the Kmeans algorithim

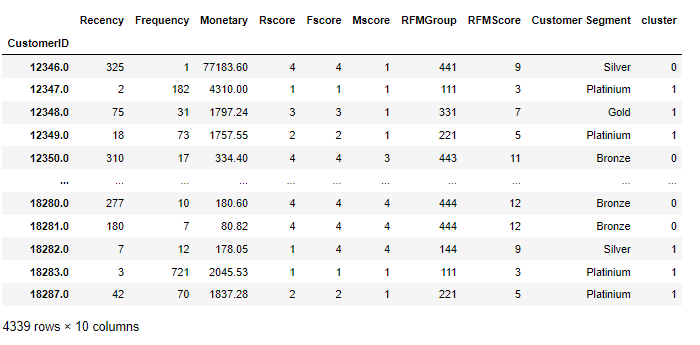
kmeans = KMeans(n\_clusters=4, random\_state=1, init='k-means++')

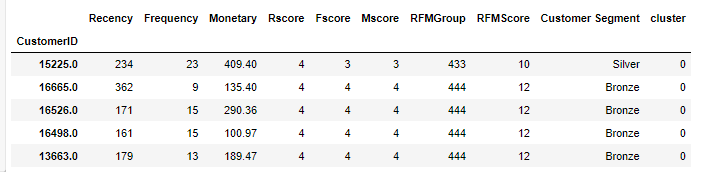
y\_pred=kmeans.fit(standard)

cluster\_labels = kmeans.labels\_

RFM\_table['cluster']=cluster\_labels

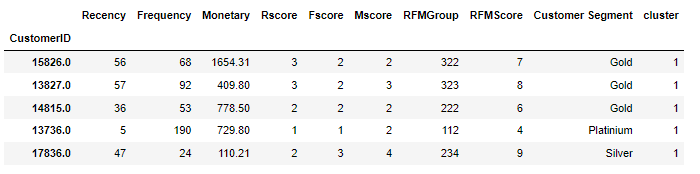
RFM\_table



RFM\_table[RFM\_table['cluster']==0].smaple(5)

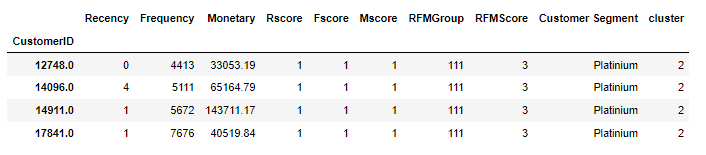
**Insight:** Cluster 0 contains the number of customers who accounts for lowest value to the firm because there RFM values are highest. Most of them are in the lost segment or on the verge of churning out

RFM\_table[RFM\_table['cluster']==1].sample(5)



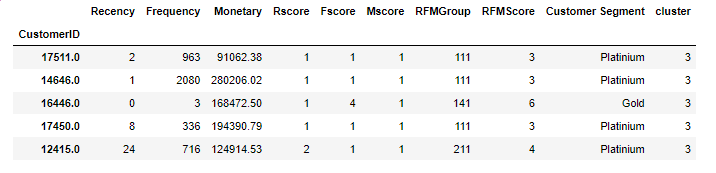
**Insight:** Cluster 1 is somewhat average collectively can respond to the targeted campaigns.

RFM\_table[RFM\_table['cluster']==2].



**Insight:** Here it can be seen that the RFM score for Cluster 2 customers with low recency, good frequency and high monetary value, These are the loyal customers to the firm.

RFM\_table[RFM\_table['cluster']==3].sample(5)



**Insight:** Cluster 3 with very high monetary value along with good frequency and recency values. These are the most valuable customers to the firm. They should be looked after periodically to access there concerns

# Plotting two dimesional plots of each attributes respectively.

X = standard.iloc[:,0:3].values

count=X.shape[1]

for i in range(0,count):

for j in range(i+1,count):

plt.figure(figsize=(15,6));

plt.scatter(X[cluster\_labels == 0, i], X[cluster\_labels == 0, j], s = 10, c = 'red', label = 'Cluster0')

plt.scatter(X[cluster\_labels == 1, i], X[cluster\_labels == 1, j], s = 10, c = 'blue', label = 'Cluster1')

plt.scatter(X[cluster\_labels == 2, i], X[cluster\_labels == 2, j], s = 10, c = 'green', label = 'Cluster2')

plt.scatter(X[cluster\_labels == 3, i], X[cluster\_labels == 3, j], s = 10, c = 'cyan', label = 'Cluster3')

plt.scatter(kmeans.cluster\_centers\_[:,i], kmeans.cluster\_centers\_[:,j], s = 50, c = 'black', label = 'Centroids')

plt.xlabel(standard.columns[i])

plt.ylabel(standard.columns[j])

plt.legend()

plt.show();

## User Interactive Online Retail Story Board for Retail Store

### 1. Retail Dashboard

a. Country Wise Analysis

b. Top Products by Sales

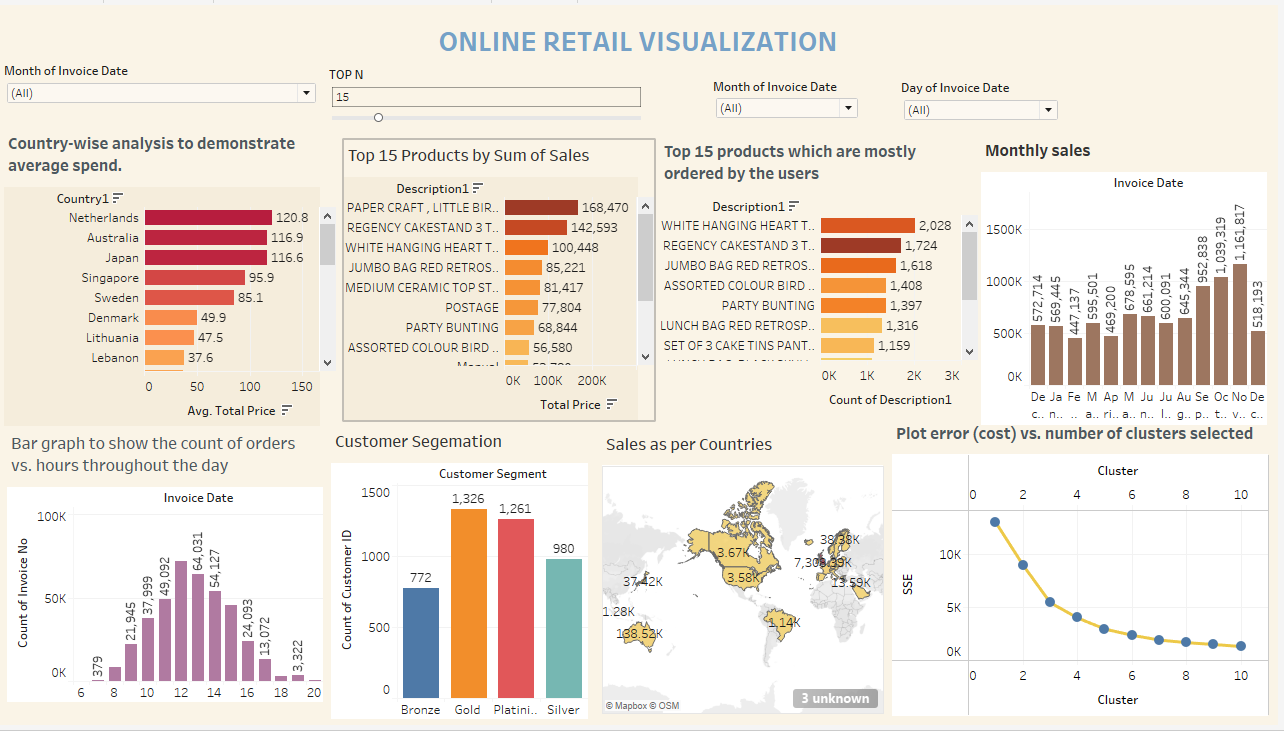
c. Top Products by Count

d. Monthly Figures

e. Count of orders Vs Hours Throughout the Day

f. Customer Segments

g. Geaographical Viz

 h. Elbow Plot -Error Cost against the no of clusters

### 2. RFM Cluster Analysis Dashboard

a. FM Heat Map

b.RM Heat Map

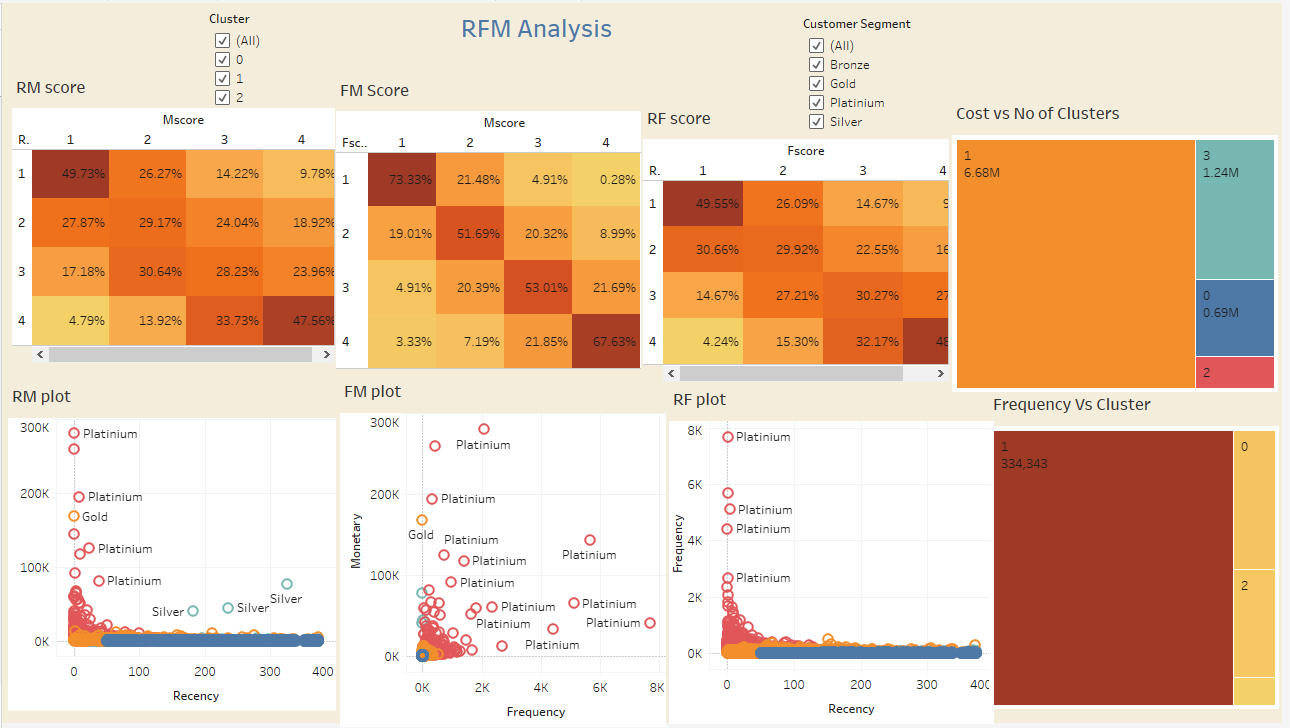
c. RF Heat Map

d. Cost Vs No of clusters

e. RF Plot

f. FM Plot

g. RM Plot

 h. Frequency Sum Vs Clusters

Here is the link to access the Tableau Dashboards: [Click here to Tableau Visualization](https://public.tableau.com/app/profile/mrinmay.saha/viz/Retail_16527085645850/RetailVisualizationStory?publish=yes)