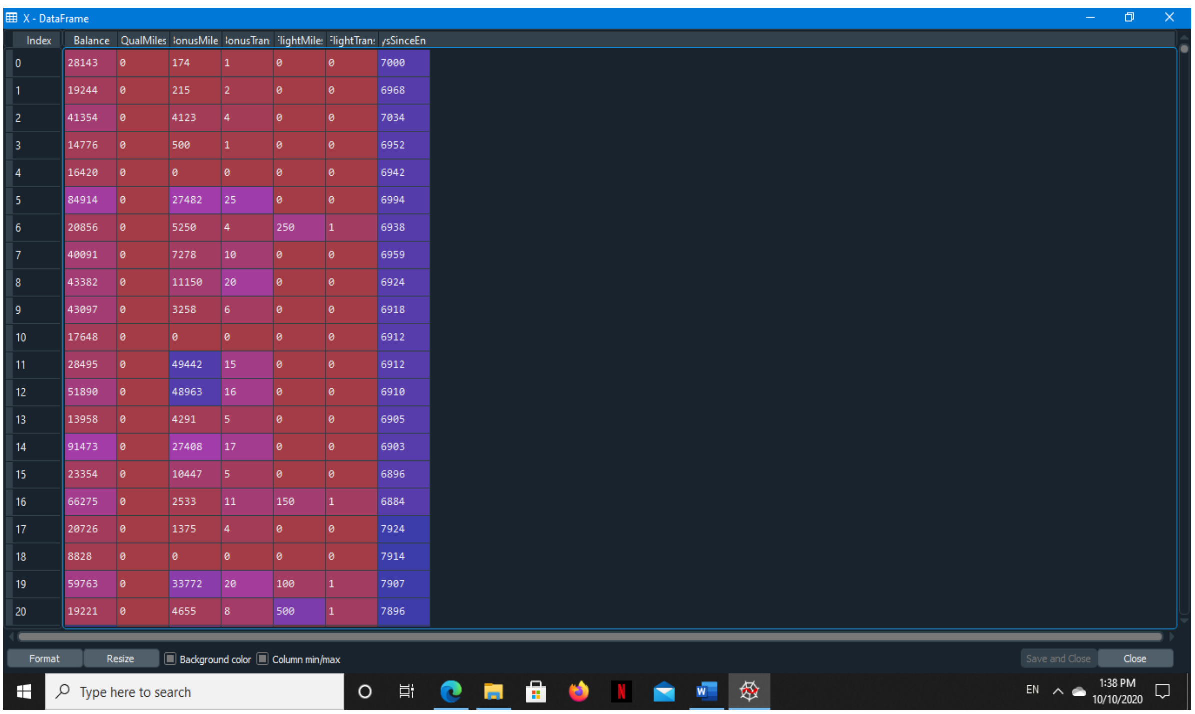
1.) Perform clustering (Both hierarchical and K means clustering) for the airlines data to obtain optimum number of clusters.

Draw the inferences from the clusters obtained.

In this Assignment first we have to load the data set using some library and the data set is in the form of excel and load that dataset using Common Separated values (CSV).

```
import numpy as np
import pandas as pd
import matplotlib. pyplot as plt
import seaborn as sns

df = pd. read_csv("AirlinesCluster.csv")
```

Output:



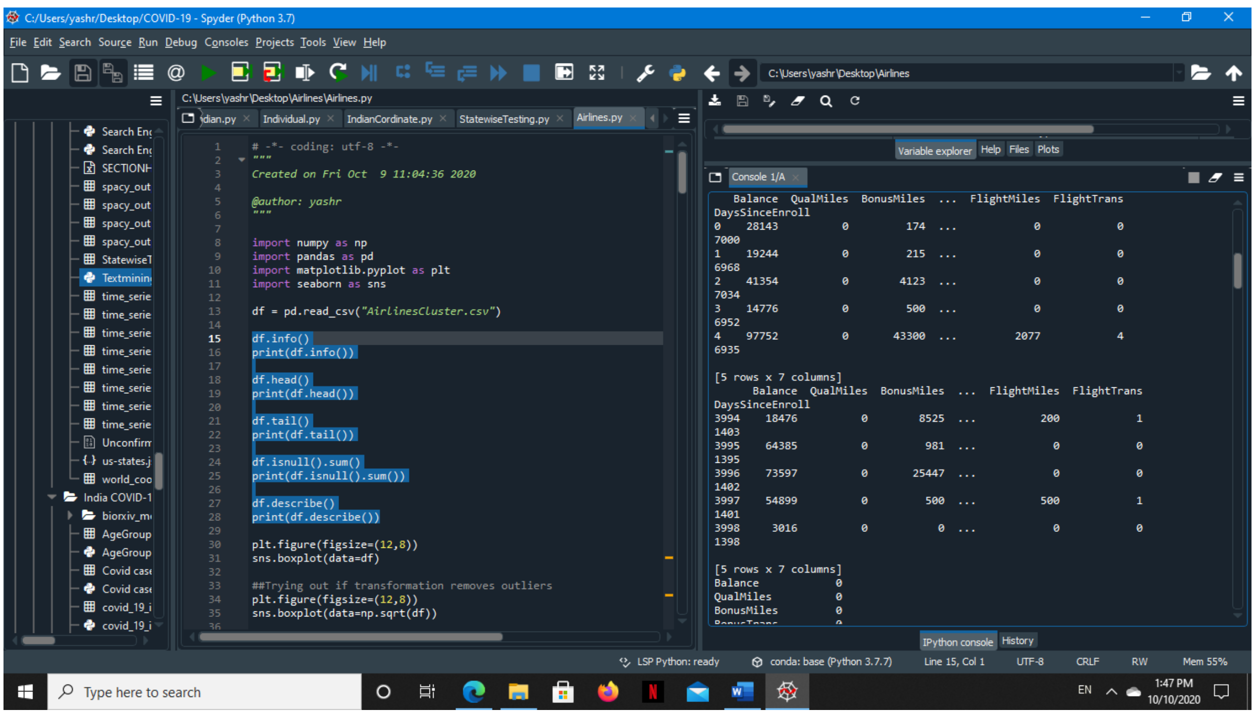
Index	Balance	QualMiles	ionusMile	ionusTran	lightMile	lightTran	ysSinceEn
0	28143	0	174	1	0	0	7000
1	19244	0	215	2	0	0	6968
2	41354	0	4123	4	0	0	7034
3	14776	0	500	1	0	0	6952
4	16420	0	0	0	0	0	6942
5	84914	0	27482	25	0	0	6994
6	20856	0	5250	4	250	1	6938
7	40091	0	7278	10	0	0	6959
8	43382	0	11150	20	0	0	6924
9	43097	0	3258	6	0	0	6918
10	17648	0	0	0	0	0	6912
11	28495	0	49442	15	0	0	6912
12	51890	0	48963	16	0	0	6910
13	13958	0	4291	5	0	0	6905
14	91473	0	27408	17	0	0	6903
15	23354	0	10447	5	0	0	6896
16	66275	0	2533	11	150	1	6884
17	20726	0	1375	4	0	0	7924
18	8828	0	0	0	0	0	7914
19	59763	0	33772	20	100	1	7907
20	19221	0	4655	8	500	1	7896

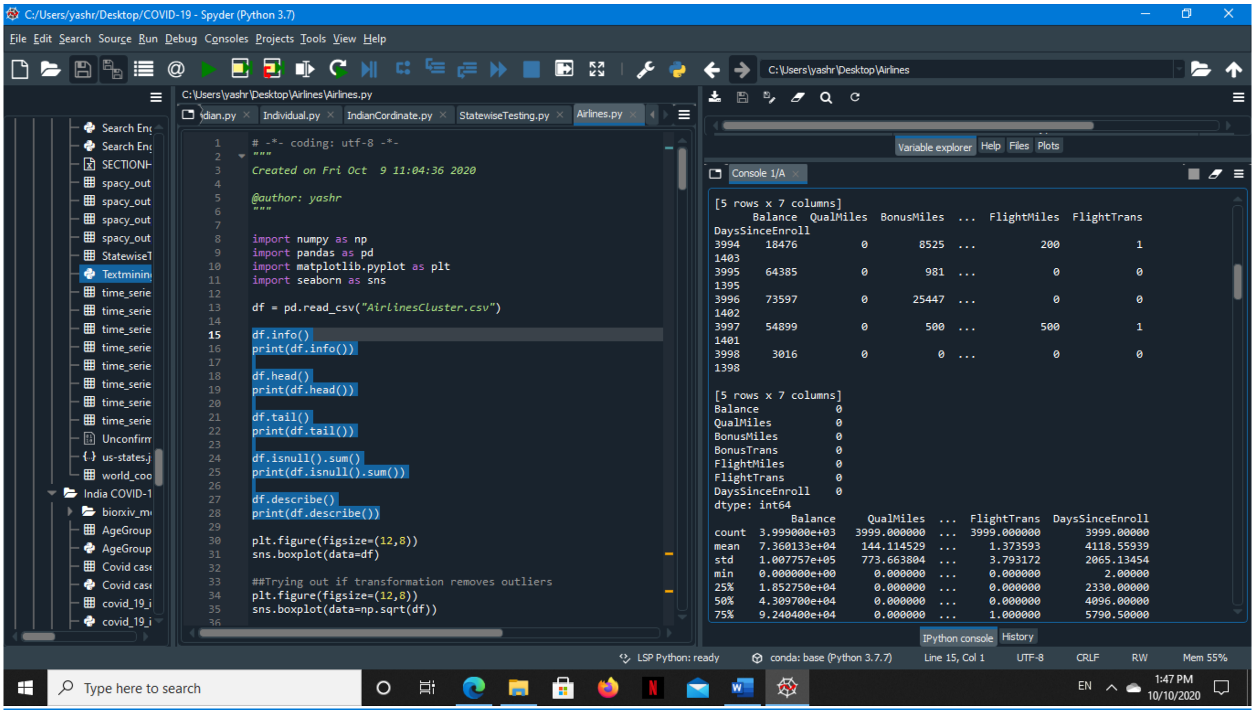
Now we have to perform some data preprocessing technique to access and clean the data set.

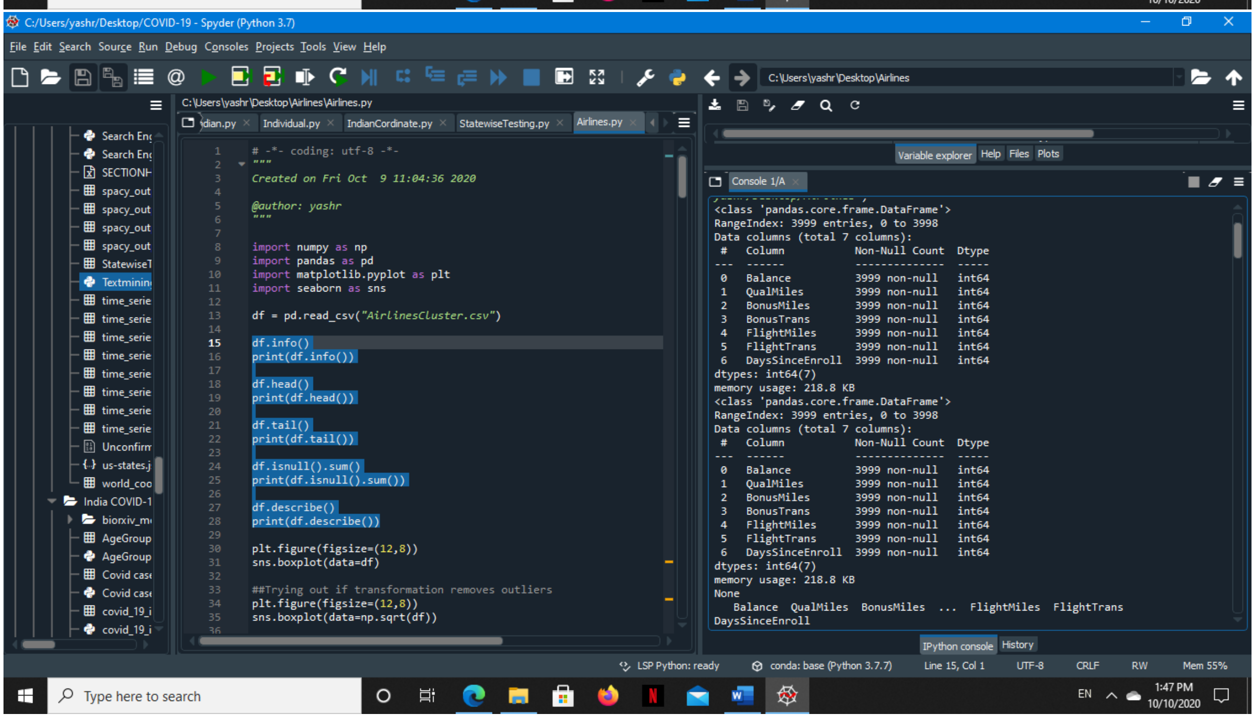
```
df.info ()
print (df.info ())
```

```
df. head ()
print (df. head ())

df. tail ()
print (df. tail ())

df. isnull (). sum ()
print (df. isnull (). sum ())

df. describe ()
print (df. describe ())
```

Output:





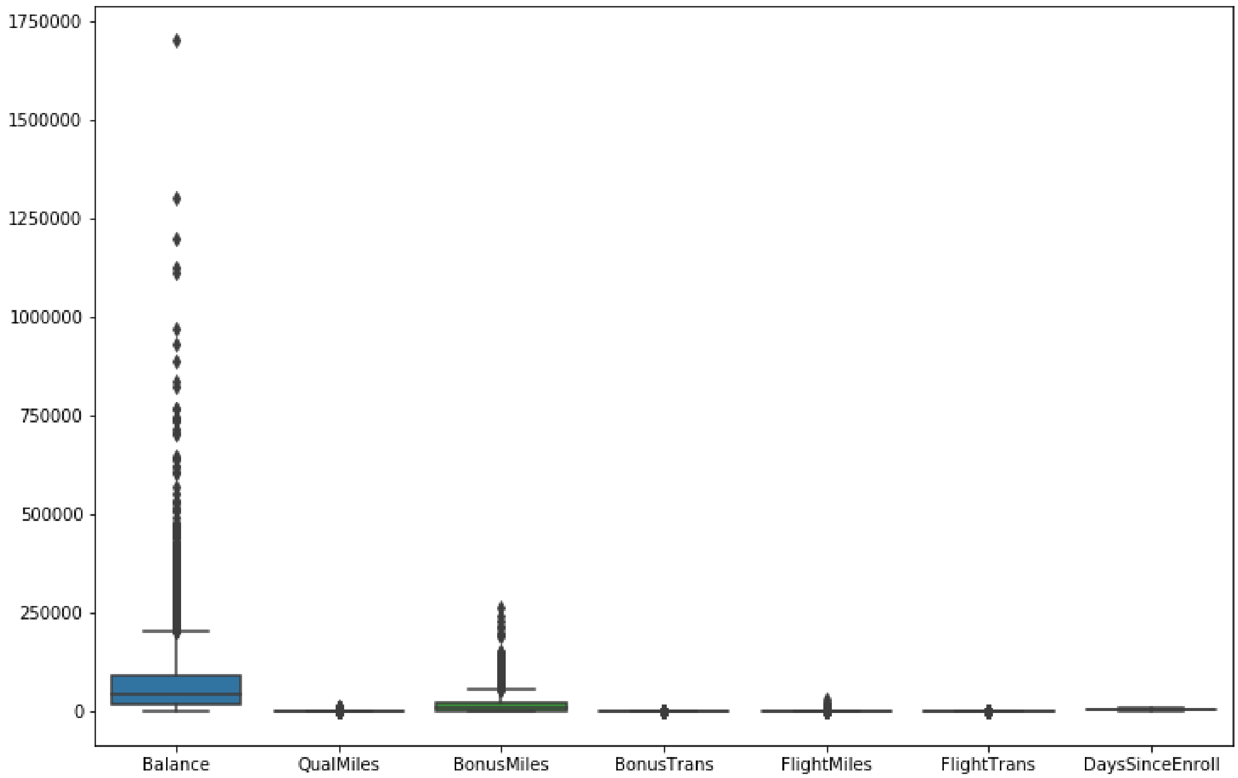


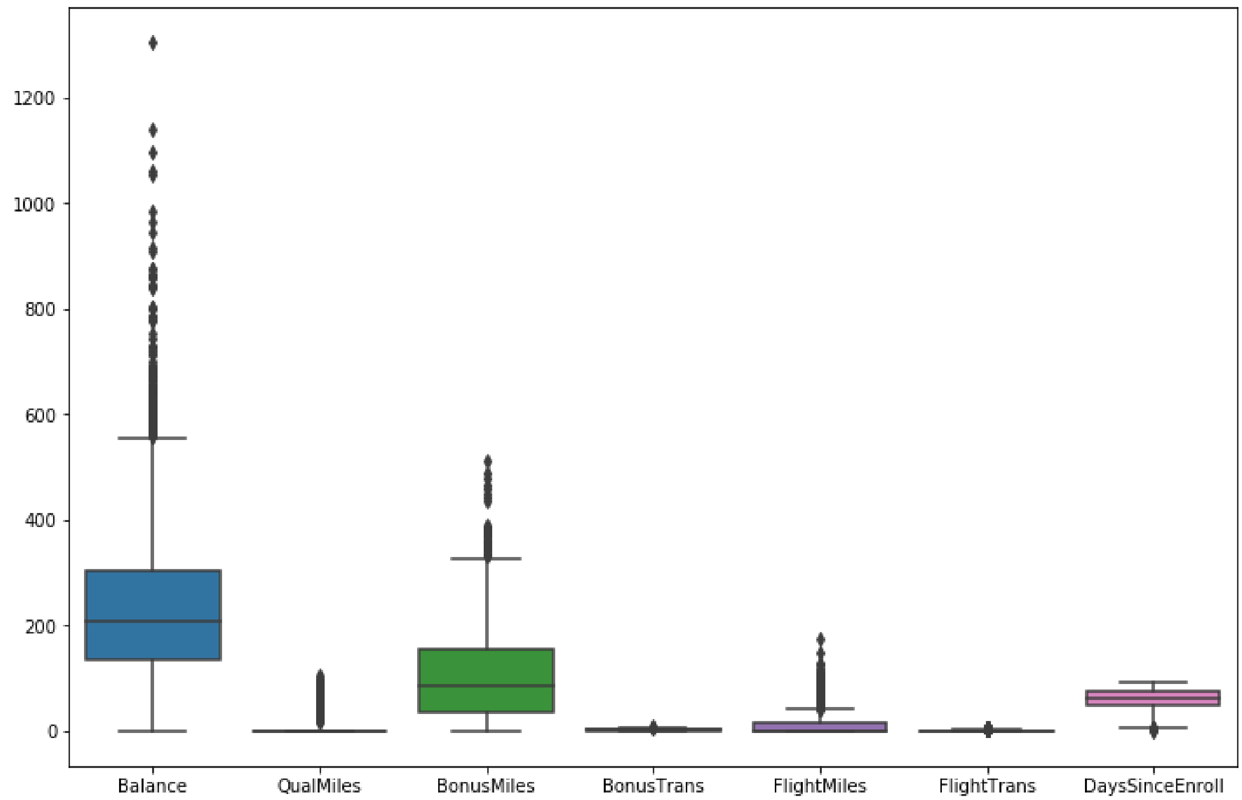
Now we have to perform some visualization to understand the dataset by using some best graphs.

plt. Figure (fig size= (12,8))
sns. Boxplot(data=df)

##Trying out if transformation removes outliers
plt. Figure (fig size= (12,8))
sns. Boxplot (data=np. sqrt(df))

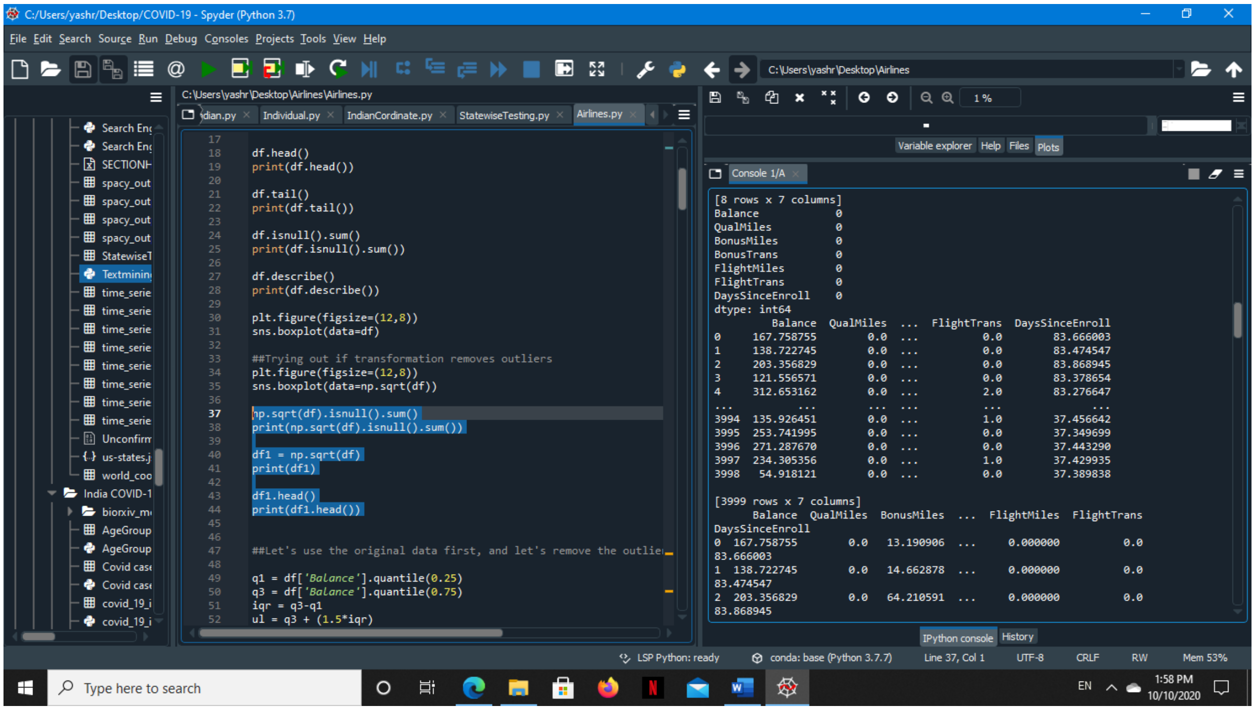
Output:





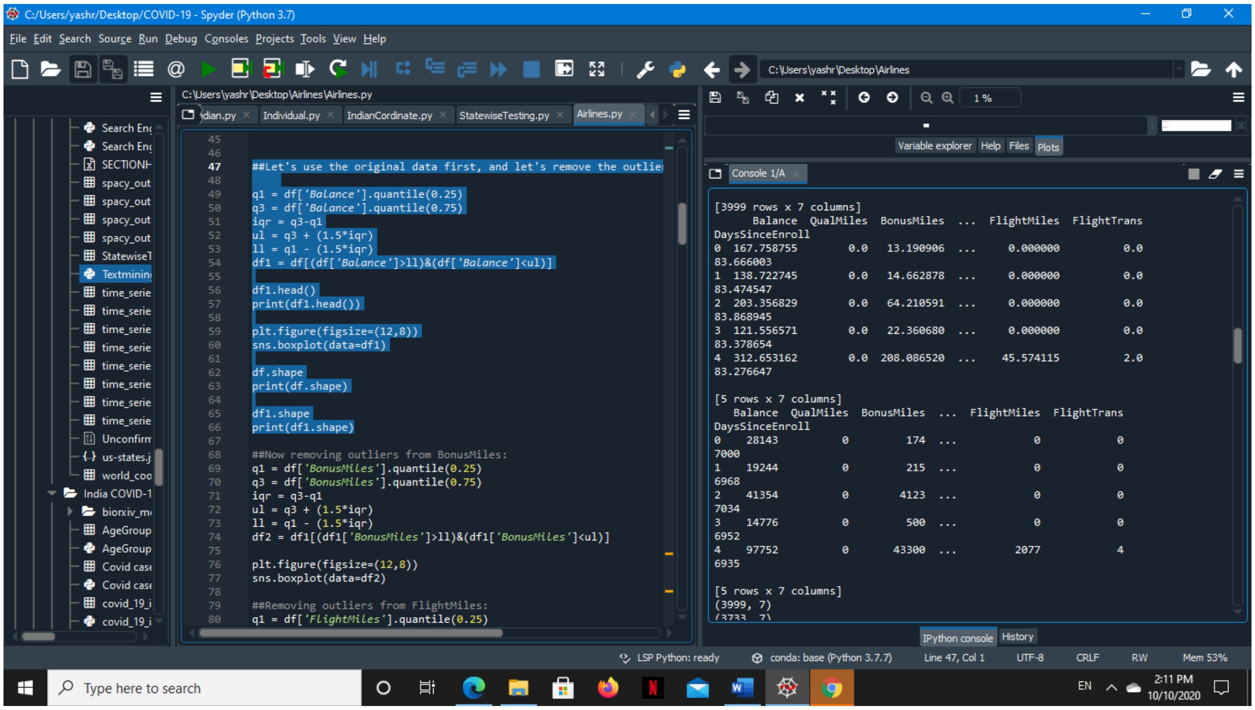

```
np. sqrt(df). isnull (). sum ()
print (np. sqrt(df). isnull (). sum ())

df1 = np. sqrt(df)
print(df1)

df1.head()
print (df1.head())
```

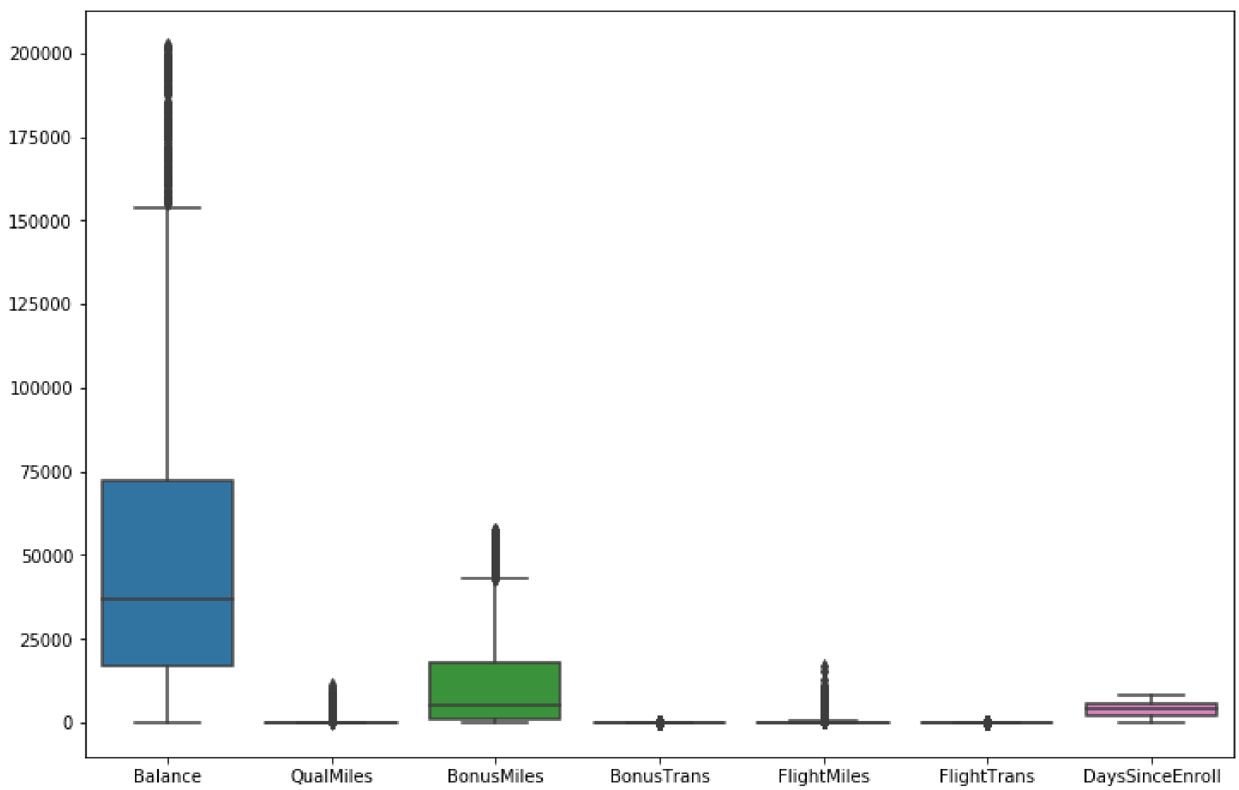
Output:

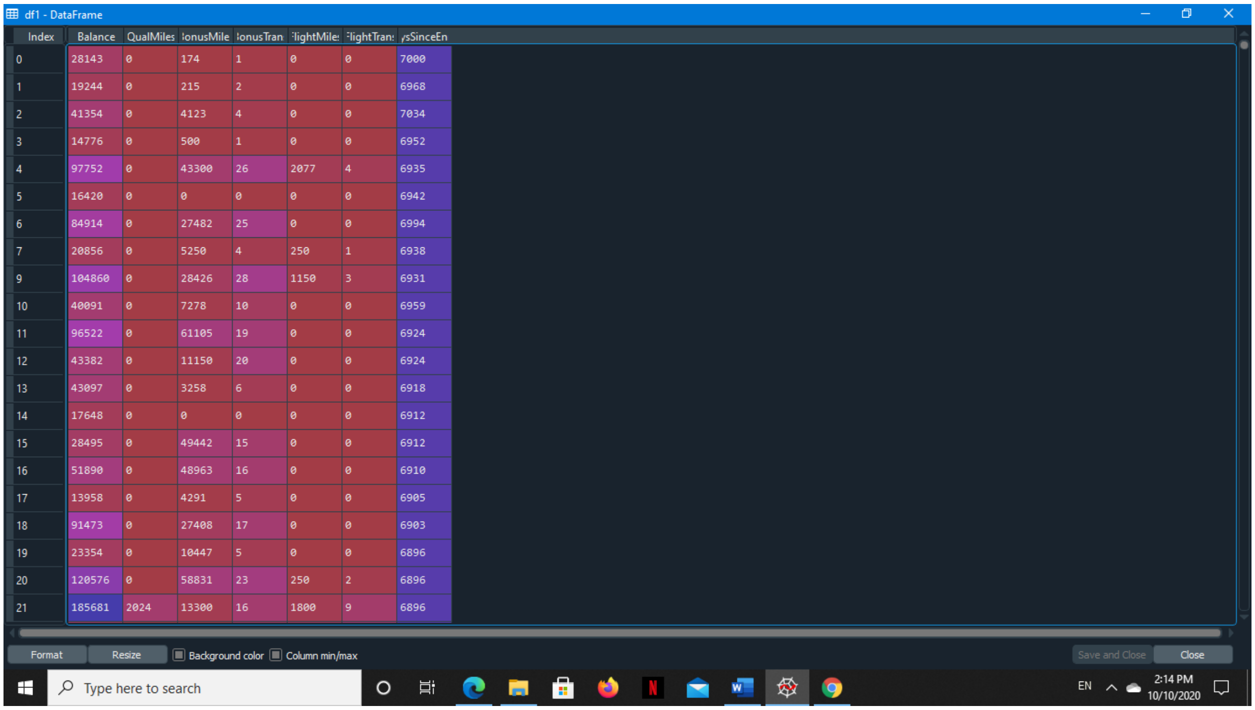


##Let's use the original data first, and let's remove the outliers from Balance first:

```
q1 = df['Balance']. quantile (0.25)
q3 = df['Balance']. quantile (0.75)
iqr = q3-q1
ul = q3 + (1.5*iqr)
ll = q1 - (1.5*iqr)
df1 = df[(df['Balance']>ll) &(df['Balance'] <ul)]

df1.head()
print (df1.head())

plt. Figure (fig size= (12,8))
sns. Boxplot(data=df1)

df. shape
print (df. shape)

df1.shape
print (df1.shape)
```


C:/Users/yashr/Desktop/COVID-19 - Spyder (Python 3.7)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\yashr\Desktop\Airlines\Airlines.py
Individual.py
IndianCordinate.py
StatewiseTesting.py
Airlines.py
##Let's use the original data first, and let's remove the outlie
q1 = df['Balance'].quantile(0.25)
q3 = df['Balance'].quantile(0.75)
iqr = q3-q1
ul = q3 + (1.5*iqr)
ll = q1 - (1.5*iqr)
df1 = df[(df['Balance']>ll)&(df['Balance']<ul)]
df1.head()
print(df1.head())
plt.figure(figsize=(12,8))
sns.boxplot(data=df1)
df.shape
print(df.shape)
df1.shape
print(df1.shape)
##Now removing outliers from BonusMiles:
q1 = df['BonusMiles'].quantile(0.25)
q3 = df['BonusMiles'].quantile(0.75)
iqr = q3-q1
ul = q3 + (1.5*iqr)
ll = q1 - (1.5*iqr)
df2 = df1[(df1['BonusMiles']>ll)&(df1['BonusMiles']<ul)]
plt.figure(figsize=(12,8))
sns.boxplot(data=df2)
##Removing outliers from FlightMiles:
q1 = df['FlightMiles'].quantile(0.25)
Variable explorer
Help
Files
Plots
Console 1/A
[3999 rows x 7 columns]
Balance QualMiles BonusMiles ... FlightMiles FlightTrans
DaysSinceEnroll
0 167.758755 0.0 13.190906 ... 0.000000 0.0
83.666003
1 138.722745 0.0 14.662878 ... 0.000000 0.0
83.474547
2 203.356829 0.0 64.210591 ... 0.000000 0.0
83.868945
3 121.556571 0.0 22.360680 ... 0.000000 0.0
83.378654
4 312.653162 0.0 208.086520 ... 45.574115 2.0
83.276647
[5 rows x 7 columns]
Balance QualMiles BonusMiles ... FlightMiles FlightTrans
DaysSinceEnroll
0 28143 0 174 ... 0 0
7000
1 19244 0 215 ... 0 0
6968
2 41354 0 4123 ... 0 0
7034
3 14776 0 500 ... 0 0
6952
4 97752 0 43300 ... 2077 4
6935
[5 rows x 7 columns]
(3999, 7)
IPython console
History
LSP Python: ready
conda: base (Python 3.7.7)
Line 47, Col 1
UTF-8
CRLF
RW
Mem 53%
Type here to search
2:11 PM
10/10/2020


200000
175000
150000
125000
100000
75000
50000
25000
0
Balance
QualMiles
BonusMiles
BonusTrans
FlightMiles
FlightTrans
DaysSinceEnroll



df1 - DataFrame

Index	Balance	QualMiles	ionusMile	ionusTran	lightMile:	FlightTran:	ysSinceEn
0	28143	0	174	1	0	0	7000
1	19244	0	215	2	0	0	6968
2	41354	0	4123	4	0	0	7034
3	14776	0	500	1	0	0	6952
4	97752	0	43300	26	2077	4	6935
5	16420	0	0	0	0	0	6942
6	84914	0	27482	25	0	0	6994
7	20856	0	5250	4	250	1	6938
9	104860	0	28426	28	1150	3	6931
10	40091	0	7278	10	0	0	6959
11	96522	0	61105	19	0	0	6924
12	43382	0	11150	20	0	0	6924
13	43097	0	3258	6	0	0	6918
14	17648	0	0	0	0	0	6912
15	28495	0	49442	15	0	0	6912
16	51890	0	48963	16	0	0	6910
17	13958	0	4291	5	0	0	6905
18	91473	0	27408	17	0	0	6903
19	23354	0	10447	5	0	0	6896
20	120576	0	58831	23	250	2	6896
21	185681	2024	13300	16	1800	9	6896

Format Resize Background color Column min/max Save and Close Close

##Now removing outliers from Bonus Miles:

q1 = df['BonusMiles']. quantile (0.25)

q3 = df['BonusMiles']. quantile (0.75)

iqr = q3-q1

ul = q3 + (1.5*iqr)

ll = q1 - (1.5*iqr)

df2 = df1[(df1['BonusMiles']>ll) &(df1['BonusMiles’] <ul)]

plt. Figure (figsize= (12,8))

sns. Boxplot(data=df2)

##Removing outliers from FlightMiles:

q1 = df['FlightMiles']. quantile (0.25)

q3 = df['FlightMiles']. quantile (0.75)

iqr = q3-q1

ul = q3 + (1.5*iqr)

ll = q1 - (1.5*iqr)

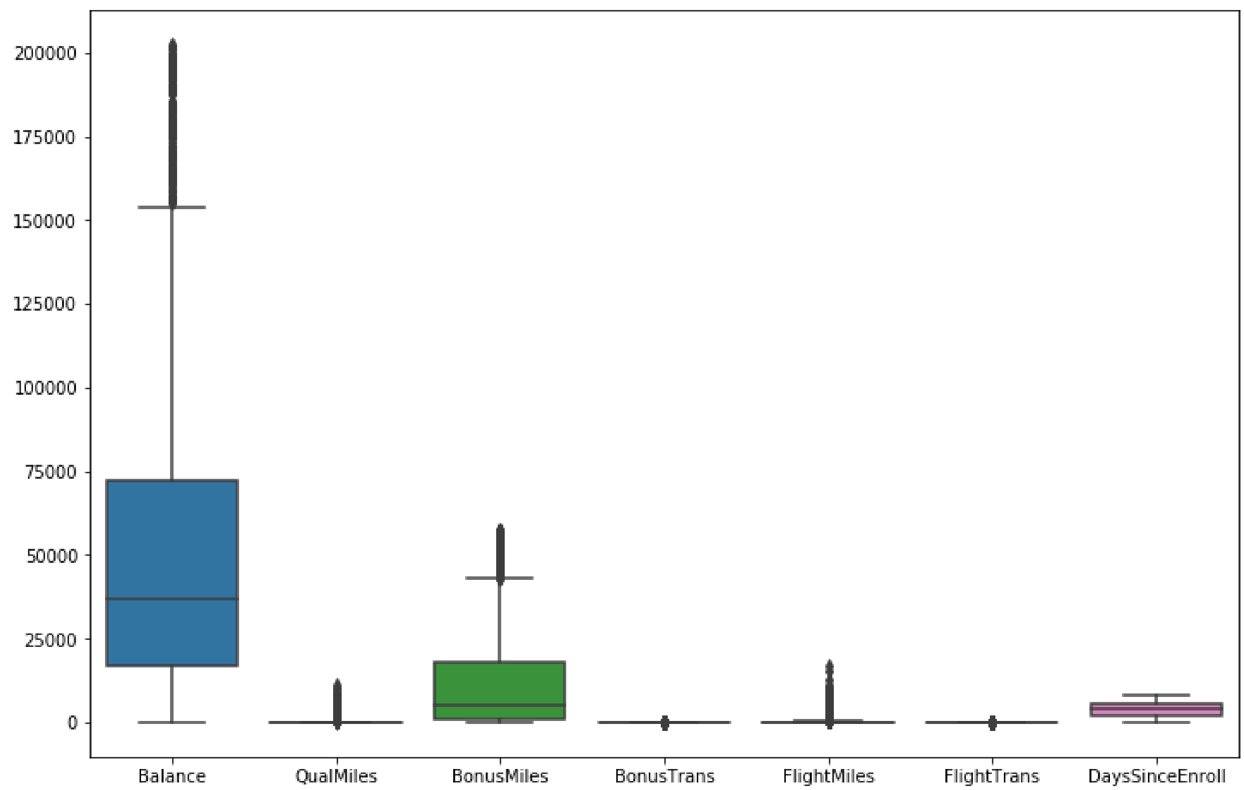
df3 = df2[(df2['FlightMiles']>ll) &(df2['FlightMiles’] <ul)]

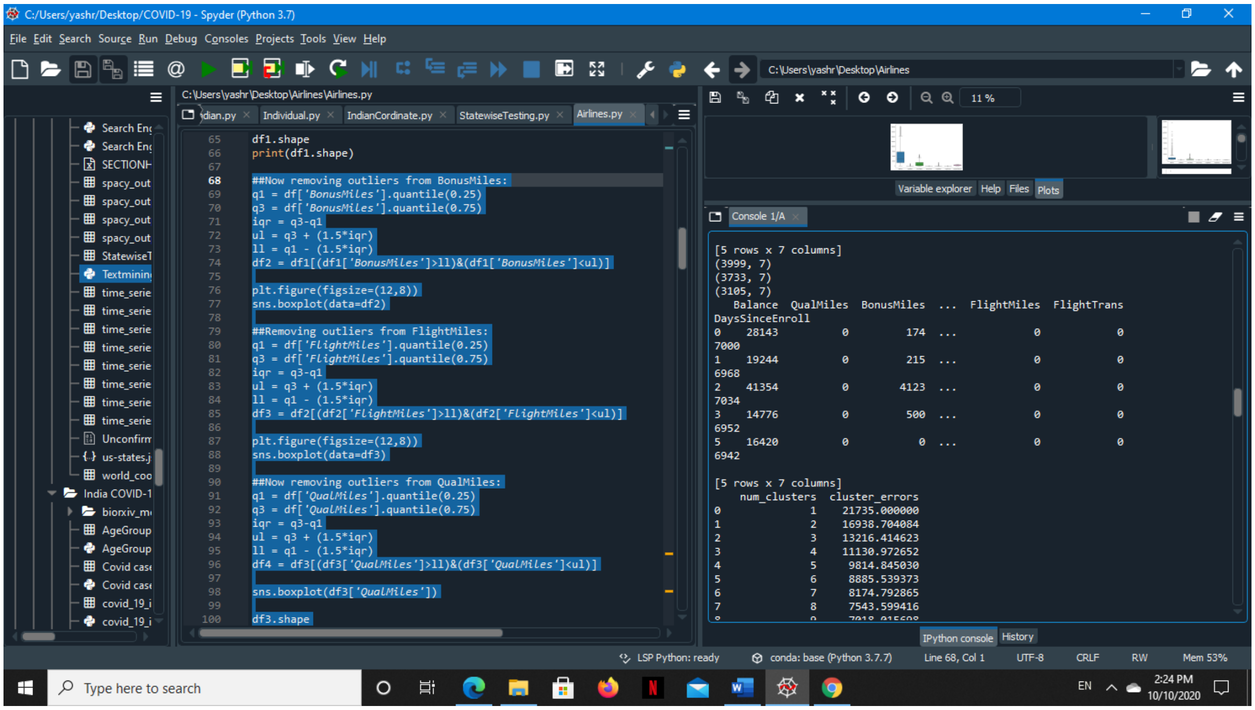
plt. Figure (figsize= (12,8))

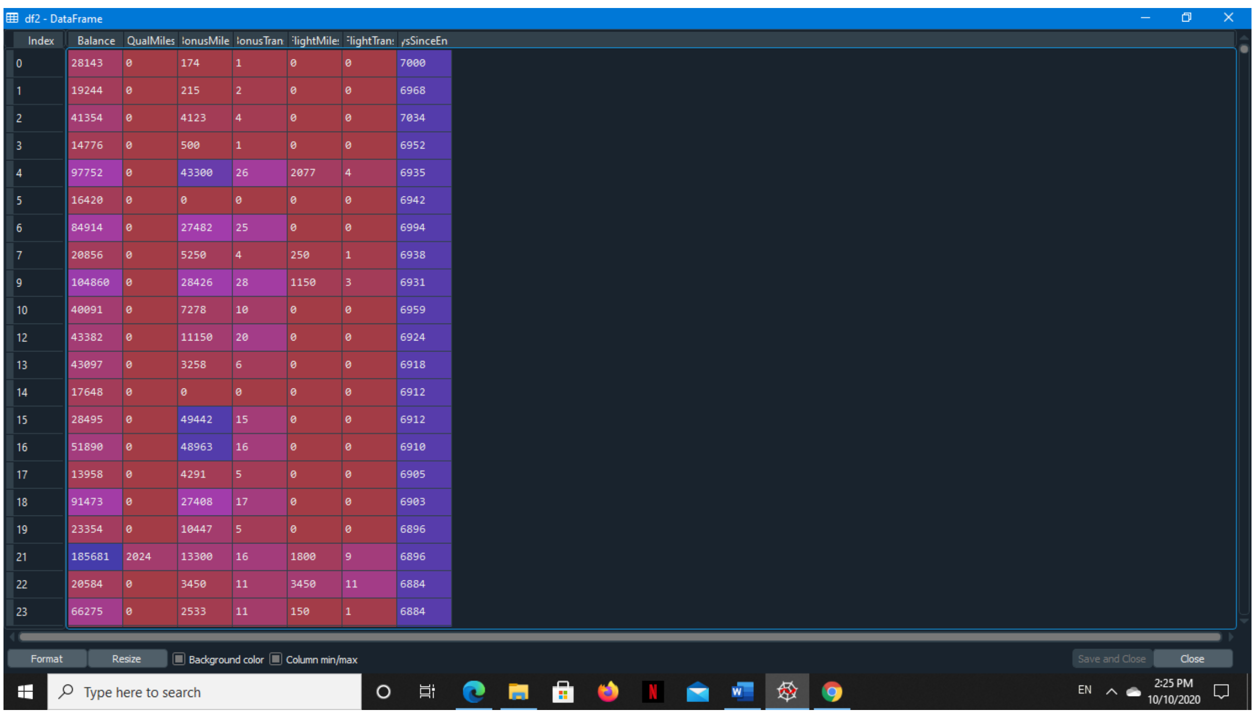
sns. Boxplot(data=df3)

##Now removing outliers from QualMiles:

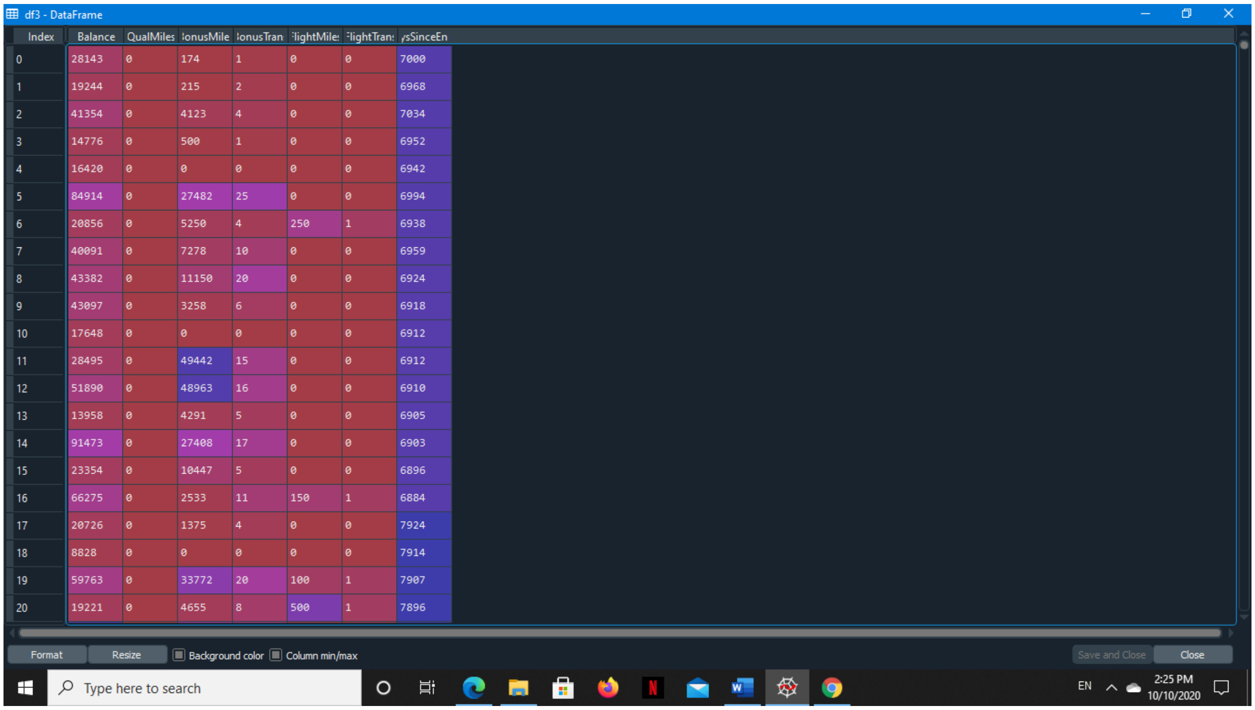
```
q1 = df['QualMiles']. quantile (0.25)
q3 = df['QualMiles']. quantile (0.75)
iqr = q3-q1
ul = q3 + (1.5*iqr)
ll = q1 - (1.5*iqr)
df4 = df3[(df3['QualMiles']>ll) &(df3['QualMiles’] <ul)]

sns. Boxplot(df3['QualMiles'])

df3.shape
print (df3.shape)

df3.head()
print (df3.head())
```






df2 - DataFrame

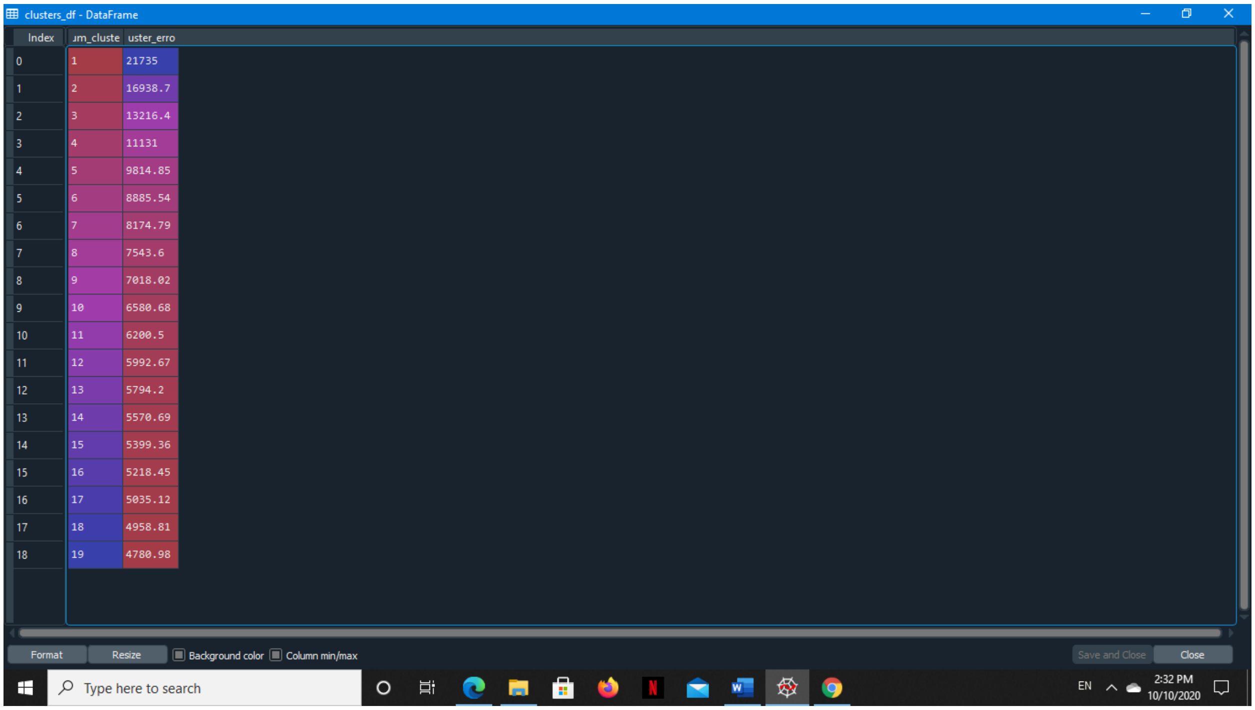
Index	Balance	QualMiles	lonusMile	lonusTran	lightMile:	lightTran:	ysSinceEn
0	28143	0	174	1	0	0	7000
1	19244	0	215	2	0	0	6968
2	41354	0	4123	4	0	0	7034
3	14776	0	500	1	0	0	6952
4	97752	0	43300	26	2077	4	6935
5	16420	0	0	0	0	0	6942
6	84914	0	27482	25	0	0	6994
7	20856	0	5250	4	250	1	6938
9	104860	0	28426	28	1150	3	6931
10	40091	0	7278	10	0	0	6959
12	43382	0	11150	20	0	0	6924
13	43097	0	3258	6	0	0	6918
14	17648	0	0	0	0	0	6912
15	28495	0	49442	15	0	0	6912
16	51890	0	48963	16	0	0	6910
17	13958	0	4291	5	0	0	6905
18	91473	0	27408	17	0	0	6903
19	23354	0	10447	5	0	0	6896
21	185681	2024	13300	16	1800	9	6896
22	20584	0	3450	11	3450	11	6884
23	66275	0	2533	11	150	1	6884

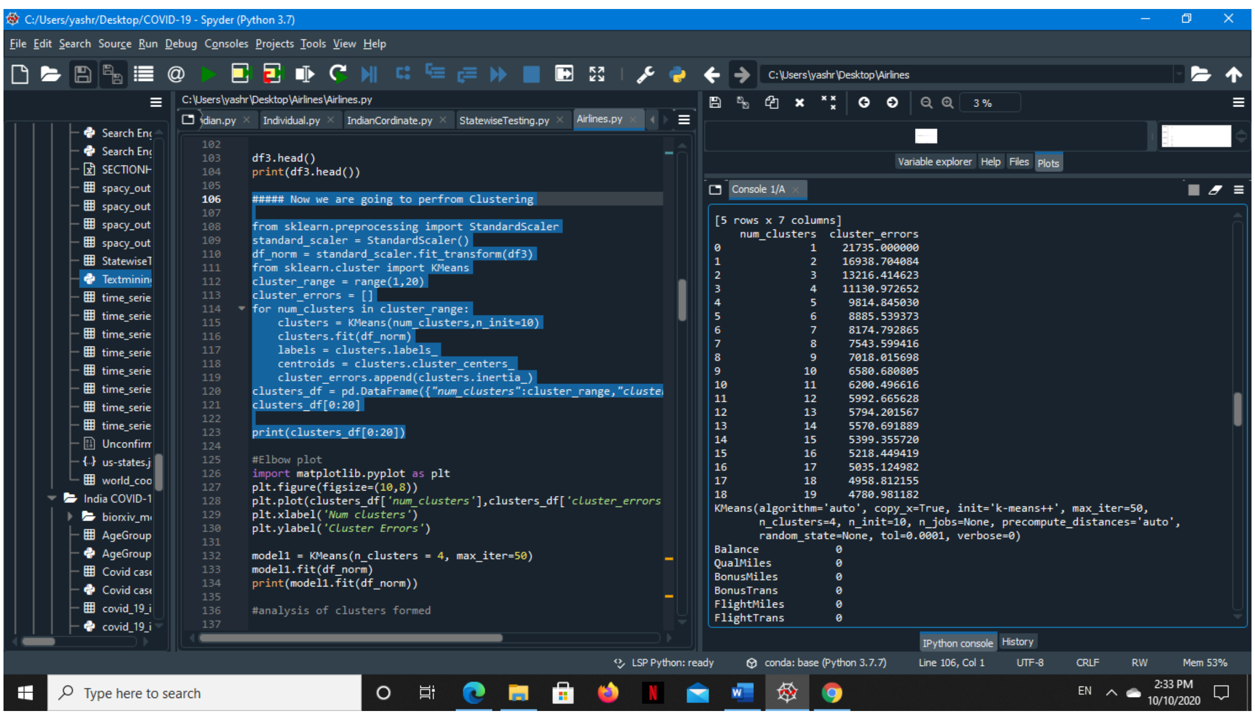


Index	Balance	QualMiles	onusMile	onusTran	lightMile	lightTran	sSinceEn
0	28143	0	174	1	0	0	7000
1	19244	0	215	2	0	0	6968
2	41354	0	4123	4	0	0	7034
3	14776	0	500	1	0	0	6952
4	16420	0	0	0	0	0	6942
5	84914	0	27482	25	0	0	6994
6	20856	0	5250	4	250	1	6938
7	40091	0	7278	10	0	0	6959
8	43382	0	11150	20	0	0	6924
9	43097	0	3258	6	0	0	6918
10	17648	0	0	0	0	0	6912
11	28495	0	49442	15	0	0	6912
12	51890	0	48963	16	0	0	6910
13	13958	0	4291	5	0	0	6905
14	91473	0	27408	17	0	0	6903
15	23354	0	10447	5	0	0	6896
16	66275	0	2533	11	150	1	6884
17	20726	0	1375	4	0	0	7924
18	8828	0	0	0	0	0	7914
19	59763	0	33772	20	100	1	7907
20	19221	0	4655	8	500	1	7896

Now we are going to perform Clustering

```
from sklearn. preprocessing import StandardScaler
standard_scaler = StandardScaler ()
df_norm = standard_scaler.fit_transform(df3)
from sklearn. cluster import KMeans
cluster_range = range (1,20)
cluster_errors = []
for num_clusters in cluster_range:
   clusters = KMeans (num_clusters, n_init=10)
   clusters. Fit(df_norm)
   labels = clusters. Labels_
   centroids = clusters. cluster_centers_
   cluster_errors. append (clusters. Inertia_)
clusters_df = pd. DataFrame ({"num_clusters": cluster_range,"cluster_errors":
cluster_errors})
clusters_df [0:20]

print (clusters_df [0:20])
```





#analysis of clusters formed

```
df3.index = pd.RangeIndex(len(df3.index))
df_km = pd.concat([df3,pd.Series(model1.labels_)],axis=1)
```

```
df_km.columns = ['Balance', 'QualMiles', 'BonusMiles', 'BonusTrans', 'FlightMiles',
    'FlightTrans', 'DaysSinceEnroll','ClusterID']

df_km.isna().sum()
print(df_km.isna().sum())

df_km
print(df_km)


km_cluster_Balance = pd.DataFrame(df_km.groupby('ClusterID')['Balance'].mean())
km_cluster_QualMiles = pd.DataFrame(df_km.groupby('ClusterID')['QualMiles'].mean())
km_cluster_BonusMiles =
pd.DataFrame(df_km.groupby('ClusterID')['BonusMiles'].mean())
km_cluster_BonusTrans =
pd.DataFrame(df_km.groupby('ClusterID')['BonusTrans'].mean())
km_cluster_FlightMiles =
pd.DataFrame(df_km.groupby('ClusterID')['FlightMiles'].mean())
km_cluster_FlightTrans =
pd.DataFrame(df_km.groupby('ClusterID')['FlightTrans'].mean())
km_cluster_DaysSinceEnroll =
pd.DataFrame(df_km.groupby('ClusterID')['DaysSinceEnroll'].mean())


df = pd.concat([pd.Series([0,1,2,3]),km_cluster_Balance,
km_cluster_QualMiles,
km_cluster_BonusMiles,
km_cluster_BonusTrans,
km_cluster_FlightMiles,
km_cluster_FlightTrans,
km_cluster_DaysSinceEnroll],axis=1)
df.columns = ['ClusterID','Balance', 'QualMiles', 'BonusMiles', 'BonusTrans', 'FlightMiles',
    'FlightTrans', 'DaysSinceEnroll']
df
print(df)

sns.barplot(data=df,x='ClusterID',y='Balance')


#People in Cluster 1 require highest number of miles to be eligible for award travel
sns.barplot(data=df,x='ClusterID',y='QualMiles')
```

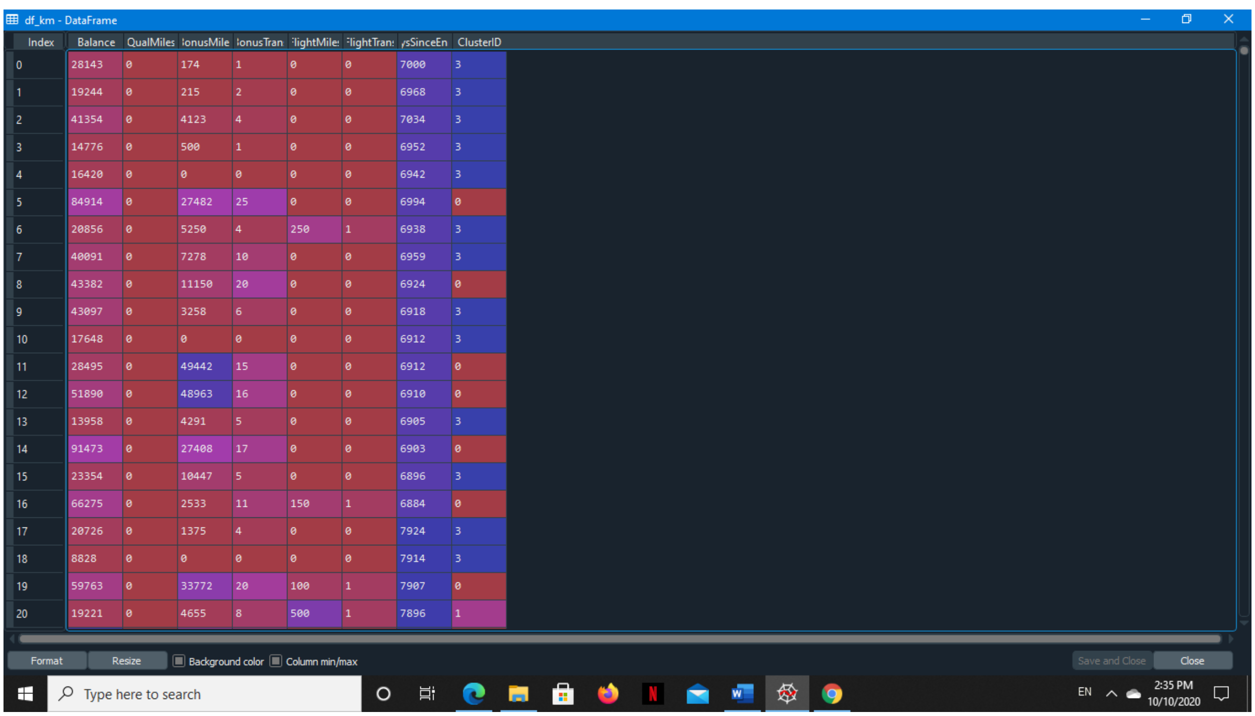
#Cluster 2 contains people who require most number of miles to qualify for top flight status
sns.barplot(data=df,x='ClusterID',y='BonusMiles')

#Cluster 1 people have the highest number of miles earned from non-flight bonus transactions in the past 12 months
sns.barplot(data=df,x='ClusterID',y='BonusTrans')

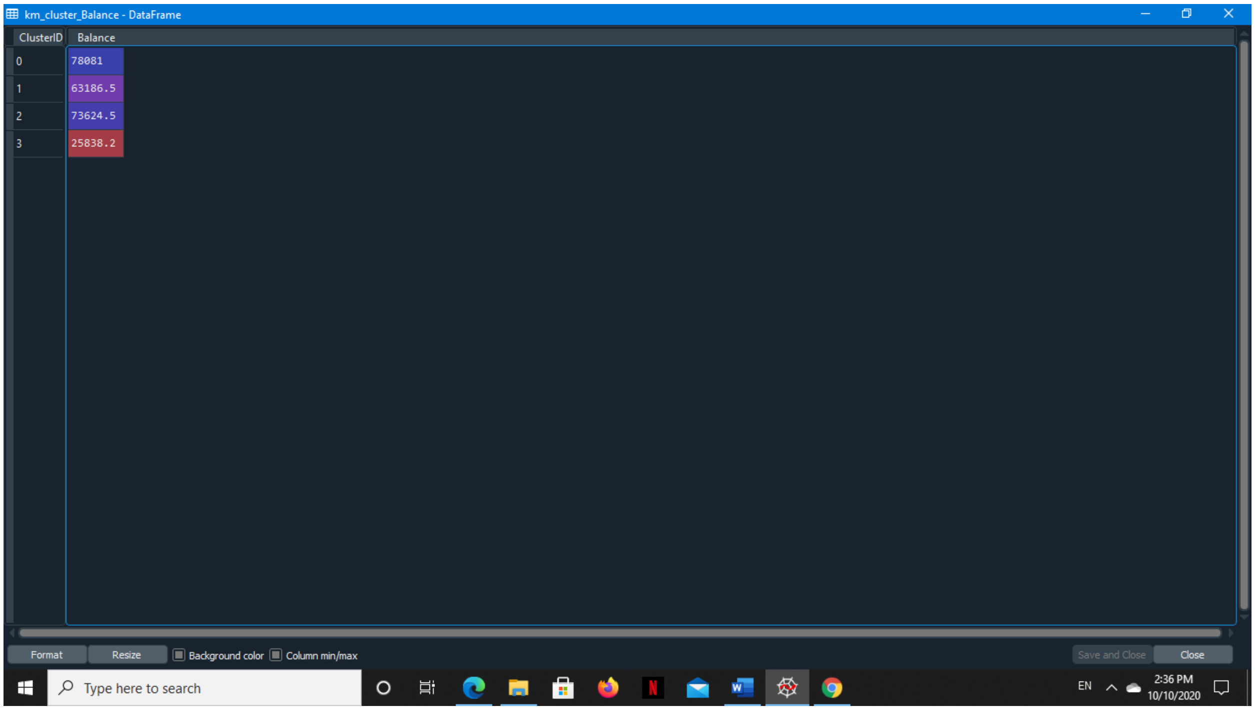
#Cluster 1 people have the highest number of non-flight bonus transactions in the past 12 months
sns.barplot(data=df,x='ClusterID',y='FlightMiles')

#Cluster 3 people have highest number of flight miles in the past 12 months, whereas we can see that flight miles are quite low
#for cluster 1 people, hence they were made to earn more flight miles through non-flight bonus transactions, so that they fly,
#and increase the business for the airline
sns.barplot(data=df,x='ClusterID',y='FlightTrans')

sns.barplot(data=df,x='ClusterID',y='DaysSinceEnroll')

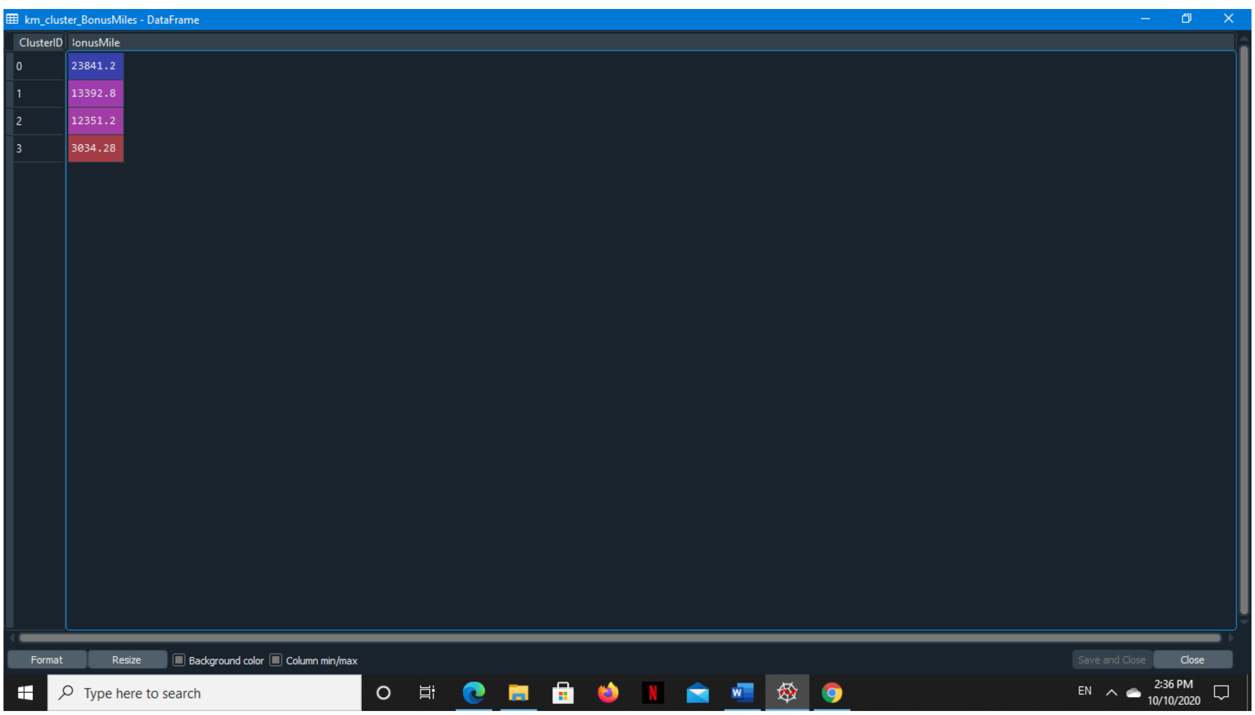

df_km - DataFrame

Index	Balance	QualMiles	›onusMile	›onusTran	›lightMile	›lightTran	›ysSinceEn	ClusterID
0	28143	0	174	1	0	0	7000	3
1	19244	0	215	2	0	0	6968	3
2	41354	0	4123	4	0	0	7034	3
3	14776	0	500	1	0	0	6952	3
4	16420	0	0	0	0	0	6942	3
5	84914	0	27482	25	0	0	6994	0
6	20856	0	5250	4	250	1	6938	3
7	40091	0	7278	10	0	0	6959	3
8	43382	0	11150	20	0	0	6924	0
9	43097	0	3258	6	0	0	6918	3
10	17648	0	0	0	0	0	6912	3
11	28495	0	49442	15	0	0	6912	0
12	51890	0	48963	16	0	0	6910	0
13	13958	0	4291	5	0	0	6905	3
14	91473	0	27408	17	0	0	6903	0
15	23354	0	10447	5	0	0	6896	3
16	66275	0	2533	11	150	1	6884	0
17	20726	0	1375	4	0	0	7924	3
18	8828	0	0	0	0	0	7914	3
19	59763	0	33772	20	100	1	7907	0
20	19221	0	4655	8	500	1	7896	1



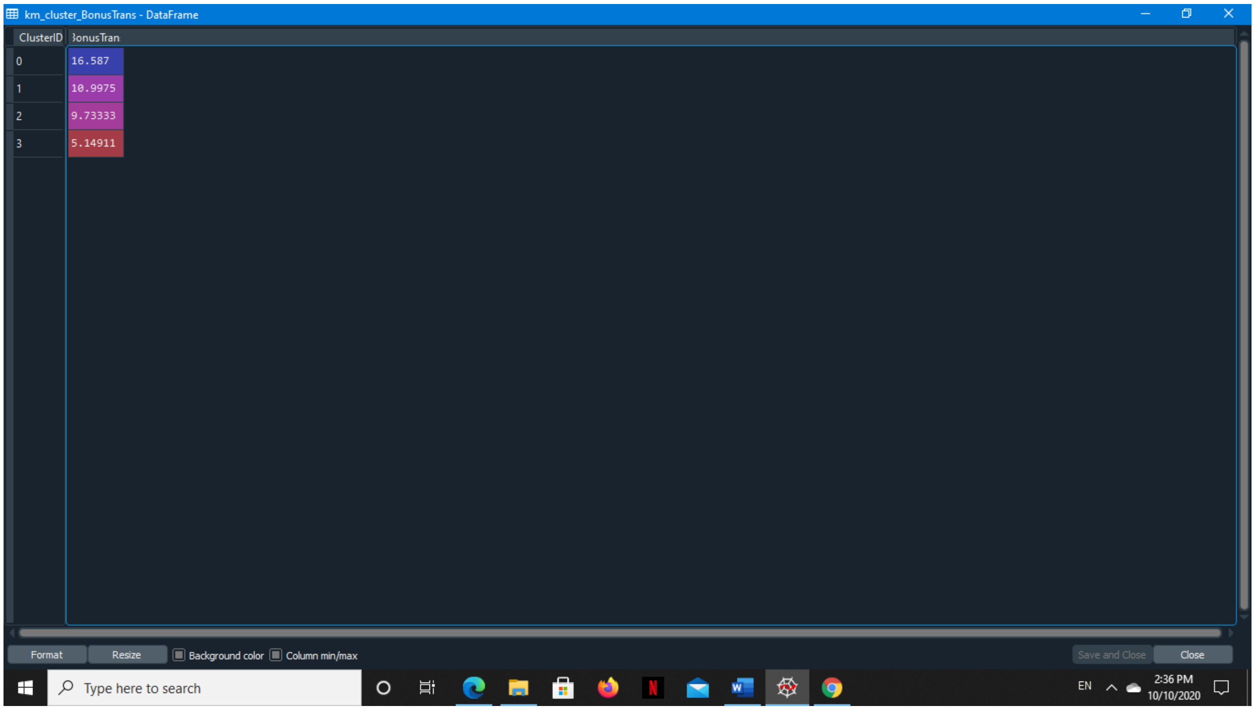
km_cluster_Balance - DataFrame

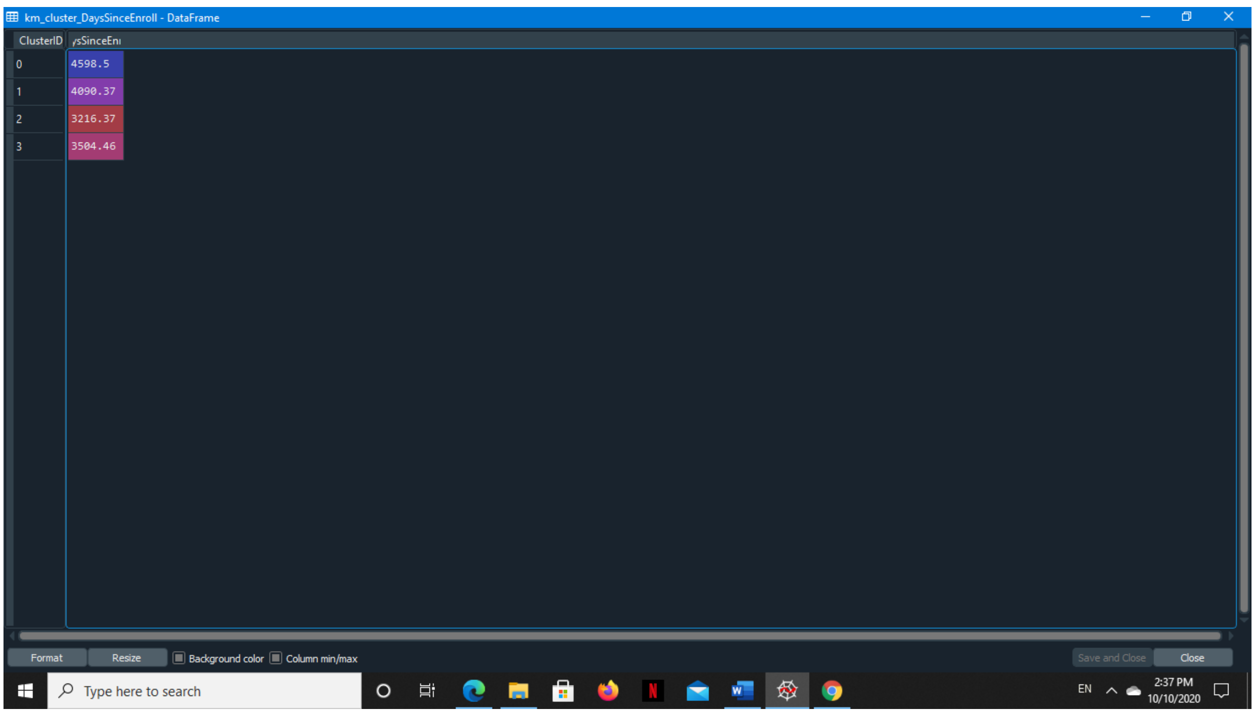
ClusterID	Balance
0	78081
1	63186.5
2	73624.5
3	25838.2

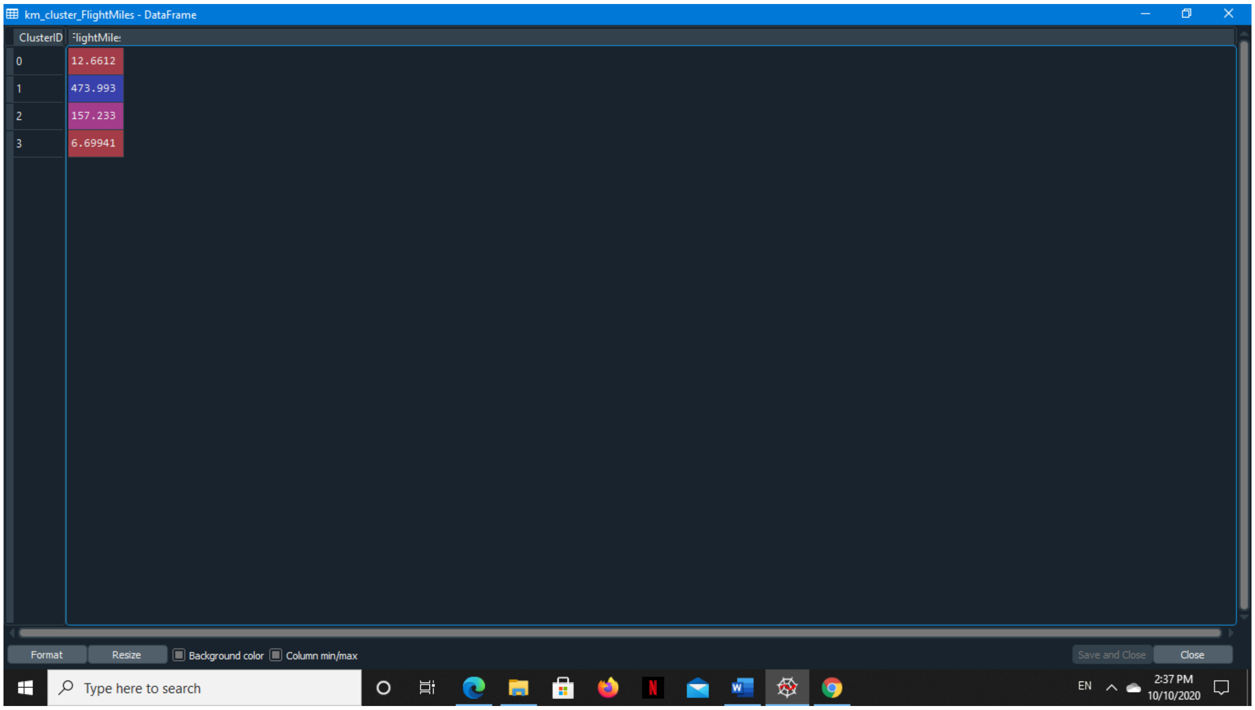


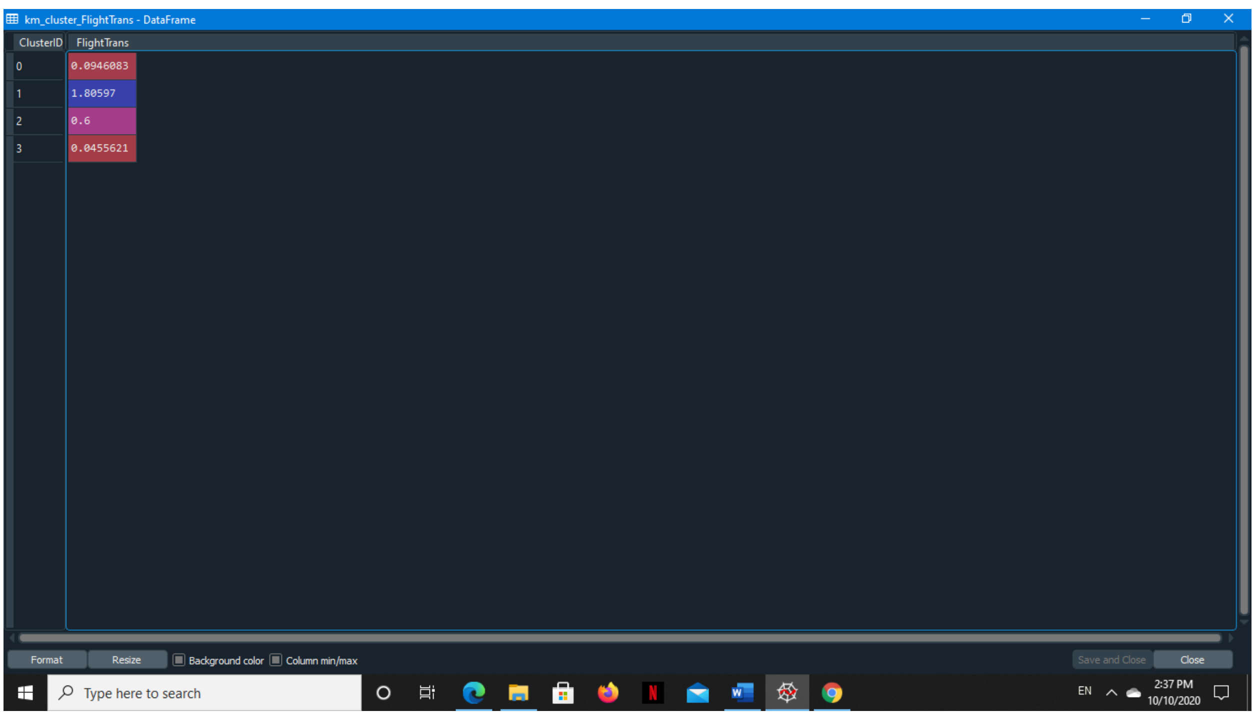
km_cluster_BonusMiles - DataFrame

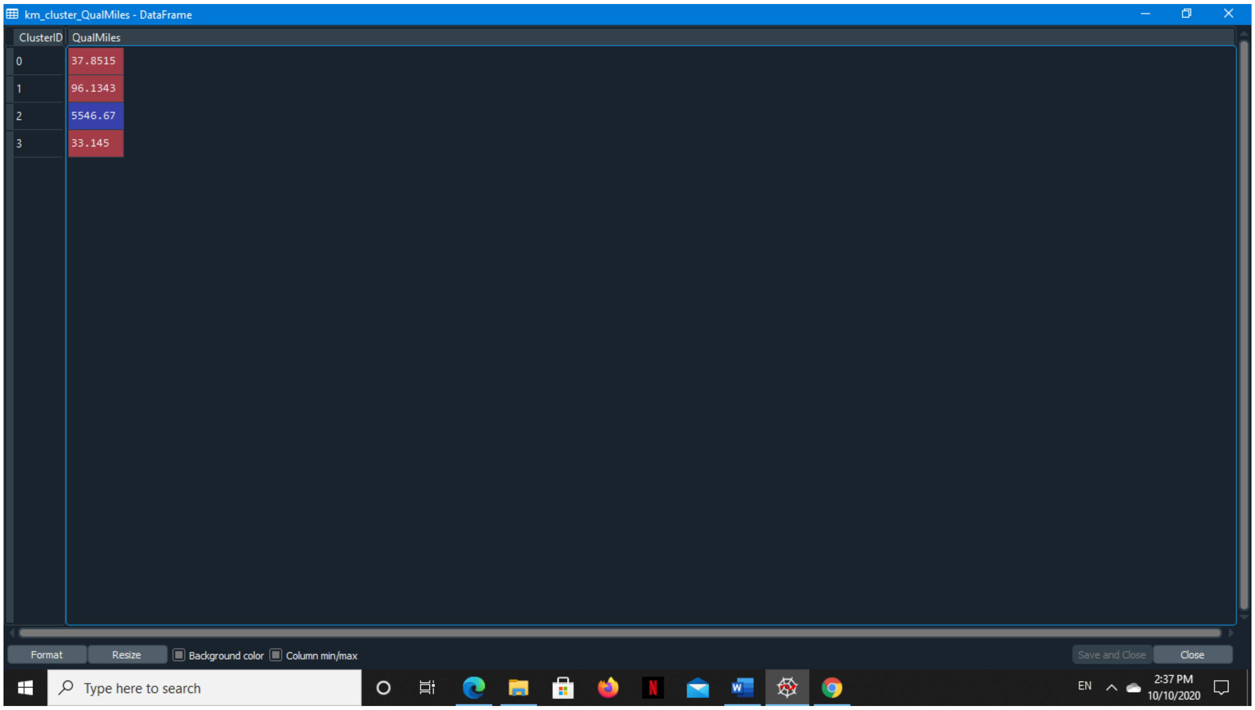
ClusterID	!onusMile
0	23841.2
1	13392.8
2	12351.2
3	3034.28


km_cluster_BonusTrans - DataFrame
ClusterID
3onusTran
0
16.587
1
10.9975
2
9.73333
3
5.14911
Format
Resize
Background color
Column min/max
Save and Close
Close
Type here to search
EN
2:36 PM
10/10/2020


km_cluster_DaysSinceEnroll - DataFrame
ClusterID
ysSinceEnı
0
4598.5
1
4090.37
2
3216.37
3
3504.46
Format
Resize
Background color
Column min/max
Save and Close
Close
Type here to search
EN
2:37 PM
10/10/2020

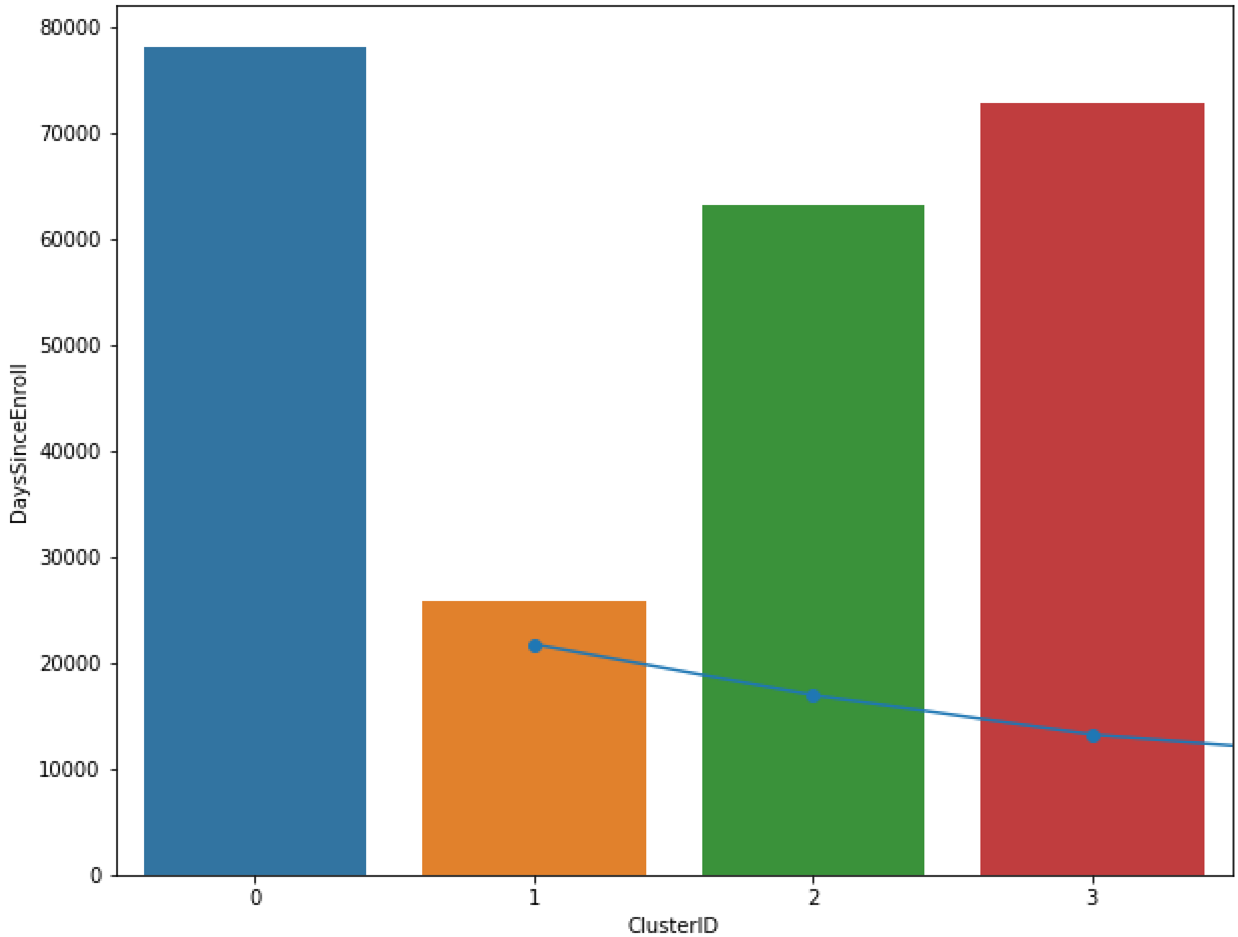

km_cluster_FlightMiles - DataFrame
ClusterID
FlightMile:
0
1
2
3
12.6612
473.993
157.233
6.69941
Format
Resize
Background color
Column min/max
Save and Close
Close
Type here to search
EN
2:37 PM
10/10/2020


km_cluster_FlightTrans - DataFrame
ClusterID
FlightTrans
0
1
2
3
0.0946083
1.80597
0.6
0.0455621
Format
Resize
Background color
Column min/max
Save and Close
Close
Type here to search
EN
2:37 PM
10/10/2020



km_cluster_QualMiles - DataFrame

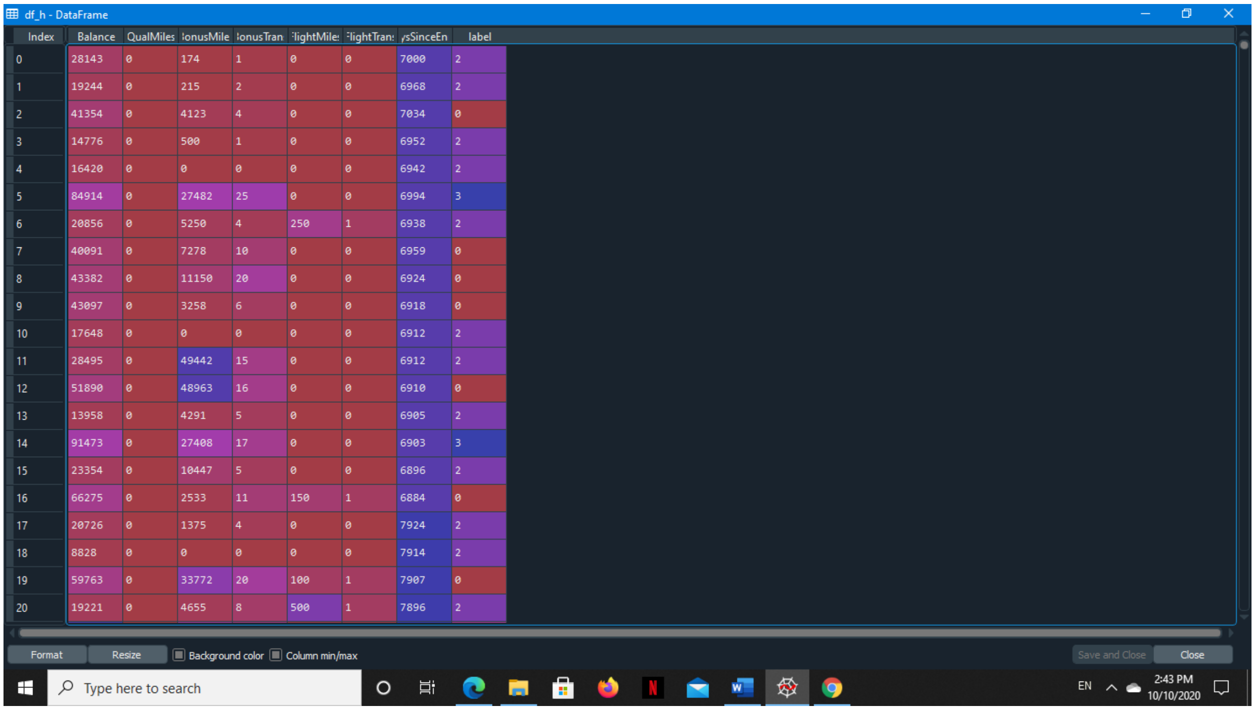
ClusterID	QualMiles
0	37.8515
1	96.1343
2	5546.67
3	33.145



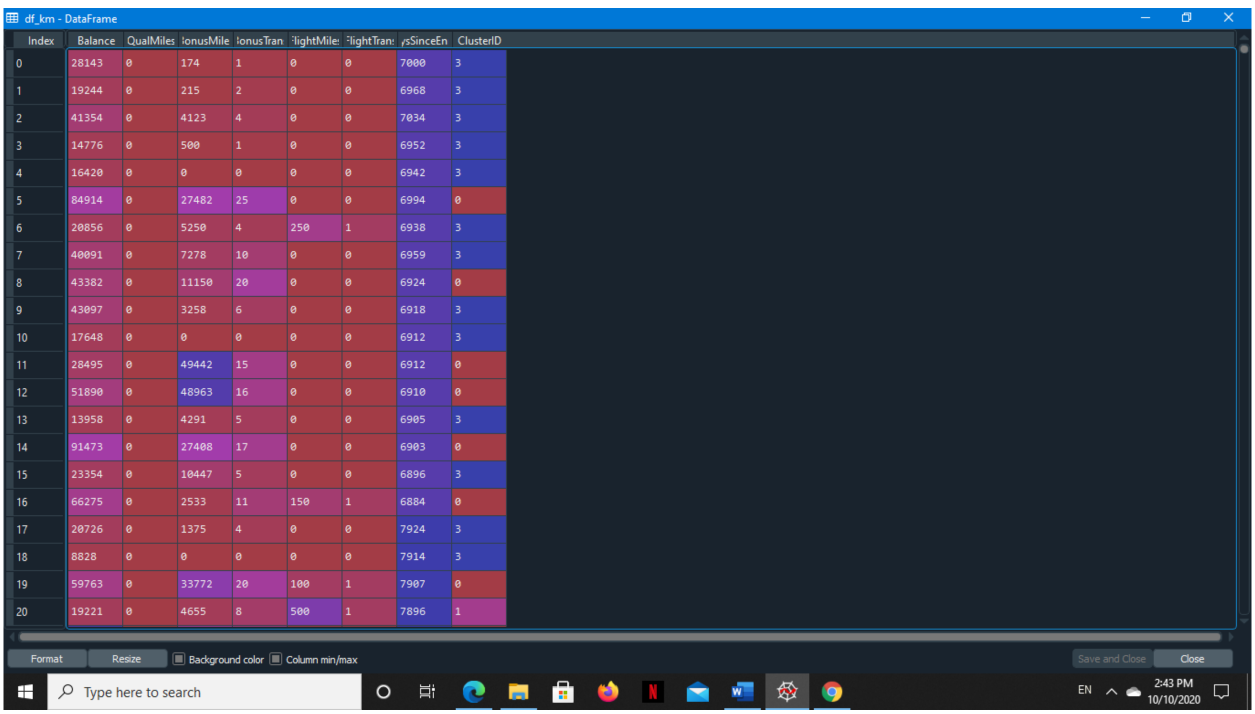
##In Cluster1, people have enrolled in the flight program for a very long time, longer than others, which is why they are being
#offered more flight miles through non-flight bonus transactions, so that they can increase the frequency of flying for customers
#who have been enrolled for a long time. This hasn't had much effect on the people though. The flying miles for Cluster1 are
#still quite less.
#Cluster0 has less flight miles, but the points they were awarded are lesser than the amount awarded to Cluster1, and that
#could be before people in cluster0 enrolled after the people in cluster1.
##Cluster 3 is not getting much fly miles through non-flight bonus transactions because they are already fliers with high miles
##and more number of transactions than the rest.

df.columns
print(df.columns)

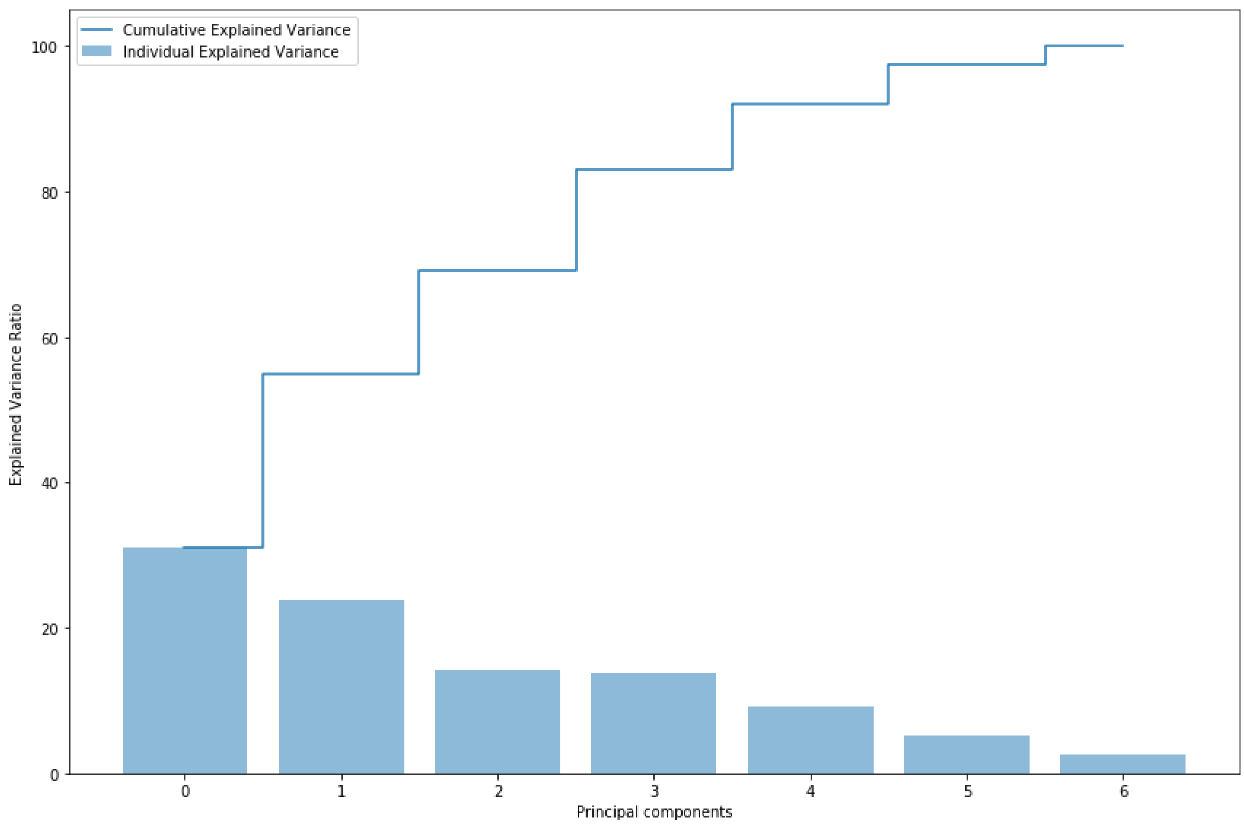
```
##Agglomerative Clustering:
from sklearn.cluster import AgglomerativeClustering

his_clus = AgglomerativeClustering(n_clusters=4,affinity='euclidean',linkage='complete')

cluster2 = his_clus.fit_predict(df3)

df_h = df3.copy(deep=True)
df_h['label'] = cluster2
df_h['label'].value_counts()
print(df_h['label'].value_counts())


his_clus = AgglomerativeClustering(n_clusters=4,affinity='euclidean',linkage='ward')

cluster2 = his_clus.fit_predict(df3)

df_h = df3.copy(deep=True)
df_h['label'] = cluster2
df_h['label'].value_counts()
print(df_h['label'].value_counts())

df_km['ClusterID'].value_counts()
print(df_km['ClusterID'].value_counts())
```



Index	Balance	QualMiles	lonusMile	lonusTran	lightMile:	FlightTran:	ysSinceEn	label
0	28143	0	174	1	0	0	7000	2
1	19244	0	215	2	0	0	6968	2
2	41354	0	4123	4	0	0	7034	0
3	14776	0	500	1	0	0	6952	2
4	16420	0	0	0	0	0	6942	2
5	84914	0	27482	25	0	0	6994	3
6	20856	0	5250	4	250	1	6938	2
7	40091	0	7278	10	0	0	6959	0
8	43382	0	11150	20	0	0	6924	0
9	43097	0	3258	6	0	0	6918	0
10	17648	0	0	0	0	0	6912	2
11	28495	0	49442	15	0	0	6912	2
12	51890	0	48963	16	0	0	6910	0
13	13958	0	4291	5	0	0	6905	2
14	91473	0	27408	17	0	0	6903	3
15	23354	0	10447	5	0	0	6896	2
16	66275	0	2533	11	150	1	6884	0
17	20726	0	1375	4	0	0	7924	2
18	8828	0	0	0	0	0	7914	2
19	59763	0	33772	20	100	1	7907	0
20	19221	0	4655	8	500	1	7896	2

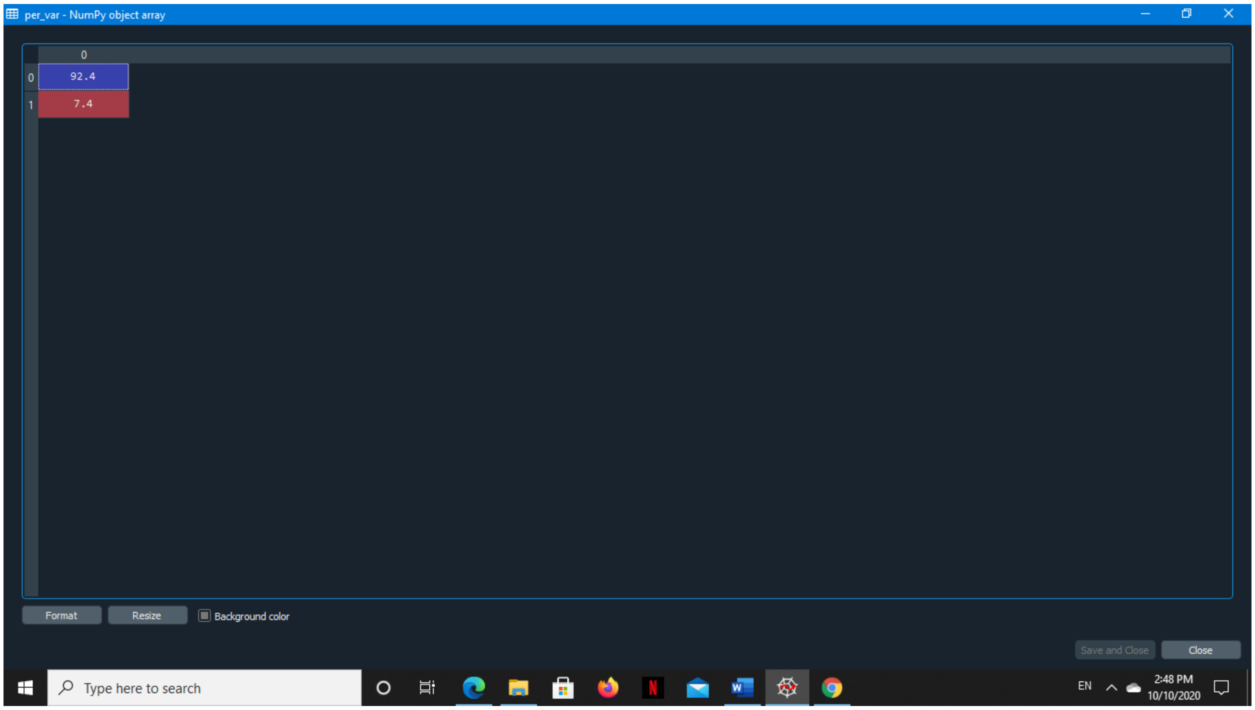


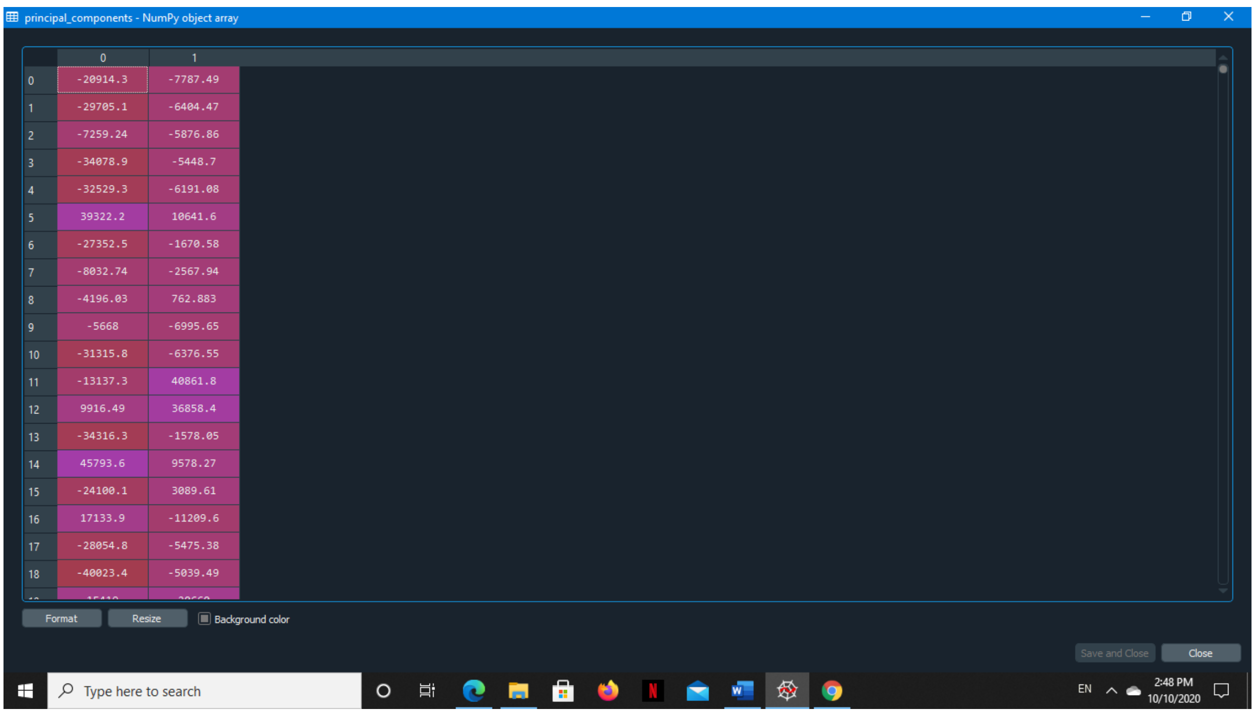
Index	Balance	QualMiles	lonusMile	lonusTran	lightMile:	FlightTran:	ysSinceEn	ClusterID
0	28143	0	174	1	0	0	7000	3
1	19244	0	215	2	0	0	6968	3
2	41354	0	4123	4	0	0	7034	3
3	14776	0	500	1	0	0	6952	3
4	16420	0	0	0	0	0	6942	3
5	84914	0	27482	25	0	0	6994	0
6	20856	0	5250	4	250	1	6938	3
7	40091	0	7278	10	0	0	6959	3
8	43382	0	11150	20	0	0	6924	0
9	43097	0	3258	6	0	0	6918	3
10	17648	0	0	0	0	0	6912	3
11	28495	0	49442	15	0	0	6912	0
12	51890	0	48963	16	0	0	6910	0
13	13958	0	4291	5	0	0	6905	3
14	91473	0	27408	17	0	0	6903	0
15	23354	0	10447	5	0	0	6896	3
16	66275	0	2533	11	150	1	6884	0
17	20726	0	1375	4	0	0	7924	3
18	8828	0	0	0	0	0	7914	3
19	59763	0	33772	20	100	1	7907	0
20	19221	0	4655	8	500	1	7896	1

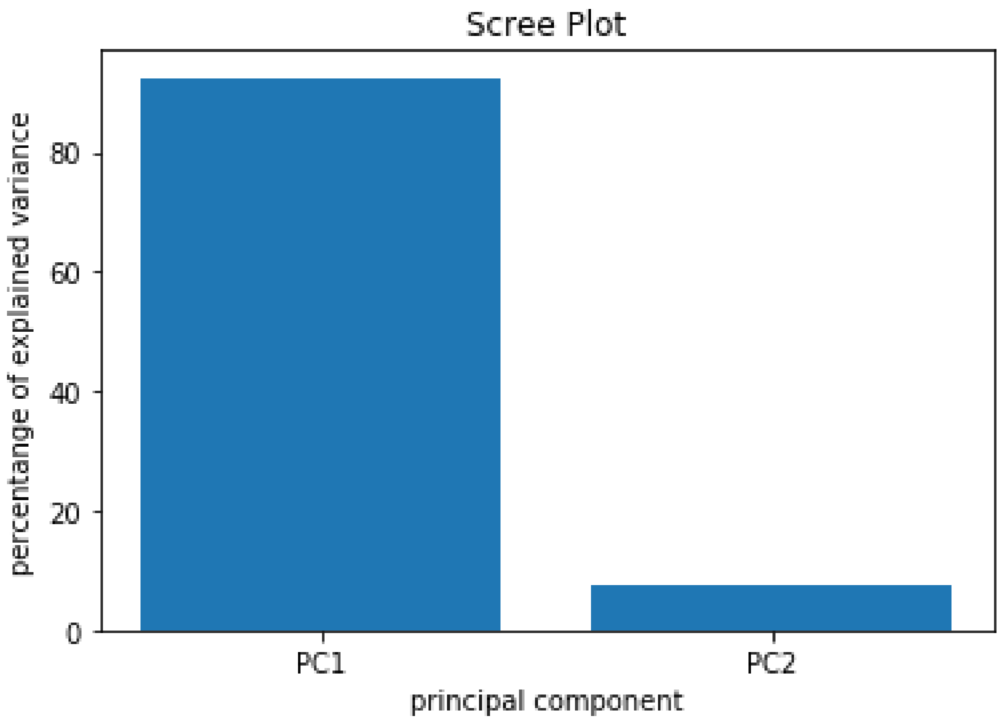


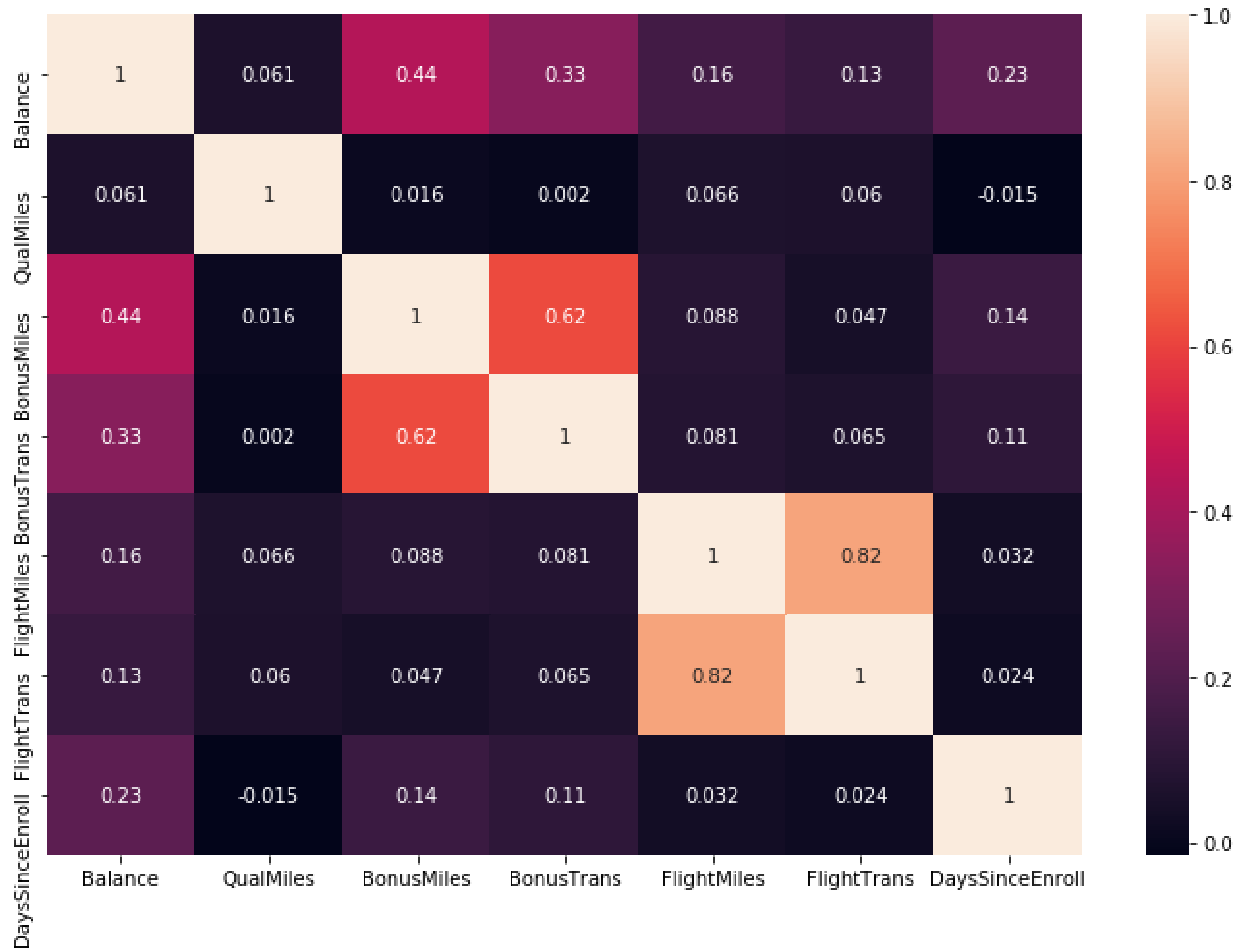

```
##WE can compare what kmeans gave and what Agglomerative Clustering gave
##NOW, Principal Component Analysis:
X_std = StandardScaler().fit_transform(df3)
cov_matrix = np.cov(X_std.T)
cov_matrix
print(cov_matrix)


#Step3: Eigen values and eigen vector
eig_vals, eig_vecs = np.linalg.eig(cov_matrix)
print(eig_vals)
print(eig_vecs)

eigen_pairs = [(np.abs(eig_vals[i]),eig_vecs[:,i]) for i in range(len(eig_vals))]
tot = sum(eig_vals)
var_exp = [(i/tot)*100 for i in sorted (eig_vals,reverse=True)]
cum_var_exp = np.cumsum(var_exp)
print("Cumulative Variance Explained",cum_var_exp)
```

```
df.shape[1]
print(df.shape[1])

plt.figure(figsize=(12,8))
plt.bar(range(7),var_exp,alpha=0.5,align='center',label='Individual Explained Variance')
plt.step(range(7),cum_var_exp,where='mid',label='Cumulative Explained Variance')
plt.ylabel('Explained Variance Ratio')
plt.xlabel('Principal components')
plt.legend(loc='best')
plt.tight_layout()
plt.show()

from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(df3)
X1 = pd.DataFrame(data = principal_components, columns = ['PC1', 'PC2'])
X1.head()
print(X1.head())

per_var = np.round(pca.explained_variance_ratio_ * 100, decimals=1)
labels = ['PC' + str(x) for x in range(1, len(per_var) + 1)]
plt.bar(x=range(1, len(per_var)+1), height=per_var, tick_label=labels)
plt.ylabel('percentange of explained variance')
plt.xlabel('principal component')
plt.title('Scree Plot')
plt.show()

df3.head()
print(df3.head())

plt.figure(figsize=(12,8))
sns.heatmap(df3.corr(),annot=True)
```


per_var - NumPy object array
0
0 92.4
1 7.4
Format
Resize
Background color
Save and Close
Close
Type here to search
EN
2:48 PM
10/10/2020


principal_components - NumPy object array
0 1
0 -20914.3 -7787.49
1 -29705.1 -6404.47
2 -7259.24 -5876.86
3 -34078.9 -5448.7
4 -32529.3 -6191.08
5 39322.2 10641.6
6 -27352.5 -1670.58
7 -8032.74 -2567.94
8 -4196.03 762.883
9 -5668 -6995.65
10 -31315.8 -6376.55
11 -13137.3 40861.8
12 9916.49 36858.4
13 -34316.3 -1578.05
14 45793.6 9578.27
15 -24100.1 3089.61
16 17133.9 -11209.6
17 -28054.8 -5475.38
18 -40023.4 -5039.49
Format
Resize
Background color
Save and Close
Close
Type here to search
EN
2:48 PM
10/10/2020


Scree Plot
percentange of explained variance
0
20
40
60
80
PC1
PC2
principal component




```
##Few of the features have high correlation, which shows that multi-collinearity will exist--one of the examples is
#FlightMiles and FlightTrans
#So, we can consider the dataframe X1 for now, and then build a model using the PCs:
from sklearn.cluster import KMeans
Kmean = KMeans(n_clusters=2)
Kmean.fit(X1)
print(Kmean.fit(X1))

KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
 n_clusters=2, n_init=10, n_jobs=1, precompute_distances='auto',
 random_state=None, tol=0.0001, verbose=0)

Kmean.cluster_centers_
print(Kmean.cluster_centers_)
```

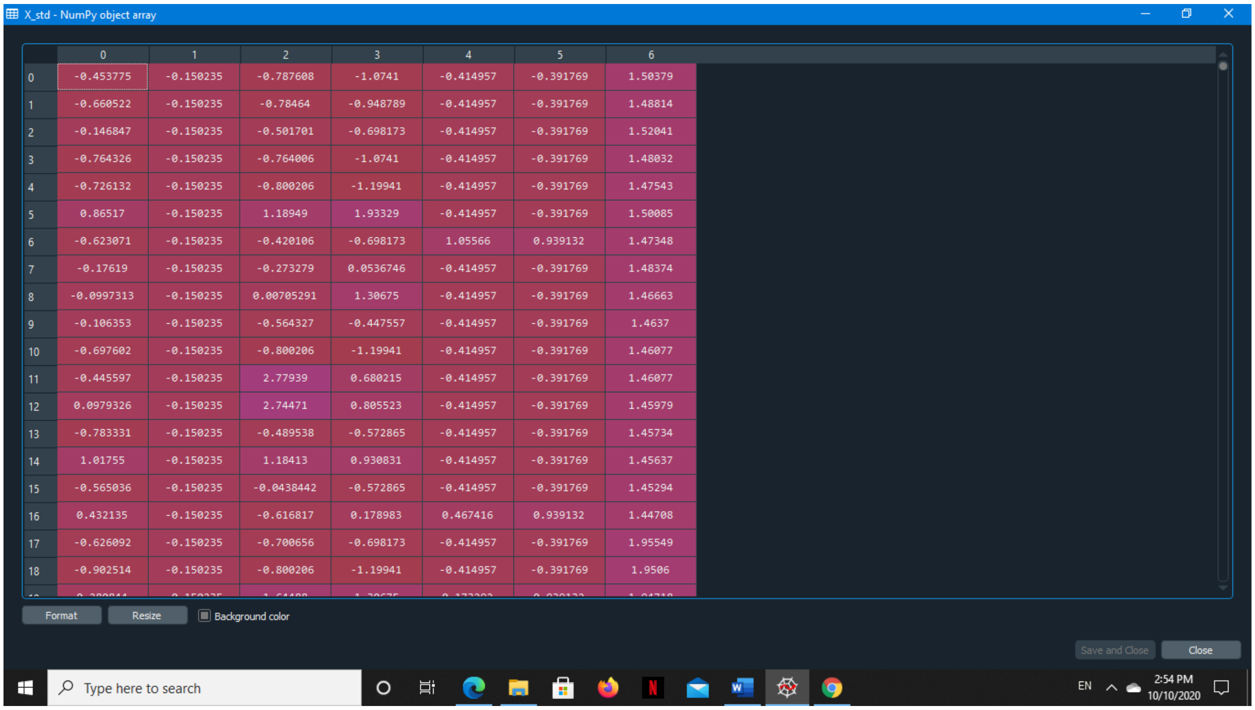
```
import matplotlib.pyplot as plt
plt.scatter(X1['PC1'], X1['PC2'], s =50, c='b')
plt.scatter(6.69202776e+04, -2.56491151e+01, s=200, c='g', marker='s')
plt.scatter(-2.00200830e+04, 7.67327084e+00, s=200, c='r', marker='s')
plt.show()

Kmean.labels_
print(Kmean.labels_)

X1['KMC'] = Kmean.fit_predict(X1[['PC1','PC2']])
sns.scatterplot(x='PC1',y='PC2',hue='KMC',data=X1,palette='spring')
plt.scatter(6.69202776e+04, -2.56491151e+01, s=200, marker='s')
plt.scatter(-2.00200830e+04, 7.67327084e+00, s=200, marker='s')
plt.show()

##Let's try DBSCAN for the same:
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.2,min_samples=10)
db.fit(X1[['PC1','PC2']])
print(db.fit(X1[['PC1','PC2']]))

X1['DBC'] = db.labels_
sns.scatterplot(x='PC1',y='PC2',hue='DBC',data=X1,palette='spring')

X1['DBC'].value_counts()
print(X1['DBC'].value_counts())
```

X - DataFrame

Index	Balance	QualMiles	onusMile	onusTran	lightMile:	lightTran:	ysSinceEn
0	28143	0	174	1	0	0	7000
1	19244	0	215	2	0	0	6968
2	41354	0	4123	4	0	0	7034
3	14776	0	500	1	0	0	6952
4	16420	0	0	0	0	0	6942
5	84914	0	27482	25	0	0	6994
6	20856	0	5250	4	250	1	6938
7	40091	0	7278	10	0	0	6959
8	43382	0	11150	20	0	0	6924
9	43097	0	3258	6	0	0	6918
10	17648	0	0	0	0	0	6912
11	28495	0	49442	15	0	0	6912
12	51890	0	48963	16	0	0	6910
13	13958	0	4291	5	0	0	6905
14	91473	0	27408	17	0	0	6903
15	23354	0	10447	5	0	0	6896
16	66275	0	2533	11	150	1	6884
17	20726	0	1375	4	0	0	7924
18	8828	0	0	0	0	0	7914
19	59763	0	33772	20	100	1	7907
20	19221	0	4655	8	500	1	7896

X1 - DataFrame

Index	PC1	PC2	KMC	DBC
0	-20914.3	-7787.49	0	-1
1	-29705.1	-6404.47	0	-1
2	-7259.24	-5876.86	0	-1
3	-34078.9	-5448.7	0	-1
4	-32529.3	-6191.08	0	-1
5	39322.2	10641.6	1	-1
6	-27352.5	-1670.58	0	-1
7	-8032.74	-2567.94	0	-1
8	-4196.03	762.883	0	-1
9	-5668	-6995.65	0	-1
10	-31315.8	-6376.55	0	-1
11	-13137.3	40861.8	0	-1
12	9916.49	36858.4	0	-1
13	-34316.3	-1578.05	0	-1
14	45793.6	9578.27	1	-1
15	-24100.1	3089.61	0	-1
16	17133.9	-11209.6	0	-1
17	-28054.8	-5475.38	0	-1
18	-40023.4	-5039.49	0	-1
19	15419	20660	0	-1
20	-29047.8	-2006	0	-1



X_std - NumPy object array

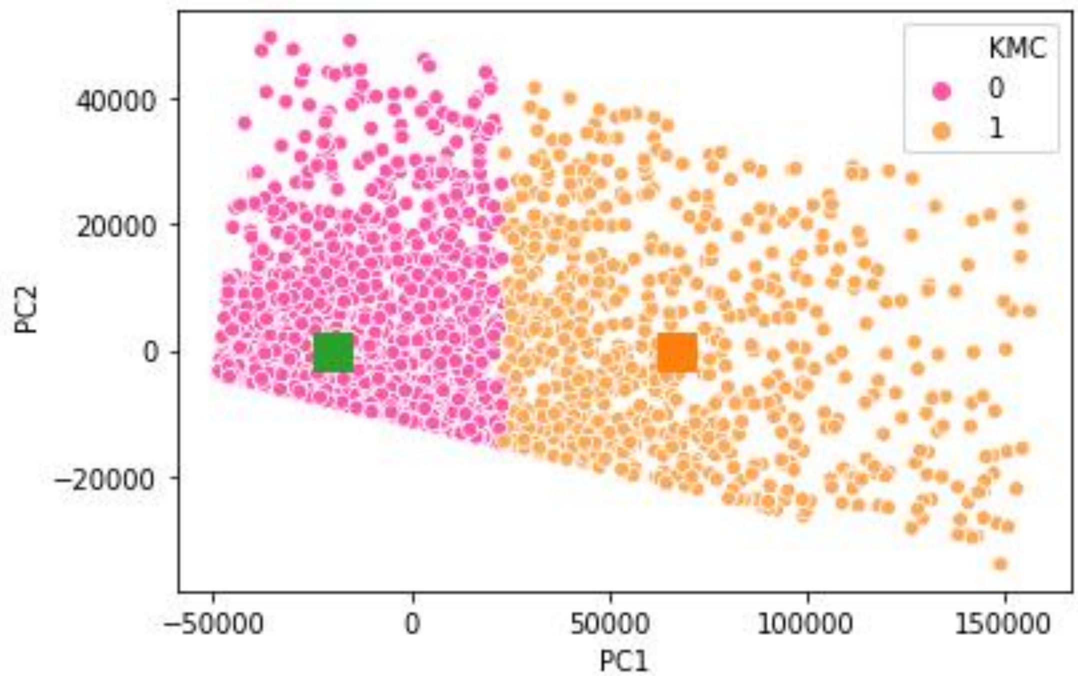
	0	1	2	3	4	5	6
0	-0.453775	-0.150235	-0.787608	-1.0741	-0.414957	-0.391769	1.50379
1	-0.660522	-0.150235	-0.78464	-0.948789	-0.414957	-0.391769	1.48814
2	-0.146847	-0.150235	-0.501701	-0.698173	-0.414957	-0.391769	1.52041
3	-0.764326	-0.150235	-0.764006	-1.0741	-0.414957	-0.391769	1.48032
4	-0.726132	-0.150235	-0.800206	-1.19941	-0.414957	-0.391769	1.47543
5	0.86517	-0.150235	1.18949	1.93329	-0.414957	-0.391769	1.50085
6	-0.623071	-0.150235	-0.420106	-0.698173	1.05566	0.939132	1.47348
7	-0.17619	-0.150235	-0.273279	0.0536746	-0.414957	-0.391769	1.48374
8	-0.0997313	-0.150235	0.00705291	1.30675	-0.414957	-0.391769	1.46663
9	-0.106353	-0.150235	-0.564327	-0.447557	-0.414957	-0.391769	1.4637
10	-0.697602	-0.150235	-0.800206	-1.19941	-0.414957	-0.391769	1.46077
11	-0.445597	-0.150235	2.77939	0.680215	-0.414957	-0.391769	1.46077
12	0.0979326	-0.150235	2.74471	0.805523	-0.414957	-0.391769	1.45979
13	-0.783331	-0.150235	-0.489538	-0.572865	-0.414957	-0.391769	1.45734
14	1.01755	-0.150235	1.18413	0.930831	-0.414957	-0.391769	1.45637
15	-0.565036	-0.150235	-0.0438442	-0.572865	-0.414957	-0.391769	1.45294
16	0.432135	-0.150235	-0.616817	0.178983	0.467416	0.939132	1.44708
17	-0.626092	-0.150235	-0.700656	-0.698173	-0.414957	-0.391769	1.95549
18	-0.902514	-0.150235	-0.800206	-1.19941	-0.414957	-0.391769	1.9506

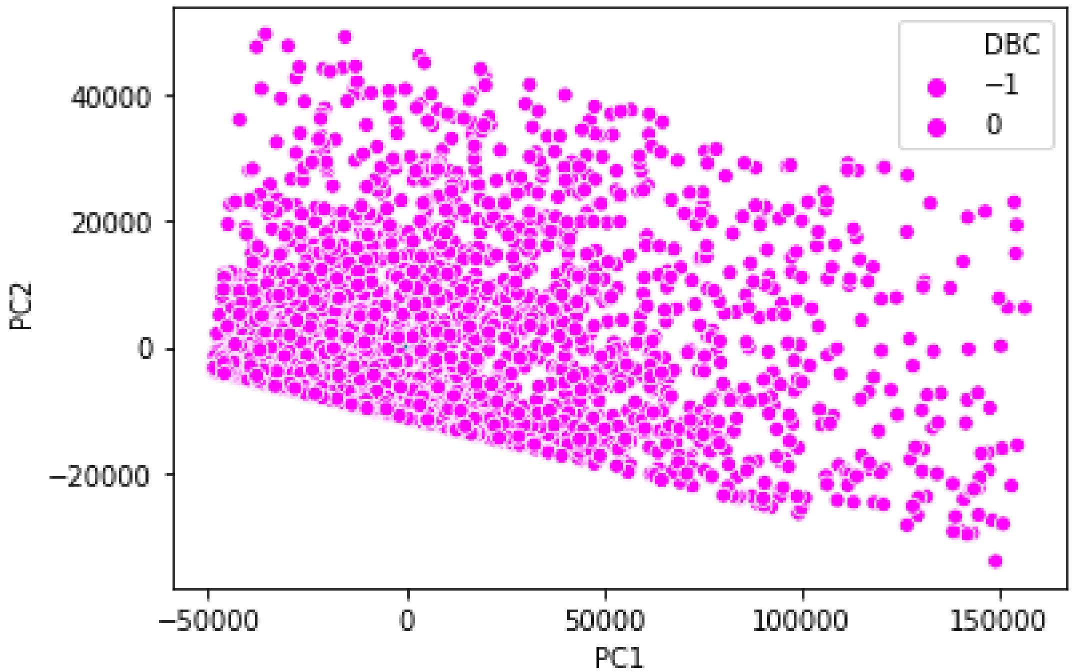
Format Resize Background color

Save and Close Close

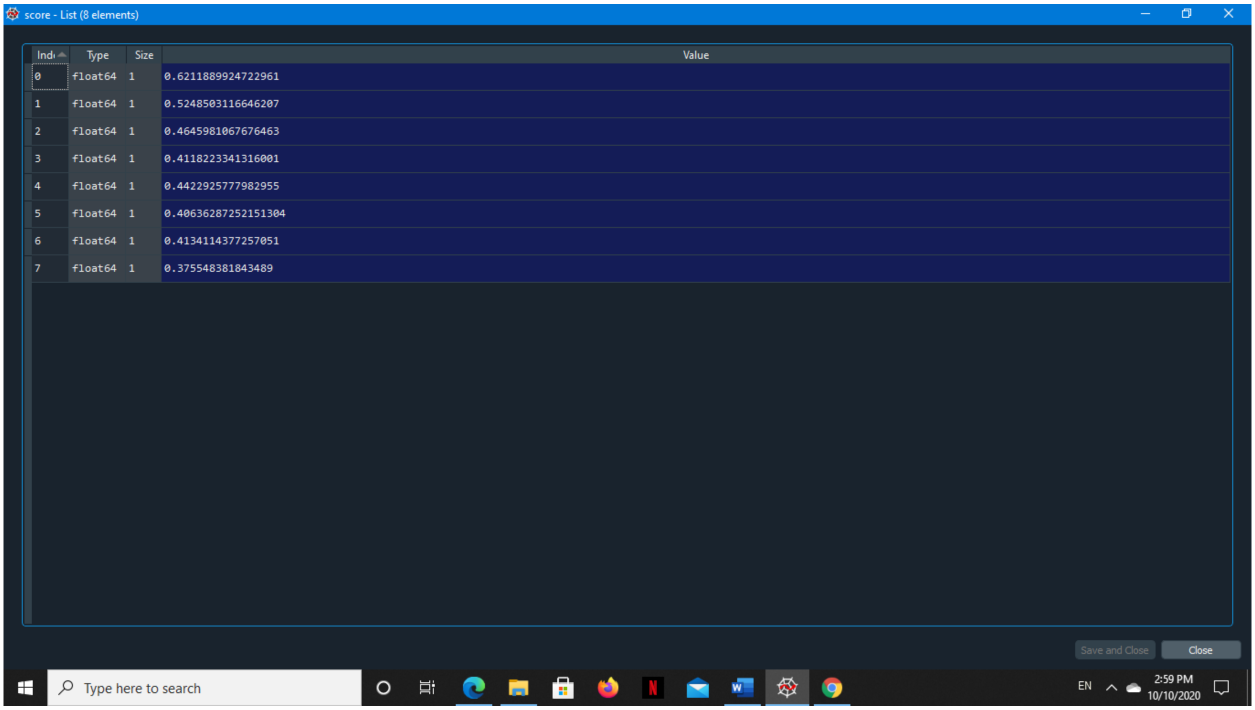
Type here to search

EN 2:54 PM 10/10/2020

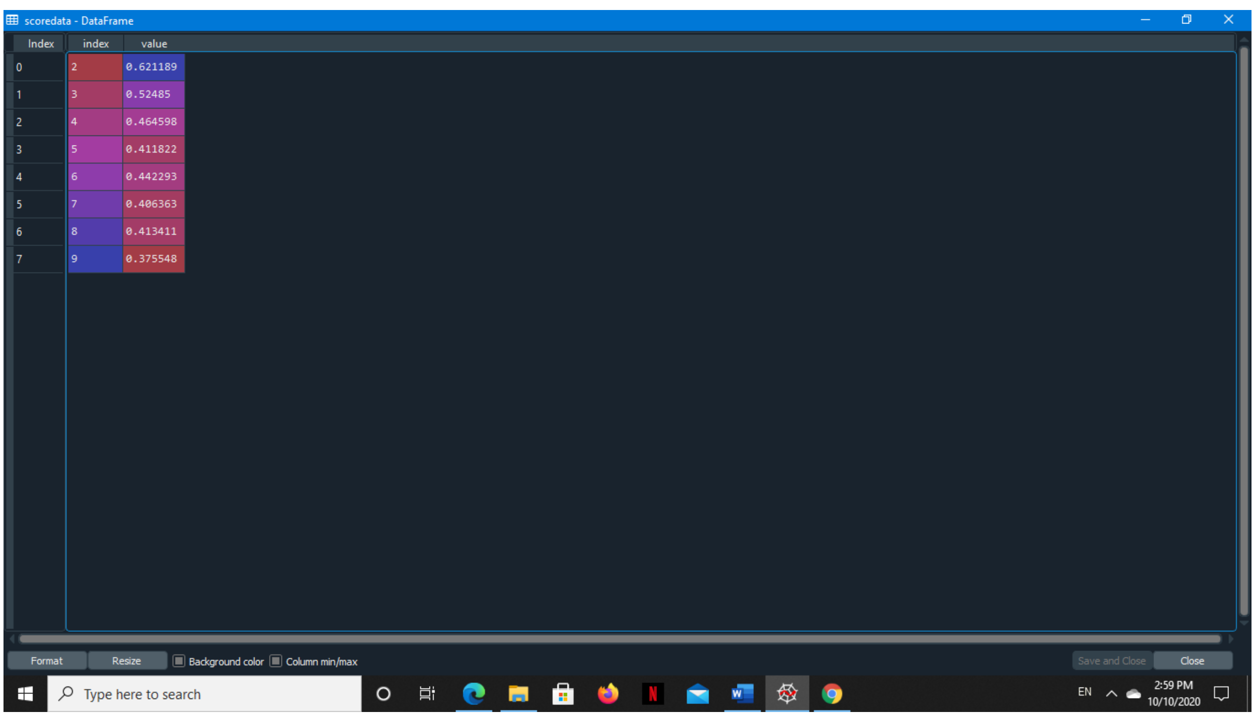




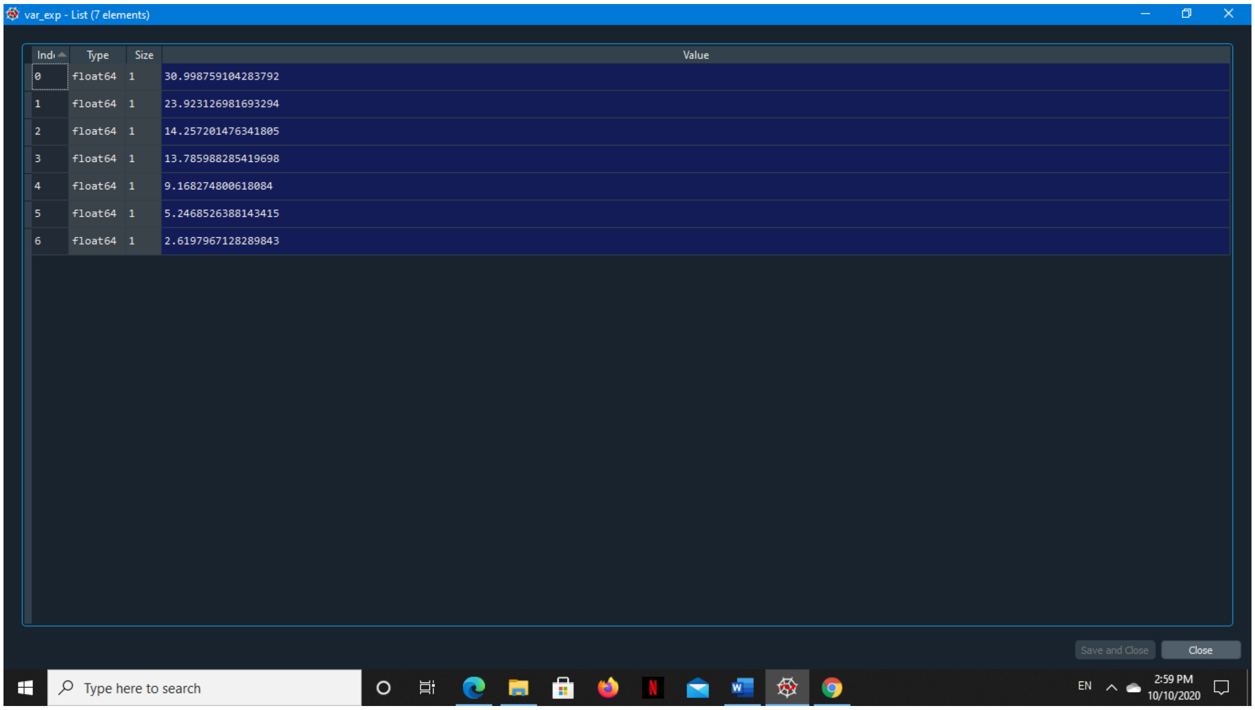

```
##So, in this case, DBSCAN was unable to classify the data into clusters
##We have already checked the inertia(Elbow plot)--let's check the Silhouette Score
from sklearn.metrics import silhouette_samples,silhouette_score
kmeans=KMeans(n_clusters=2)
X=df3
model = kmeans.fit(X=df3)
y=model.labels_
silhouette_score(X,y)
print(silhouette_score(X,y))

score = []
for n_clusters in range(2,10):
   kmeans = KMeans(n_clusters=n_clusters)
   kmeans.fit(X)
   labels = kmeans.labels_
   centroids = kmeans.cluster_centers_
   score.append(silhouette_score(X, labels, metric='euclidean'))
plt.plot(score)

scoredata =
pd.DataFrame(score,index=[2,3,4,5,6,7,8,9]).reset_index().rename(columns={0:'value'})
```

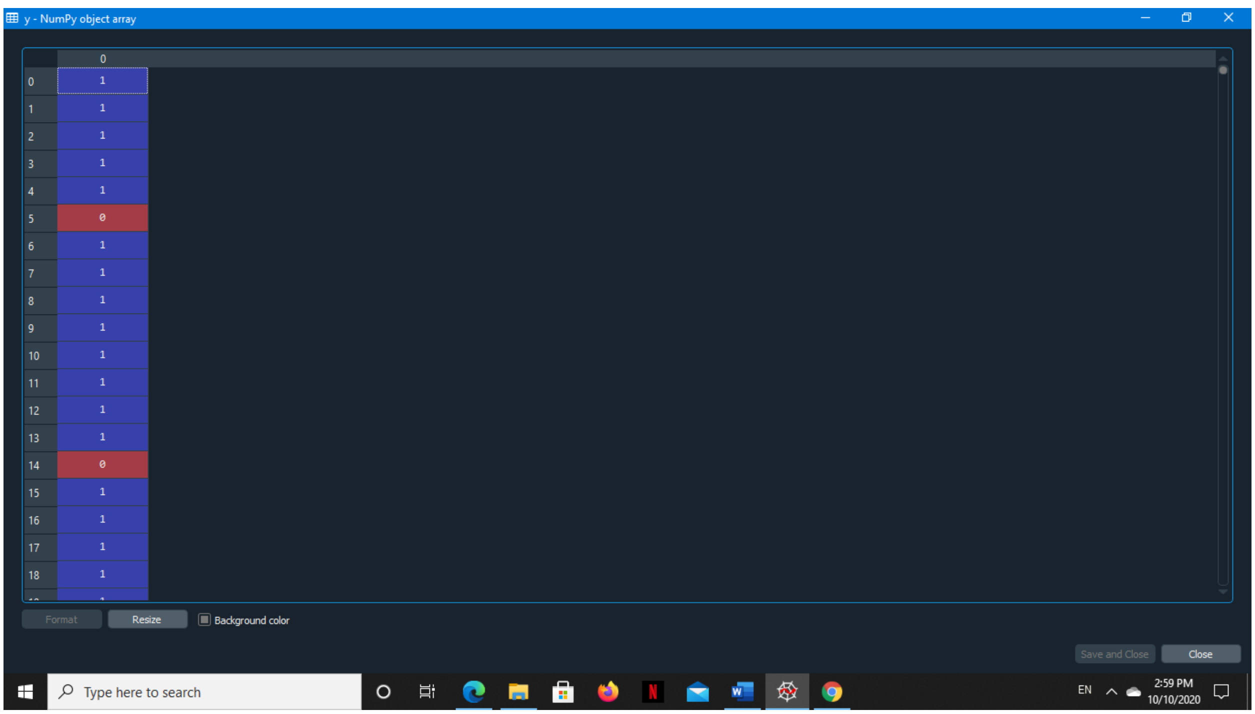

score - List (8 elements)

Index	Type	Size	Value
0	float64	1	0.6211889924722961
1	float64	1	0.5248503116646207
2	float64	1	0.4645981067676463
3	float64	1	0.4118223341316001
4	float64	1	0.4422925777982955
5	float64	1	0.40636287252151304
6	float64	1	0.4134114377257051
7	float64	1	0.375548381843489


scoredata - DataFrame

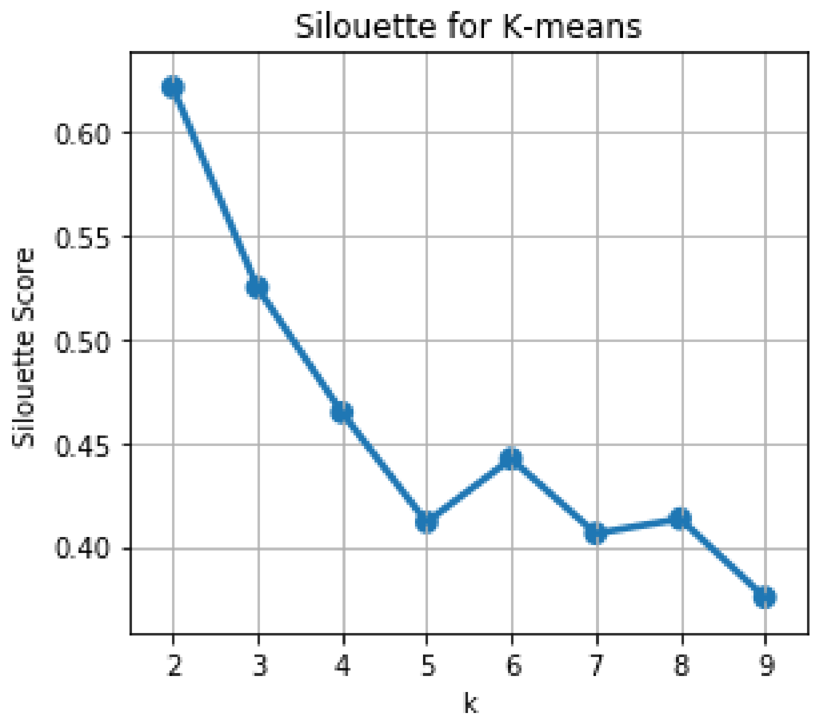
Index	index	value
0	2	0.621189
1	3	0.52485
2	4	0.464598
3	5	0.411822
4	6	0.442293
5	7	0.406363
6	8	0.413411
7	9	0.375548





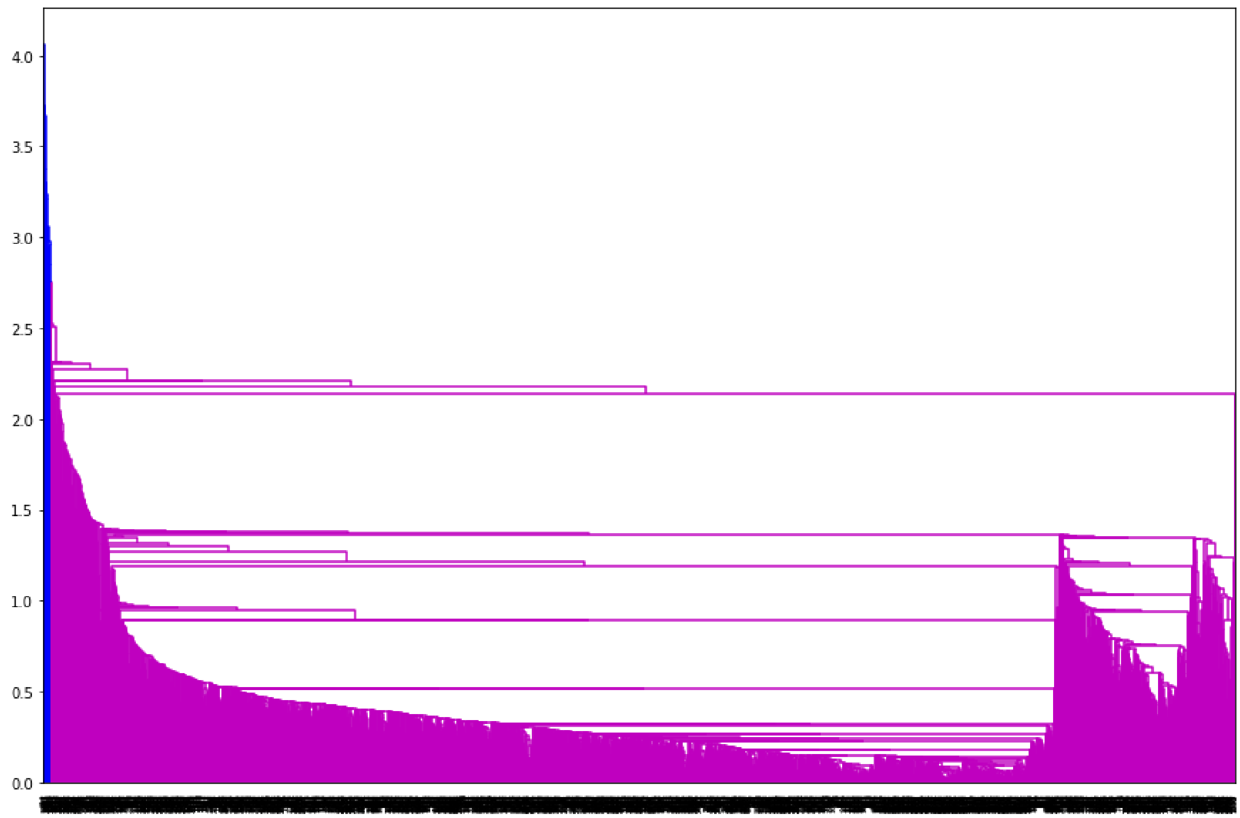
Set the size of the plot
##Better way to plot
plt.figure(figsize=(10,4))

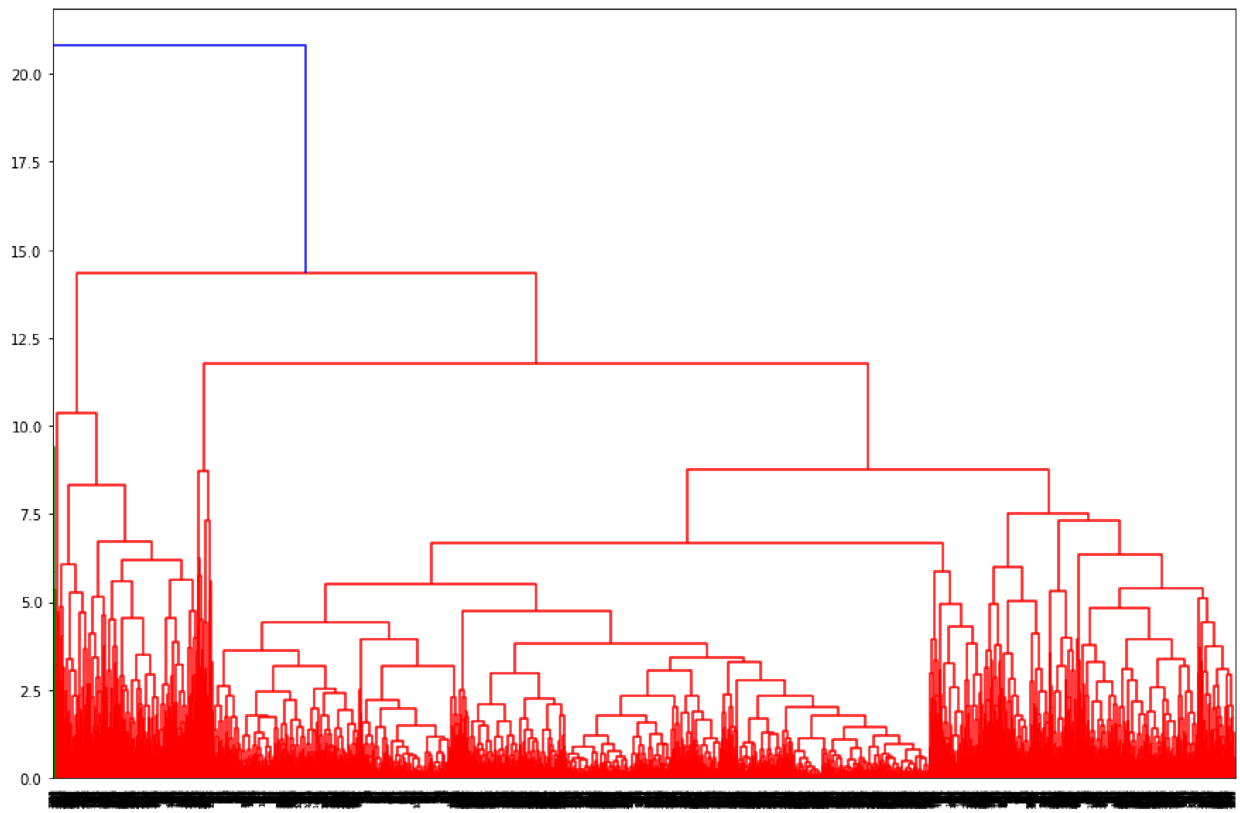
```
plt.subplot(1, 2, 1)
sns.pointplot(data=scoredata,x='index',y='value')
plt.grid(True)
plt.ylabel("Silouette Score")
plt.xlabel("k")
plt.title("Silouette for K-means")
```

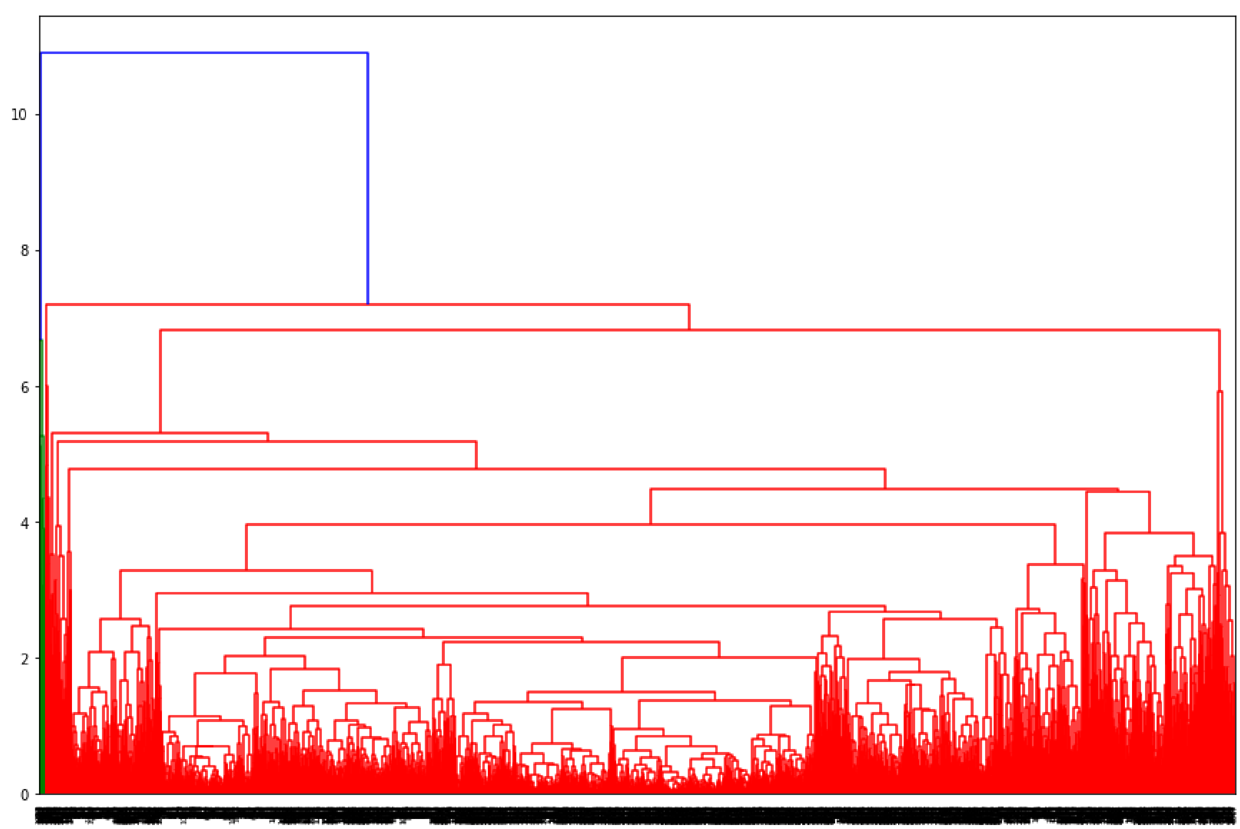



```
from scipy.cluster.hierarchy import linkage, cut_tree, dendrogram
#Hierarchial Clustering:
plt.figure(figsize=(15,10))
mergings = linkage(df_norm, method='single',metric='euclidean')
dendrogram(mergings)
plt.show()

plt.figure(figsize=(15,10))
mergings = linkage(df_norm, method='complete',metric='euclidean')
dendrogram(mergings)
plt.show()

plt.figure(figsize=(15,10))
```

```
mergings = linkage(df_norm, method='average',metric='euclidean')
dendrogram(mergings)
plt.show()
```




0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


0
2
4
6
8
10