## Business Problem 1:

Perform clustering for the crime data and identify the number of clusters formed and draw inferences.

Data Description:

Murder -- Muder rates in different places of United States

Assualt- Assualt rate in different places of United States

UrbanPop - urban population in different places of United States

Rape - Rape rate in different places of United States

### Part 1: Hierarchical Clustering:

### Solution:

**Data loading:**

**Codes:**

```
import pandas as pd
import matplotlib.pylab as plt
crm = pd.read_csv('E:\Data\Assignments\i made\clusterinng\crime_data.csv')
```

**Normalization the Data:**

**Codes:**

```
def norm_func(i):
   x = (i-i.mean())/(i.std())
   return (x)
```

Converting the normalized data into a Data frame by considering only numerical data

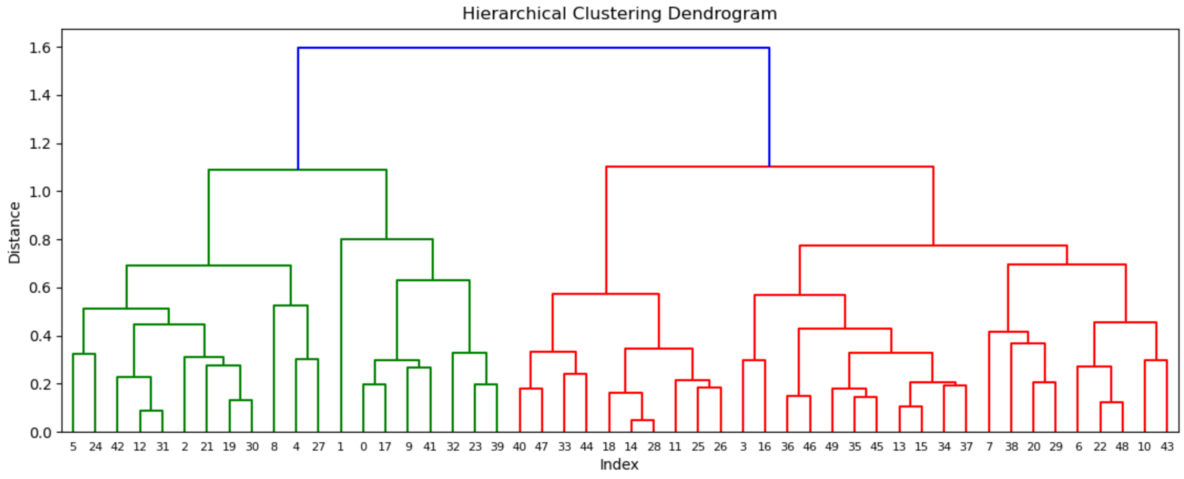
**Codes:**

```
df_norm = norm_func(crm.iloc[:,1:])
```

**Creating Dendrogram:**

**Codes:**

```
from scipy.cluster.hierarchy import linkage

import scipy.cluster.hierarchy as sch


z = linkage(df_norm, method="complete",metric="euclidean")


plt.figure(figsize=(15,5));plt.title('HierarchicalClusteringDendrogram');plt.xlabel('Index');plt.ylabel('Distance')

sch.dendrogram(

   z,

   leaf_rotation=0.

   leaf_font_size=8.

)

plt.show()
```



**Applying Agglomerative Clustering choosing 3 as clusters from the Dendrogram:**

**Codes:**

From sklearn.cluster import AgglomerativeClustering

h_complete=AgglomerativeClustering(n_clusters=3,linkage='complete',affinity="euclidean")
.fit(df_norm)

cluster_labels=pd.Series(h_complete.labels_)

**Creating a new column named as clust and storing it in the data frame.**

crm['clust']=cluster_labels

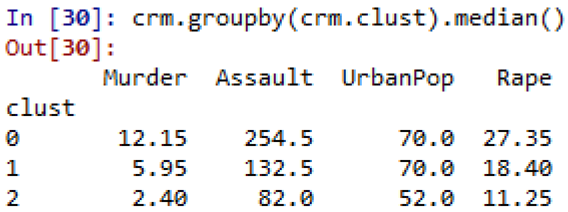
crm = crm.iloc[:,[5,0,1,2,3,4]]

crm.head()

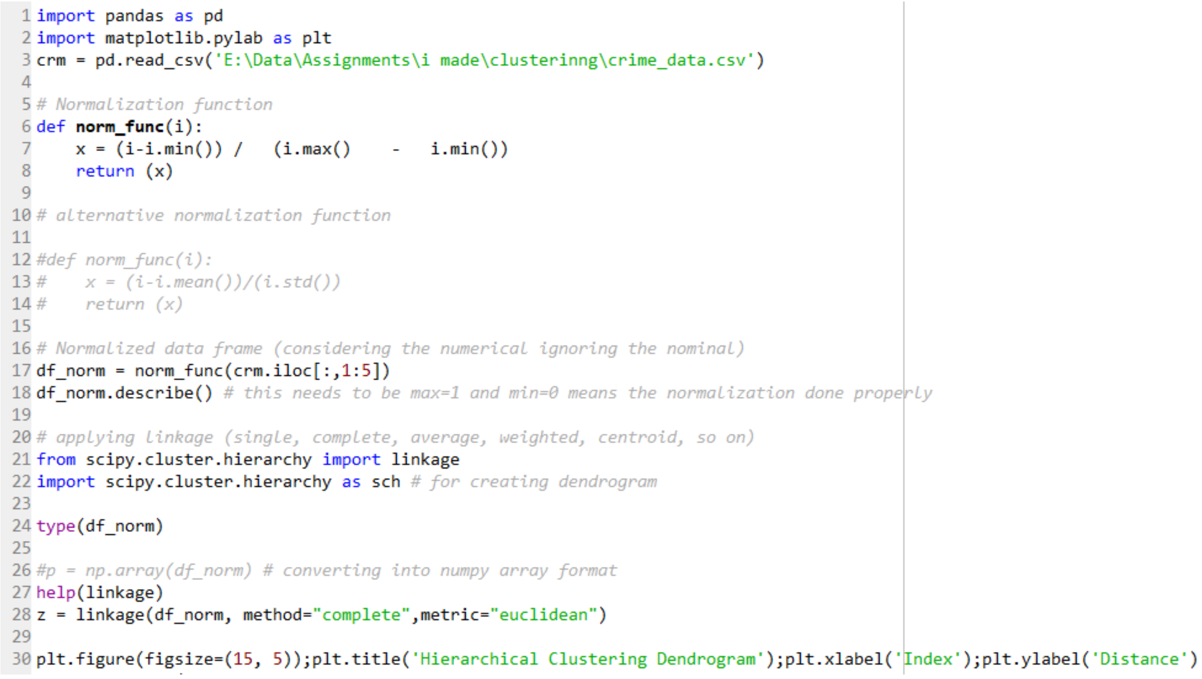
**Getting aggregate mean of each cluster:**

**Codes:**

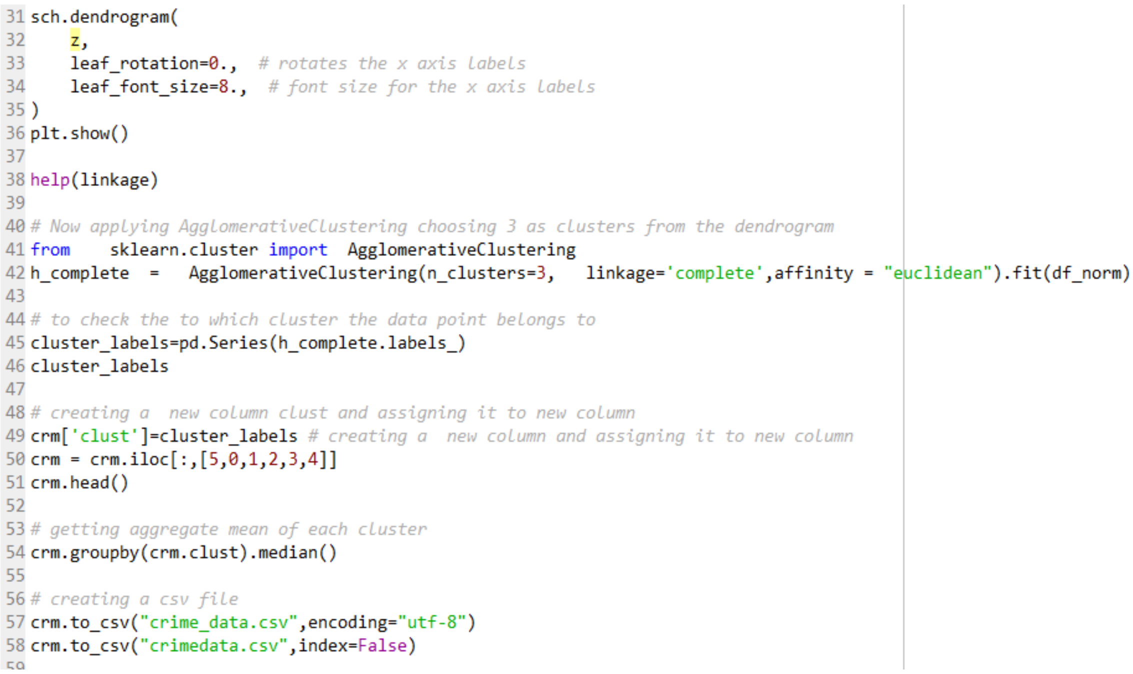
crm.groupby(crm.clust).median()



```
In [30]: crm.groupby(crm.clust).median()
Out[30]:
       Murder  Assault  UrbanPop   Rape
clust
0       12.15    254.5      70.0  27.35
1        5.95    132.5      70.0  18.40
2        2.40     82.0      52.0  11.25
```



```
import pandas as pd
import matplotlib.pylab as plt
crm = pd.read_csv('E:\Data\Assignments\i made\clusteringg\crime_data.csv')

# Normalization function
def norm_func(i):
    x = (i-i.min()) /   (i.max()    -   i.min())
    return (x)

# alternative normalization function

#def norm_func(i):
#    x = (i-i.mean())/(i.std())
#    return (x)

# Normalized data frame (considering the numerical ignoring the nominal)
df_norm = norm_func(crm.iloc[:,1:5])
df_norm.describe() # this needs to be max=1 and min=0 means the normalization done properly

# applying linkage (single, complete, average, weighted, centroid, so on)
from scipy.cluster.hierarchy import linkage
import scipy.cluster.hierarchy as sch # for creating dendrogram

type(df_norm)

#p = np.array(df_norm) # converting into numpy array format
help(linkage)
z = linkage(df_norm, method="complete",metric="euclidean")

plt.figure(figsize=(15, 5));plt.title('Hierarchical Clustering Dendrogram');plt.xlabel('Index');plt.ylabel('Distance')
```



```
sch.dendrogram(
    z,
    leaf_rotation=0.,  # rotates the x axis labels
    leaf_font_size=8.,  # font size for the x axis labels
)
plt.show()

help(linkage)

# Now applying AgglomerativeClustering choosing 3 as clusters from the dendrogram
from    sklearn.cluster import  AgglomerativeClustering
h_complete  =   AgglomerativeClustering(n_clusters=3,   linkage='complete',affinity = "euclidean").fit(df_norm)

# to check the to which cluster the data point belongs to
cluster_labels=pd.Series(h_complete.labels_)
cluster_labels

# creating a  new column clust and assigning it to new column
crm['clust']=cluster_labels # creating a  new column and assigning it to new column
crm = crm.iloc[:,[5,0,1,2,3,4]]
crm.head()

# getting aggregate mean of each cluster
crm.groupby(crm.clust).median()

# creating a csv file
crm.to_csv("crime_data.csv",encoding="utf-8")
crm.to_csv("crimedata.csv",index=False)
```

## Part 2: K-Means Clustering:

## Solution:

**Data loading:**

**Codes:**

```
import pandas as pd

import matplotlib.pylab as plt

from    sklearn.cluster import KMeans

from scipy.spatial.distance import cdist

import numpy as np


crm = pd.read_csv('E:\Data\Assignments\i made\clusterinng\crime_data.csv')
```

**EDA:**

**Normalization function:**

**Codes:**

```
def norm_func(i):

   x = (i-i.min())          /          (i.max()          -          i.min())

   return (x)
```

Forming the Normalized data frame (considering the numerical part of data)

**Codes:**

```
df_norm = norm_func(crm.iloc[:,1:])

df_norm.head()
```

**To get the scree plot or elbow curve:**

**Codes:**

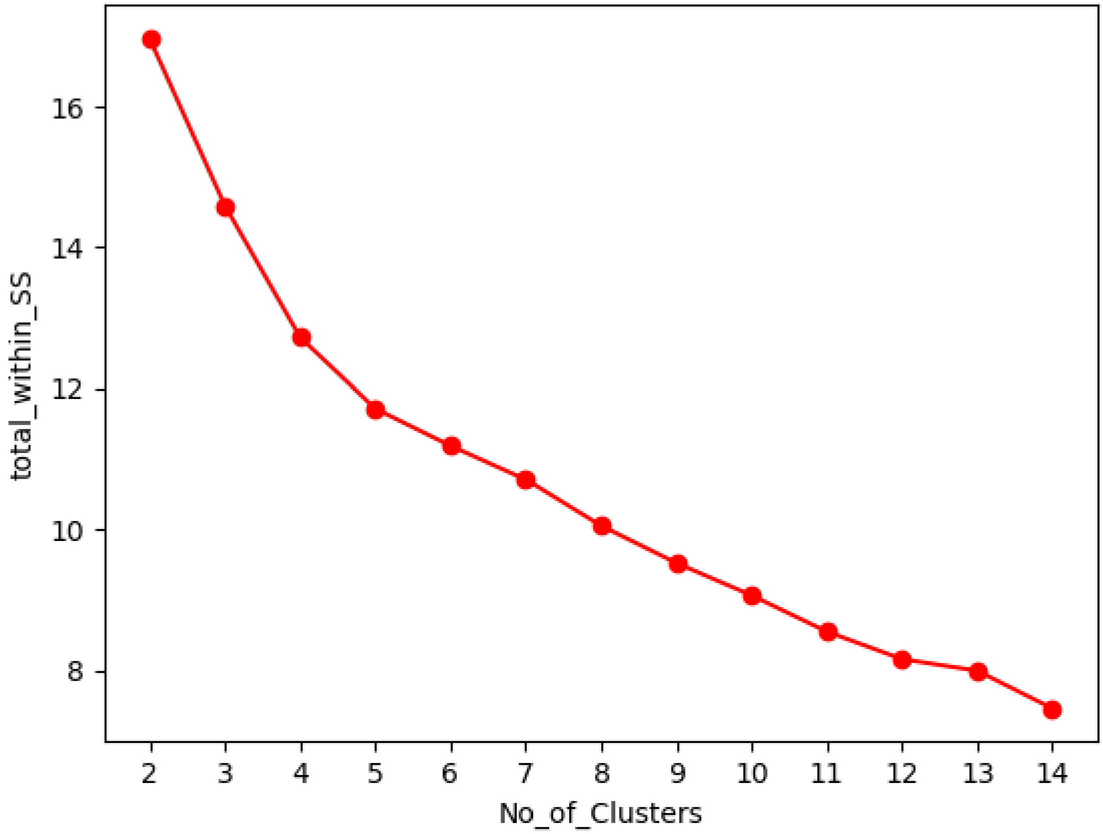
```
k = list(range(2,15))

k
```

```
TWSS = [] # variable for storing total within sum of squares for each kmeans

for i in k:

  kmeans = KMeans(n_clusters = i)

  kmeans.fit(df_norm)

  WSS = [] {variable for storing within sum of squares for each cluster}

  for j in range(i):

WSS.append(sum(cdist(df_norm.iloc[kmeans.labels_==j,:],kmeans.cluster_centers_[j].reshape(1,df_norm.shape[1]),"euclidean")))

  TWSS.append(sum(WSS))
```

**Scree plot:**

**Codes:**

```
plt.plot(k,TWSS, 'ro-');plt.xlabel("No_of_Clusters");plt.ylabel("total_within_SS");plt.xticks(k)
```



**Selecting 5 clusters from the above scree plot which is the optimum number of clusters:**

**Codes:**

model=KMeans(n_clusters=5)

model.fit(df_norm)

**Getting the labels of clusters assigned to each row:**

**Codes:**

model.labels_

**Converting numpy array into pandas series object:**

**Codes:**

md=pd.Series(model.labels_)

**Creating a  new column and assigning it to new column:**

**Codes:**

crm['clust']=md

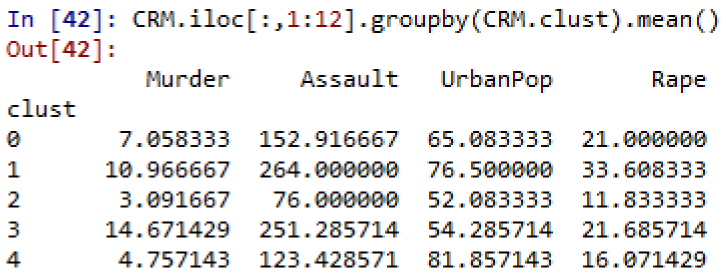
df_norm.head()

**Projecting the clust column at first and rest later:**

**Codes:**

CRM = crm.iloc[:,[5,0,1,2,3,4]]

CRM.iloc[:,1:12].groupby(CRM.clust).mean()

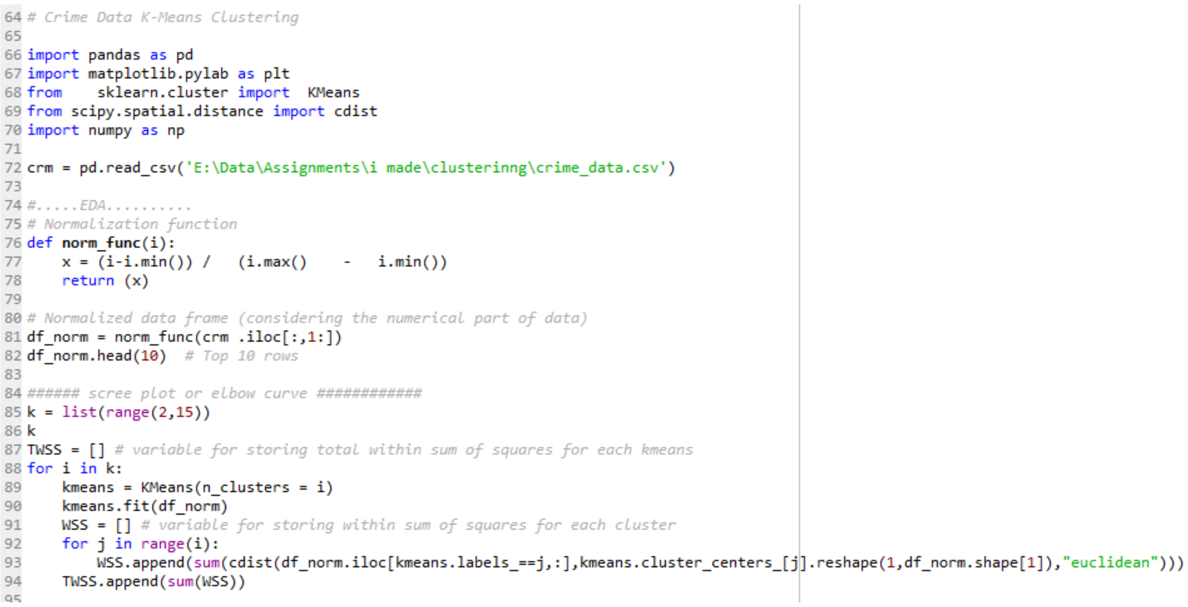


```
In [42]: CRM.iloc[:,1:12].groupby(CRM.clust).mean()
Out[42]:
          Murder     Assault   UrbanPop       Rape
clust
0       7.058333  152.916667  65.083333  21.000000
1      10.966667  264.000000  76.500000  33.608333
2       3.091667   76.000000  52.083333  11.833333
3      14.671429  251.285714  54.285714  21.685714
4       4.757143  123.428571  81.857143  16.071429
```

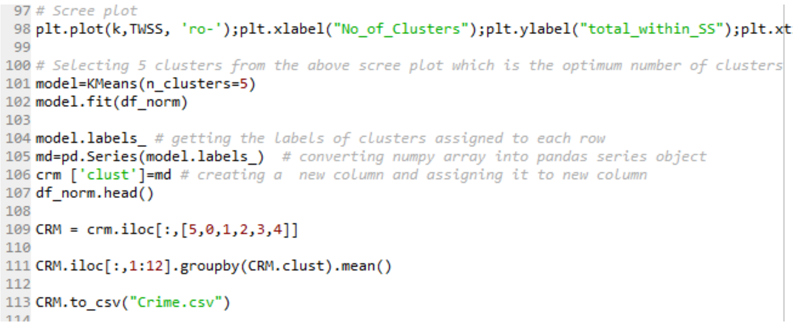
## Exporting the results into a csv file:

## Codes:

CRM.to_csv("Crime.csv")



```
# Crime Data K-Means Clustering

import pandas as pd
import matplotlib.pylab as plt
from    sklearn.cluster import  KMeans
from scipy.spatial.distance import cdist
import numpy as np

crm = pd.read_csv('E:\Data\Assignments\i made\clusterinng\crime_data.csv')

#.....EDA..........
# Normalization function
def norm_func(i):
    x = (i-i.min()) /   (i.max()     -   i.min())
    return (x)

# Normalized data frame (considering the numerical part of data)
df_norm = norm_func(crm .iloc[:,1:])
df_norm.head(10)  # Top 10 rows

###### scree plot or elbow curve ############
k = list(range(2,15))
k
TWSS = [] # variable for storing total within sum of squares for each kmeans
for i in k:
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(df_norm)
    WSS = [] # variable for storing within sum of squares for each cluster
    for j in range(i):
        WSS.append(sum(cdist(df_norm.iloc[kmeans.labels_==j,:],kmeans.cluster_centers_[j].reshape(1,df_norm.shape[1]),"euclidean")))
    TWSS.append(sum(WSS))
```



```
# Scree plot
plt.plot(k,TWSS, 'ro-');plt.xlabel("No_of_Clusters");plt.ylabel("total_within_SS");plt.xticks(k)

# Selecting 5 clusters from the above scree plot which is the optimum number of clusters
model=KMeans(n_clusters=5)
model.fit(df_norm)

model.labels_ # getting the labels of clusters assigned to each row
md=pd.Series(model.labels_)  # converting numpy array into pandas series object
crm ['clust']=md # creating a  new column and assigning it to new column
df_norm.head()

CRM = crm.iloc[:,[5,0,1,2,3,4]]

CRM.iloc[:,1:12].groupby(CRM.clust).mean()

CRM.to_csv("Crime.csv")
```

## Business Problem 2:

**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.

Data Description:

The file EastWestAirlinescontains information on passengers who belong to an airline's frequent flier program. For each passenger the data include information on their mileage history and on different ways they accrued or spent miles in the last year. The goal is to try to identify clusters of passengers that have similar characteristics for the purpose of targeting different segments for different types of mileage offers

ID --Unique ID

Balance--Number of miles eligible for award travel

Qual_mile--Number of miles counted as qualifying for Topflight status

cc1_miles -- Number of miles earned with freq. flyer credit card in the past 12 months:

cc2_miles -- Number of miles earned with Rewards credit card in the past 12 months:

cc3_miles -- Number of miles earned with Small Business credit card in the past 12 months:

1 = under 5,000

2 = 5,000 - 10,000

3 = 10,001 - 25,000

4 = 25,001 - 50,000

5 = over 50,000

Bonus_miles--Number of miles earned from non-flight bonus transactions in the past 12 months

Bonus_trans--Number of non-flight bonus transactions in the past 12 months

Flight_miles_12mo--Number of flight miles in the past 12 months

Flight_trans_12--Number of flight transactions in the past 12 months

Days_since_enrolled--Number of days since enrolled in flier program

Award--whether that person had award flight (free flight) or not

## Part 1: Hierarchical Clustering:

## Solution:

**Data loading:**

**Codes:**

```
import pandas as pd

import matplotlib.pylab as plt

ewa = pd.read_csv('E:\Data\Assignments\i made\clusterinng\EastWestAirlines.csv')
```

**Normalization the Data:**

**Codes:**

```
def norm_func(i):

   x = (i-i.mean())/(i.std())

   return (x)
```

Converting the normalized data into a Data frame by considering only numerical data

**Codes:**

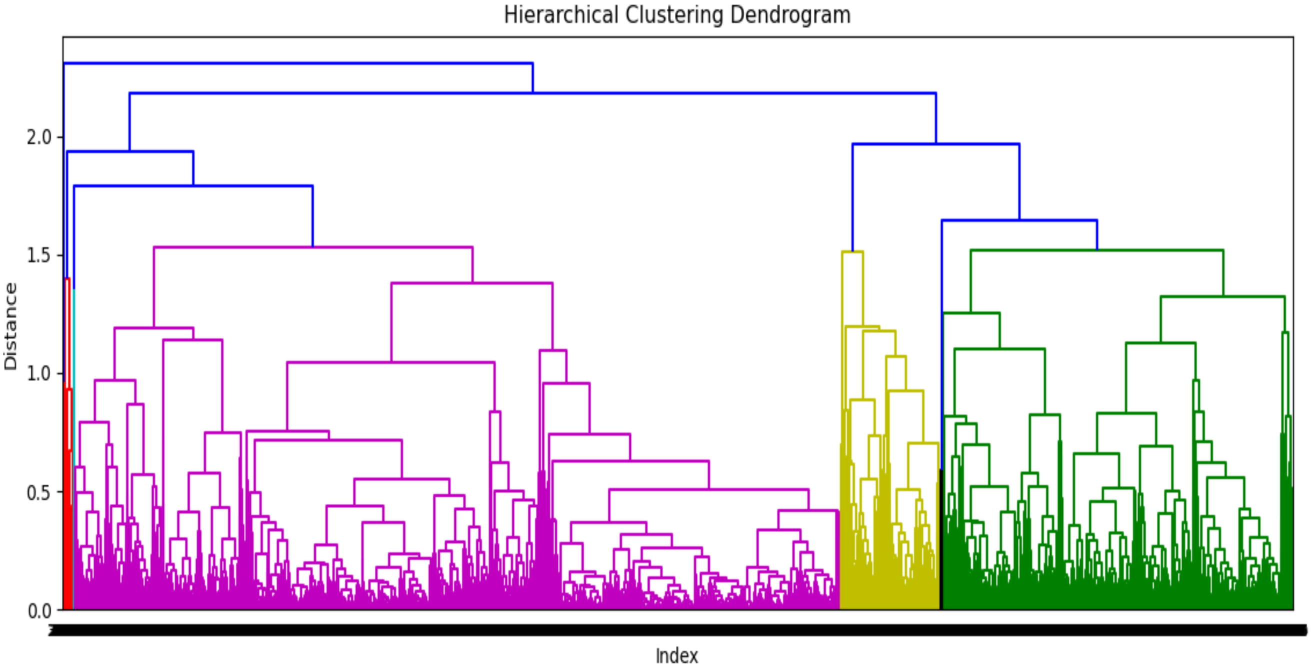
```
df_norm = norm_func(EWA.iloc[:,0:12])

df_norm.describe()
```

**Creating Dendrogram:**

**Codes:**

```
from scipy.cluster.hierarchy import linkage

import scipy.cluster.hierarchy as sch
```

```
z = linkage(df_norm, method="complete",metric="euclidean")


plt.figure(figsize=(15,5));plt.title('HierarchicalClusteringDendrogram');plt.xlabel('Index');plt.y
label('Distance')

sch.dendrogram(

   z,

   leaf_rotation=0.

   leaf_font_size=8.

)

plt.show()
```



**Applying Agglomerative Clustering choosing 3 as clusters from the Dendrogram:**

**Codes:**

```
From sklearn.cluster import AgglomerativeClustering
```

h_complete=AgglomerativeClustering(n_clusters=3,linkage='complete',affinity="euclidean") .fit(df_norm)

cluster_labels=pd.Series(h_complete.labels_)

**Creating a new column named as clust and storing it in the data frame.**

EWA ['clust']=cluster_labels

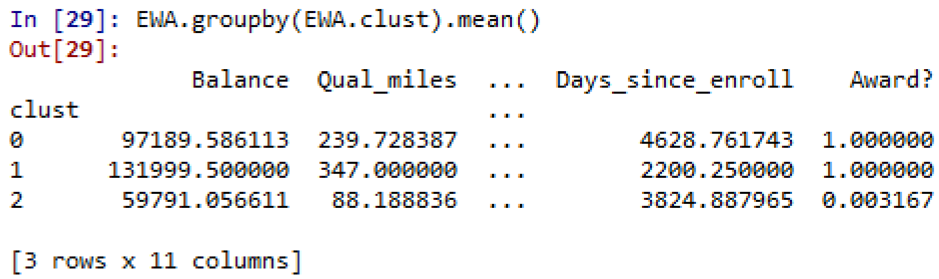
EWA = crm.iloc[:,[5,0,1,2,3,4]]

EWA.head()

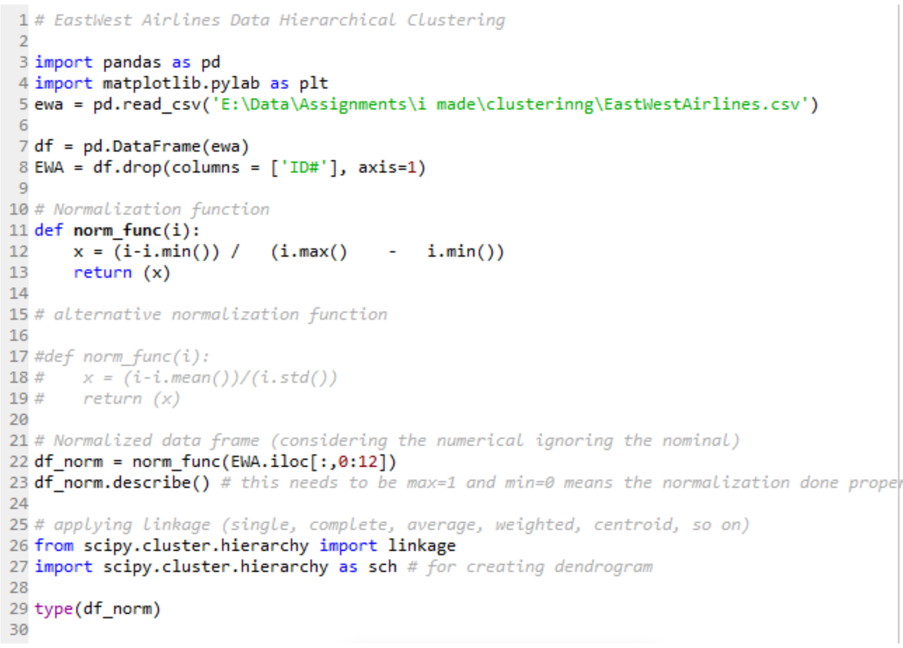
**Getting aggregate mean of each cluster:**

**Codes:**

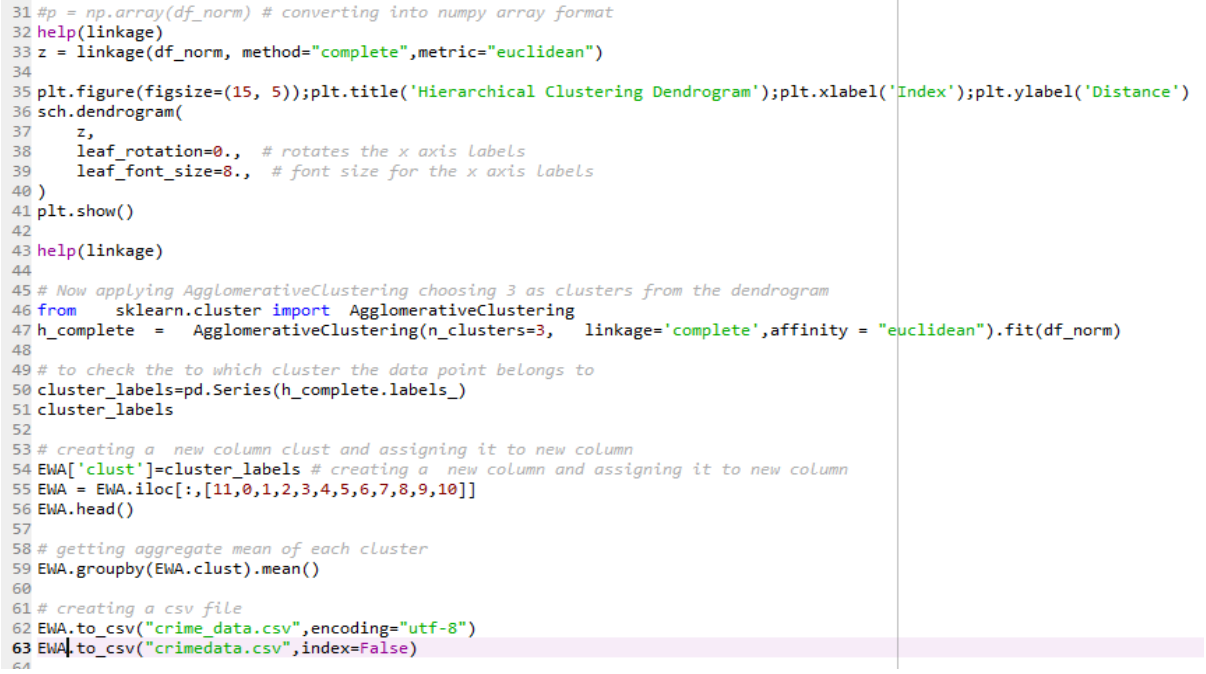
EWA.groupby(EWA.clust).mean()



```
In [29]: EWA.groupby(EWA.clust).mean()
Out[29]:
             Balance  Qual_miles  ...  Days_since_enroll    Award?
clust                             ...
0       97189.586113  239.728387  ...        4628.761743  1.000000
1      131999.500000  347.000000  ...        2200.250000  1.000000
2       59791.056611   88.188836  ...        3824.887965  0.003167

[3 rows x 11 columns]
```



```
# EastWest Airlines Data Hierarchical Clustering

import pandas as pd
import matplotlib.pylab as plt
ewa = pd.read_csv('E:\Data\Assignments\i made\clusteringg\EastWestAirlines.csv')

df = pd.DataFrame(ewa)
EWA = df.drop(columns = ['ID#'], axis=1)

# Normalization function
def norm_func(i):
    x = (i-i.min()) /   (i.max()    -   i.min())
    return (x)

# alternative normalization function

#def norm_func(i):
#    x = (i-i.mean())/(i.std())
#    return (x)

# Normalized data frame (considering the numerical ignoring the nominal)
df_norm = norm_func(EWA.iloc[:,0:12])
df_norm.describe() # this needs to be max=1 and min=0 means the normalization done properly

# applying linkage (single, complete, average, weighted, centroid, so on)
from scipy.cluster.hierarchy import linkage
import scipy.cluster.hierarchy as sch # for creating dendrogram

type(df_norm)

```



```
#p = np.array(df_norm) # converting into numpy array format
help(linkage)
z = linkage(df_norm, method="complete",metric="euclidean")

plt.figure(figsize=(15, 5));plt.title('Hierarchical Clustering Dendrogram');plt.xlabel('Index');plt.ylabel('Distance')
sch.dendrogram(
    z,
    leaf_rotation=0.,  # rotates the x axis labels
    leaf_font_size=8.,  # font size for the x axis labels
)
plt.show()

help(linkage)

# Now applying AgglomerativeClustering choosing 3 as clusters from the dendrogram
from    sklearn.cluster import  AgglomerativeClustering
h_complete  =   AgglomerativeClustering(n_clusters=3,   linkage='complete',affinity = "euclidean").fit(df_norm)

# to check the to which cluster the data point belongs to
cluster_labels=pd.Series(h_complete.labels_)
cluster_labels

# creating a  new column clust and assigning it to new column
EWA['clust']=cluster_labels # creating a  new column and assigning it to new column
EWA = EWA.iloc[:,[11,0,1,2,3,4,5,6,7,8,9,10]]
EWA.head()

# getting aggregate mean of each cluster
EWA.groupby(EWA.clust).mean()

# creating a csv file
EWA.to_csv("crime_data.csv",encoding="utf-8")
EWA.to_csv("crimedata.csv",index=False)
```

## Part 2: K-Means Clustering:

## Solution:

**Data loading:**

**Codes:**

```
import pandas as pd

import matplotlib.pylab as plt

from sklearn.cluster import KMeans

from scipy.spatial.distance import cdist

import numpy as np


ewa = pd.read_csv('E:\Data\Assignments\i made\clusterinng\EastWestAirlines.csv')
```

**EDA:**

**Normalization function:**

**Codes:**

```
def norm_func(i):
   x = (i-i.min()) / (i.max() - i.min())
   return (x)
```

Forming the Normalized data frame (considering the numerical part of data)

**Codes:**

```
df_norm = norm_func(ewa.iloc[:,1:])

df_norm.head()
```

**To get the scree plot or elbow curve:**

**Codes:**

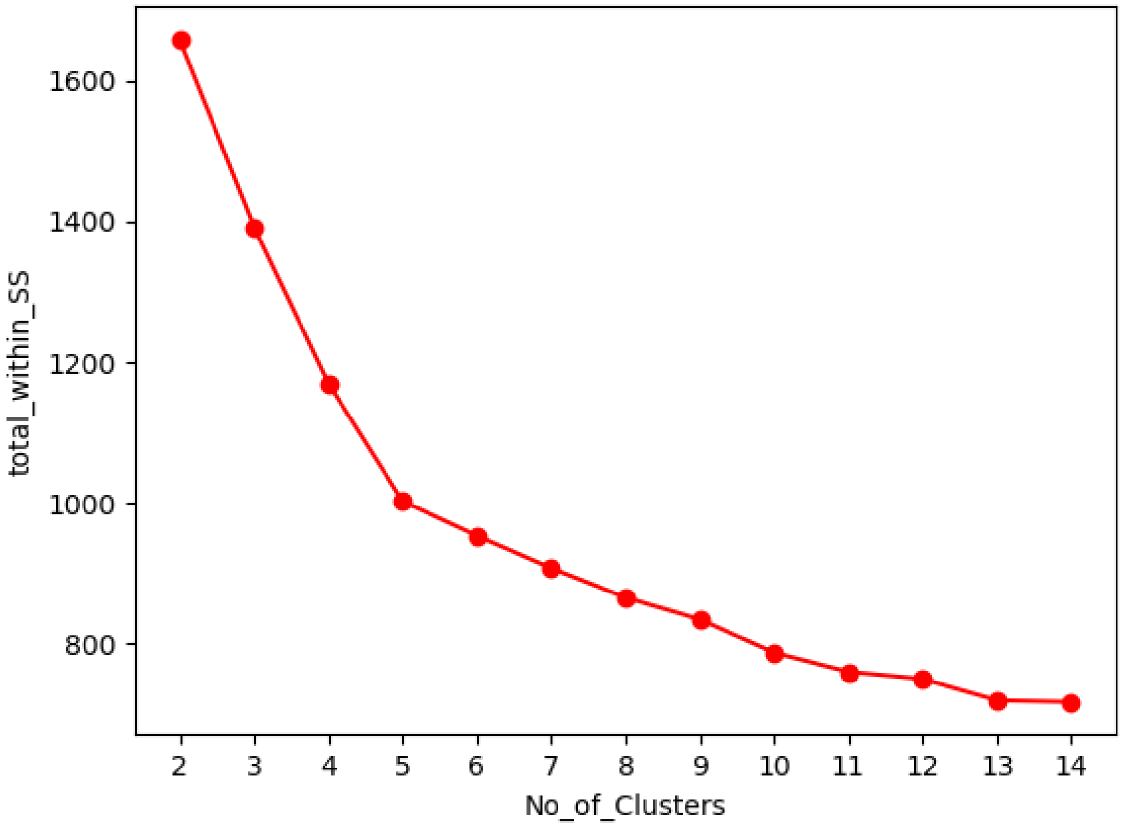
```
k = list(range(2,15))

k
```

```
TWSS = [] # variable for storing total within sum of squares for each kmeans
for i in k:
   kmeans = KMeans(n_clusters = i)
   kmeans.fit(df_norm)
   WSS = [] {variable for storing within sum of squares for each cluster}
   for j in range(i):
WSS.append(sum(cdist(df_norm.iloc[kmeans.labels_==j,:],kmeans.cluster_centers_[j].reshape(1,df_norm.shape[1]),"euclidean")))
   TWSS.append(sum(WSS))
```

**Scree plot:**

**Codes:**

```
plt.plot(k,TWSS, 'ro-');plt.xlabel("No_of_Clusters");plt.ylabel("total_within_SS");plt.xticks(k)
```



**Selecting 5 clusters from the above scree plot which is the optimum number of clusters:**

**Codes:**

model=KMeans(n_clusters=5)

model.fit(df_norm)

**Getting the labels of clusters assigned to each row:**

**Codes:**

model.labels_

**Converting numpy array into pandas series object:**

**Codes:**

md=pd.Series(model.labels_)

**Creating a  new column and assigning it to new column:**

**Codes:**

ewa['clust']=md

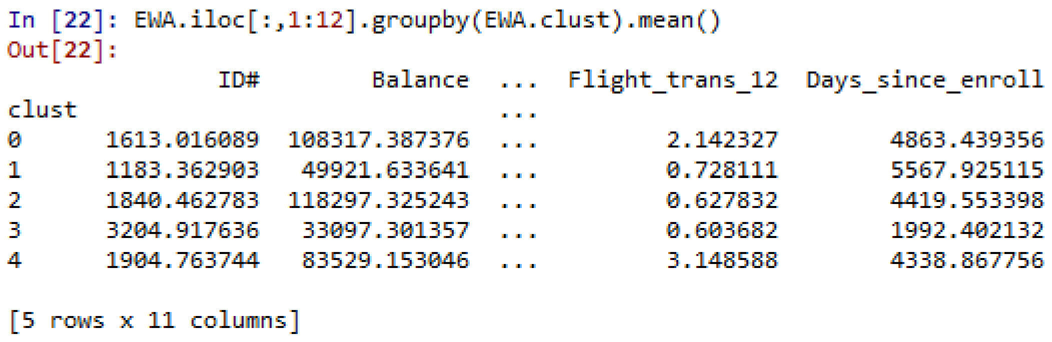
df_norm.head()

**Projecting the clust column at first and rest later:**

**Codes:**

EWA = ewa.iloc[:,[12,0,1,2,3,4,5,6,7,8,9,10,11]]

EWA.iloc[:,1:12].groupby(EWA.clust).mean()

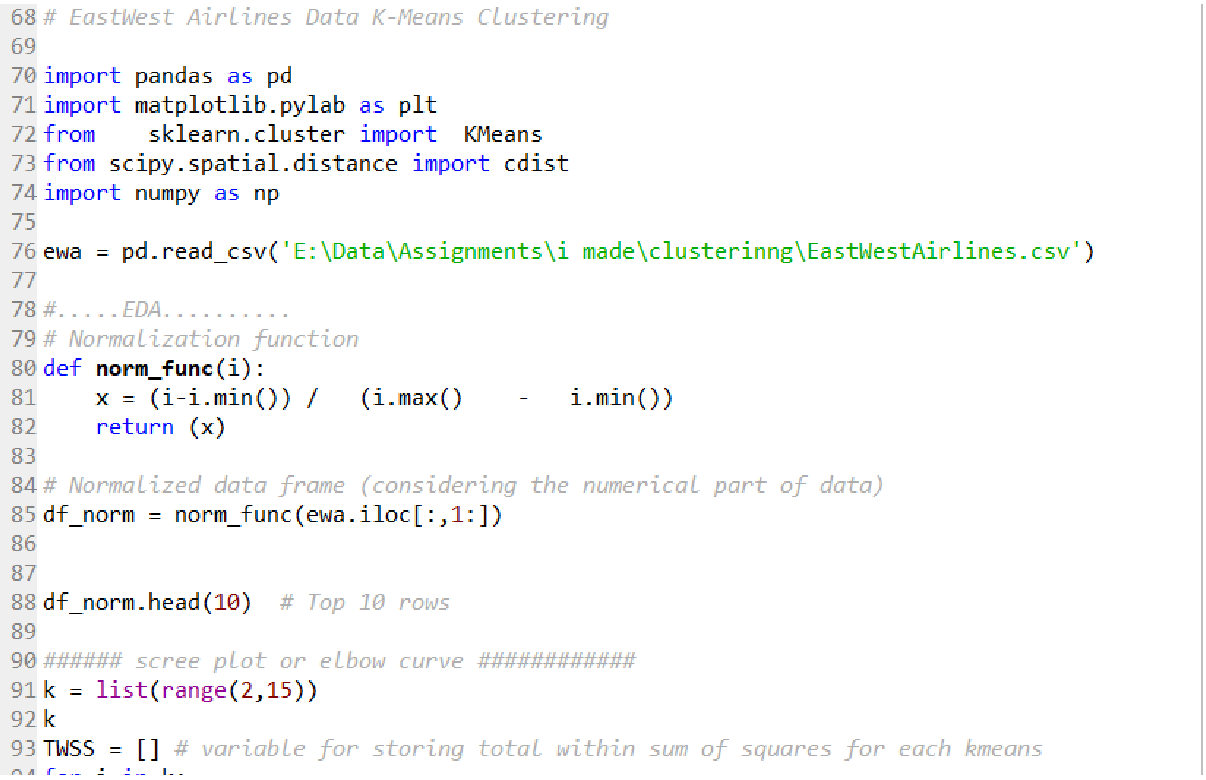


```
In [22]: EWA.iloc[:,1:12].groupby(EWA.clust).mean()
Out[22]:
               ID#        Balance  ...  Flight_trans_12  Days_since_enroll
clust                              ...
0      1613.016089  108317.387376  ...         2.142327        4863.439356
1      1183.362903   49921.633641  ...         0.728111        5567.925115
2      1840.462783  118297.325243  ...         0.627832        4419.553398
3      3204.917636   33097.301357  ...         0.603682        1992.402132
4      1904.763744   83529.153046  ...         3.148588        4338.867756

[5 rows x 11 columns]
```

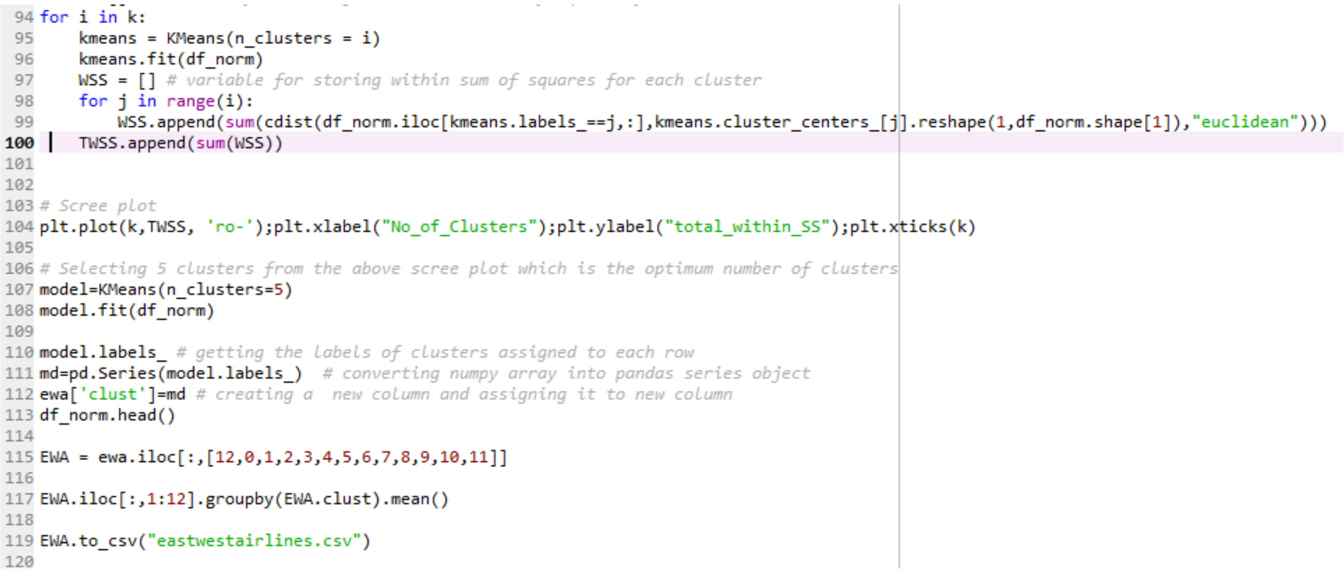
### Exporting the results into a csv file:

### Codes:

EWA.to_csv("eastwestairlines.csv")



```
# EastWest Airlines Data K-Means Clustering

import pandas as pd
import matplotlib.pylab as plt
from    sklearn.cluster import  KMeans
from scipy.spatial.distance import cdist
import numpy as np

ewa = pd.read_csv('E:\Data\Assignments\i made\clusterinng\EastWestAirlines.csv')

#.....EDA..........
# Normalization function
def norm_func(i):
    x = (i-i.min()) /   (i.max()    -   i.min())
    return (x)

# Normalized data frame (considering the numerical part of data)
df_norm = norm_func(ewa.iloc[:,1:])


df_norm.head(10)  # Top 10 rows

###### scree plot or elbow curve ############
k = list(range(2,15))
k
TWSS = [] # variable for storing total within sum of squares for each kmeans
```



```
for i in k:
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(df_norm)
    WSS = [] # variable for storing within sum of squares for each cluster
    for j in range(i):
        WSS.append(sum(cdist(df_norm.iloc[kmeans.labels_==j,:],kmeans.cluster_centers_[j].reshape(1,df_norm.shape[1]),"euclidean")))
    TWSS.append(sum(WSS))


# Scree plot
plt.plot(k,TWSS, 'ro-');plt.xlabel("No_of_Clusters");plt.ylabel("total_within_SS");plt.xticks(k)

# Selecting 5 clusters from the above scree plot which is the optimum number of clusters
model=KMeans(n_clusters=5)
model.fit(df_norm)

model.labels_ # getting the labels of clusters assigned to each row
md=pd.Series(model.labels_)  # converting numpy array into pandas series object
ewa['clust']=md # creating a  new column and assigning it to new column
df_norm.head()

EWA = ewa.iloc[:,[12,0,1,2,3,4,5,6,7,8,9,10,11]]

EWA.iloc[:,1:12].groupby(EWA.clust).mean()

EWA.to_csv("eastwestairlines.csv")
```