**Jupyter Notebook**

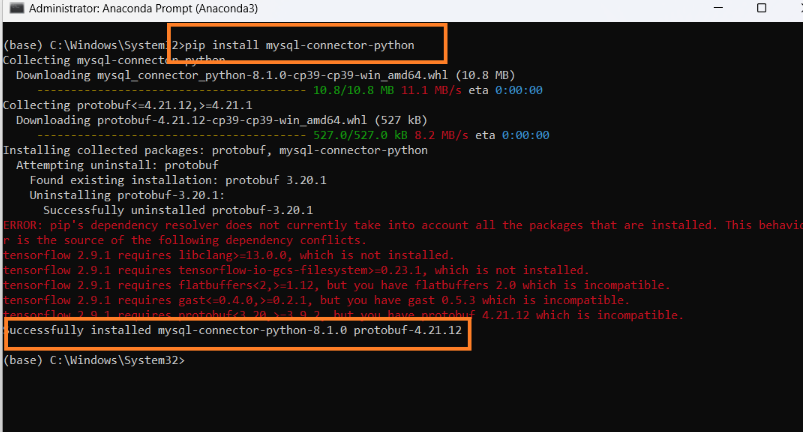
**Clustering Account IDs based on Loan Amount and Balance**

**Independent variables**

* Account\_id
* loan\_amount
* Balance

**Install MySQL Connector**

pip install mysql-connector-python



**Important Libraries**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn.cluster import KMeans

from scipy.cluster.hierarchy import dendrogram, linkage

from sklearn.cluster import AgglomerativeClustering

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 20, 10

**Create SQL**

sql= "select \* from acc\_ord\_card\_disp\_client\_dist aocdcd join loan\_trans lt on lt.account\_id= aocdcd.account\_id"

**Create Connection and Read Data as data Frame**

import mysql.connector as connection

try:

mydb = connection.connect(host="localhost", database = 'capstone\_prj',user="root", passwd="Bangalore",use\_pure=True)

df = pd.read\_sql(sql,mydb)

mydb.close() #close the connection

except Exception as e:

mydb.close()

print(str(e))

**Display Data**

display (df)

**Display Shape**

display (df.shape)

**Display all columns**

df.columns

**Display Null values**

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

**Create x – independent variable**

x=df[['loan\_amount','balance']].values

display (x)

**Min Max Scaler**

from sklearn.preprocessing import MinMaxScaler

minmax= MinMaxScaler()

minmax\_x = minmax.fit\_transform(x)

print (minmax\_x)

**Elbow method -WCSS**

wcss = []

for i in range(2,15):

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

kmeans.fit(minmax\_x)

wcss.append (kmeans.inertia\_)

display (wcss)

**Plot Graph**

plt.plot (range(2,15),wcss)

plt.title('Elbow Method')

plt.xlabel('Number of clusters')

plt.ylabel ('wcss')

plt.show()

**Create K Means with 6 Clusters**

kmeans = KMeans (n\_clusters = 6 , init = 'k-means++')

kmeans = kmeans.fit(minmax\_x)

y\_kmeans =kmeans.labels\_

display (y\_kmeans)

**Plot Clusters**

plt.scatter (minmax\_x[y\_kmeans == 0,0],minmax\_x[y\_kmeans == 0,1] ,s = 100, c='r', label = 'Cluster 1')

plt.scatter (minmax\_x[y\_kmeans == 1,0],minmax\_x[y\_kmeans == 1,1] ,s = 100, c='b', label = 'Cluster 2')

plt.scatter (minmax\_x[y\_kmeans == 2,0],minmax\_x[y\_kmeans == 2,1] ,s = 100, c='g', label = 'Cluster 3')

plt.scatter (minmax\_x[y\_kmeans == 3,0],minmax\_x[y\_kmeans == 3,1] ,s = 100, c='c', label = 'Cluster 4')

plt.scatter (minmax\_x[y\_kmeans == 4,0],minmax\_x[y\_kmeans == 4,1] ,s = 100, c='m', label = 'Cluster 5')

plt.scatter (minmax\_x[y\_kmeans == 5,0],minmax\_x[y\_kmeans == 5,1] ,s = 100, c='k', label = 'Cluster 6')

plt.legend()

plt.show()

**Create Data Frame with Account ID, Loan Amount, Balance and Clusters**

x\_final=pd.concat([df.iloc[:,4],pd.DataFrame(x), pd.DataFrame(y\_kmeans)], axis =1 )

x\_final.columns= ['account\_id','loan\_amount','balance','cluster']

x\_final.to\_excel('C:\\Noble\\Training\\Course Content\\My SQL\\MY SQL Project Python\\K\_Means.xlsx', index =None)

display (x\_final)

**Number of records in cluster**

display (x\_final.cluster.value\_counts())

**Hierarchical Clustering**

**Create X – Loan Amount and Balance**

x\_hc=df[['loan\_amount','balance']].values

display (x\_hc)

**Min Max Scaler**

from sklearn.preprocessing import MinMaxScaler

minmax= MinMaxScaler()

minmax\_x = minmax.fit\_transform(x\_hc)

print (minmax\_x)

**Create Dendrogram with Method – Centroid**

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(minmax\_x, method='centroid'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('ED')

plt.show()

**Create Dendrogram with Method – Single**

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(minmax\_x, method='single'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('ED')

plt.show()

plt.show()

**Create Dendrogram with Method – ward**

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(minmax\_x, method='ward'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('ED')

plt.show()

**Create Dendrogram with Method – Average**

import scipy.cluster.hierarchy as sch

dendrogram = sch.dendrogram(sch.linkage(minmax\_x, method='average'))

plt.title('Dendrogram')

plt.xlabel('Customers')

plt.ylabel('ED')

plt.show()

**AgglomerativeClustering with Linkage = ward**

from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters=2, metric='euclidean', linkage='ward')

y\_hc = hc.fit\_predict(minmax\_x)

display (y\_hc)

**Value Count -Number of records in Each cluster**

pd.DataFrame(y\_hc).value\_counts()

**Plot Graph**

plt.scatter(minmax\_x [y\_hc==0,0], minmax\_x [y\_hc==0,1], s=100, c='red', label='Cluster1')

plt.scatter(minmax\_x [y\_hc==1,0], minmax\_x [y\_hc==1,1], s=100, c='blue', label='Cluster2')

plt.legend()

plt.show()

**AgglomerativeClustering with Linkage = Average**

from sklearn.cluster import AgglomerativeClustering

hc = AgglomerativeClustering(n\_clusters=2, metric ='euclidean', linkage='average')

y\_hc = hc.fit\_predict(minmax\_x)

display (y\_hc)

**Value Count -Number of records in Each cluster**

pd.DataFrame(y\_hc).value\_counts()

**Plot Graph**

plt.scatter(minmax\_x[y\_hc==0,0], minmax\_x [y\_hc==0,1], s=100, c='red', label='Cluster1')

plt.scatter(minmax\_x [y\_hc==1,0], minmax\_x [y\_hc==1,1], s=100, c='blue', label='Cluster2')

plt.legend()

plt.show()

**Display the cluster and export Data to Excel**

xhc\_final=pd.concat([df.iloc[:,4],pd.DataFrame(x\_hc), pd.DataFrame(y\_hc)], axis =1 )

xhc\_final.columns= ['account\_id','loan\_amount','balance','cluster']

xhc\_final.to\_excel('C:\\Noble\\Training\\Course Content\\My SQL\\MY SQL Project Python\\hc.xlsx',index =None)

display (xhc\_final)