https://www.kaggle.com/datasets/kyanyoga/sample-sales-data

Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#Importing the required libraries.

from sklearn.cluster import KMeans, k_means #For clustering
from sklearn.decomposition import PCA #Linear Dimensionality reduction.

df = pd.read_csv("sales_data_sample.csv") #Loading the dataset.

#Preprocessing

df.head()

df.shape

df.describe()

df.info()

df.isnull().sum()

df.dtypes

df_drop  = ['ADDRESSLINE1', 'ADDRESSLINE2', 'STATUS','POSTALCODE', 'CITY',
'TERRITORY', 'PHONE', 'STATE', 'CONTACTFIRSTNAME', 'CONTACTLASTNAME',
'CUSTOMERNAME', 'ORDERNUMBER']
df = df.drop(df_drop, axis=1) #Dropping the categorical uneccessary columns
along with columns having null values. Can't fill the null values are there are
alot of null values

df.isnull().sum()

df.dtypes

# Checking the categorical columns.
df['COUNTRY'].unique()

df['PRODUCTLINE'].unique()

df['DEALSIZE'].unique()

productline = pd.get_dummies(df['PRODUCTLINE']) #Converting the categorical
columns.
Dealsize = pd.get_dummies(df['DEALSIZE'])


df = pd.concat([df,productline,Dealsize], axis = 1)
```

```
df_drop  = ['COUNTRY','PRODUCTLINE','DEALSIZE'] #Dropping Country too as there
are alot of countries.
df = df.drop(df_drop, axis=1)


df['PRODUCTCODE'] = pd.Categorical(df['PRODUCTCODE']).codes #Converting the
datatype.


df.drop('ORDERDATE', axis=1, inplace=True) #Dropping the Orderdate as Month is
already included.

df.dtypes

Plotting the Elbow Plot to determine the number of clusters.

distortions = [] # Within Cluster Sum of Squares from the centroid
K = range(1,10)
for k in K:
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(df)
    distortions.append(kmeanModel.inertia_)   #Appeding the intertia to the
Distortions


plt.figure(figsize=(16,8))
plt.plot(K, distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()

As the number of k increases Inertia decreases.
Observations: A Elbow can be observed at 3 and after that the curve decreases
gradually.


X_train = df.values #Returns a numpy array.

X_train.shape

model = KMeans(n_clusters=3,random_state=2) #Number of cluster = 3
model = model.fit(X_train) #Fitting the values to create a model.
predictions = model.predict(X_train) #Predicting the cluster values (0,1,or 2)

unique,counts = np.unique(predictions,return_counts=True)

counts = counts.reshape(1,3)

counts_df=pd.DataFrame(counts,columns=['Cluster1','Cluster2','Cluster3'])

counts_df.head()
```

Visualization

```
pca = PCA(n_components=2) #Converting all the features into 2 columns to make it
easy to visualize using Principal COmponent Analysis.

reduced_X=pd.DataFrame(pca.fit_transform(X_train),columns=['PCA1','PCA2'])
#Creating a DataFrame.

reduced_X.head()

#Plotting the normal Scatter Plot
plt.figure(figsize=(14,10))
plt.scatter(reduced_X['PCA1'],reduced_X['PCA2'])

model.cluster_centers_ #Finding the centriods. (3 Centriods in total. Each Array
contains a centroids for particular feature )

reduced_centers = pca.transform(model.cluster_centers_) #Transforming the
centroids into 3 in x and y coordinates

reduced_centers

plt.figure(figsize=(14,10))
plt.scatter(reduced_X['PCA1'],reduced_X['PCA2'])
plt.scatter(reduced_centers[:,0],reduced_centers[:,1],color='black',marker='x',s
=300) #Plotting the centroids

reduced_X['Clusters'] = predictions #Adding the Clusters to the reduced
dataframe.

reduced_X.head()

#Plotting the clusters
plt.figure(figsize=(14,10))
#                     taking the cluster number and first column
taking the same cluster number and second column      Assigning the color
plt.scatter(reduced_X[reduced_X['Clusters'] ==
0].loc[:,'PCA1'],reduced_X[reduced_X['Clusters'] ==
0].loc[:,'PCA2'],color='slateblue')
plt.scatter(reduced_X[reduced_X['Clusters'] ==
1].loc[:,'PCA1'],reduced_X[reduced_X['Clusters'] ==
1].loc[:,'PCA2'],color='springgreen')
plt.scatter(reduced_X[reduced_X['Clusters'] ==
2].loc[:,'PCA1'],reduced_X[reduced_X['Clusters'] ==
2].loc[:,'PCA2'],color='indigo')


plt.scatter(reduced_centers[:,0],reduced_centers[:,1],color='black',marker='x',s
=300)
```