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

In [198]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#Importing the required libraries.
```

In [199]:
```
from sklearn.cluster import KMeans, k_means #For clustering
from sklearn.decomposition import PCA #Linear Dimensionality reduction.
```

In [200]:
```
df = pd.read_csv("sales_data_sample.csv") #Loading the dataset.
```

Preprocessing

In [201]:
```
df.head()
```

...

In [202]:
```
df.shape
```

...

In [203]:
```
df.describe()
```

...

In [204]:
```
df.info()
```

...

In [205]:
```
df.isnull().sum()
```

...

In [206]:
```
df.dtypes
```

...

In [207]:
```
df_drop  = ['ADDRESSLINE1', 'ADDRESSLINE2', 'STATUS','POSTALCODE', 'CITY', 'TERRI
df = df.drop(df_drop, axis=1) #Dropping the categorical uneccessary columns along
```

In [208]:
```
df.isnull().sum()
```
...

In [209]:
```
df.dtypes
```
...

In [ ]:
```
# Checking the categorical columns.
```

In [210]:
```
df['COUNTRY'].unique()
```
...

In [211]:
```
df['PRODUCTLINE'].unique()
```
...

In [212]:
```
df['DEALSIZE'].unique()
```
...

In [213]:
```
productline = pd.get_dummies(df['PRODUCTLINE']) #Converting the categorical colum
Dealsize = pd.get_dummies(df['DEALSIZE'])
```

In [214]:
```
df = pd.concat([df,productline,Dealsize], axis = 1)
```

In [215]:
```
df_drop  = ['COUNTRY','PRODUCTLINE','DEALSIZE'] #Dropping Country too as there ar
df = df.drop(df_drop, axis=1)
```

In [216]:
```
df['PRODUCTCODE'] = pd.Categorical(df['PRODUCTCODE']).codes #Converting the datat
```

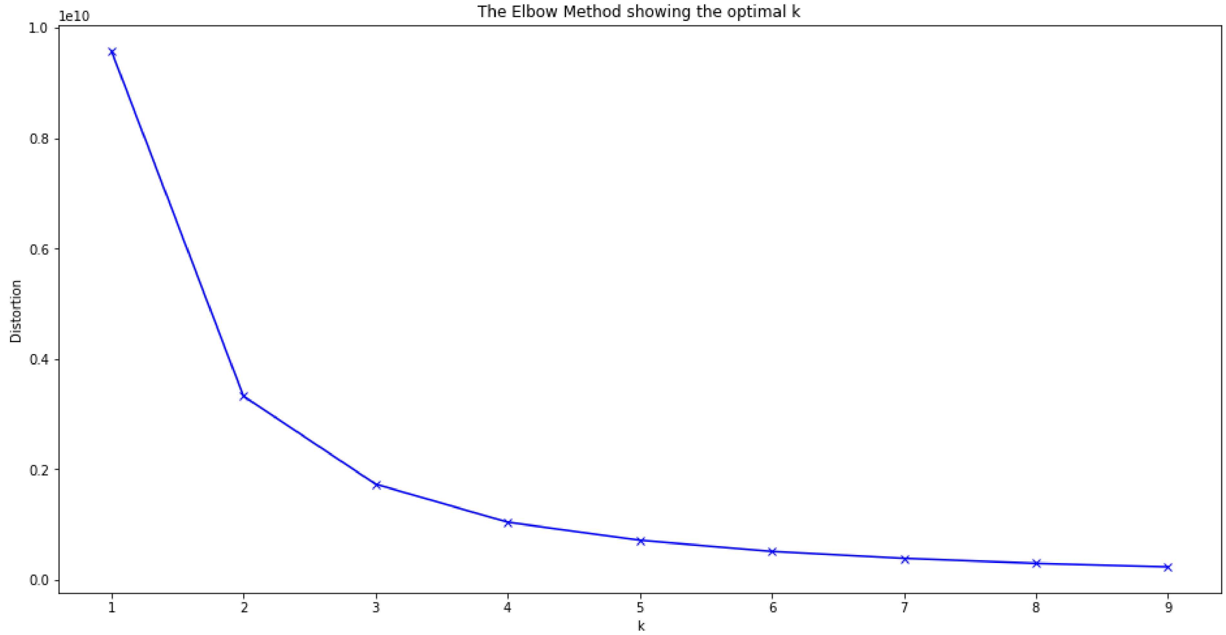
In [217]:
```
df.drop('ORDERDATE', axis=1, inplace=True) #Dropping the Orderdate as Month is al
```

In [218]:
```
df.dtypes #All the datatypes are converted into numeric
```
...

Plotting the Elbow Plot to determine the number of clusters.

In [219]:
```
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 Disto
```

In [220]:
```
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.

In [221]:
```
X_train = df.values #Returns a numpy array.
```

In [222]:
```
X_train.shape
```

Out[222]:
```
(2823, 19)
```

In [223]:
```
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)
```

In [225]:
```
unique,counts = np.unique(predictions,return_counts=True)
```

In [226]:
```
counts = counts.reshape(1,3)
```

In [227]:
```
counts_df = pd.DataFrame(counts,columns=['Cluster1','Cluster2','Cluster3'])
```

In [228]:
```
counts_df.head()
```

Out[228]:

	Cluster1	Cluster2	Cluster3
0	1083	1367	373

Visualization

In [229]:
```
pca = PCA(n_components=2) #Converting all the features into 2 columns to make it
```

In [230]:
```
reduced_X = pd.DataFrame(pca.fit_transform(X_train),columns=['PCA1','PCA2']) #Cre
```

In [231]:
```
reduced_X.head()
```

Out[231]:

	PCA1	PCA2
0	-682.488323	-42.819535
1	-787.665502	-41.694991
2	330.732170	-26.481208
3	193.040232	-26.285766
4	1651.532874	-6.891196

In [232]:
```
#Plotting the normal Scatter Plot
plt.figure(figsize=(14,10))
plt.scatter(reduced_X['PCA1'],reduced_X['PCA2'])
```

...

In [233]:
```
model.cluster_centers_ #Finding the centriods. (3 Centriods in total. Each Array
```

...

In [234]:
```
reduced_centers = pca.transform(model.cluster_centers_) #Transforming the centroi
```

In [235]:
```
reduced_centers
```

Out[235]:
```
array([[ 5.84994044e+02, -4.36786931e+00],
       [-1.43005891e+03,  2.60041009e+00],
       [ 3.54247180e+03,  3.15185487e+00]])
```

In [236]:
```
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=
```

...

In [237]:
```
reduced_X['Clusters'] = predictions #Adding the Clusters to the reduced dataframe
```

In [238]:
```
reduced_X.head()
```

...

In [239]:
```
#Plotting the clusters
plt.figure(figsize=(14,10))
#                      taking the cluster number and first column           taking
plt.scatter(reduced_X[reduced_X['Clusters'] == 0].loc[:,'PCA1'],reduced_X[reduced
plt.scatter(reduced_X[reduced_X['Clusters'] == 1].loc[:,'PCA1'],reduced_X[reduced
plt.scatter(reduced_X[reduced_X['Clusters'] == 2].loc[:,'PCA1'],reduced_X[reduced


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

Out[239]: <matplotlib.collections.PathCollection at 0x218dce9e1f0>