Practical No. 6

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

Code :

In [82]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [83]:
```
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

In [84]:
```
df = pd.read_csv("sales_data_sample.csv", encoding ="Latin-1")
```

In [85]:
```
df.head()
```

Out[85]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	OR
0	10107	30	95.70	2	2871.00	
1	10121	34	81.35	5	2765.90	
2	10134	41	94.74	2	3884.34	
3	10145	45	83.26	6	3746.70	
4	10159	49	100.00	14	5205.27	1(

5 rows × 25 columns

In [86]:
```
df.shape
```

Out[86]:
```
(2823, 25)
```

In [87]:
```
df.describe()
```

Out[87]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	S
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.0
mean	10258.725115	35.092809	83.658544	6.466171	3553.8
std	92.085478	9.741443	20.174277	4.225841	1841.8
min	10100.000000	6.000000	26.880000	1.000000	482.1
25%	10180.000000	27.000000	68.860000	3.000000	2203.4
50%	10262.000000	35.000000	95.700000	6.000000	3184.8
75%	10333.500000	43.000000	100.000000	9.000000	4508.0
max	10425.000000	97.000000	100.000000	18.000000	14082.8

In [88]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2823 entries, 0 to 2822
Data columns (total 25 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   ORDERNUMBER       2823 non-null   int64  
 1   QUANTITYORDERED   2823 non-null   int64  
 2   PRICEEACH         2823 non-null   float64
 3   ORDERLINENUMBER   2823 non-null   int64  
 4   SALES             2823 non-null   float64
 5   ORDERDATE         2823 non-null   object 
 6   STATUS            2823 non-null   object 
 7   QTR_ID            2823 non-null   int64  
 8   MONTH_ID          2823 non-null   int64  
 9   YEAR_ID           2823 non-null   int64  
 10  PRODUCTLINE       2823 non-null   object 
 11  MSRP              2823 non-null   int64  
 12  PRODUCTCODE       2823 non-null   object 
 13  CUSTOMERNAME      2823 non-null   object 
 14  PHONE             2823 non-null   object 
 15  ADDRESSLINE1      2823 non-null   object 
 16  ADDRESSLINE2      302 non-null    object 
 17  CITY              2823 non-null   object 
 18  STATE             1337 non-null   object 
 19  POSTALCODE        2747 non-null   object 
 20  COUNTRY           2823 non-null   object 
 21  TERRITORY         1749 non-null   object 
 22  CONTACTLASTNAME   2823 non-null   object 
 23  CONTACTFIRSTNAME  2823 non-null   object 
 24  DEALSIZE          2823 non-null   object 
dtypes: float64(2), int64(7), object(16)
memory usage: 551.5+ KB
```

In [89]:
```
df.dtypes
```

Out[89]:
```
ORDERNUMBER          int64
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
ORDERDATE           object
STATUS              object
QTR_ID               int64
MONTH_ID             int64
YEAR_ID              int64
PRODUCTLINE         object
MSRP                 int64
PRODUCTCODE         object
CUSTOMERNAME        object
PHONE               object
ADDRESSLINE1        object
ADDRESSLINE2        object
CITY                object
STATE               object
POSTALCODE          object
COUNTRY             object
TERRITORY           object
CONTACTLASTNAME     object
CONTACTFIRSTNAME    object
DEALSIZE            object
dtype: object
```

In [90]:
```
df.isnull().sum()
```

Out[90]:
```
ORDERNUMBER          0
QUANTITYORDERED      0
PRICEEACH            0
ORDERLINENUMBER      0
SALES                0
ORDERDATE            0
STATUS               0
QTR_ID               0
MONTH_ID             0
YEAR_ID              0
PRODUCTLINE          0
MSRP                 0
PRODUCTCODE          0
CUSTOMERNAME         0
PHONE                0
ADDRESSLINE1         0
ADDRESSLINE2      2521
CITY                 0
STATE             1486
POSTALCODE          76
COUNTRY              0
TERRITORY         1074
CONTACTLASTNAME      0
CONTACTFIRSTNAME     0
DEALSIZE             0
dtype: int64
```

In [91]:
```
df_drop = [
    'ORDERNUMBER', 'ORDERDATE', 'STATUS', 'PRODUCTLINE', 'PRODUCTCODE',
    'CUSTOMERNAME', 'PHONE', 'ADDRESSLINE1', 'ADDRESSLINE2',
    'CITY', 'STATE', 'POSTALCODE', 'COUNTRY', 'TERRITORY',
```

```
    'CONTACTLASTNAME', 'CONTACTFIRSTNAME', 'DEALSIZE'
]
```

In [92]:
```
df_cleaned = df.drop(df_drop,axis=1)
```

In [93]:
```
df.isnull().sum()
```

Out[93]:
```
ORDERNUMBER           0
QUANTITYORDERED       0
PRICEEACH             0
ORDERLINENUMBER       0
SALES                 0
ORDERDATE             0
STATUS                0
QTR_ID                0
MONTH_ID              0
YEAR_ID               0
PRODUCTLINE           0
MSRP                  0
PRODUCTCODE           0
CUSTOMERNAME          0
PHONE                 0
ADDRESSLINE1          0
ADDRESSLINE2       2521
CITY                  0
STATE              1486
POSTALCODE           76
COUNTRY               0
TERRITORY          1074
CONTACTLASTNAME       0
CONTACTFIRSTNAME      0
DEALSIZE              0
dtype: int64
```

In [94]:
```
df.dtypes
```

Out[94]:
```
ORDERNUMBER          int64
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
ORDERDATE           object
STATUS              object
QTR_ID               int64
MONTH_ID             int64
YEAR_ID              int64
PRODUCTLINE         object
MSRP                 int64
PRODUCTCODE         object
CUSTOMERNAME        object
PHONE               object
ADDRESSLINE1        object
ADDRESSLINE2        object
CITY                object
STATE               object
POSTALCODE          object
COUNTRY             object
TERRITORY           object
CONTACTLASTNAME     object
CONTACTFIRSTNAME    object
DEALSIZE            object
dtype: object
```

In [95]:
```
df['COUNTRY'].unique()
```

Out[95]:
```
array(['USA', 'France', 'Norway', 'Australia', 'Finland', 'Austria', 'UK',
       'Spain', 'Sweden', 'Singapore', 'Canada', 'Japan', 'Italy',
       'Denmark', 'Belgium', 'Philippines', 'Germany', 'Switzerland',
       'Ireland'], dtype=object)
```

In [96]:
```
df['PRODUCTLINE'].unique()
```

Out[96]:
```
array(['Motorcycles', 'Classic Cars', 'Trucks and Buses', 'Vintage Cars',
       'Planes', 'Ships', 'Trains'], dtype=object)
```

In [97]:
```
df['DEALSIZE'].unique()
```

Out[97]:
```
array(['Small', 'Medium', 'Large'], dtype=object)
```

In [98]:
```
productline = pd.get_dummies (df['PRODUCTLINE'])
Dealsize = pd.get_dummies (df['DEALSIZE'])
```

In [99]:
```
df = pd.concat([df, productline, Dealsize], axis=1)
```

In [100…
```
df = pd.concat([df, productline, Dealsize], axis=1)
```

In [101…
```
df_drop = ['COUNTRY', 'PRODUCTLINE', 'DEALSIZE']
df = df.drop(df_drop, axis =1 )
```

In [102…
```
df [ 'PRODUCTCODE'] = pd. Categorical (df [ 'PRODUCTCODE']).codes
```

In [103…
```
df.drop('ORDERDATE', axis = 1, inplace=True)
```

In [104…
```
df.dtypes
```

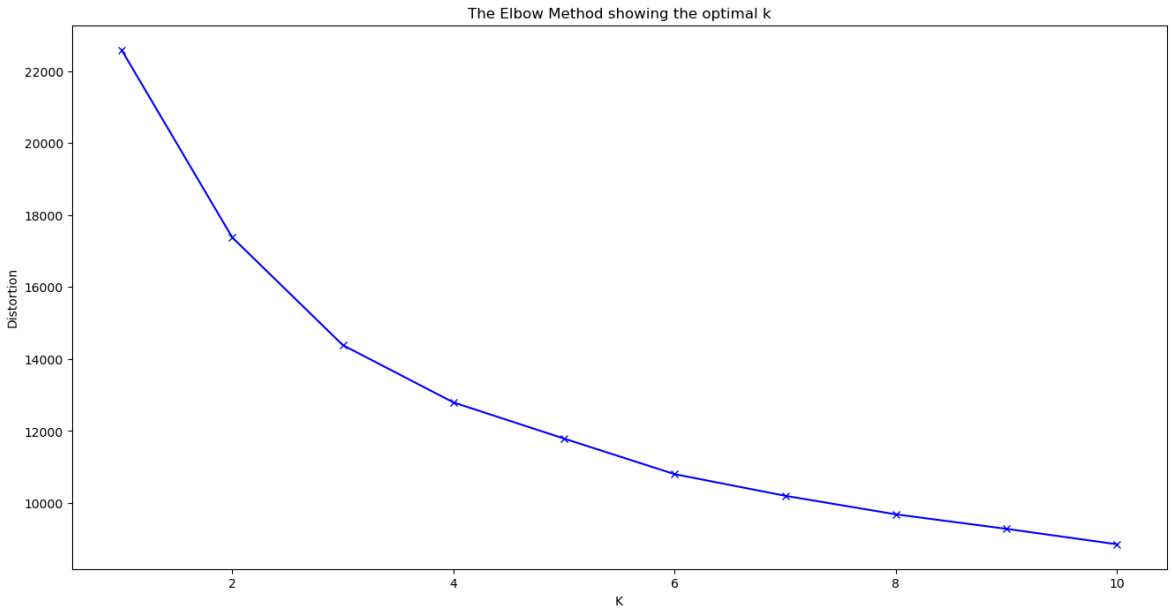
Out[104…
```
ORDERNUMBER           int64
QUANTITYORDERED       int64
PRICEEACH           float64
ORDERLINENUMBER       int64
SALES               float64
STATUS               object
QTR_ID                int64
MONTH_ID              int64
YEAR_ID               int64
MSRP                  int64
PRODUCTCODE            int8
CUSTOMERNAME         object
PHONE                object
ADDRESSLINE1         object
ADDRESSLINE2         object
CITY                 object
STATE                object
POSTALCODE           object
TERRITORY            object
CONTACTLASTNAME      object
CONTACTFIRSTNAME     object
Classic Cars           bool
Motorcycles            bool
Planes                 bool
Ships                  bool
Trains                 bool
Trucks and Buses       bool
Vintage Cars           bool
Large                  bool
Medium                 bool
Small                  bool
Classic Cars           bool
Motorcycles            bool
Planes                 bool
Ships                  bool
Trains                 bool
Trucks and Buses       bool
Vintage Cars           bool
Large                  bool
Medium                 bool
Small                  bool
dtype: object
```

In [105…
```
# Scale numeric data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df_cleaned)
```

In [106…
```
# Elbow method
distortions = []
K = range(1, 11)

for k in K:
    kmeanModel = KMeans(n_clusters=k, random_state=42)
    kmeanModel.fit(scaled_data)    # <<--- use scaled numeric data
    distortions.append(kmeanModel.inertia_)
```

In [107…
```
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()
```



In [108…
```
x_train = df.values
```

In [110…
```
x_train.shap
```

Out[110…
```
(2823, 41)
```

In [112…
```
x_train = df_cleaned.select_dtypes(include=['int64', 'float64'])
```

In [114…
```
scaler = StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
```

In [115…
```
# Fit KMeans with 3 clusters (example)
model = KMeans(n_clusters=3, random_state=2)
model.fit(x_train_scaled)
```

Out[115…
```
KMeans
KMeans(n_clusters=3, random_state=2)
```

In [116…
```
predictions = model.predict(x_train_scaled)
df_cleaned["Cluster"] = predictions
```

In [117…
```
print(df_cleaned.head())
```

```
   QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER    SALES  QTR_ID  MONTH_ID  \
0               30      95.70                2  2871.00       1         2
1               34      81.35                5  2765.90       2         5
2               41      94.74                2  3884.34       3         7
3               45      83.26                6  3746.70       3         8
4               49     100.00               14  5205.27       4        10

   YEAR_ID  MSRP  Cluster
0     2003    95        0
1     2003    95        1
2     2003    95        2
3     2003    95        2
4     2003    95        2
```

In [119…

```
pca = PCA(n_components=2)
reduced_X = pd.DataFrame(pca.fit_transform(x_train_scaled),
                         columns=['PCA1', 'PCA2'])

# Add cluster labels
reduced_X['Cluster'] = predictions

# Plot clusters
plt.figure(figsize=(8,6))
plt.scatter(reduced_X['PCA1'], reduced_X['PCA2'],
            c=reduced_X['Cluster'], cmap='rainbow')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.title('Customer Segmentation using KMeans (PCA Visualization)')
plt.show()
```