# 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()
```

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | ORDERDATE | STATUS |
|---|---|---|---|---|---|---|---|
| **0** | 10107 | 30 | 95.70 | 2 | 2871.00 | 2/24/2003 0:00 | Shipped |
| **1** | 10121 | 34 | 81.35 | 5 | 2765.90 | 5/7/2003 0:00 | Shipped |
| **2** | 10134 | 41 | 94.74 | 2 | 3884.34 | 7/1/2003 0:00 | Shipped |
| **3** | 10145 | 45 | 83.26 | 6 | 3746.70 | 8/25/2003 0:00 | Shipped |
| **4** | 10159 | 49 | 100.00 | 14 | 5205.27 | 10/10/2003 0:00 | Shipped |

5 rows × 25 columns

```
df.shape
```

```
(2823, 25)
```

```
df.describe()
```

| | ORDERNUMBER | QUANTITYORDERED | PRICEEACH | ORDERLINENUMBER | SALES | QT |
|---|---|---|---|---|---|---|
| **count** | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.000000 | 2823.00 |
| **mean** | 10258.725115 | 35.092809 | 83.658544 | 6.466171 | 3553.889072 | 2.71 |
| **std** | 92.085478 | 9.741443 | 20.174277 | 4.225841 | 1841.865106 | 1.20 |
| **min** | 10100.000000 | 6.000000 | 26.880000 | 1.000000 | 482.130000 | 1.00 |
| **25%** | 10180.000000 | 27.000000 | 68.860000 | 3.000000 | 2203.430000 | 2.00 |
| **50%** | 10262.000000 | 35.000000 | 95.700000 | 6.000000 | 3184.800000 | 3.00 |
| **75%** | 10333.500000 | 43.000000 | 100.000000 | 9.000000 | 4508.000000 | 4.00 |
| **max** | 10425.000000 | 97.000000 | 100.000000 | 18.000000 | 14082.800000 | 4.00 |

```
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)
```

```
df.isnull().sum()
```

```
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
```

```
df.dtypes
```

```
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
```

```
df_drop  = ['ADDRESSLINE1', 'ADDRESSLINE2', 'STATUS','POSTALCODE', 'CITY', 'TERRITORY', 'PHON
df = df.drop(df_drop, axis=1) #Dropping the categorical uneccessary columns along with column
```

```
df.isnull().sum()
```

```
QUANTITYORDERED    0
PRICEEACH          0
ORDERLINENUMBER    0
SALES              0
ORDERDATE          0
QTR_ID             0
MONTH_ID           0
YEAR_ID            0
PRODUCTLINE        0
MSRP               0
PRODUCTCODE        0
COUNTRY            0
DEALSIZE           0
dtype: int64
```

```
df.dtypes
```

```
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
ORDERDATE           object
QTR_ID               int64
MONTH_ID             int64
YEAR_ID              int64
PRODUCTLINE         object
MSRP                 int64
PRODUCTCODE         object
COUNTRY             object
DEALSIZE            object
dtype: object
```

```
# Checking the categorical columns.
```

```
df['COUNTRY'].unique()
```

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

```
df['PRODUCTLINE'].unique()
```

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

```
df['DEALSIZE'].unique()
```

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

```
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 co
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 includ
```

```
df.dtypes #All the datatypes are converted into numeric
```

```
QUANTITYORDERED      int64
PRICEEACH          float64
ORDERLINENUMBER      int64
SALES              float64
QTR_ID               int64
MONTH_ID             int64
YEAR_ID              int64
MSRP                 int64
PRODUCTCODE           int8
Classic Cars         uint8
Motorcycles          uint8
Planes               uint8
Ships                uint8
Trains               uint8
Trucks and Buses     uint8
Vintage Cars         uint8
Large                uint8
Medium               uint8
Small                uint8
dtype: object
```

## Plotting the Elbow Plot to determine the number of clusters.

[ ] ↳ *2 cells hidden*

## As the number of k increases Inertia decreases.

Observations: A Elbow can be observed at 3 and after that the curve decreases gradually.

[ ] ↳ *7 cells hidden*

## Visualization

[ ] ↳ *12 cells hidden*