Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method. Dataset link : https://www.kaggle.com/datasets/kyanyoga/sample-sales-data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

df = pd.read_csv('sales_data_sample.csv')
df
```

```
      ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER
SALES  \
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
...           ...              ...        ...              ...       ..
.
2818        10350               20     100.00               15
2244.40
2819        10373               29     100.00                1
3978.51
2820        10386               43     100.00                4
5417.57
2821        10397               34      62.24                1
2116.16
2822        10414               47      65.52                9
3079.44

            ORDERDATE     STATUS  QTR_ID  MONTH_ID  YEAR_ID  ...  \
0      2/24/2003 0:00    Shipped       1         2     2003  ...
1       5/7/2003 0:00    Shipped       2         5     2003  ...
2       7/1/2003 0:00    Shipped       3         7     2003  ...
3      8/25/2003 0:00    Shipped       3         8     2003  ...
4     10/10/2003 0:00    Shipped       4        10     2003  ...
...               ...        ...     ...       ...      ...  ...
2818   12/2/2004 0:00    Shipped       4        12     2004  ...
2819   1/31/2005 0:00    Shipped       1         1     2005  ...
2820    3/1/2005 0:00   Resolved       1         3     2005  ...
2821   3/28/2005 0:00    Shipped       1         3     2005  ...
2822    5/6/2005 0:00    On Hold       2         5     2005  ...
```

	ADDRESSLINE1	ADDRESSLINE2	CITY	STATE
0	897 Long Airport Avenue	NaN	NYC	NY
1	59 rue de l'Abbaye	NaN	Reims	NaN
2	27 rue du Colonel Pierre Avia	NaN	Paris	NaN
3	78934 Hillside Dr.	NaN	Pasadena	CA
4	7734 Strong St.	NaN	San Francisco	CA
...	...	...	...	...
2818	C/ Moralzarzal, 86	NaN	Madrid	NaN
2819	Torikatu 38	NaN	Oulu	NaN
2820	C/ Moralzarzal, 86	NaN	Madrid	NaN
2821	1 rue Alsace-Lorraine	NaN	Toulouse	NaN
2822	8616 Spinnaker Dr.	NaN	Boston	MA

	POSTALCODE	COUNTRY	TERRITORY	CONTACTLASTNAME	CONTACTFIRSTNAME	DEALSIZE
0	10022	USA	NaN	Yu	Kwai	Small
1	51100	France	EMEA	Henriot	Paul	Small
2	75508	France	EMEA	Da Cunha	Daniel	Medium
3	90003	USA	NaN	Young	Julie	Medium
4	NaN	USA	NaN	Brown	Julie	Medium
...	...	...	...	...	...	...
2818	28034	Spain	EMEA	Freyre	Diego	Small
2819	90110	Finland	EMEA	Koskitalo	Pirkko	Medium
2820	28034	Spain	EMEA	Freyre	Diego	Medium
2821	31000	France	EMEA	Roulet	Annette	Small
2822	51003	USA	NaN	Yoshido	Juri	Medium

```
[2823 rows x 25 columns]
```

```
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
```

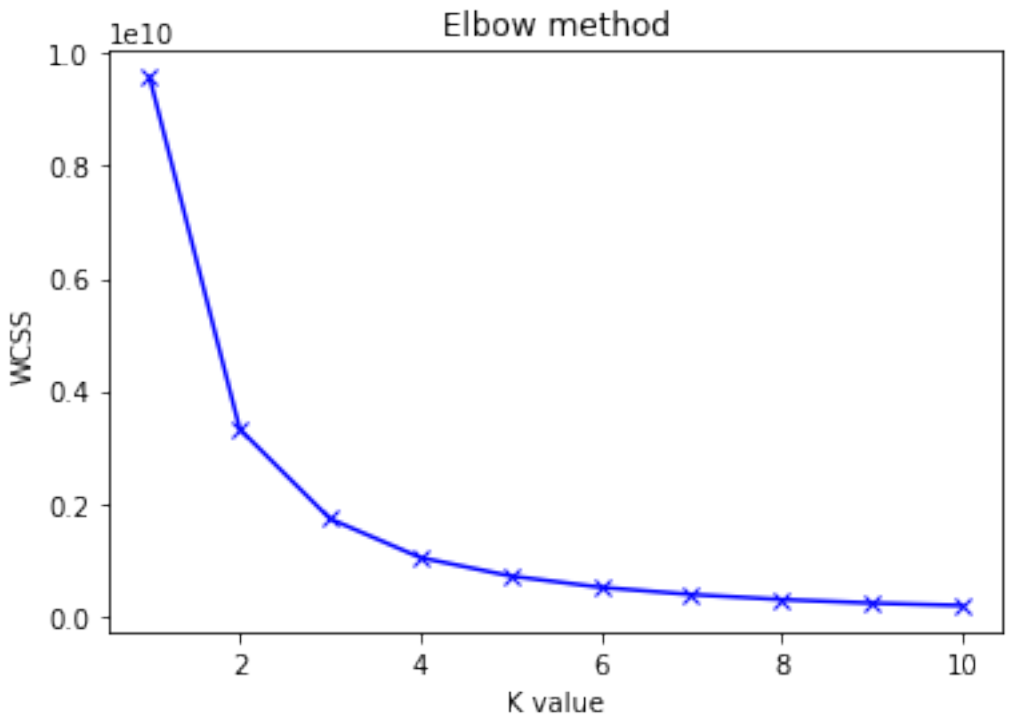
```
X = df.iloc[:, [3,4]].values
```

```
 wcss = []   #within cluster sum of square

for i in range(1,11):
    #init argument is the method for initializing the centroid
    kmeans = KMeans(n_clusters=i, init="k-means++", random_state=42)
    kmeans.fit(X)
    #we calculate wcss value for each k value
    wcss.append(kmeans.inertia_)

ks = [1,2,3,4,5,6,7,8,9,10]
plt.plot(ks, wcss, 'bx-')
plt.title("Elbow method")
plt.xlabel("K value")
plt.ylabel("WCSS")
```

```
Text(0, 0.5, 'WCSS')
```




```
df.describe()
```

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

	SALES	QTR_ID	MONTH_ID	YEAR_ID	MSRP
count	2823.000000	2823.000000	2823.000000	2823.00000	2823.000000
mean	3553.889072	2.717676	7.092455	2003.81509	100.715551
std	1841.865106	1.203878	3.656633	0.69967	40.187912
min	482.130000	1.000000	1.000000	2003.00000	33.000000
25%	2203.430000	2.000000	4.000000	2003.00000	68.000000
50%	3184.800000	3.000000	8.000000	2004.00000	99.000000
75%	4508.000000	4.000000	11.000000	2004.00000	124.000000

```
max    14082.800000     4.000000    12.000000  2005.00000   214.000000
```

```
# mean is far from std this indicates high variance
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
scaled = ss.fit_transform(X)

wcss =[]

for i in range(1,11):
    clustering = KMeans(n_clusters=i, init="k-means++",
random_state=42)
    clustering.fit(scaled)
    wcss.append(clustering.inertia_)

ks = [1,2,3,4,5,6,7,8,9,10]
plt.plot(ks, wcss, 'bx-')
plt.title("Elbow method")
plt.xlabel("K value")
plt.ylabel("WCSS")
```

```
Text(0, 0.5, 'WCSS')
```