Assignment NO 5 Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset.

Determine the number of clusters using the elbow method.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df =pd.read_csv("sales_data_sample.csv",encoding='latin')
df
```

Out[2]:

	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

2823 rows × 25 columns

In [3]:

```
df.dtypes
```

Out[3]:

```
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 [4]:

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

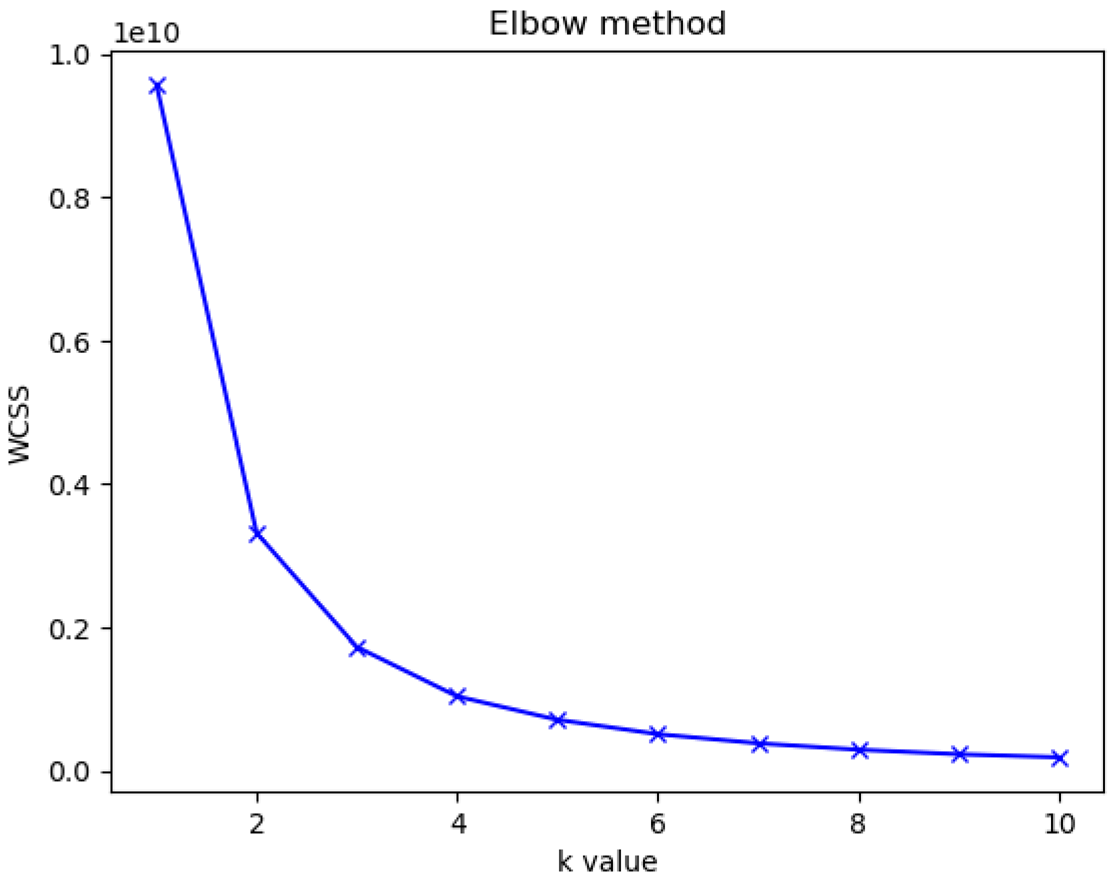
In [5]:

```
WCSS =[]  #within cluster sum of square
for i in range(1,11):
    #init argument is the method for initializing the centriod
    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")
```

Out[5]:

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



In [6]:

```
df.describe()
```

Out[6]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	
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 [7]:

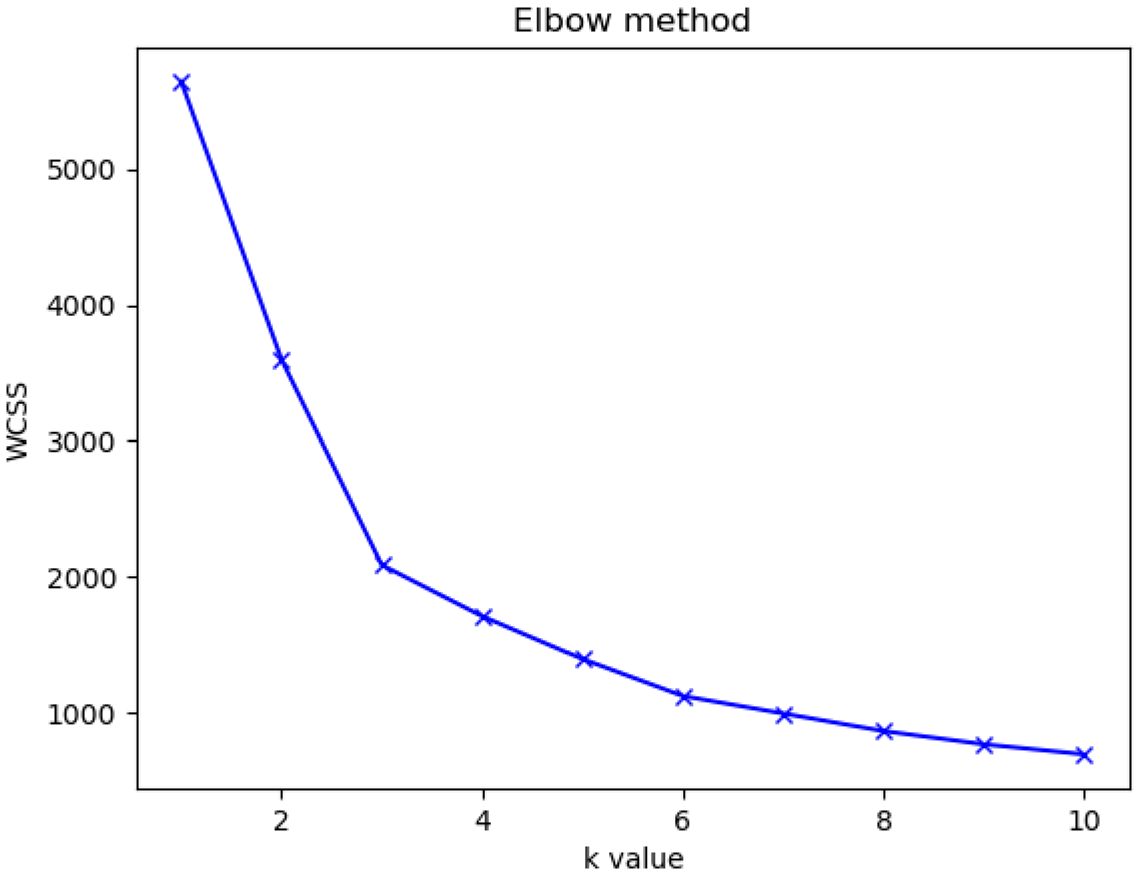
```
# mean is far from std this indicates high variance
from sklearn.preprocessing import StandardScaler
ss =StandardScaler()
scaled =ss.fit_transform(X)
```

In [8]:

```
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")
```

```
Out[8]:  Text(0, 0.5, 'WCSS')
```



```
In [ ]:
```