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

```
df = pd.read_csv("sales_data_sample.csv", sep=",", encoding='Latin-1')
```

```
df
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDERDATE	
0	10107	30	95.70	2	2871.00	2/24/2003 0:00	S
1	10121	34	81.35	5	2765.90	5/7/2003 0:00	S
2	10134	41	94.74	2	3884.34	7/1/2003 0:00	S
3	10145	45	83.26	6	3746.70	8/25/2003 0:00	S
4	10159	49	100.00	14	5205.27	10/10/2003 0:00	S
...	...	...	...	...	...	...	
2818	10350	20	100.00	15	2244.40	12/2/2004 0:00	S
2819	10373	29	100.00	1	3978.51	1/31/2005 0:00	S
2820	10386	43	100.00	4	5417.57	3/1/2005 0:00	R
2821	10397	34	62.24	1	2116.16	3/28/2005 0:00	S
2822	10414	47	65.52	9	3079.44	5/6/2005 0:00	(

2823 rows × 25 columns

```
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
```

```
df.describe()
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.000000	2823
mean	10258.725115	35.092809	83.658544	6.466171	3553.889072	2
std	92.085478	9.741443	20.174277	4.225841	1841.865106	1
min	10100.000000	6.000000	26.880000	1.000000	482.130000	1
25%	10180.000000	27.000000	68.860000	3.000000	2203.430000	2
50%	10262.000000	35.000000	95.700000	6.000000	3184.800000	3
75%	10333.500000	43.000000	100.000000	9.000000	4508.000000	4
max	10425.000000	97.000000	100.000000	18.000000	14082.800000	4

```
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
```

```
# Removing Null Value

df.dropna(subset=['ADDRESSLINE2'], inplace=True)
df.dropna(subset=['STATE'], inplace=True)
df.dropna(subset=['TERRITORY'], inplace=True)
```

```
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        0
CITY                0
STATE               0
POSTALCODE          0
COUNTRY             0
TERRITORY           0
CONTACTLASTNAME     0
CONTACTFIRSTNAME    0
DEALSIZE            0
dtype: int64
```

```
df.describe()
```

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	QTI
count	147.000000	147.000000	147.000000	147.000000	147.000000	147.00
mean	10268.204082	33.986395	84.138639	6.673469	3446.003537	2.67
std	106.742036	9.740458	19.897255	3.830596	1717.988835	1.17
min	10120.000000	15.000000	26.880000	1.000000	652.350000	1.00
25%	10148.000000	26.000000	66.390000	3.000000	2249.005000	2.00
50%	10270.000000	33.000000	98.050000	6.000000	3160.740000	3.00
75%	10361.000000	41.000000	100.000000	10.000000	4410.060000	4.00
max	10420.000000	66.000000	100.000000	15.000000	9774.030000	4.00

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

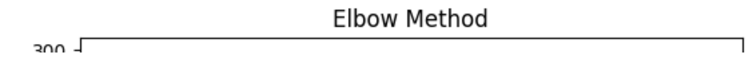
WCSS is the sum of the squared distance between each point and the centroid in a cluster.

```
# Initialize an empty list to store the within-cluster sum of squares (WCSS)
wcss = []
```

```
# Determine the WCSS for a range of cluster numbers (e.g., 1 to 10)
for i in range(1,11):
  kmeans = KMeans(n_clusters= i, random_state=2)
  kmeans.fit(X)
  wcss.append(kmeans.inertia_)
```

```
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fro
      warnings.warn(
```

```
# Plot the WCSS values to identify the elbow point
plt.plot(range(1,11), wcss, 'bx-')
plt.title("Elbow Method")
plt.xlabel("Number of Cluster(Kvalue)")
plt.ylabel("WCSS")
plt.show()
```



```
scale = StandardScaler()
scaled_data = scale.fit_transform(X)

# Based on the plot, visually determine the optimal number of clusters, where the WCSS starts to level off (the "elbow" point)

# Implement K-Means clustering with the optimal number of clusters
# Let's assume you found the optimal number of clusters to be 'k'
k = 3  # Replace with the number you determined from the elbow method

kmeans = KMeans(n_clusters=3,random_state = 0)
cluster_lables = kmeans.fit_predict(scaled_data)
```

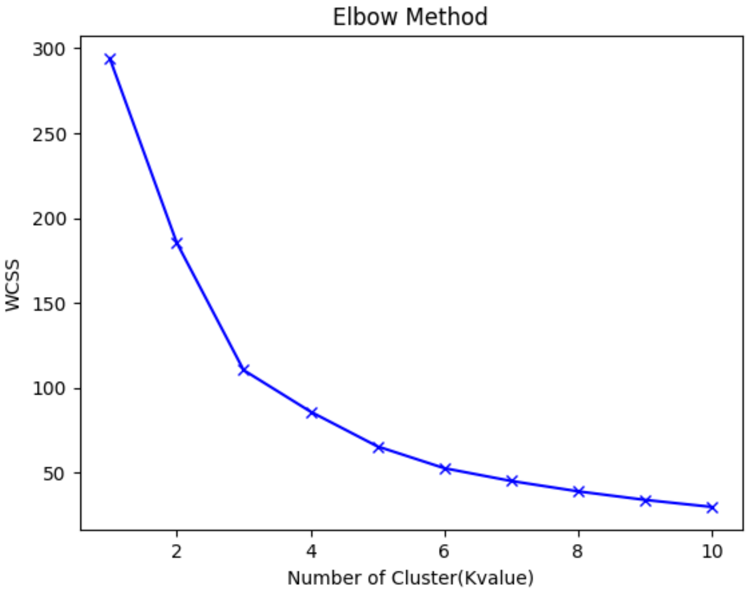
```
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
```

```
wcss = []

for i in range(1,11):
  kmeans = KMeans(n_clusters= i, random_state=2)
  kmeans.fit(scaled_data)
  wcss.append(kmeans.inertia_)
```

```
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
    /usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
      warnings.warn(
```

```
# Plot the WCSS values to identify the elbow point
plt.plot(range(1,11), wcss,'bx-')
plt.title("Elbow Method")
plt.xlabel("Number of Cluster(Kvalue)")
plt.ylabel("WCSS")
plt.show()
```



Conclusion

For K-means Clustering the optimal number of clusters are 3