```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('sales_data_sample.csv', encoding="unicode_escape")  #imp
df.head()
```

```
df.isna().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
```

Saved successfully! ✕

```
to_drop = ['PHONE','ADDRESSLINE1','ADDRESSLINE2','CITY','STATE','POSTALCODE',
             'TERRITORY','CONTACTLASTNAME', 'CONTACTFIRSTNAME','CUSTOMERNAME','ORDERNUMBER']
df = df.drop(to_drop, axis=1)
df.isna().sum()
```

```
df = df[['QUANTITYORDERED', 'ORDERLINENUMBER']]
```

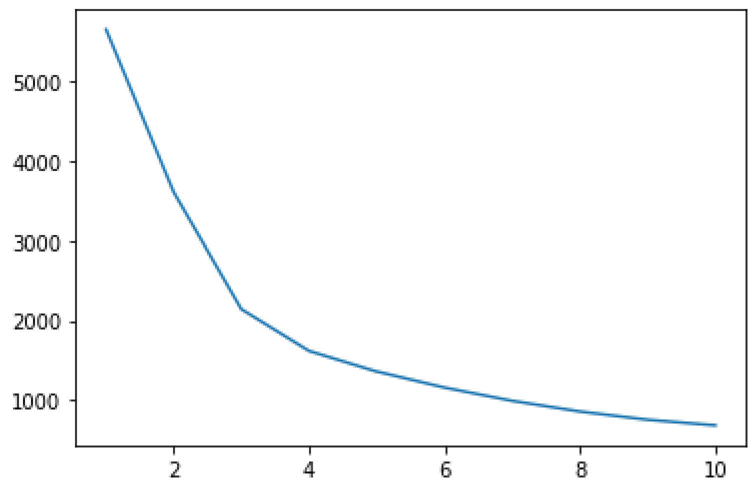
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
df_sc = sc.fit_transform(df)
```

```
from sklearn.cluster import KMeans
wcss = []

for i in range(1,11):
  model = KMeans(n_clusters=i, random_state=30)
  model.fit(df_sc)
  wcss.append(model.inertia_)

k = [1,2,3,4,5,6,7,8,9,10]
sns.lineplot(x=k, y=wcss)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f677b368790>
```



Inertia measures how well a dataset was clustered by K-Means. It is calculated by measuring the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster. A good model is one with low inertia AND a low number of clusters (K).

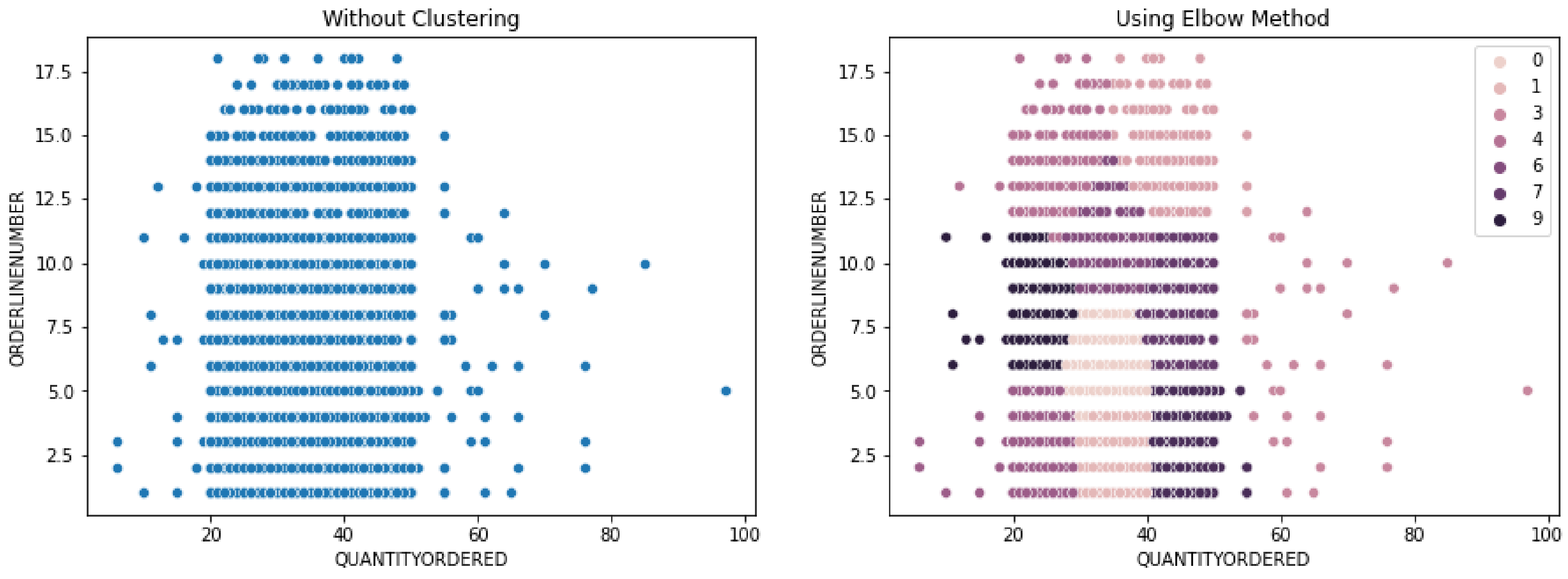
WCSS stands for Within Cluster Sum of Squares.

WCSS is the sum of squares of the distances of each data point in all clusters to their respective centroids.

The elbow method runs k-means clustering on the dataset for a range of values for k (say from 1-10) and then for each value of k computes an average score for all clusters. By default, the distortion score is computed, the sum of square distances from each point to its assigned center.

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize= (15,5))
sns.scatterplot(ax=axes[0], data=df, x='QUANTITYORDERED', y='ORDERLINENUMBER').set_title('Without Clustering')
sns.scatterplot(ax=axes[1] ,data=df, x='QUANTITYORDERED', y='ORDERLINENUMBER', hue=model.labels_).set_title('Using Elbow Method')
```

```
Text(0.5, 1.0, 'Using Elbow Method')
```


Without Clustering
Using Elbow Method
ORDERLINENUMBER
QUANTITYORDERED