Madhur Jaripatke

Roll No. 52

BE A Computer

RMDSSOE, Warje, Pune

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

Importing Libraries

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import warnings
from sklearn.preprocessing import StandardScaler
warnings.filterwarnings('ignore')
```

Loading The Dataset

In [2]:

```
df = pd.read_csv('./Datasets/sales_data_sample.csv', encoding='ISO-8859-1')
df
```

Out[2]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	
0	10107	30	95.70	2	2
1	10121	34	81.35	5	2
2	10134	41	94.74	2	3
3	10145	45	83.26	6	3
4	10159	49	100.00	14	5
...	...	...	...	...	
2818	10350	20	100.00	15	2
2819	10373	29	100.00	1	3
2820	10386	43	100.00	4	5
2821	10397	34	62.24	1	2
2822	10414	47	65.52	9	3

2823 rows × 25 columns

Exploratory Data Analysis (EDA)

In [3]:

```
df.describe()
```

Out[3]:

	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

In [4]:

```
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
```

Data Preprocessing

In [5]:

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

In [6]:

```
scaler = StandardScaler()
scaled_values = scaler.fit_transform(df.values)
```

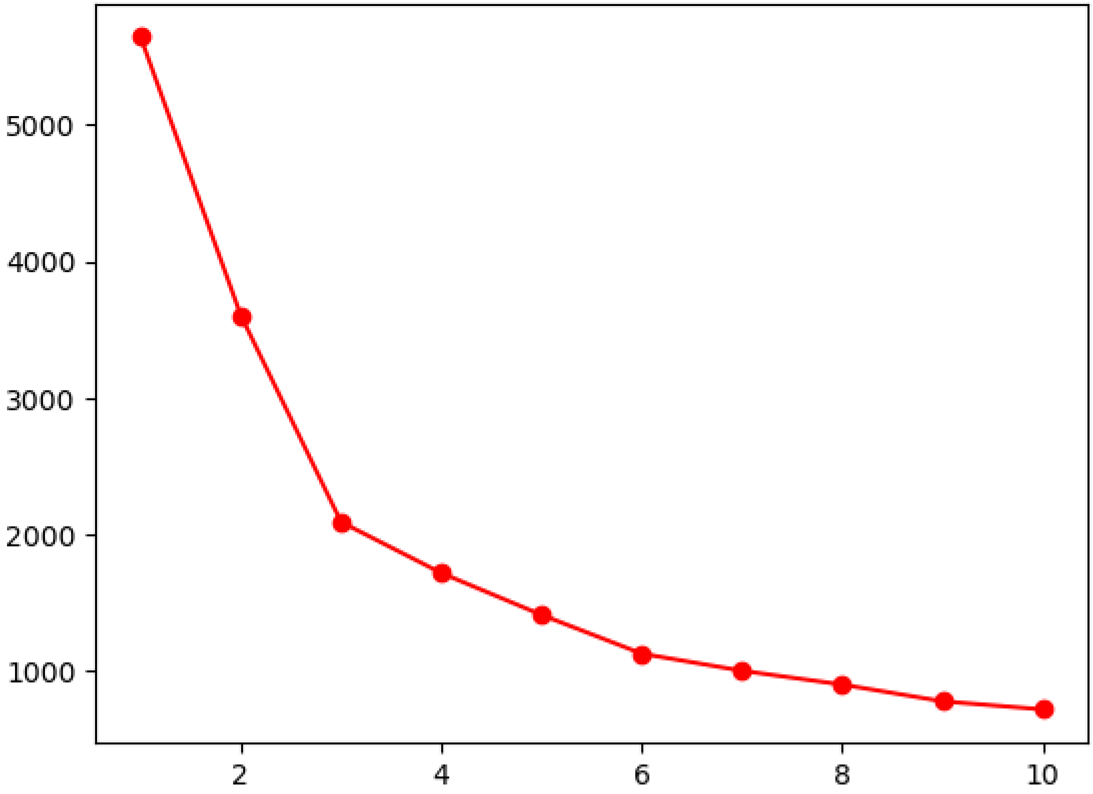
Elbow Method to Determine Optimal Number of Clusters

In [7]:

```
wcss = []
for i in range(1, 11):
    model = KMeans(n_clusters=i, init='k-means++')
    model.fit_predict(scaled_values)
    wcss.append(model.inertia_)
```

In [8]:

```
plt.plot(range(1, 11), wcss, 'ro-')
plt.show()
```



K-Means Clustering

In [9]:

```
model = KMeans(n_clusters=7, init='k-means++')
clusters = model.fit_predict(scaled_values)
df['Cluster'] = clusters
```

In [10]:

```
df
```

Out[10]:

	ORDERLINENUMBER	SALES	Cluster
0	2	2871.00	3
1	5	2765.90	1
2	2	3884.34	6
3	6	3746.70	1
4	14	5205.27	4
...	...	...	...
2818	15	2244.40	4
2819	1	3978.51	6
2820	4	5417.57	6
2821	1	2116.16	3
2822	9	3079.44	5

2823 rows × 3 columns

In [11]:

```
model.inertia_
```

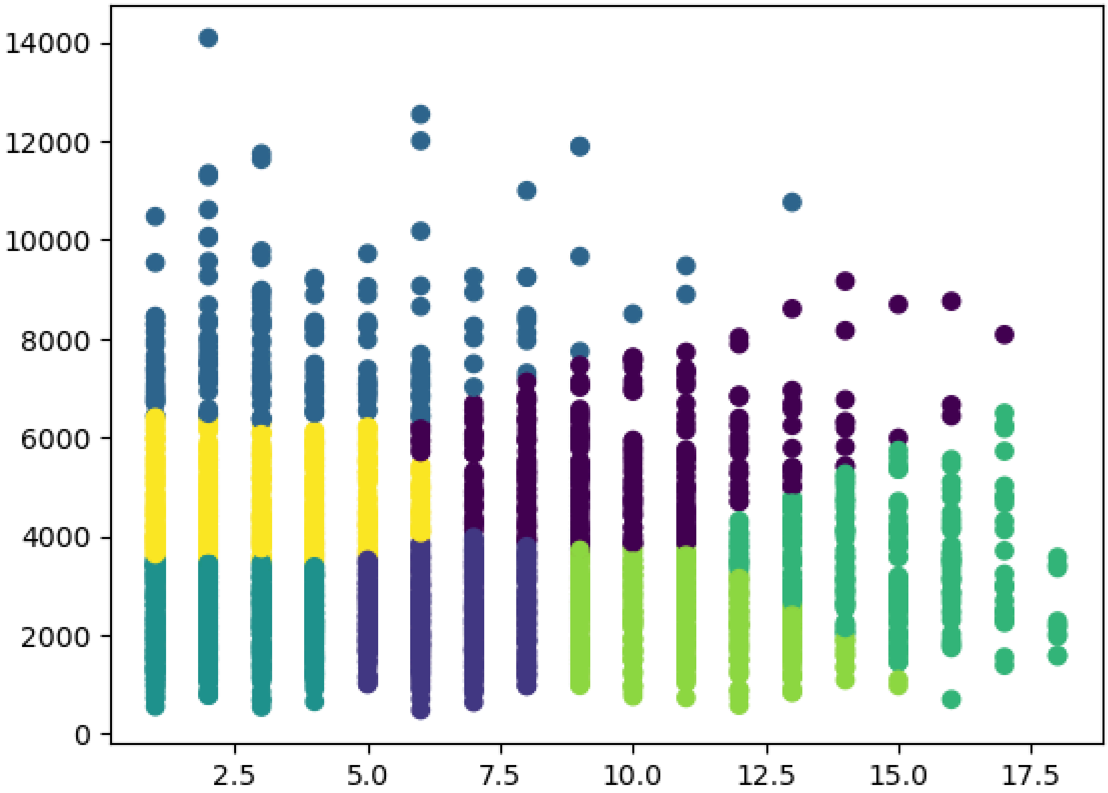
Out[11]:

```
1017.000815938199
```

Results

In [12]:

```
plt.scatter(df['ORDERLINENUMBER'], df['SALES'], c=model.labels_)
plt.show()
```


14000
12000
10000
8000
6000
4000
2000
0
2.5
5.0
7.5
10.0
12.5
15.0
17.5