Assignment 5 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 (https://www.kaggle.com/datasets/kyanyoga/sample-sales-data)

In [6]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import dendrogram,linkage
```

In [10]:
```
df = pd.read_csv("sales_data_sample.csv",encoding = "ISO-8859-1");
df
```

Out[10]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	OR
0	10107	30	95.70	2	2871.00	
1	10121	34	81.35	5	2765.90	5/7
2	10134	41	94.74	2	3884.34	7/1
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	3/1
2821	10397	34	62.24	1	2116.16	
2822	10414	47	65.52	9	3079.44	5/6

2823 rows × 25 columns

In [14]:
```
df.shape
```

Out[14]:
```
(2823, 25)
```

In [16]:
```
df.head()
```

Out[16]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	ORDEI
0	10107	30	95.70	2	2871.00	2/2
1	10121	34	81.35	5	2765.90	5/7/20(
2	10134	41	94.74	2	3884.34	7/1/20(
3	10145	45	83.26	6	3746.70	8/2
4	10159	49	100.00	14	5205.27	10/1

5 rows × 25 columns

In [18]:
```
df.tail()
```

Out[18]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	OR
2818	10350	20	100.00	15	2244.40	
2819	10373	29	100.00	1	3978.51	
2820	10386	43	100.00	4	5417.57	3/1
2821	10397	34	62.24	1	2116.16	
2822	10414	47	65.52	9	3079.44	5/6

5 rows × 25 columns

In [20]:
```
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
```

In [22]:
```
df.describe()
```

Out[22]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALE
count	2823.000000	2823.000000	2823.000000	2823.000000	2823.00000
mean	10258.725115	35.092809	83.658544	6.466171	3553.88907
std	92.085478	9.741443	20.174277	4.225841	1841.86510
min	10100.000000	6.000000	26.880000	1.000000	482.13000
25%	10180.000000	27.000000	68.860000	3.000000	2203.43000
50%	10262.000000	35.000000	95.700000	6.000000	3184.80000
75%	10333.500000	43.000000	100.000000	9.000000	4508.00000
max	10425.000000	97.000000	100.000000	18.000000	14082.80000

In [24]:
```
df.columns
```

Out[24]:
```
Index(['ORDERNUMBER', 'QUANTITYORDERED', 'PRICEEACH', 'ORDERLINENUMBER',
       'SALES', 'ORDERDATE', 'STATUS', 'QTR_ID', 'MONTH_ID', 'YEAR_ID',
       'PRODUCTLINE', 'MSRP', 'PRODUCTCODE', 'CUSTOMERNAME', 'PHONE',
       'ADDRESSLINE1', 'ADDRESSLINE2', 'CITY', 'STATE', 'POSTALCODE',
       'COUNTRY', 'TERRITORY', 'CONTACTLASTNAME', 'CONTACTFIRSTNAME',
       'DEALSIZE'],
      dtype='object')
```

In [26]:
```
df.dtypes
```

Out[26]:
```
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 [30]:
```
df.isnull().sum()
```

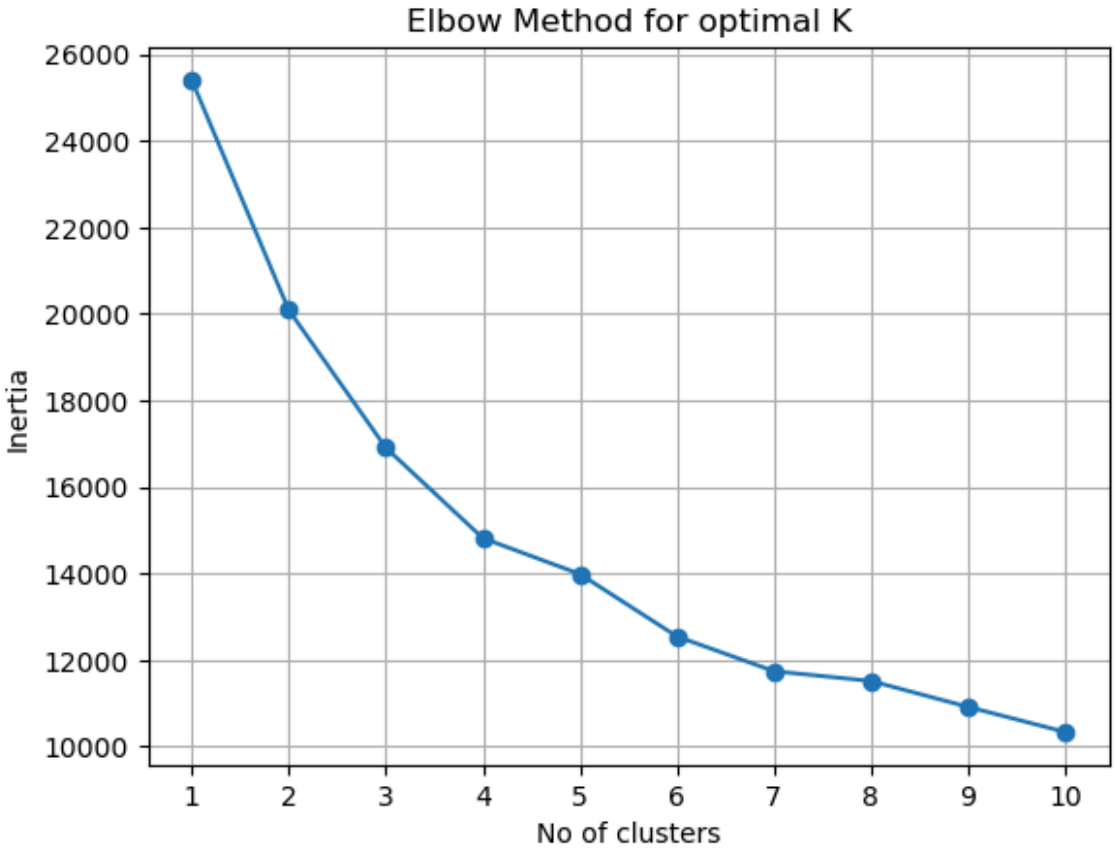
Out[30]:
```
ORDERNUMBER           0
QUANTITYORDERED       0
PRICEEACH             0
ORDERLINENUMBER       0
SALES                 0
ORDERDATE             0
STATUS                0
QTR_ID                0
MONTH_ID              0
YEAR_ID               0
PRODUCTLINE           0
MSRP                  0
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
```

In [32]:
```
df = df.drop("ADDRESSLINE1",axis = 1)
```

In [36]:
```
x = df.select_dtypes(include="number")
sc = StandardScaler()
x_scaled = sc.fit_transform(x)
```

In [38]:
```
#elbow method
inertia = []
for k in range(1,11):
    kmeans = KMeans(n_clusters = k,random_state=0)
    kmeans.fit(x_scaled)
    inertia.append(kmeans.inertia_)
```

In [44]:
```
K = range(1,11)
plt.plot(K,inertia,marker="o")
plt.xticks(K)
plt.title("Elbow Method for optimal K")
plt.xlabel("No of clusters")
plt.ylabel("Inertia")
plt.grid()
plt.show()
```



In [46]:
```
optimal_k = 4
print("Optimal Number of clusters chosen by elbow method is: ",optimal_k)
```

```
Optimal Number of clusters chosen by elbow method is:  4
```

In [52]:
```
kmeans = KMeans(n_clusters = optimal_k,random_state =0)
kmeans.fit(x_scaled)
clusters = kmeans.predict(x_scaled)
```

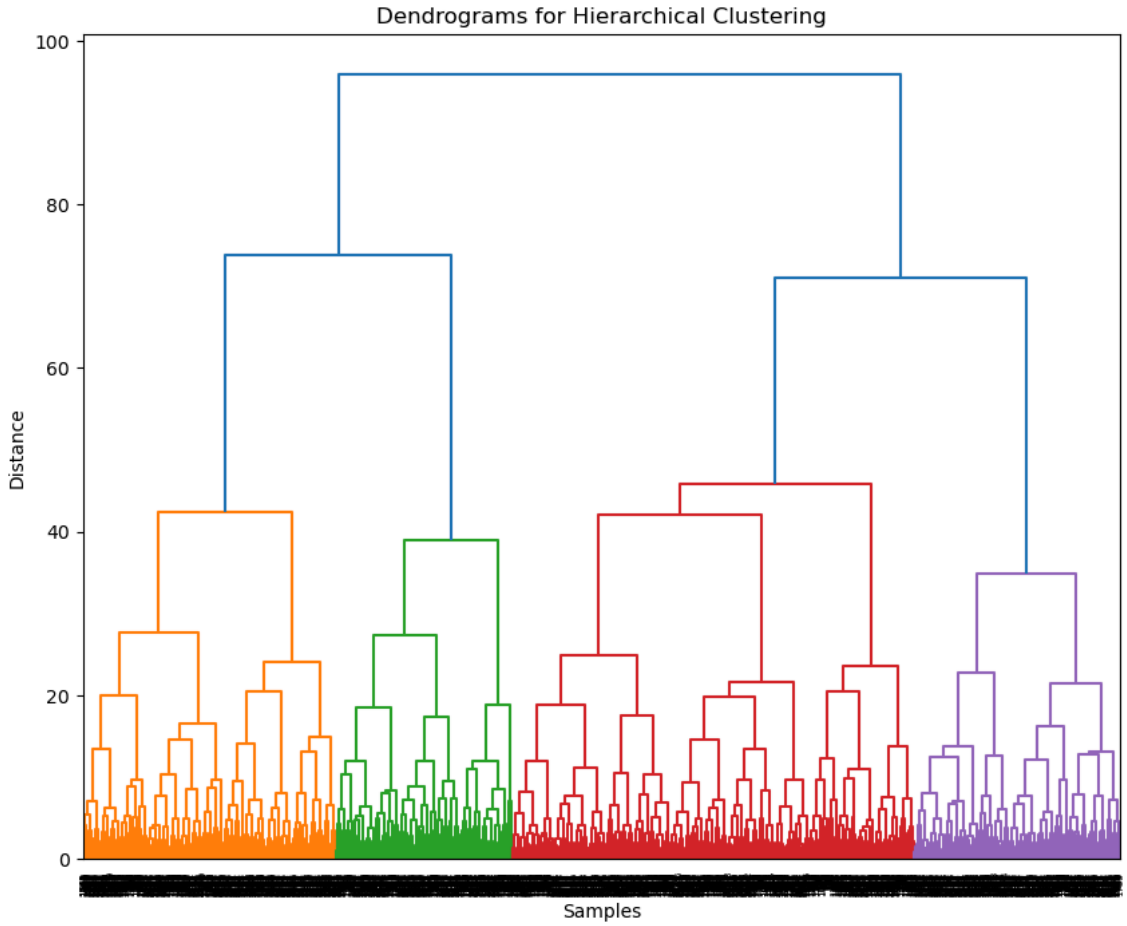
In [54]:
```
df["Cluster"] = clusters
df
```

Out[54]:

	ORDERNUMBER	QUANTITYORDERED	PRICEEACH	ORDERLINENUMBER	SALES	OR
0	10107	30	95.70	2	2871.00	
1	10121	34	81.35	5	2765.90	5/7
2	10134	41	94.74	2	3884.34	7/1
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	3/1
2821	10397	34	62.24	1	2116.16	
2822	10414	47	65.52	9	3079.44	5/6

2823 rows × 25 columns

In [56]:
```
#Hierachical clustering and dendrogram
linked = linkage(x_scaled,"ward")
plt.figure(figsize = (10,8))
dendrogram(linked,orientation = "top",distance_sort = "descending",show_lea
plt.title("Dendrograms for Hierarchical Clustering")
plt.xlabel("Samples")
plt.ylabel("Distance")
plt.show()
```



In [ ]: