In [ ]:
```python
# 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
```

In [5]:
```python
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
# from yellowbrick.cluster import KElbowVisualizer
```

In [25]:
```python
data = pd.read_csv('sales_data_sample.csv', sep = ',', encoding = 'Latin-1')
```

In [26]:
```python
data
```

Out[26]:

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

2823 rows × 25 columns

In [ ]:
```python
# Prepare the data as needed (feature selection, preprocessing, etc.)
```

In [29]:
```python
# Step 2: Select relevant features for clustering (e.g., 'QUANTITYORDERED', 'PRICEEACH')
selected_features = data[['QUANTITYORDERED', 'PRICEEACH']]
selected_features
```

Out[29]:

	QUANTITYORDERED	PRICEEACH
0	30	95.70
1	34	81.35
2	41	94.74
3	45	83.26
4	49	100.00
...	...	...
2818	20	100.00
2819	29	100.00
2820	43	100.00
2821	34	62.24
2822	47	65.52

2823 rows × 2 columns

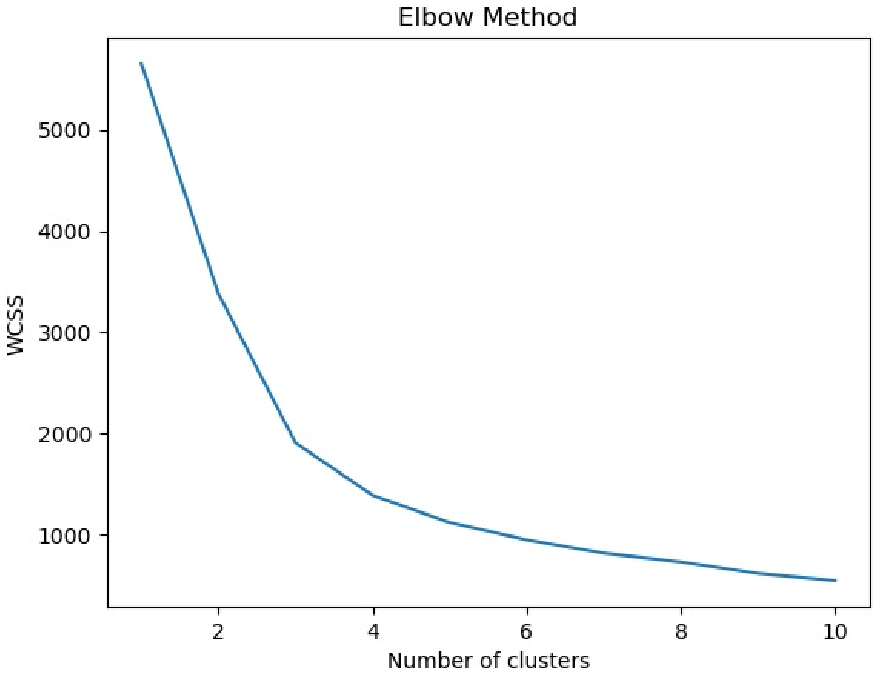
In [30]:
```python
# Step 3: Normalize the data (if needed)
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
normalized_features = scaler.fit_transform(selected_features)
```

In [31]:
```
normalized_features
```

Out[31]:
```
array([[-0.52289086,  0.5969775 ],
       [-0.11220131, -0.11445035],
       [ 0.60650538,  0.54938372],
       ...,
       [ 0.81185016,  0.81015797],
       [-0.11220131, -1.06186404],
       [ 1.2225397 , -0.89925195]])
```

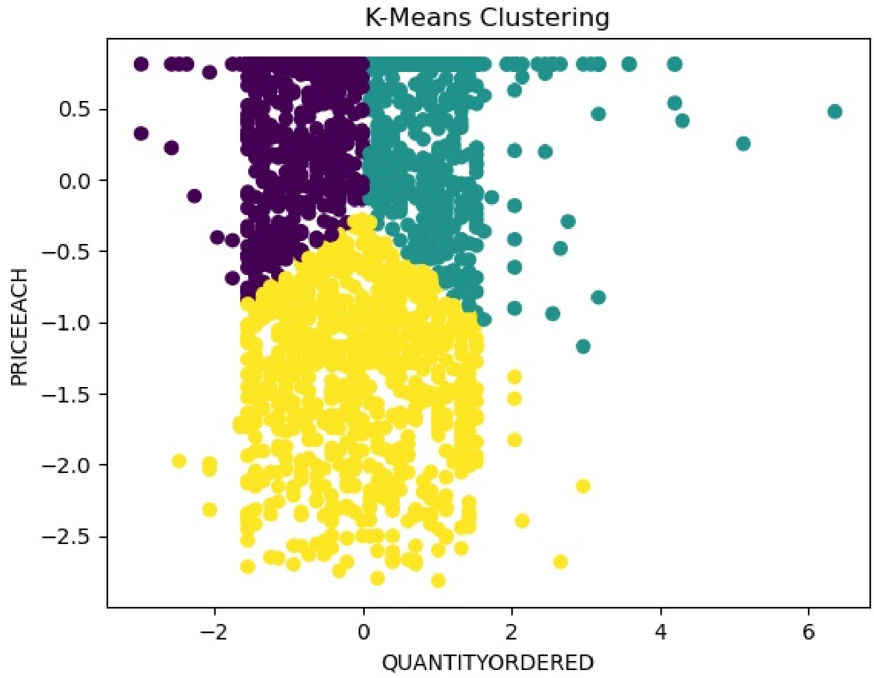
In [32]:
```
# Step 4: Determine the optimal number of clusters using the elbow method
wcss = []  # Within-cluster sum of squares
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(normalized_features)
    wcss.append(kmeans.inertia_)
```

In [33]:
```
# Plot the elbow graph
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



In [34]:
```
# Step 5: Choose the optimal number of clusters (elbow point) and perform K-Means clustering
optimal_clusters = 3  # Adjust based on the elbow point in the graph
kmeans = KMeans(n_clusters=optimal_clusters, init='k-means++', max_iter=300, n_init=10, random_state=0)
cluster_labels = kmeans.fit_predict(normalized_features)
```

In [35]:
```
# Step 6: Visualize the clusters (if possible)
plt.scatter(normalized_features[:, 0], normalized_features[:, 1], c=cluster_labels, cmap='viridis')
plt.xlabel('QUANTITYORDERED')
plt.ylabel('PRICEEACH')
plt.title('K-Means Clustering')
plt.show()
```



In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js