In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
import scipy.cluster.hierarchy as sch
```

In [4]:
```
# 1. Loading and Preprocessing the Iris dataset
iris = datasets.load_iris()
data = pd.DataFrame(iris.data, columns=iris.feature_names)
```

In [6]:
```
data
```

Out[6]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

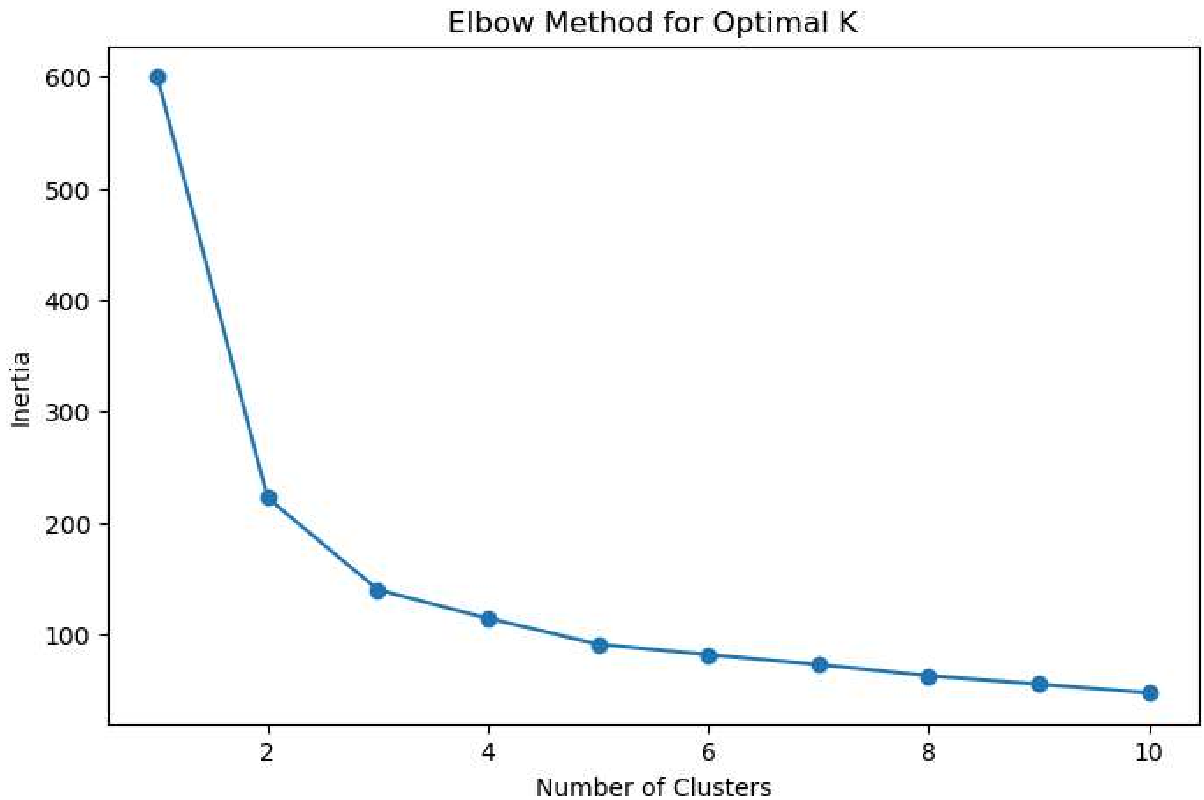
150 rows × 4 columns

In [12]:
```
# Standardizing the dataset
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
```

In [28]:
```
# 2A. KMeans Clustering
# Brief Description:
# KMeans clustering partitions the dataset into K clusters by iteratively assigning
# and updating the centroids until convergence.

# Determining the optimal number of clusters using the Elbow Method
inertia = []
k_values = range(1, 11)
for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
```

```
    kmeans.fit(data_scaled)
    inertia.append(kmeans.inertia_)
```

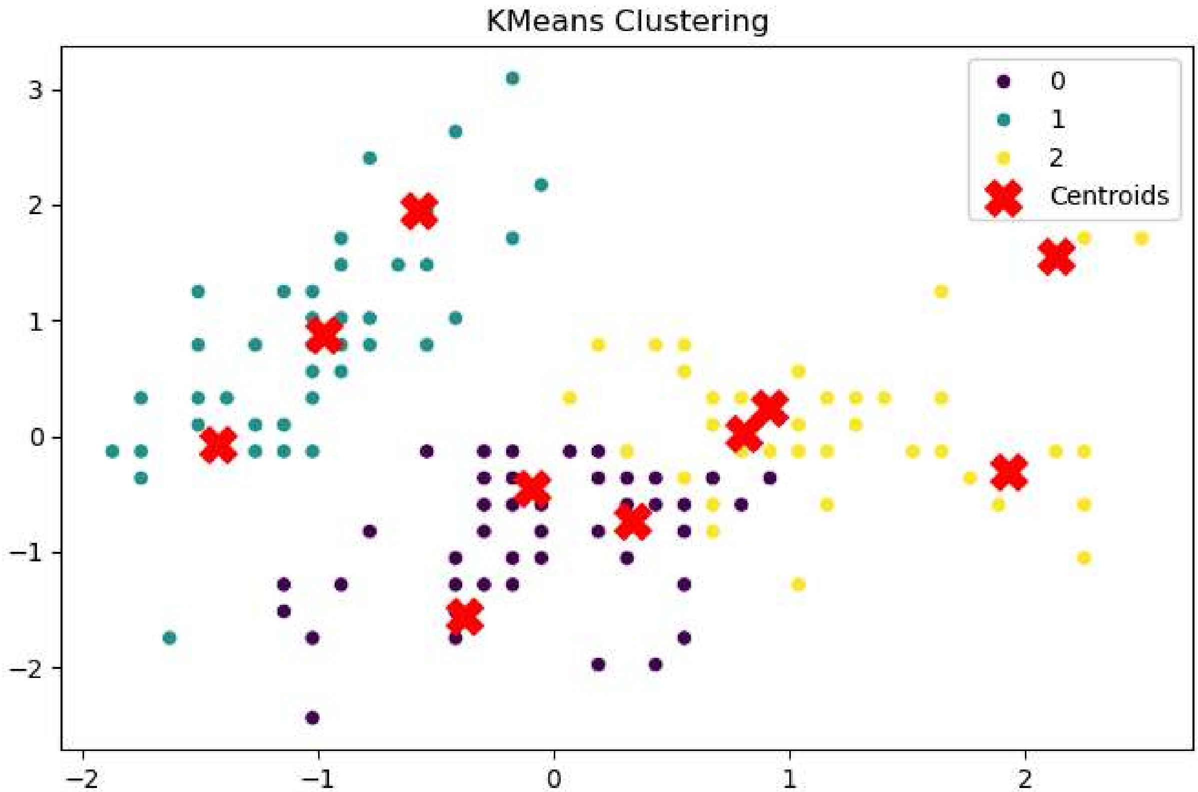
In [20]:
```
plt.figure(figsize=(8, 5))
plt.plot(k_values, inertia, marker='o', linestyle='-')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.show()
```



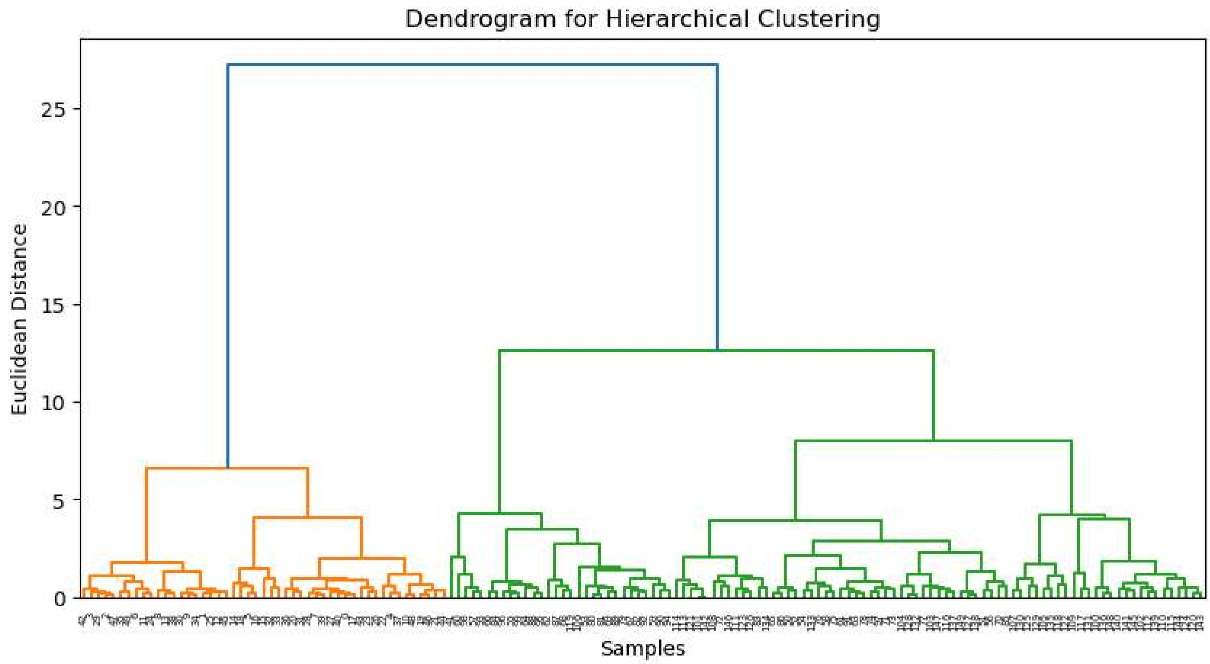
In [24]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [26]:
```
# Applying KMeans with the chosen number of clusters (K=3 from the Elbow Method)
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
clusters_kmeans = kmeans.fit_predict(data_scaled)
data['KMeans Cluster'] = clusters_kmeans
```

In [30]:
```
# Visualizing KMeans Clusters
plt.figure(figsize=(8, 5))
sns.scatterplot(x=data_scaled[:, 0], y=data_scaled[:, 1], hue=clusters_kmeans, pale
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200, c=
plt.title('KMeans Clustering')
plt.legend()
plt.show()
```

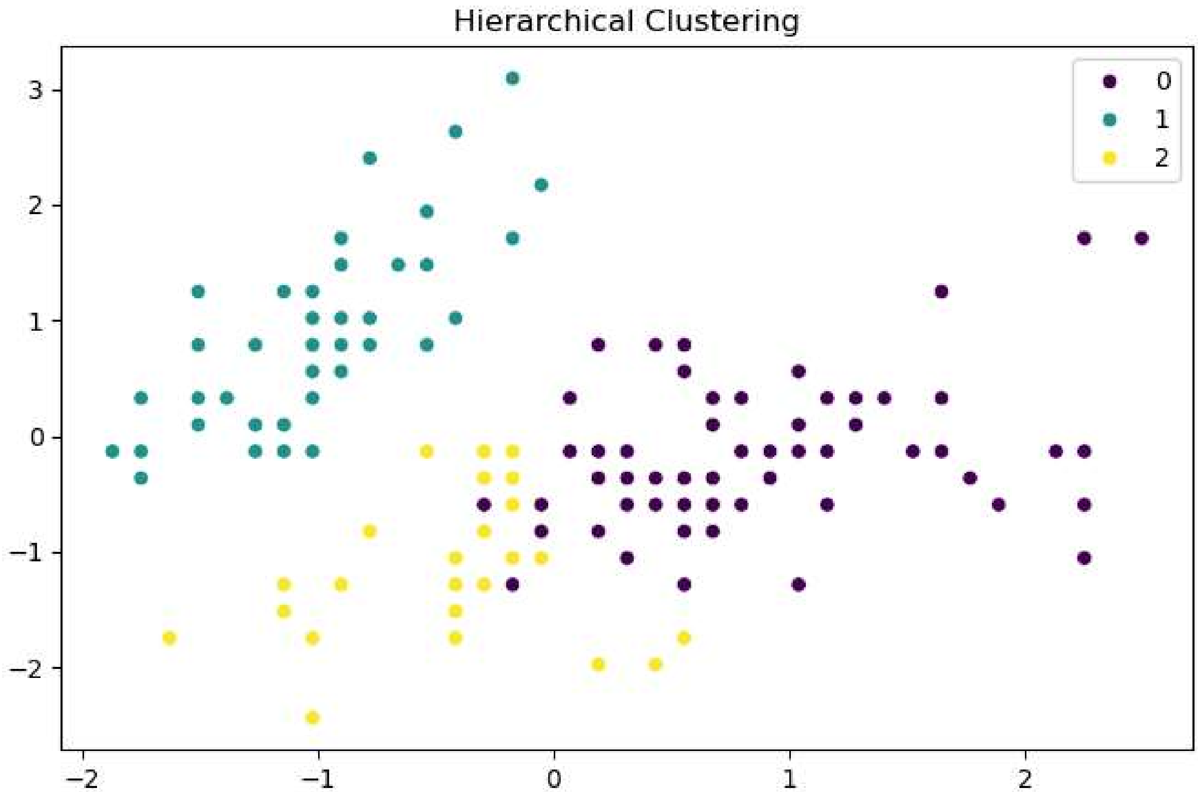


In [32]:
```
# 2B. Hierarchical Clustering
# Brief Description:
# Hierarchical clustering builds a hierarchy of clusters by either merging or split
# It does not require specifying the number of clusters beforehand.

# Creating a Dendrogram to determine the number of clusters
plt.figure(figsize=(10, 5))
dendrogram = sch.dendrogram(sch.linkage(data_scaled, method='ward'))
plt.title('Dendrogram for Hierarchical Clustering')
plt.xlabel('Samples')
plt.ylabel('Euclidean Distance')
plt.show()
```



In [36]:
```
# Applying Hierarchical Clustering with 3 clusters
hierarchical = AgglomerativeClustering(n_clusters=3, linkage='ward')
clusters_hierarchical = hierarchical.fit_predict(data_scaled)
data['Hierarchical Cluster'] = clusters_hierarchical
```

In [38]:
```
# Visualizing Hierarchical Clusters
plt.figure(figsize=(8, 5))
sns.scatterplot(x=data_scaled[:, 0], y=data_scaled[:, 1], hue=clusters_hierarchical
plt.title('Hierarchical Clustering')
plt.show()

# Save processed dataset
data.to_csv('iris_clusters.csv', index=False)
```


Hierarchical Clustering
0
1
2
3
2
1
0
−1
−2
−2
−1
0
1
2