In [15]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.model_selection import train_test_split
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.metrics import silhouette_score
```

In [16]:
```
data = pd.read_csv("Mall_Customers.csv")
print("Dataset Loaded Successfully")
print(data.head())
```

```
Dataset Loaded Successfully
   CustomerID   Genre  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

In [17]:
```
print("\n Missing Values:\n", data.isnull().sum())
```

```
 Missing Values:
 CustomerID                0
Genre                      0
Age                        0
Annual Income (k$)         0
Spending Score (1-100)     0
dtype: int64
```

In [19]:
```
print(data.columns.tolist())
```

```
['CustomerID', 'Genre', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)']
```

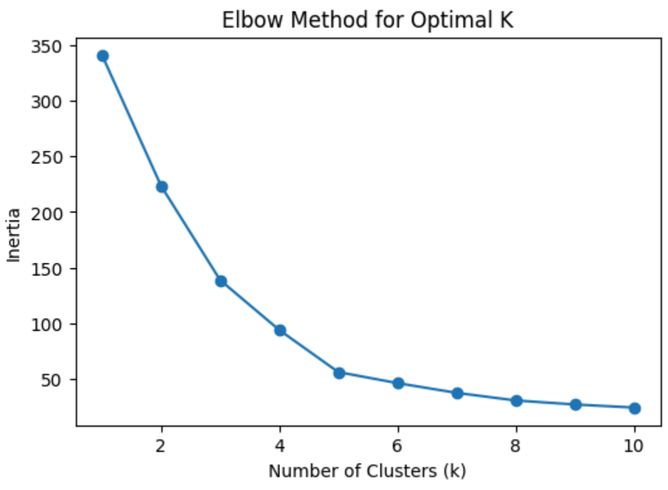
In [20]:
```
X = data[['Annual Income (k$)', 'Spending Score (1-100)']]
```

In [21]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [22]:
```
X_train, X_test = train_test_split(X_scaled, test_size=0.2, random_state=42)
```

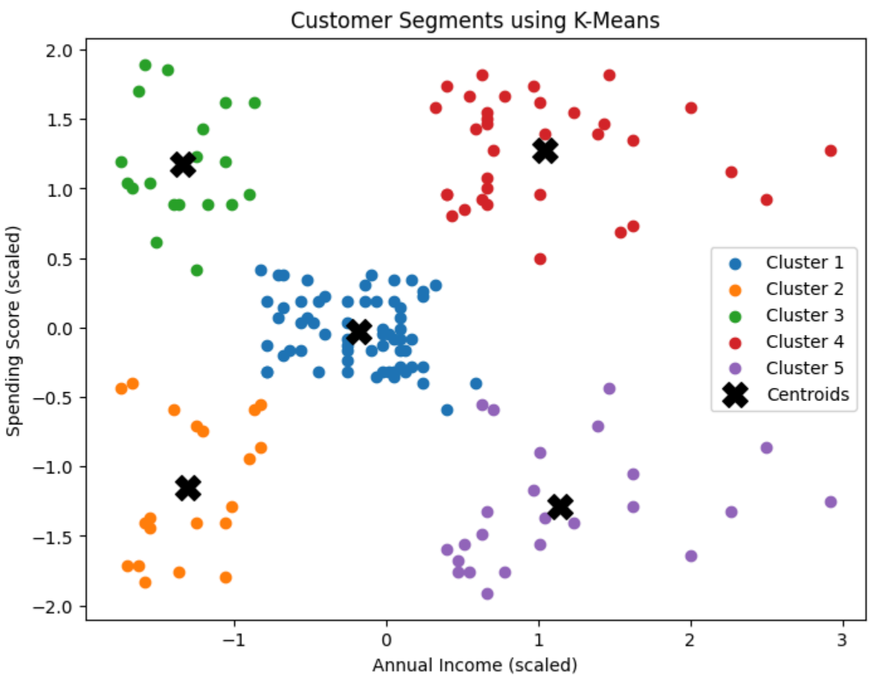
In [23]:
```
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_train)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(6,4))
plt.plot(range(1, 11), inertia, marker='o')
plt.title("Elbow Method for Optimal K")
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")
plt.show()
```



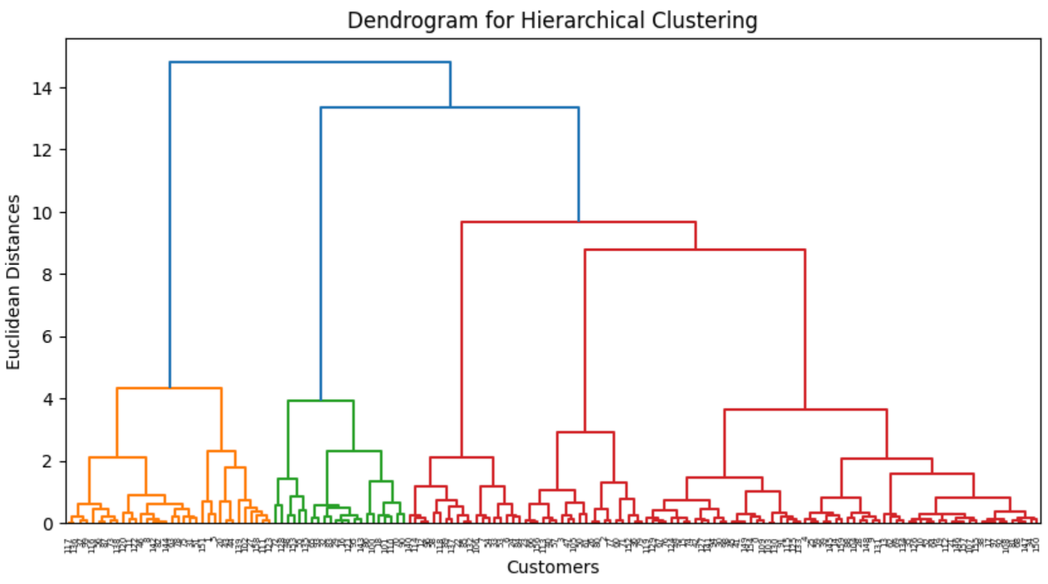

In [24]:
```
kmeans = KMeans(n_clusters=5, random_state=42, n_init=10)
y_kmeans = kmeans.fit_predict(X_train)
```

In [25]:
```
plt.figure(figsize=(8,6))
for i in range(5):
    plt.scatter(X_train[y_kmeans==i, 0], X_train[y_kmeans==i, 1], label=f'Cluster {i+1}')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],
            s=200, c='black', marker='X', label='Centroids')
plt.title("Customer Segments using K-Means")
plt.xlabel("Annual Income (scaled)")
plt.ylabel("Spending Score (scaled)")
plt.legend()
plt.show()
```




In [26]:
```
linked = linkage(X_train, method='ward')

plt.figure(figsize=(10, 5))
dendrogram(linked)
plt.title("Dendrogram for Hierarchical Clustering")
plt.xlabel("Customers")
plt.ylabel("Euclidean Distances")
plt.show()
```




In [27]:
```
hc = AgglomerativeClustering(n_clusters=5, linkage='ward')
y_hc = hc.fit_predict(X_train)
```

In [28]:
```
plt.figure(figsize=(8,6))
for i in range(5):
    plt.scatter(X_train[y_hc==i, 0], X_train[y_hc==i, 1], label=f'Cluster {i+1}')
plt.title("Customer Segments using Hierarchical Clustering")
plt.xlabel("Annual Income (scaled)")
plt.ylabel("Spending Score (scaled)")
plt.legend()
plt.show()
```


In [29]:
```
kmeans_score = silhouette_score(X_train, y_kmeans)
hc_score = silhouette_score(X_train, y_hc)
```

In [30]:
```
print("\n Silhouette Score for K-Means:", round(kmeans_score, 3))
print(" Silhouette Score for Hierarchical:", round(hc_score, 3))
```

```
Silhouette Score for K-Means: 0.551
Silhouette Score for Hierarchical: 0.552
```

In [ ]: