KMeans Clustering

KMeans clustering is an unsupervised algorithm that groups data into a predefined number of clusters, k. The process begins by randomly initializing k centroids. Each data point is then assigned to the nearest centroid based on a distance metric, typically Euclidean distance. After all points are assigned, the centroids are recalculated by taking the mean of the points within each cluster. This assignment and recalculation process is repeated until the cluster assignments stabilize, meaning they no longer change significantly. The final output consists of clusters where data points within each cluster are more similar to each other than to those in other clusters.

KMeans clustering is suitable for the Iris dataset because it contains well-defined groups of data points based on distinct measurements of iris flowers, such as sepal length and width, and petal length and width. With three species of Iris represented in the dataset, KMeans can effectively identify these natural groupings without prior knowledge of the labels. Additionally, the dataset is relatively small and exhibits a clear separation between clusters, making it ideal for KMeans, which performs best when clusters are compact and well-separated. This allows for meaningful insights into the relationships among different flower species based on their features.

In [1]:
```
1 import seaborn as sns
2 df=sns.load_dataset('iris')
3 df
```

Out[1]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [2]:
```
features=df.drop('species',axis=1)
features
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width
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 [4]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

```

In [5]:
```
scaler=StandardScaler()
scaled_features=scaler.fit_transform(features)
```

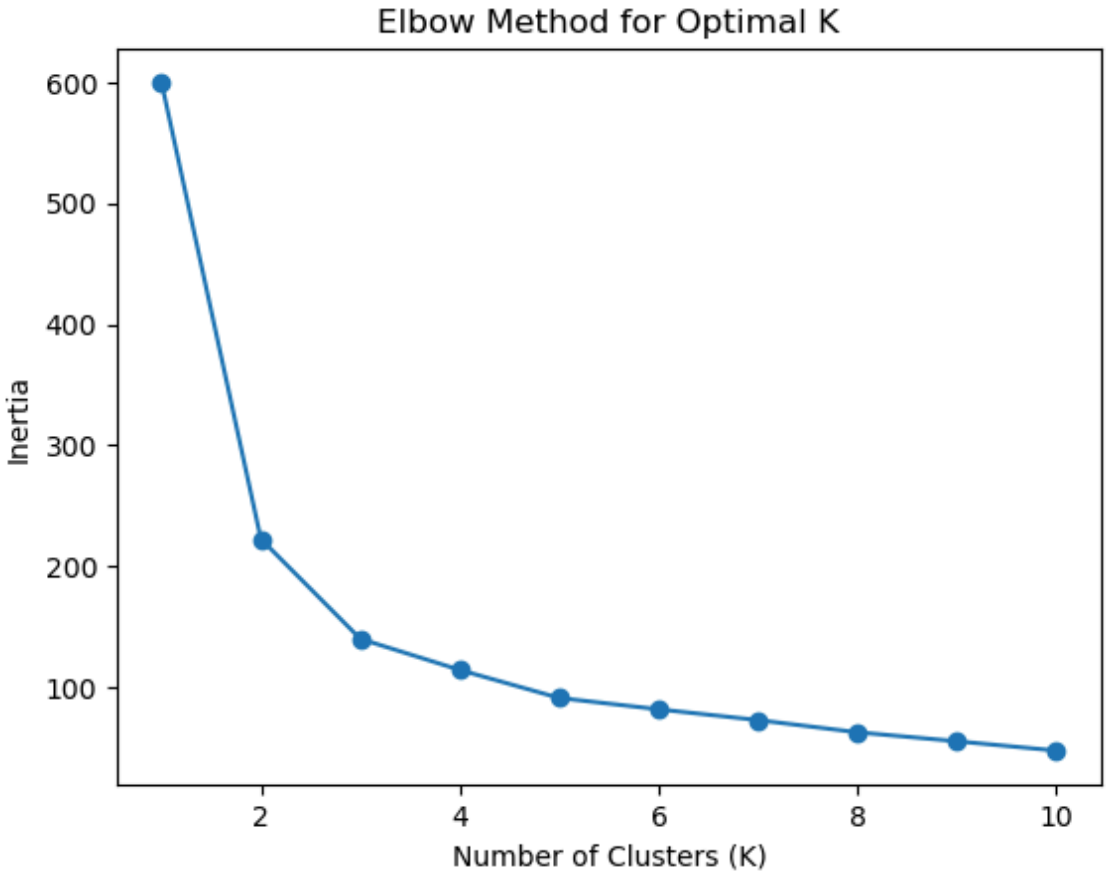
In [7]:

```
inertia=[]
k_values=range(1,11)
for k in k_values:
    kmeans=KMeans(n_clusters=k,random_state=42)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
```

```
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [8]:
```
import matplotlib.pyplot as plt
plt.plot(k_values,inertia,'o-')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.show()
```



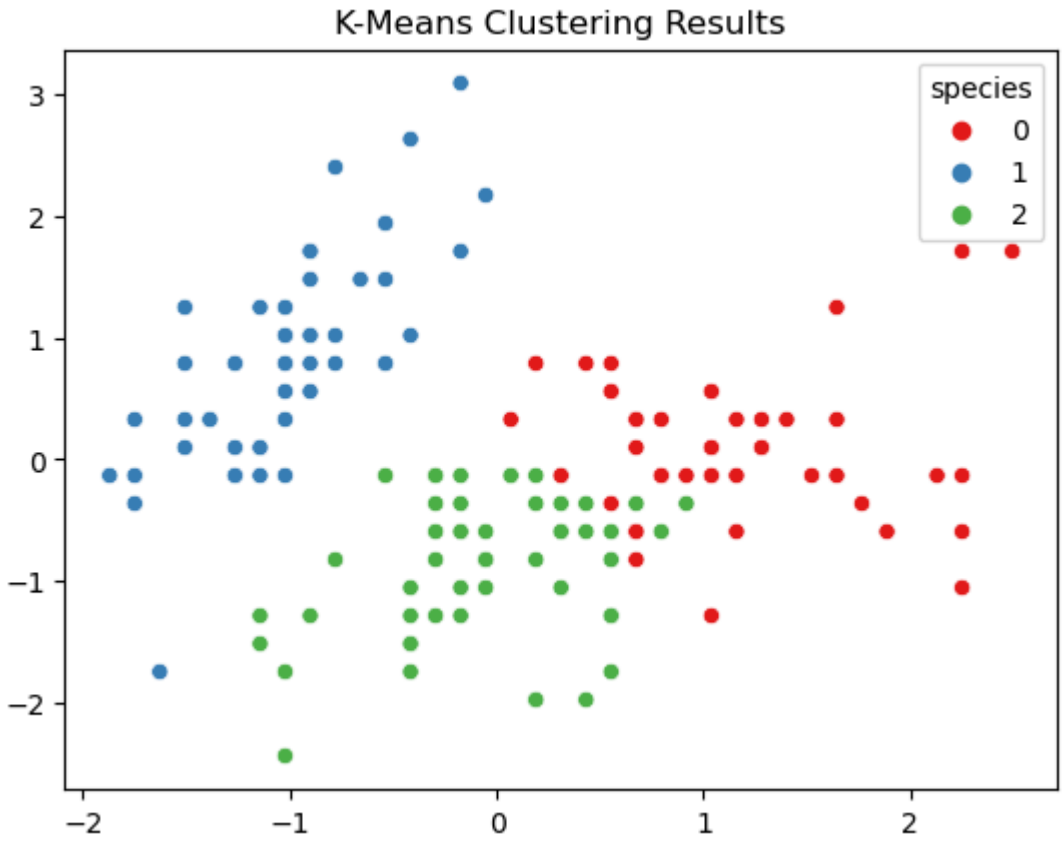
In [9]:
```
kmean=KMeans(n_clusters=3)
kmean.fit(scaled_features)
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'a
uto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MK
L, when there are less chunks than available threads. You can avoid it by
setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[9]:
```
KMeans
KMeans(n_clusters=3)
```

In [10]:
```
df['species']=kmean.labels_
```

In [11]:
```
sns.scatterplot(x=scaled_features[:,0],y=scaled_features[:,1],hue=df['
plt.title('K-Means Clustering Results')
plt.show()
```



In [12]:
```
from sklearn.metrics import silhouette_score
silhouette_score=silhouette_score(scaled_features,kmean.labels_)
silhouette_score
```

Out[12]: 0.45994823920518646

Hierarchical clustering

Hierarchical clustering is an unsupervised learning method that builds a hierarchy of clusters by either merging smaller clusters into larger ones (agglomerative) or dividing larger clusters into smaller ones (divisive). In agglomerative clustering, each data point starts as its own cluster, and pairs of clusters are merged based on their similarity, typically using a distance metric like Euclidean distance. This process continues until all points are grouped into a single cluster, resulting in a dendrogram that visually represents the clustering hierarchy.*italicized text*

Hierarchical clustering is important for the Iris dataset because it allows for the exploration of the relationships between different species of iris flowers without requiring a predefined number of clusters. By creating a dendrogram, it visually illustrates how the species are related based on their features, enabling insights into the data structure. This method can reveal the natural groupings and similarities among the flowers, helping to identify potential

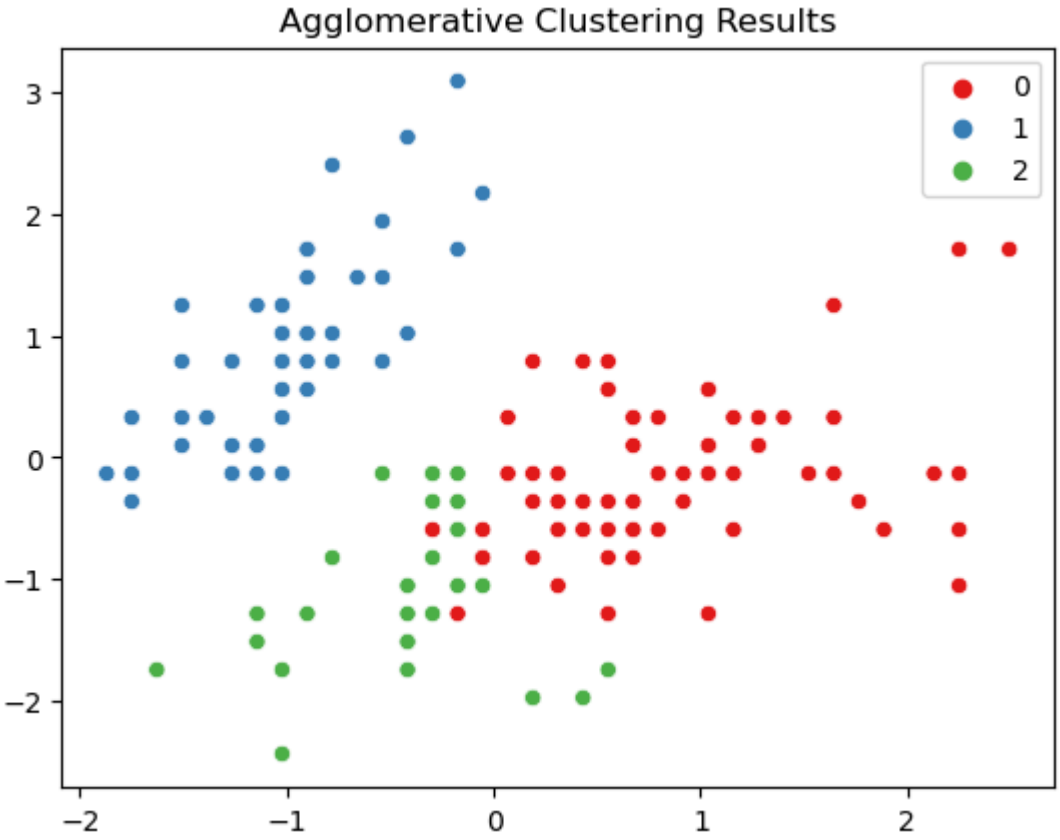
overlaps or distinctions between species. Additionally, hierarchical clustering can be

In [13]:
```
from sklearn.cluster import AgglomerativeClustering
clustering=AgglomerativeClustering(n_clusters=3)
clustering.fit(scaled_features)
```

Out[13]:
```
AgglomerativeClustering
AgglomerativeClustering(n_clusters=3)
```

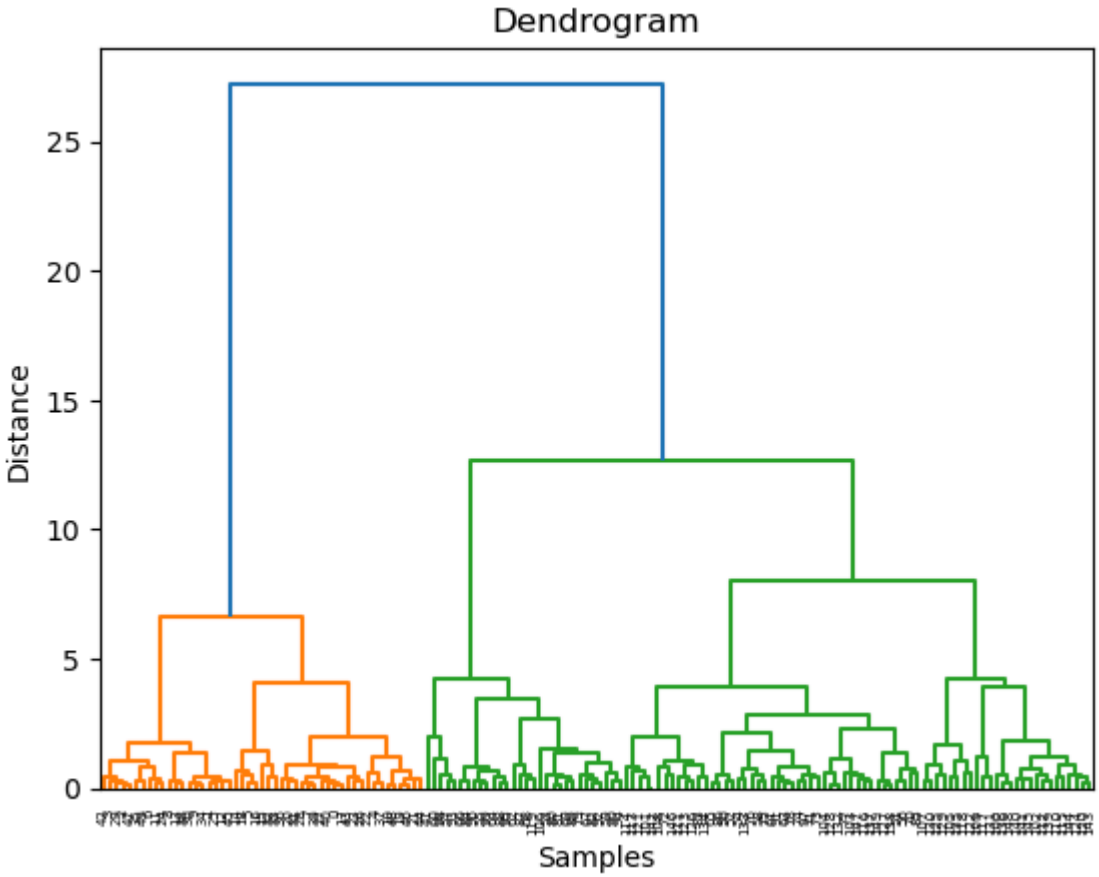
In [14]:
```
sns.scatterplot(x=scaled_features[:,0],y=scaled_features[:,1],hue=clus
plt.title('Agglomerative Clustering Results')
plt.show()

```



In [15]:
```
from scipy.cluster.hierarchy import dendrogram,linkage
linkage_matrix=linkage(scaled_features,method='ward')
dendrogram(linkage_matrix)
plt.title('Dendrogram')
plt.xlabel('Samples')
plt.ylabel('Distance')
```

Out[15]: Text(0, 0.5, 'Distance')



In [ ]:
```
1
```

In [ ]:
```
1
```

In [ ]:
```
1
```