Dataset

Use the Iris dataset available in the sklearn library.

In [3]:
```
from sklearn.datasets import load_iris
import pandas as pd

# Load the Iris dataset
iris = load_iris()

# Create a DataFrame with the features and target
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target

# Display the first few rows of the DataFrame
print(df)
```

```
     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   

     target  
0         0  
1         0  
2         0  
3         0  
4         0  
..      ...  
145       2  
146       2  
147       2  
148       2  
149       2  

[150 rows x 5 columns]
```

In [ ]:

```

```

Q1 : 1. Loading and Preprocessing :

Load the Iris dataset from sklearn. Drop the species column since this is a clustering problem.

In [4]:

```python
from sklearn.datasets import load_iris
import pandas as pd

# Load the Iris dataset
iris = load_iris()

# Create a DataFrame with the features
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

# Display the first few rows before dropping the species column (target col
df
```

Out[4]:

	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

2.Clustering Algorithm Implementation :

Implement the following two clustering algorithms:

(A) K-Means Clustering

How K-Means Clustering Works: K-Means is an unsupervised learning algorithm used for clustering data into k distinct groups. The algorithm works as follows:

Initialization: Randomly choose k centroids, where k is the number of clusters. Assignment Step: Assign each data point to the nearest centroid, forming k clusters. Update Step: Recalculate the centroids as the mean of all the points in each cluster. Repeat: Iterate the assignment and update steps until the centroids no longer change or the change is below a predefined threshold. Why K-Means Clustering is Suitable for the Iris Dataset: The Iris dataset contains three distinct species of iris flowers, making it a natural candidate for clustering. Since K-Means is effective at finding spherical clusters in a dataset, it can be used to identify the natural groupings in the Iris dataset, corresponding to the three species.

Let's implement K-Means clustering on the Iris dataset and visualize the clusters.

(B) Hierarchical Clustering

How Hierarchical Clustering Works: Hierarchical clustering is another unsupervised learning algorithm that builds a hierarchy of clusters:

Agglomerative (Bottom-Up) Approach:

Start with each data point as its own cluster. Iteratively merge the two closest clusters until all points are in a single cluster or a predefined number of clusters is reached. Divisive (Top-Down) Approach:

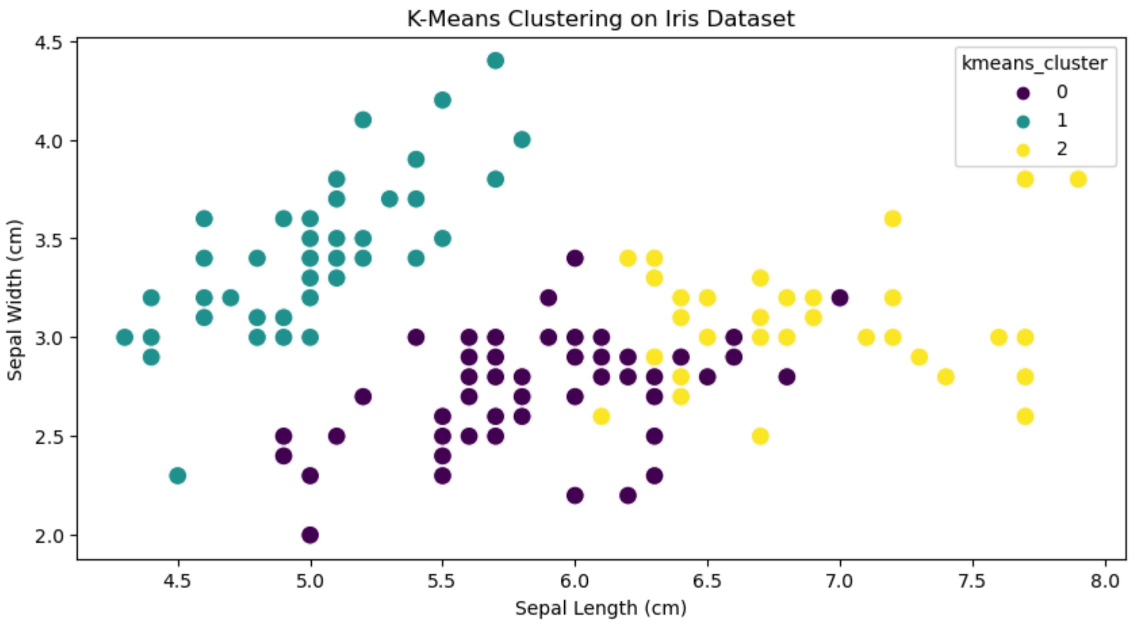
Start with all data points in a single cluster. Iteratively split clusters until each point is in its own cluster. The agglomerative approach is more common and can be visualized using a dendrogram, which shows the hierarchy of cluster formations.

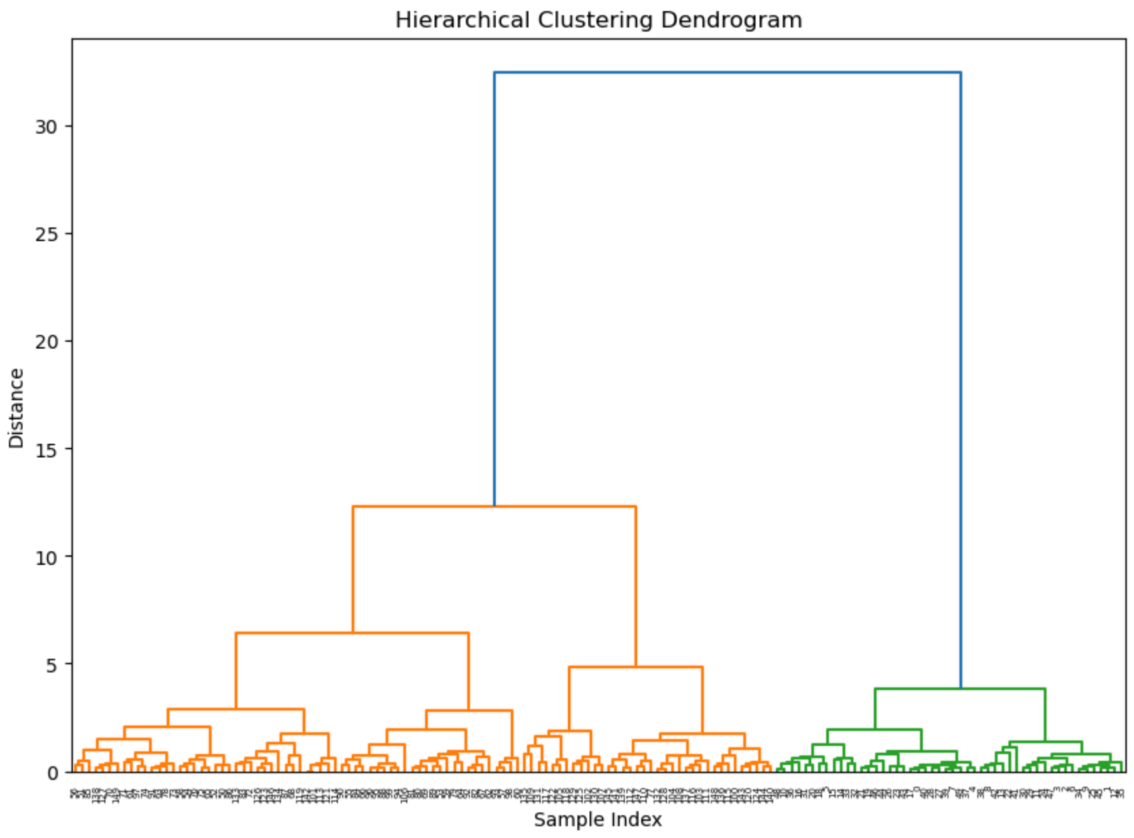
Why Hierarchical Clustering is Suitable for the Iris Dataset: Hierarchical clustering does not require the number of clusters to be specified in advance and can reveal the structure of data at different levels of granularity. This is particularly useful for exploring the natural groupings within the Iris dataset and determining if the species have a hierarchical relationship.

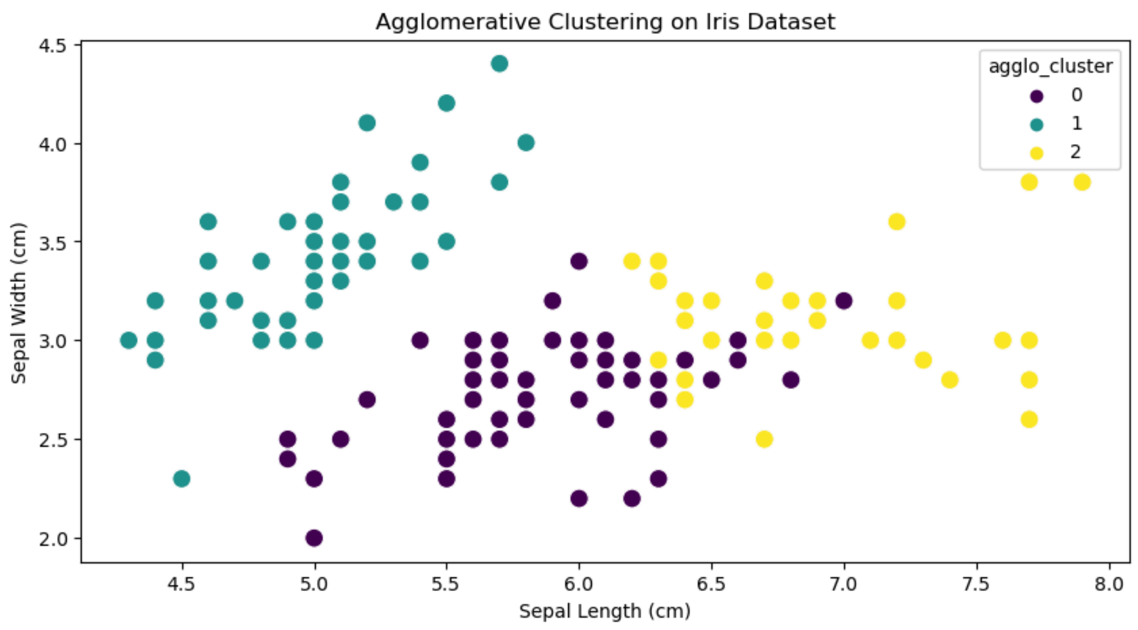
Let's implement hierarchical clustering on the Iris dataset and visualize the clusters using a dendrogram and scatter plots.

In [5]:
```
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
import seaborn as sns

# Apply K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
df['kmeans_cluster'] = kmeans.fit_predict(df)

# Visualize K-Means Clusters
plt.figure(figsize=(10, 5))
sns.scatterplot(x=df['sepal length (cm)'], y=df['sepal width (cm)'],
                hue=df['kmeans_cluster'], palette='viridis', s=100)
plt.title('K-Means Clustering on Iris Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()

# Hierarchical Clustering using Agglomerative Approach
linked = linkage(df.drop('kmeans_cluster', axis=1), method='ward')

# Dendrogram for Hierarchical Clustering
plt.figure(figsize=(10, 7))
dendrogram(linked, orientation='top', distance_sort='descending', show_leaf
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Sample Index')
plt.ylabel('Distance')
plt.show()

# Apply Agglomerative Clustering
agglo = AgglomerativeClustering(n_clusters=3)
df['agglo_cluster'] = agglo.fit_predict(df.drop('kmeans_cluster', axis=1))

# Visualize Agglomerative Clustering
plt.figure(figsize=(10, 5))
sns.scatterplot(x=df['sepal length (cm)'], y=df['sepal width (cm)'],
                hue=df['agglo_cluster'], palette='viridis', s=100)
plt.title('Agglomerative Clustering on Iris Dataset')
plt.xlabel('Sepal Length (cm)')
plt.ylabel('Sepal Width (cm)')
plt.show()
```

```
C:\Users\Admin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:141
2: FutureWarning: The default value of `n_init` will change from 10 to 'au
to' 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:\Users\Admin\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:143
6: UserWarning: KMeans is known to have a memory leak on Windows with MKL,
when there are less chunks than available threads. You can avoid it by set
ting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```


K-Means Clustering on Iris Dataset
kmeans_cluster
0
1
2
Sepal Width (cm)
Sepal Length (cm)
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Hierarchical Clustering Dendrogram
Distance
Sample Index
30
25
20
15
10
5
0



(A) K-Means Clustering

The K-Means clustering algorithm was applied to the Iris dataset, and the resulting clusters were visualized in a scatter plot. The plot shows how the data points were grouped into three clusters based on their sepal length and sepal width.

(B) Hierarchical Clustering

Hierarchical clustering was applied using the agglomerative approach, and a dendrogram was generated to visualize the hierarchical relationships between data points. The dendrogram illustrates how data points merge into clusters at different levels of similarity.

The scatter plot below the dendrogram shows the results of agglomerative clustering, with data points grouped into three clusters based on their sepal length and sepal width.

Both clustering methods reveal the natural grouping of the Iris dataset into three distinct clusters, which correspond to the three species of iris flowers.