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

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=data.drop(columns='species',axis=0)
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 [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [4]:
```
features.describe()
```

Out[4]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

KMeans Clustering

KMeans is an iterative clustering algorithm that divides the dataset into a predefined number of clusters (k).It randomly initializes k centroids, then assigns each point to the nearest centroid.Centroids are recalculated based on the mean of points assigned to them, and the process repeats until convergence.

KMeans is suitable for the Iris dataset because Iris has 3 species (which can be thought of as natural clusters). KMeans works well when the number of clusters is known, and Iris has clear boundaries between species based on feature values.

In [5]:
```
from sklearn.preprocessing import StandardScaler
standard=StandardScaler()
scaled_features=standard.fit_transform(features)
from sklearn.cluster import KMeans
```

In [6]:
```
inertia=[]
k_values=range(1,11)
for k in k_values:
    kmeans=KMeans(n_clusters=k)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)
```

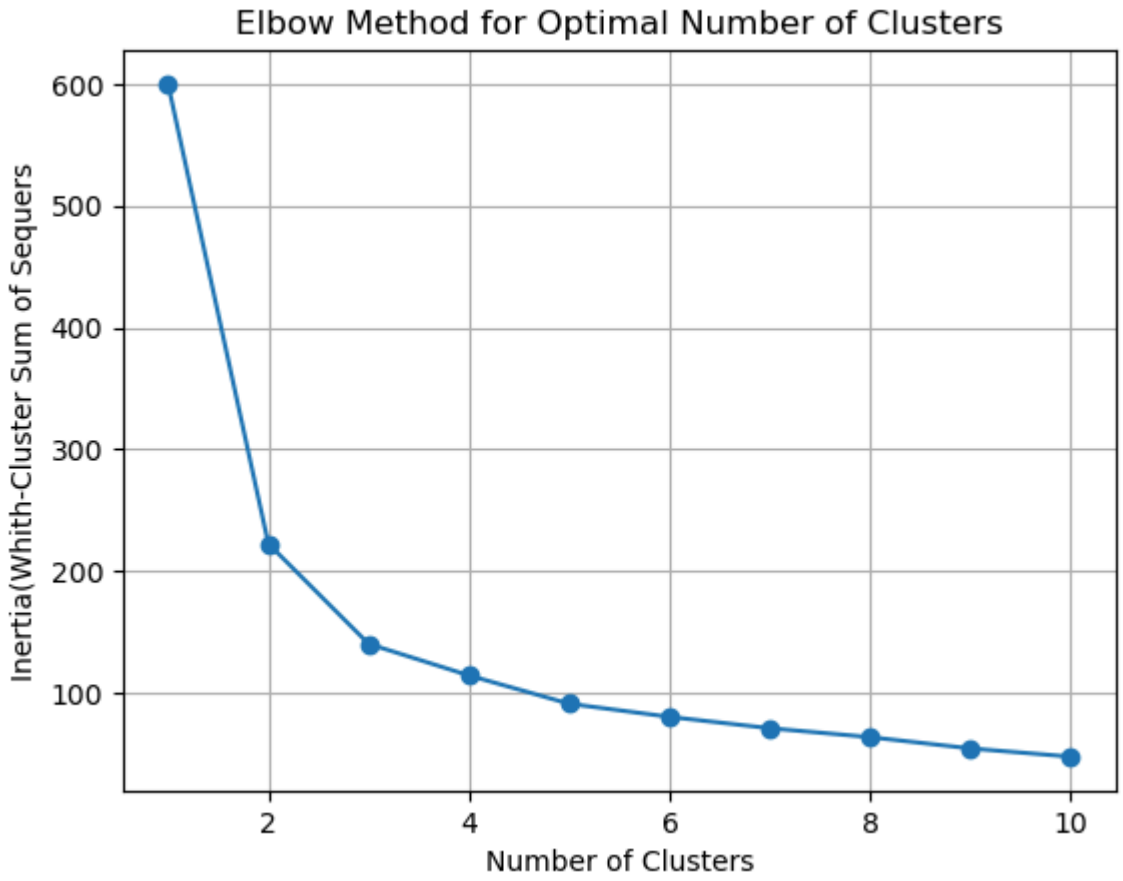
```
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
```

In [7]:
```
inertia
```

Out[7]:
```
[599.9999999999999,
 222.36170496502297,
 140.03275277428642,
 114.09223843568266,
 90.83306341870077,
 80.05860439723962,
 70.72575920187039,
 63.40079852485549,
 54.13523872625404,
 47.3551040908476]
```

In [8]:
```
import matplotlib.pyplot as plt
plt.plot(k_values,inertia,marker='o')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia(Whith-Cluster Sum of Sequers')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.grid(True)
plt.show()
```



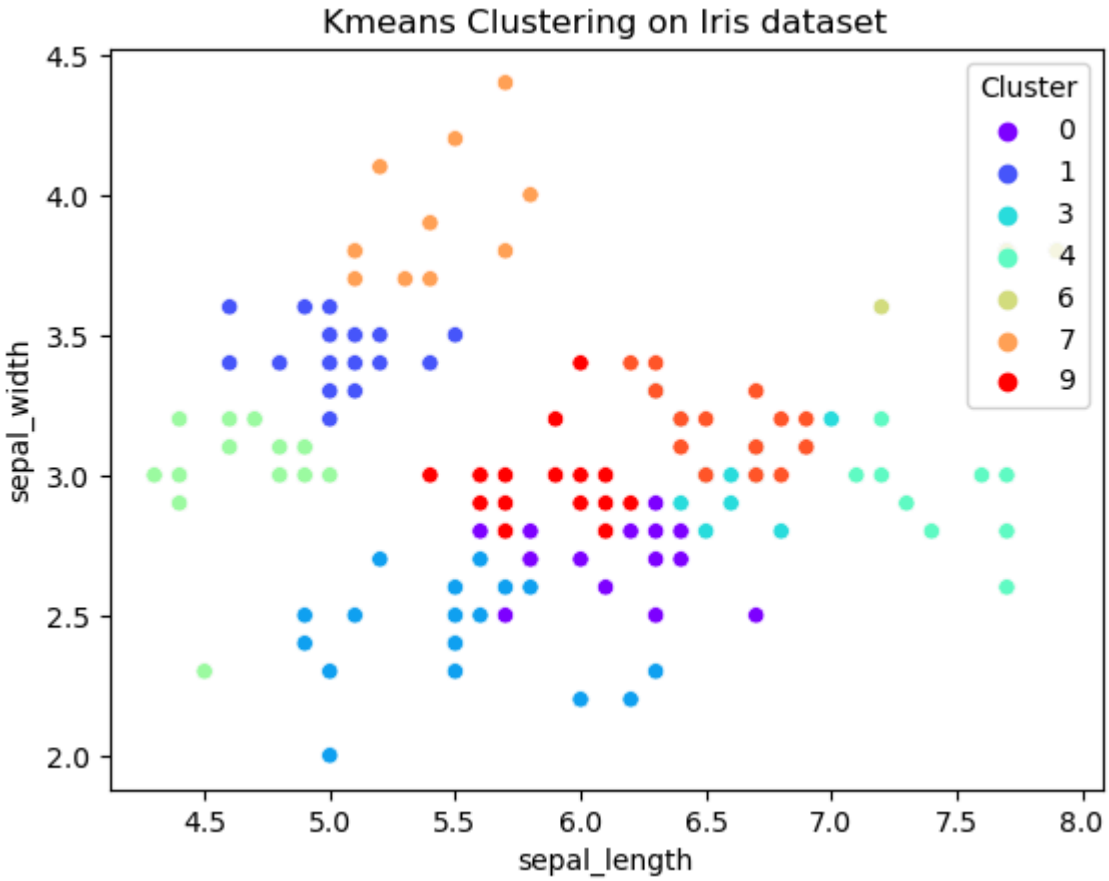
In [9]:
```
from sklearn.cluster import KMeans
Kmeans=KMeans(n_clusters=3)
kmeans.fit_predict(scaled_features)
data['Cluster']=kmeans.labels_
```

```
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureW
arning: The default value of `n_init` will change from 10 to 'auto' 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\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWar
ning: KMeans is known to have a memory leak on Windows with MKL, when there are les
s chunks than available threads. You can avoid it by setting the environment variab
le OMP_NUM_THREADS=1.
  warnings.warn(
```

In [10]:
```
list(data)
```

Out[10]:
```
['sepal_length',
 'sepal_width',
 'petal_length',
 'petal_width',
 'species',
 'Cluster']
```

In [11]:
```
import seaborn as sns
sns.scatterplot(x=features['sepal_length'],y=features['sepal_width'],hue=data['Clust
plt.title('Kmeans Clustering on Iris dataset')
plt.show()
```



b) Hierarchial Clustering

Brief Description: Hierarchical clustering builds a hierarchy of clusters using either an agglomerative (bottom-up) or divisive (top-down) approach. In agglomerative clustering, it starts with each point as its own cluster and merges the closest pairs of clusters iteratively until a single cluster is formed. The results are often visualized using a dendrogram.

Why is Hierarchical clustering suitable for the Iris dataset? Hierarchical clustering is good for small datasets like Iris since it helps in visualizing how data points cluster at different levels of granularity. It does not require the number of clusters to be pre-specified, allowing for exploration.

In [18]:
```
data=sns.load_dataset('iris')
```

In [13]:
```
from sklearn.cluster import AgglomerativeClustering
hc=AgglomerativeClustering(n_clusters=3,affinity='euclidean',linkage='ward')
data['cluster']=hc.fit_predict(scaled_features)
data
```

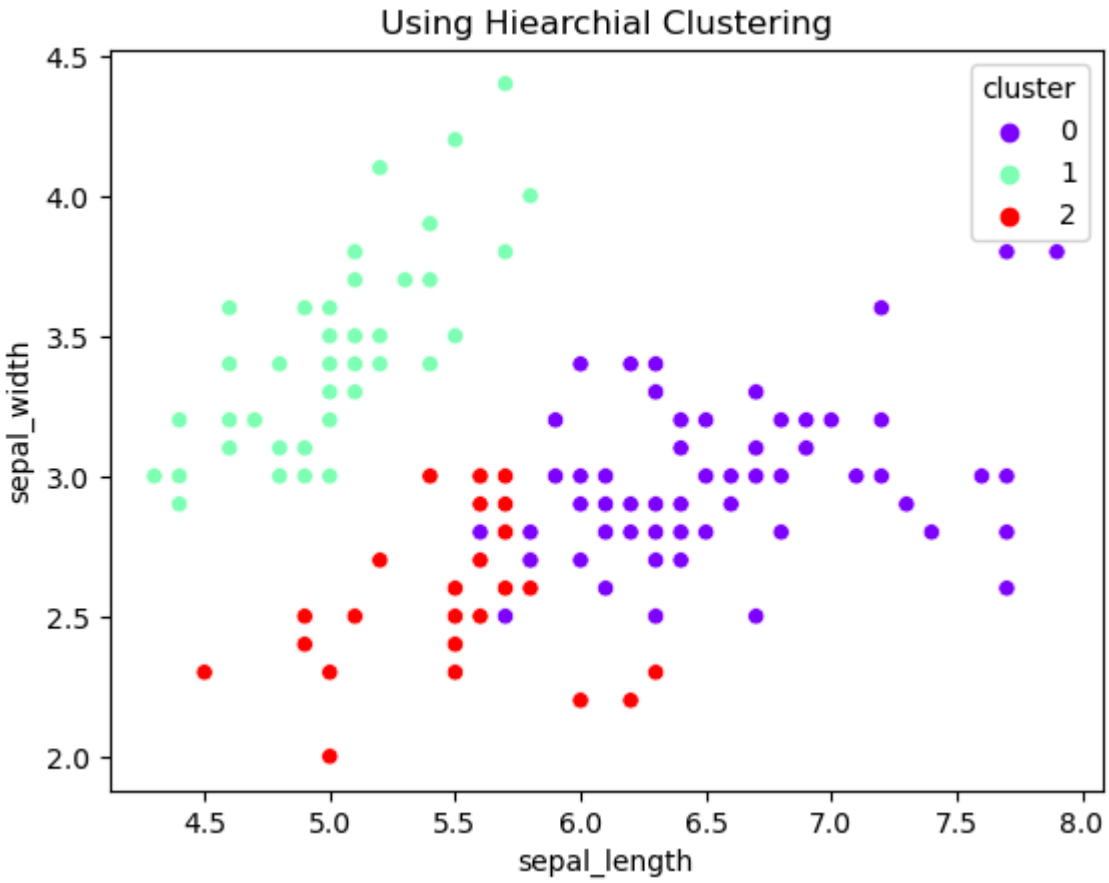
```
C:\Users\aathi\anaconda3\Lib\site-packages\sklearn\cluster\_agglomerative.py:1005:
FutureWarning: Attribute `affinity` was deprecated in version 1.2 and will be remov
ed in 1.4. Use `metric` instead
  warnings.warn(
```

Out[13]:

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

150 rows × 6 columns

In [14]:
```
sns.scatterplot(x='sepal_length',y='sepal_width',data=data,hue='cluster',palette='ra
plt.title('Using Hiearchial Clustering')
plt.show()
```



In [15]:
```
from scipy.cluster.hierarchy import dendrogram,linkage
z=linkage(scaled_features,method='ward')
```

In [16]:
```
lab=data['species'].tolist()
```

In [17]:
```
dendrogram(z,labels=lab,leaf_rotation=90)
plt.title('Dendrogram of Agglomerative Hierarchical Clustering')
plt.xlabel('Species')
plt.ylabel('Distance')
plt.show()
```