Objective:

The objective of this assessment is to evaluate your understanding and ability to apply clustering techniques to a real-world dataset.

1. Loading and Preprocessing ¶

*Load the Iris dataset from sklearn.

In [9]:
```
from sklearn.datasets import load_iris
import pandas as pd
iris=load_iris()
df=pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species']=iris.target
df['species']=df['species'].map({0:'setosa',1:'versicolor',2:'virginica'})
df.head()
```

Out[9]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	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

*Drop the species column since this is a clustering problem.

In [16]:
```
data=df.drop(columns=['species'])
```

In [17]:
```
data
```

Out[17]:

	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:

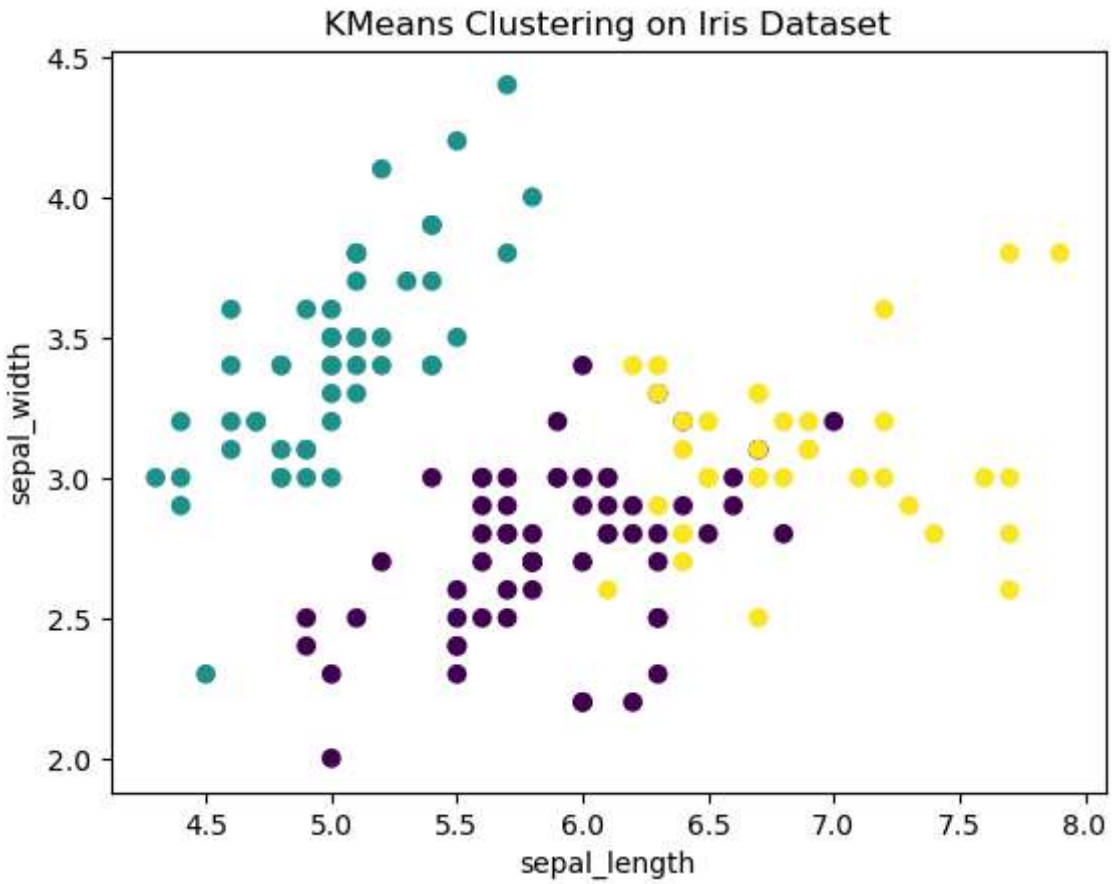
A) KMeans Clustering

Description of KMeans Clustering: KMeans clustering is an iterative algorithm that partitions the dataset into K distinct, non-overlapping clusters. The algorithm assigns each data point to the cluster with the nearest mean, which serves as the centroid of the cluster. The centroids are then updated based on the mean of the points in each cluster. This process repeats until the centroids no longer change, indicating that the algorithm has converged.

Suitability for the Iris Dataset: KMeans is suitable for the Iris dataset because it assumes that the clusters are roughly spherical and equally sized, which aligns with the dataset's characteristics. The Iris dataset has three species, and KMeans can attempt to find these underlying groupings based on the features.

In [18]:
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
kmeans=KMeans(n_clusters=3,random_state=42)
kmeans.fit(data)
df['KMeans_cluster']=kmeans.labels_
plt.scatter(df.iloc[:,0],df.iloc[:,1],c=df['KMeans_cluster'],cmap='viridis')
plt.xlabel('sepal_length')
plt.ylabel('sepal_width')
plt.title('KMeans Clustering on Iris Dataset')
plt.show()
```

```
C:\Users\15196\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1412: F
utureWarning: 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\15196\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: U
serWarning: KMeans is known to have a memory leak on Windows with MKL, when t
here are less chunks than available threads. You can avoid it by setting the
environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```



B) Hierarchical Clustering

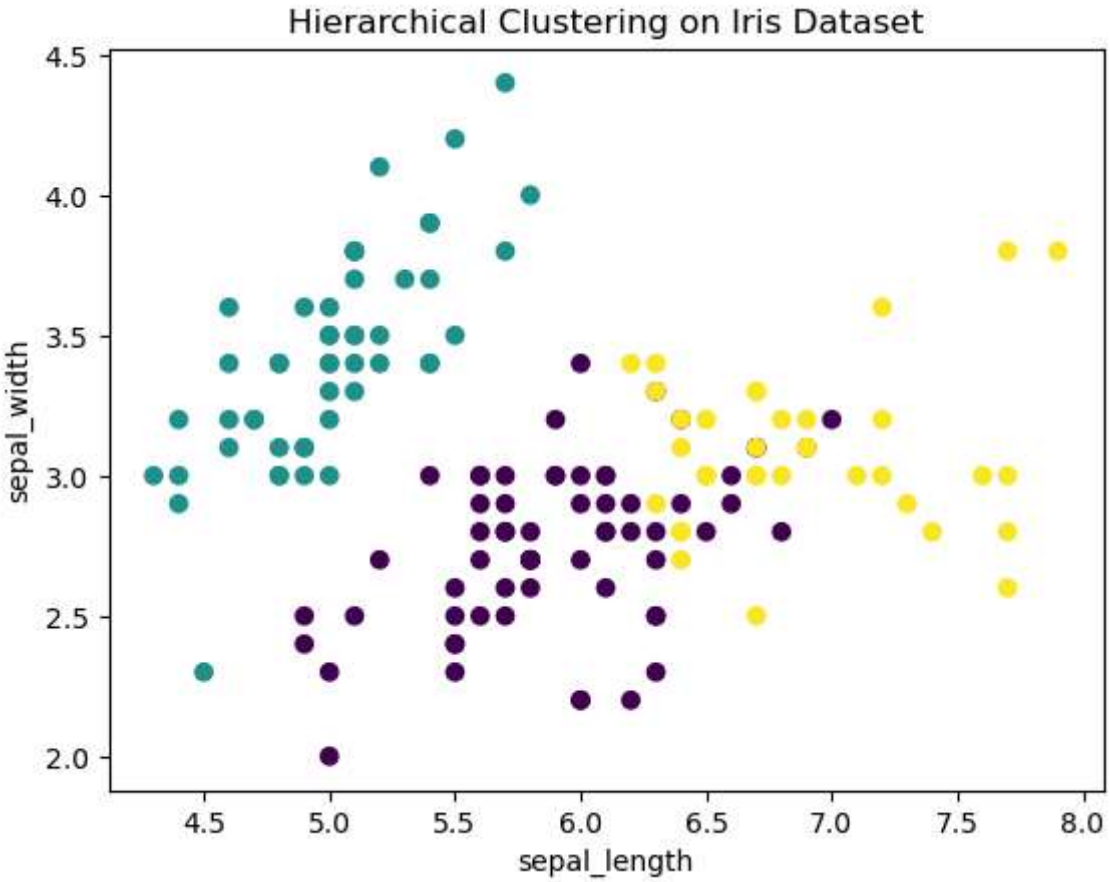
Description of Hierarchical Clustering: Hierarchical clustering creates a tree structure (dendrogram) representing nested groupings of data points. There are two main types: agglomerative (bottom-up) and divisive (top-down). Agglomerative clustering starts with each

data point as its own cluster and repeatedly merges the closest pairs of clusters until all data points belong to a single cluster. Divisive clustering works in the opposite direction, starting with all data points in a single cluster and splitting them iteratively.

Suitability for the Iris Dataset: Hierarchical clustering is suitable for the Iris dataset because it doesn't require specifying the number of clusters beforehand and can provide insights into the data's structure at different levels of granularity.

In [20]:
```
from scipy.cluster.hierarchy import dendrogram,linkage
from sklearn.cluster import AgglomerativeClustering
linked=linkage(data,'ward')
plt.figure(figsize=(10,7))
dendrogram(linked,orientation='top',distance_sort='descending',show_leaf_count
plt.title('Hierarchial Clustering Dendrogram')
plt.show()

agglomerative = AgglomerativeClustering(n_clusters=3)
data['Hierarchical_Cluster']=agglomerative.fit_predict(data)
plt.scatter(data.iloc[:,0],df.iloc[:,1], c= data['Hierarchical_Cluster'],cmap=
plt.xlabel('sepal_length')
plt.ylabel('sepal_width')
plt.title('Hierarchical Clustering on Iris Dataset')
plt.show()
```


Hierarchical Clustering on Iris Dataset
sepal_width
sepal_length
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

In [ ]: