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

Dataset Use the Iris dataset available in the sklearn library.

Key components to be fulfilled :

1. Loading and Preprocessing

In [1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Load the Iris dataset from sklearn.

In [2]:
```
data = sns.load_dataset('iris')
data
```

Out[2]:

	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 [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]:
```
data.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

In [5]:
```
data.shape
```

Out[5]:
```
(150, 5)
```

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

Drop the species column since this is a clustering problem.

In [7]:
```
features = data.drop('species',axis=1)
features
```

Out[7]:

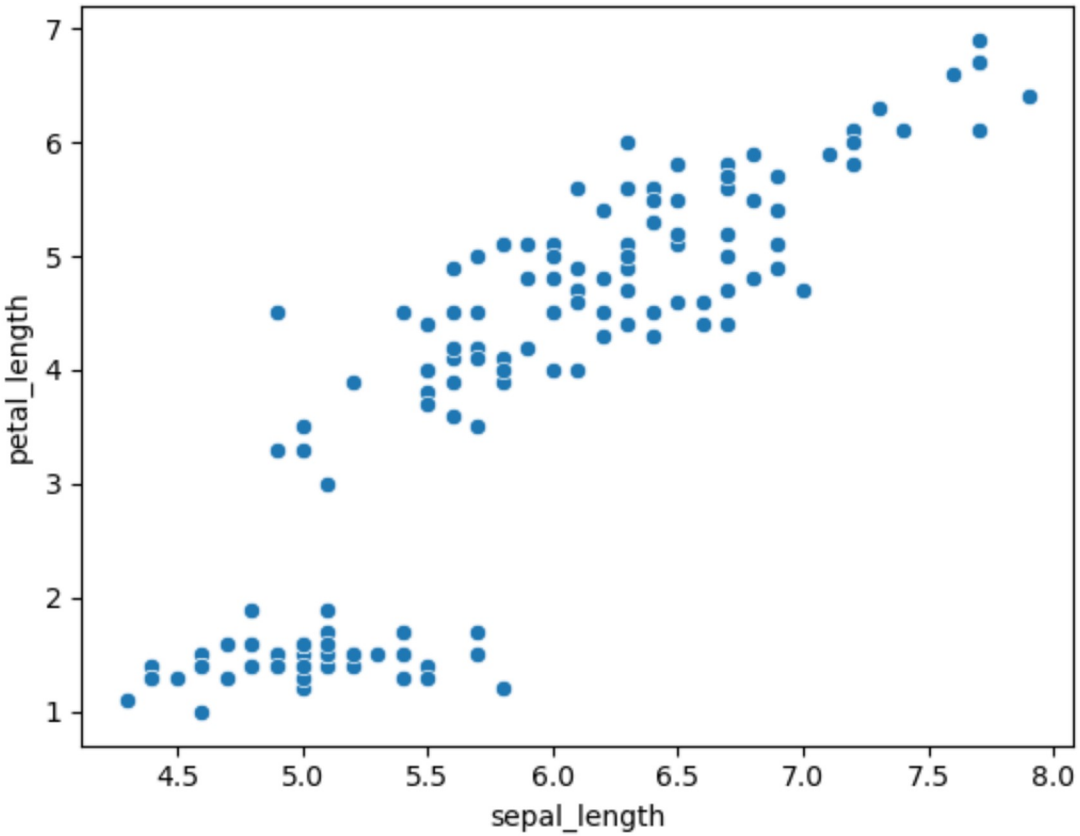
	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

2.Clustering Algorithm Implementation

In [8]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

In [9]:
```
sns.scatterplot(x='sepal_length',y='petal_length',data=data)
plt.show()
```



In [12]:
```
label = kmeans.labels_
label
```

Out[12]:
```
array([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2,
       2, 2, 2, 2, 2, 2, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0])
```

In [13]:
```
# silhoutte score measure how well the cluster is sperated
from sklearn.metrics import silhouette_score

silh = silhouette_score(scaled_features , label)
```

```
print('silhoutte score',silh)
```

```
silhoutte score 0.4565352255831264
```

In [14]:
```
# ELBOW METHOD

inertia=[]
k_values = range(1,10)

for i in k_values:
    kmeans = KMeans(n_clusters=i)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)
```

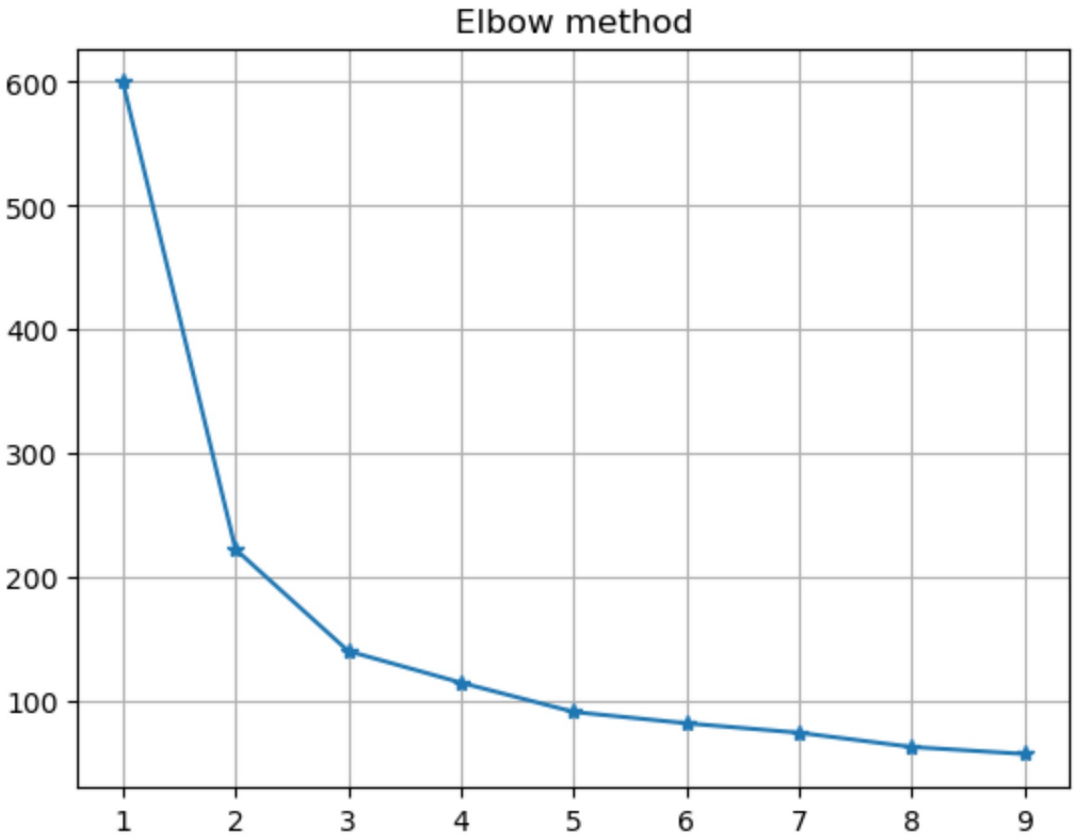
```
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [15]:

```
inertia
```

Out[15]:
```
[599.9999999999999,
 222.36170496502297,
 140.0820210962166,
 114.55684421262914,
 90.92751382392046,
 81.7502677331235,
 74.10782307487395,
 62.60128344279277,
 56.962380942000834]
```

In [16]:
```
plt.plot(k_values , inertia, marker='*')
plt.title('Elbow method')
plt.grid(True)
plt.show()
```



K-MEANS CLUSTERING

K-Mean is a Unsupervised learning algorithm ,Clustering is which the data is grouped according to the similarity of the data .The nearest mean value of dataset makes a group/clusters. As the result similar data points are grouped together.

In [17]:
```
from sklearn.cluster import KMeans

kmeans=KMeans(n_clusters=3)
data['Cluster'] = kmeans.fit_predict(scaled_features)
data
```

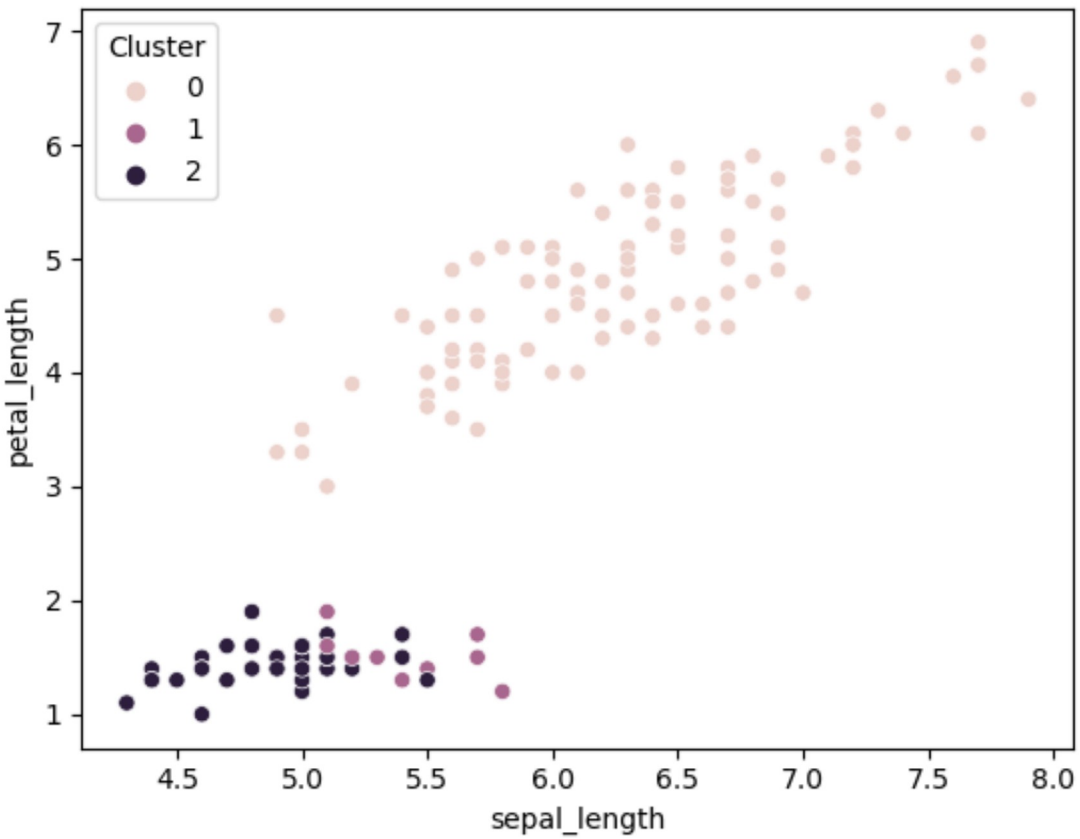
```
C:\Users\Acer\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1429: UserWarning: KMeans is known to have a mem
ory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the envi
ronment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[17]:

	sepal_length	sepal_width	petal_length	petal_width	species	Cluster
0	5.1	3.5	1.4	0.2	setosa	2
1	4.9	3.0	1.4	0.2	setosa	2
2	4.7	3.2	1.3	0.2	setosa	2
3	4.6	3.1	1.5	0.2	setosa	2
4	5.0	3.6	1.4	0.2	setosa	2
...	...	...	...	...	...	...
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 [18]:
```
sns.scatterplot(x='sepal_length',y='petal_length',data=data , hue=data['Cluster'])
plt.show()
```



K-Means Clustering is suitable for the dataset "IRIS" because it group the data very effectively based on the measurements and data can be easily understand beacuse of the simplicity and the speed of algorithm . it makes the analysis very efficient and easy visualization

HIERARCHICAL CLUSTERING

Hierarchical clustering ia a Unsupervised learning algorithm , It is grouped according to the similaries of data in the form of hierarchy , were data visualized as Dendrogram.

In [19]:
```
from sklearn.cluster import AgglomerativeClustering
```

```
Ag = AgglomerativeClustering(n_clusters=3)
data['Cluster'] = Ag.fit_predict(scaled_features)
data
```

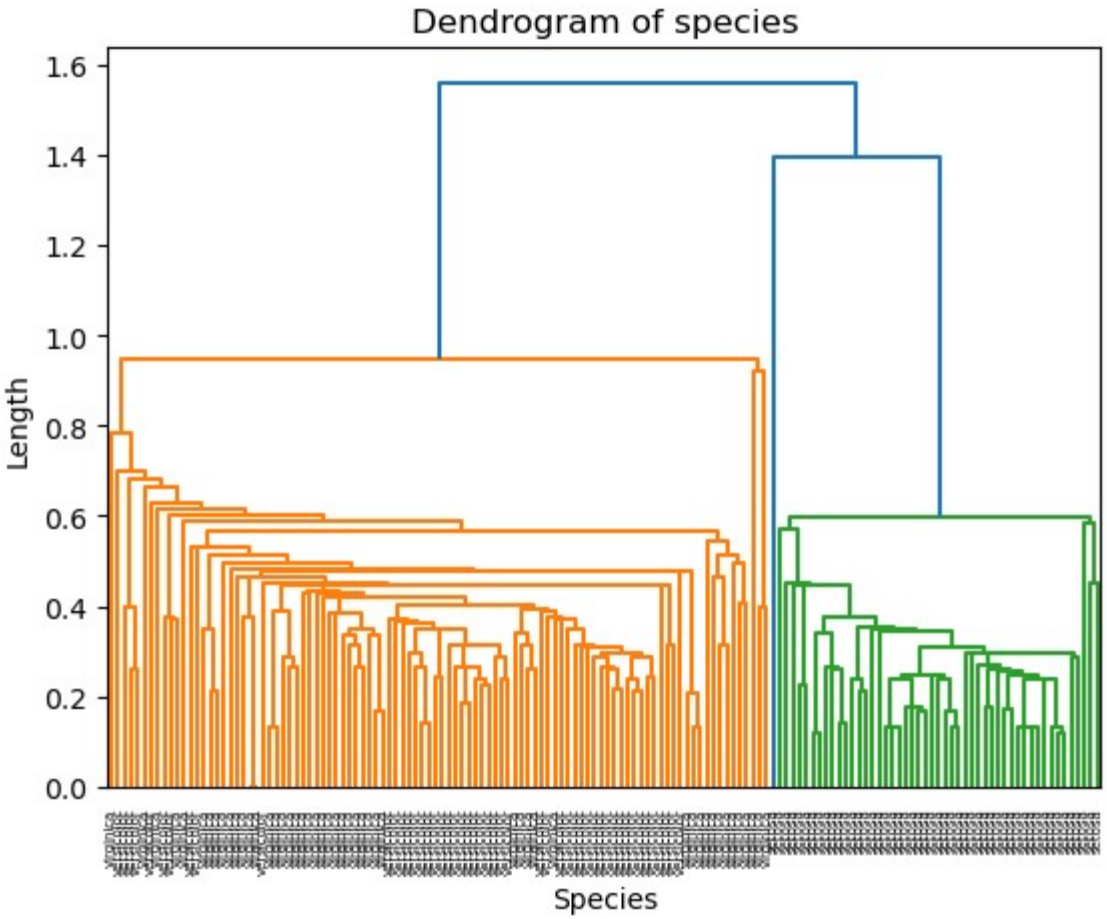
Out[19]:

	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 [20]:
```
from scipy.cluster.hierarchy import dendrogram , linkage
h=linkage(scaled_features)
label=data['species'].tolist()
```

In [21]:
```
dendrogram(h,labels=label)
plt.title('Dendrogram of species')
plt.xlabel('Species')
plt.ylabel('Length')
plt.show()
```



Hierarchical Clustering is suitable for dataset named Iris because it helps identify natural groupings based on the measurements.Using the Dendrogram it hepls to understand the relationships of the data and reveal meaningful patterns.