ML:Iris(CLUSTERING)

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 ¶

Data Exploration:

In [2]:
```
# importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
# Load the Iris dataset
from sklearn import datasets
iris = datasets.load_iris()
df=pd.DataFrame(data=iris.data,columns=iris.feature_names)
df
```

Out[3]:

	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

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
dtypes: float64(4)
memory usage: 4.8 KB
```

In [5]:
```
# checking duplicates
df.duplicated().sum()
```

Out[5]:
```
1
```

In [6]:
```
# remove duplicates
df=df.drop_duplicates()
```

In [7]:
```

df.duplicated().sum()
```

Out[7]:
```
0
```

In [8]:
```
# missing values
df.isnull().sum()
```

Out[8]:
```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

In [9]:
```
df.describe()
```

Out[9]:

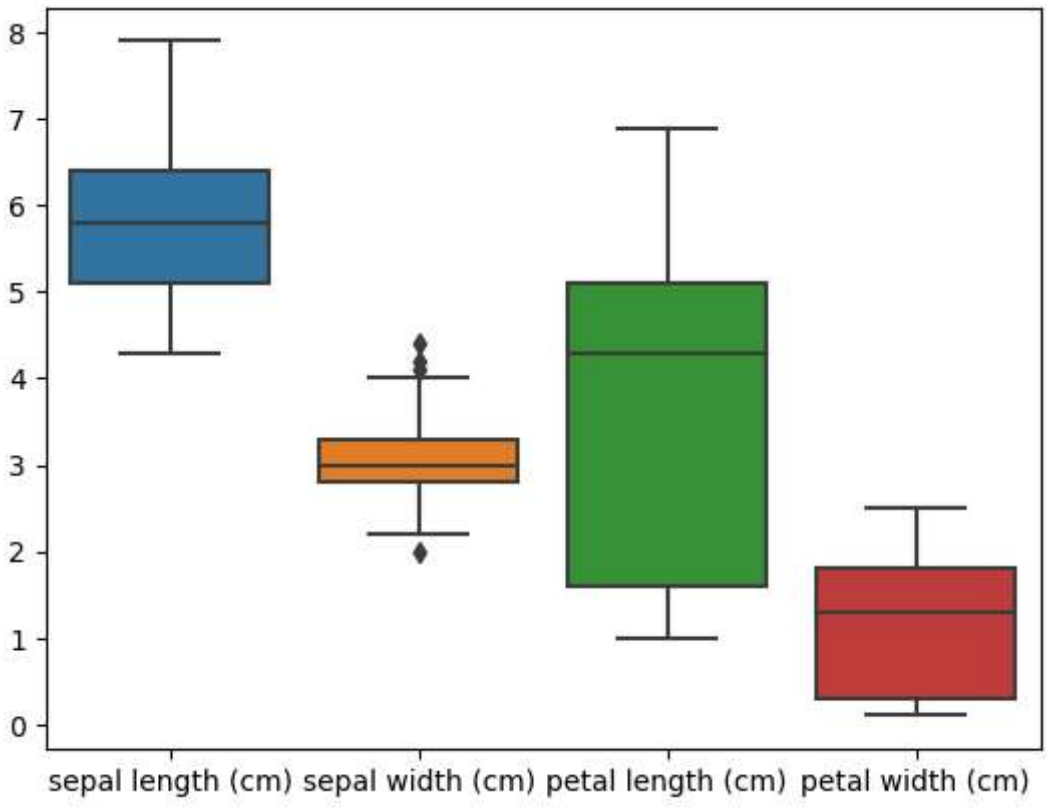
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
count	149.000000	149.000000	149.000000	149.000000
mean	5.843624	3.059732	3.748993	1.194631
std	0.830851	0.436342	1.767791	0.762622
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.300000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [10]:
```
#find outliers
df.skew()
```

Out[10]:
```
sepal length (cm)    0.312826
sepal width (cm)     0.307149
petal length (cm)   -0.263101
petal width (cm)    -0.090076
dtype: float64
```

In [11]:
```
# To plot for finding the outlier column
sns.boxplot(df)
```

Out[11]: <Axes: >



In [12]:
```
# Remove the outlier all columns by using functions of IQR method
def remove_outliers(df,columns):
    df_filtered = df.copy()

    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1

        lower_whisker = Q1 - 1.5 * IQR
        upper_whisker = Q3 + 1.5 * IQR

        df_filtered = df_filtered[(df_filtered[col] <= upper_whisker) & (df_fil
    return df_filtered
```

In [13]:
```
dff=remove_outliers(df,["sepal width (cm)"])
dff
```

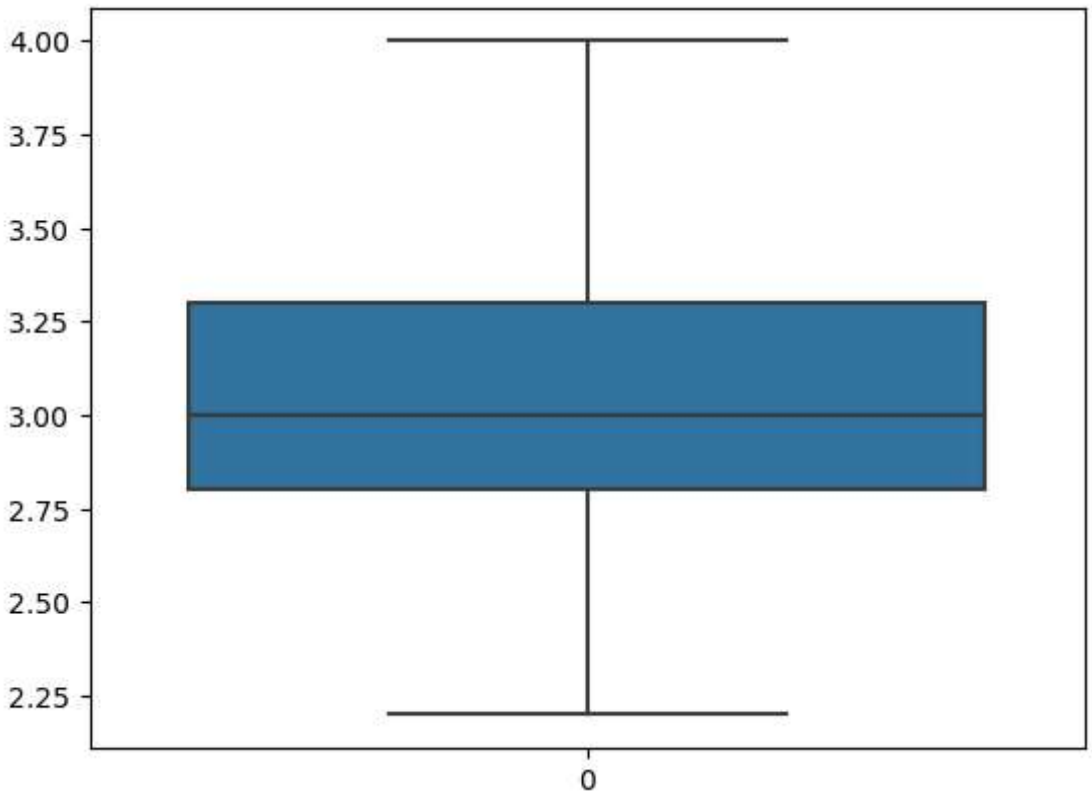
Out[13]:

	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

145 rows × 4 columns

In [14]:
```
# After removed the outlier column
sns.boxplot(dff['sepal width (cm)'])
```

Out[14]: <Axes: >



In [15]:
```
dff['sepal width (cm)'].skew() # dff is the cleaned data
```

Out[15]:
```
0.11859094247253242
```

In [16]:
```
# data scaling
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaled_features =scaler.fit_transform(dff)
scaled_features # scaled data
```

Out[16]:
```
array([[-9.07876527e-01,  1.15220566e+00, -1.36654787e+00,
        -1.34146218e+00],
       [-1.14766177e+00, -1.07747739e-01, -1.36654787e+00,
        -1.34146218e+00],
       [-1.38744701e+00,  3.96233620e-01, -1.42353650e+00,
        -1.34146218e+00],
       [-1.50733963e+00,  1.44242940e-01, -1.30955925e+00,
        -1.34146218e+00],
       [-1.02776915e+00,  1.40419634e+00, -1.36654787e+00,
        -1.34146218e+00],
       [-5.48198668e-01,  2.16016837e+00, -1.19558200e+00,
        -1.07735897e+00],
       [-1.50733963e+00,  9.00214978e-01, -1.36654787e+00,
        -1.20941058e+00],
       [-1.02776915e+00,  9.00214978e-01, -1.30955925e+00,
        -1.34146218e+00],
       [-1.74712487e+00, -3.59738418e-01, -1.36654787e+00,
        -1.34146218e+00],
       [-1.14766177e+00,  1.44242940e-01, -1.30955925e+00,
        -1.47351379e+00],
       [-5.48198668e-01,  1.65618701e+00, -1.30955925e+00,
        -1.34146218e+00],
       [-1.26755439e+00,  9.00214978e-01, -1.25257063e+00,
        -1.34146218e+00],
       [-1.26755439e+00, -1.07747739e-01, -1.36654787e+00,
        -1.47351379e+00],
       [-1.86701749e+00, -1.07747739e-01, -1.53751374e+00,
        -1.47351379e+00],
       [-6.86281892e-02,  2.41215905e+00, -1.48052512e+00,
        -1.34146218e+00],
       [-5.48198668e-01,  2.16016837e+00, -1.42353650e+00,
        -1.07735897e+00],
       [-9.07876527e-01,  1.15220566e+00, -1.36654787e+00,
        -1.20941058e+00],
       [-1.88520809e-01,  1.90817769e+00, -1.19558200e+00,
        -1.20941058e+00],
       [-9.07876527e-01,  1.90817769e+00, -1.30955925e+00,
        -1.20941058e+00],
       [-5.48198668e-01,  9.00214978e-01, -1.19558200e+00,
        -1.34146218e+00],
       [-9.07876527e-01,  1.65618701e+00, -1.30955925e+00,
        -1.07735897e+00],
       [-1.50733963e+00,  1.40419634e+00, -1.59450236e+00,
        -1.34146218e+00],
       [-9.07876527e-01,  6.48224299e-01, -1.19558200e+00,
        -9.45307363e-01],
       [-1.26755439e+00,  9.00214978e-01, -1.08160476e+00,
        -1.34146218e+00],
       [-1.02776915e+00, -1.07747739e-01, -1.25257063e+00,
        -1.34146218e+00],
       [-1.02776915e+00,  9.00214978e-01, -1.25257063e+00,
        -1.07735897e+00],
       [-7.87983908e-01,  1.15220566e+00, -1.30955925e+00,
        -1.34146218e+00],
       [-7.87983908e-01,  9.00214978e-01, -1.36654787e+00,
        -1.34146218e+00],
       [-1.38744701e+00,  3.96233620e-01, -1.25257063e+00,
```

```
 -1.34146218e+00],
[-1.26755439e+00,  1.44242940e-01, -1.25257063e+00,
 -1.34146218e+00],
[-5.48198668e-01,  9.00214978e-01, -1.30955925e+00,
 -1.07735897e+00],
[-1.14766177e+00,  1.44242940e-01, -1.30955925e+00,
 -1.34146218e+00],
[-1.02776915e+00,  3.96233620e-01, -1.48052512e+00,
 -1.34146218e+00],
[-4.28306048e-01,  1.15220566e+00, -1.42353650e+00,
 -1.34146218e+00],
[-1.14766177e+00,  1.40419634e+00, -1.36654787e+00,
 -1.47351379e+00],
[-1.74712487e+00, -1.07747739e-01, -1.42353650e+00,
 -1.34146218e+00],
[-9.07876527e-01,  9.00214978e-01, -1.30955925e+00,
 -1.34146218e+00],
[-1.02776915e+00,  1.15220566e+00, -1.42353650e+00,
 -1.20941058e+00],
[-1.62723225e+00, -1.87168249e+00, -1.42353650e+00,
 -1.20941058e+00],
[-1.74712487e+00,  3.96233620e-01, -1.42353650e+00,
 -1.34146218e+00],
[-1.02776915e+00,  1.15220566e+00, -1.25257063e+00,
 -8.13255756e-01],
[-9.07876527e-01,  1.90817769e+00, -1.08160476e+00,
 -1.07735897e+00],
[-1.26755439e+00, -1.07747739e-01, -1.36654787e+00,
 -1.20941058e+00],
[-9.07876527e-01,  1.90817769e+00, -1.25257063e+00,
 -1.34146218e+00],
[-1.50733963e+00,  3.96233620e-01, -1.36654787e+00,
 -1.34146218e+00],
[-6.68091288e-01,  1.65618701e+00, -1.30955925e+00,
 -1.34146218e+00],
[-1.02776915e+00,  6.48224299e-01, -1.36654787e+00,
 -1.34146218e+00],
[ 1.37008325e+00,  3.96233620e-01,  5.14076680e-01,
  2.43157096e-01],
[ 6.50727529e-01,  3.96233620e-01,  4.00099435e-01,
  3.75208703e-01],
[ 1.25019063e+00,  1.44242940e-01,  6.28053926e-01,
  3.75208703e-01],
[-4.28306048e-01, -1.87168249e+00,  1.15156321e-01,
  1.11105490e-01],
[ 7.70620149e-01, -6.11729097e-01,  4.57088057e-01,
  3.75208703e-01],
[-1.88520809e-01, -6.11729097e-01,  4.00099435e-01,
  1.11105490e-01],
[ 5.30834909e-01,  6.48224299e-01,  5.14076680e-01,
  5.07260309e-01],
[-1.14766177e+00, -1.61969181e+00, -2.83764039e-01,
 -2.85049330e-01],
[ 8.90512769e-01, -3.59738418e-01,  4.57088057e-01,
  1.11105490e-01],
[-7.87983908e-01, -8.63719776e-01,  5.81676978e-02,
  2.43157096e-01],
```

```
[ 5.12644305e-02, -1.07747739e-01,  2.29133566e-01,
  3.75208703e-01],
[ 1.71157050e-01, -2.12367317e+00,  1.15156321e-01,
 -2.85049330e-01],
[ 2.91049670e-01, -3.59738418e-01,  5.14076680e-01,
  2.43157096e-01],
[-3.08413429e-01, -3.59738418e-01, -1.12798171e-01,
  1.11105490e-01],
[ 1.01040539e+00,  1.44242940e-01,  3.43110812e-01,
  2.43157096e-01],
[-3.08413429e-01, -1.07747739e-01,  4.00099435e-01,
  3.75208703e-01],
[-6.86281892e-02, -8.63719776e-01,  1.72144943e-01,
 -2.85049330e-01],
[ 4.10942290e-01, -2.12367317e+00,  4.00099435e-01,
  3.75208703e-01],
[-3.08413429e-01, -1.36770113e+00,  5.81676978e-02,
 -1.52997723e-01],
[ 5.12644305e-02,  3.96233620e-01,  5.71065303e-01,
  7.71363522e-01],
[ 2.91049670e-01, -6.11729097e-01,  1.15156321e-01,
  1.11105490e-01],
[ 5.30834909e-01, -1.36770113e+00,  6.28053926e-01,
  3.75208703e-01],
[ 2.91049670e-01, -6.11729097e-01,  5.14076680e-01,
 -2.09461169e-02],
[ 6.50727529e-01, -3.59738418e-01,  2.86122189e-01,
  1.11105490e-01],
[ 8.90512769e-01, -1.07747739e-01,  3.43110812e-01,
  2.43157096e-01],
[ 1.13029801e+00, -6.11729097e-01,  5.71065303e-01,
  2.43157096e-01],
[ 1.01040539e+00, -1.07747739e-01,  6.85042549e-01,
  6.39311916e-01],
[ 1.71157050e-01, -3.59738418e-01,  4.00099435e-01,
  3.75208703e-01],
[-1.88520809e-01, -1.11571045e+00, -1.69786793e-01,
 -2.85049330e-01],
[-4.28306048e-01, -1.61969181e+00,  1.17907495e-03,
 -1.52997723e-01],
[-4.28306048e-01, -1.61969181e+00, -5.58095478e-02,
 -2.85049330e-01],
[-6.86281892e-02, -8.63719776e-01,  5.81676978e-02,
 -2.09461169e-02],
[ 1.71157050e-01, -8.63719776e-01,  7.42031171e-01,
  5.07260309e-01],
[-5.48198668e-01, -1.07747739e-01,  4.00099435e-01,
  3.75208703e-01],
[ 1.71157050e-01,  9.00214978e-01,  4.00099435e-01,
  5.07260309e-01],
[ 1.01040539e+00,  1.44242940e-01,  5.14076680e-01,
  3.75208703e-01],
[ 5.30834909e-01, -1.87168249e+00,  3.43110812e-01,
  1.11105490e-01],
[-3.08413429e-01, -1.07747739e-01,  1.72144943e-01,
  1.11105490e-01],
[-4.28306048e-01, -1.36770113e+00,  1.15156321e-01,
```

```
  1.11105490e-01],
[-4.28306048e-01, -1.11571045e+00,  3.43110812e-01,
 -2.09461169e-02],
[ 2.91049670e-01, -1.07747739e-01,  4.57088057e-01,
  2.43157096e-01],
[-6.86281892e-02, -1.11571045e+00,  1.15156321e-01,
 -2.09461169e-02],
[-1.02776915e+00, -1.87168249e+00, -2.83764039e-01,
 -2.85049330e-01],
[-3.08413429e-01, -8.63719776e-01,  2.29133566e-01,
  1.11105490e-01],
[-1.88520809e-01, -1.07747739e-01,  2.29133566e-01,
 -2.09461169e-02],
[-1.88520809e-01, -3.59738418e-01,  2.29133566e-01,
  1.11105490e-01],
[ 4.10942290e-01, -3.59738418e-01,  2.86122189e-01,
  1.11105490e-01],
[-9.07876527e-01, -1.36770113e+00, -4.54729907e-01,
 -1.52997723e-01],
[-1.88520809e-01, -6.11729097e-01,  1.72144943e-01,
  1.11105490e-01],
[ 5.30834909e-01,  6.48224299e-01,  1.25492878e+00,
  1.69572477e+00],
[-6.86281892e-02, -8.63719776e-01,  7.42031171e-01,
  9.03415129e-01],
[ 1.48997587e+00, -1.07747739e-01,  1.19794015e+00,
  1.16751834e+00],
[ 5.30834909e-01, -3.59738418e-01,  1.02697429e+00,
  7.71363522e-01],
[ 7.70620149e-01, -1.07747739e-01,  1.14095153e+00,
  1.29956995e+00],
[ 2.08943897e+00, -1.07747739e-01,  1.59686051e+00,
  1.16751834e+00],
[-1.14766177e+00, -1.36770113e+00,  4.00099435e-01,
  6.39311916e-01],
[ 1.72976111e+00, -3.59738418e-01,  1.42589464e+00,
  7.71363522e-01],
[ 1.01040539e+00, -1.36770113e+00,  1.14095153e+00,
  7.71363522e-01],
[ 1.60986849e+00,  1.40419634e+00,  1.31191740e+00,
  1.69572477e+00],
[ 7.70620149e-01,  3.96233620e-01,  7.42031171e-01,
  1.03546674e+00],
[ 6.50727529e-01, -8.63719776e-01,  8.56008417e-01,
  9.03415129e-01],
[ 1.13029801e+00, -1.07747739e-01,  9.69985663e-01,
  1.16751834e+00],
[-1.88520809e-01, -1.36770113e+00,  6.85042549e-01,
  1.03546674e+00],
[-6.86281892e-02, -6.11729097e-01,  7.42031171e-01,
  1.56367316e+00],
[ 6.50727529e-01,  3.96233620e-01,  8.56008417e-01,
  1.43162156e+00],
[ 7.70620149e-01, -1.07747739e-01,  9.69985663e-01,
  7.71363522e-01],
[ 2.20933159e+00,  1.90817769e+00,  1.65384914e+00,
  1.29956995e+00],
```

```
[ 2.20933159e+00, -1.11571045e+00,  1.76782638e+00,
  1.43162156e+00],
[ 1.71157050e-01, -2.12367317e+00,  6.85042549e-01,
  3.75208703e-01],
[ 1.25019063e+00,  3.96233620e-01,  1.08396291e+00,
  1.43162156e+00],
[-3.08413429e-01, -6.11729097e-01,  6.28053926e-01,
  1.03546674e+00],
[ 2.20933159e+00, -6.11729097e-01,  1.65384914e+00,
  1.03546674e+00],
[ 5.30834909e-01, -8.63719776e-01,  6.28053926e-01,
  7.71363522e-01],
[ 1.01040539e+00,  6.48224299e-01,  1.08396291e+00,
  1.16751834e+00],
[ 1.60986849e+00,  3.96233620e-01,  1.25492878e+00,
  7.71363522e-01],
[ 4.10942290e-01, -6.11729097e-01,  5.71065303e-01,
  7.71363522e-01],
[ 2.91049670e-01, -1.07747739e-01,  6.28053926e-01,
  7.71363522e-01],
[ 6.50727529e-01, -6.11729097e-01,  1.02697429e+00,
  1.16751834e+00],
[ 1.60986849e+00, -1.07747739e-01,  1.14095153e+00,
  5.07260309e-01],
[ 1.84965373e+00, -6.11729097e-01,  1.31191740e+00,
  9.03415129e-01],
[ 2.44911683e+00,  1.90817769e+00,  1.48288327e+00,
  1.03546674e+00],
[ 6.50727529e-01, -6.11729097e-01,  1.02697429e+00,
  1.29956995e+00],
[ 5.30834909e-01, -6.11729097e-01,  7.42031171e-01,
  3.75208703e-01],
[ 2.91049670e-01, -1.11571045e+00,  1.02697429e+00,
  2.43157096e-01],
[ 2.20933159e+00, -1.07747739e-01,  1.31191740e+00,
  1.43162156e+00],
[ 5.30834909e-01,  9.00214978e-01,  1.02697429e+00,
  1.56367316e+00],
[ 6.50727529e-01,  1.44242940e-01,  9.69985663e-01,
  7.71363522e-01],
[ 1.71157050e-01, -1.07747739e-01,  5.71065303e-01,
  7.71363522e-01],
[ 1.25019063e+00,  1.44242940e-01,  9.12997040e-01,
  1.16751834e+00],
[ 1.01040539e+00,  1.44242940e-01,  1.02697429e+00,
  1.56367316e+00],
[ 1.25019063e+00,  1.44242940e-01,  7.42031171e-01,
  1.43162156e+00],
[ 1.13029801e+00,  3.96233620e-01,  1.19794015e+00,
  1.43162156e+00],
[ 1.01040539e+00,  6.48224299e-01,  1.08396291e+00,
  1.69572477e+00],
[ 1.01040539e+00, -1.07747739e-01,  7.99019794e-01,
  1.43162156e+00],
[ 5.30834909e-01, -1.36770113e+00,  6.85042549e-01,
  9.03415129e-01],
[ 7.70620149e-01, -1.07747739e-01,  7.99019794e-01,
```

```
  1.03546674e+00],
[ 4.10942290e-01,  9.00214978e-01,  9.12997040e-01,
  1.43162156e+00],
[ 5.12644305e-02, -1.07747739e-01,  7.42031171e-01,
  7.71363522e-01]])
```

2.Clustering Algorithm Implementation

A) KMeans Clustering

KMeans clustering is an unsupervised ML algorithm used to group a dataset into k clusters.It is an iterative algorithm that starts by randomly selecting k centroids,the entire dataset is divided into clusters based on the distance of the data points from the centroid.In the new clusters,the centroids are calculated by taking the mean of the data points.

1. First,we will select k random entries from the dataset and use them as centroids.
2. Now,we will find the distance of each entry in the dataset from the centroids.We can use any distance metric such as euclidean distance,Manhattan distance, or squared euclidean distance.
3. after finding the distance of each data entry from the centroids,we will start assigning the data points to clusters.We will assign each data point to the cluster with the centroid to which it has the least distance.
4. After assigning the points to the clusters,we will calculate the new centroid of the clusters.For this ,we will use the mean or each data point in the same cluster as the new centroid.If the newly created centroids are the same as the centroids in the previous iteration,we will consider the current clusters to be final.Hence,we will stop the execution of the algorithm.

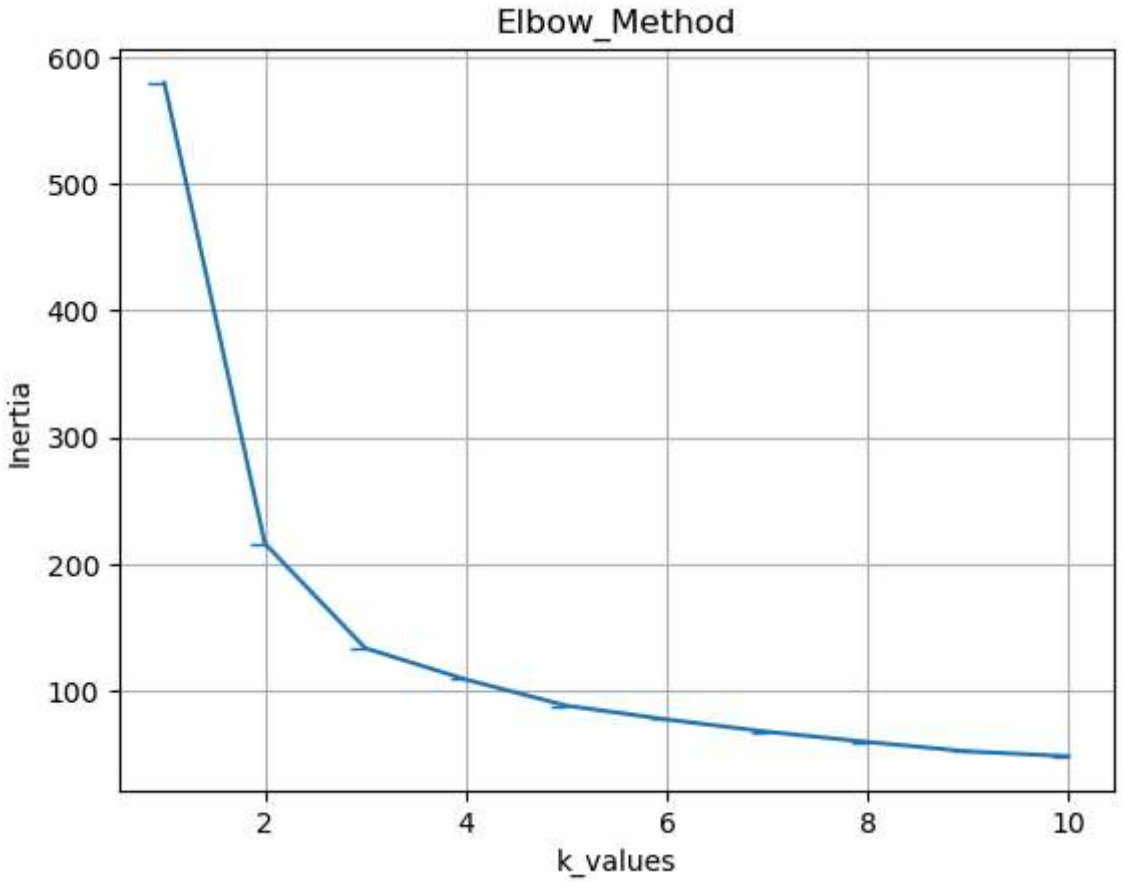
In [17]:
```
# To implement KMeans clustering
from sklearn.cluster import KMeans
inertia=[]
k_values = range(1,11)
for k in k_values:
    kmeans = KMeans(n_clusters=k)
    kmeans.fit(scaled_features)
    inertia.append(kmeans.inertia_)
```

```
C:\Users\PWORLD\anaconda3\thesli.anacnda.entri\Lib\site-packages\sklearn\cl
uster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\PWORLD\anaconda3\thesli.anacnda.entri\Lib\site-packages\sklearn\cl
uster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You
can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\PWORLD\anaconda3\thesli.anacnda.entri\Lib\site-packages\sklearn\cl
uster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will ch
ange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to supp
ress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\PWORLD\anaconda3\thesli.anacnda.entri\Lib\site-packages\sklearn\cl
uster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak o
n Windows with MKL, when there are less chunks than available threads. You
can avoid it by setting the environment variable OMP_NUM_THREADS=1.
```

In [18]:
```
inertia
```

Out[18]:
```
[580.0,
 216.12430203348623,
 133.38341555248672,
 108.64866652278866,
 87.9948137336724,
 77.07189033760116,
 67.35958940626162,
 59.266523730967236,
 51.843482354773016,
 48.018539406095655]
```

In [19]:
```
# plot the Elbow Curve
plt.plot(k_values,inertia,marker=0)
plt.xlabel('k_values')
plt.ylabel('Inertia')
plt.title('Elbow_Method')
plt.grid(True)
plt.show()
```



In [20]:
```
kmeans = KMeans(n_clusters=3)
dff['cluster']= kmeans.fit_predict(scaled_features)
```

```
C:\Users\PWORLD\anaconda3\thesli.anacnda.entri\Lib\site-packages\sklearn\clus
ter\_kmeans.py:1412: FutureWarning: The default value of `n_init` will change
from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress th
e warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\PWORLD\anaconda3\thesli.anacnda.entri\Lib\site-packages\sklearn\clus
ter\_kmeans.py:1436: UserWarning: KMeans is known to have a memory leak on Wi
ndows with MKL, when there are less chunks than available threads. You can av
oid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [21]:
```
dff
```

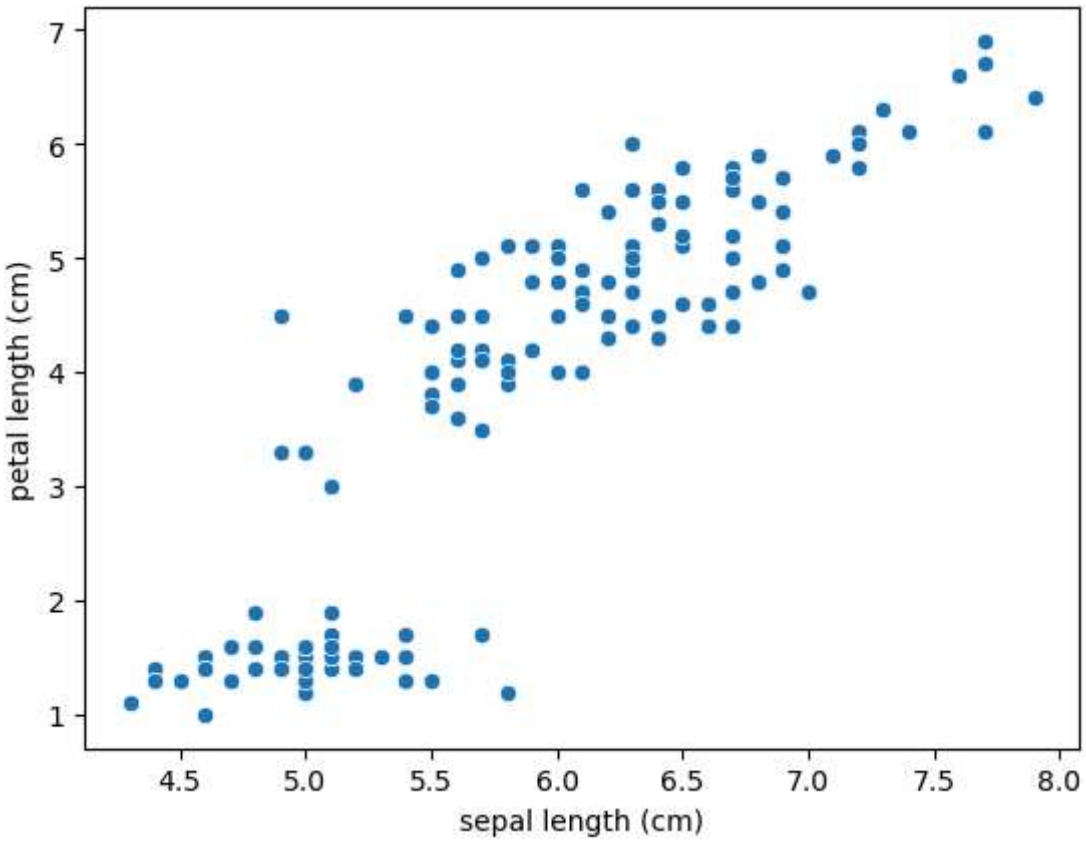
Out[21]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	cluster
0	5.1	3.5	1.4	0.2	1
1	4.9	3.0	1.4	0.2	1
2	4.7	3.2	1.3	0.2	1
3	4.6	3.1	1.5	0.2	1
4	5.0	3.6	1.4	0.2	1
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	0
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	0
148	6.2	3.4	5.4	2.3	0
149	5.9	3.0	5.1	1.8	2

145 rows × 5 columns

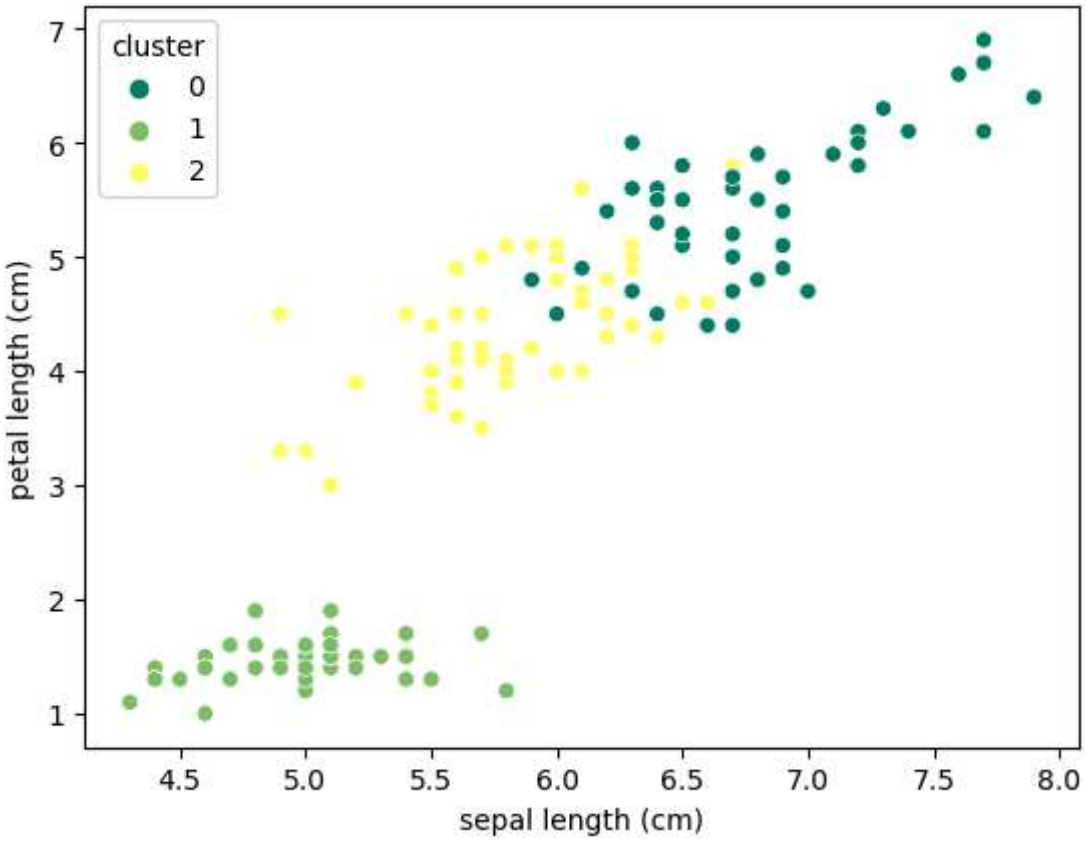
In [22]:
```
sns.scatterplot(x=dff['sepal length (cm)'],y=dff['petal length (cm)'])
```

Out[22]:
```
<Axes: xlabel='sepal length (cm)', ylabel='petal length (cm)'>
```



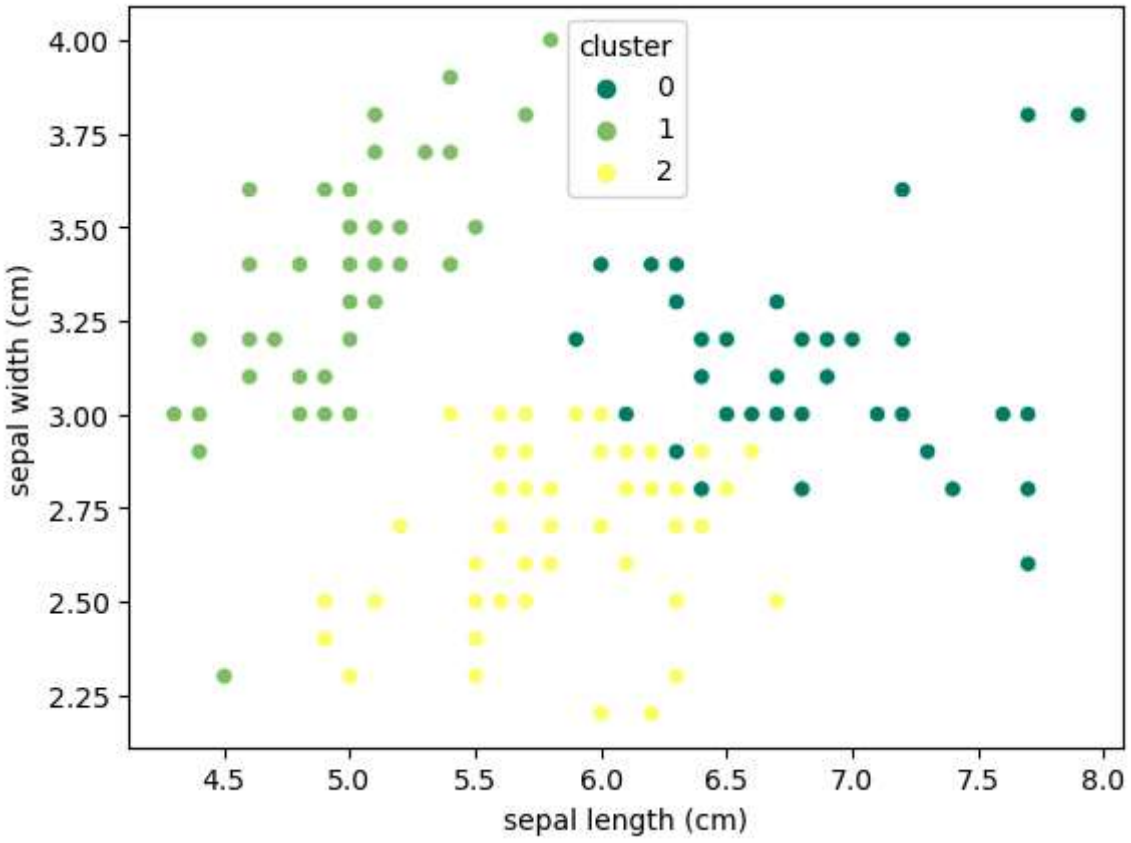
In [23]:
```
sns.scatterplot(x=dff['sepal length (cm)'],y=dff['petal length (cm)'],hue=dff['
```

Out[23]:
```
<Axes: xlabel='sepal length (cm)', ylabel='petal length (cm)'>
```



In [24]:
```
sns.scatterplot(x=dff['sepal length (cm)'],y=dff['sepal width (cm)'],hue=dff['
```

Out[24]:
```
<Axes: xlabel='sepal length (cm)', ylabel='sepal width (cm)'>
```



In [25]:
```
kmeans.labels_
```

Out[25]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
       0, 2, 2, 2, 2, 0, 0, 0, 2, 2, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0, 0, 2, 0, 2, 0, 0, 2, 0, 2,
       2, 0, 0, 0, 0, 2, 0, 2, 0, 2, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 2, 0,
       0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 2])
```

In [28]:
```
from sklearn.metrics import silhouette_score
silhouette_score(scaled_features,kmeans.labels_)
```

Out[28]:
```
0.4649687421839816
```

In this dataset,kmeans clustering is good result.it is easy to implement.Because,kmeans clustering is an iterable algorithm and a relatively simple algorithm. We can use kmeans clustering for even 10 records or even 10 million records in adataset.It will give us results in both cases.

B) Hierarchical Clustering

Hierarchical agglomerative clustering is a clustering method that groups items into clusters based on similarities.It's a bottom-up approach that starts with each item as a singleton cluster and merges paires of clusters until all items are in one large cluster.

1.Compute a proximity matrix using a distance metric. 2.Use a linkage function to group objects into a hierarchical cluster tree based on the distance matrix. 3.Merge data points with close proximity to form a cluster . 4.Repeat steps 2 and 3 until asingle cluster remains.

In [30]:
```
# To implement the Agglomerative clustering
from sklearn.cluster import AgglomerativeClustering
model =AgglomerativeClustering(n_clusters = 3,affinity='euclidean',linkage='war
model.fit(scaled_features)
```

```
C:\Users\PWORLD\anaconda3\thesli.anacnda.entri\Lib\site-packages\sklearn\clus
ter\_agglomerative.py:1005: FutureWarning: Attribute `affinity` was deprecate
d in version 1.2 and will be removed in 1.4. Use `metric` instead
  warnings.warn(
```

Out[30]:
```
AgglomerativeClustering

AgglomerativeClustering(affinity='euclidean', n_clusters=3)
```

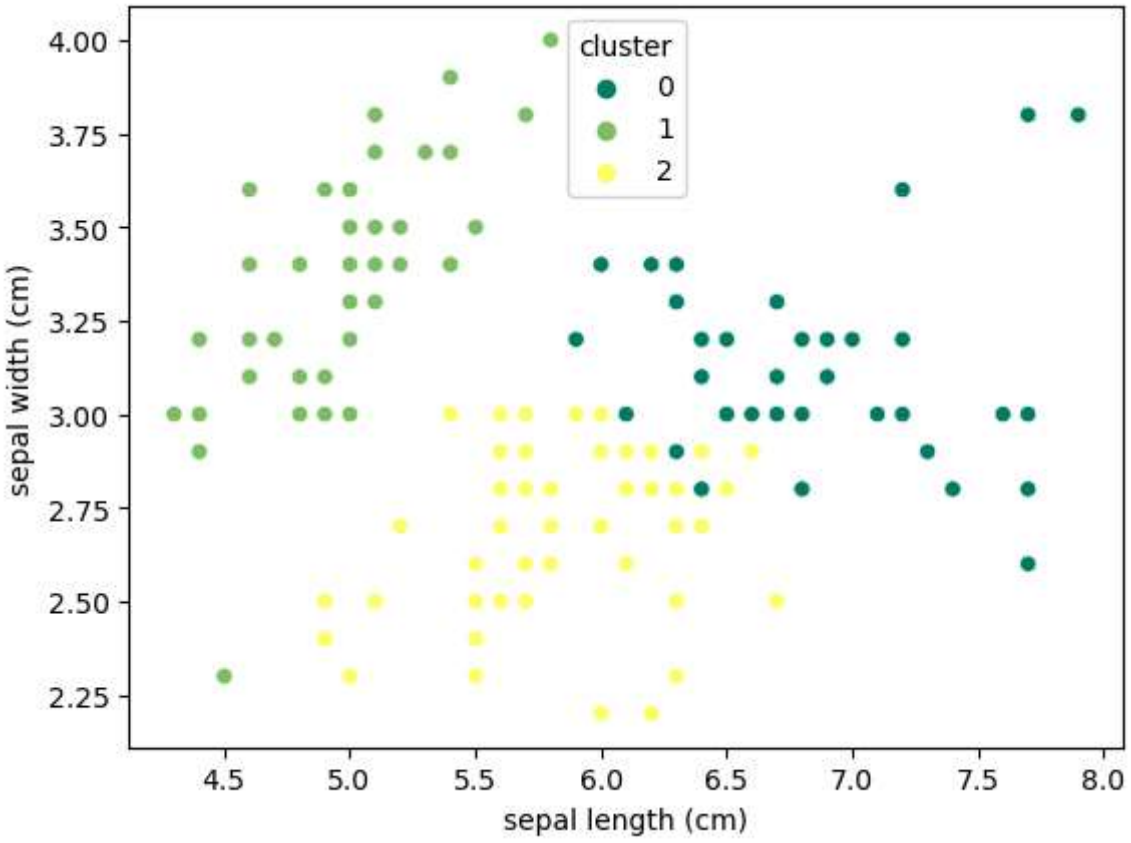
In [31]:
```
dff['clusters']= model.fit_predict(scaled_features)
scaled_features
```

```
C:\Users\PWORLD\anaconda3\thesli.anacnda.entri\Lib\site-packages\sklearn\cl
uster\_agglomerative.py:1005: FutureWarning: Attribute `affinity` was depre
cated in version 1.2 and will be removed in 1.4. Use `metric` instead
  warnings.warn(
```

Out[31]:
```
array([[-9.07876527e-01,  1.15220566e+00, -1.36654787e+00,
        -1.34146218e+00],
       [-1.14766177e+00, -1.07747739e-01, -1.36654787e+00,
        -1.34146218e+00],
       [-1.38744701e+00,  3.96233620e-01, -1.42353650e+00,
        -1.34146218e+00],
       [-1.50733963e+00,  1.44242940e-01, -1.30955925e+00,
        -1.34146218e+00],
       [-1.02776915e+00,  1.40419634e+00, -1.36654787e+00,
        -1.34146218e+00],
       [-5.48198668e-01,  2.16016837e+00, -1.19558200e+00,
        -1.07735897e+00],
       [-1.50733963e+00,  9.00214978e-01, -1.36654787e+00,
        -1.20941058e+00],
       [-1.02776915e+00,  9.00214978e-01, -1.30955925e+00
```

In [32]:
```
sns.scatterplot(x=dff['sepal length (cm)'],y=dff['sepal width (cm)'],hue=dff['
```

Out[32]: <Axes: xlabel='sepal length (cm)', ylabel='sepal width (cm)'>

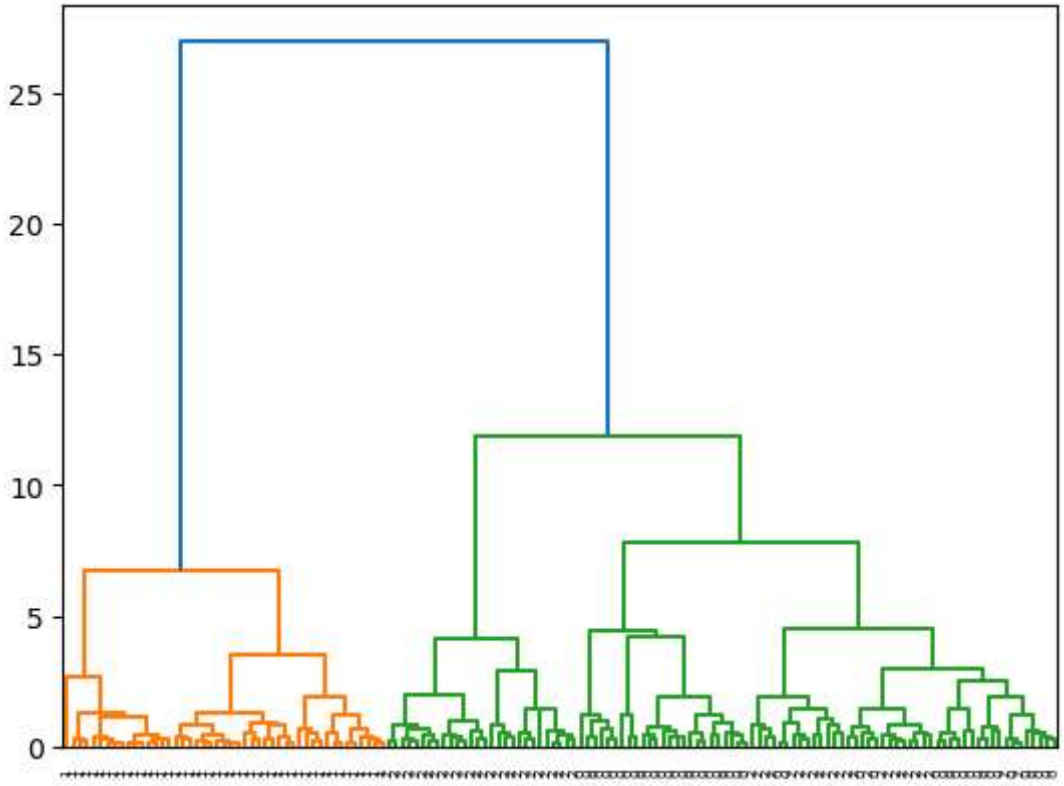


In [33]:
```
from scipy.cluster.hierarchy import dendrogram,linkage
# Perform hierarchical clustering
z= linkage(scaled_features,method='ward')
z
```

Out[33]:
```
array([[7.00000000e+00, 3.60000000e+01, 1.19892620e-01, 2.00000000e+00],
       [1.00000000e+01, 4.50000000e+01, 1.19892620e-01, 2.00000000e+00],
       [9.00000000e+00, 3.10000000e+01, 1.32051607e-01, 2.00000000e+00],
       [0.00000000e+00, 1.60000000e+01, 1.32051607e-01, 2.00000000e+00],
       [1.24000000e+02, 1.28000000e+02, 1.32051607e-01, 2.00000000e+00],
       [1.23000000e+02, 1.34000000e+02, 1.32747668e-01, 2.00000000e+00],
       [2.00000000e+00, 4.40000000e+01, 1.32747668e-01, 2.00000000e+00],
       [7.60000000e+01, 7.70000000e+01, 1.43823955e-01, 2.00000000e+00],
       [1.80000000e+01, 4.30000000e+01, 1.43823955e-01, 2.00000000e+00],
       [1.00000000e+00, 2.40000000e+01, 1.65423858e-01, 2.00000000e+00],
       [1.16000000e+02, 1.38000000e+02, 1.65423858e-01, 2.00000000e+00],
       [1.10000000e+01, 2.30000000e+01, 1.70965868e-01, 2.00000000e+00],
       [3.70000000e+01, 1.48000000e+02, 1.71197158e-01, 3.00000000e+00],
       [2.90000000e+01, 1.47000000e+02, 1.71197158e-01, 3.00000000e+00],
       [4.00000000e+00, 3.40000000e+01, 1.78358815e-01, 2.00000000e+00],
       [2.80000000e+01, 1.51000000e+02, 1.78481389e-01, 3.00000000e+00],
       [8.40000000e+01, 9.10000000e+01, 1.87242010e-01, 2.00000000e+00],
       [1.32000000e+02, 1.43000000e+02, 2.11666435e-01, 2.00000000e+00],
       [5.90000000e+01, 7.40000000e+01, 2.11666435e-01, 2.00000000e+00],
```

In [34]:
```
lab=dff['cluster'].tolist()
```

In [35]:
```
# plot the dendrogram
dendrogram(z,labels=lab,leaf_rotation=90)
plt.show()
```



AHC can be represented graphically as a dendrogram,which is a tree structure with the root as the cluster that has all observations and the leaves as individual observations.Where there is no overlap in the vertical lines of the bars.The number of line is the optimal number of clusters.So, the optimal number of the clusters is three.Also,if you recall,the Iris dataset has three classes and we got the same number from the above dendrogram.

It is good at identifying small clusters.AHC is less snsitive to outliers than KMeans clustering and slower and more computationally complex than other algorithms. However,Hierarchical clustering has quadratic or higher complexity,making it unfeasible for big data.For large datasets with over 10,000 data points,K-Means is usually the better choice.

In [ ]:
```
```