This is the Iris dataset. The dataset 150 data points.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

Id

SepalLengthCm

SepalWidthCm

PetalLengthCm

PetalWidthCm

Species

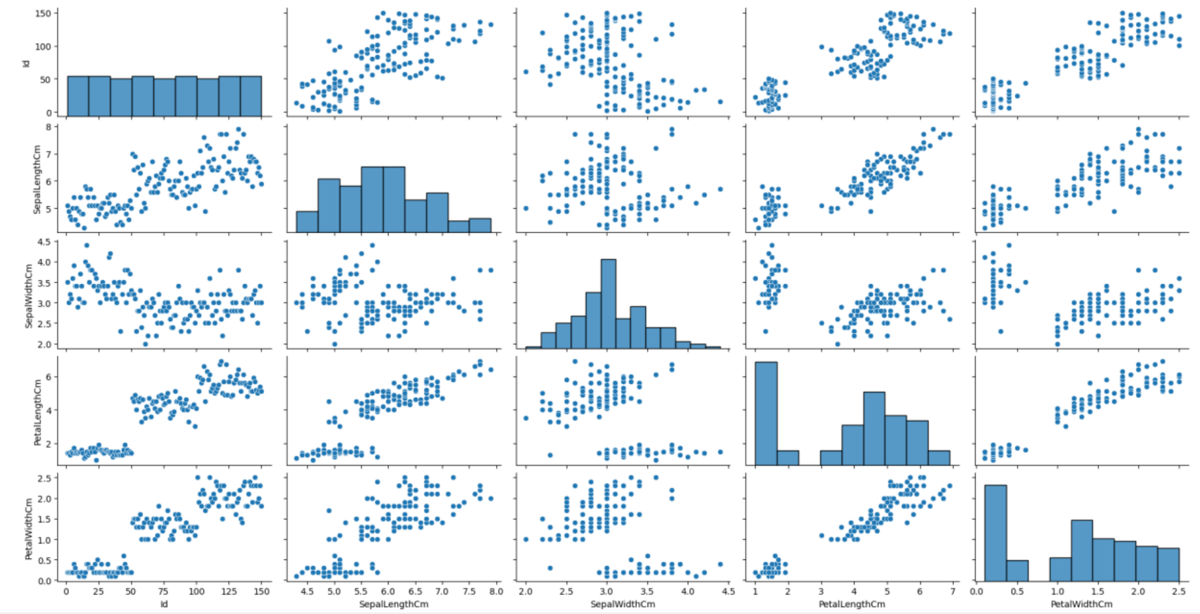
1.Firstly, treat all outliers and missing values in the dataset.

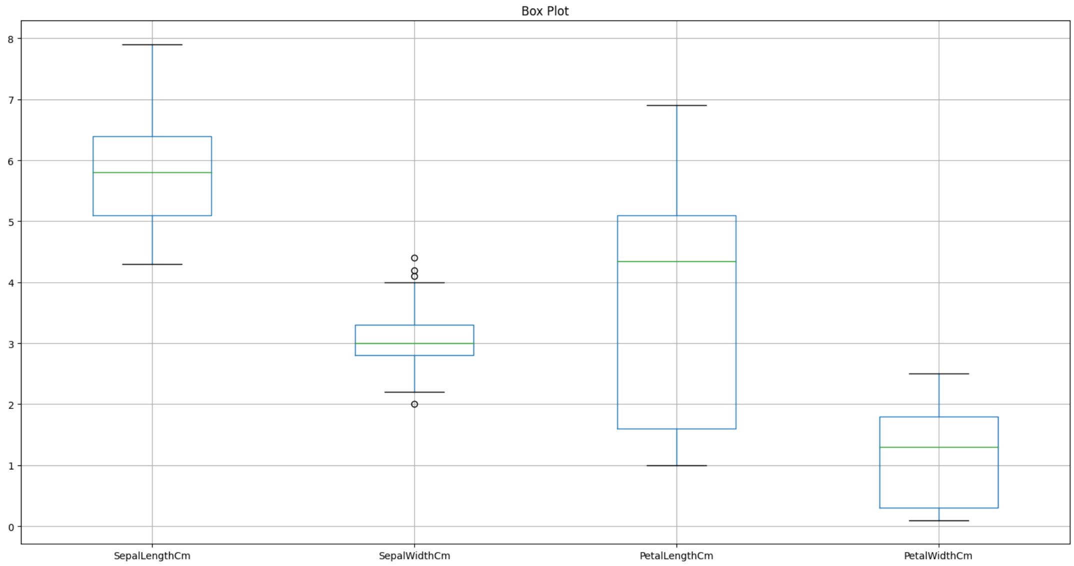
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
# Load the dataset
df = pd.read_csv('Dataset_Day13.csv')
df.info()
# pair plot for additional insight
sns.pairplot(df)
plt.show()
# calculate missing-value percentage
missing_value_percent = df.isna().sum() / len(df) *
100
print(missing_value_percent)
df.boxplot(column=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'])
plt.title('Box Plot')
plt.show()
OutlierData = pd.DataFrame()
temp = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
for col in ['SepalLengthCm', 'SepalWidthCm',
```

```
'PetalLengthCm', 'PetalWidthCm']:
    Q1 = temp[col].quantile(0.25)  # Gives 25th
Percentile or Q1
    Q3 = temp[col].quantile(0.75)  # Gives 75th
Percentile or Q3

    IQR = Q3 - Q1

    UpperBound = Q3 + 1.5 * IQR
    LowerBound = Q1 - 1.5 * IQR

    OutlierData[col] = temp[col][(temp[col] <
LowerBound) | (temp[col] > UpperBound)]
    print(len(OutlierData))
```

File - Day13Q1

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\tejas\PycharmProjects\pythonProject\START\Day13Q1.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Id               0.0
SepalLengthCm    0.0
SepalWidthCm     0.0
PetalLengthCm    0.0
PetalWidthCm     0.0
Species          0.0
dtype: float64
0
4
4
4

Process finished with exit code 0

```

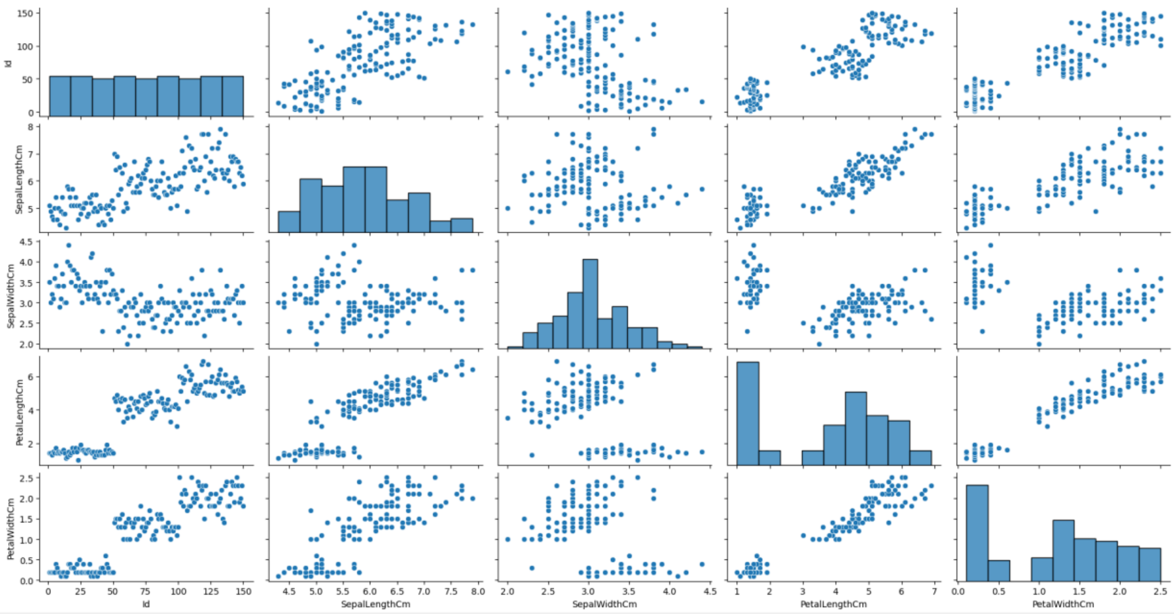


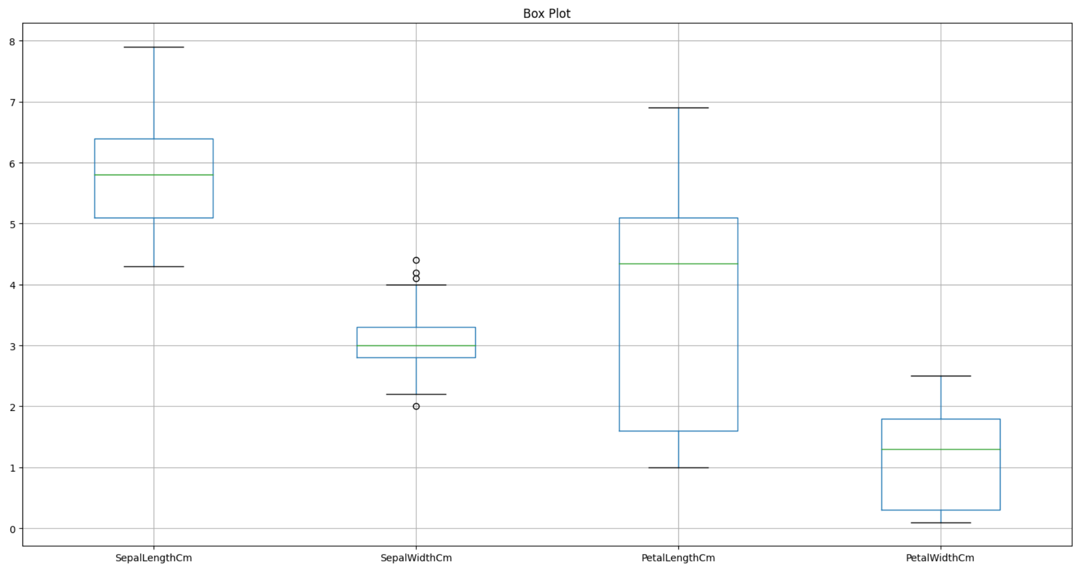


2. Complete all basic data descriptive statistics by `Species`

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv('Dataset_Day13.csv')
df.info()
# pair plot for additional insight
sns.pairplot(df)
```

```
plt.show()
# calculate missing-value percentage
missing_value_percent = df.isna().sum() / len(df)
* 100
print(missing_value_percent)
df.boxplot(column=['SepalLengthCm',
'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm'])
plt.title('Box Plot')
plt.show()
OutlierData = pd.DataFrame()
temp = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
for col in ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']:
    Q1 = temp[col].quantile(0.25)  # Gives 25th
Percentile or Q1
    Q3 = temp[col].quantile(0.75)  # Gives 75th
Percentile or Q3

    IQR = Q3 - Q1

    UpperBound = Q3 + 1.5 * IQR
    LowerBound = Q1 - 1.5 * IQR

    OutlierData[col] = temp[col][(temp[col] <
LowerBound) | (temp[col] > UpperBound)]
    print(len(OutlierData))
# group the dataset based on the "Species" column
descriptive_stats =
df.groupby('Species').describe()
print(descriptive_stats)
```


Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm

File - Day13Q2

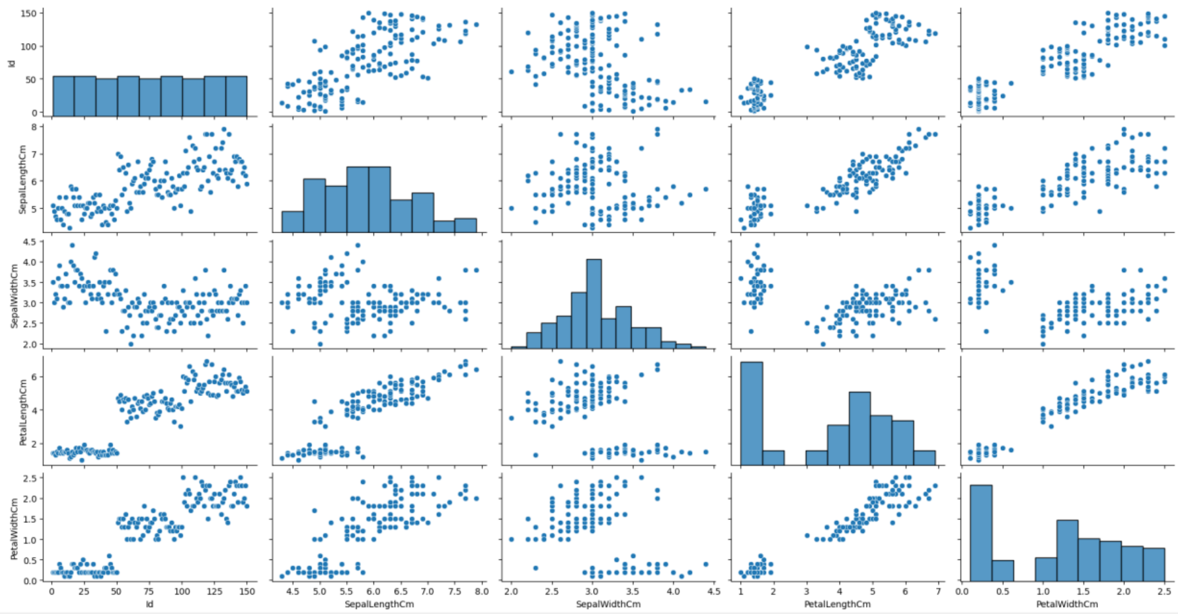
```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day13Q2.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Id               0.0
SepalLengthCm    0.0
SepalWidthCm     0.0
PetalLengthCm    0.0
PetalWidthCm     0.0
Species          0.0
dtype: float64
0
4
4
4
                  Id                                  ... PetalWidthCm

                 count   mean       std    min      25%  ...          min  25%
50%  75%  max
Species
                                                     ...

Iris-setosa      50.0   25.5  14.57738    1.0    13.25  ...          0.1  0.2  0
.2  0.3  0.6
Iris-versicolor  50.0   75.5  14.57738   51.0    63.25  ...          1.0  1.2  1
.3  1.5  1.8
Iris-virginica   50.0  125.5  14.57738  101.0   113.25  ...          1.4  1.8  2
.0  2.3  2.5

[3 rows x 40 columns]

Process finished with exit code 0

```

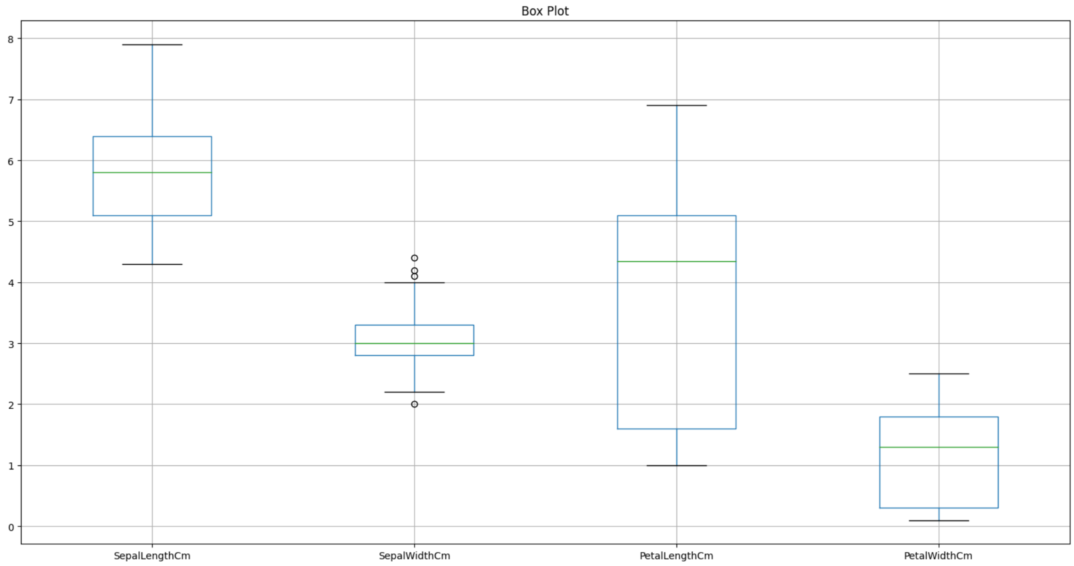
3.Use the `Sepal Length` , `Sepal Width`, `Petal Length` and `Petal Width` to find K-Means clusters.

```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
```

```
# Load the dataset
df = pd.read_csv('Dataset_Day13.csv')
df.info()
# pair plot for additional insight
sns.pairplot(df)
plt.show()
# calculate missing-value percentage
missing_value_percent = df.isna().sum() / len(df) *
100
print(missing_value_percent)
df.boxplot(column=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'])
plt.title('Box Plot')
plt.show()
OutlierData = pd.DataFrame()
temp = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
for col in ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']:
    Q1 = temp[col].quantile(0.25)  # Gives 25th
Percentile or Q1
    Q3 = temp[col].quantile(0.75)  # Gives 75th
Percentile or Q3

    IQR = Q3 - Q1

    UpperBound = Q3 + 1.5 * IQR
    LowerBound = Q1 - 1.5 * IQR

    OutlierData[col] = temp[col][(temp[col] <
LowerBound) | (temp[col] > UpperBound)]
    print(len(OutlierData))
# group the dataset based on the "Species" column
descriptive_stats =
df.groupby('Species').describe()
print(descriptive_stats)
X = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
# specify the number of clusters
k = 3
# create a kmeans instance
km = KMeans(n_clusters=k, n_init=25,
```

```
random_state=1234)
# fit the data to the kmeans model
km.fit(X)
# get the cluster labels for each data point
cluster_labels = km.labels_
# the total within cluster sum of squares
clusterWCSS = km.inertia_
print(cluster_labels)
print(clusterWCSS)
```


Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm

File - Day13Q3

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day13Q3.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Id              0.0
SepalLengthCm   0.0
SepalWidthCm    0.0
PetalLengthCm   0.0
PetalWidthCm    0.0
Species         0.0
dtype: float64
0
4
4
4
                   Id                                ... PetalWidthCm

                count   mean       std     min      25%  ...          min  25%
50%  75%  max
Species
                                                        ...

Iris-setosa      50.0   25.5  14.57738    1.0    13.25  ...          0.1  0.2  0
.2  0.3  0.6
Iris-versicolor  50.0   75.5  14.57738   51.0    63.25  ...          1.0  1.2  1
.3  1.5  1.8
Iris-virginica   50.0  125.5  14.57738  101.0   113.25  ...          1.4  1.8  2
.0  2.3  2.5

[3 rows x 40 columns]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 1 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1
 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1
 1 2]
78.94084142614601

Process finished with exit code 0

```

4.Find the optimum cluster number based on, *elbow method, silhouette method* and *Calinski Harabasz Score.*

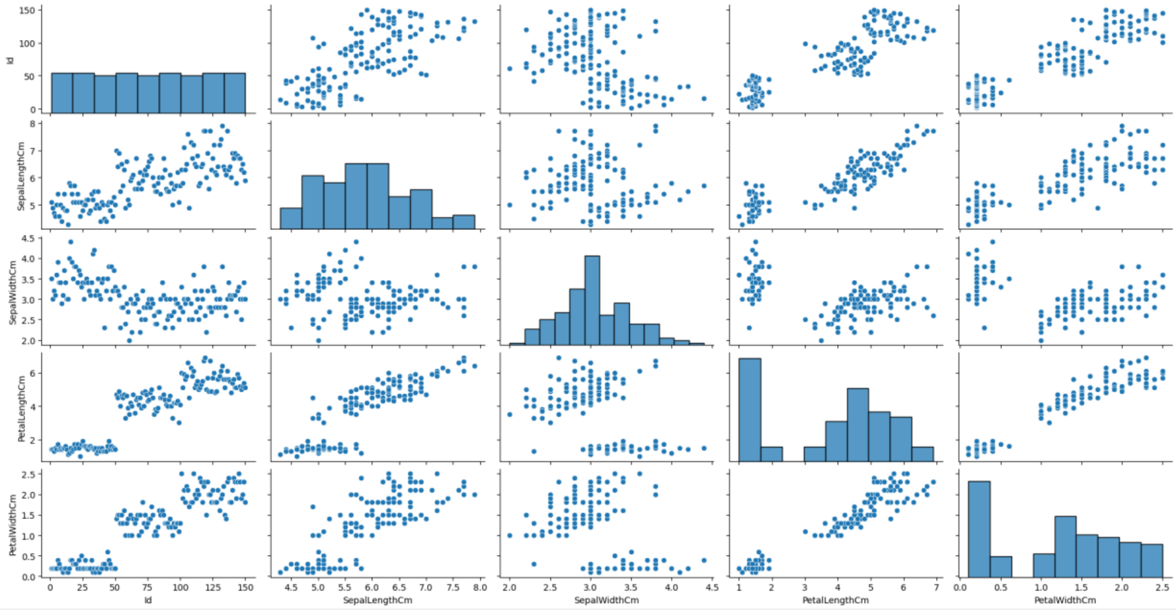
<u>Optimal Cluster number is 3 as we get the same score in elbow method and Calinski Harabasz Score.</u>

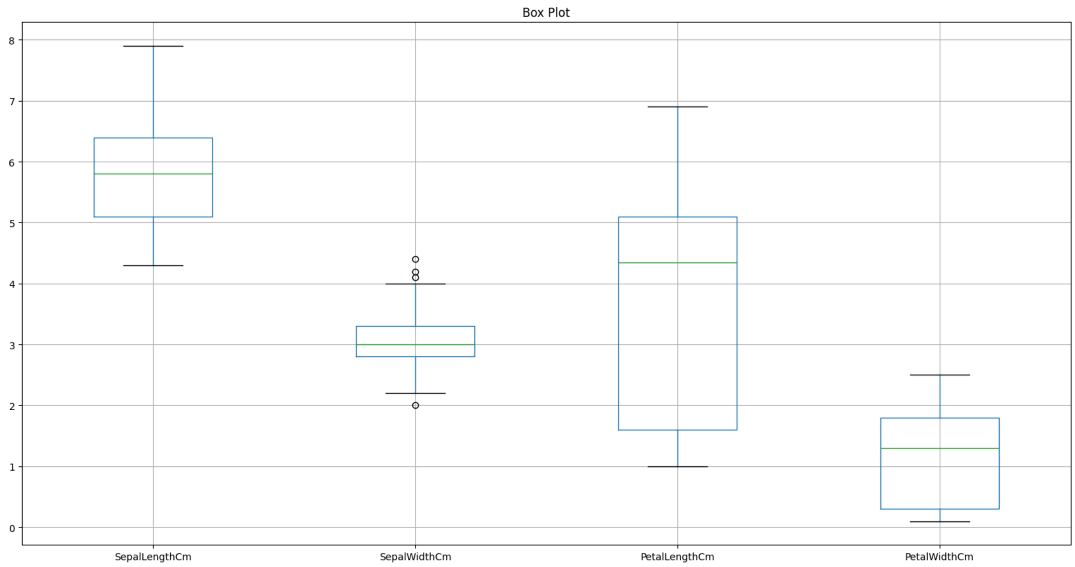
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

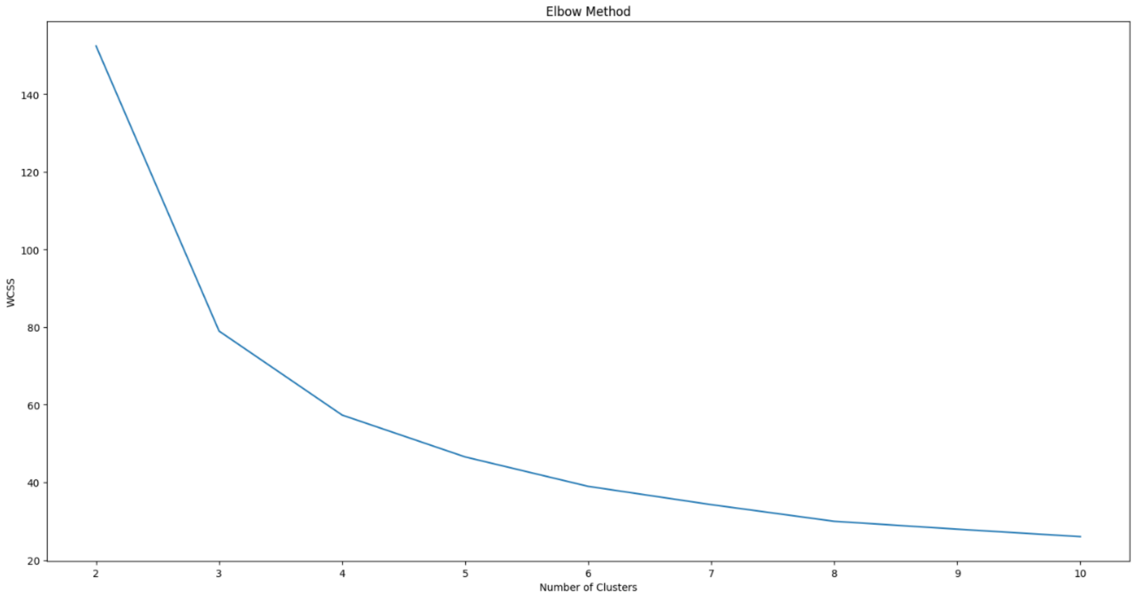
```python
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score

# Load the dataset
df = pd.read_csv('Dataset_Day13.csv')
df.info()
# pair plot for additional insight
sns.pairplot(df)
plt.show()
# calculate missing-value percentage
missing_value_percent = df.isna().sum() / len(df) *
100
print(missing_value_percent)
df.boxplot(column=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'])
plt.title('Box Plot')
plt.show()
OutlierData = pd.DataFrame()
temp = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
for col in ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']:
    Q1 = temp[col].quantile(0.25)  # Gives 25th
Percentile or Q1
    Q3 = temp[col].quantile(0.75)  # Gives 75th
Percentile or Q3

    IQR = Q3 - Q1

    UpperBound = Q3 + 1.5 * IQR
    LowerBound = Q1 - 1.5 * IQR

    OutlierData[col] = temp[col][(temp[col] <
LowerBound) | (temp[col] > UpperBound)]
    print(len(OutlierData))
# group the dataset based on the "Species" column
descriptive_stats =
df.groupby('Species').describe()
print(descriptive_stats)
X = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
```

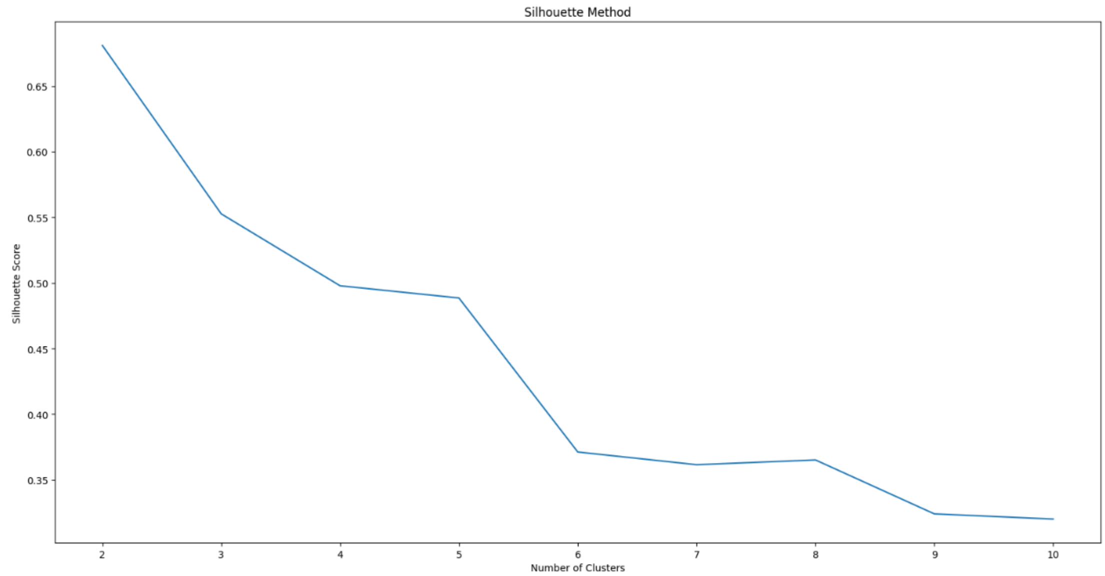
```
# specify the number of clusters
k = 3
# create a kmeans instance
km = KMeans(n_clusters=k, n_init=25,
random_state=1234)
# fit the data to the kmeans model
km.fit(X)
# get the cluster labels for each data point
cluster_labels = km.labels_
# the total within cluster sum of squares
clusterWCSS = km.inertia_
print(cluster_labels)
print(clusterWCSS)
# tabulation of the size of the clusters
pd.Series(km.labels_).value_counts().sort_index()
# 'km.cluster_centers_' :gives the cluster
centroids
cluster_centers = pd.DataFrame(km.cluster_centers_,

columns=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'])
print(cluster_centers)
# to get the correct centroids, we need to un-scale
the data,
cluster_centers_unscaled = pd.DataFrame()
for i in ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']:
    cluster_centers_unscaled[i] =
(cluster_centers[i] * df[i].std()) + df[i].mean()
print(cluster_centers_unscaled)
wcss = []
for k in range(2, 11):
    km = KMeans(n_clusters=k, n_init=25,
random_state=1234)
    km.fit(X)
    wcss.append(km.inertia_)

plt.plot(range(2, 11), wcss)
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.title('Elbow Method')
plt.show()
```

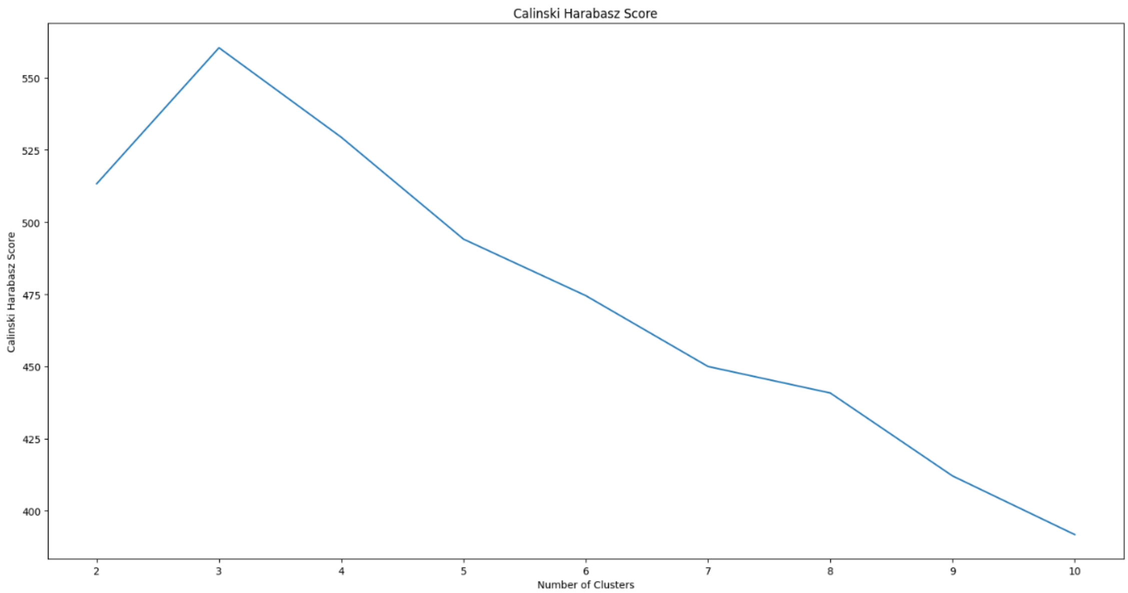
```
silhouette = []
for k in range(2, 11):
    km = KMeans(n_clusters = k, n_init = 25,
random_state = 1234)
    km.fit(X)
    silhouette.append(silhouette_score(X,
km.labels_))

plt.plot(range(2, 11), silhouette)
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Method')
plt.show()
calinski = []
for k in range(2, 11):
    km = KMeans(n_clusters = k, n_init = 25,
random_state = 1234)
    km.fit(X)
    calinski.append(calinski_harabasz_score(X,
km.labels_))

plt.plot(range(2, 11), calinski)
plt.xlabel('Number of Clusters')
plt.ylabel('Calinski Harabasz Score')
plt.title('Calinski Harabasz Score')
plt.show()
```


Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Elbow Method
WCSS
Number of Clusters
20
40
60
80
100
120
140
2
3
4
5
6
7
8
9
10


Silhouette Method
Silhouette Score
0.65
0.60
0.55
0.50
0.45
0.40
0.35
2
3
4
5
6
7
8
9
10
Number of Clusters


Calinski Harabasz Score
Calinski Harabasz Score
550
525
500
475
450
425
400
2
3
4
5
6
7
8
9
10
Number of Clusters

File - Day13Q4

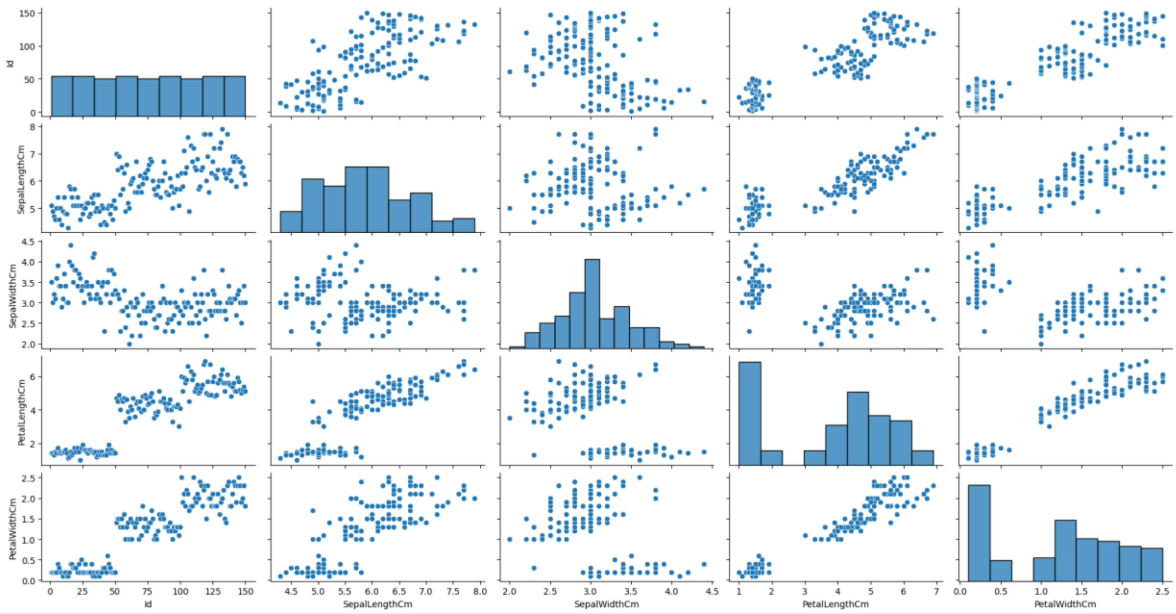
```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day13Q4.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Id               0.0
SepalLengthCm    0.0
SepalWidthCm     0.0
PetalLengthCm    0.0
PetalWidthCm     0.0
Species          0.0
dtype: float64
0
4
4
4
                   Id                                ... PetalWidthCm

                count   mean       std    min     25%  ...          min  25%
50%  75%  max
Species
                                                        ...

Iris-setosa      50.0   25.5  14.57738    1.0   13.25  ...          0.1  0.2  0
.2  0.3  0.6
Iris-versicolor  50.0   75.5  14.57738   51.0   63.25  ...          1.0  1.2  1
.3  1.5  1.8
Iris-virginica   50.0  125.5  14.57738  101.0  113.25  ...          1.4  1.8  2
.0  2.3  2.5

[3 rows x 40 columns]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 1 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1
 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1
 1 2]
78.94084142614601
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0       5.006000      3.418000       1.464000      0.244000
1       6.850000      3.073684       5.742105      2.071053
2       5.901613      2.748387       4.393548      1.433871
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0       9.988632      4.536025       6.341778      1.384878
1      11.515586      4.386732      13.890154      2.779213
2      10.730259      4.245685      11.510733      2.292941

Process finished with exit code 0

```

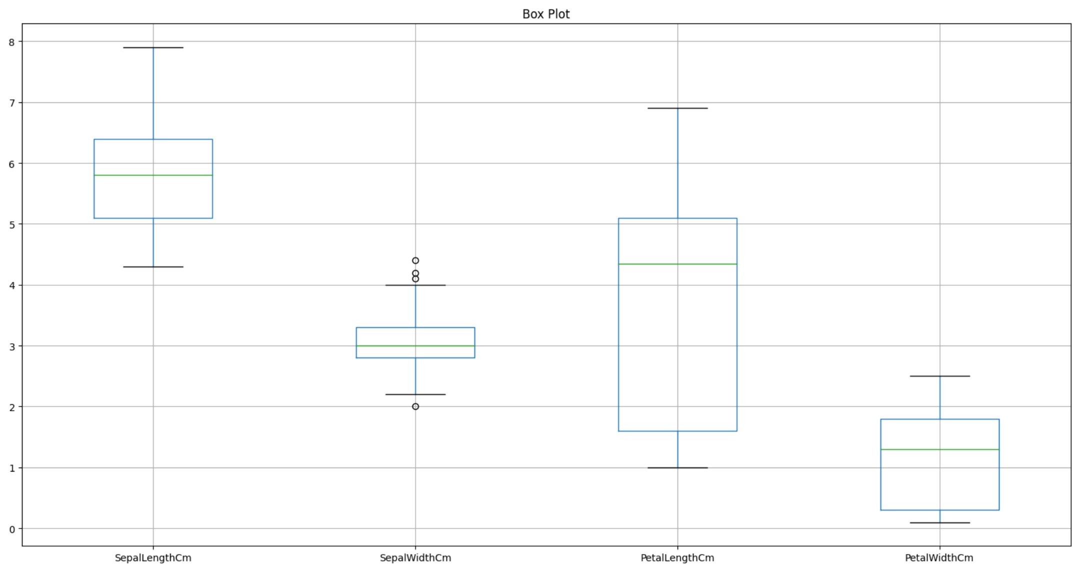
5.Tabulate the proportion of each `Species` among the clusters found as a result of evaluation in task 4.

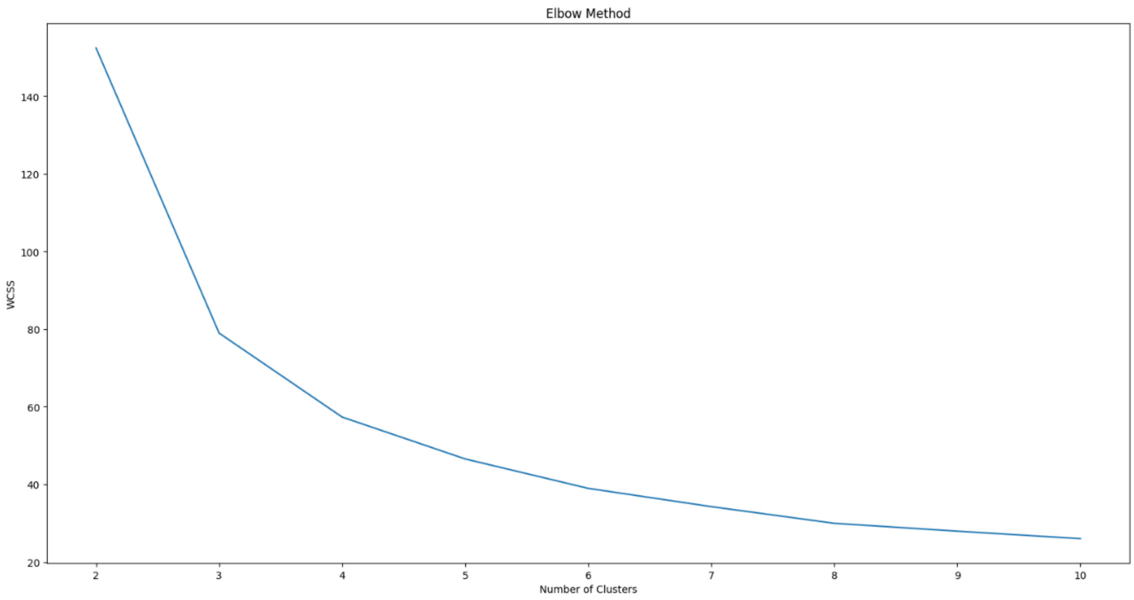
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score

# Load the dataset
df = pd.read_csv('Dataset_Day13.csv')
df.info()
# pair plot for additional insight
sns.pairplot(df)
plt.show()
# calculate missing-value percentage
missing_value_percent = df.isna().sum() / len(df) *
100
print(missing_value_percent)
df.boxplot(column=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'])
plt.title('Box Plot')
plt.show()
OutlierData = pd.DataFrame()
temp = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
for col in ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']:
    Q1 = temp[col].quantile(0.25)  # Gives 25th
Percentile or Q1
    Q3 = temp[col].quantile(0.75)  # Gives 75th
Percentile or Q3

    IQR = Q3 - Q1

    UpperBound = Q3 + 1.5 * IQR
    LowerBound = Q1 - 1.5 * IQR

    OutlierData[col] = temp[col][(temp[col] <
LowerBound) | (temp[col] > UpperBound)]
    print(len(OutlierData))
# group the dataset based on the "Species" column
descriptive_stats =
```

```
df.groupby('Species').describe()
print(descriptive_stats)
X = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
# specify the number of clusters
k = 3
# create a kmeans instance
km = KMeans(n_clusters=k, n_init=25,
random_state=1234)
# fit the data to the kmeans model
km.fit(X)
# get the cluster labels for each data point
cluster_labels = km.labels_
# the total within cluster sum of squares
clusterWCSS = km.inertia_
print(cluster_labels)
print(clusterWCSS)
# tabulation of the size of the clusters
pd.Series(km.labels_).value_counts().sort_index()
# 'km.cluster_centers_' :gives the cluster
centroids
cluster_centers = pd.DataFrame(km.cluster_centers_,

columns=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'])
print(cluster_centers)
# to get the correct centroids, we need to un-scale
the data,
cluster_centers_unscaled = pd.DataFrame()
for i in ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']:
    cluster_centers_unscaled[i] =
(cluster_centers[i] * df[i].std()) + df[i].mean()
print(cluster_centers_unscaled)
wcss = []
for k in range(2, 11):
    km = KMeans(n_clusters=k, n_init=25,
random_state=1234)
    km.fit(X)
    wcss.append(km.inertia_)

plt.plot(range(2, 11), wcss)
```

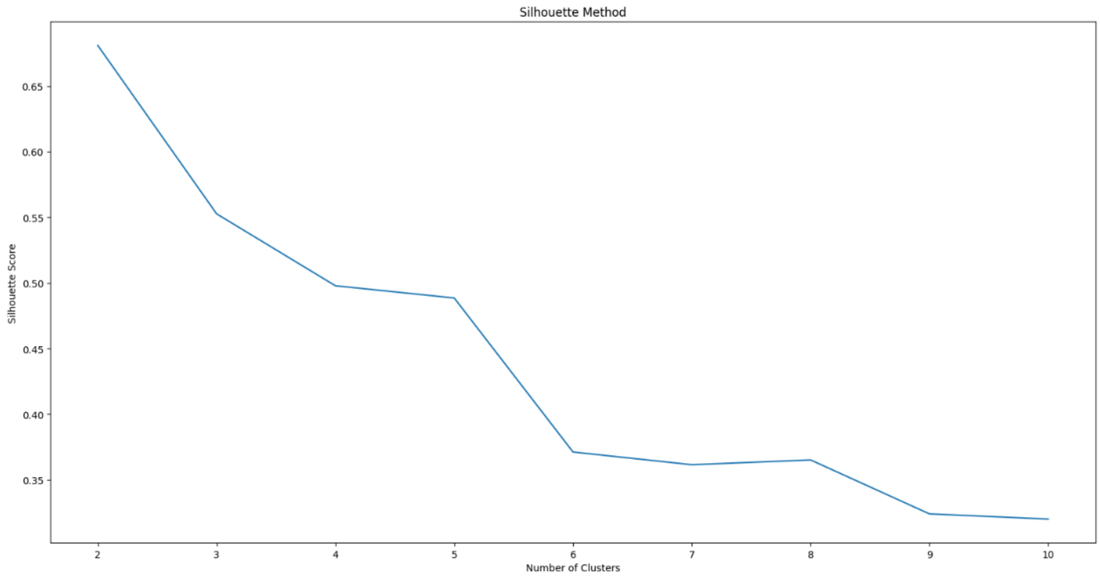
```python
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.title('Elbow Method')
plt.show()
silhouette = []
for k in range(2, 11):
    km = KMeans(n_clusters = k, n_init = 25,
random_state = 1234)
    km.fit(X)
    silhouette.append(silhouette_score(X,
km.labels_))

plt.plot(range(2, 11), silhouette)
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Method')
plt.show()
calinski = []
for k in range(2, 11):
    km = KMeans(n_clusters = k, n_init = 25,
random_state = 1234)
    km.fit(X)
    calinski.append(calinski_harabasz_score(X,
km.labels_))

plt.plot(range(2, 11), calinski)
plt.xlabel('Number of Clusters')
plt.ylabel('Calinski Harabasz Score')
plt.title('Calinski Harabasz Score')
plt.show()
cluster_df = pd.DataFrame({'Cluster':
cluster_labels, 'Species': df['Species']})
cross_tab = pd.crosstab(cluster_df['Cluster'],
cluster_df['Species'])
proportion = cross_tab.div(cross_tab.sum(axis=1),
axis=0) * 100
print(proportion)
```

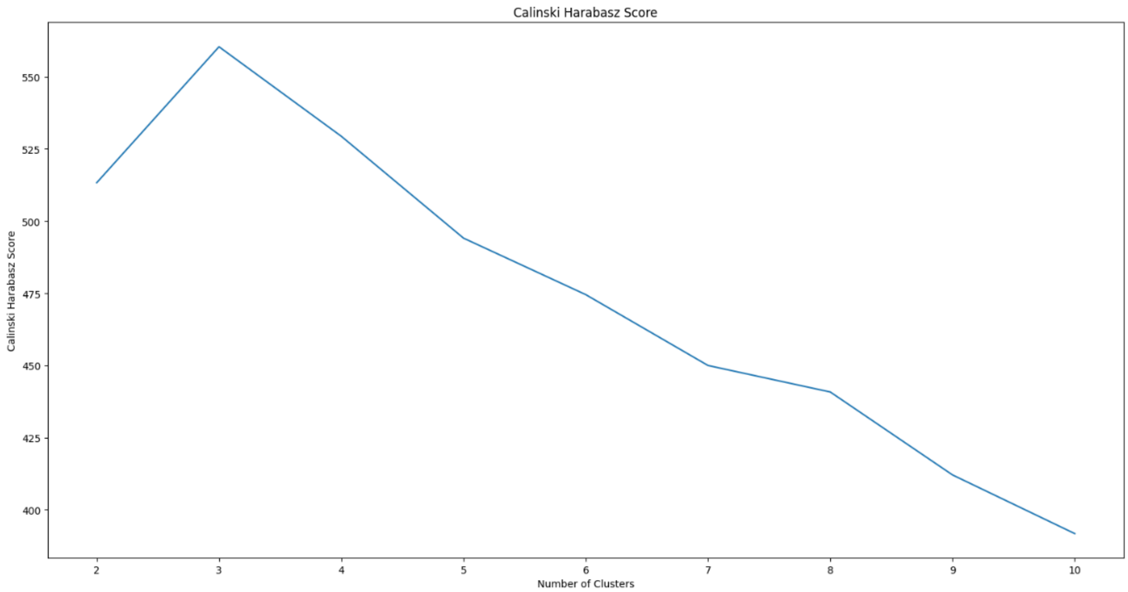

Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm



Elbow Method
WCSS
Number of Clusters
20
40
60
80
100
120
140
2
3
4
5
6
7
8
9
10


Silhouette Method
Silhouette Score
0.65
0.60
0.55
0.50
0.45
0.40
0.35
2
3
4
5
6
7
8
9
10
Number of Clusters


Calinski Harabasz Score
Calinski Harabasz Score
550
525
500
475
450
425
400
2
3
4
5
6
7
8
9
10
Number of Clusters

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day13Q5.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Id               0.0
SepalLengthCm    0.0
SepalWidthCm     0.0
PetalLengthCm    0.0
PetalWidthCm     0.0
Species          0.0
dtype: float64
0
4
4
4
                   Id                                    ... PetalWidthCm

                count   mean       std    min     25%  ...          min  25%
50%  75%  max
Species
                                                      ...

Iris-setosa      50.0   25.5  14.57738    1.0   13.25  ...          0.1  0.2  0
.2  0.3  0.6
Iris-versicolor  50.0   75.5  14.57738   51.0   63.25  ...          1.0  1.2  1
.3  1.5  1.8
Iris-virginica   50.0  125.5  14.57738  101.0  113.25  ...          1.4  1.8  2
.0  2.3  2.5

[3 rows x 40 columns]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 1 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1
 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1
 1 2]
78.94084142614601
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0       5.006000      3.418000       1.464000      0.244000
1       6.850000      3.073684       5.742105      2.071053
2       5.901613      2.748387       4.393548      1.433871
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0       9.988632      4.536025       6.341778      1.384878
1      11.515586      4.386732      13.890154      2.779213
2      10.730259      4.245685      11.510733      2.292941
Species  Iris-setosa  Iris-versicolor  Iris-virginica
Cluster
0              100.0         0.000000        0.000000
1                0.0         5.263158       94.736842
```

File - Day13Q5

```
2                0.0        77.419355        22.580645

Process finished with exit code 0
```

6.Share your insights on the data based on the clusters [optional]

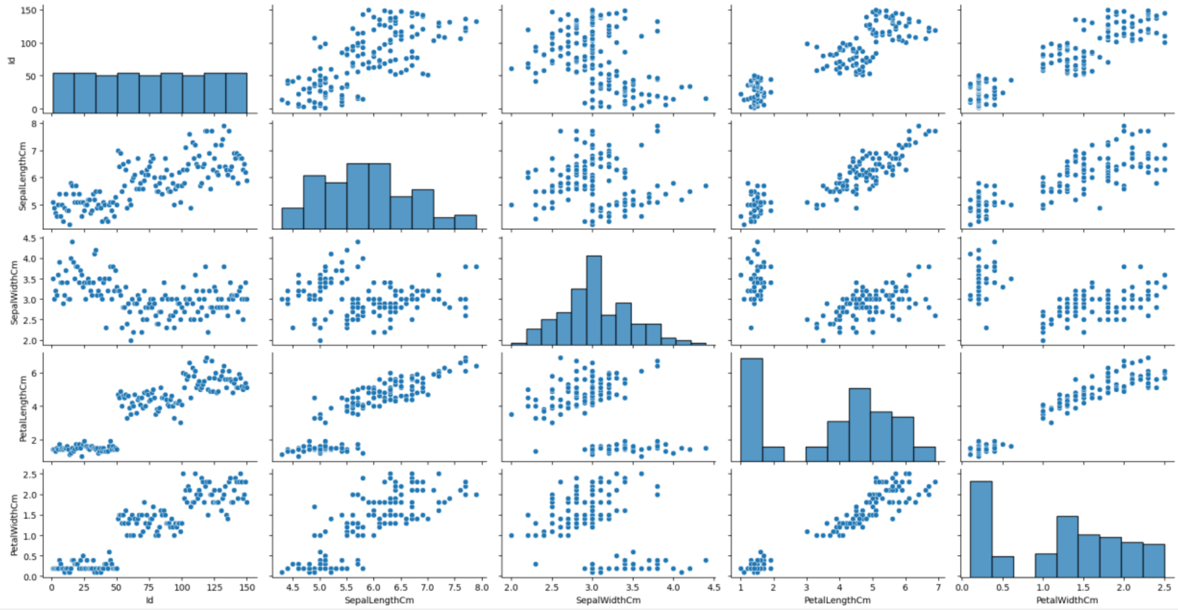
Petal width and petal length are better differentiators than sepal width and sepal length .In the petal width-petal length plot we can see that setosa has smallest petal width and petal length, versicolor lies between setosa and virginica , virginica has the largest petal width and petal length. But from the sepal width-sepal length plot we don't get much idea about the species differentiation, albeit we can observe that sepal length gives us the idea that versicolor and virginca are somehow related or have shared qualities, sepal width doesn't give us any idea AT ALL. So it's not a good differentiator

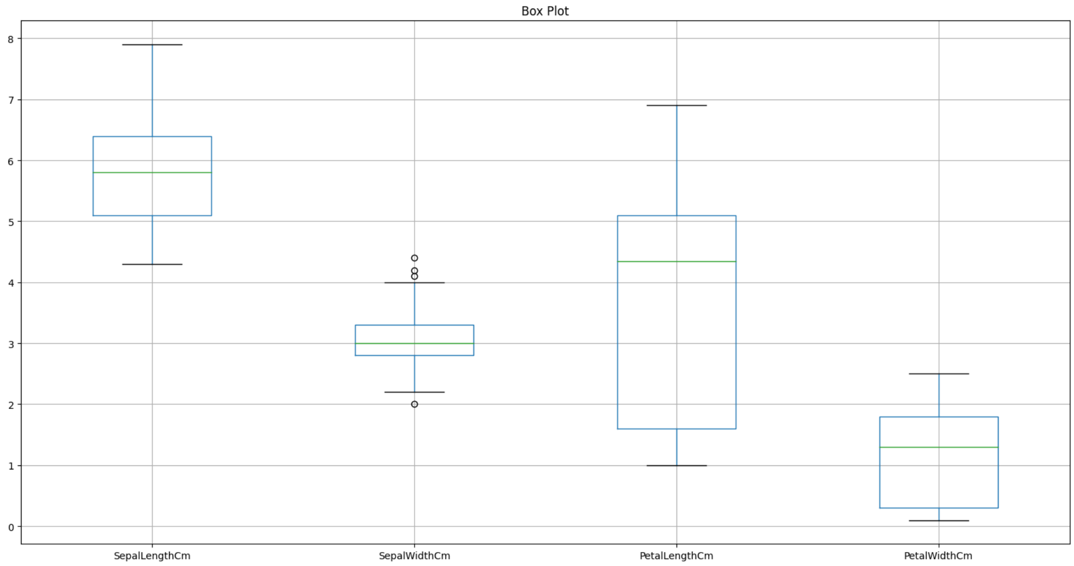
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score

# Load the dataset
df = pd.read_csv('Dataset_Day13.csv')
df.info()
# pair plot for additional insight
sns.pairplot(df)
plt.show()
# calculate missing-value percentage
missing_value_percent = df.isna().sum() / len(df) *
100
print(missing_value_percent)
df.boxplot(column=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'])
plt.title('Box Plot')
```

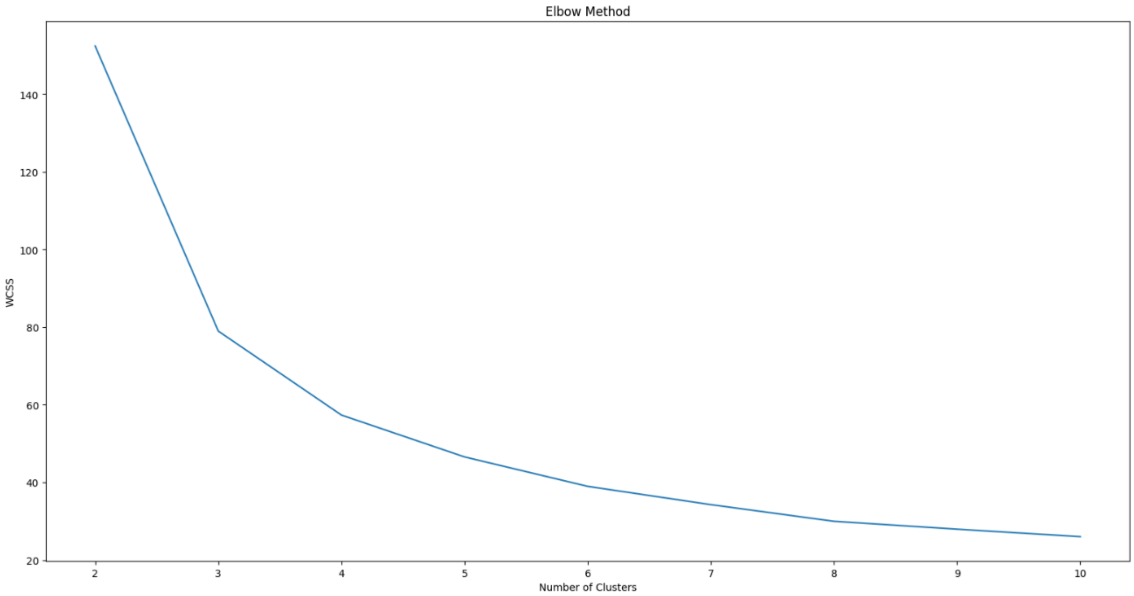
```
plt.show()
OutlierData = pd.DataFrame()
temp = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
for col in ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']:
    Q1 = temp[col].quantile(0.25)  # Gives 25th
Percentile or Q1
    Q3 = temp[col].quantile(0.75)  # Gives 75th
Percentile or Q3

    IQR = Q3 - Q1

    UpperBound = Q3 + 1.5 * IQR
    LowerBound = Q1 - 1.5 * IQR

    OutlierData[col] = temp[col][(temp[col] <
LowerBound) | (temp[col] > UpperBound)]
    print(len(OutlierData))
# group the dataset based on the "Species" column
descriptive_stats =
df.groupby('Species').describe()
print(descriptive_stats)
X = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
# specify the number of clusters
k = 3
# create a kmeans instance
km = KMeans(n_clusters=k, n_init=25,
random_state=1234)
# fit the data to the kmeans model
km.fit(X)
# get the cluster labels for each data point
cluster_labels = km.labels_
# the total within cluster sum of squares
clusterWCSS = km.inertia_
print(cluster_labels)
print(clusterWCSS)
# tabulation of the size of the clusters
pd.Series(km.labels_).value_counts().sort_index()
# 'km.cluster_centers_' :gives the cluster
centroids
```

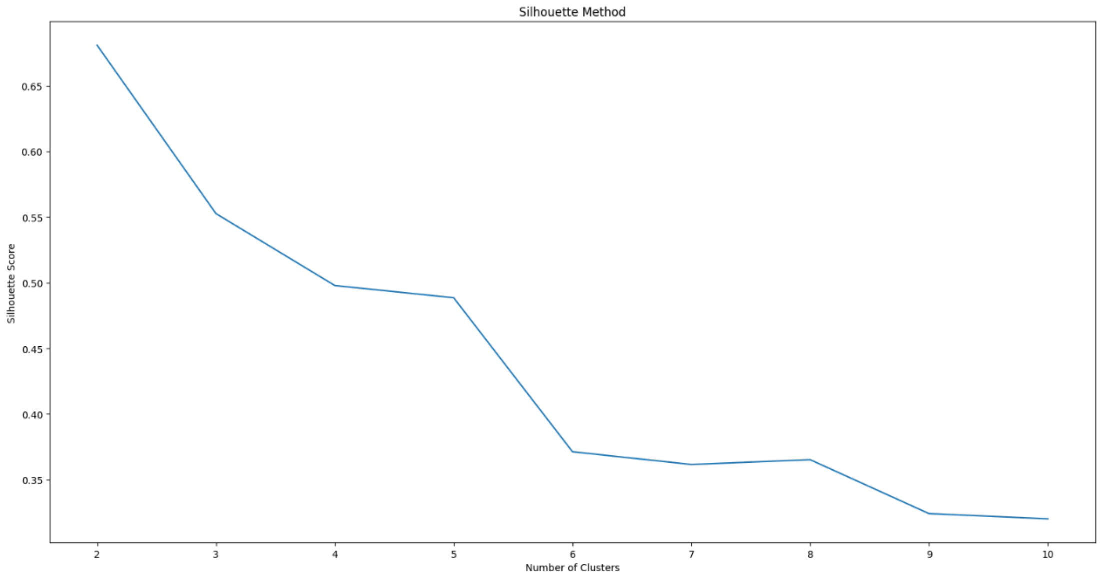
```
cluster_centers = pd.DataFrame(km.cluster_centers_,

columns=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'])
print(cluster_centers)
# to get the correct centroids, we need to un-scale
the data,
cluster_centers_unscaled = pd.DataFrame()
for i in ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']:
    cluster_centers_unscaled[i] =
(cluster_centers[i] * df[i].std()) + df[i].mean()
print(cluster_centers_unscaled)
wcss = []
for k in range(2, 11):
    km = KMeans(n_clusters=k, n_init=25,
random_state=1234)
    km.fit(X)
    wcss.append(km.inertia_)

plt.plot(range(2, 11), wcss)
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.title('Elbow Method')
plt.show()
silhouette = []
for k in range(2, 11):
    km = KMeans(n_clusters = k, n_init = 25,
random_state = 1234)
    km.fit(X)
    silhouette.append(silhouette_score(X,
km.labels_))

plt.plot(range(2, 11), silhouette)
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Method')
plt.show()
calinski = []
for k in range(2, 11):
    km = KMeans(n_clusters = k, n_init = 25,
random_state = 1234)
```

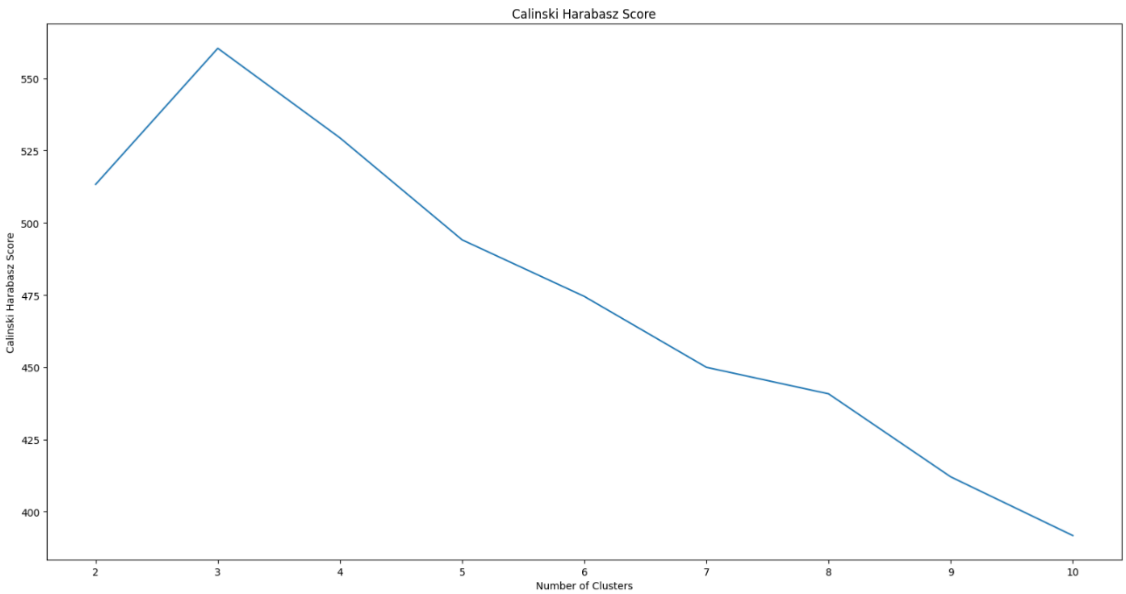
```
    km.fit(X)
    calinski.append(calinski_harabasz_score(X,
km.labels_))

plt.plot(range(2, 11), calinski)
plt.xlabel('Number of Clusters')
plt.ylabel('Calinski Harabasz Score')
plt.title('Calinski Harabasz Score')
plt.show()
cluster_df = pd.DataFrame({'Cluster':
cluster_labels, 'Species': df['Species']})
cross_tab = pd.crosstab(cluster_df['Cluster'],
cluster_df['Species'])
proportion = cross_tab.div(cross_tab.sum(axis=1),
axis=0) * 100
print(proportion)
sns.scatterplot(x='PetalLengthCm',
y='PetalWidthCm',
                hue='Species', data=df, )
plt.show()
sns.scatterplot(x='SepalLengthCm',
y='SepalWidthCm',
                hue='Species', data=df, )
plt.show()
```


Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Elbow Method
WCSS
Number of Clusters
140
120
100
80
60
40
20
2
3
4
5
6
7
8
9
10


Silhouette Method
Silhouette Score
0.65
0.60
0.55
0.50
0.45
0.40
0.35
2
3
4
5
6
7
8
9
10
Number of Clusters


Calinski Harabasz Score
Calinski Harabasz Score
550
525
500
475
450
425
400
2
3
4
5
6
7
8
9
10
Number of Clusters

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day13Q5.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
Id               0.0
SepalLengthCm    0.0
SepalWidthCm     0.0
PetalLengthCm    0.0
PetalWidthCm     0.0
Species          0.0
dtype: float64
0
4
4
4
                   Id                                ... PetalWidthCm

                count   mean       std    min     25%  ...          min  25%
50%  75%  max
Species
                                                    ...

Iris-setosa      50.0   25.5  14.57738    1.0   13.25  ...          0.1  0.2  0
.2  0.3  0.6
Iris-versicolor  50.0   75.5  14.57738   51.0   63.25  ...          1.0  1.2  1
.3  1.5  1.8
Iris-virginica   50.0  125.5  14.57738  101.0  113.25  ...          1.4  1.8  2
.0  2.3  2.5

[3 rows x 40 columns]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 1 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 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1 1 1 1 2 1 1 1 1
 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1
 1 2]
78.94084142614601
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0       5.006000      3.418000       1.464000      0.244000
1       6.850000      3.073684       5.742105      2.071053
2       5.901613      2.748387       4.393548      1.433871
   SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
0       9.988632      4.536025       6.341778      1.384878
1      11.515586      4.386732      13.890154      2.779213
2      10.730259      4.245685      11.510733      2.292941
Species  Iris-setosa  Iris-versicolor  Iris-virginica
Cluster
0              100.0         0.000000        0.000000
1                0.0         5.263158       94.736842
```

File - Day13Q5

```
52 2                 0.0        77.419355        22.580645
53
54 Process finished with exit code 0
55
```