This is the Iris dataset with150 data points (same dataset as Day 13)

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

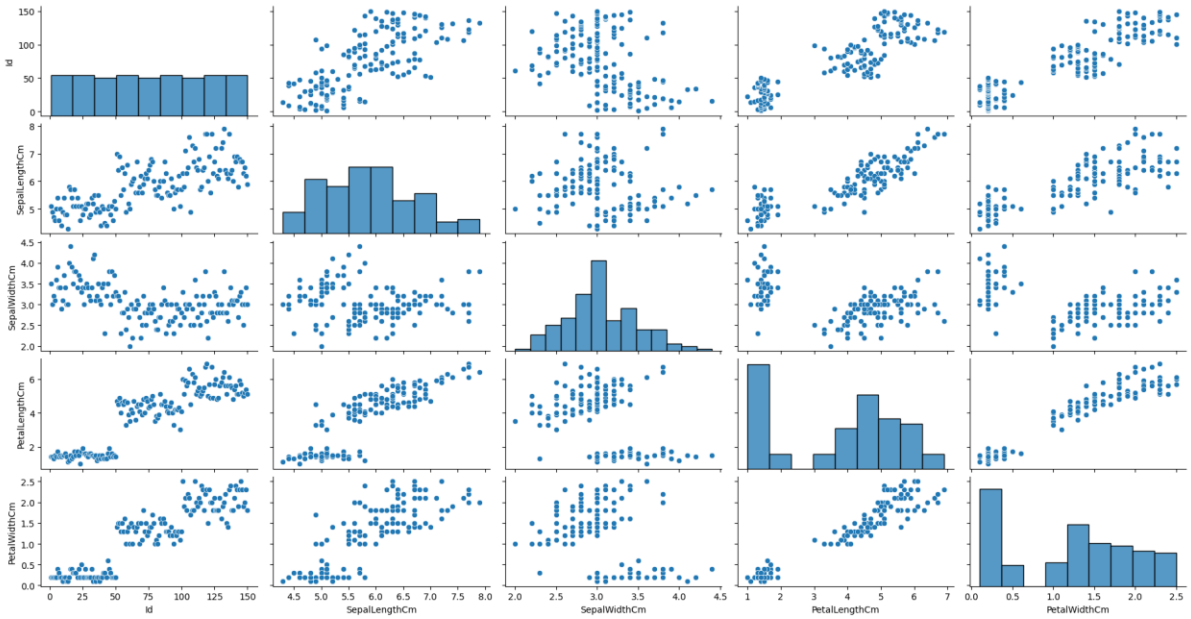
PetalWidthCm

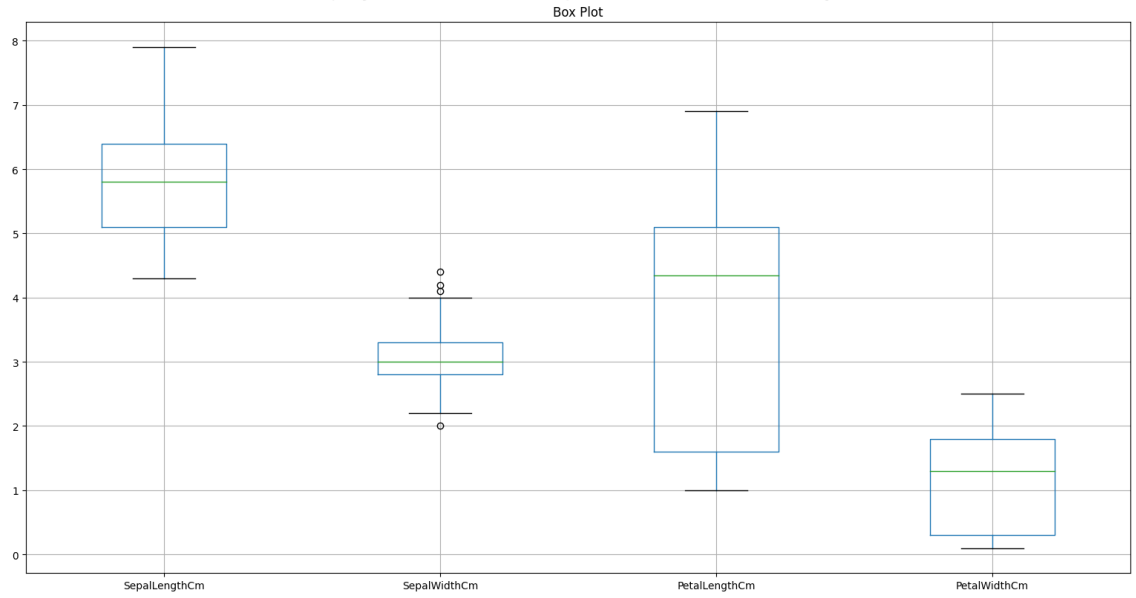
Species

1. Treat outliers and missing values if present and scale the data.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
# 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)
columns=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']
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))
# Scale the data
df[columns] =
MinMaxScaler().fit_transform(df[columns])
print(df.info())
```


Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm

File - Day14Q1

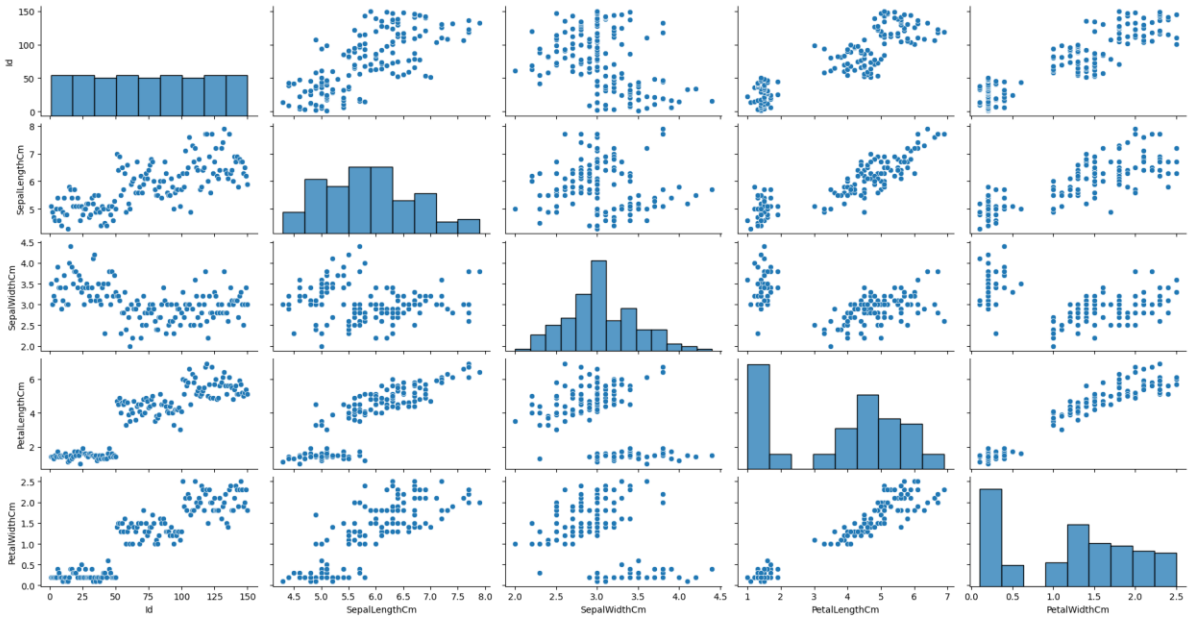
```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day14Q1.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 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
<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
None

Process finished with exit code 0

```

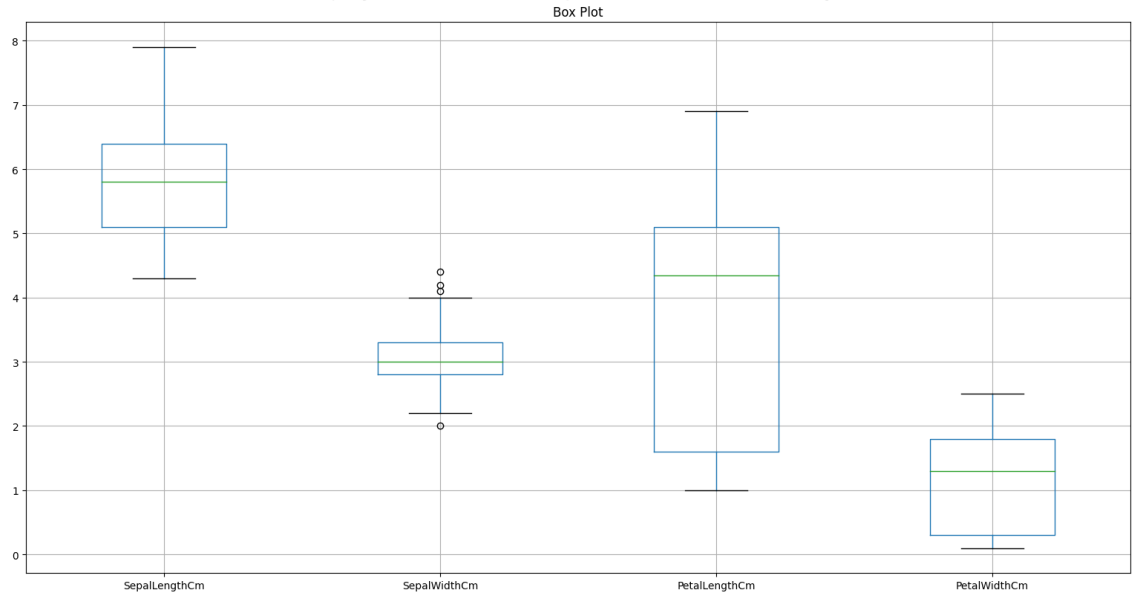
2. Fit the DBSCAN clusters for the default parameter values and also show the *Species* distribution in each of the default clusters.

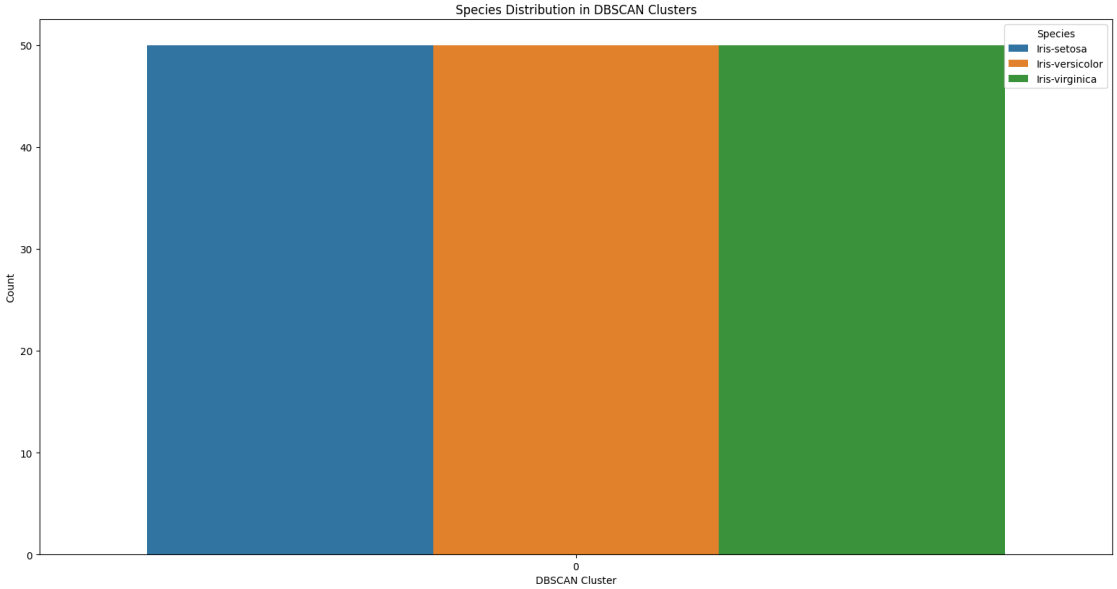
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.cluster import DBSCAN
# Load the dataset
```

```
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)
columns=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']
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))
# Scale the data
df[columns] =
MinMaxScaler().fit_transform(df[columns])
print(df.info())
X = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
# Fit the data to the DBSCAN model
cluster_labels = DBSCAN().fit_predict(X)
# Add the cluster labels to the original DataFrame
```

```
df['DBSCAN_Cluster'] = cluster_labels
# Show the species distribution in each of the
default clusters
species_distribution =
df.groupby(['DBSCAN_Cluster',
'Species']).size().reset_index(name='Count')
# Plot the distribution
sns.barplot(x='DBSCAN_Cluster', y='Count',
hue='Species', data=species_distribution)
plt.xlabel('DBSCAN Cluster')
plt.ylabel('Count')
plt.title('Species Distribution in DBSCAN
Clusters')
plt.show()
```


Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Species Distribution in DBSCAN Clusters
Species
Iris-setosa
Iris-versicolor
Iris-virginica
Count
0
10
20
30
40
50
0
DBSCAN Cluster

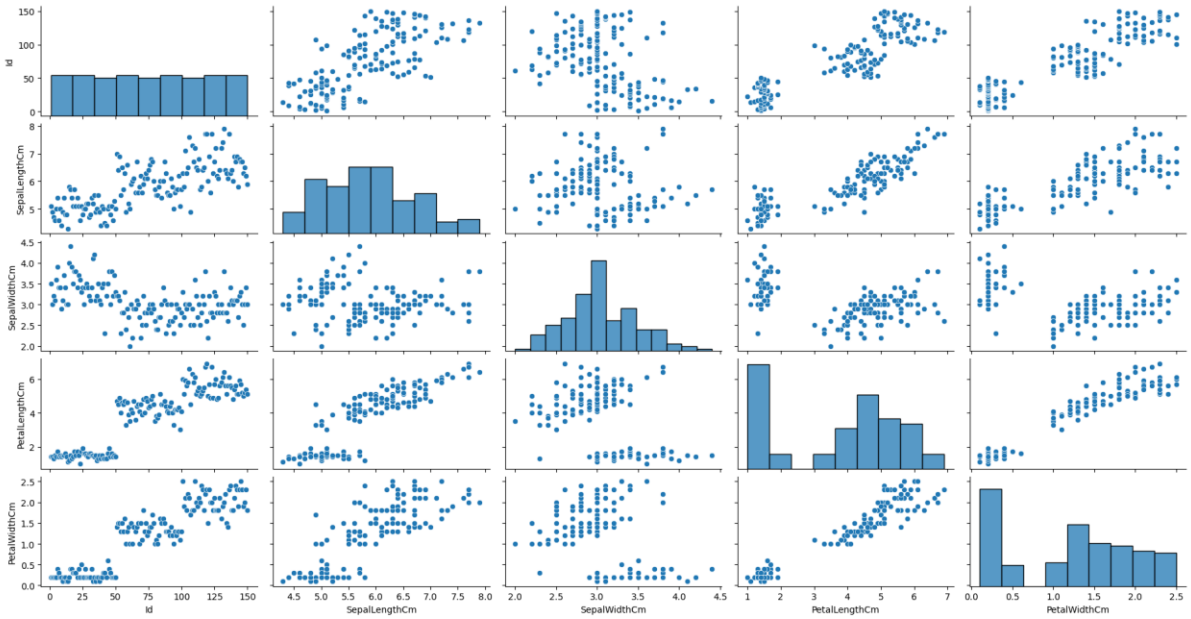
```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day14Q2.py
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 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
<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
None

Process finished with exit code 0

```

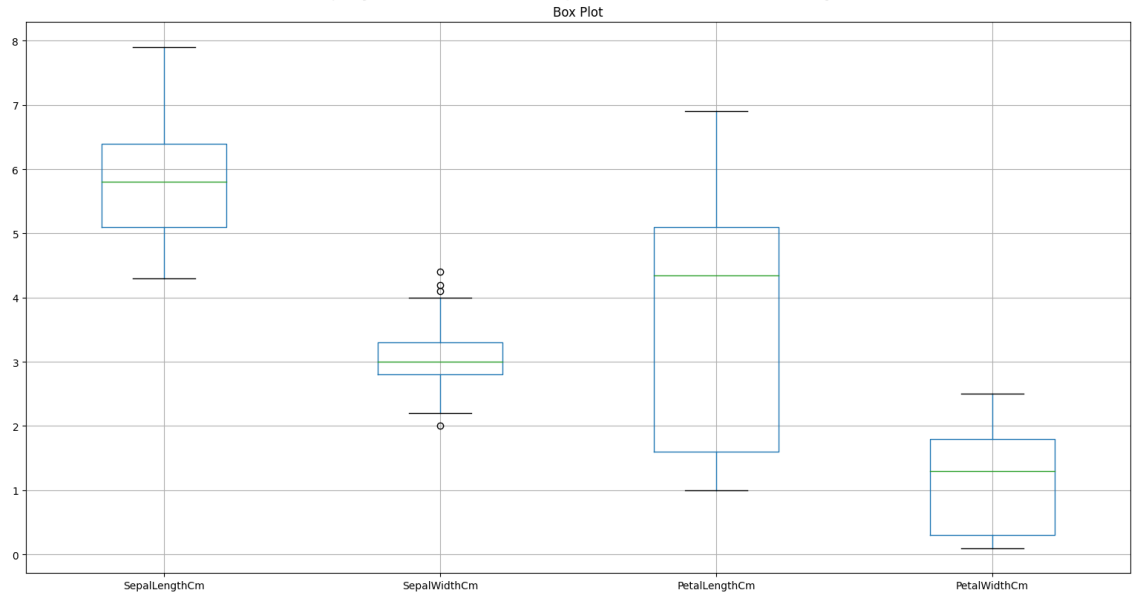
3. Use nearest neighbour algorithm to find the most optimal value of ‘eps’ parameter.

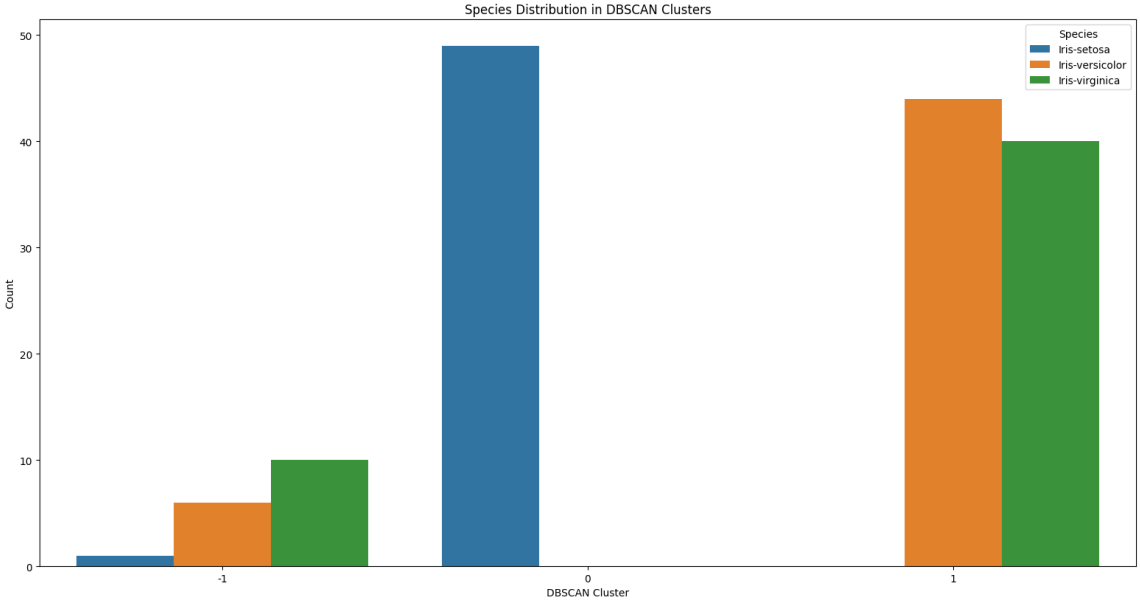
```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import DBSCAN

# Load the dataset
df = pd.read_csv('Dataset_Day13.csv')
```

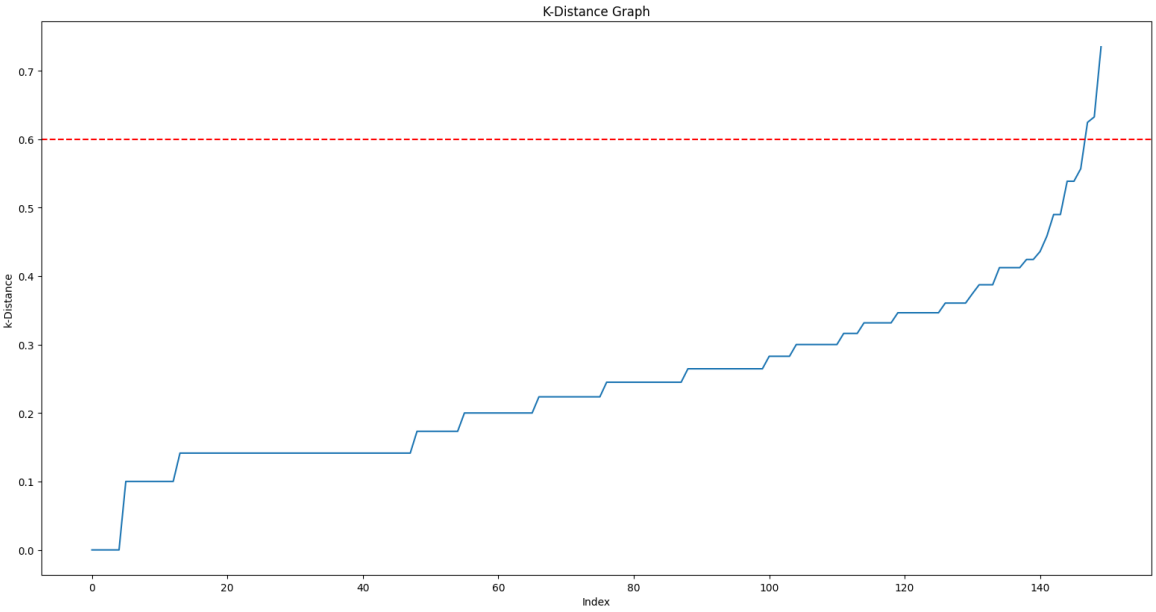
```
df.info()
# pair plot for additional insight
sns.pairplot(df)
plt.show()
# calculate missing-value percentage
missing_value_percent = df.isna().sum() /
len(df) * 100
print(missing_value_percent)
columns = ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']
df.boxplot(column=['SepalLengthCm',
'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))
# Scale the data
df[columns] =
MinMaxScaler().fit_transform(df[columns])
print(df.info())
X = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
# Fit the data to the DBSCAN model
cluster_labels = DBSCAN().fit_predict(X)
# Add the cluster labels to the original
```

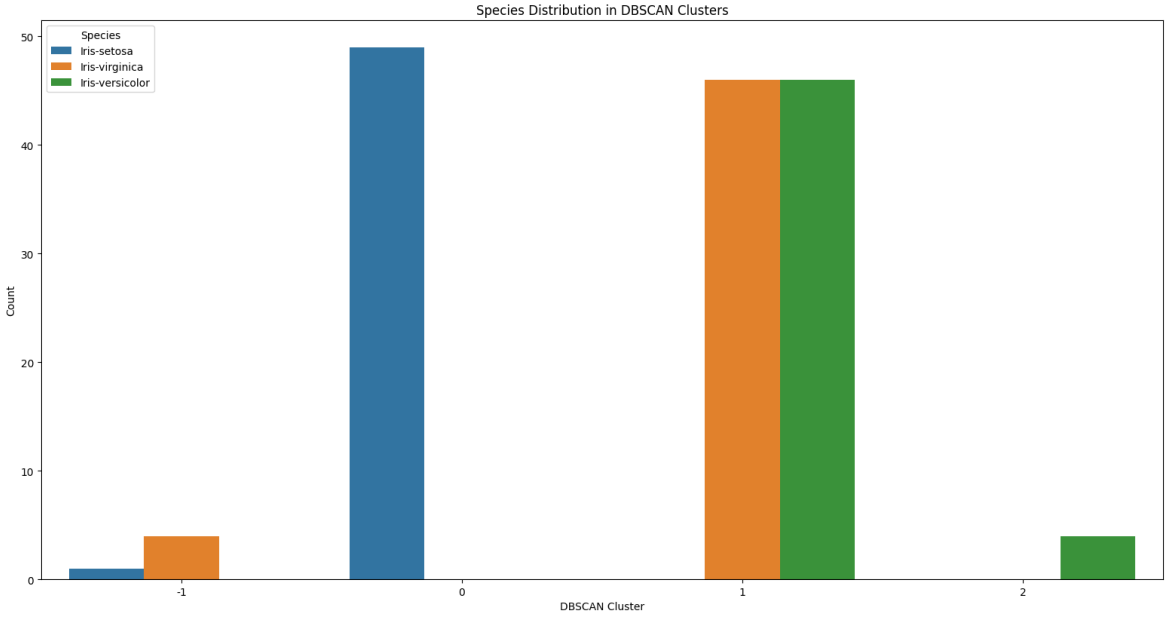
```
DataFrame
df['DBSCAN_Cluster'] = cluster_labels
# Show the species distribution in each of the
default clusters
species_distribution =
df.groupby(['DBSCAN_Cluster',
'Species']).size().reset_index(name='Count')
# Plot the distribution
sns.barplot(x='DBSCAN_Cluster', y='Count',
hue='Species', data=species_distribution)
plt.xlabel('DBSCAN Cluster')
plt.ylabel('Count')
plt.title('Species Distribution in DBSCAN
Clusters')
plt.show()
k = 7
nn = NearestNeighbors(n_neighbors=k).fit(X)
distances, indices = nn.kneighbors(X)
distances = np.sort(distances, axis=0)[:, 1]
plt.plot(distances)
plt.axhline(y=0.6, color='r', ls="--")
plt.xlabel('Index')
plt.ylabel('k-Distance')
plt.title('K-Distance Graph')
plt.show()
optimal_eps = float(input("Enter the optimal
eps value based on the plot: "))
dbscan = DBSCAN(eps=optimal_eps, min_samples=3)
cluster_labels = dbscan.fit_predict(X)
df['DBSCAN_Cluster'] = cluster_labels
species_distribution =
df.groupby(['DBSCAN_Cluster',
'Species']).size().reset_index(name='Count')
sns.barplot(x='DBSCAN_Cluster', y='Count',
hue='Species', data=species_distribution)
plt.xlabel('DBSCAN Cluster')
plt.ylabel('Count')
plt.title('Species Distribution in DBSCAN
Clusters')
plt.show()
print(f"Optimal eps value: {optimal_eps:.3f}")
```


Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Species Distribution in DBSCAN Clusters
Species
Iris-setosa
Iris-versicolor
Iris-virginica
Count
DBSCAN Cluster
0
10
20
30
40
50
-1
0
1


K-Distance Graph
k-Distance
Index
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
120
140


Species Distribution in DBSCAN Clusters
Species
Iris-setosa
Iris-virginica
Iris-versicolor
Count
DBSCAN Cluster
0
10
20
30
40
50
-1
0
1
2

File - Day14Q3

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day14Q3.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
Enter the optimal eps value based on the plot: 0.6
Optimal eps value: 0.600

Process finished with exit code 0

```

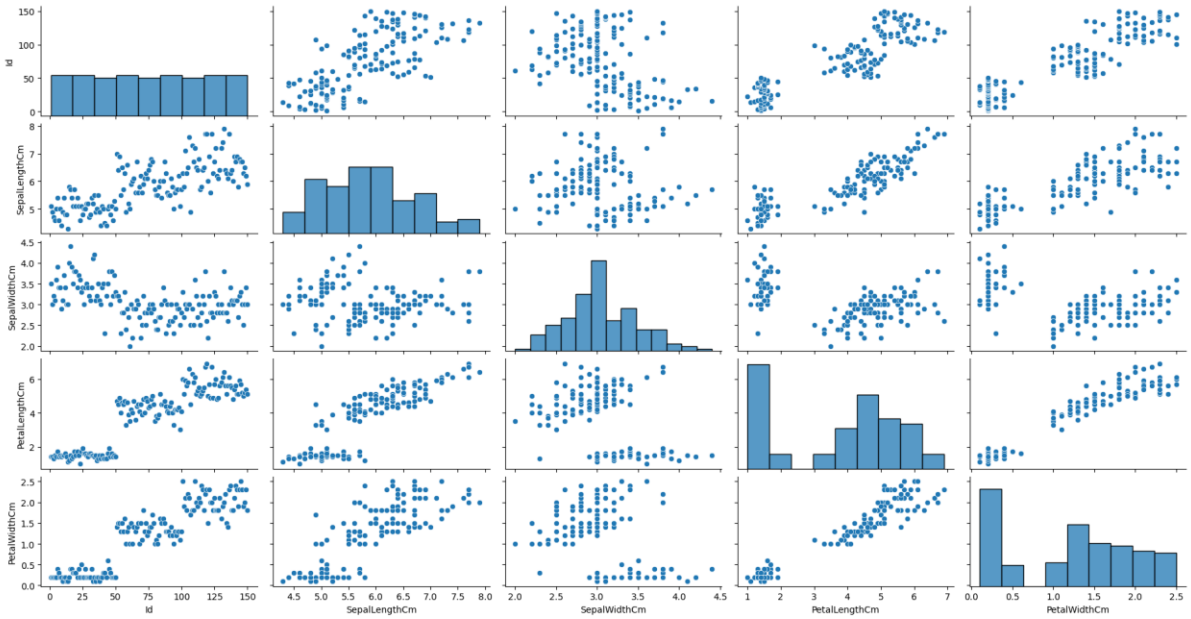
4. Use the ‘eps’ value in (2.) and find the most optimal value of ‘min_samples’ using silhouette score.

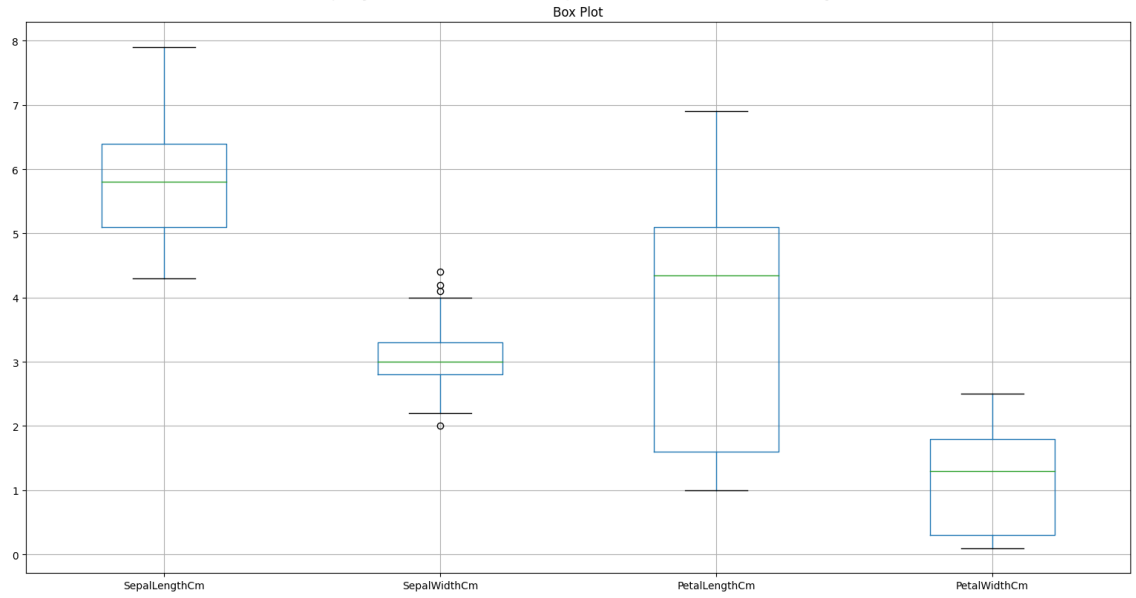
```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import DBSCAN

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

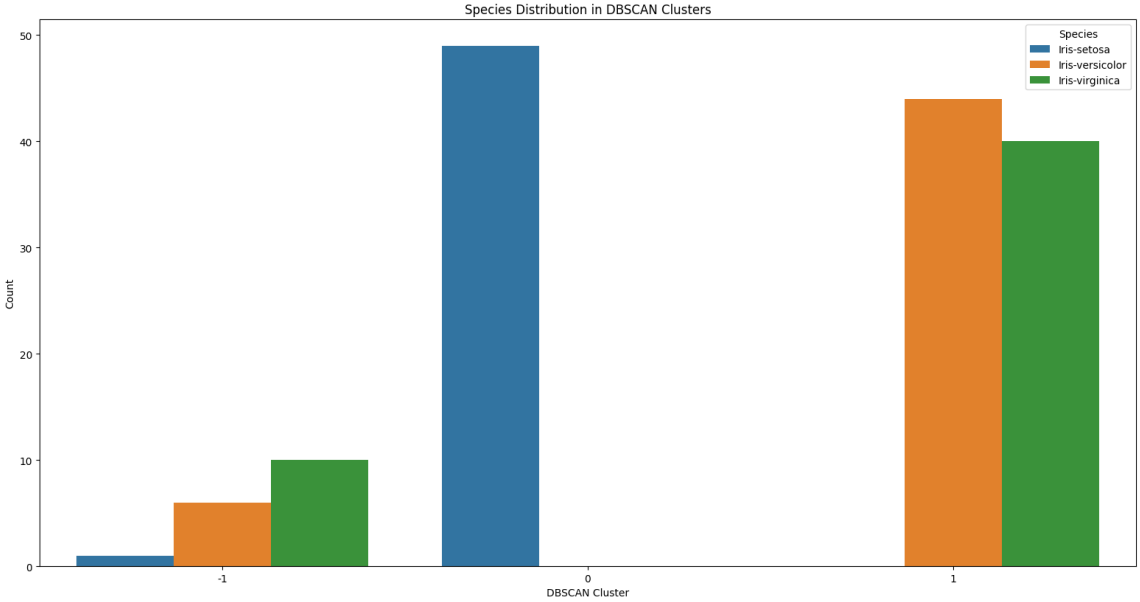
```
plt.show()
# calculate missing-value percentage
missing_value_percent = df.isna().sum() / len(df) *
100
print(missing_value_percent)
columns = ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']
df.boxplot(column=['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm'])
plt.title('Box 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))
# Scale the data
df[columns] =
MinMaxScaler().fit_transform(df[columns])
print(df.info())
X = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
# Fit the data to the DBSCAN model
cluster_labels = DBSCAN().fit_predict(X)
# Add the cluster labels to the original DataFrame
df['DBSCAN_Cluster'] = cluster_labels
# Show the species distribution in each of the
default clusters
species_distribution =
```

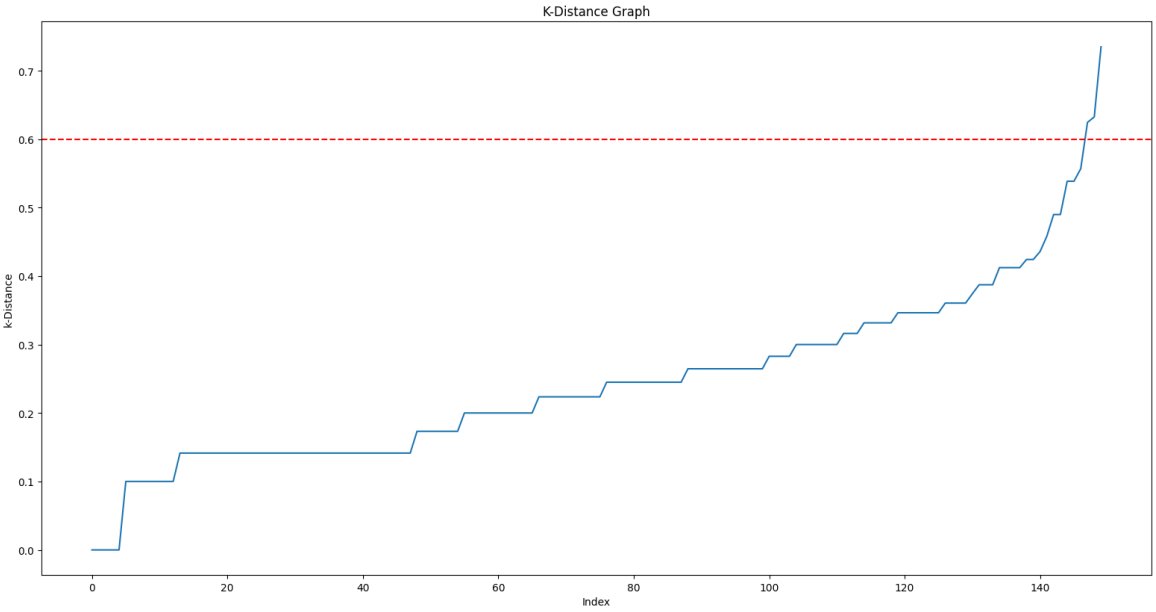
```
df.groupby(['DBSCAN_Cluster',
'Species']).size().reset_index(name='Count')
# Plot the distribution
sns.barplot(x='DBSCAN_Cluster', y='Count',
hue='Species', data=species_distribution)
plt.xlabel('DBSCAN Cluster')
plt.ylabel('Count')
plt.title('Species Distribution in DBSCAN
Clusters')
plt.show()
k = 7
nn = NearestNeighbors(n_neighbors=k).fit(X)
distances, indices = nn.kneighbors(X)
distances = np.sort(distances, axis=0)[:, 1]
plt.plot(distances)
plt.axhline(y=0.6, color='r', ls="--")
plt.xlabel('Index')
plt.ylabel('k-Distance')
plt.title('K-Distance Graph')
plt.show()
optimal_eps = float(input("Enter the optimal eps
value based on the plot: "))
dbscan = DBSCAN(eps=optimal_eps, min_samples=3)
cluster_labels = dbscan.fit_predict(X)
df['DBSCAN_Cluster'] = cluster_labels
species_distribution =
df.groupby(['DBSCAN_Cluster',
'Species']).size().reset_index(name='Count')
sns.barplot(x='DBSCAN_Cluster', y='Count',
hue='Species', data=species_distribution)
plt.xlabel('DBSCAN Cluster')
plt.ylabel('Count')
plt.title('Species Distribution in DBSCAN
Clusters')
plt.show()
print(f"Optimal eps value: {optimal_eps:.3f}")
min_samples_values = range(2, 11)
silhouette_scores = []
for min_samples in min_samples_values:
    dbscan = DBSCAN(eps=optimal_eps,
min_samples=min_samples)
    cluster_labels = dbscan.fit_predict(X)
```

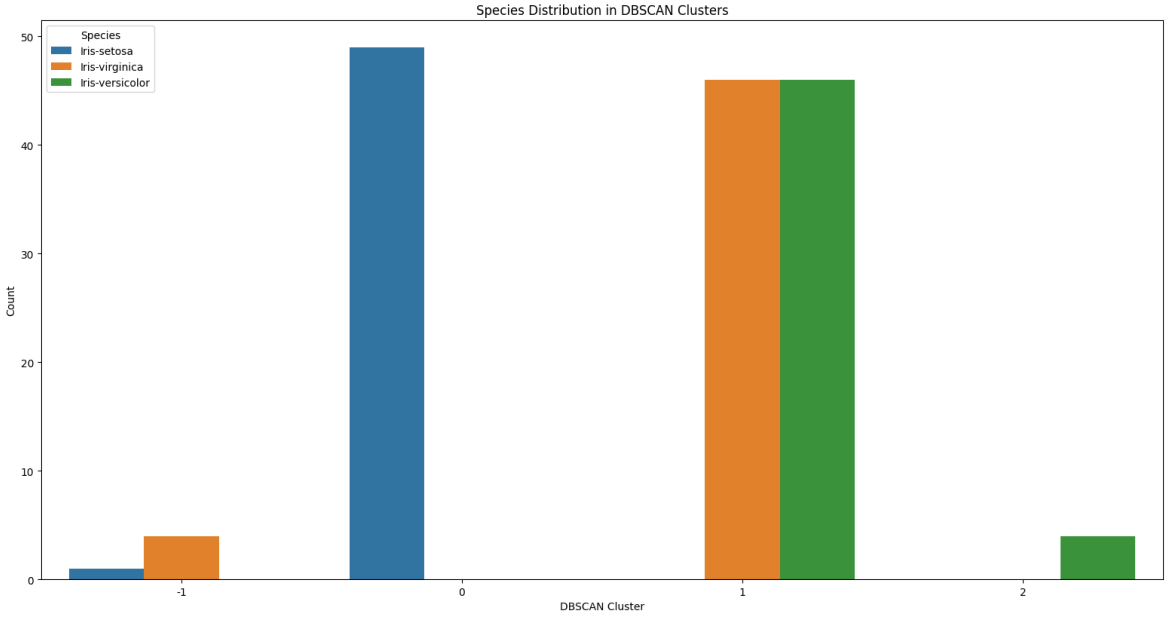
```
    # Check for single cluster case, as
silhouette_score requires at least 2 clusters
    if len(set(cluster_labels)) > 1:

silhouette_scores.append(silhouette_score(X,
cluster_labels))
    else:
        silhouette_scores.append(-1)
plt.plot(min_samples_values, silhouette_scores)
plt.xlabel('min_samples')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score for Different
min_samples')
plt.grid()
plt.show()
# Find the most optimal 'min_samples' value that
gives the highest silhouette score
optimal_min_samples =
min_samples_values[np.argmax(silhouette_scores)]
print(f"Optimal min_samples value:
{optimal_min_samples}")
```

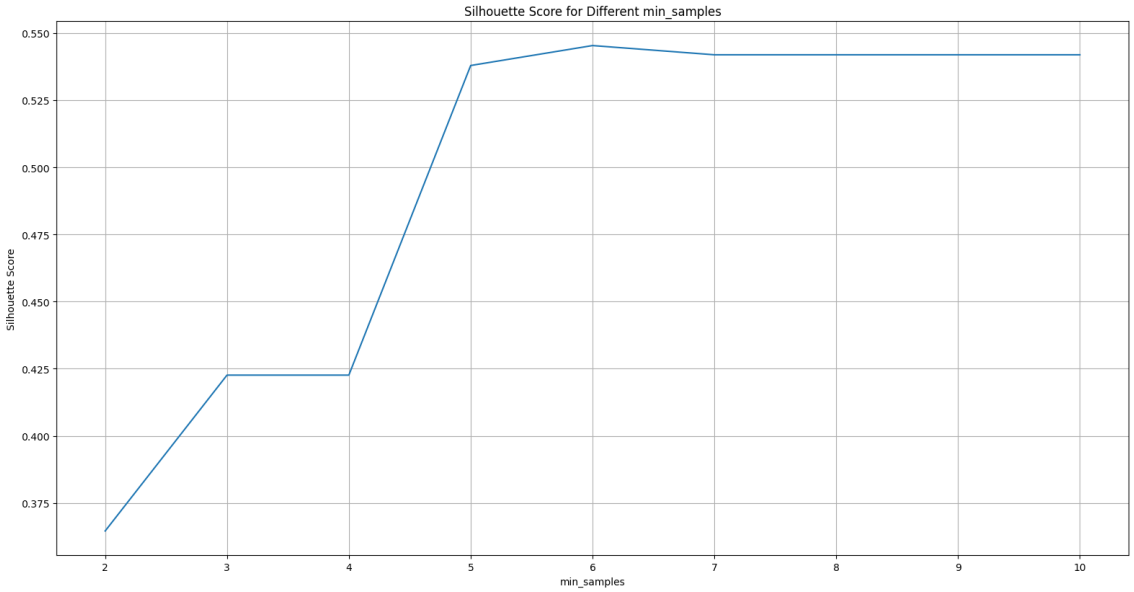

Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Species Distribution in DBSCAN Clusters
Species
Iris-setosa
Iris-versicolor
Iris-virginica
Count
DBSCAN Cluster
0
10
20
30
40
50
-1
0
1


K-Distance Graph
k-Distance
Index
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
120
140


Species Distribution in DBSCAN Clusters
Species
Iris-setosa
Iris-virginica
Iris-versicolor
Count
DBSCAN Cluster
0
10
20
30
40
50
-1
0
1
2



File - Day14Q4

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\
tejas\PycharmProjects\pythonProject\START\Day14Q4.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
Enter the optimal eps value based on the plot: 0.6
Optimal eps value: 0.600
Optimal min_samples value: 6

Process finished with exit code 0

```

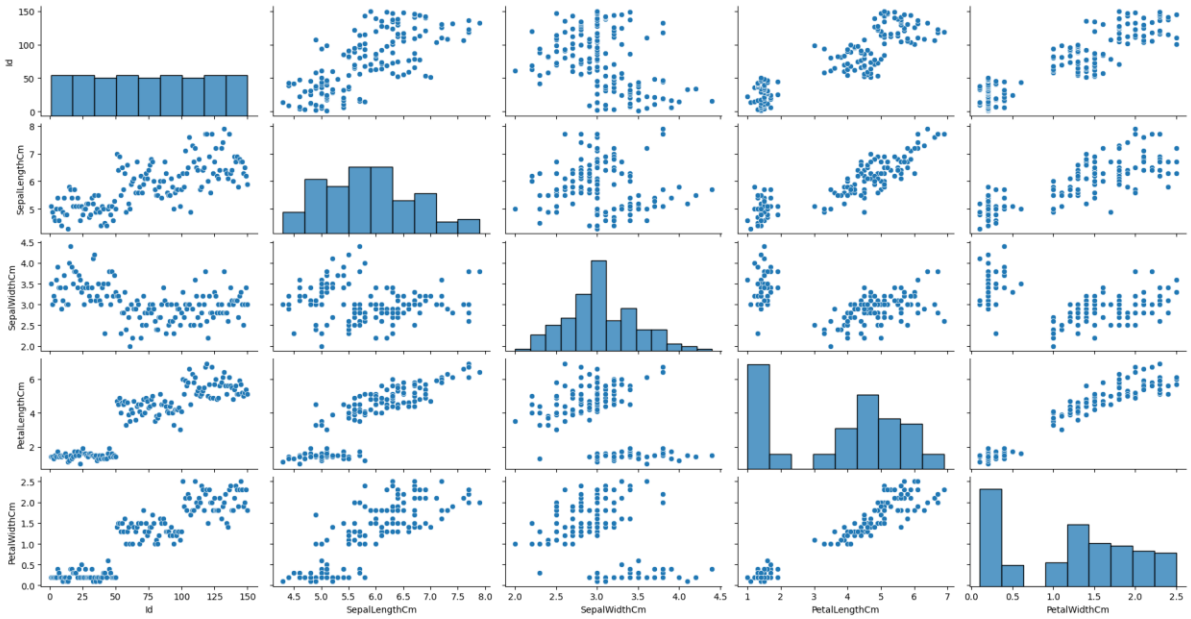
5. Find all the outliers using the DBSCAN algorithm.

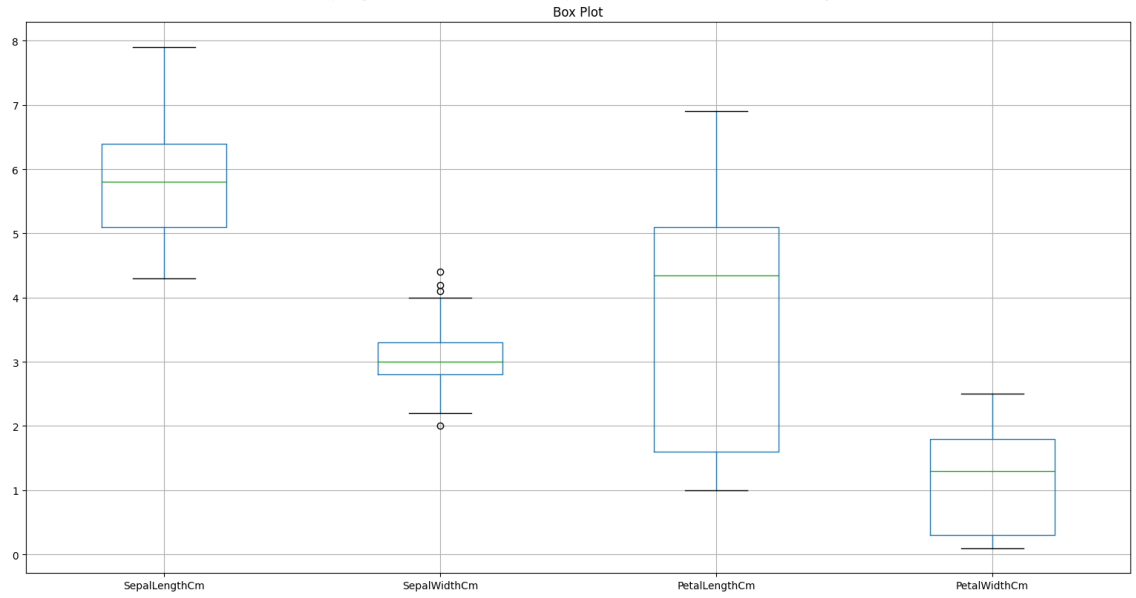
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import NearestNeighbors
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score

# 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)
columns = ['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']
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)]
```

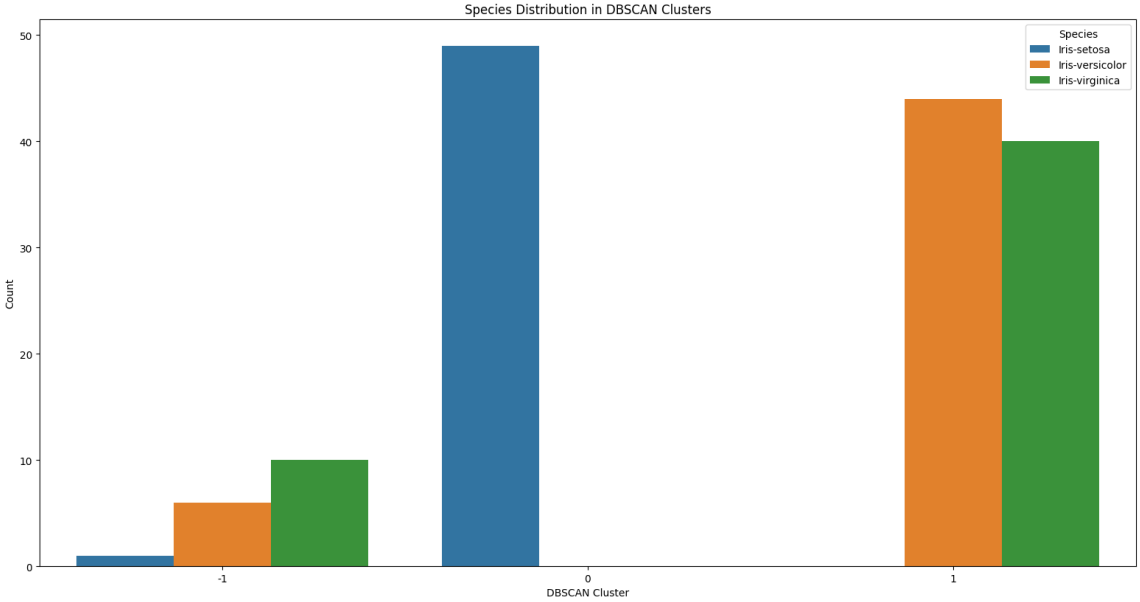
```
    print(len(OutlierData))
# Scale the data
# df[columns] =
MinMaxScaler().fit_transform(df[columns])
# print(df.info())
X = df[['SepalLengthCm', 'SepalWidthCm',
'PetalLengthCm', 'PetalWidthCm']]
# Fit the data to the DBSCAN model
cluster_labels = DBSCAN().fit_predict(X)
# Add the cluster labels to the original DataFrame
df['DBSCAN_Cluster'] = cluster_labels
# Show the species distribution in each of the
default clusters
species_distribution =
df.groupby(['DBSCAN_Cluster',
'Species']).size().reset_index(name='Count')
# Plot the distribution
sns.barplot(x='DBSCAN_Cluster', y='Count',
hue='Species', data=species_distribution)
plt.xlabel('DBSCAN Cluster')
plt.ylabel('Count')
plt.title('Species Distribution in DBSCAN
Clusters')
plt.show()
k = 7
nn = NearestNeighbors(n_neighbors=k).fit(X)
distances, indices = nn.kneighbors(X)
distances = np.sort(distances, axis=0)[:, 1]
plt.plot(distances)
plt.axhline(y=0.6, color='r', ls="--")
plt.xlabel('Index')
plt.ylabel('k-Distance')
plt.title('K-Distance Graph')
plt.show()
optimal_eps = float(input("Enter the optimal eps
value based on the plot: "))
dbscan = DBSCAN(eps=optimal_eps, min_samples=3)
cluster_labels = dbscan.fit_predict(X)
df['DBSCAN_Cluster'] = cluster_labels
species_distribution =
df.groupby(['DBSCAN_Cluster',
'Species']).size().reset_index(name='Count')
```

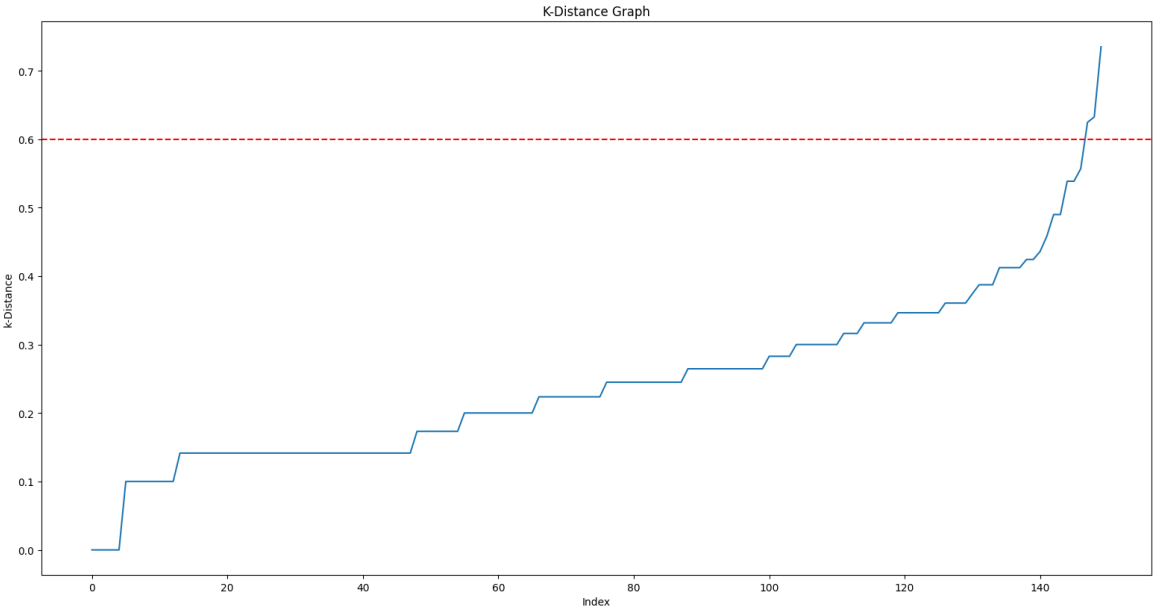
```
sns.barplot(x='DBSCAN_Cluster', y='Count',
hue='Species', data=species_distribution)
plt.xlabel('DBSCAN Cluster')
plt.ylabel('Count')
plt.title('Species Distribution in DBSCAN
Clusters')
plt.show()
print(f"Optimal eps value: {optimal_eps:.3f}")
min_samples_values = range(2, 11)
silhouette_scores = []
for min_samples in min_samples_values:
    dbscan = DBSCAN(eps=optimal_eps,
min_samples=min_samples)
    cluster_labels = dbscan.fit_predict(X)
    # Check for single cluster case, as
silhouette_score requires at least 2 clusters
    if len(set(cluster_labels)) > 1:

silhouette_scores.append(silhouette_score(X,
cluster_labels))
    else:
        silhouette_scores.append(-1)
plt.plot(min_samples_values, silhouette_scores)
plt.xlabel('min_samples')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score for Different
min_samples')
plt.grid()
plt.show()
# Find the most optimal 'min_samples' value that
gives the highest silhouette score
optimal_min_samples =
min_samples_values[np.argmax(silhouette_scores)]
print(f"Optimal min_samples value:
{optimal_min_samples}")
dbscan = DBSCAN(eps=optimal_eps, min_samples=3)
cluster_labels = dbscan.fit_predict(X)
df['DBSCAN_Cluster'] = cluster_labels
species_distribution =
df.groupby(['DBSCAN_Cluster',
'Species']).size().reset_index(name='Count')
sns.barplot(x='DBSCAN_Cluster', y='Count',
```

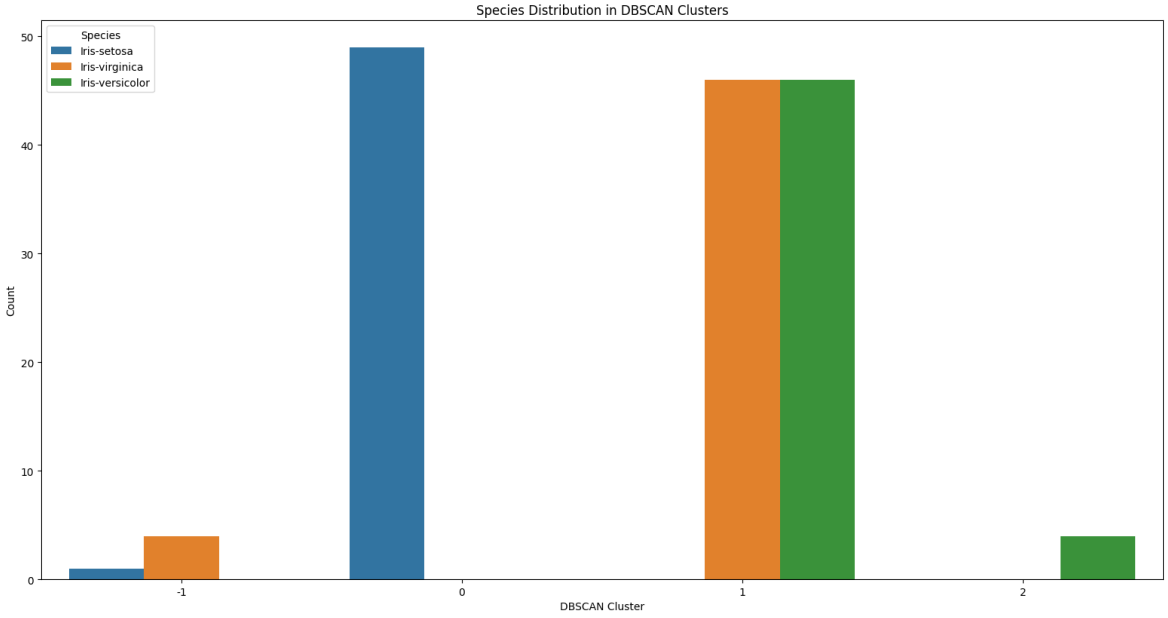
```
hue='Species', data=species_distribution)
plt.xlabel('DBSCAN Cluster')
plt.ylabel('Count')
plt.title('Species Distribution in DBSCAN
Clusters')
plt.show()

# Identify outliers with cluster label '-1'
outliers = X[cluster_labels == -1]
print("Outliers:")
print(outliers)
```

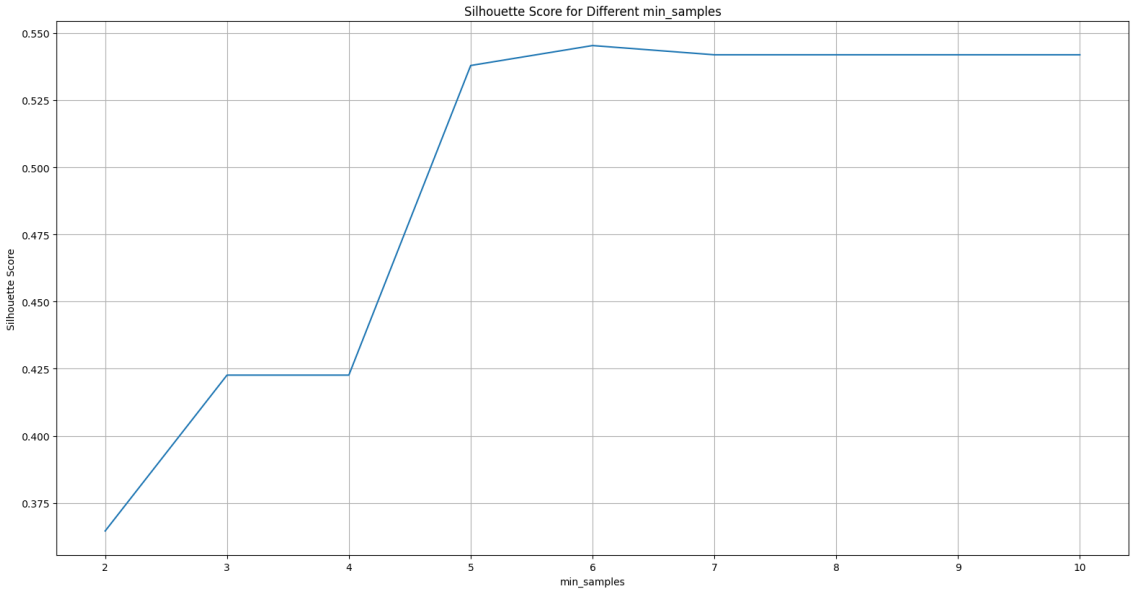

Id
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm

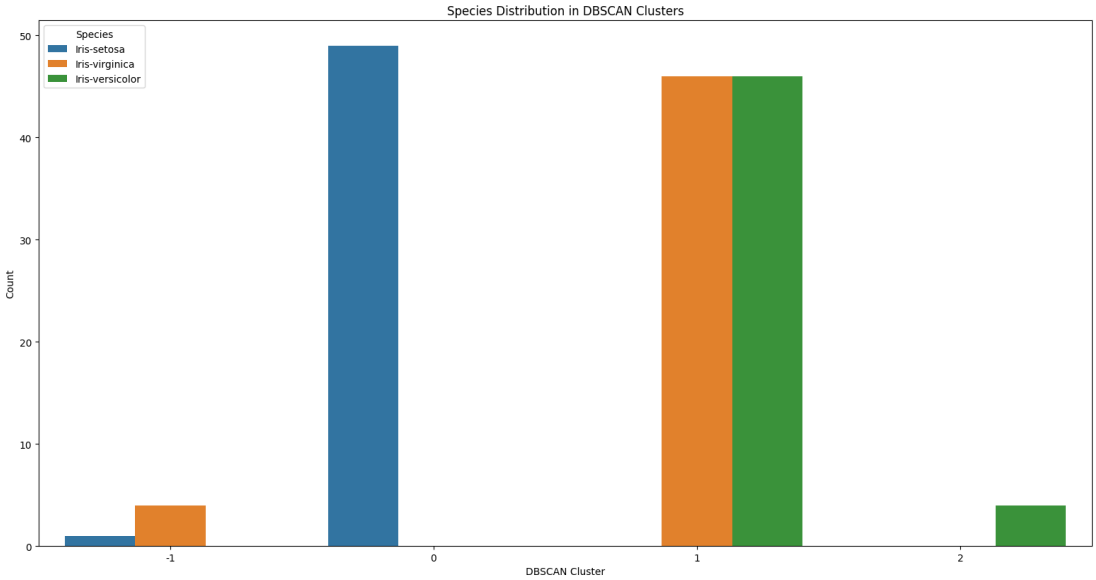

Box Plot
SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm


Species Distribution in DBSCAN Clusters
Species
Iris-setosa
Iris-versicolor
Iris-virginica
Count
DBSCAN Cluster
0
10
20
30
40
50
-1
0
1


K-Distance Graph
k-Distance
Index
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
120
140


Species Distribution in DBSCAN Clusters
Species
Iris-setosa
Iris-virginica
Iris-versicolor
Count
DBSCAN Cluster
0
10
20
30
40
50
-1
0
1
2


Silhouette Score for Different min_samples
Silhouette Score
0.550
0.525
0.500
0.475
0.450
0.425
0.400
0.375
2
3
4
5
6
7
8
9
10
min_samples


Species Distribution in DBSCAN Clusters
Species
Iris-setosa
Iris-virginica
Iris-versicolor
Count
50
40
30
20
10
0
-1
0
1
2
DBSCAN Cluster

```
C:\Users\tejas\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\tejas\PycharmProjects\pythonProject\START\Day14Q5.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
Enter the optimal eps value based on the plot: 0.6
Optimal eps value: 0.600
Optimal min_samples value: 6
Outliers:
      SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm
41              4.5           2.3            1.3           0.3
106             4.9           2.5            4.5           1.7
109             7.2           3.6            6.1           2.5
117             7.7           3.8            6.7           2.2
131             7.9           3.8            6.4           2.0

Process finished with exit code 0

```