```
import pandas as pd
df = pd.read_csv('/content/Iris.csv')
```

```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

```
df.head()
```

```
   Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm      Species
0   1            5.1           3.5            1.4           0.2  Iris-setosa
1   2            4.9           3.0            1.4           0.2  Iris-setosa
2   3            4.7           3.2            1.3           0.2  Iris-setosa
3   4            4.6           3.1            1.5           0.2  Iris-setosa
4   5            5.0           3.6            1.4           0.2  Iris-setosa
```

Next steps: Generate code with df   View recommended plots

```
df.shape
```

```
(150, 6)
```

```
df.info()
```

```
<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
```

```
df.isnull().sum()
```

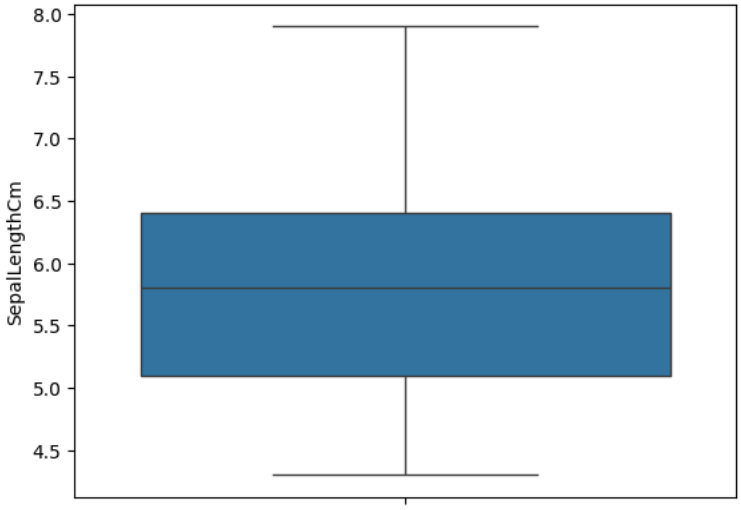
```
Id               0
SepalLengthCm    0
SepalWidthCm     0
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
df[df.duplicated()]
```

```
Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  Species
```

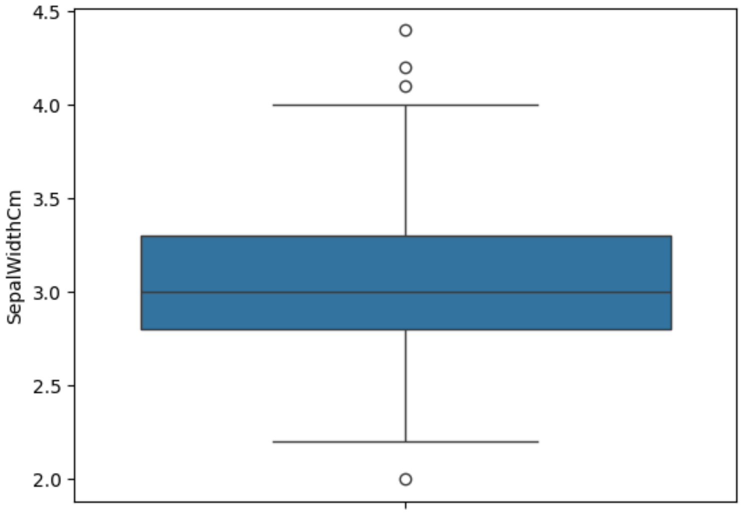
```
import seaborn as sns
sns.boxplot(df['SepalLengthCm'])
```

```
<Axes: ylabel='SepalLengthCm'>
```



```
sns.boxplot(df['SepalWidthCm'])
```

```
<Axes: ylabel='SepalWidthCm'>
```



```
q1 = df['SepalWidthCm'].quantile(0.25)
print(q1)
```

```
2.8
```

```
q3 = df['SepalWidthCm'].quantile(0.75)
q3
```

```
3.3
```

```
IQR = q3-q1
IQR
```

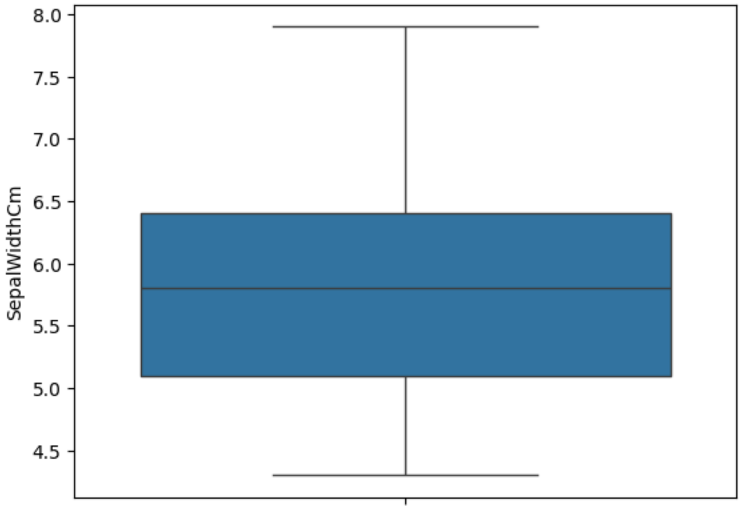
```
0.5
```

```
lowerBound = q1 -(1.5*IQR)
print(lowerBound)
upperBound = q3 + (1.5*IQR)
print(upperBound)
```

```
2.05
4.05
```

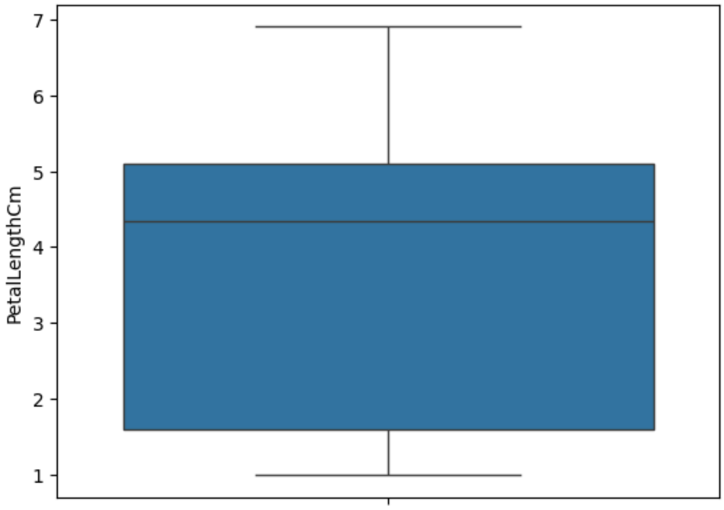
```
import numpy as np
df['SepalWidthCm']=np.where(df['SepalWidthCm']<upperBound,upperBound,df['SepalWidthCm'])
sns.boxplot(df['SepalWidthCm'])
```

```
<Axes: ylabel='SepalWidthCm'>
```



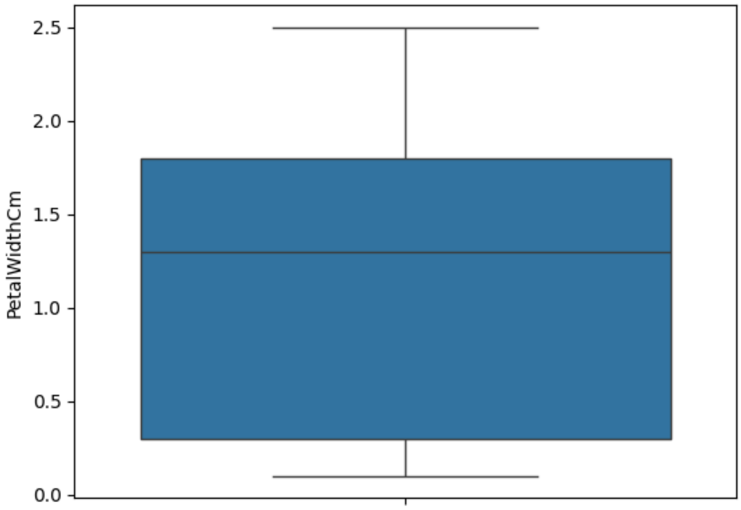
```
sns.boxplot(df['PetalLengthCm'])
```

```
<Axes: ylabel='PetalLengthCm'>
```



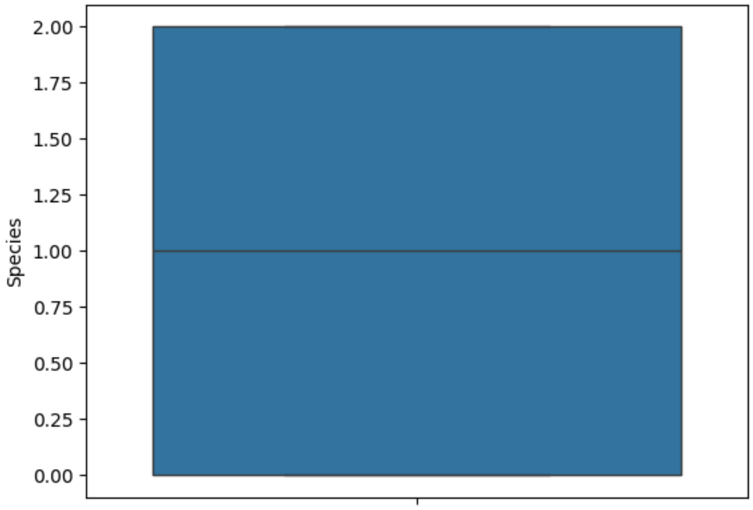
```
sns.boxplot(df['PetalWidthCm'])
```

```
<Axes: ylabel='PetalWidthCm'>
```



```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Species']=le.fit_transform(df[['Species']])
sns.boxplot(df['Species'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_label.py:116: DataConv
  y = column_or_1d(y, warn=True)
<Axes: ylabel='Species'>
```



```
from sklearn.preprocessing import StandardScaler
Scaler = StandardScaler()
df['Id']=Scaler.fit_transform(df[['Id']])
df['SepalLengthCm']=Scaler.fit_transform(df[['SepalLengthCm']])
df['SepalWidthCm']=Scaler.fit_transform(df[['SepalWidthCm']])
df['PetalLengthCm']=Scaler.fit_transform(df[['PetalLengthCm']])
df['PetalWidthCm']=Scaler.fit_transform(df[['PetalWidthCm']])
```

```
df.head()
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	-1.720542	-0.900681	-0.900681	-1.341272	-1.312977	0
1	-1.697448	-1.143017	-1.143017	-1.341272	-1.312977	0
2	-1.674353	-1.385353	-1.385353	-1.398138	-1.312977	0
3	-1.651258	-1.506521	-1.506521	-1.284407	-1.312977	0
4	-1.628164	-1.021849	-1.021849	-1.341272	-1.312977	0

Next steps: Generate code with df View recommended plots

```
x= df.iloc[:,0:-1]
x
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
0	-1.720542	-0.900681	-0.900681	-1.341272	-1.312977
1	-1.697448	-1.143017	-1.143017	-1.341272	-1.312977
2	-1.674353	-1.385353	-1.385353	-1.398138	-1.312977
3	-1.651258	-1.506521	-1.506521	-1.284407	-1.312977
4	-1.628164	-1.021849	-1.021849	-1.341272	-1.312977
...	...	...	...	...	...
145	1.628164	1.038005	1.038005	0.819624	1.447956
146	1.651258	0.553333	0.553333	0.705893	0.922064
147	1.674353	0.795669	0.795669	0.819624	1.053537
148	1.697448	0.432165	0.432165	0.933356	1.447956
149	1.720542	0.068662	0.068662	0.762759	0.790591

150 rows × 5 columns

Next steps: Generate code with x   View recommended plots

```
y = df.iloc[:,-1]
y
```

```
0      0
1      0
2      0
3      0
4      0
      ..
145    2
146    2
147    2
148    2
149    2
Name: Species, Length: 150, dtype: int64
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=42)
```

```
x_train.shape
```

```
(120, 5)
```

```
x_test.shape
```

```
(30, 5)
```

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion='entropy', max_depth=3)
dtc.fit(x_train,y_train)
```

```
▸ DecisionTreeClassifier
```

```
y_pred = dtc.predict(x_test)
y_pred
```

```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0])
```

+ Code   + Text

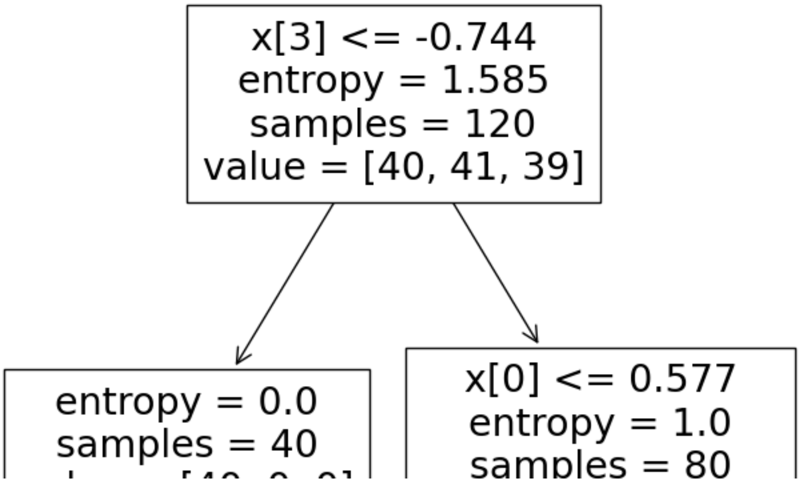
```
dtc.predict([[151,0.5,1,2,1.5]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionTreeClas
  warnings.warn(
array([2])
```

```
from sklearn.metrics import accuracy_score
accuracy_score(y_pred,y_test)
```

```
1.0
```

```
import matplotlib.pyplot as plt
from sklearn import tree
plt.figure(figsize=(10,10))
tree.plot_tree(dtc.fit(x_train,y_train))
plt.show()
```


x[3] <= -0.744
entropy = 1.585
samples = 120
value = [40, 41, 39]
entropy = 0.0
samples = 40
value = [40, 0, 0]
x[0] <= 0.577
entropy = 1.0
samples = 80