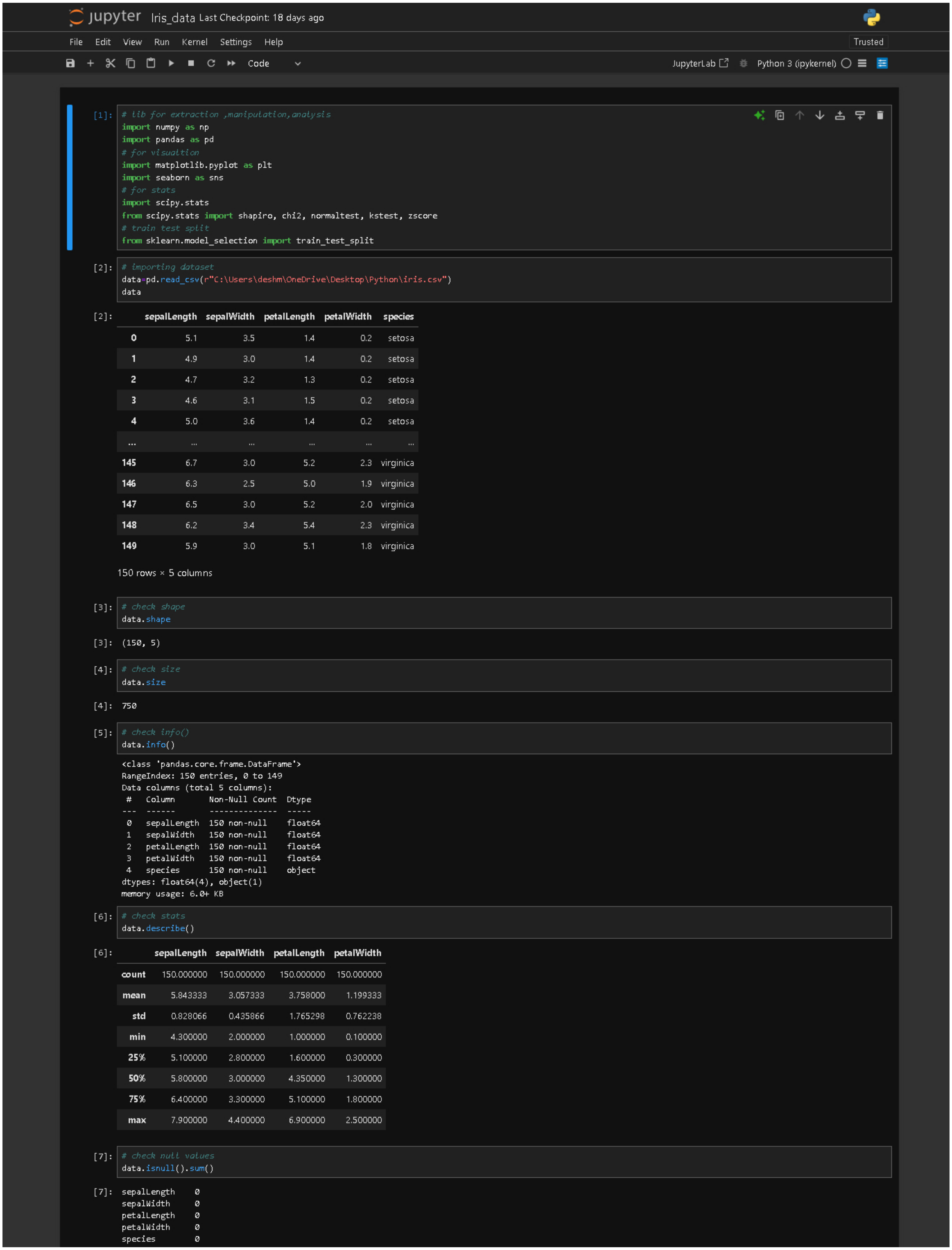


[1]:
```python
# lib for extraction ,manipulation,analysis
import numpy as np
import pandas as pd
# for visualtion
import matplotlib.pyplot as plt
import seaborn as sns
# for stats
import scipy.stats
from scipy.stats import shapiro, chi2, normaltest, kstest, zscore
# train test split
from sklearn.model_selection import train_test_split
```

[2]:
```python
# importing dataset
data=pd.read_csv(r"C:\Users\deshm\OneDrive\Desktop\Python\iris.csv")
data
```

[2]:

	sepalLength	sepalWidth	petalLength	petalWidth	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

[3]:
```python
# check shape
data.shape
```

[3]:
```
(150, 5)
```

[4]:
```python
# check size
data.size
```

[4]:
```
750
```

[5]:
```python
# check info()
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sepalLength  150 non-null    float64
 1   sepalWidth   150 non-null    float64
 2   petalLength  150 non-null    float64
 3   petalWidth   150 non-null    float64
 4   species      150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

[6]:
```python
# check stats
data.describe()
```

[6]:

	sepalLength	sepalWidth	petalLength	petalWidth
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

[7]:
```python
# check null values
data.isnull().sum()
```

[7]:
```
sepalLength    0
sepalWidth     0
petalLength    0
petalWidth     0
species        0
```

```
dtype: int64
```

[8]:
```
# analysis of sepal_length
mean= data["sepalLength"].mean()
print(mean)
```
```
5.843333333333334
```

[9]:
```
median= data["sepalLength"].median()
print(median)
```
```
5.8
```

[10]:
```
mode= data["sepalLength"].mode()[0]
print(mode)
```
```
5.0
```

[11]:
```
var=data["sepalLength"].var()
print(var)
```
```
0.6856935123042505
```

[12]:
```
std=data["sepalLength"].std
print(std)
```
```
<bound method Series.std of 0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: sepalLength, Length: 150, dtype: float64>
```

[13]:
```
skew=data["sepalLength"].skew()
print(skew)
```
```
0.3149109566369728
```

[14]:
```
# analysis of SepalLength
mean= data["petalLength"].mean()
print(mean)
```
```
3.7580000000000005
```

[15]:
```
median= data["petalLength"].median()
print(median)
```
```
4.35
```

[16]:
```
mode= data["petalLength"].mode()[0]
print(mode)
```
```
1.4
```

[17]:
```
var=data["petalLength"].var()
print(var)
```
```
3.1162778523489942
```

[18]:
```
std=data["petalLength"].std
print(std)
```
```
<bound method Series.std of 0      1.4
1      1.4
2      1.3
3      1.5
4      1.4
      ...
145    5.2
146    5.0
147    5.2
148    5.4
149    5.1
Name: petalLength, Length: 150, dtype: float64>
```

[19]:
```
skew=data["petalLength"].skew()
print(skew)
```
```
-0.27488417975101276
```

[20]:
```
# analysis of petalwidth
mean= data["petalWidth"].mean()
print(mean)
```
```
1.1993333333333336
```

[21]:
```
median= data["petalWidth"].median()
print(median)
```
```
1.3
```

[22]:
```
mode= data["petalWidth"].mode()
print(mode)
```
```
0    0.2
Name: petalWidth, dtype: float64
```

[23]:
```
var= data["petalWidth"].var()
print(var)
```
```
0.5810062639821029
```

[24]:
```
std= data["petalWidth"].std()
```

```
print(std)
```

```
0.7622376689603465
```

[25]:
```
skew= data["petalWidth"].skew()
print(skew)
```

```
-0.10296674764898116
```

[26]:
```
# analysis of specie
unique_species=data["species"].unique()
unique_space_count=data["species"].value_counts()
print(unique_species)
print(unique_space_count)
```

```
['setosa' 'versicolor' 'virginica']
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

[27]:
```
# Bivariate analysis-covariance between sapallength and sepalwidth column
covariance=data.iloc[:,[0,1]].cov()
covariance
```

[27]:

	sepalLength	sepalWidth
sepalLength	0.685694	-0.042434
sepalWidth	-0.042434	0.189979

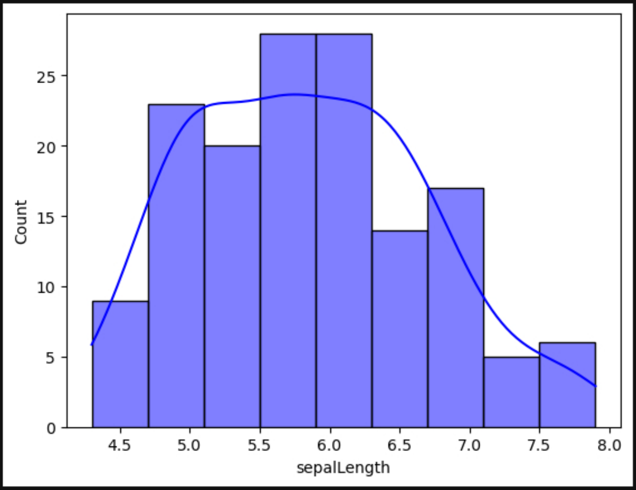
[28]:
```
# Bivariate analysis-covariance between patellength and petalwidth column
covariance=data.iloc[:,[2,3]].cov()
covariance
```

[28]:

	petalLength	petalWidth
petalLength	3.116278	1.295609
petalWidth	1.295609	0.581006

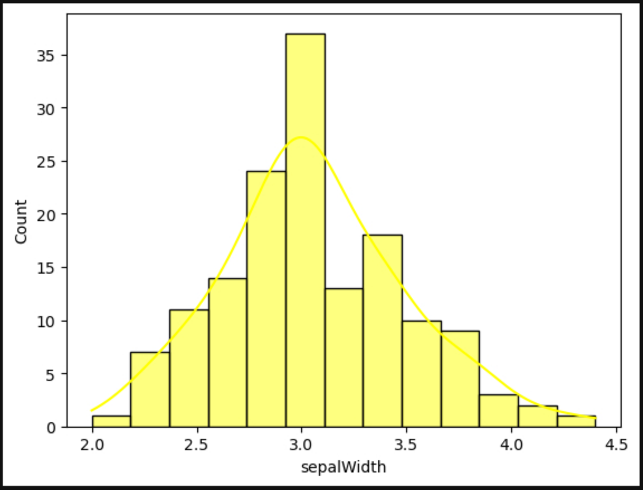
[29]:
```
sns.histplot(data=data,x="sepalLength",kde=True,color="blue")
```

[29]:
```
<Axes: xlabel='sepalLength', ylabel='Count'>
```



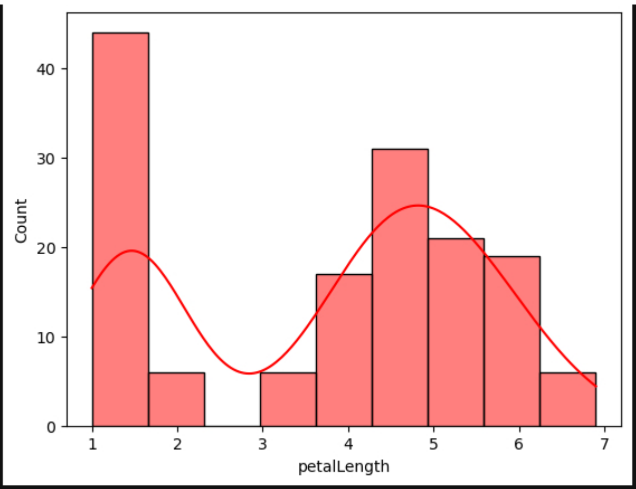
[30]:
```
sns.histplot(data=data,x="sepalWidth",kde=True,color="yellow")
```

[30]:
```
<Axes: xlabel='sepalWidth', ylabel='Count'>
```



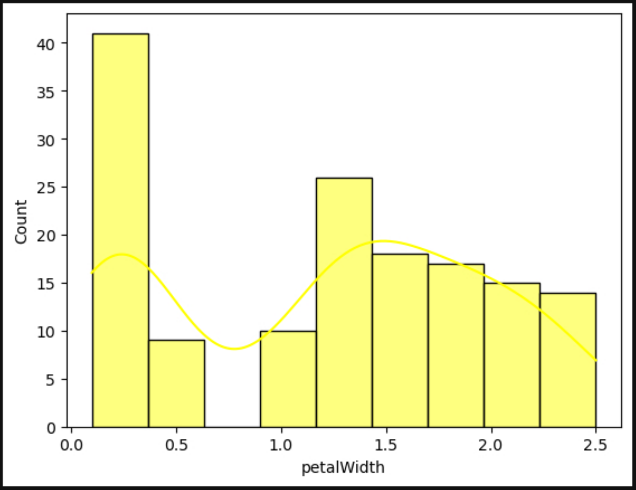
[31]:
```
sns.histplot(data=data,x="petalLength",kde=True,color="red")
```

[31]:
```
<Axes: xlabel='petalLength', ylabel='Count'>
```



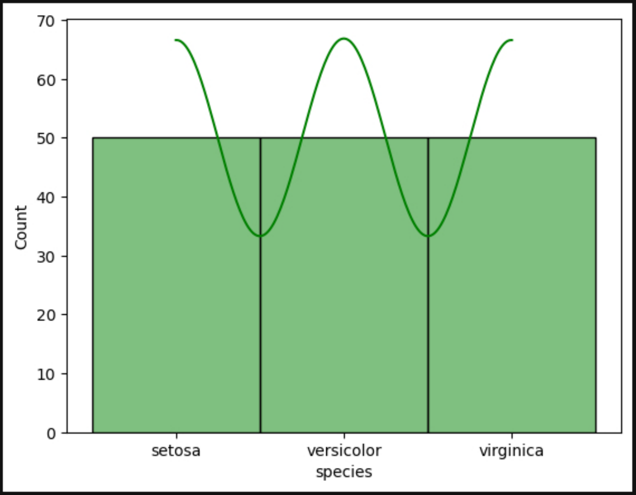
```
[32]: sns.histplot(data=data,x="petalWidth",kde=True,color="yellow")
```

```
[32]: <Axes: xlabel='petalWidth', ylabel='Count'>
```



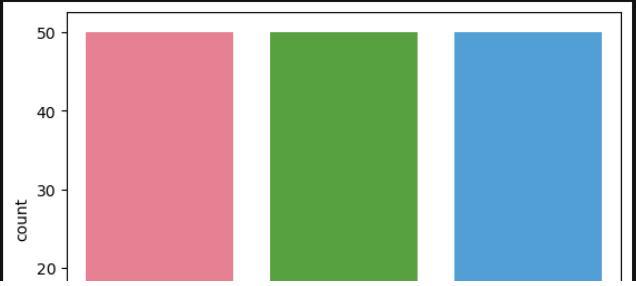
```
[33]: sns.histplot(data=data,x="species",kde=True,color="green")
```

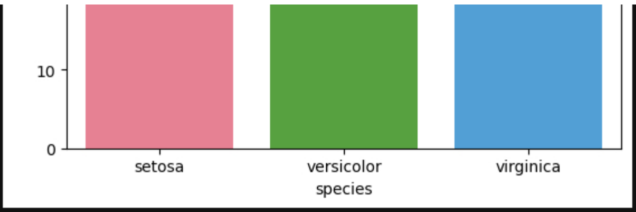
```
[33]: <Axes: xlabel='species', ylabel='Count'>
```



```
[34]: sns.countplot(data=data,x="species",hue="species",palette="husl")
      plt.show
```

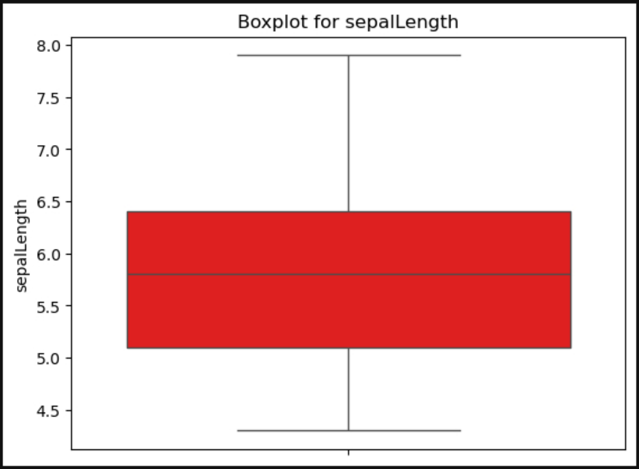
```
[34]: <function matplotlib.pyplot.show(close=None, block=None)>
```





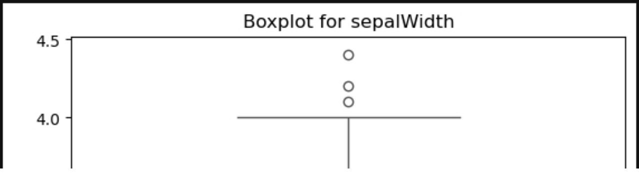
```
[35]: # checking and handling of outliers

def Checking_and_Handling_Of_Outliers(data, col):
    sns.boxplot(data[col], color = "Red")
    plt.title(f"Boxplot for {col}")
    plt.show()

    q1 = data[col].quantile(0.25)
    q3 = data[col].quantile(0.75)

    iqr = q3 - q1

    LowerTail = q1 - 1.5*iqr
    UpperTail = q3 + 1.5*iqr

    print(f"25% Quantile q1 = {q1}\n75% Quantile q3 = {q3}\nIQR = {iqr}\n")
    print("-"*80)
    print(f"Lower Tail = {LowerTail}\nUpper Tail = {UpperTail}")
    print("-"*80)

    # Checking for Outliers
    Outliers = data[(data[col] < LowerTail) | (data[col] > UpperTail)]
    print("\nOutliers :\n",Outliers)
    print("-"*80)

    #HAndling of Outliers :
    data.loc[data[col] < LowerTail, col] = LowerTail # all outliers less than lowertail, assigned by lowertail value
    data.loc[data[col] > UpperTail, col] = UpperTail # all outliers greater than uppertail, assigned by uppertail value

    print("After handling of Outliers data:\n")
    print(data.head())
```

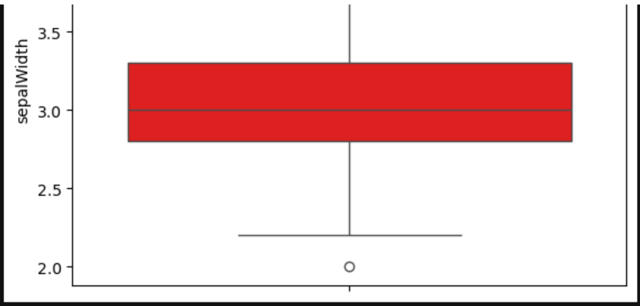
```
[36]: Checking_and_Handling_Of_Outliers(data,"sepalLength")
```



```
25% Quantile q1 = 5.1
75% Quantile q3 = 6.4
IQR = 1.3000000000000007

--------------------------------------------------------------------------------
Lower Tail = 3.1499999999999986
Upper Tail = 8.350000000000001
--------------------------------------------------------------------------------

Outliers :
 Empty DataFrame
Columns: [sepalLength, sepalWidth, petalLength, petalWidth, species]
Index: []
--------------------------------------------------------------------------------
After handling of Outliers data:

   sepalLength  sepalWidth  petalLength  petalWidth species
0          5.1         3.5          1.4         0.2  setosa
1          4.9         3.0          1.4         0.2  setosa
2          4.7         3.2          1.3         0.2  setosa
3          4.6         3.1          1.5         0.2  setosa
4          5.0         3.6          1.4         0.2  setosa
```

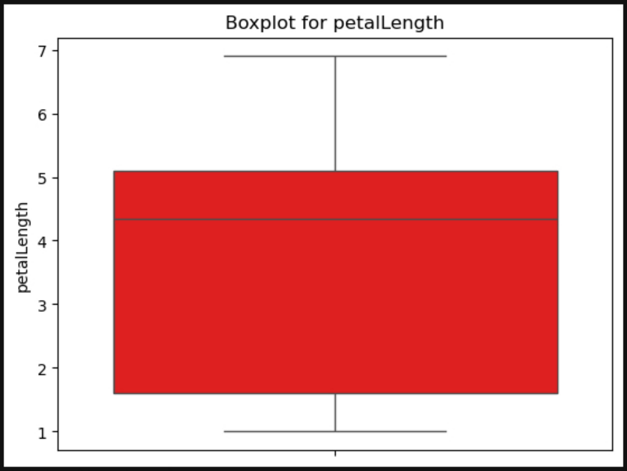
```
[37]: Checking_and_Handling_Of_Outliers(data,"sepalWidth")
```





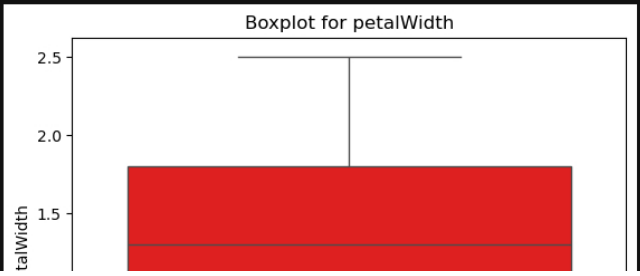
```
25% Quantile q1 = 2.8
75% Quantile q3 = 3.3
IQR = 0.5

--------------------------------------------------------------------------------
Lower Tail = 2.05
Upper Tail = 4.05
--------------------------------------------------------------------------------

Outliers :
     sepalLength  sepalWidth  petalLength  petalWidth     species
15          5.7         4.4          1.5         0.4      setosa
32          5.2         4.1          1.5         0.1      setosa
33          5.5         4.2          1.4         0.2      setosa
60          5.0         2.0          3.5         1.0  versicolor
--------------------------------------------------------------------------------
After handling of Outliers data:

   sepalLength  sepalWidth  petalLength  petalWidth species
0          5.1         3.5          1.4         0.2  setosa
1          4.9         3.0          1.4         0.2  setosa
2          4.7         3.2          1.3         0.2  setosa
3          4.6         3.1          1.5         0.2  setosa
4          5.0         3.6          1.4         0.2  setosa
```

```
[38]: Checking_and_Handling_Of_Outliers(data,"petalLength")
```



```
25% Quantile q1 = 1.6
75% Quantile q3 = 5.1
IQR = 3.4999999999999996

--------------------------------------------------------------------------------
Lower Tail = -3.649999999999999
Upper Tail = 10.349999999999998
--------------------------------------------------------------------------------

Outliers :
 Empty DataFrame
Columns: [sepalLength, sepalWidth, petalLength, petalWidth, species]
Index: []
--------------------------------------------------------------------------------
After handling of Outliers data:

   sepalLength  sepalWidth  petalLength  petalWidth species
0          5.1         3.5          1.4         0.2  setosa
1          4.9         3.0          1.4         0.2  setosa
2          4.7         3.2          1.3         0.2  setosa
3          4.6         3.1          1.5         0.2  setosa
4          5.0         3.6          1.4         0.2  setosa
```

```
[39]: Checking_and_Handling_Of_Outliers(data,"petalWidth")
```



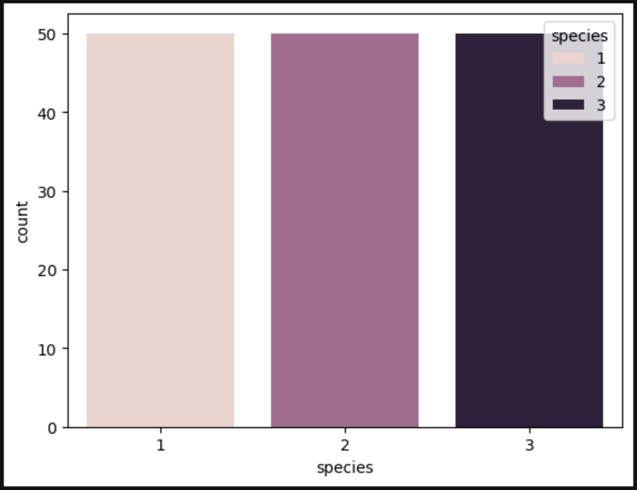
```
25% Quantile q1 = 0.3
75% Quantile q3 = 1.8
IQR = 1.5

--------------------------------------------------------------------------------
Lower Tail = -1.95
Upper Tail = 4.05
--------------------------------------------------------------------------------

Outliers :
 Empty DataFrame
Columns: [sepalLength, sepalWidth, petalLength, petalWidth, species]
Index: []
--------------------------------------------------------------------------------
After handling of Outliers data:

   sepalLength  sepalWidth  petalLength  petalWidth species
0          5.1         3.5          1.4         0.2  setosa
1          4.9         3.0          1.4         0.2  setosa
2          4.7         3.2          1.3         0.2  setosa
3          4.6         3.1          1.5         0.2  setosa
4          5.0         3.6          1.4         0.2  setosa
```

[40]:
```
data["species"].replace({"setosa":1,"versicolor":2,"virginica":3},inplace=True)
```

```
C:\Users\deshm\AppData\Local\Temp\ipykernel_31372\999452005.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chaine
d assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  data["species"].replace({"setosa":1,"versicolor":2,"virginica":3},inplace=True)
C:\Users\deshm\AppData\Local\Temp\ipykernel_31372\999452005.py:1: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a futu
re version. To retain the old behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the future behavior, set `pd.set_option('future.no_si
lent_downcasting', True)`
  data["species"].replace({"setosa":1,"versicolor":2,"virginica":3},inplace=True)
```

[41]:
```
sns.countplot(x=data["species"],hue=data["species"])
```

[41]: `<Axes: xlabel='species', ylabel='count'>`



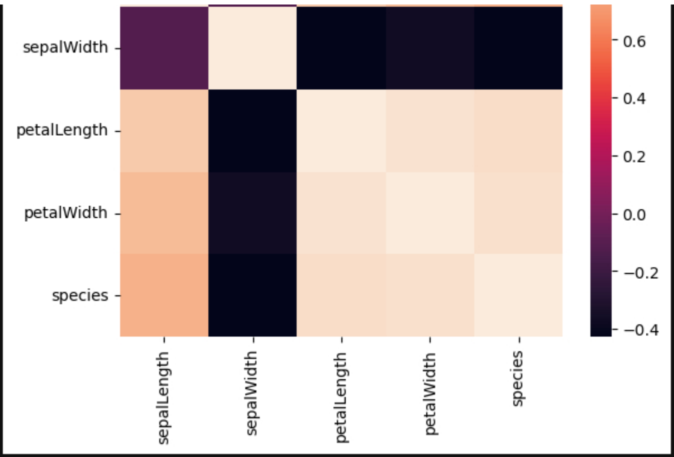
[42]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   sepalLength  150 non-null    float64
 1   sepalWidth   150 non-null    float64
 2   petalLength  150 non-null    float64
 3   petalWidth   150 non-null    float64
 4   species      150 non-null    int64  
dtypes: float64(4), int64(1)
memory usage: 6.0 KB
```

[43]:
```
corr=data.corr()
```

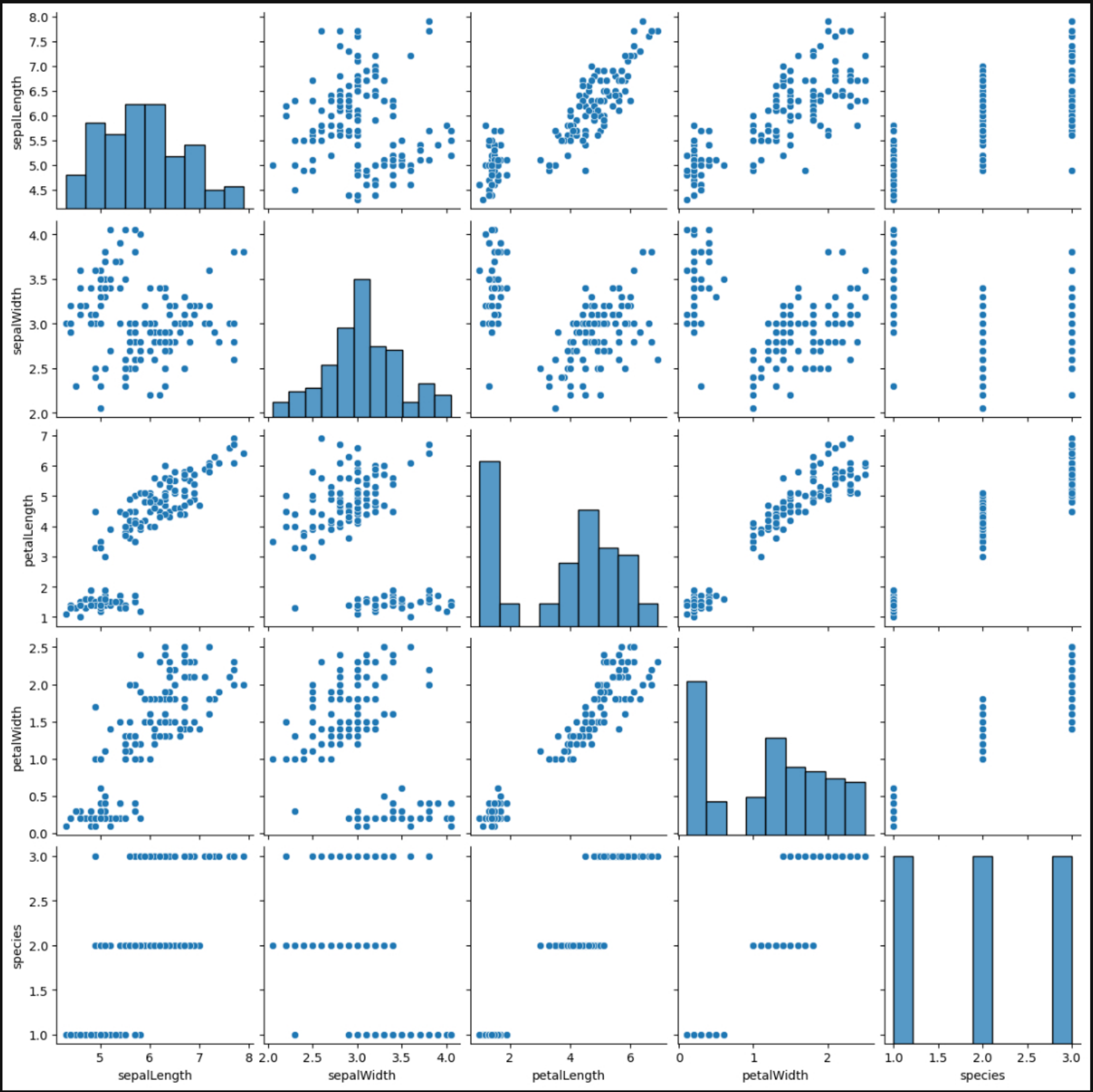
[44]:
```
sns.heatmap(corr)
```

[44]: `<Axes: >`



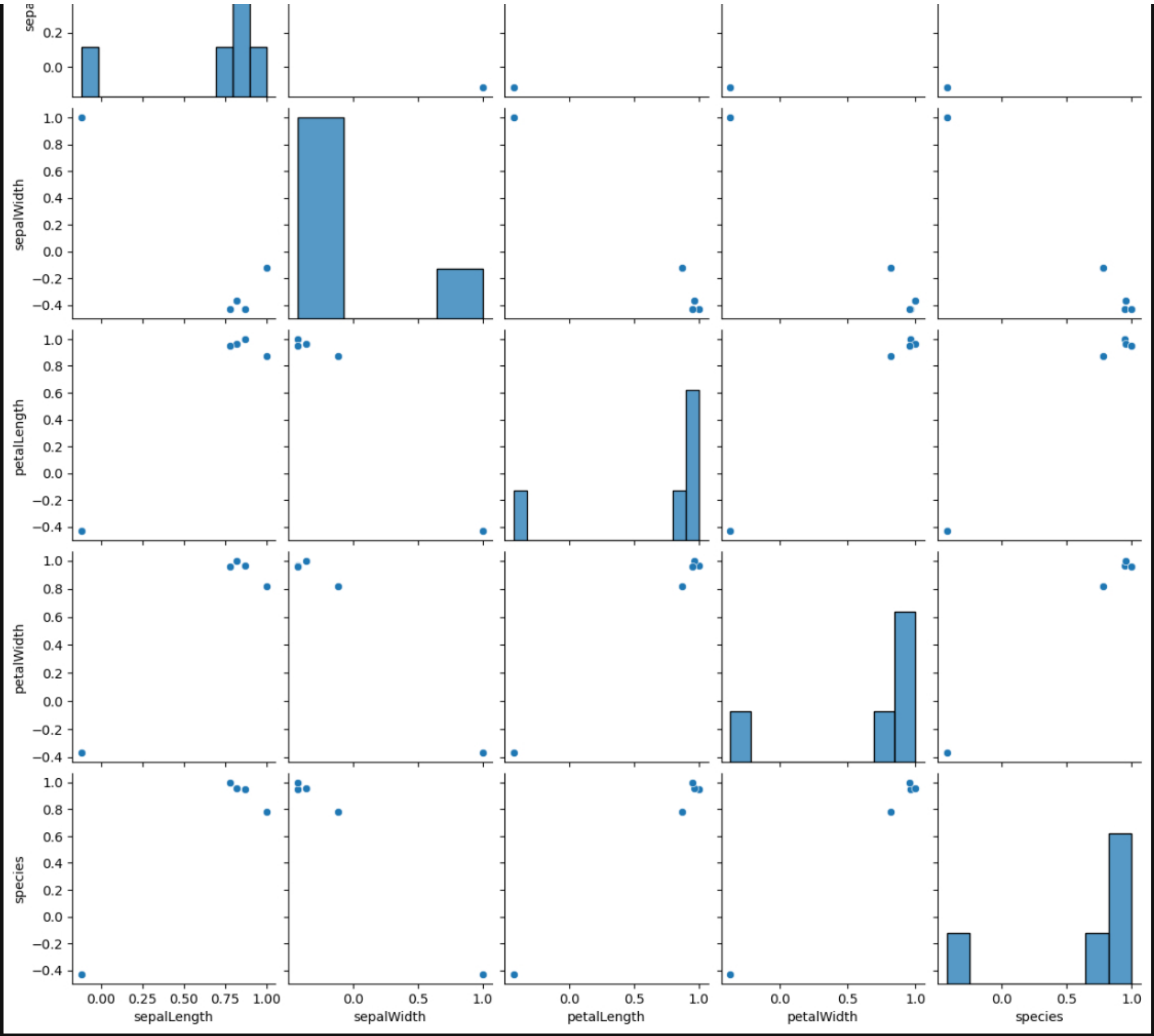
[45]:
```
sns.pairplot(data)
```

[45]:
```
<seaborn.axisgrid.PairGrid at 0x224a96fb620>
```



[46]:
```
sns.pairplot(corr)
```

[46]:
```
<seaborn.axisgrid.PairGrid at 0x224aa85cb60>
```



```
[47]: # dependent and independent variable
      x=data.iloc[:,:3]
      y=data["species"]

[48]: # training and testing data splitting by 80:20
      xtrain, xtest, ytrain, ytest=train_test_split(x,y,test_size=0.2)

[49]: xtrain.shape

[49]: (120, 3)

[50]: xtest.shape

[50]: (30, 3)

[51]: ytrain.shape

[51]: (120,)

[52]: ytest.shape

[52]: (30,)

[ ]:
```