```
#1.Download the dataset
#2.Load the dataset into the tool
from google.colab import files
uploaded=files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving abalone.csv to abalone.csv
```

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

```
import io

df = pd.read_csv(io.BytesIO(uploaded['abalone.csv']))
print(df)
```

```
     Sex  Length  Diameter  Height  Whole weight  Shucked weight  \
0      M   0.455     0.365   0.095        0.5140          0.2245   
1      M   0.350     0.265   0.090        0.2255          0.0995   
2      F   0.530     0.420   0.135        0.6770          0.2565   
3      M   0.440     0.365   0.125        0.5160          0.2155   
4      I   0.330     0.255   0.080        0.2050          0.0895   
...   ..     ...       ...     ...           ...             ...   
4172   F   0.565     0.450   0.165        0.8870          0.3700   
4173   M   0.590     0.440   0.135        0.9660          0.4390   
4174   M   0.600     0.475   0.205        1.1760          0.5255   
4175   F   0.625     0.485   0.150        1.0945          0.5310   
4176   M   0.710     0.555   0.195        1.9485          0.9455   

      Viscera weight  Shell weight  Rings  
0             0.1010        0.1500     15  
1             0.0485        0.0700      7  
2             0.1415        0.2100      9  
3             0.1140        0.1550     10  
4             0.0395        0.0550      7  
...              ...           ...    ...  
4172          0.2390        0.2490     11  
4173          0.2145        0.2605     10  
4174          0.2875        0.3080      9  
4175          0.2610        0.2960     10  
4176          0.3765        0.4950     12  

[4177 rows x 9 columns]
```

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

```
df
```

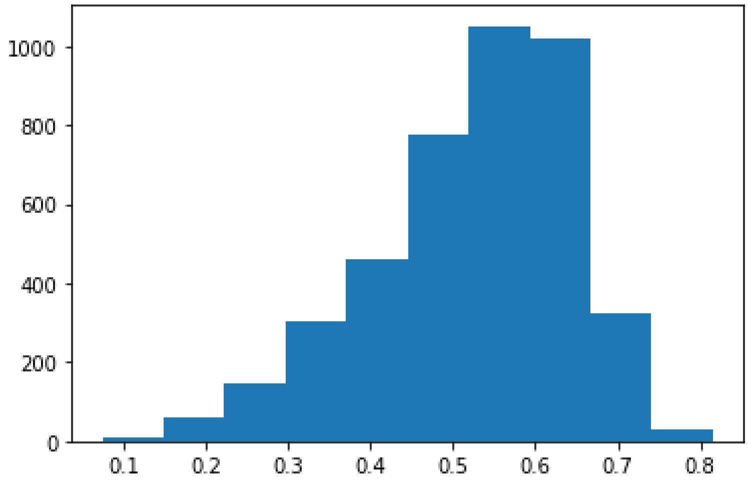
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	2	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	2	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	0	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	2	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

3.VISUALIZATIONS

Univariate Analysis

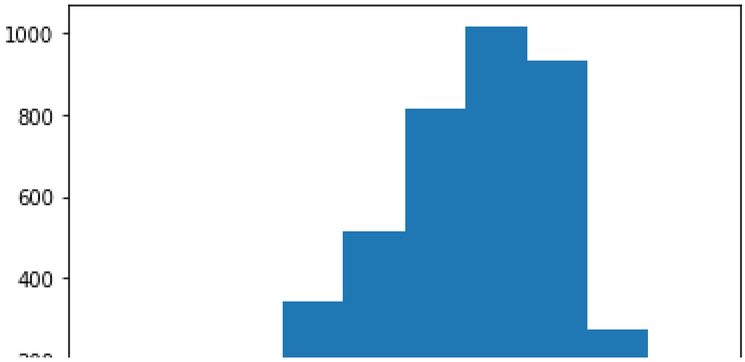
```
plt.hist(df['Length'])
```

```
(array([   7.,   60.,  147.,  304.,  460.,  778., 1051., 1017.,  324.,
          29.]),
 array([0.075, 0.149, 0.223, 0.297, 0.371, 0.445, 0.519, 0.593, 0.667,
        0.741, 0.815]),
 <a list of 10 Patch objects>)
```



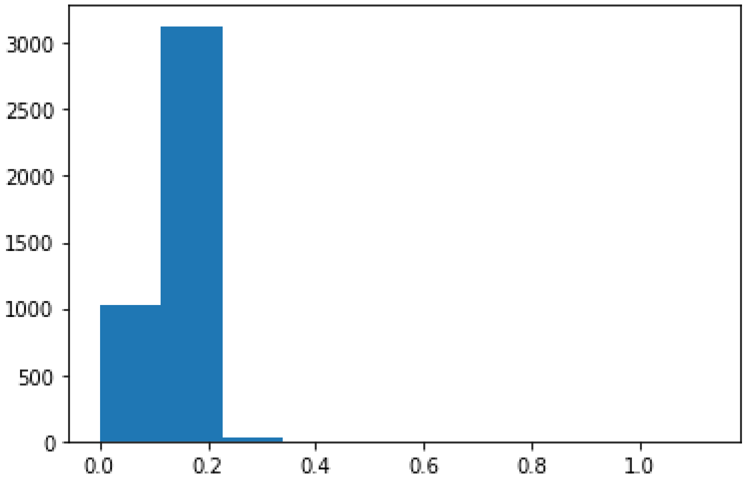
```
plt.hist(df['Diameter'])
```

```
(array([  13.,   66.,  180.,  344.,  513.,  812., 1017.,  934.,  275.,
          23.]),
 array([0.055 , 0.1145, 0.174 , 0.2335, 0.293 , 0.3525, 0.412 , 0.4715,
        0.531 , 0.5905, 0.65  ]),
 <a list of 10 Patch objects>)
```



```
plt.hist(df['Height'])
```

```
(array([1.023e+03, 3.129e+03, 2.300e+01, 0.000e+00, 1.000e+00, 0.000e+00,
        0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]),
 array([0.   , 0.113, 0.226, 0.339, 0.452, 0.565, 0.678, 0.791, 0.904,
        1.017, 1.13 ]),
 <a list of 10 Patch objects>)
```

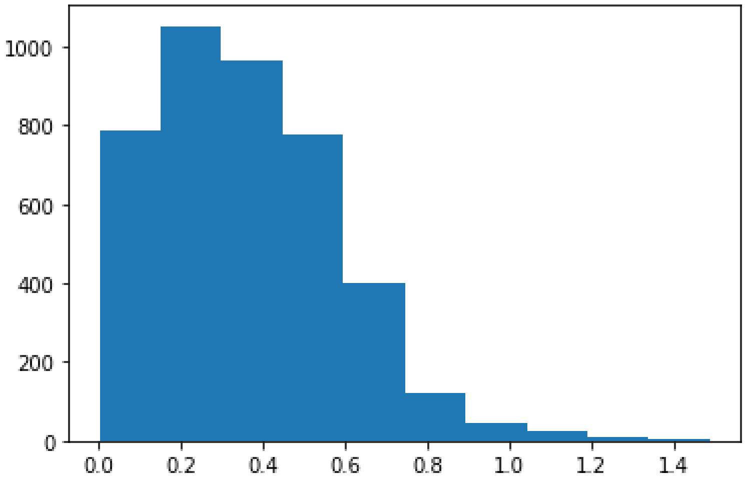


```
plt.hist(df['Whole weight'])
```

```
(array([632., 783., 827., 824., 616., 286., 129.,  58.,  16.,   6.]),
 array([2.00000e-03, 2.84350e-01, 5.66700e-01, 8.49050e-01, 1.13140e+00,
```

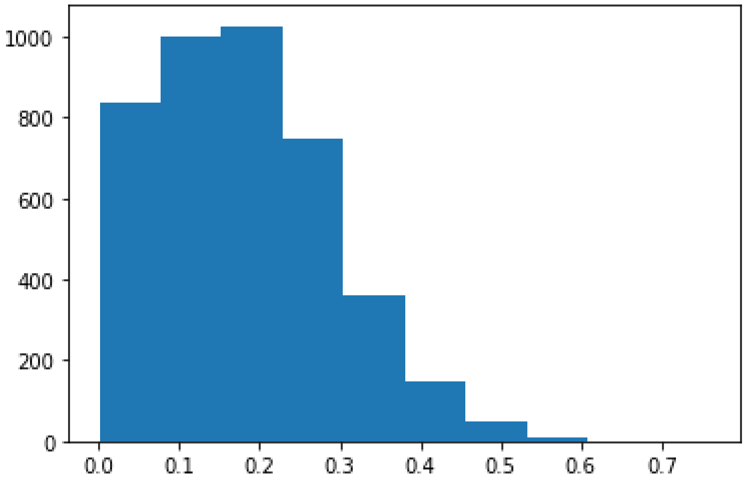
```
plt.hist(df['Shucked weight'])
```

```
(array([ 786., 1052.,  962.,  775.,  399.,  123.,   46.,   24.,    7.,
           3.]),
 array([1.0000e-03, 1.4970e-01, 2.9840e-01, 4.4710e-01, 5.9580e-01,
        7.4450e-01, 8.9320e-01, 1.0419e+00, 1.1906e+00, 1.3393e+00,
        1.4880e+00]),
 <a list of 10 Patch objects>)
```



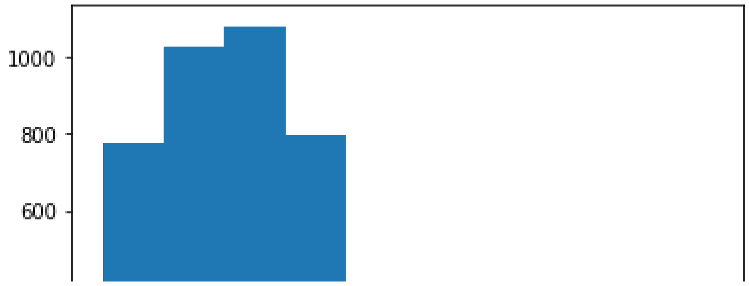
```
plt.hist(df['Viscera weight'])
```

```
(array([8.350e+02, 9.990e+02, 1.027e+03, 7.470e+02, 3.630e+02, 1.470e+02,
        5.000e+01, 7.000e+00, 1.000e+00, 1.000e+00]),
 array([5.0000e-04, 7.6450e-02, 1.5240e-01, 2.2835e-01, 3.0430e-01,
        3.8025e-01, 4.5620e-01, 5.3215e-01, 6.0810e-01, 6.8405e-01,
        7.6000e-01]),
 <a list of 10 Patch objects>)
```



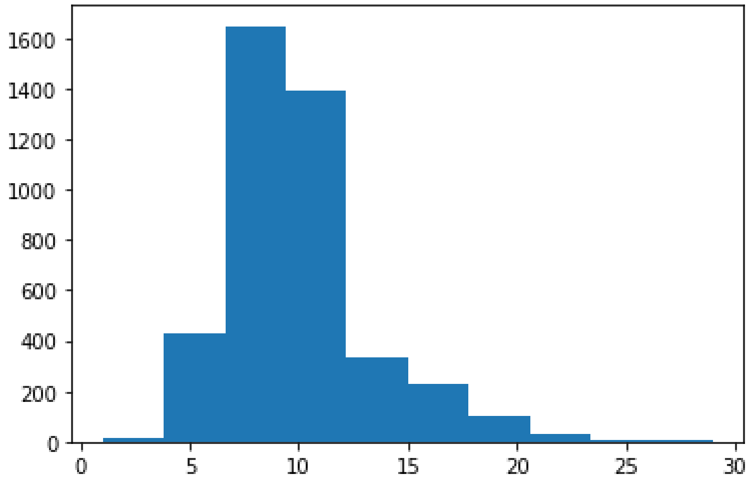
```
plt.hist(df['Shell weight'])
```

```
(array([7.770e+02, 1.023e+03, 1.078e+03, 7.980e+02, 3.490e+02, 1.040e+02,
        3.300e+01, 9.000e+00, 5.000e+00, 1.000e+00]),
 array([0.0015 , 0.10185, 0.2022 , 0.30255, 0.4029 , 0.50325, 0.6036 ,
        0.70395, 0.8043 , 0.90465, 1.005  ]),
 <a list of 10 Patch objects>)
```



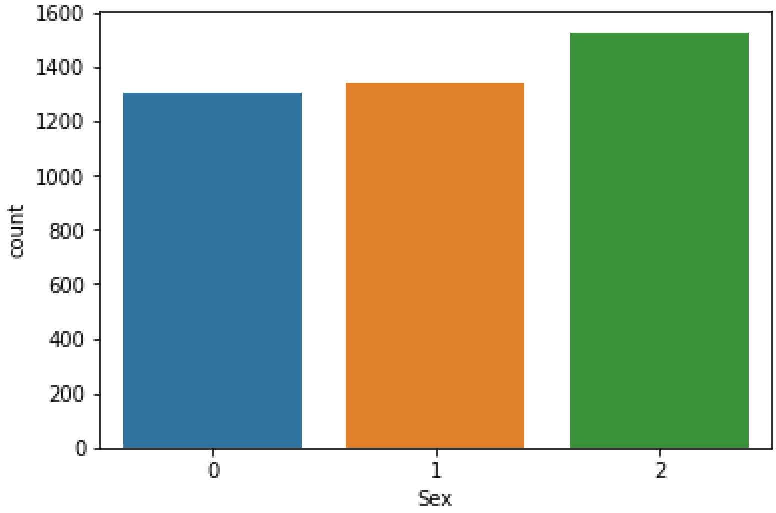
```
plt.hist(df['Rings'])
```

```
(array([  17.,  431., 1648., 1388.,  329.,  228.,  100.,   29.,    4.,
           3.]),
 array([ 1. ,  3.8,  6.6,  9.4, 12.2, 15. , 17.8, 20.6, 23.4, 26.2, 29. ]),
 <a list of 10 Patch objects>)
```



```
sns.countplot(df['Sex'])
```

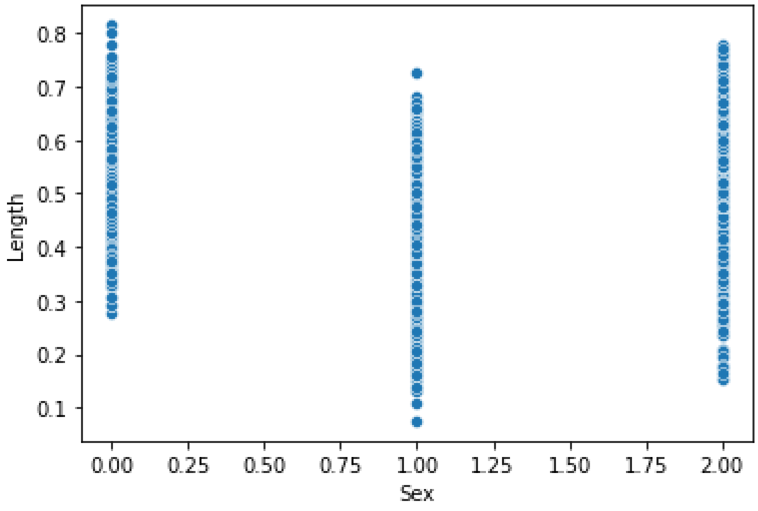
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7febab8ff290>
```



Bivariate Analysis

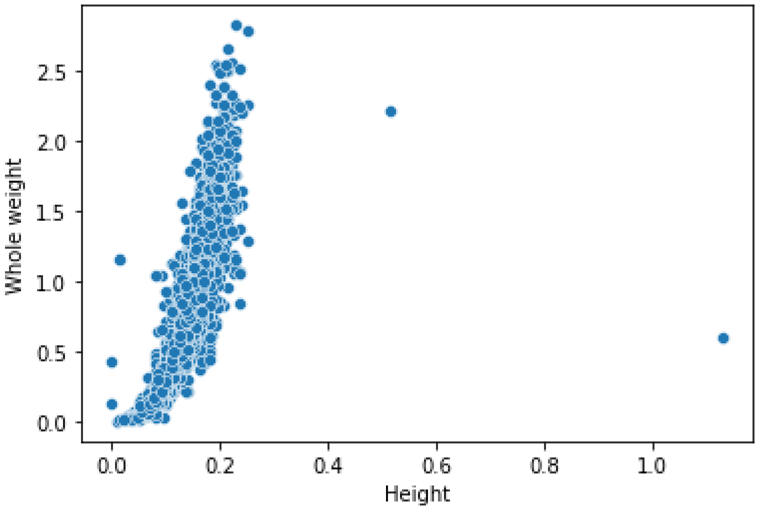
```
sns.scatterplot(df['Sex'], df['Length'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7febab7e9e90>
```



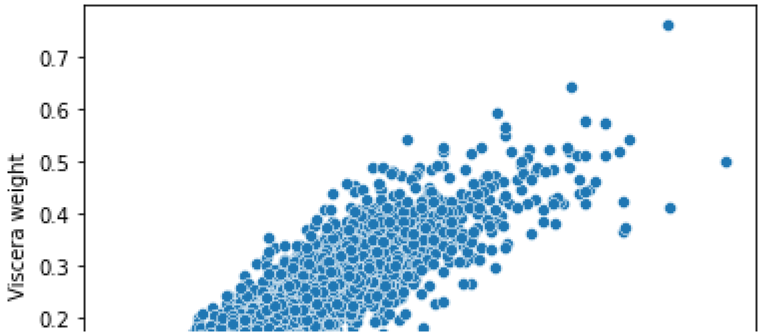
```
sns.scatterplot(df['Height'], df['Whole weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7febab778390>
```



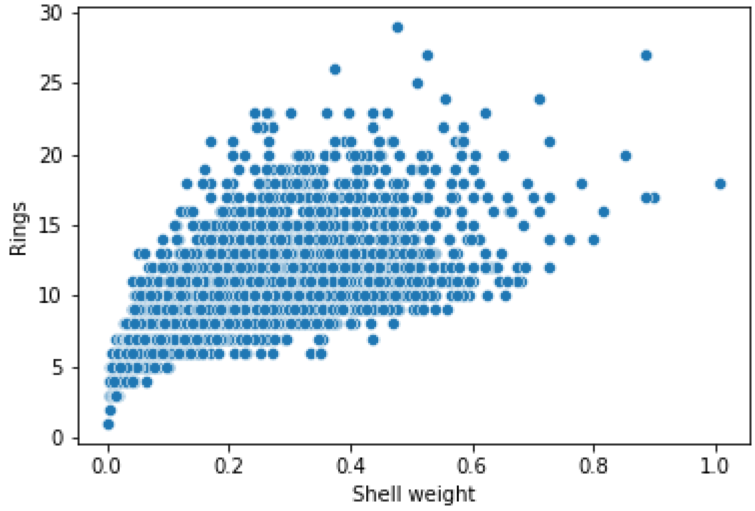
```
sns.scatterplot(df['Shucked weight'], df['Viscera weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7febab752510>
```



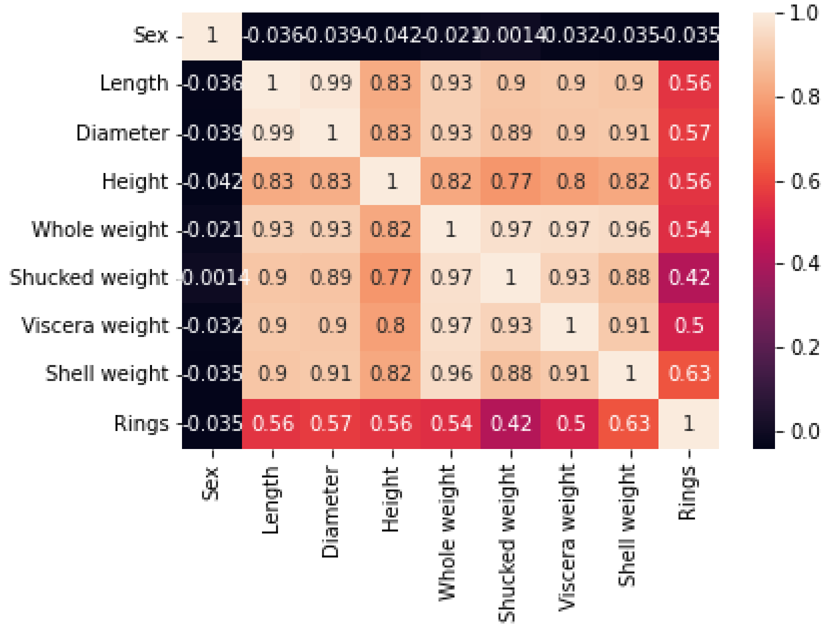
```
sns.scatterplot(df['Shell weight'], df['Rings'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7febab65a490>
```



```
sns.heatmap(df.corr(), annot = True)
```

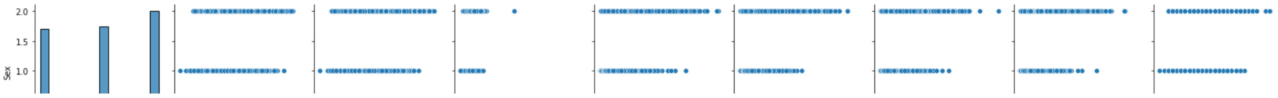
```
<matplotlib.axes._subplots.AxesSubplot at 0x7febab61d310>
```



Multi Variate Analysis

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7feba8c52790>
```



4.Descriptive Statistics

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4177 non-null   int64  
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64  
dtypes: float64(7), int64(2)
memory usage: 293.8 KB
```

```
df.describe()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	41
mean	1.052909	0.523992	0.407881	0.139516	0.828742	0.359367	
std	0.822240	0.120093	0.099240	0.041827	0.490389	0.221963	
min	0.000000	0.075000	0.055000	0.000000	0.002000	0.001000	
25%	0.000000	0.450000	0.350000	0.115000	0.441500	0.186000	
50%	1.000000	0.545000	0.425000	0.140000	0.799500	0.336000	
75%	2.000000	0.615000	0.480000	0.165000	1.153000	0.502000	

```
df.skew()
```

```
Sex              -0.098155
Length           -0.639873
Diameter         -0.609198
Height            3.128817
Whole weight      0.530959
Shucked weight    0.719098
Viscera weight    0.591852
Shell weight      0.620927
Rings             1.114102
dtype: float64
```

```
df.kurt()
```

```
Sex               -1.514387
Length             0.064621
Diameter          -0.045476
Height            76.025509
Whole weight      -0.023644
Shucked weight     0.595124
Viscera weight     0.084012
Shell weight       0.531926
Rings              2.330687
dtype: float64
```

```
df.corr()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	w
Sex	1.000000	-0.036066	-0.038874	-0.042077	-0.021391	-0.001373	-0.032067	-0.0
Length	-0.036066	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.8
Diameter	-0.038874	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.9
Height	-0.042077	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.8
Whole weight	-0.021391	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.9
Shucked weight	-0.001373	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.8
Viscera weight	-0.032067	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.9

```
df.var()
```

```
Sex                0.676079
Length             0.014422
Diameter           0.009849
Height             0.001750
Whole weight       0.240481
Shucked weight     0.049268
Viscera weight     0.012015
Shell weight       0.019377
Rings             10.395266
dtype: float64
```

```
df.std()
```

```
Sex               0.822240
Length            0.120093
Diameter          0.099240
Height            0.041827
Whole weight      0.490389
Shucked weight    0.221963
```

```
Viscera weight    0.109614
Shell weight      0.139203
Rings             3.224169
dtype: float64
```

5.Checking for missing values

```
df.isna().sum()
```

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

```
df.isna().sum().sum()
```

```
0
```

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

```
0
```

6.Finding & Handling Ouliers

```
quantile = df.quantile(q = [0.25, 0.75])
quantile
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0.25	0.0	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	8.0
0.75	2.0	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329	11.0

```
IQR = quantile.iloc[1] - quantile.iloc[0]
IQR
```

```
Sex               2.0000
Length            0.1650
Diameter          0.1300
Height            0.0500
Whole weight      0.7115
Shucked weight    0.3160
Viscera weight    0.1595
Shell weight      0.1990
Rings             3.0000
dtype: float64
```

```
upper = quantile.iloc[1] + (1.5 *IQR)
upper
```

```
Sex                5.00000
Length             0.86250
Diameter           0.67500
Height             0.24000
Whole weight       2.22025
Shucked weight     0.97600
Viscera weight     0.49225
Shell weight       0.62750
Rings             15.50000
dtype: float64
```

```
lower = quantile.iloc[0] - (1.5* IQR)
lower
```

```
Sex              -3.00000
Length            0.20250
Diameter          0.15500
Height            0.04000
Whole weight     -0.62575
Shucked weight   -0.28800
Viscera weight   -0.14575
Shell weight     -0.16850
Rings             3.50000
dtype: float64
```

```
df.mean()
```

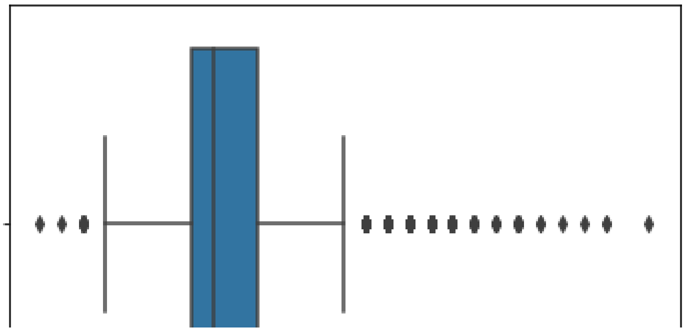
```
Sex               1.052909
Length            0.523992
Diameter          0.407881
Height            0.139516
Whole weight      0.828742
Shucked weight    0.359367
Viscera weight    0.180594
Shell weight      0.238831
Rings             9.933684
dtype: float64
```

```
df['Length'].max()
```

```
0.815
```

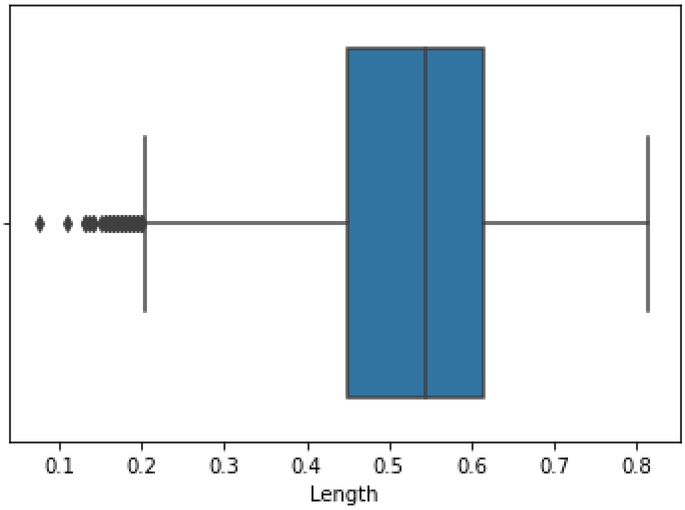
```
sns.boxplot(df['Rings'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feba6ad2ad0>
```



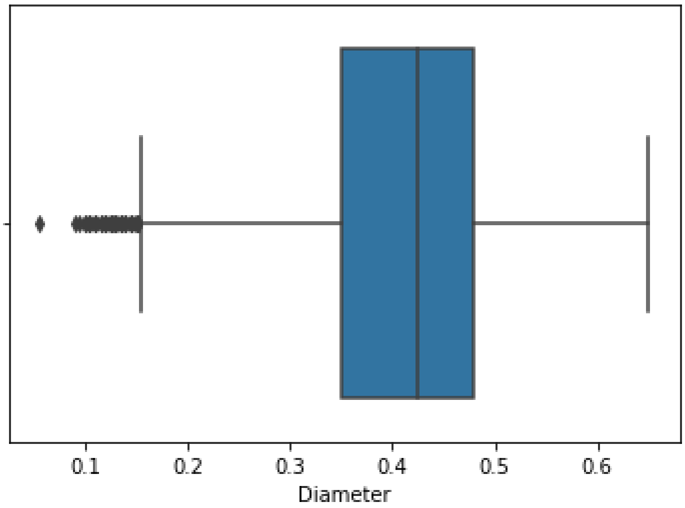
```
sns.boxplot(df['Length'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feba69a4810>
```



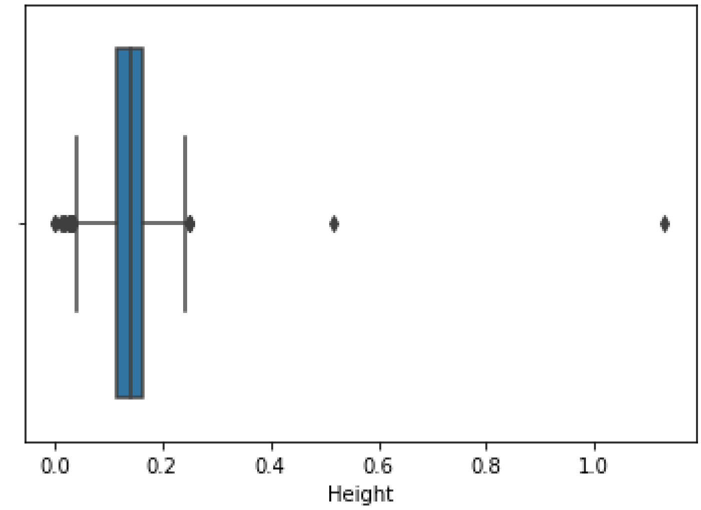
```
sns.boxplot(df['Diameter'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feba4e3f5d0>
```



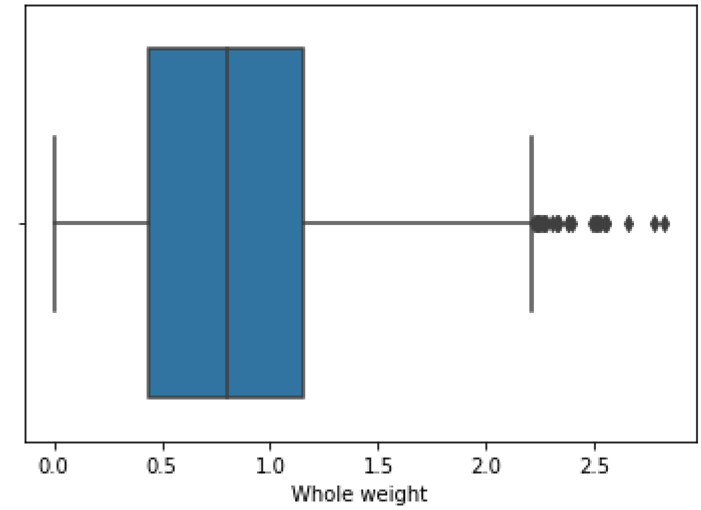
```
sns.boxplot(df['Height'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feba4e076d0>
```



```
sns.boxplot(df['Whole weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pas
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feba4d77f10>
```

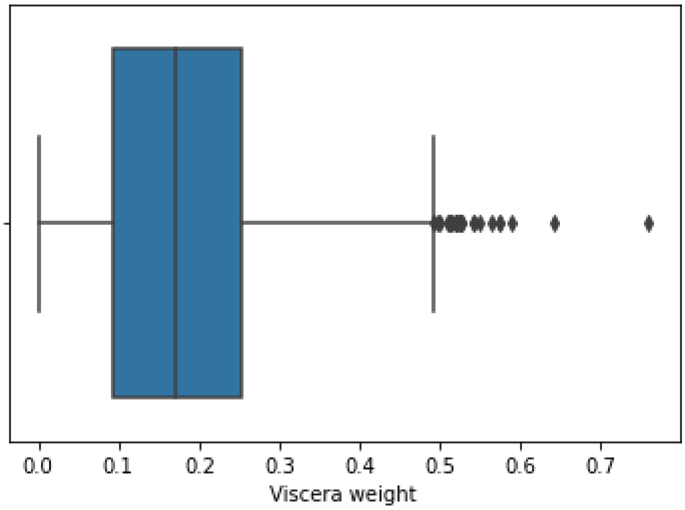


```
sns.boxplot(df['Shucked weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feba4cd1750>
```

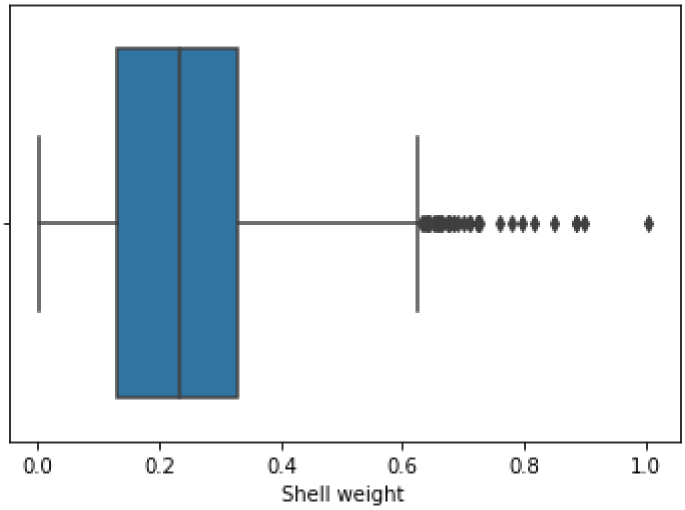
```
sns.boxplot(df['Viscera weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feba4cd1810>
```



```
sns.boxplot(df['Shell weight'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7feba4c41d50>
```



7.Check for categorical column and perform encoding

```
from sklearn.preprocessing import LabelEncoder
```

```
le=LabelEncoder()
print(df.Sex.value_counts())
df.Sex=le.fit_transform(df.Sex)
print(df.Sex.value_counts())
```

```
2    1528
1    1342
0    1307
Name: Sex, dtype: int64
2    1528
1    1342
0    1307
Name: Sex, dtype: int64
```

8.Split the data into dependent and independent variables

```
x=df.iloc[:,[0,7]].values
x
```

```
array([[2.   , 0.15 ],
       [2.   , 0.07 ],
       [0.   , 0.21 ],
       ...,
       [2.   , 0.308],
       [0.   , 0.296],
       [2.   , 0.495]])
```

```
y=df.iloc[:,8].values
y
#dependent
```

```
array([15,  7,  9, ...,  9, 10, 12])
```

9.Scale the independent variables

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x=ss.fit_transform(x)
x
```

```
array([[ 1.15198011, -0.63821689],
       [ 1.15198011, -1.21298732],
       [-1.28068972, -0.20713907],
       ...,
       [ 1.15198011,  0.49695471],
       [-1.28068972,  0.41073914],
       [ 1.15198011,  1.84048058]])
```

10.Split the data intp training and testing

```
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=0)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
     (3341, 2)
     (836, 2)
     (3341,)
     (836,)
```

11.Build the model 12.Train the model

```
from sklearn.linear_model import LinearRegression
mlr=LinearRegression()
mlr.fit(x_train,y_train)
```

```
     LinearRegression()
```

13.Test the model

```
mlr.predict(x_test[0:5])
```

```
     array([11.2719321 ,  9.27390152, 11.02122356,  6.78830546, 11.88079569])
```

14.Measure the performance using metrics

```
from sklearn.metrics import r2_score
r2_score(mlr.predict(x_test),y_test)
```

```
     -0.696307580804389
```

✓ 0s completed at 6:18 PM ● ✕