8.65452914, 10.19571868,
13.47423856, 9.0
6825076, 6.69843582, 1
3.38213142, 9.62823486,
8.20174551, 7.7
9183041, 9.3338472 , 1
1.08195328, 11.25321895,
6.11231204, 10.6
960639 , 9.23348159,
7.76425036, 11.65342323,
12.6024271 , 7.4
9694081, 9.71678931,
7.41119139, 6.94925679,
6.34706174, 9.9
9734923, 6.70117631, 1
0.71374432, 9.59457302,
7.07847213, 6.6
940933 , 9.30356123, 1
3.66698224, 9.71369221,
17.36952958, 7.8
1225327, 8.86909973,
9.29540502, 11.03405521,
12.90720962, 13.0
3952065, 4.90843127,
9.50619996, 10.09434256,
8.67296752, 9.0
3746047, 8.33310609, 1
0.60445018, 9.66636969,
7.67351279, 8.7
4447193, 12.37470593,
7.70552082, 11.35599144,
11.25726129, 10.0
2276461, 8.01953433, 1
1.39538114, 7.92288557,
11.02588274, 7.0
2530311, 10.80014326, 1
3.22266766, 11.41469264,
7.5577235 , 6.8
3654146, 6.97820486, 1
0.29150052, 9.1851768 ,
9.72122817, 9.2
9569276, 11.98122676,
9.87982582, 8.55374278,

In [ ]:
```
from sklearn.preprocessi
x = scale(x)
x
```

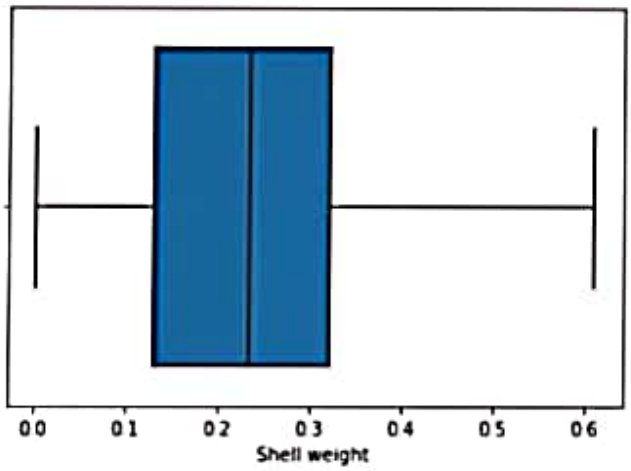
Out[ ]:
```
array([[-0.0105225 , -0.
67088921, -0.50179694,
..., -0.61037964,
        -0.7328165 , -0.
64358742],
       [-0.0105225 , -1.
61376082, -1.57304487,
..., -1.22513334,
        -1.24343929, -1.
25742181],
       [-1.26630752,  0.
00259051,  0.08738942,
..., -0.45300269,
        -0.33890749, -0.
18321163],
       ...,
       [-0.0105225 ,  0.
63117159,  0.67657577,
...,  0.86994729,
         1.08111018,  0.
56873549],
       [-1.26630752,  0.
85566483,  0.78370057,
...,  0.89699645,
         0.82336724,  0.
47666033],
       [-0.0105225 ,  1.
61894185,  1.53357412,
...,  0.00683308,
         1.94673739,  2.
00357336]])
```

**9.Split the data into training and testing**

In [ ]:
```
from sklearn.model_selec
```

In [ ]:
```
data['Shell weight']=np.
sns.boxplot(data['Shell
```

Out[ ]:



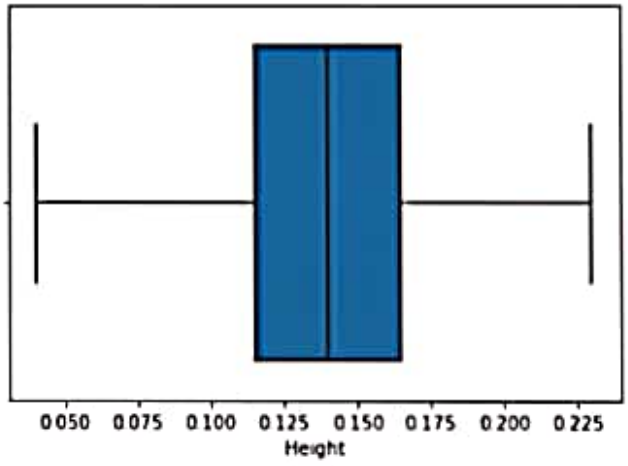

**6.Check for Categorical columns and perform encoding.**

In [ ]:
```
data['Sex'].replace({'M'
data
```

Out[ ]:

| | Sex | Length | Diameter | H |
|---|---|---|---|---|
| **0** | 1 | 0.455 | 0.365 | |
| **1** | 1 | 0.350 | 0.265 | |
| **2** | 0 | 0.530 | 0.420 | |
| **3** | 1 | 0.440 | 0.365 | |
| **4** | 2 | 0.330 | 0.255 | |
| **...** | ... | ... | ... | |
| **4172** | 0 | 0.565 | 0.450 | |
| **4173** | 1 | 0.590 | 0.440 | |
| **4174** | 1 | 0.600 | 0.475 | |

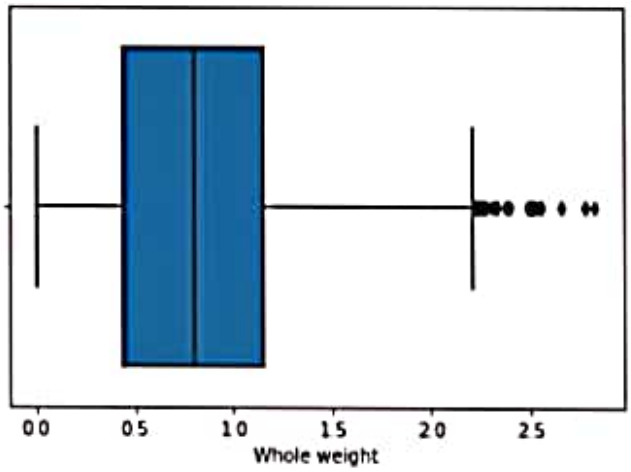
Out[ ]:




In [ ]:

```
sns.boxplot(data['Whole
```

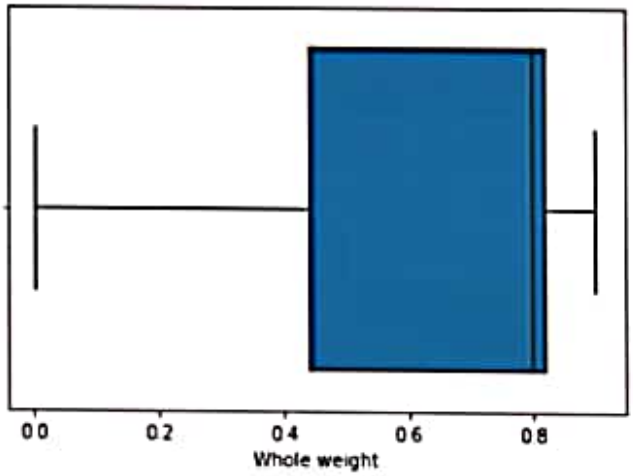
Out[ ]:



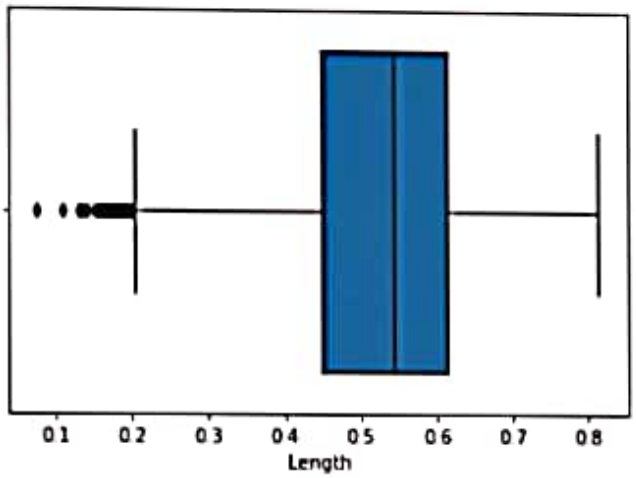

In [ ]:

```
data['Whole weight']=np.
sns.boxplot(data['Whole
```

Out[ ]:



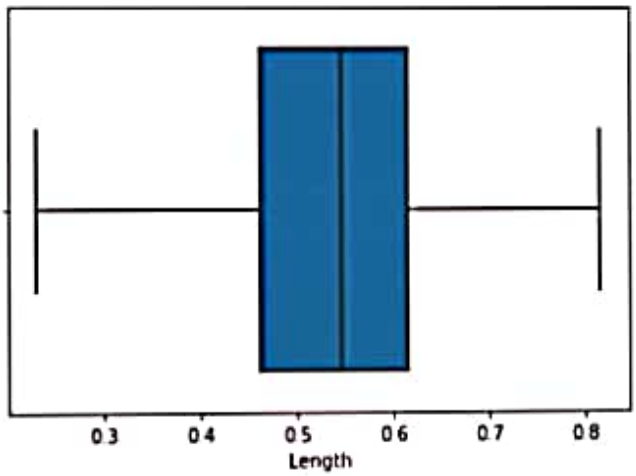
Out[ ]:




In [ ]:

```
data['Length']=np.where(
sns.boxplot(data['Length
```

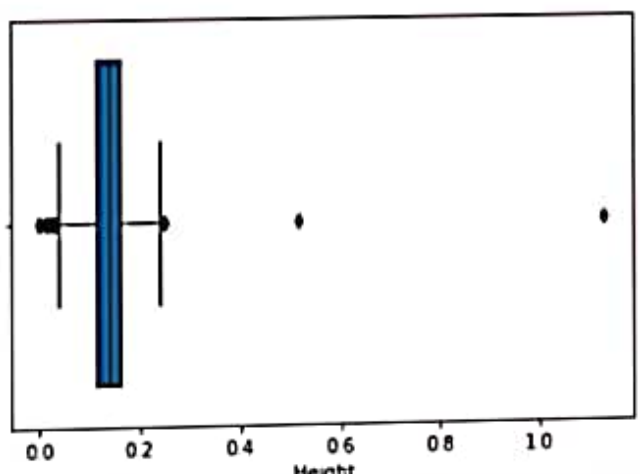
Out[ ]:




In [ ]:

```
sns.boxplot(data['Height
```

Out[ ]:



Out[ ]:

| | Length | Diameter | Height |
|---|---|---|---|
| **0.25** | 0.450 | 0.35 | 0.115 |
| **0.75** | 0.615 | 0.48 | 0.165 |

In [ ]:
```
iqr=quant.loc[0.75]-quar
iqr
```

Out[ ]:
```
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
```

In [ ]:
```
low=quant.loc[0.25]-(1.5
low
```

Out[ ]:
```
Length              0.2025
0
Diameter            0.1550
0
Height              0.0400
0
Whole weight       -0.6257
5
Shucked weight     -0.2880
0
Viscera weight     -0.1457
5
Shell weight       -0.1685
0
Rings               3.5000
0
dtype: float64
```

In [ ]:
```
data.isna().sum()
```

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

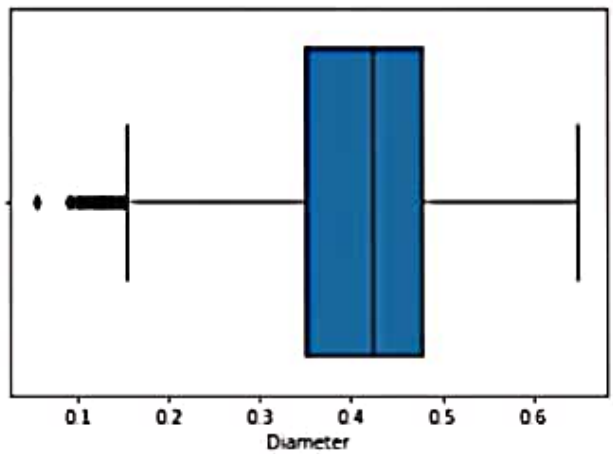
In [ ]:
```
data.isna().any().sum()
```

Out[ ]: 0

**5.Find the outliers and replace them outliers**

In [ ]:
```
sns.boxplot(data['Diamet
```

Out[ ]:




In [ ]:
```
quant=data.quantile(q=[(
quant
```

In [ ]:
```
data.isna()
```

Out[ ]:

| | Sex | Length | Diameter |
|---|---|---|---|
| 0 | False | False | False |
| 1 | False | False | False |
| 2 | False | False | False |
| 3 | False | False | False |
| 4 | False | False | False |
| ... | ... | ... | ... |
| 4172 | False | False | False |
| 4173 | False | False | False |
| 4174 | False | False | False |
| 4175 | False | False | False |
| 4176 | False | False | False |

4177 rows × 9 columns

In [ ]:
```
data.isna().any()
```

Out[ ]:
```
Sex               False
Length            False
Diameter          False
Height            False
Whole weight      False
Shucked weight    False
Viscera weight    False
Shell weight      False
Rings             False
dtype: bool
```

In [ ]:
```
data.var()
```

Out[ ]:
```
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
```

In [ ]:
```
data.nunique()
```

Out[ ]:
```
Sex                3
Length           134
Diameter         111
Height            51
Whole weight    2429
Shucked weight  1515
Viscera weight   880
Shell weight     926
Rings             28
dtype: int64
```

**4.Check for missing values and deal with them**

In [ ]:
```
data.isna()
```

In [ ]:
```
data.describe()
```

Out[ ]:

| | Length | Diamete |
|---|---|---|
| count | 4177.000000 | 4177.00000 |
| mean | 0.523992 | 0.40788 |
| std | 0.120093 | 0.09924 |
| min | 0.075000 | 0.05500 |
| 25% | 0.450000 | 0.35000 |
| 50% | 0.545000 | 0.42500 |
| 75% | 0.615000 | 0.48000 |
| max | 0.815000 | 0.65000 |

In [ ]:
```
data.mode().T
```

Out[ ]:

| | 0 | 1 |
|---|---|---|
| Sex | M | NaN |
| Length | 0.55 | 0.625 |
| Diameter | 0.45 | NaN |
| Height | 0.15 | NaN |
| Whole weight | 0.2225 | NaN |
| Shucked weight | 0.175 | NaN |
| Viscera weight | 0.1715 | NaN |
| Shell weight | 0.275 | NaN |
| Rings | 9.0 | NaN |

In [ ]:

```
data.head()
```

Out[ ]:

| | Sex | Length | Diameter | Heigl |
|---|---|---|---|---|
| **0** | M | 0.455 | 0.365 | 0.09 |
| **1** | M | 0.350 | 0.265 | 0.09 |
| **2** | F | 0.530 | 0.420 | 0.13 |
| **3** | M | 0.440 | 0.365 | 0.12 |
| **4** | I | 0.330 | 0.255 | 0.08 |

In [ ]:

```
data.tail()
```

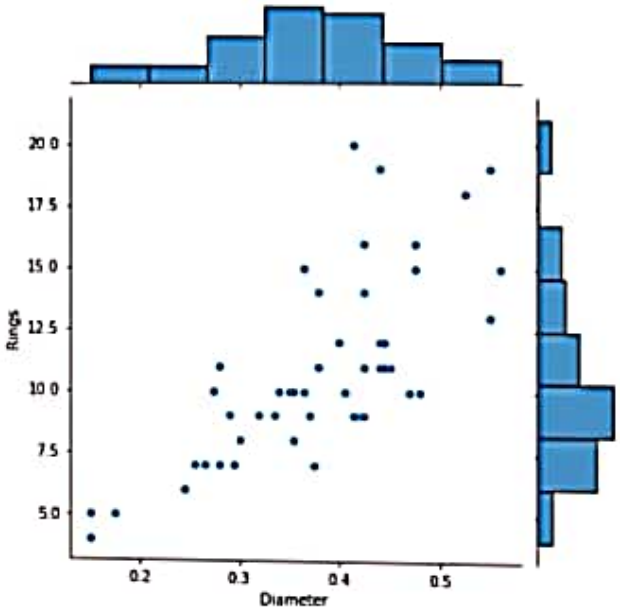
Out[ ]:

| | Sex | Length | Diameter | H |
|---|---|---|---|---|
| **4172** | F | 0.565 | 0.450 | |
| **4173** | M | 0.590 | 0.440 | |
| **4174** | M | 0.600 | 0.475 | |
| **4175** | F | 0.625 | 0.485 | |
| **4176** | M | 0.710 | 0.555 | |

In [ ]:

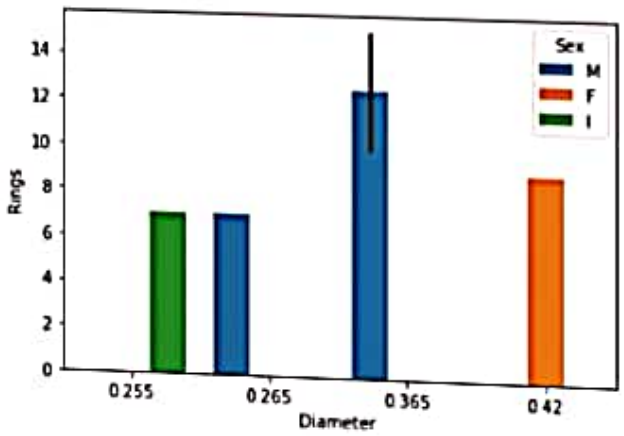
```
data.info()
```

```
RangeIndex: 4177 entrie
s, 0 to 4176
Data columns (total 9 co
lumns):
 #   Column          Non
-Null Count  Dtype
---  ------          ---
----------  -----
```



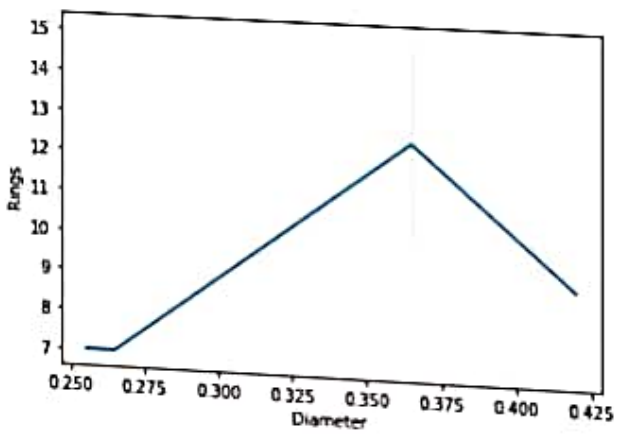

In [ ]:
```
sns.barplot('Diameter','
```

Out[ ]:



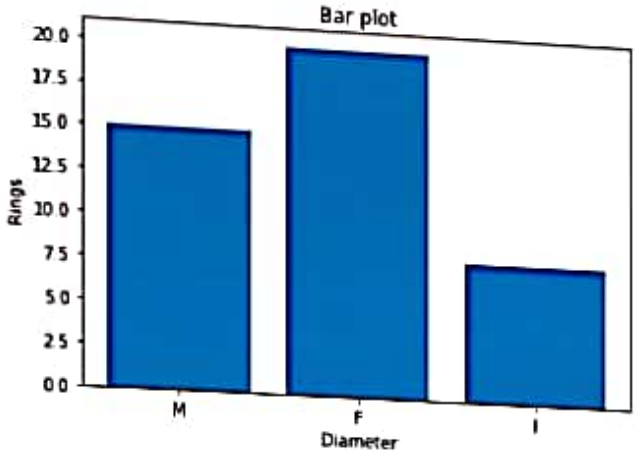

In [ ]:
```
sns.lineplot(data['Diame
```

Out[ ]:



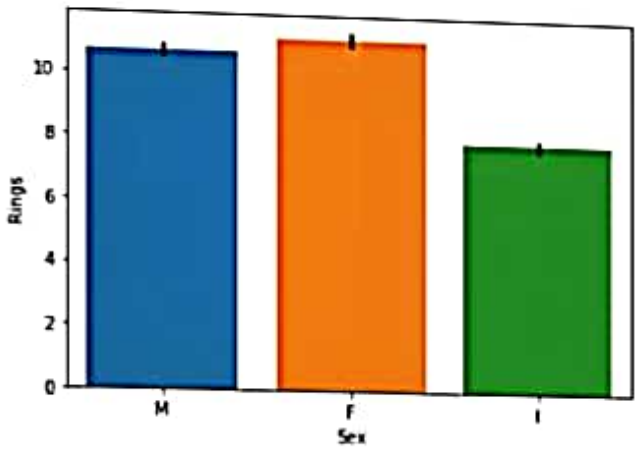
In [ ]:
```
plt.bar(data['Sex'].head
plt.title('Bar plot')
plt.xlabel('Diameter')
plt.ylabel('Rings')
```

Out[ ]: Text(0, 0.5, 'Rings')



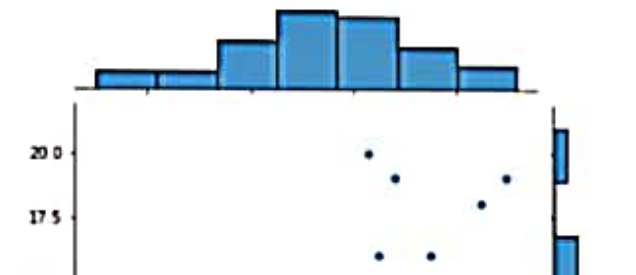

In [ ]:
```
sns.barplot(data['Sex'],
```

Out[ ]:



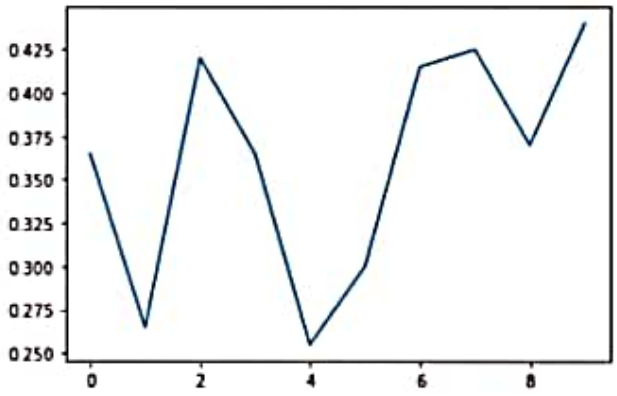

In [ ]:
```
sns.jointplot(data['Diam
```

Out[ ]:



In [ ]:
```
plt.plot(data['Diameter'
```

Out[ ]: []




In [ ]:
```
plt.pie(data['Diameter']
```

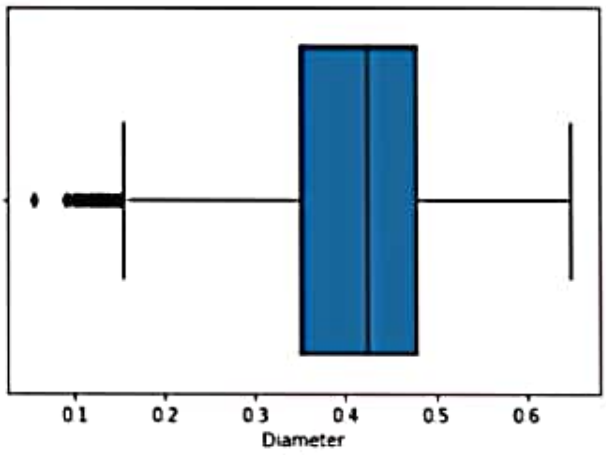
Out[ ]:
```
([,
  ,
  ,
  ,
  ],
 [Text(0.850721562611055
7, 0.6973326486753676,
''),
  Text(-0.32611344931648
134, 1.0505474849691026,
''),
  Text(-1.09980536640789
08, -0.020691931287471 4, ''),
  Text(-0.08269436219656
089, -1.096887251480709,
''),
  Text(0.975844636228721
8, -0.507668440956924 1,
'')],
 [Text(0.464029943242393
94, 0.380363262913836 9,
'21.856'),
  Text(-0.17788006326353
```

Out[ ]:

| | Sex | Length | Diameter | Heigl |
|---|---|---|---|---|
| **0** | M | 0.455 | 0.365 | 0.09 |
| **1** | M | 0.350 | 0.265 | 0.09 |
| **2** | F | 0.530 | 0.420 | 0.13 |
| **3** | M | 0.440 | 0.365 | 0.12 |
| **4** | I | 0.330 | 0.255 | 0.08 |

In [ ]:
```
sns.boxplot(data['Diamet
```

Out[ ]:




In [ ]:
```
plt.hist(data['Diameter'
```

Out[ ]:
```
(array([  13.,   66.,  1
80.,  344.,  513.,  81
2., 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  ]),
 )
```

## 1.Loading Dataset into tool

In [ ]:
```
from google.colab import
uploaded = files.upload(
```

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 ab
alone.csv
```

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot
import seaborn as sns
import warnings
warnings.filterwarnings(
```

In [ ]:
```
data = pd.read_csv("abal
```

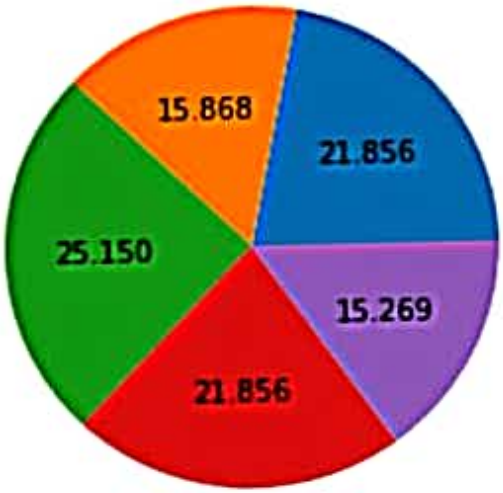
## 2.Performing Visualization

### Univariate Analysis

In [ ]:
```
data.head()
```

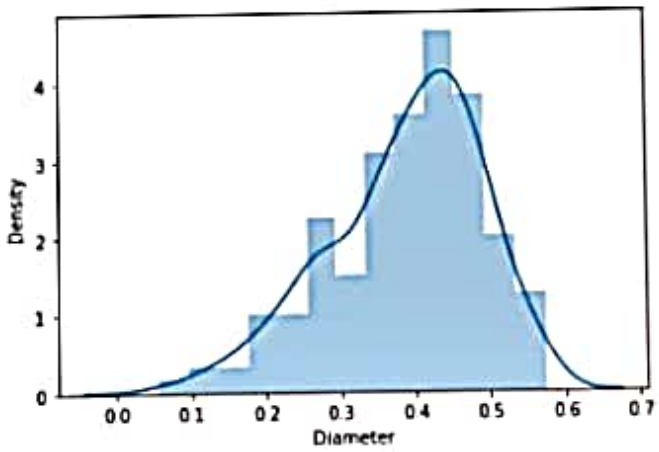
Out[ ]:

| | Sex | Length | Diameter | Heigl |
|---|---|---|---|---|
| 0 | M | 0.455 | 0.365 | 0.09 |
| 1 | M | 0.350 | 0.265 | 0.09 |



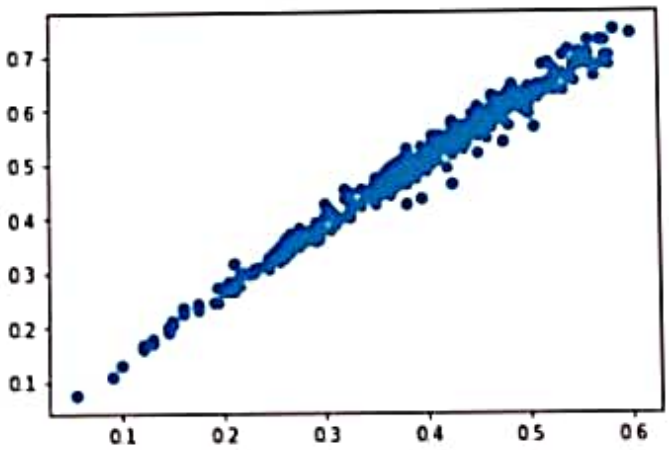

In [ ]:
```
sns.distplot(data['Diame
```

Out[ ]:



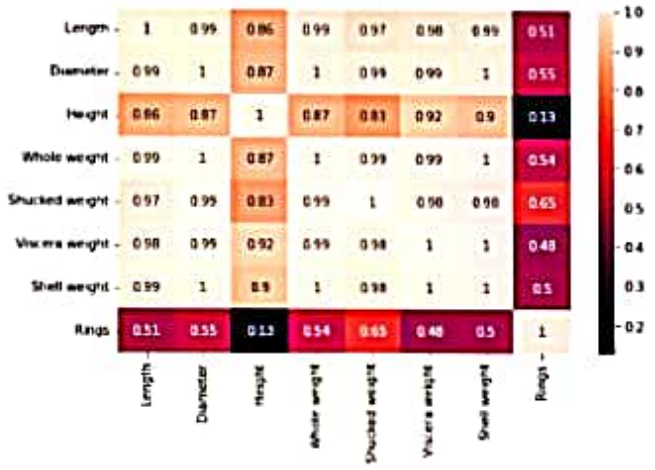

In [ ]:
```
plt.scatter(data['Diamet
```

Out[ ]:



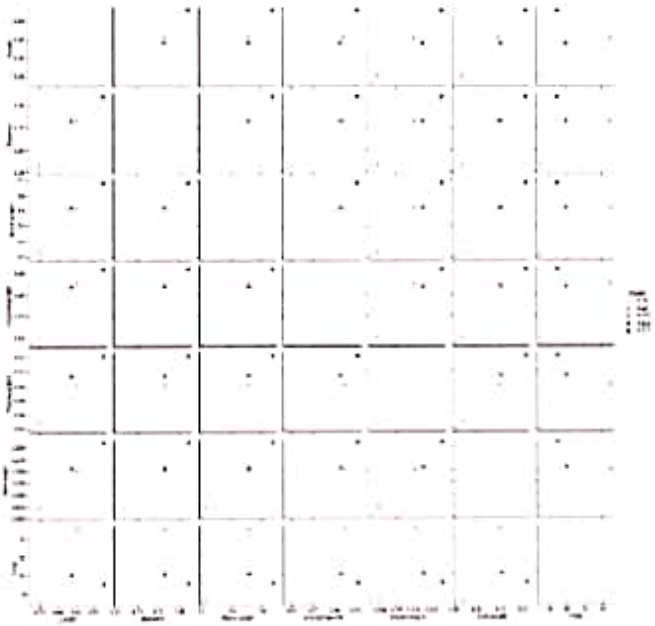
In [ ]:
```
fig=plt.figure(figsize=(
sns.heatmap(data.head().
```

Out[ ]:




In [ ]:
```
sns.pairplot(data.head()
```

Out[ ]:



In [ ]:
```
sns.pairplot(data.head()
```

Out[ ]:

In [ ]:
```
data.shape
```

Out[ ]: (4177, 9)

In [ ]:
```
data.kurt()
```

Out[ ]:
```
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
```

In [ ]:
```
data.skew()
```

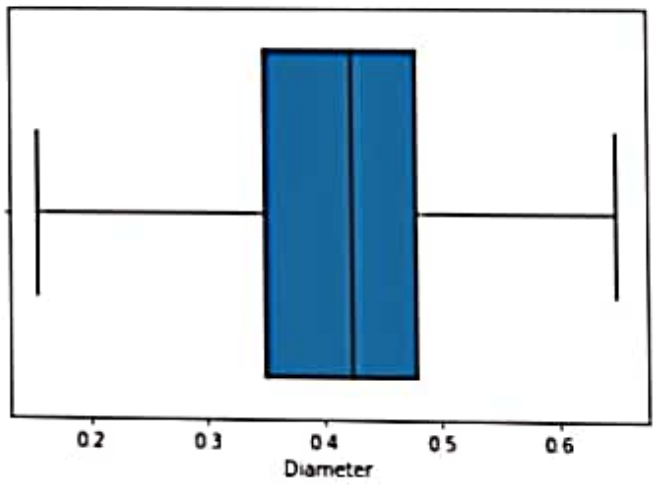
Out[ ]:
```
Length           -0.639873
Diameter         -0.609198
Height            3.128817
Whole weight      0.530959
Shucked weight    0.719098
Viscera weight    0.591852
```

In [ ]:
```
up=quant.loc[0.75]+(1.5*
up
```

Out[ ]:
```
Length            0.862
50
Diameter          0.675
00
Height            0.240
00
Whole weight      2.220
25
Shucked weight    0.976
00
Viscera weight    0.492
25
Shell weight      0.627
50
Rings            15.500
00
dtype: float64
```

In [ ]:
```
data['Diameter']=np.wher
sns.boxplot(data['Diamet
```

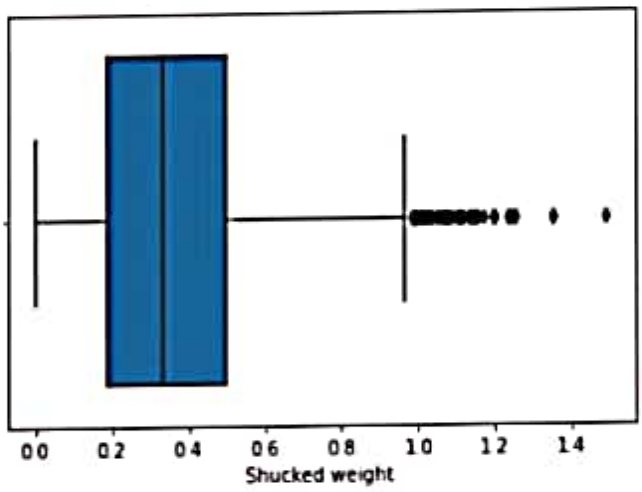
Out[ ]:




In [ ]:
```
sns.boxplot(data['Length
```

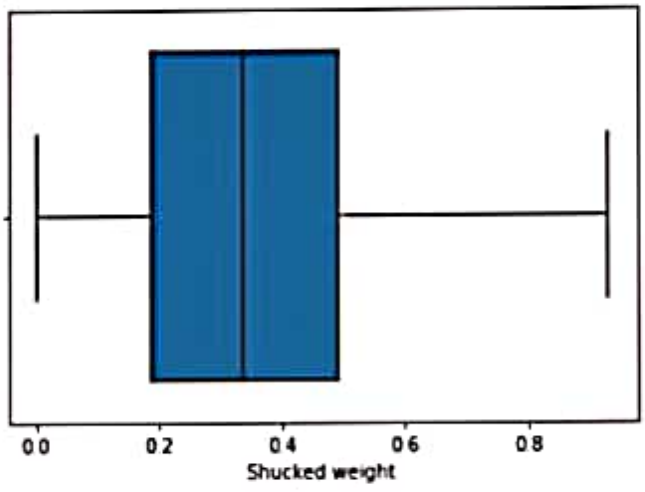
In [ ]:
```
sns.boxplot(data['Shucke
```

Out[ ]:



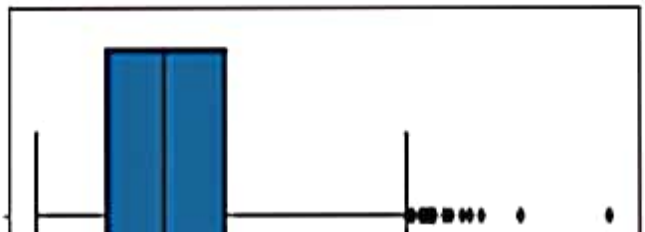

In [ ]:
```
data['Shucked weight']=r
sns.boxplot(data['Shucke
```

Out[ ]:

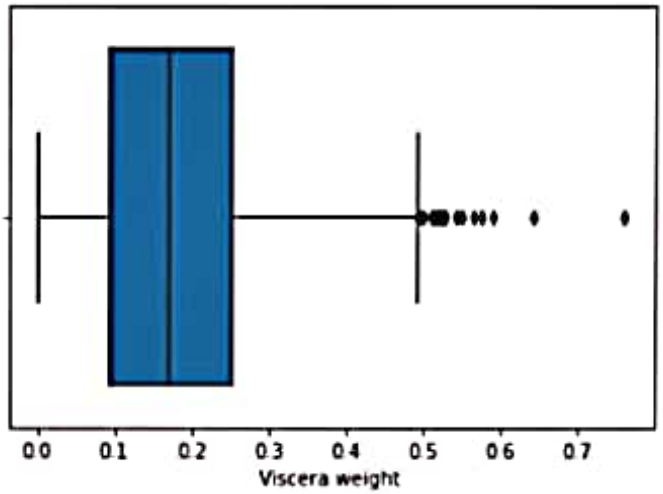



In [ ]:
```
sns.boxplot(data['Viscer
```

Out[ ]:



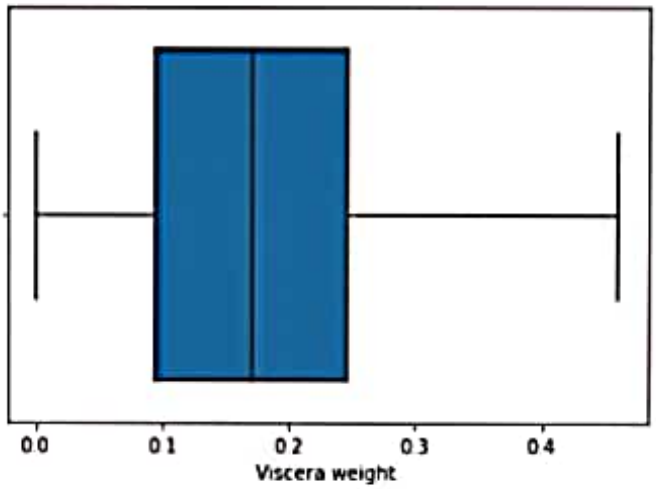
Out[ ]:




In [ ]:

```
data['Viscera weight']=r
sns.boxplot(data['Viscer
```

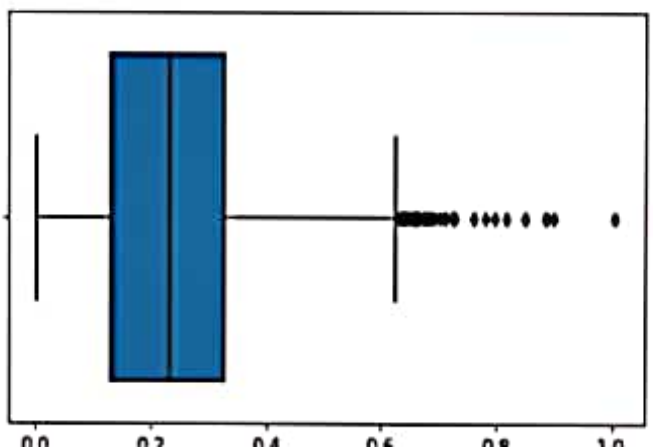
Out[ ]:




In [ ]:

```
sns.boxplot(data['Shell
```

Out[ ]:



In [ ]:

```
x=data.drop(columns= ['F
y=data['Rings']
x
```

Out[ ]:

| | Sex | Length | Diameter | H |
|---|---|---|---|---|
| **0** | 1 | 0.455 | 0.365 | |
| **1** | 1 | 0.350 | 0.265 | |
| **2** | 0 | 0.530 | 0.420 | |
| **3** | 1 | 0.440 | 0.365 | |
| **4** | 2 | 0.330 | 0.255 | |
| **...** | ... | ... | ... | |
| **4172** | 0 | 0.565 | 0.450 | |
| **4173** | 1 | 0.590 | 0.440 | |
| **4174** | 1 | 0.600 | 0.475 | |
| **4175** | 0 | 0.625 | 0.485 | |
| **4176** | 1 | 0.710 | 0.555 | |

4177 rows × 8 columns

In [ ]:

```
y
```

Out[ ]:

```
0        15
1         7
2         9
3        10
4         7
        ..
4172     11
4173     10
4174      9
4175     10
```

## 10.Build the Model

In [ ]:
```
from sklearn.linear_mode
MLR=LinearRegression()
```

## 11.Train the model

In [ ]:
```
MLR.fit(x_train,y_train)
```

Out[ ]:
```
LinearRegression()
```

## 12.Test the model

In [ ]:
```
y_pred=MLR.predict(x_tes
y_pred
```

Out[ ]:
```
array([ 6.3204331 , 10.4
1671748, 13.91911179, 1
2.29316277,  8.7273177 ,
       11.04369928, 12.4
0210281, 11.6992544 , 1
2.01785949,  6.57983392,
       11.91353764, 10.7
9661591, 11.56560952, 1
0.14326497, 13.16762604,
        9.34621768, 10.7
6904478, 11.88283609,
9.34461447, 10.08802992,
       12.80140942,  9.5
8177975, 11.20908126, 1
0.3662699 , 10.0168299 ,
       15.92815446, 15.9
3700213,  7.36066362, 1
3.2889134 , 10.1579858 ,
       11.62833855, 11.0
```

8.27295199, 12.0413644 ,
10.43536813, 11.1
2820999, 10.56478101, 1
2.12900686, 9.0459273 ,
6.50569617, 8.6
5471113, 11.17391657,
4.17641665, 6.45933408,
10.94174559, 10.5
6404265, 7.32806471, 1
0.90718067, 8.76983179,
9.54866214, 9.7
1969088, 9.19215908, 1
1.19107958, 9.95023994,
10.33050587, 11.9
8860703, 5.76011208,
8.82560871, 8.26963359,
6.41006108, 7.6
2776781, 7.77958091, 1
0.53587014, 8.89399096,
11.50322847, 6.4
6552063, 6.62035734, 1
1.27313616, 8.28747988,
12.05544015, 11.6
973709 , 12.73972343, 1
1.36996234, 7.97256548,
9.42073857, 11.2
5296103, 8.05208624, 1
0.99827477, 8.28671759,
11.9443616 , 11.8
2872121, 9.74400382,
8.90145486, 8.57310105,
7.40827472, 11.1
7489105, 10.0697987 ,
9.82070981, 7.33964403,
14.9428325 , 7.7
6026974, 12.77292992,
6.50073351, 11.29473941,
11.88889387, 7.6
7192672, 11.10156897, 1
2.84247625, 6.80849608,
12.6708819 , 9.5
6757524, 10.85921143, 1
0.87947611, 12.27788605,