1.Loading Dataset into tool

In [ ]:
```
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
```

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

In [ ]:
```
data = pd.read_csv("abalone.csv")
```

2.Performing Visualization

Univariate Analysis

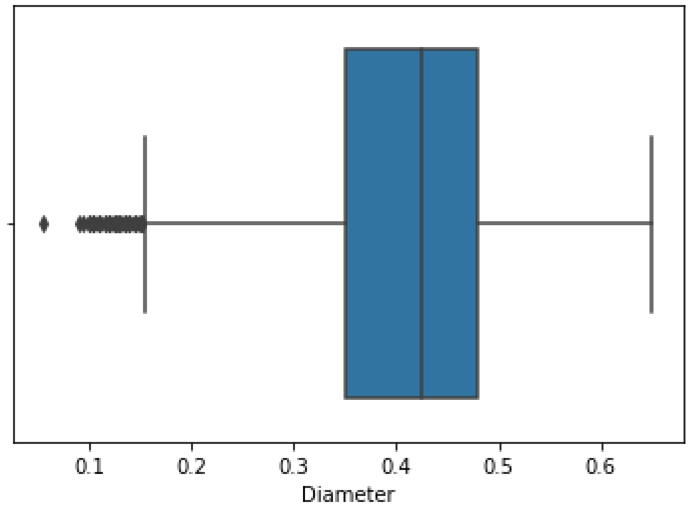
In [ ]:
```
data.head()
```

Out[4]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

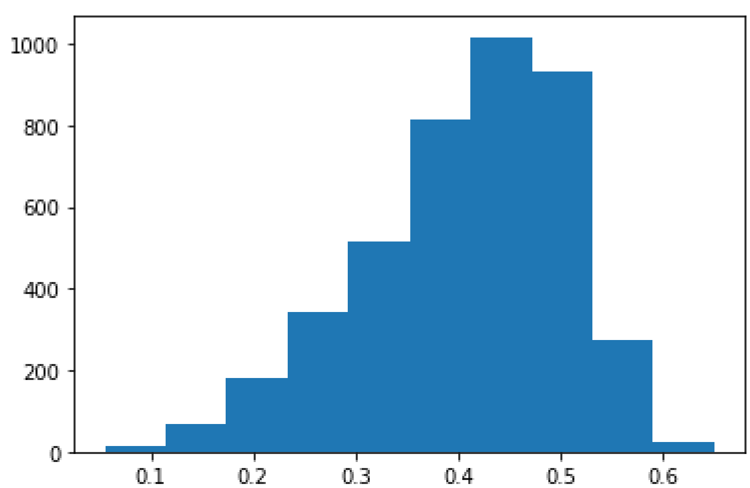
In [ ]:
```
sns.boxplot(data['Diameter'])
```

Out[5]: `<matplotlib.axes._subplots.AxesSubplot at 0x7fcd75630590>`



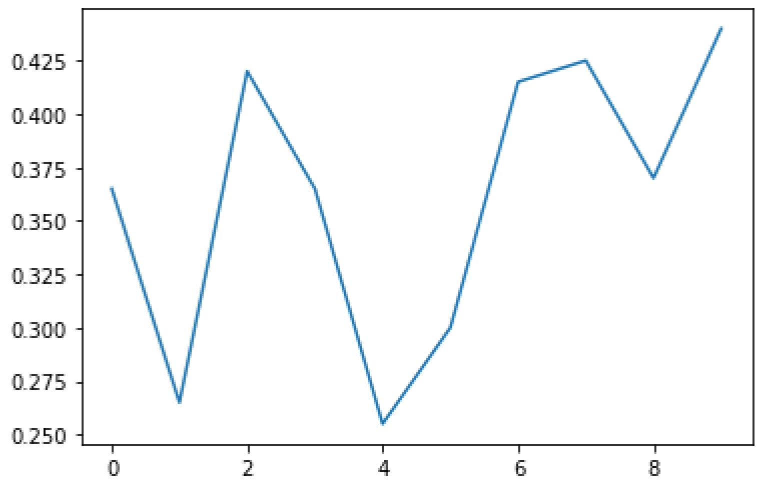
In [ ]:
```
plt.hist(data['Diameter'])
```

Out[6]:
```
(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>)
```



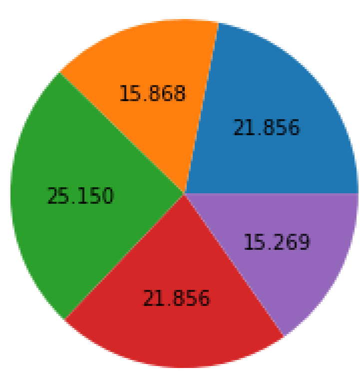
In [ ]:
```
plt.plot(data['Diameter'].head(10))
```

Out[7]:
```
[&lt;matplotlib.lines.Line2D at 0x7fcd750dcdd0>]
```



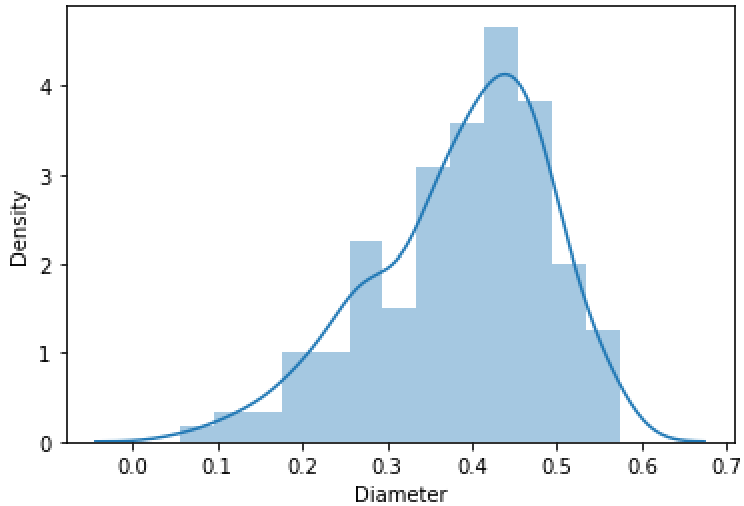
In [ ]:
```
plt.pie(data['Diameter'].head(),autopct='%.3f')
```

Out[8]:
```
([<matplotlib.patches.Wedge at 0x7fcd74fb7a10>,
  <matplotlib.patches.Wedge at 0x7fcd74f44210>,
  <matplotlib.patches.Wedge at 0x7fcd74f44190>,
  <matplotlib.patches.Wedge at 0x7fcd74f4f350>,
  <matplotlib.patches.Wedge at 0x7fcd74f4fe90>],
 [Text(0.8507215626110557, 0.6973326486753676, ''),
  Text(-0.32611344931648134, 1.0505474849691026, ''),
  Text(-1.0998053664078908, -0.02069193128747144, ''),
  Text(-0.08269436219656089, -1.096887251480709, ''),
  Text(0.9758446362287218, -0.5076684409569241, '')],
 [Text(0.46402994324239394, 0.3803632629138369, '21.856'),
  Text(-0.17788006326353525, 0.5730259008922377, '15.868'),
  Text(-0.5998938362224858, -0.011286507974984419, '25.150'),
  Text(-0.045106015743578656, -0.5983021371712958, '21.856'),
  Text(0.5322788924883937, -0.2769100587037768, '15.269')])
```



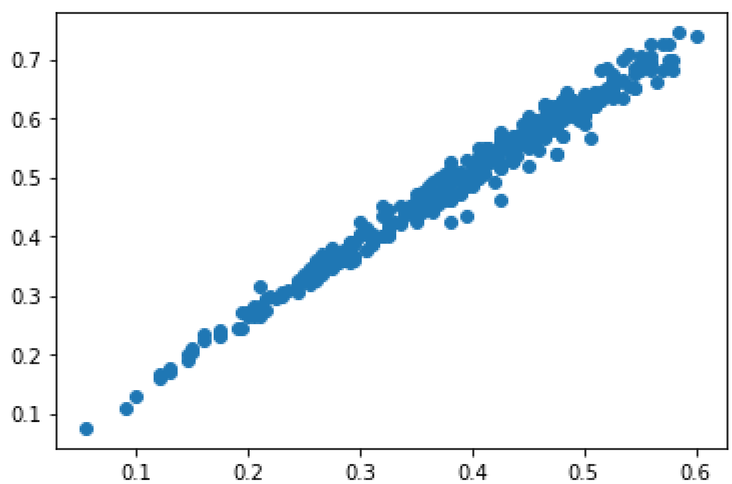
In [ ]:
```
sns.distplot(data['Diameter'].head(300))
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd74f71810>



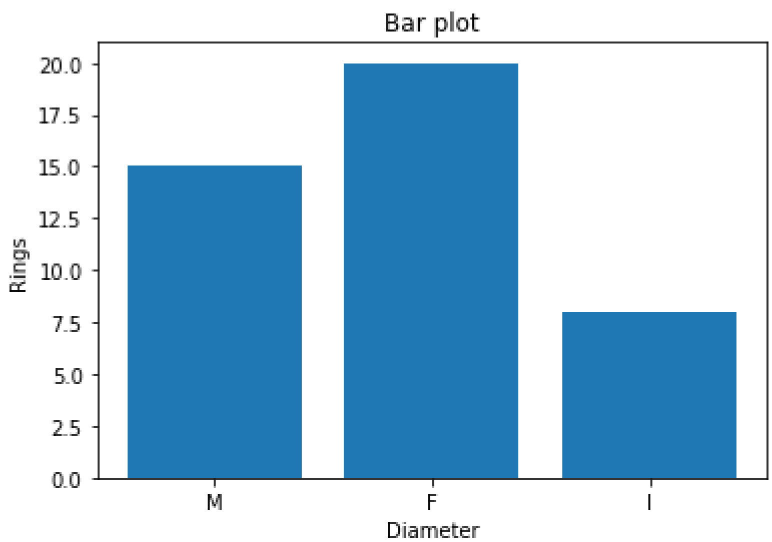
In [ ]:
```
plt.scatter(data['Diameter'].head(400),data['Length'].head(400))
```

Out[11]: <matplotlib.collections.PathCollection at 0x7fcd74f7c5d0>



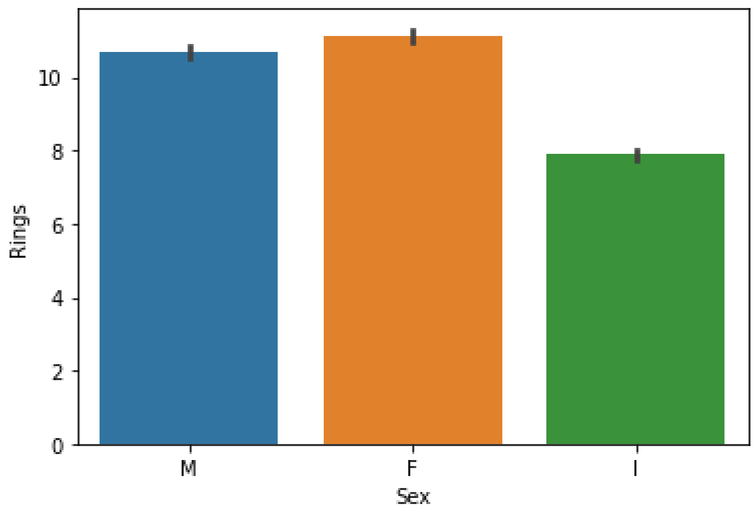
In [ ]:
```
plt.bar(data['Sex'].head(20),data['Rings'].head(20))
plt.title('Bar plot')
plt.xlabel('Diameter')
plt.ylabel('Rings')
```

Out[12]:
```
Text(0, 0.5, 'Rings')
```



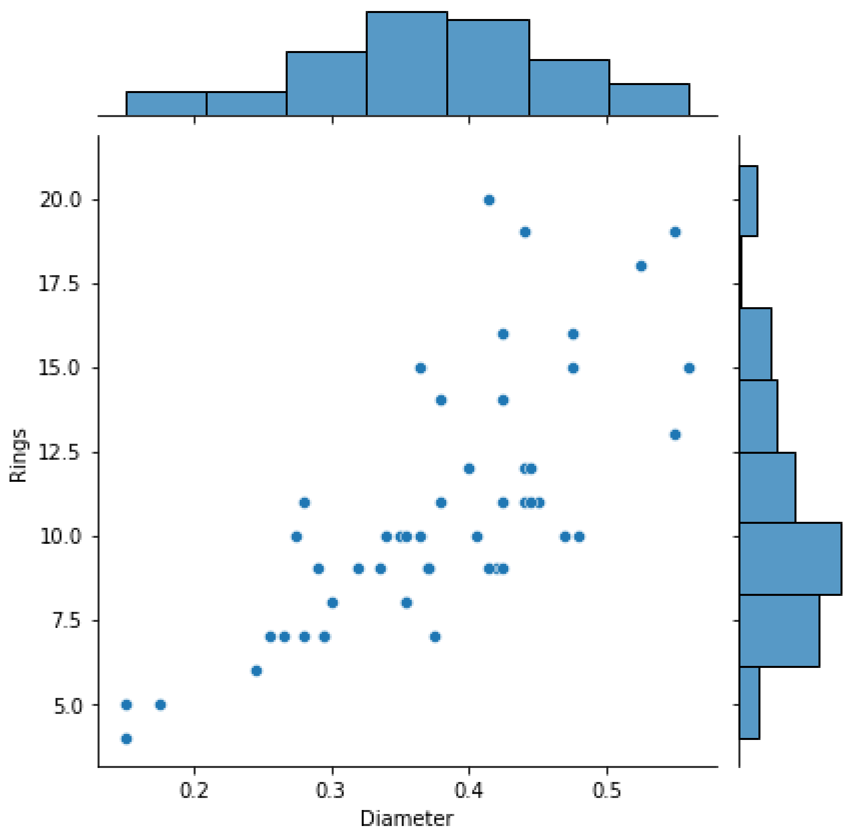
In [ ]:
```
sns.barplot(data['Sex'], data['Rings'])
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd742b0710>



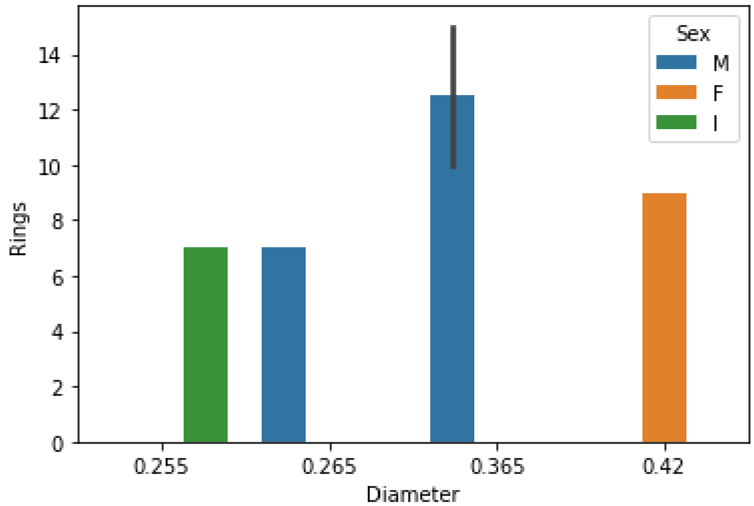
In [ ]:
```
sns.jointplot(data['Diameter'].head(50),data['Rings'].head(100))
```

Out[14]: <seaborn.axisgrid.JointGrid at 0x7fcd7420d210>



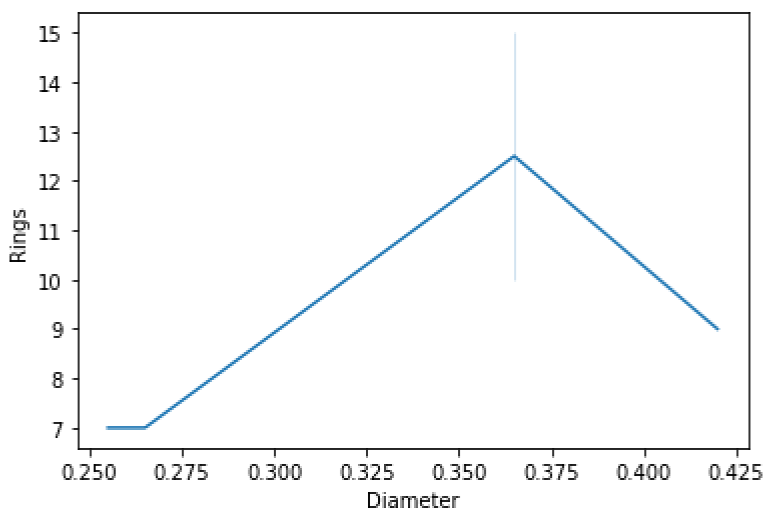
In [ ]:
```
sns.barplot('Diameter','Rings',hue='Sex',data=data.head())
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd71889f90>



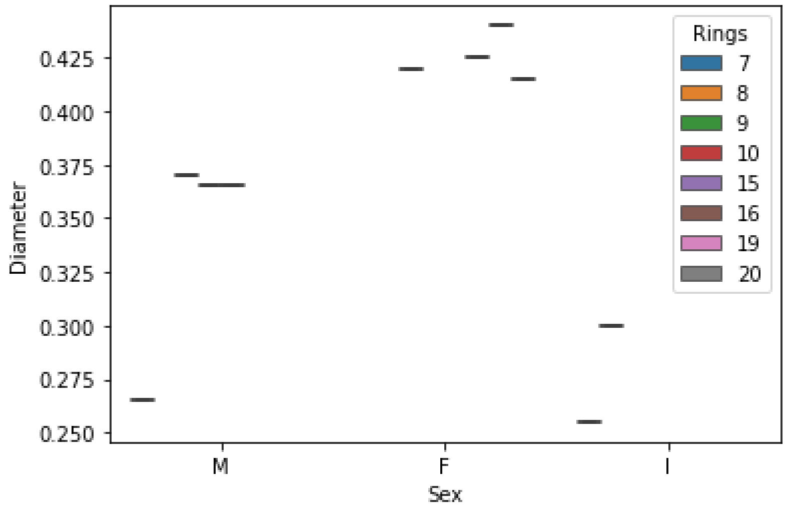
In [ ]:
```
sns.lineplot(data['Diameter'].head(),data['Rings'].head())
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd717c4ad0>



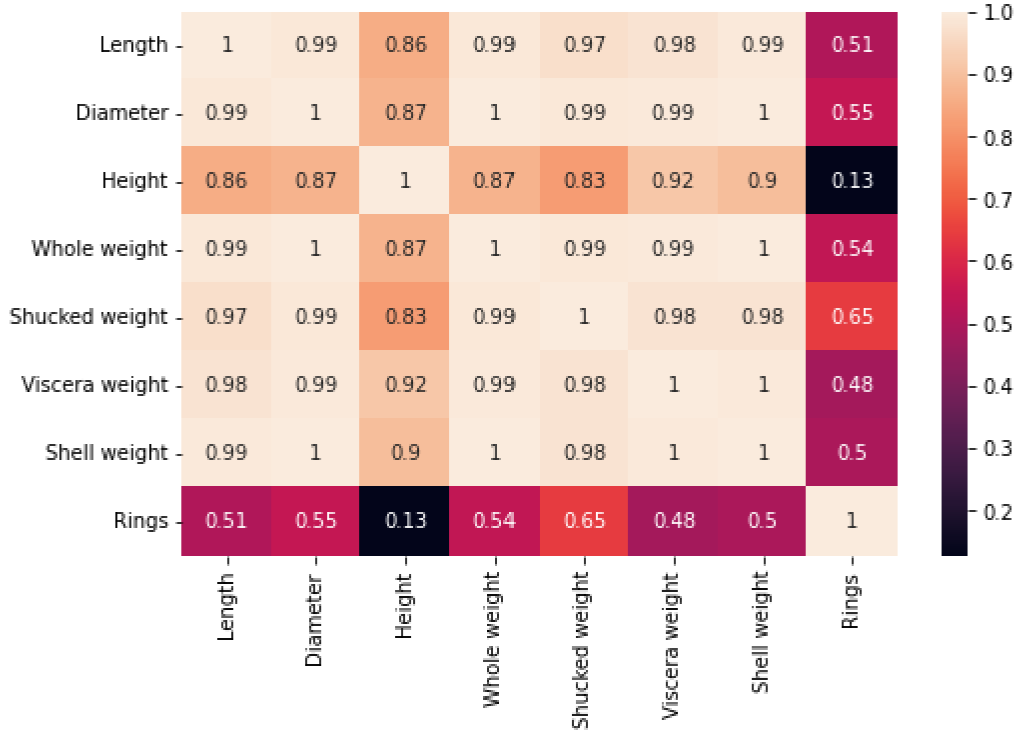
In [ ]:
```
sns.boxplot(data['Sex'].head(10),data['Diameter'].head(10),data['Rings'].head(10)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd71752ad0>
```



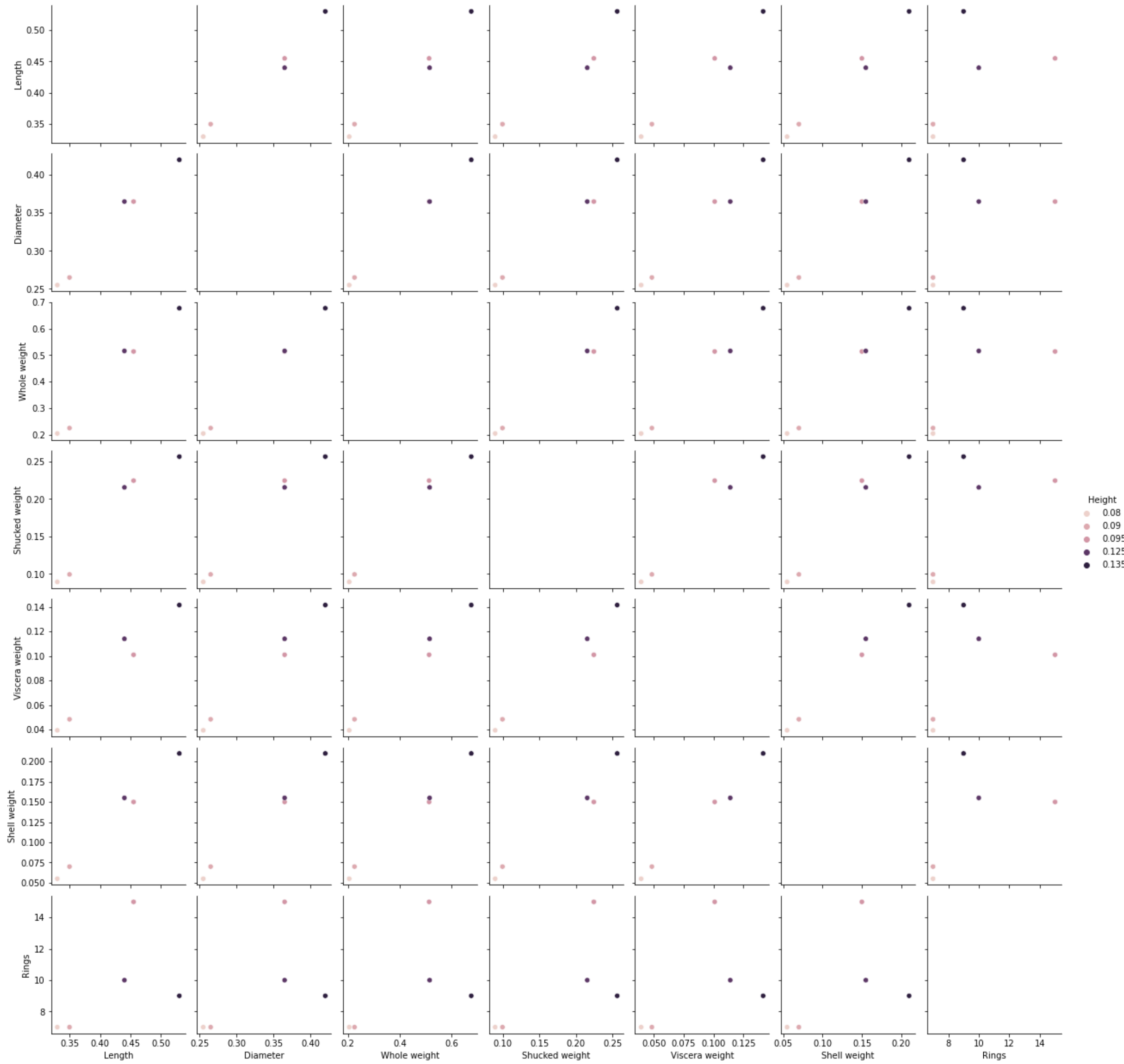
In [ ]:
```
fig=plt.figure(figsize=(8,5))
sns.heatmap(data.head().corr(),annot=True)
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd7160a6d0>



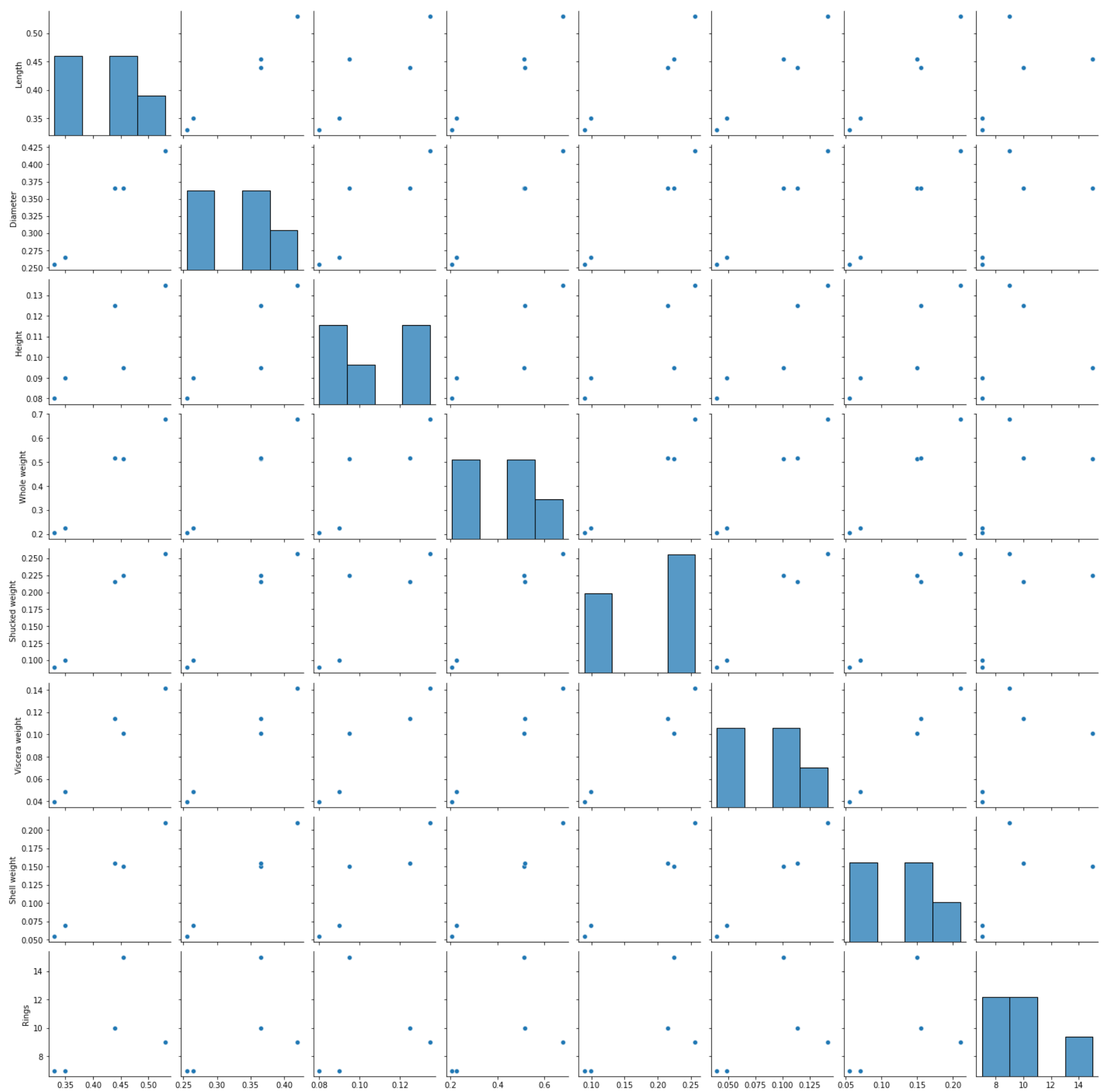
In [ ]:
```
sns.pairplot(data.head(),hue='Height')
```

Out[19]: <seaborn.axisgrid.PairGrid at 0x7fcd7149b090>



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

Out[20]: <seaborn.axisgrid.PairGrid at 0x7fcd6fe9b6d0>



3.Perform Descriptive Statistics on the dataset

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

Out[21]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

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

Out[22]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

In [ ]:
```
data.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   object 
 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(1), object(1)
memory usage: 293.8+ KB
```

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

Out[24]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000

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

Out[25]:

	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.shape
```

Out[26]:
```
(4177, 9)
```

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

Out[27]:
```
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[28]:
```
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
```

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

Out[29]:
```
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[30]:
```
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()
```

Out[31]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
4172	False	False	False	False	False	False	False	False	False
4173	False	False	False	False	False	False	False	False	False
4174	False	False	False	False	False	False	False	False	False
4175	False	False	False	False	False	False	False	False	False
4176	False	False	False	False	False	False	False	False	False

4177 rows × 9 columns

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

Out[32]:
```
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.isna().sum()
```

Out[33]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

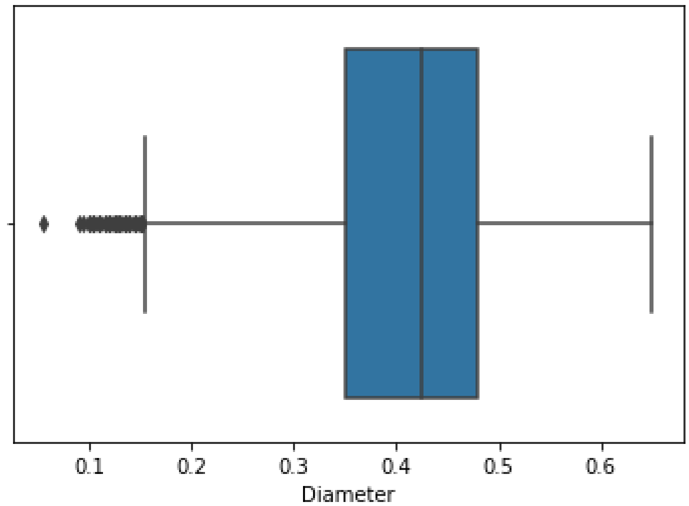
In [ ]:
```
data.isna().any().sum()
```

Out[34]:
```
0
```

5.Find the outliers and replace them outliers

In [ ]:
```
sns.boxplot(data['Diameter'])
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd6cc0b690>
```



In [ ]:
```
quant=data.quantile(q=[0.25,0.75])
quant
```

Out[36]:

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

In [ ]:
```
iqr=quant.loc[0.75]-quant.loc[0.25]
iqr
```

Out[37]:
```
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*iqr)
low
```

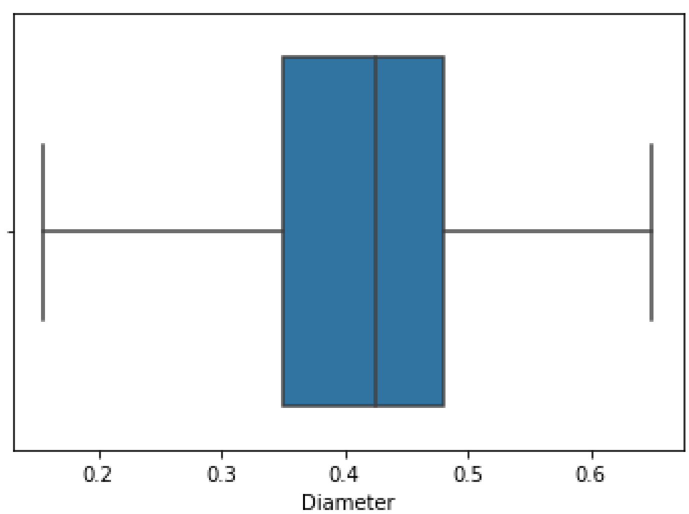
Out[38]:
```
Length            0.20250
Diameter          0.15500
Height            0.04000
Whole weight     -0.62575
Shucked weight   -0.28800
Viscera weight   -0.14575
Shell weight     -0.16850
Rings             3.50000
dtype: float64
```

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

Out[39]:
```
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
```

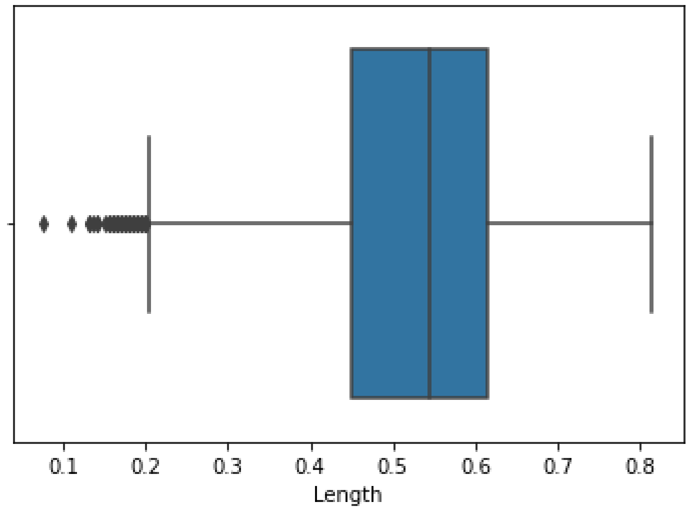
In [ ]:
```
data['Diameter']=np.where(data['Diameter']<0.155,0.4078,data['Diameter'])
sns.boxplot(data['Diameter'])
```

Out[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcd6cbe1510>
```



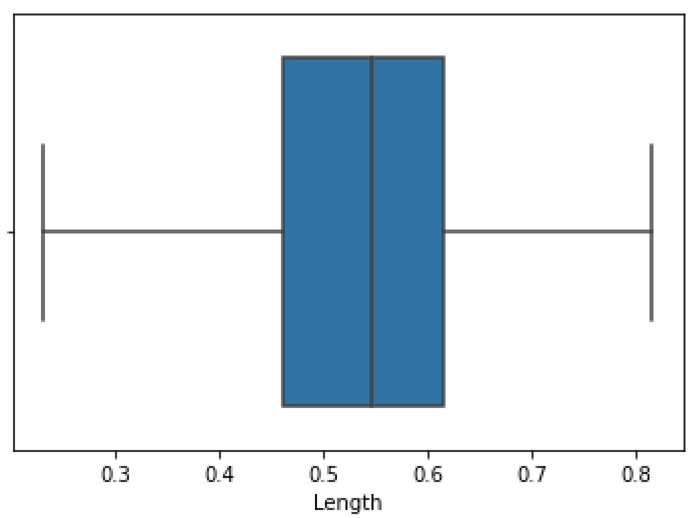
In [ ]:
```
sns.boxplot(data['Length'])
```

Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6cb41410>



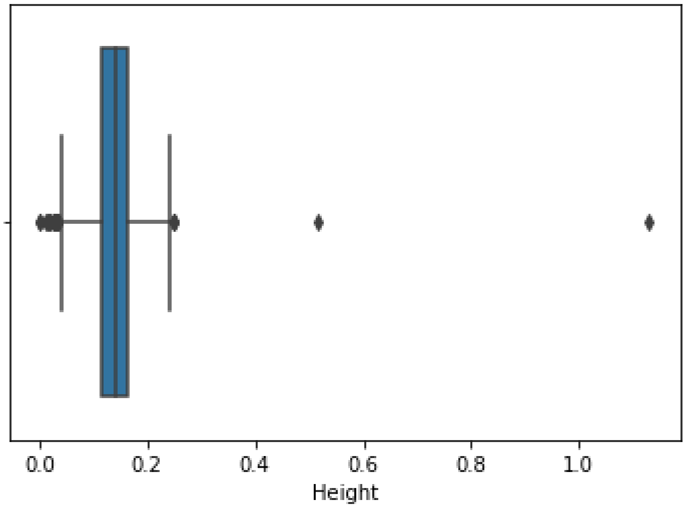
In [ ]:
```
data['Length']=np.where(data['Length']<0.23,0.52, data['Length'])
sns.boxplot(data['Length'])
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6cb31350>



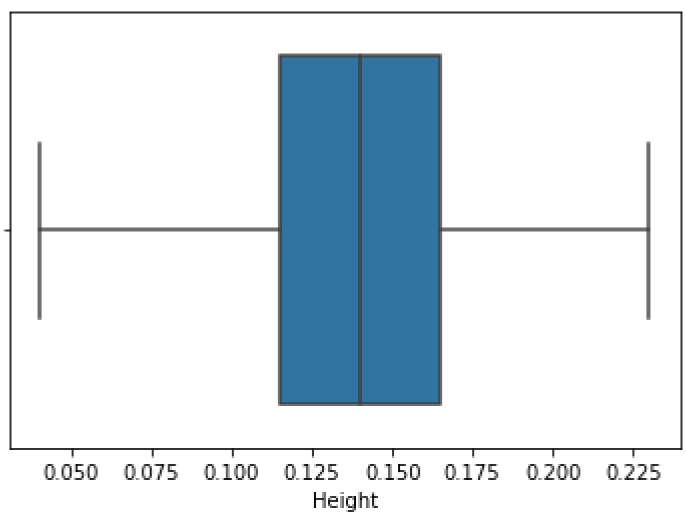
In [ ]:
```
sns.boxplot(data['Height'])
```

Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6ca91950>



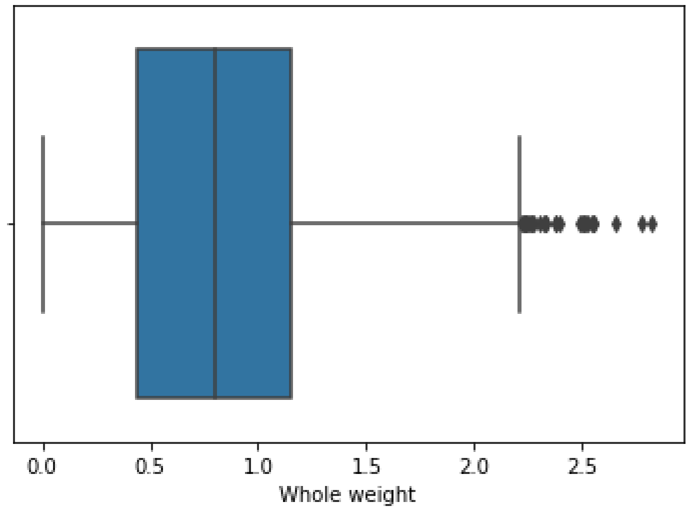
In [ ]:
```
data['Height']=np.where(data['Height']<0.04,0.139, data['Height'])
data['Height']=np.where(data['Height']>0.23,0.139, data['Height'])
sns.boxplot(data['Height'])
```

Out[44]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6ca82050>



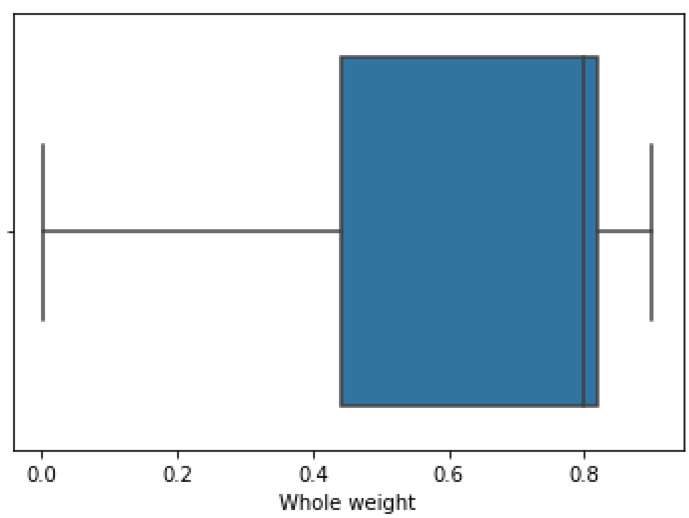
In [ ]:
```
sns.boxplot(data['Whole weight'])
```

Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6c9f2090>



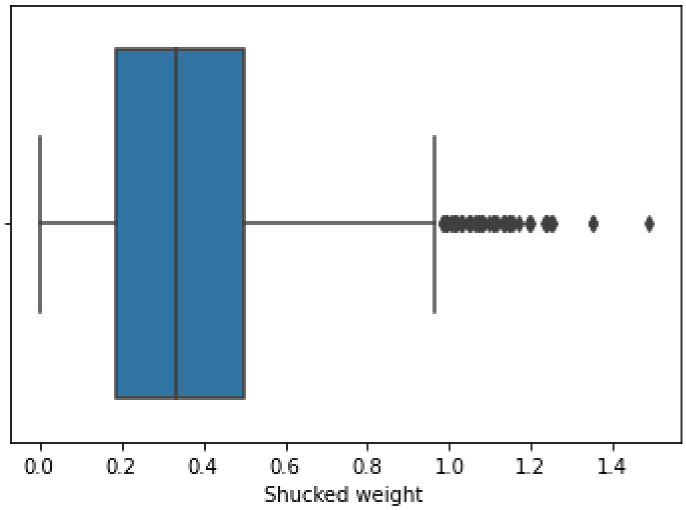
In [ ]:
```
data['Whole weight']=np.where(data['Whole weight']>0.9,0.82, data['Whole weight']
sns.boxplot(data['Whole weight'])
```

Out[46]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6c9e9390>



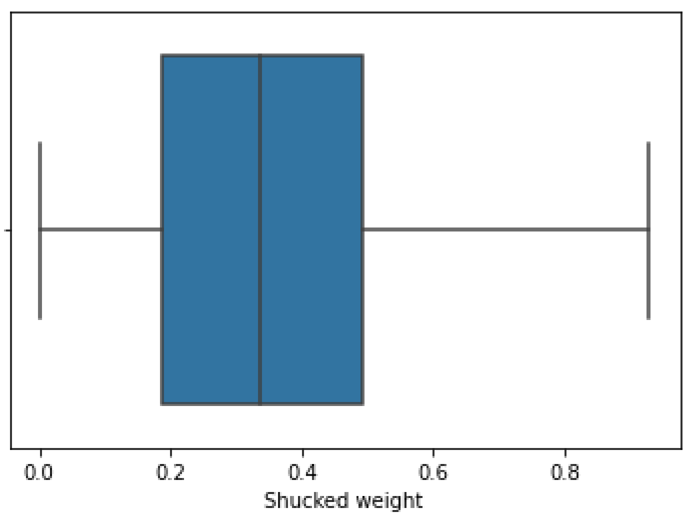
In [ ]:
```
sns.boxplot(data['Shucked weight'])
```

Out[47]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6c8c3f10>



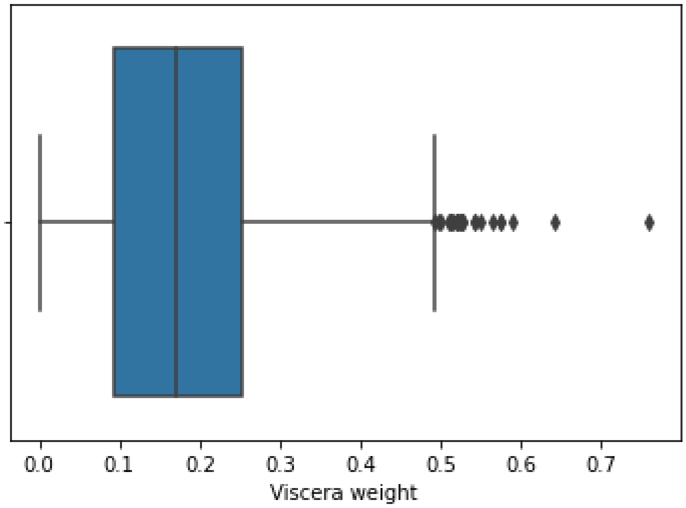
In [ ]:
```
data['Shucked weight']=np.where(data['Shucked weight']>0.93,0.35, data['Shucked w
sns.boxplot(data['Shucked weight'])
```

Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6c836f50>



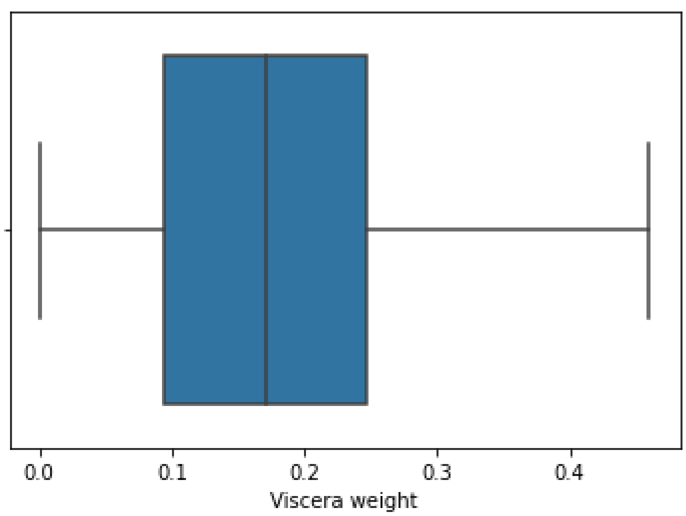
In [ ]:
```
sns.boxplot(data['Viscera weight'])
```

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6c801ad0>



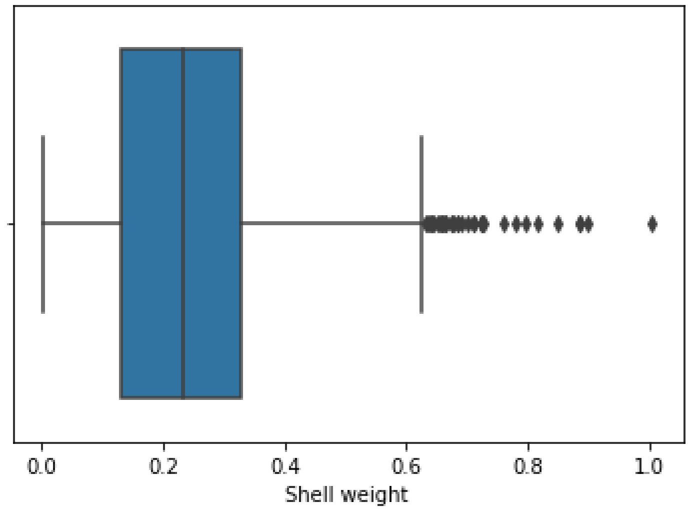
In [ ]:
```
data['Viscera weight']=np.where(data['Viscera weight']>0.46,0.18, data['Viscera w
sns.boxplot(data['Viscera weight'])
```

Out[50]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6c785b90>



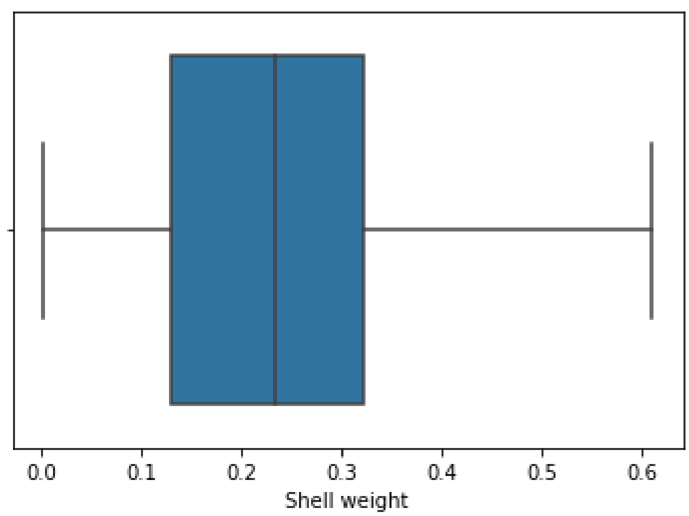
In [ ]:
```
sns.boxplot(data['Shell weight'])
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6c768b50>



In [ ]:
```
data['Shell weight']=np.where(data['Shell weight']>0.61,0.2388, data['Shell weigh
sns.boxplot(data['Shell weight'])
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x7fcd6c6df850>



6.Check for Categorical columns and perform encoding.

In [ ]:
```
data['Sex'].replace({'M':1,'F':0,'I':2},inplace=True)
data
```

Out[53]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	1	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	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	2	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	1	0.590	0.440	0.135	0.8200	0.4390	0.2145	0.2605	10
4174	1	0.600	0.475	0.205	0.8200	0.5255	0.2875	0.3080	9
4175	0	0.625	0.485	0.150	0.8200	0.5310	0.2610	0.2960	10
4176	1	0.710	0.555	0.195	0.8200	0.3500	0.3765	0.4950	12

4177 rows × 9 columns

7.Split the data into dependent and independent variables.

In [ ]:
```
x=data.drop(columns= ['Rings'])
y=data['Rings']
x
```

Out[54]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550
...	...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490
4173	1	0.590	0.440	0.135	0.8200	0.4390	0.2145	0.2605
4174	1	0.600	0.475	0.205	0.8200	0.5255	0.2875	0.3080
4175	0	0.625	0.485	0.150	0.8200	0.5310	0.2610	0.2960
4176	1	0.710	0.555	0.195	0.8200	0.3500	0.3765	0.4950

4177 rows × 8 columns

In [ ]:
```
y
```

Out[55]:
```
0       15
1        7
2        9
3       10
4        7
        ..
4172    11
4173    10
4174     9
4175    10
4176    12
Name: Rings, Length: 4177, dtype: int64
```

8.Scale the independent variables

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

Out[56]:
```
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_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2)
print(x_train.shape, x_test.shape)
```

```
(3341, 8) (836, 8)
```

10.Build the Model

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

11.Train the model

In [ ]:
```
MLR.fit(x_train,y_train)
```
Out[59]:
```
LinearRegression()
```

12.Test the model

In [ ]:
```
y_pred=MLR.predict(x_test)
y_pred
```
Out[60]:
```
array([ 6.3204331 , 10.41671748, 13.91911179, 12.29316277,  8.7273177 ,
       11.04369928, 12.40210281, 11.6992544 , 12.01785949,  6.57983392,
       11.91353764, 10.79661591, 11.56560952, 10.14326497, 13.16762604,
        9.34621768, 10.76904478, 11.88283609,  9.34461447, 10.08802992,
       12.80140942,  9.58177975, 11.20908126, 10.3662699 , 10.0168299 ,
       15.92815446, 15.93700213,  7.36066362, 13.2889134 , 10.1579858 ,
       11.62833855, 11.08597007, 11.60253151, 11.74194458,  9.75151497,
        9.16685512,  7.93960537, 10.04563481, 10.81773394, 10.55133893,
        7.19389026,  9.30303442, 10.83957317, 10.63432914, 10.19371808,
       13.47423856,  9.06825076,  6.69843582, 13.38213142,  9.62823486,
        8.20174551,  7.79183041,  9.3338472 , 11.08195328, 11.25321895,
        6.11231204, 10.6960639 ,  9.23348159,  7.76425036, 11.65342323,
       12.6024271 ,  7.49694081,  9.71678931,  7.41119139,  6.94925679,
        6.34706174,  9.99734923,  6.70117631, 10.71374432,  9.59457302,
        7.07847213,  6.6940933 ,  9.30356123, 13.66698224,  9.71369221,
       17.36952958,  7.81225327,  8.86909973,  9.29540502, 11.03405521,
       12.90720962, 13.03952065,  4.90843127,  9.50619996, 10.09434256,
        8.67296752,  9.03746047,  8.33310609, 10.60445018,  9.66636969,
        7.67351279,  8.74447193, 12.37470593,  7.70552082, 11.35599144,
```

In [ ]:
```
pred=MLR.predict(x_train)
pred
```
Out[61]:
```
array([9.67807776, 9.90237308, 8.732808  , ..., 8.23154309, 9.17793652,
       8.04066563])
```

In [ ]:
```
from sklearn.metrics import r2_score
accuracy=r2_score(y_test,y_pred)
accuracy
```
Out[62]:
```
0.45246173731319095
```

In [ ]:
```
MLR.predict([[1,0.455,0.365,0.095,0.5140,0.2245,0.1010,0.150]])
```
Out[63]:
```
array([9.88121105])
```

13.Measure the performance using Metrics

In [ ]:
```
from sklearn import metrics
from sklearn.metrics import mean_squared_error
np.sqrt(mean_squared_error(y_test,y_pred))
```
Out[64]:
```
2.426157459129611
```

LASSO

In [ ]:
```
from sklearn.linear_model import Lasso, Ridge
#intialising model
lso=Lasso(alpha=0.01,normalize=True)
#fit the model
lso.fit(x_train,y_train)
Lasso(alpha=0.01, normalize=True)
#prediction on test data
lso_pred=lso.predict(x_test)
#coef
coef=lso.coef_
coef
```

Out[65]:
```
array([-0.        ,  0.        ,  0.        ,  0.47895382,  0.1231748 ,
        0.        ,  0.        ,  0.84464209])
```

In [ ]:
```
from sklearn import metrics
from sklearn.metrics import mean_squared_error
metrics.r2_score(y_test,lso_pred)
```

Out[66]:
```
0.3408644820717798
```

In [ ]:
```
np.sqrt(mean_squared_error(y_test,lso_pred))
```

Out[67]:
```
2.661945158379675
```

RIDGE

In [ ]:
```
#initialising model
rg=Ridge(alpha=0.01,normalize=True)
#fit the model
rg.fit(x_train,y_train)
Ridge(alpha=0.01, normalize=True)
#prediction
rg_pred=rg.predict(x_test)
rg_pred
```

Out[68]:
```
array([ 6.31931908, 10.30764994, 13.79582136, 12.31898366,  8.76153971,
       11.03161104, 12.36947473, 11.61494959, 11.9751636 ,  6.61427002,
       11.96025268, 10.72794019, 11.47832347, 10.11563414, 13.06595338,
        9.39260908, 10.76339441, 11.91725195,  9.36307394, 10.08739488,
       12.81067168,  9.60509865, 11.22161077, 10.34000965,  9.99490475,
       15.73170012, 15.73827506,  7.39070197, 13.28647279, 10.27222883,
       11.60238358, 11.12815632, 11.54610466, 11.74210077,  9.74066812,
        9.18758732,  7.95443356,  9.97019442, 10.84527542, 10.5864829 ,
        7.21708698,  9.26208697, 10.7752225 , 10.59013091, 10.22155425,
       13.35380749,  9.15950505,  6.7399079 , 13.2683363 ,  9.60102394,
        8.2303643 ,  7.8098864 ,  9.39868717, 11.12458359, 11.22465236,
        6.08517442, 10.71988191,  9.22838517,  7.83437767, 11.55747904,
       12.53949383,  7.51301724,  9.78148647,  7.37997405,  6.95728771,
        6.35737948,  9.93938109,  6.69320708, 10.65733704,  9.63910534,
        7.08460623,  6.75306126,  9.35523418, 13.54420957,  9.75927226,
       17.10845005,  7.80794412,  8.86341648,  9.31305116, 10.93831159,
       12.7937996 , 12.90820043,  5.0134048 ,  9.52092556, 10.09121624,
        8.69256796,  9.05325416,  8.38837108, 10.60016343,  9.6674175 ,
        7.68947352,  8.75395963, 12.3557545 ,  7.68394763, 11.31578086,
```

In [ ]:
```
rg.coef_
```

Out[69]:
```
array([-0.3074739 , -0.73150514,  0.23303655,  0.99723138,  0.94304227,
       -1.36153292, -0.05594202,  1.75904754])
```

In [ ]:
```
metrics.r2_score(y_test,rg_pred)
```

Out[70]:
```
0.45111716055161055
```

In [ ]:
```
np.sqrt(mean_squared_error(y_test,rg_pred))
```

Out[71]:
```
2.4291345612955157
```