```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy
from scipy import stats
from sklearn.preprocessing import OneHotEncoder
```

In [32]:

```
dataset = pd.read_csv("C:\\Users\\Devi\\Downloads\\abalone.csv")
```

In [33]:

```
dataset
```

Out[33]:

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
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

4177 rows × 9 columns

Univariate Analysis

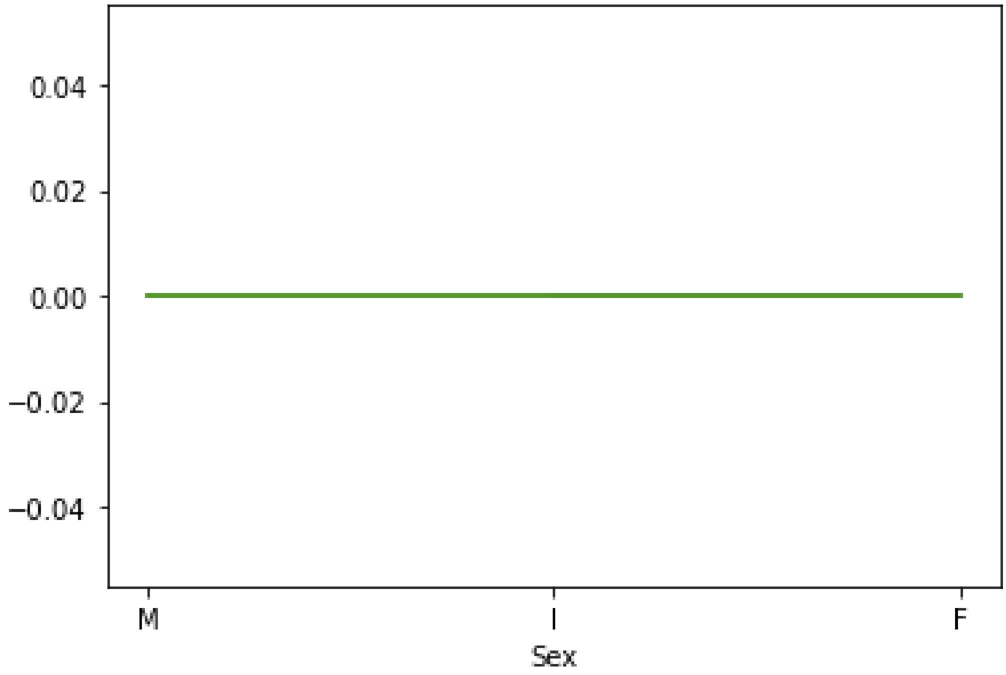
In [17]:

```
df_1=dataset.loc[dataset['Rings']==7]
df_2=dataset.loc[dataset['Rings']==9]
```

```
df_3=dataset.loc[dataset['Rings']==10]
```

In [18]:

```
plt.plot(df_1['Sex'],np.zeros_like(df_1['Sex']))
plt.plot(df_2['Sex'],np.zeros_like(df_2['Sex']))
plt.plot(df_3['Sex'],np.zeros_like(df_3['Sex']))
plt.xlabel('Sex')
plt.show()
```

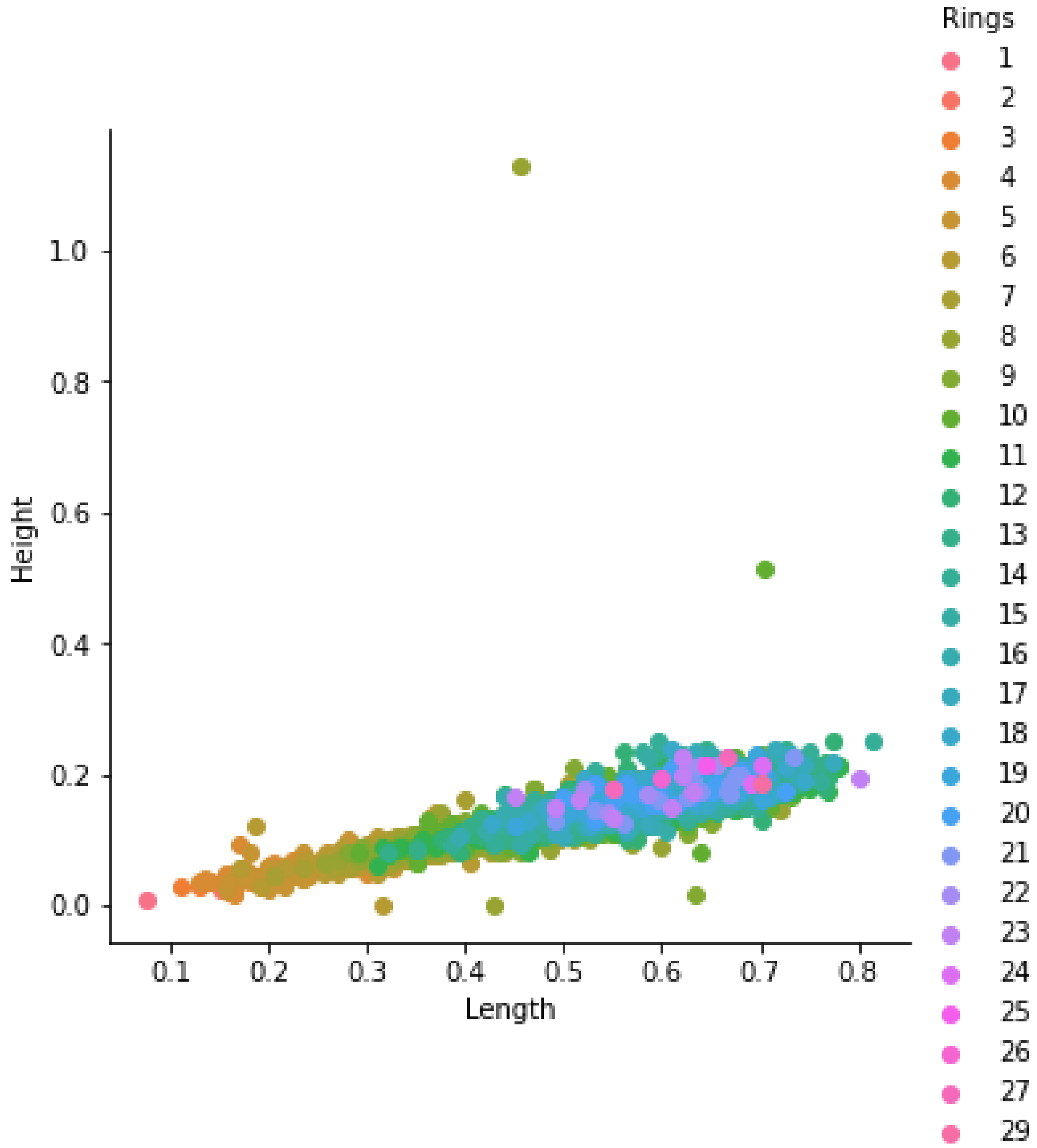


Bi-Variate Analysis

In [21]:

```
sns.FacetGrid(dataset,hue="Rings",size=5).map(plt.scatter,"Length","Height"
).add_legend();
```

```
C:\Users\Devi\anaconda3\lib\site-packages\seaborn\axisgrid.py:337: UserWarn
ing: The `size` parameter has been renamed to `height`; please update your
code.
  warnings.warn(msg, UserWarning)
```



Multi-Variate Analysis

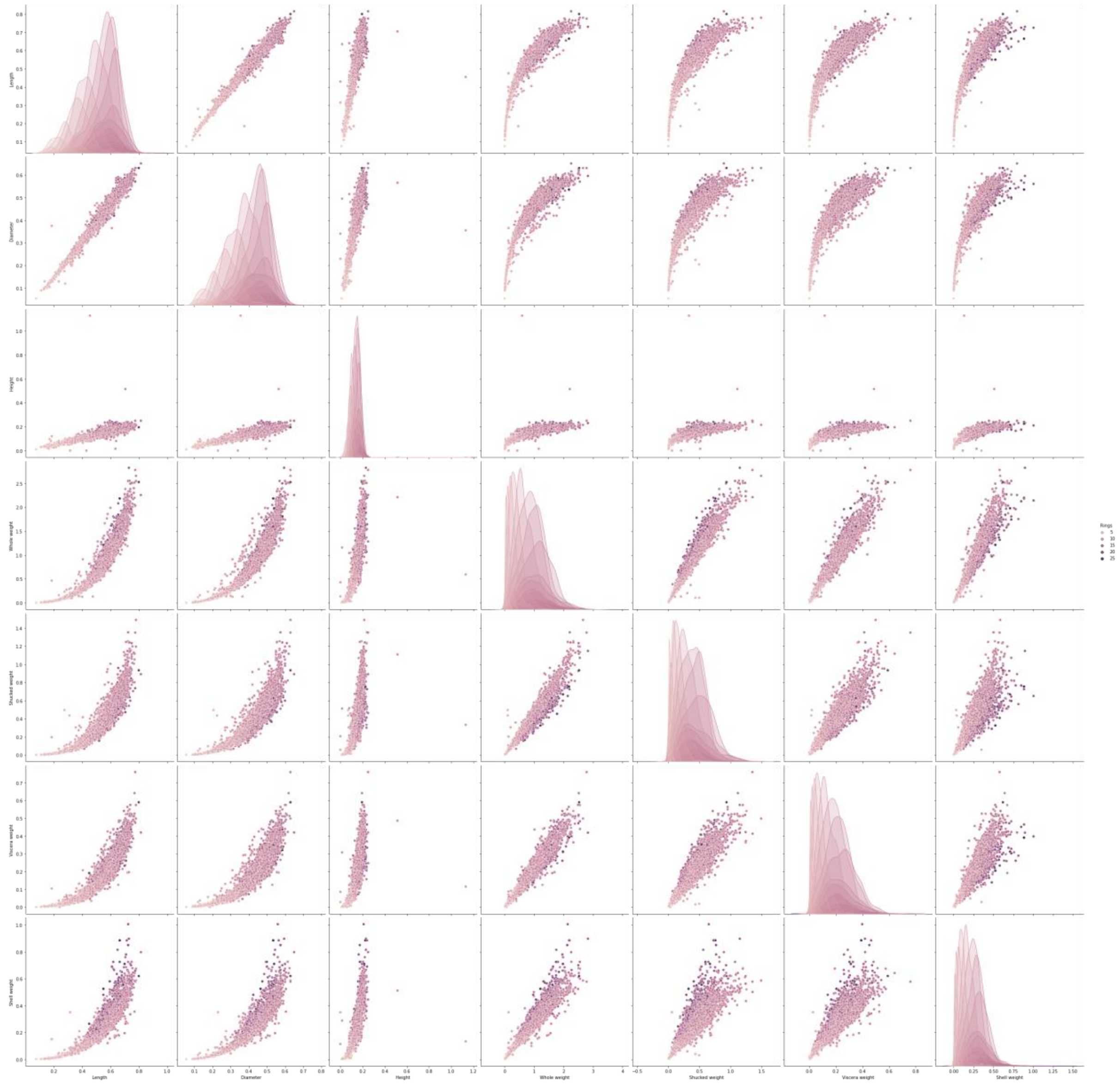
In [23]:

```
sns.pairplot(dataset,hue="Rings",size=5)
```

```
C:\Users\Devi\anaconda3\lib\site-packages\seaborn\axisgrid.py:2076: UserWar
ning: The `size` parameter has been renamed to `height`; please update your
code.
  warnings.warn(msg, UserWarning)
```

Out[23]:

```
<seaborn.axisgrid.PairGrid at 0x1e71ab33d60>
```



Descriptive statistics

In [24]:

```
dataset.sum()
```

Out[24]:

```
Sex               MMFMIIFFMFFMMFFMIFMMMIFFFFFMMMMFMFFMFFFMFFIIII...
Length                                                     2188.715
Diameter                                                    1703.72
Height                                                       582.76
Whole weight                                               3461.656
Shucked weight                                             1501.078
Viscera weight                                             754.3395
Shell weight                                               997.5965
Rings                                                         41493
dtype: object
```

In [25]:

```
dataset.sum(axis=1)
```

```
C:\Users\Devi\AppData\Local\Temp\ipykernel_6272\3445892410.py:1: FutureWarn
ing: Dropping of nuisance columns in DataFrame reductions (with 'numeric_on
ly=None') is deprecated; in a future version this will raise TypeError.  Se
lect only valid columns before calling the reduction.
  dataset.sum(axis=1)
```

Out[25]:

```
0       16.9045
1        8.1485
2       11.3700
3       11.9305
4        8.0540
         ...
4172    13.9250
4173    13.0450
4174    12.5770
4175    13.4425
4176    17.2255
Length: 4177, dtype: float64
```

In [26]:

```
dataset.median()
```

```
C:\Users\Devi\AppData\Local\Temp\ipykernel_6272\4167803218.py:1: FutureWarn
ing: Dropping of nuisance columns in DataFrame reductions (with 'numeric_on
ly=None') is deprecated; in a future version this will raise TypeError.  Se
lect only valid columns before calling the reduction.
  dataset.median()
```

Out[26]:

```
Length            0.5450
Diameter          0.4250
Height            0.1400
Whole weight      0.7995
Shucked weight    0.3360
Viscera weight    0.1710
Shell weight      0.2340
Rings             9.0000
dtype: float64
```

In [27]:

```
dataset.mean()
```

```
C:\Users\Devi\AppData\Local\Temp\ipykernel_6272\1799472221.py:1: FutureWarn
ing: Dropping of nuisance columns in DataFrame reductions (with 'numeric_on
ly=None') is deprecated; in a future version this will raise TypeError.  Se
lect only valid columns before calling the reduction.
  dataset.mean()
```

Out[27]:

```
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
```

In [28]:

```
dataset.max()
```

Out[28]:

```
Sex                    M
Length             0.815
Diameter            0.65
Height              1.13
Whole weight      2.8255
Shucked weight     1.488
Viscera weight      0.76
Shell weight       1.005
Rings                 29
dtype: object
```

In [29]:

```
dataset.std()
```

```
C:\Users\Devi\AppData\Local\Temp\ipykernel_6272\178401259.py:1: FutureWarni
ng: Dropping of nuisance columns in DataFrame reductions (with 'numeric_onl
y=None') is deprecated; in a future version this will raise TypeError.  Sel
ect only valid columns before calling the reduction.
  dataset.std()
```

Out[29]:

```
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
```

In [30]:

```
dataset.var()
```

```
C:\Users\Devi\AppData\Local\Temp\ipykernel_6272\2458428038.py:1: FutureWarn
ing: Dropping of nuisance columns in DataFrame reductions (with 'numeric_on
ly=None') is deprecated; in a future version this will raise TypeError.  Se
lect only valid columns before calling the reduction.
  dataset.var()
```

Out[30]:

```
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 [32]:

```
Rings=dataset.Rings
Rings.value_counts()
```

Out[32]:

```
9     689
10    634
8     568
11    487
7     391
12    267
6     259
```

```
13    203
14    126
5     115
15    103
16     67
17     58
4      57
18     42
19     32
20     26
3      15
21     14
23      9
22      6
27      2
24      2
1       1
26      1
29      1
2       1
25      1
Name: Rings, dtype: int64
```

In [33]:

```
dataset.describe()
```

Out[33]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	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	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

Missing values

In [34]:

```
dataset.shape
```

Out[34]:

```
(4177, 9)
```

In [35]:

```
dataset.isnull()
```

Out[35]:

	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 [36]:

```
dataset.isnull().sum()
```

Out[36]:

```
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 [37]:

```
dataset.isnull().sum().sum()
```

Out[37]:

```
0
```

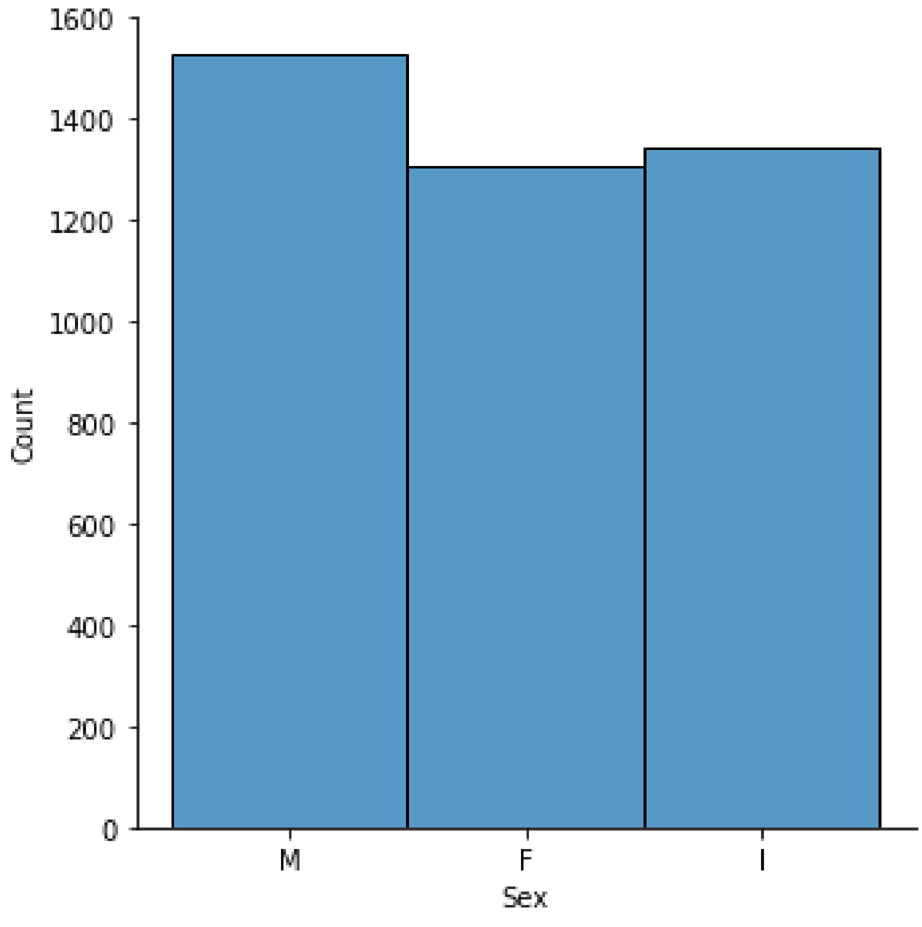
outliers

In [38]:

```
sns.displot(dataset['Sex'])
```

Out[38]:

```
<seaborn.axisgrid.FacetGrid at 0x1e721679820>
```

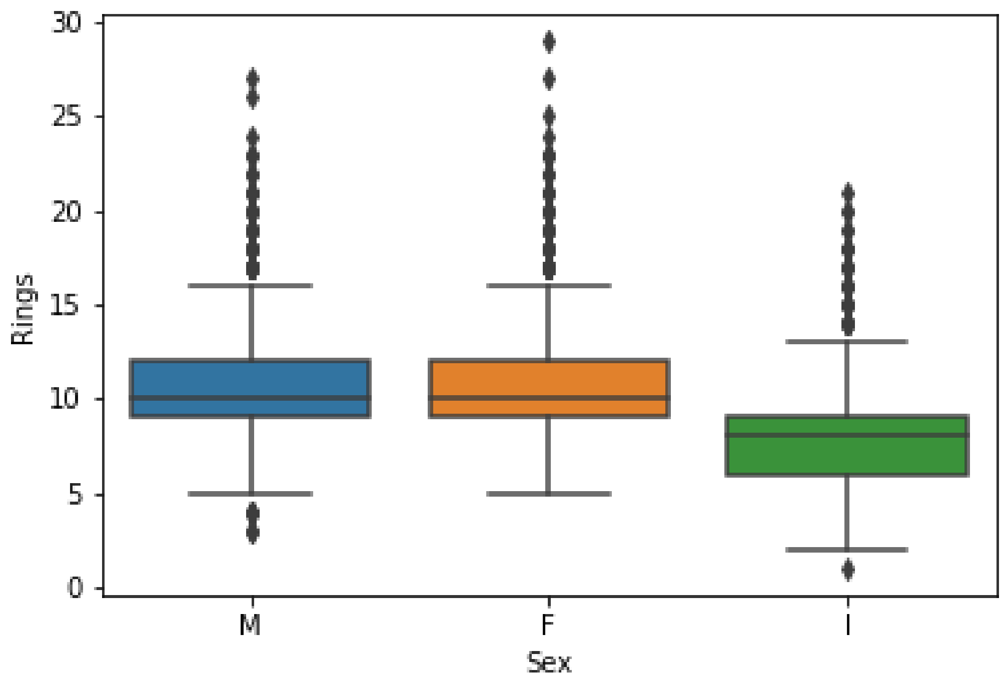


In [39]:

```
sns.boxplot(x='Sex',y='Rings',data=dataset)
```

Out[39]:

```
<AxesSubplot:xlabel='Sex', ylabel='Rings'>
```

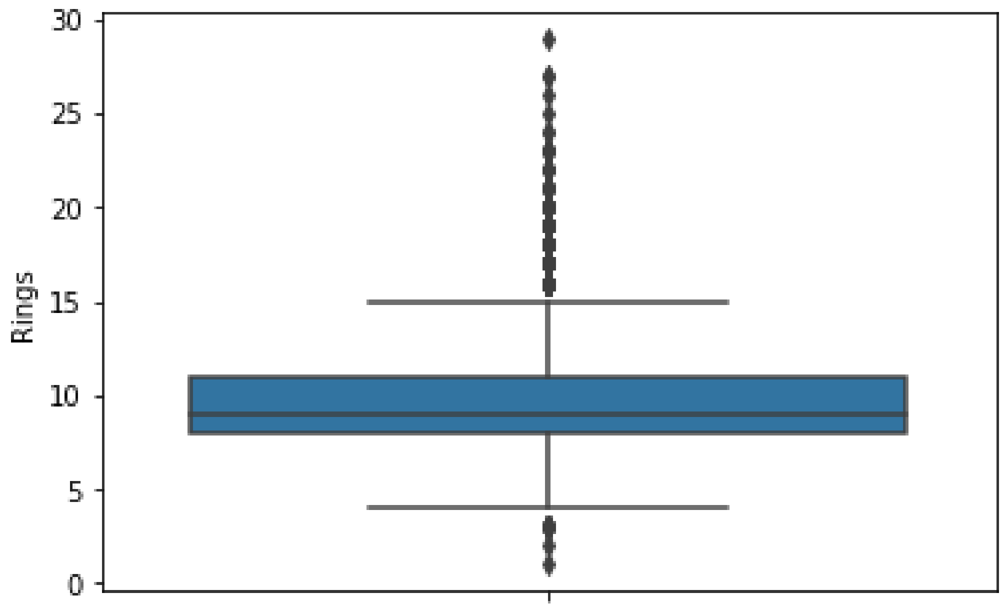


In [40]:

```
sns.boxplot(y='Rings',data=dataset)
```

Out[40]:

```
<AxesSubplot:ylabel='Rings'>
```



In [10]:

```
dataset['Rings'].mean()
```

Out[10]:

```
9.933684462532918
```

In [11]:

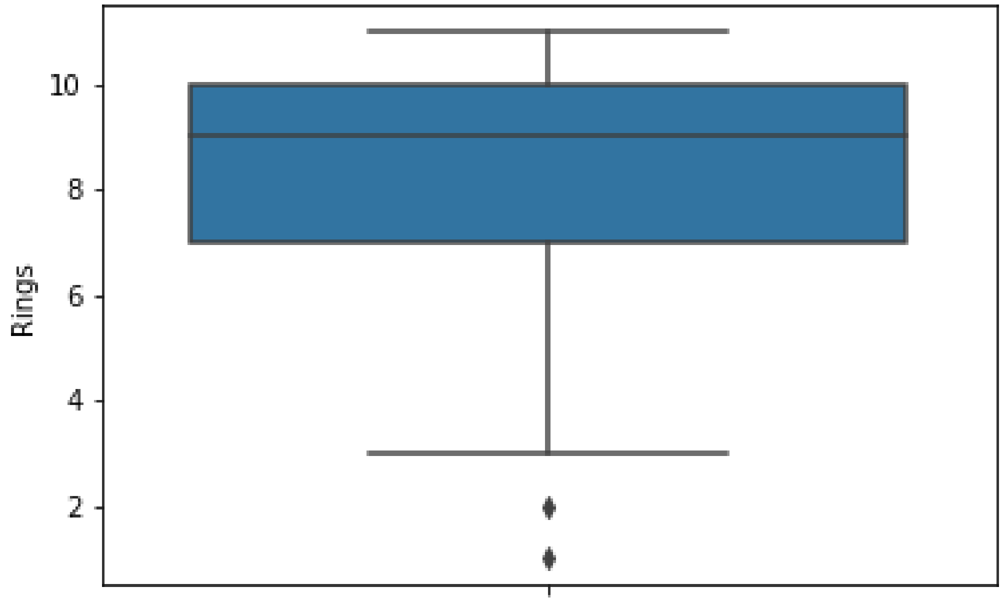
```
data1=dataset[dataset['Rings']<12]
```

In [12]:

```
sns.boxplot(y='Rings',data=data1)
```

Out[12]:

<AxesSubplot:ylabel='Rings'>



Categorial Encoding

In [13]:

```
data_tips=pd.get_dummies(dataset)
data_tips
```

Out[13]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Sex_F	Sex_I	Sex_M
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9	1	0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10	0	0	1
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11	1	0	0

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Sex_F	Sex_I	Sex_M
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10	0	0	1
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9	0	0	1
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10	1	0	0
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12	0	0	1

4177 rows × 11 columns

In [16]:

```
one_encde=OneHotEncoder(sparse=False)
encoded_arr=one_encde.fit_transform(dataset[['Length','Height','Sex','Diameter']])
encoded_arr
```

Out[16]:

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

Split the data into dependent and independent variables

In [6]:

```
x=dataset.iloc[:,1:4]
y=dataset.iloc[:,4]
x
y
```

Out[6]:

```
0       0.5140
1       0.2255
2       0.6770
3       0.5160
4       0.2050
         ...
4172    0.8870
```

```
4173    0.9660
4174    1.1760
4175    1.0945
4176    1.9485
Name: Whole weight, Length: 4177, dtype: float64
```

Scale the independent variables

In [9]:

```
independent=dataset.iloc[1:,1:7].values
```

In [10]:

```
independent
```

Out[10]:

```
array([[0.35  , 0.265 , 0.09  , 0.2255, 0.0995, 0.0485],
       [0.53  , 0.42  , 0.135 , 0.677 , 0.2565, 0.1415],
       [0.44  , 0.365 , 0.125 , 0.516 , 0.2155, 0.114 ],
       ...,
       [0.6   , 0.475 , 0.205 , 1.176 , 0.5255, 0.2875],
       [0.625 , 0.485 , 0.15  , 1.0945, 0.531 , 0.261 ],
       [0.71  , 0.555 , 0.195 , 1.9485, 0.9455, 0.3765]])
```

In [11]:

```
##Dependent variables
```

In [12]:

```
dependent=dataset.iloc[1:,9:].values
dependent
```

Out[12]:

```
array([], shape=(4176, 0), dtype=float64)
```

Split the data into training and testing

In [68]:

```
from sklearn.model_selection import train_test_split
```

In [69]:

```
x_train,x_test,y_train,y_test=train_test_split(independent,dependent,test_s
ize=0.2,random_state=5)
```

In [70]:

```
x_train
```

Out[70]:

```
array([[0.565 , 0.435 , 0.15  , 0.99  , 0.5795, 0.1825],
       [0.48  , 0.37  , 0.125 , 0.5435, 0.244 , 0.101 ],
       [0.44  , 0.35  , 0.12  , 0.375 , 0.1425, 0.0965],
       ...,
       [0.555 , 0.43  , 0.125 , 0.7005, 0.3395, 0.1355],
       [0.51  , 0.395 , 0.145 , 0.6185, 0.216 , 0.1385],
       [0.595 , 0.47  , 0.155 , 1.2015, 0.492 , 0.3865]])
```

In [71]:

```
x_test
```

Out[71]:

```
array([[0.455 , 0.365 , 0.11  , 0.385 , 0.166 , 0.046 ],
```

```
       [0.47  , 0.37  , 0.18  , 0.51  , 0.1915, 0.1285],
       [0.72  , 0.575 , 0.17  , 1.9335, 0.913 , 0.389 ],
       ...,
       [0.275 , 0.215 , 0.075 , 0.1155, 0.0485, 0.029 ],
       [0.39  , 0.3   , 0.09  , 0.252 , 0.1065, 0.053 ],
       [0.585 , 0.46  , 0.165 , 1.1135, 0.5825, 0.2345]])
```

In [72]:

```
y_train
```

Out[72]:

```
array([], shape=(3340, 0), dtype=float64)
```

In [73]:

```
y_test
```

Out[73]:

```
array([], shape=(836, 0), dtype=float64)
```

Build the Model

In [74]:

```
from sklearn import datasets
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
```

In [75]:

```
iris=datasets.load_iris()
```

In [76]:

```
print(iris.feature_names)
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
```

In [77]:

```
print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

In [78]:

```
iris.data
```

Out[78]:

```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
```

```
[5.7, 3.8, 1.7, 0.3],
[5.1, 3.8, 1.5, 0.3],
[5.4, 3.4, 1.7, 0.2],
[5.1, 3.7, 1.5, 0.4],
[4.6, 3.6, 1. , 0.2],
[5.1, 3.3, 1.7, 0.5],
[4.8, 3.4, 1.9, 0.2],
[5. , 3. , 1.6, 0.2],
[5. , 3.4, 1.6, 0.4],
[5.2, 3.5, 1.5, 0.2],
[5.2, 3.4, 1.4, 0.2],
[4.7, 3.2, 1.6, 0.2],
[4.8, 3.1, 1.6, 0.2],
[5.4, 3.4, 1.5, 0.4],
[5.2, 4.1, 1.5, 0.1],
[5.5, 4.2, 1.4, 0.2],
[4.9, 3.1, 1.5, 0.2],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.6, 1.4, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
```

```
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
[6.4, 2.8, 5.6, 2.1],
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
[7.9, 3.8, 6.4, 2. ],
```

```
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

In [79]:

```
iris.target
```

Out[79]:

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

In [80]:

```
x=iris.data
y=iris.target
```

In [81]:

```
x.shape
```

Out[81]:

```
(150, 4)
```

In [82]:

```
y.shape
```

Out[82]:

```
(150,)
```

In [83]:

```
clf=RandomForestClassifier()
```

In [84]:

```
clf.fit(x,y)
```

Out[84]:

```
RandomForestClassifier()
```

In [85]:

```
print(clf.feature_importances_)
```

```
[0.09934163 0.02933361 0.41455791 0.45676685]
```

In [86]:

```
x[0]
```

Out[86]:

```
array([5.1, 3.5, 1.4, 0.2])
```

In [87]:

```
print(clf.predict([[5.1, 3.5, 1.4, 0.2]]))
```

```
[0]
```

In [88]:

```
print(clf.predict(x[[0]]))
```

```
[0]
```

In [89]:

```
print(clf.predict_proba(x[[0]]))
```

```
[[1. 0. 0.]]
```

In [90]:

```
clf.fit(iris.data,iris.target_names[iris.target])
```

Out[90]:

```
RandomForestClassifier()
```

Train & Test the model

In [93]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

In [94]:

```
x_train.shape,y_train.shape
```

Out[94]:

```
((120, 4), (120,))
```

In [95]:

```
x_test.shape,y_test.shape
```

Out[95]:

```
((30, 4), (30,))
```

In [96]:

```
clf.fit(x_train,y_train)
```

Out[96]:

```
RandomForestClassifier()
```

In [97]:

```
print(clf.predict([[5.1, 3.5, 1.4, 0.2]]))
```

```
[0]
```

In [98]:

```
print(clf.predict_proba([[5.1, 3.5, 1.4, 0.2]]))
```

```
[[1. 0. 0.]]
```

In [99]:

```
print(clf.predict(x_test))
```

```
[0 2 0 2 2 0 2 0 2 1 1 2 1 1 2 2 1 1 0 2 2 0 0 1 2 1 0 0 2 0]
```

In [100]:

```
print(y_test)
```

```
[0 2 0 2 2 0 2 0 2 1 1 2 1 1 2 2 1 1 0 2 2 0 0 1 2 1 0 0 2 0]
```

In [101]:

```
print(clf.score(x_test,y_test))
```

```
1.0
```