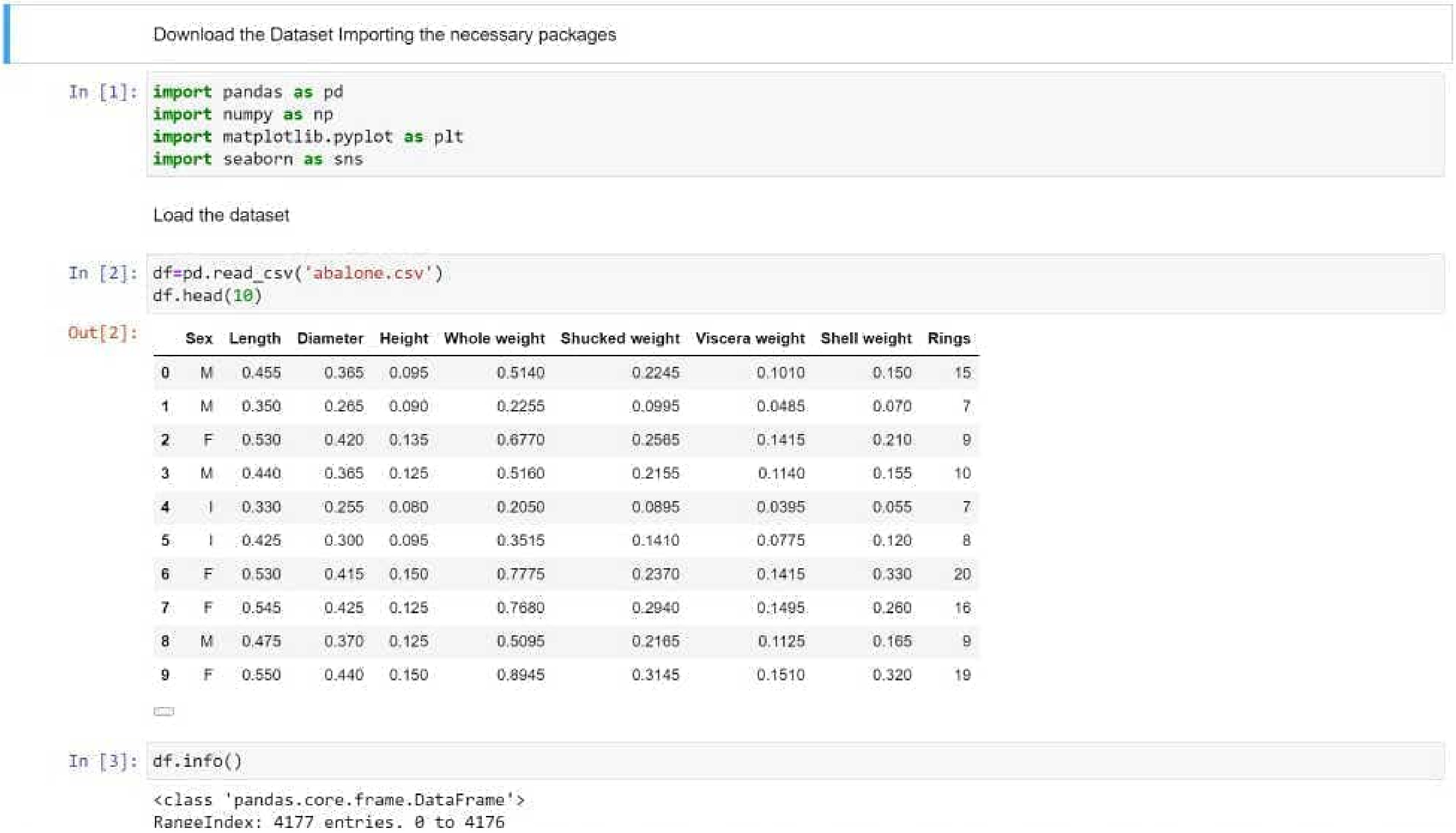


Download the Dataset Importing the necessary packages

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
```

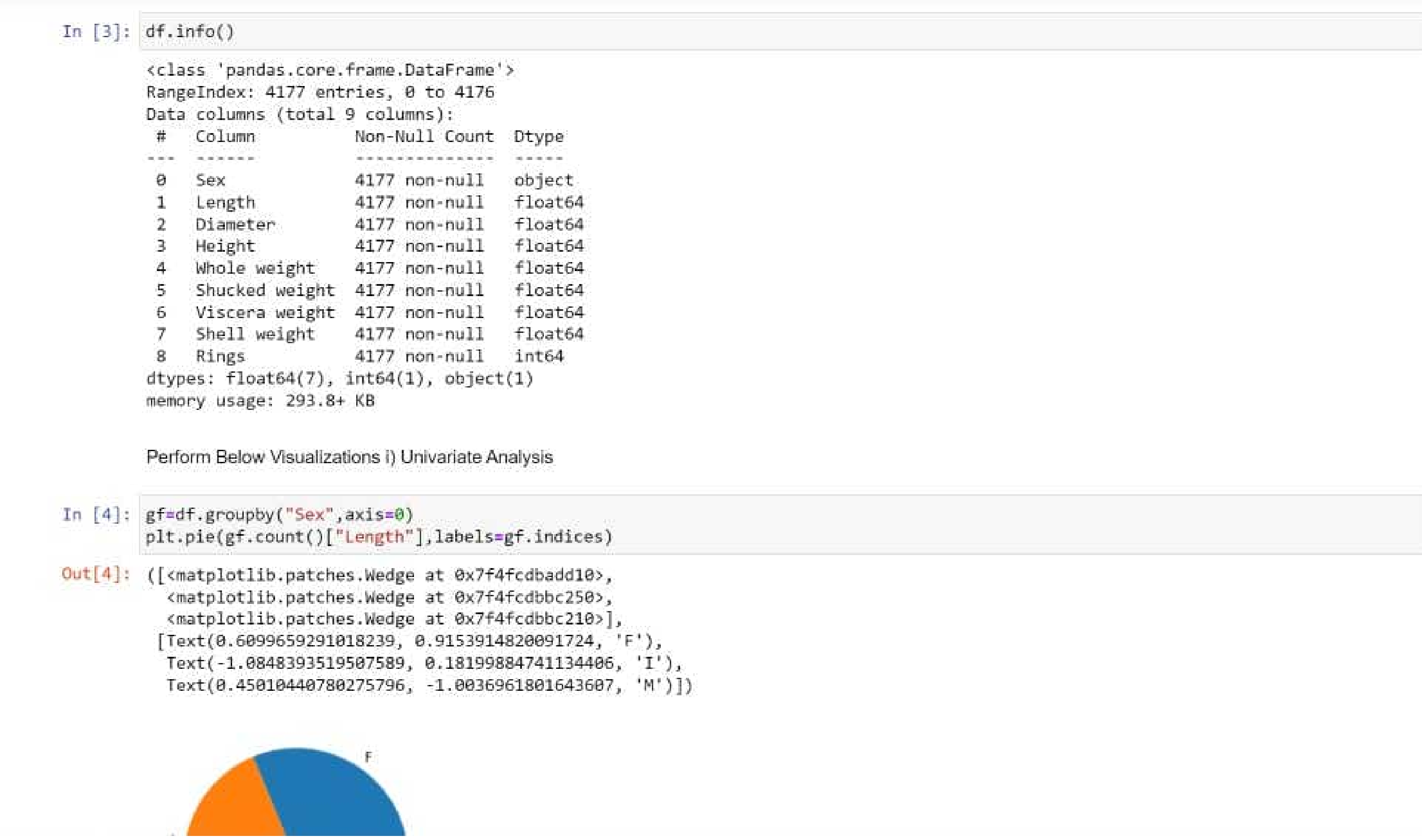
Load the dataset

```
In [2]: df=pd.read_csv('abalone.csv')
        df.head(10)
```

Out[2]:

	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
5	I	0.425	0.300	0.095	0.3515	0.1410	0.0775	0.120	8
6	F	0.530	0.415	0.150	0.7775	0.2370	0.1415	0.330	20
7	F	0.545	0.425	0.125	0.7680	0.2940	0.1495	0.260	16
8	M	0.475	0.370	0.125	0.5095	0.2165	0.1125	0.165	9
9	F	0.550	0.440	0.150	0.8945	0.3145	0.1510	0.320	19

```
In [3]: df.info()

        <class 'pandas.core.frame.DataFrame'>
        RangeIndex: 4177 entries, 0 to 4176
```



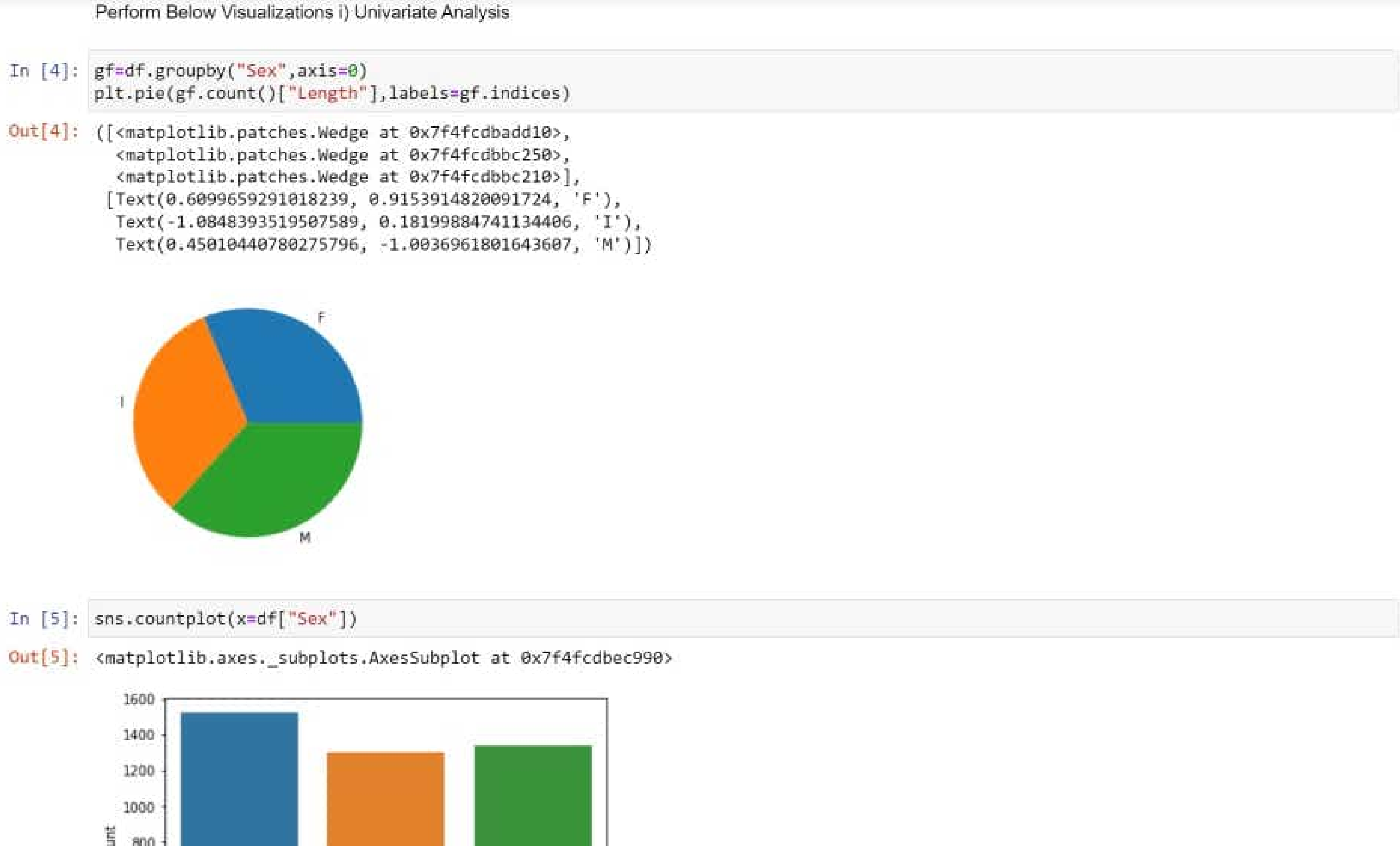
In [3]:
```
df.info()
```

```
<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   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
```

Perform Below Visualizations i) Univariate Analysis

In [4]:
```
gf=df.groupby("Sex",axis=0)
plt.pie(gf.count()["Length"],labels=gf.indices)
```

Out[4]:
```
([<matplotlib.patches.Wedge at 0x7f4fcdbadd10>,
  <matplotlib.patches.Wedge at 0x7f4fcdbbc250>,
  <matplotlib.patches.Wedge at 0x7f4fcdbbc210>],
 [Text(0.6099659291018239, 0.9153914820091724, 'F'),
  Text(-1.0848393519507589, 0.18199884741134406, 'I'),
  Text(0.45010440780275796, -1.0036961801643607, 'M')])
```



Perform Below Visualizations i) Univariate Analysis

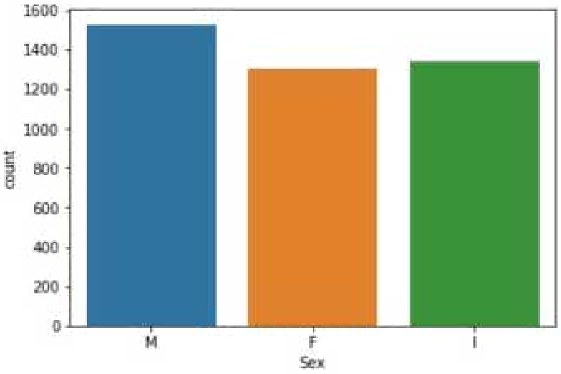
```
In [4]: gf=df.groupby("Sex",axis=0)
        plt.pie(gf.count()["Length"],labels=gf.indices)
```

```
Out[4]: ([<matplotlib.patches.Wedge at 0x7f4fcdbadd10>,
          <matplotlib.patches.Wedge at 0x7f4fcdbbc250>,
          <matplotlib.patches.Wedge at 0x7f4fcdbbc210>],
         [Text(0.6099659291018239, 0.9153914820091724, 'F'),
          Text(-1.0848393519507589, 0.18199884741134406, 'I'),
          Text(0.45010440780275796, -1.0036961801643607, 'M')])
```

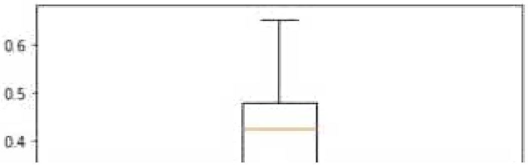
```
In [5]: sns.countplot(x=df["Sex"])
```

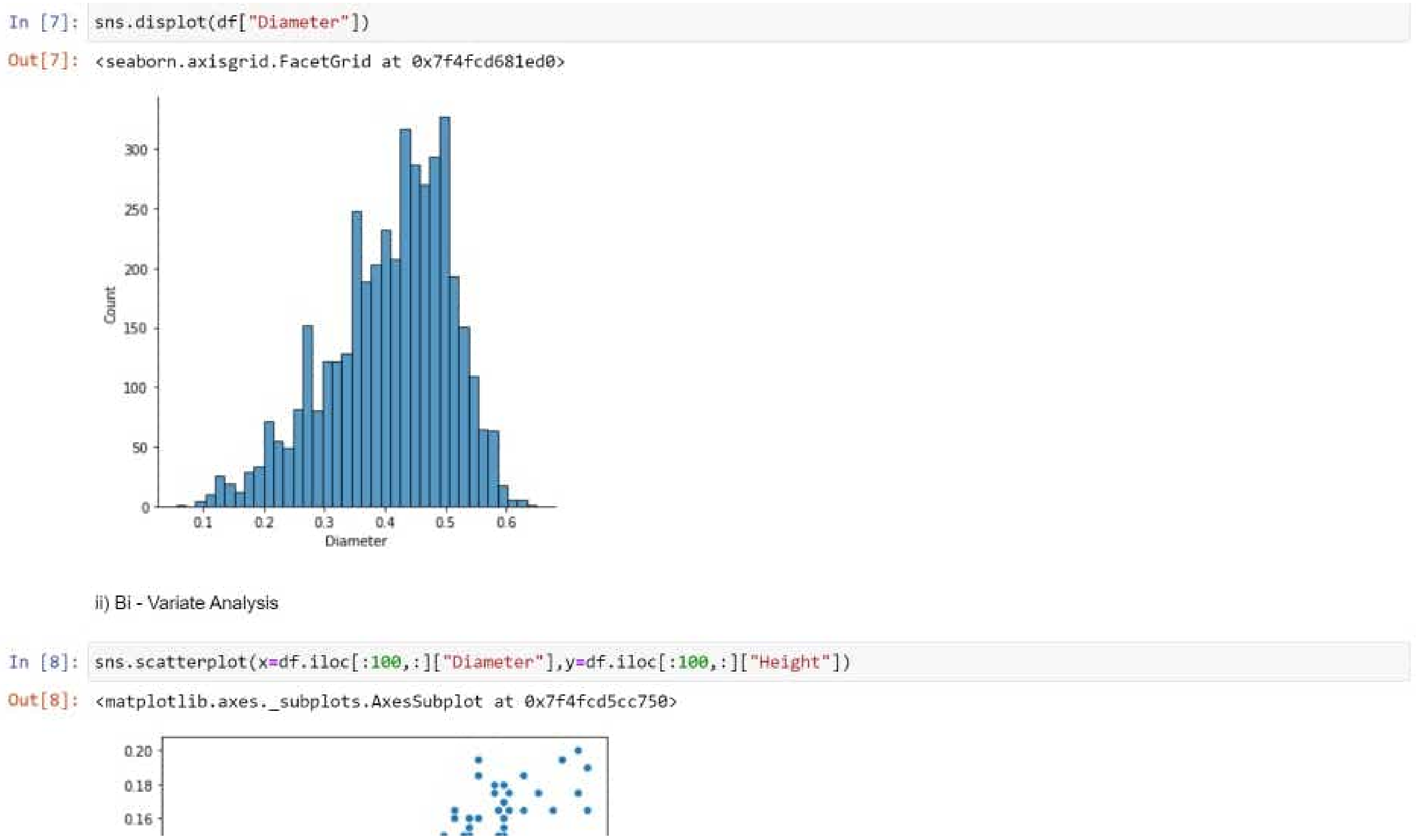
```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4fcdbec990>
```

```
In [5]: sns.countplot(x=df["Sex"])

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4fcdbec990>
```



```
In [6]: plt.boxplot(df["Diameter"])

Out[6]: {'whiskers': [<matplotlib.lines.Line2D at 0x7f4fcd6f5810>,
          <matplotlib.lines.Line2D at 0x7f4fcd67a6d0>],
         'caps': [<matplotlib.lines.Line2D at 0x7f4fcd67ac10>,
          <matplotlib.lines.Line2D at 0x7f4fcd681190>],
         'boxes': [<matplotlib.lines.Line2D at 0x7f4fe3b26a10>],
         'medians': [<matplotlib.lines.Line2D at 0x7f4fcd681710>],
         'fliers': [<matplotlib.lines.Line2D at 0x7f4fcd681c50>],
         'means': []}
```





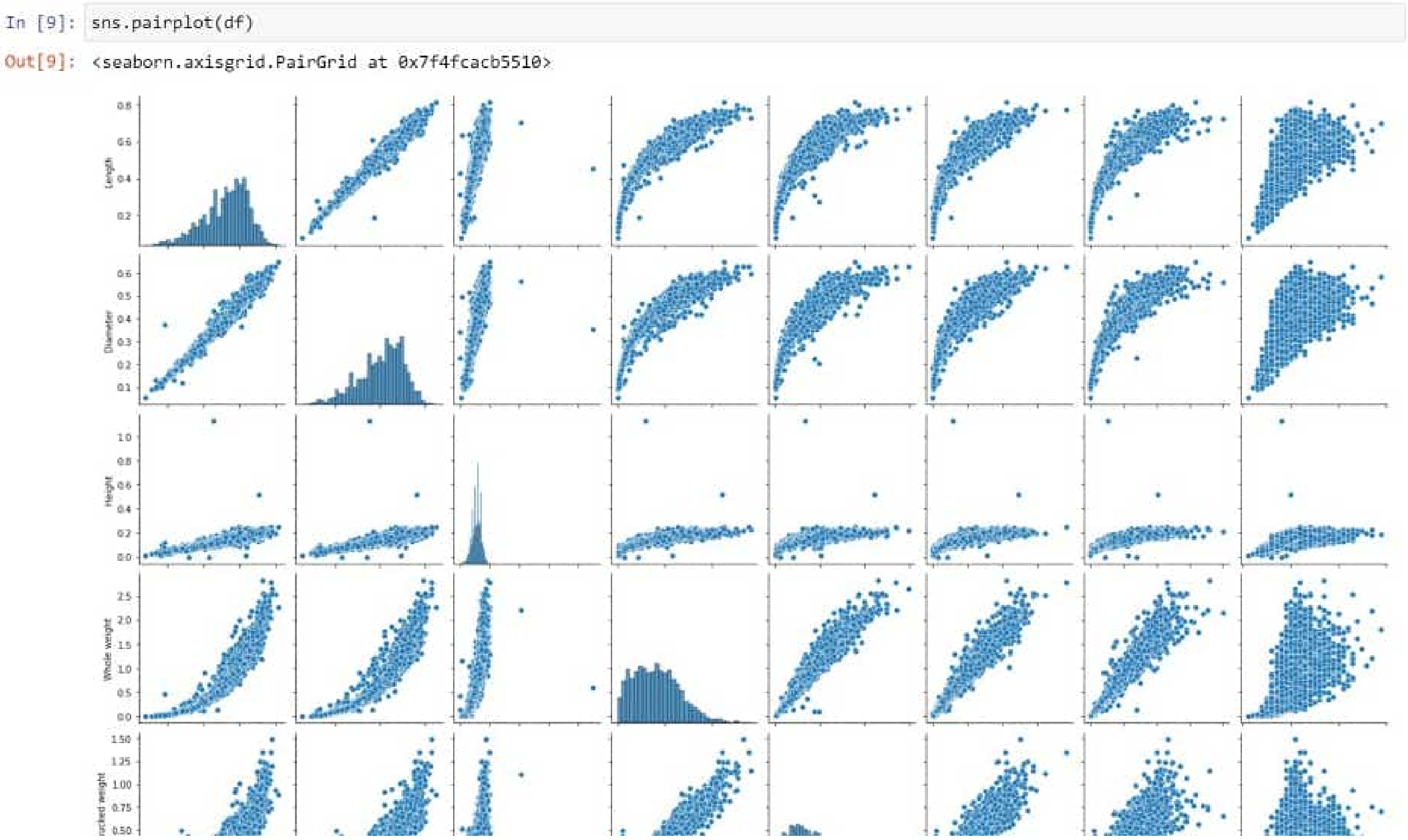
```
In [7]: sns.displot(df["Diameter"])
```

```
Out[7]: <seaborn.axisgrid.FacetGrid at 0x7f4fcd681ed0>
```

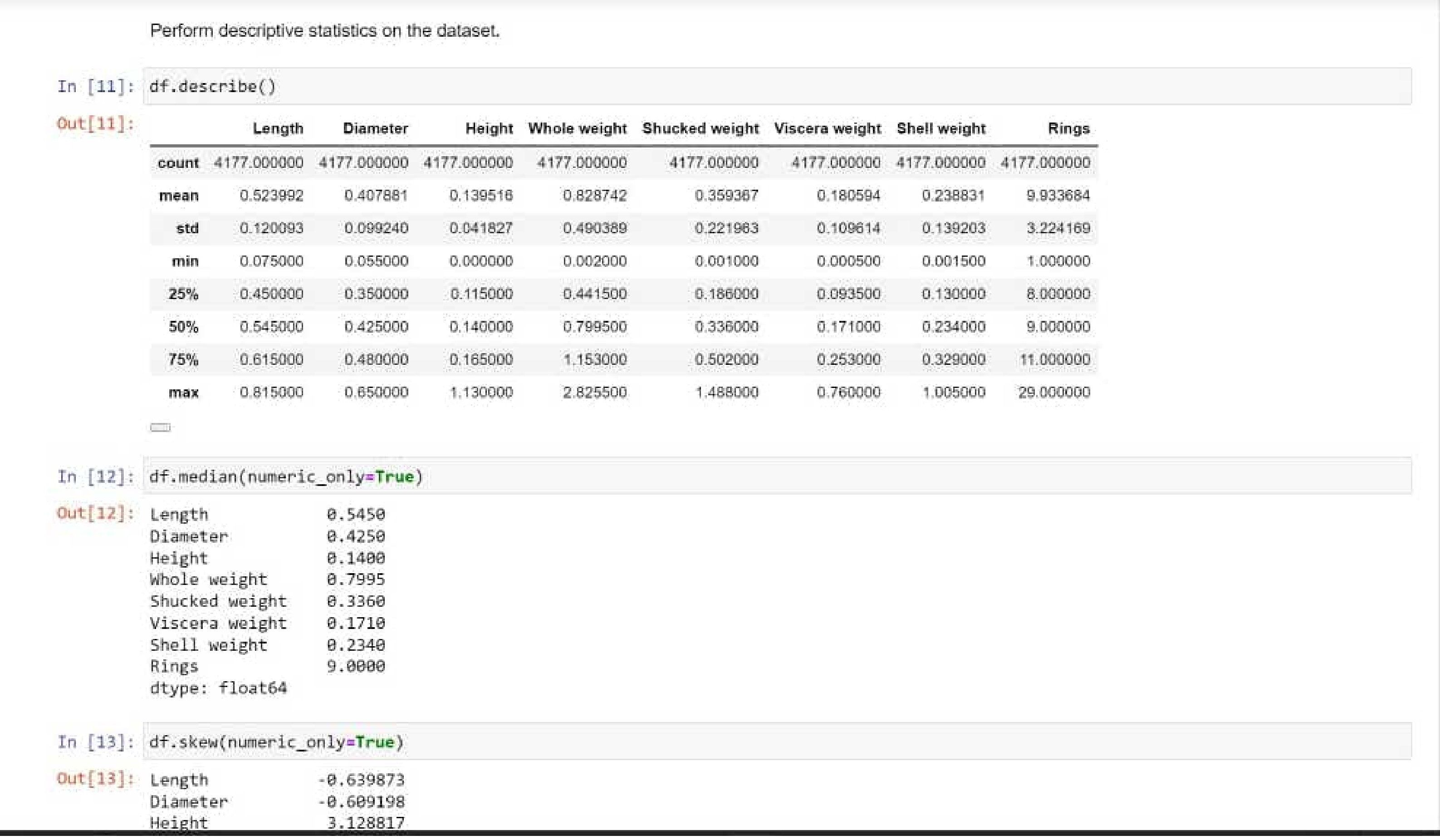
ii) Bi - Variate Analysis

```
In [8]: sns.scatterplot(x=df.iloc[:100,:]["Diameter"],y=df.iloc[:100,:]["Height"])
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4fcd5cc750>
```



```
In [9]: sns.pairplot(df)

Out[9]: <seaborn.axisgrid.PairGrid at 0x7f4fcacb5510>
```



Perform descriptive statistics on the dataset.

In [11]:
```
df.describe()
```

Out[11]:

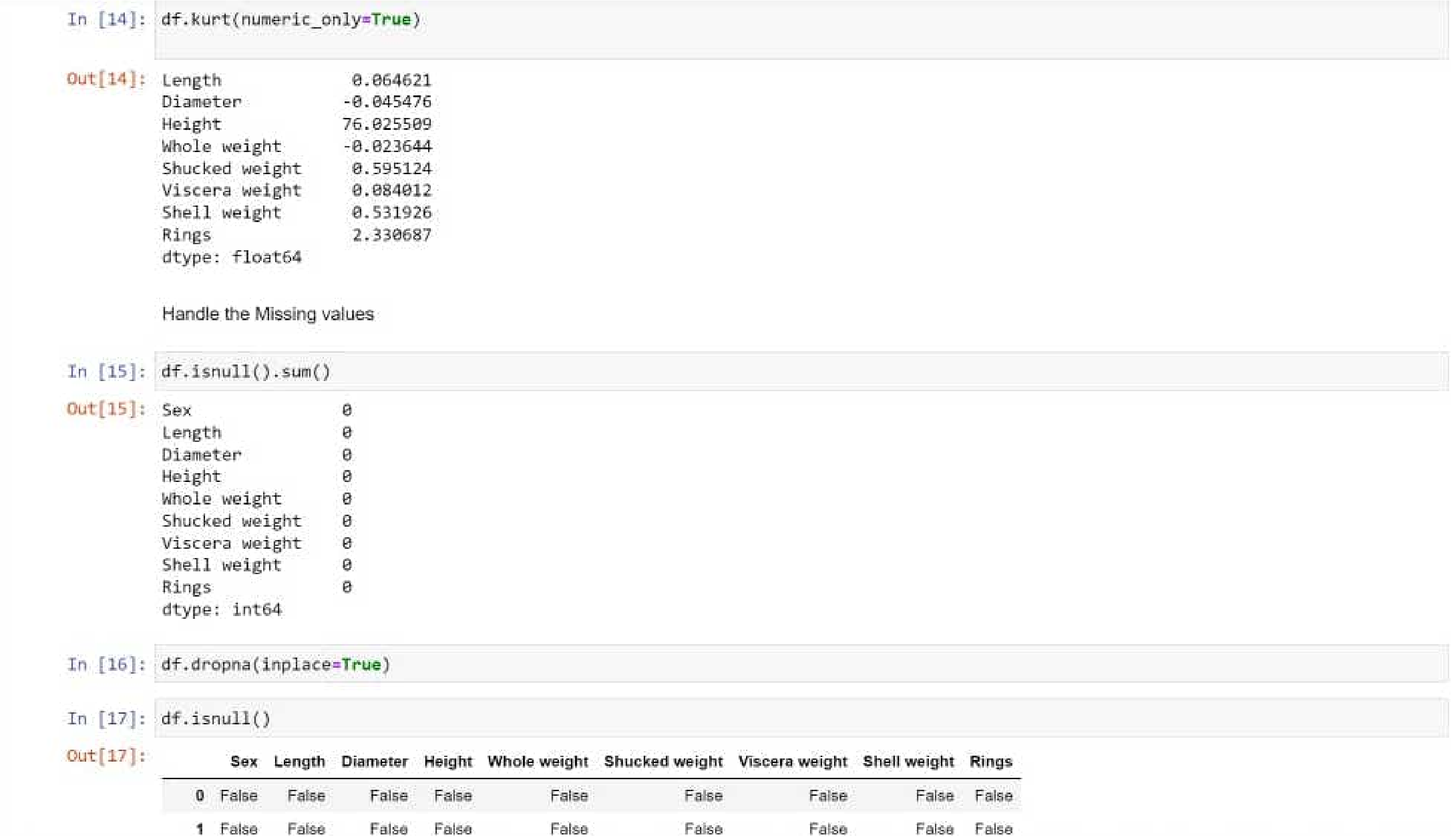
	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	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

In [12]:
```
df.median(numeric_only=True)
```

Out[12]:
```
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 [13]:
```
df.skew(numeric_only=True)
```

Out[13]:
```
Length           -0.639873
Diameter         -0.609198
Height            3.128817
```



```
In [14]: df.kurt(numeric_only=True)
```

```
Out[14]: 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
```

Handle the Missing values

```
In [15]: df.isnull().sum()
```

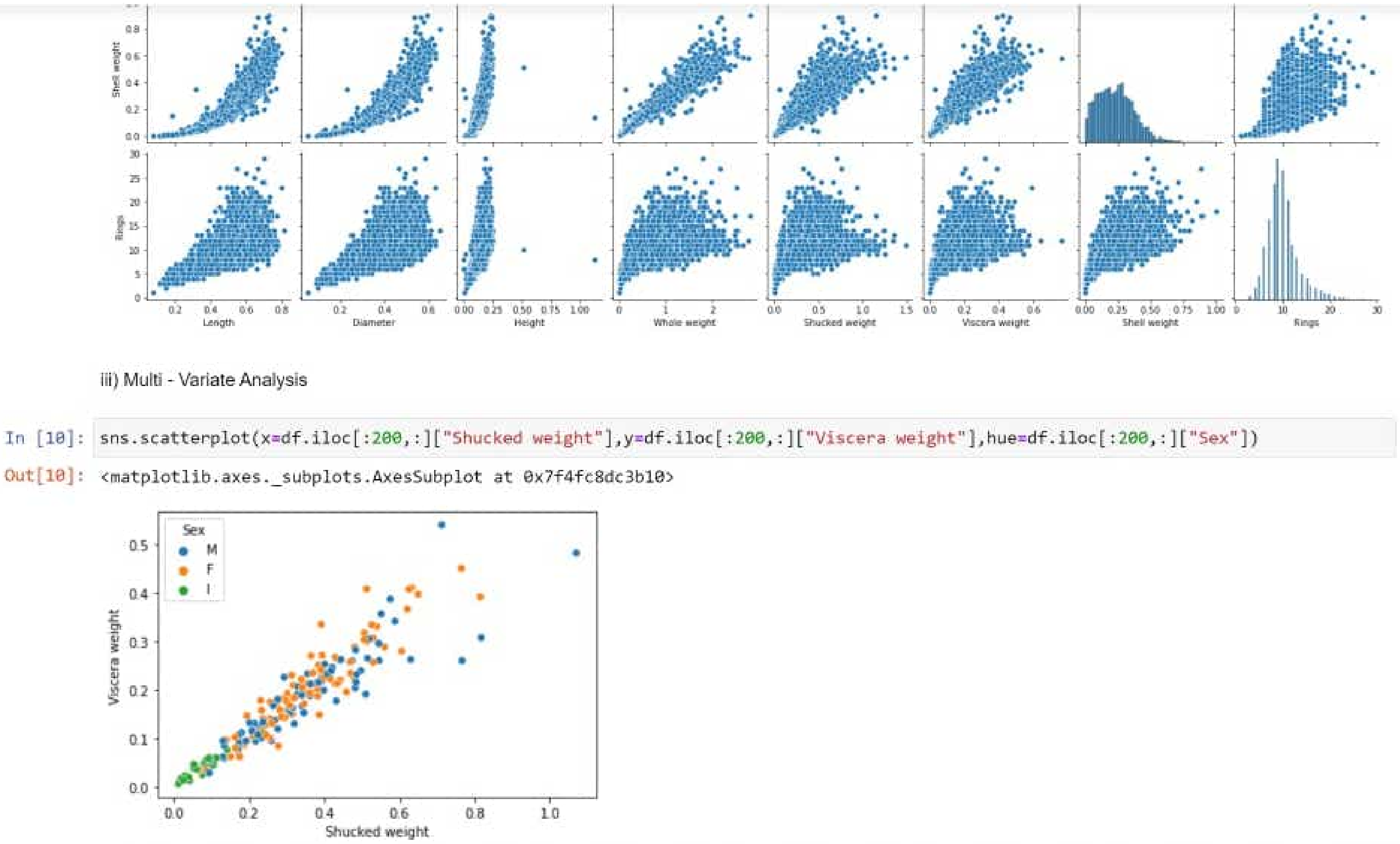
```
Out[15]: Sex               0
         Length            0
         Diameter          0
         Height            0
         Whole weight      0
         Shucked weight    0
         Viscera weight    0
         Shell weight      0
         Rings             0
         dtype: int64
```

```
In [16]: df.dropna(inplace=True)
```

```
In [17]: df.isnull()
```

Out[17]:

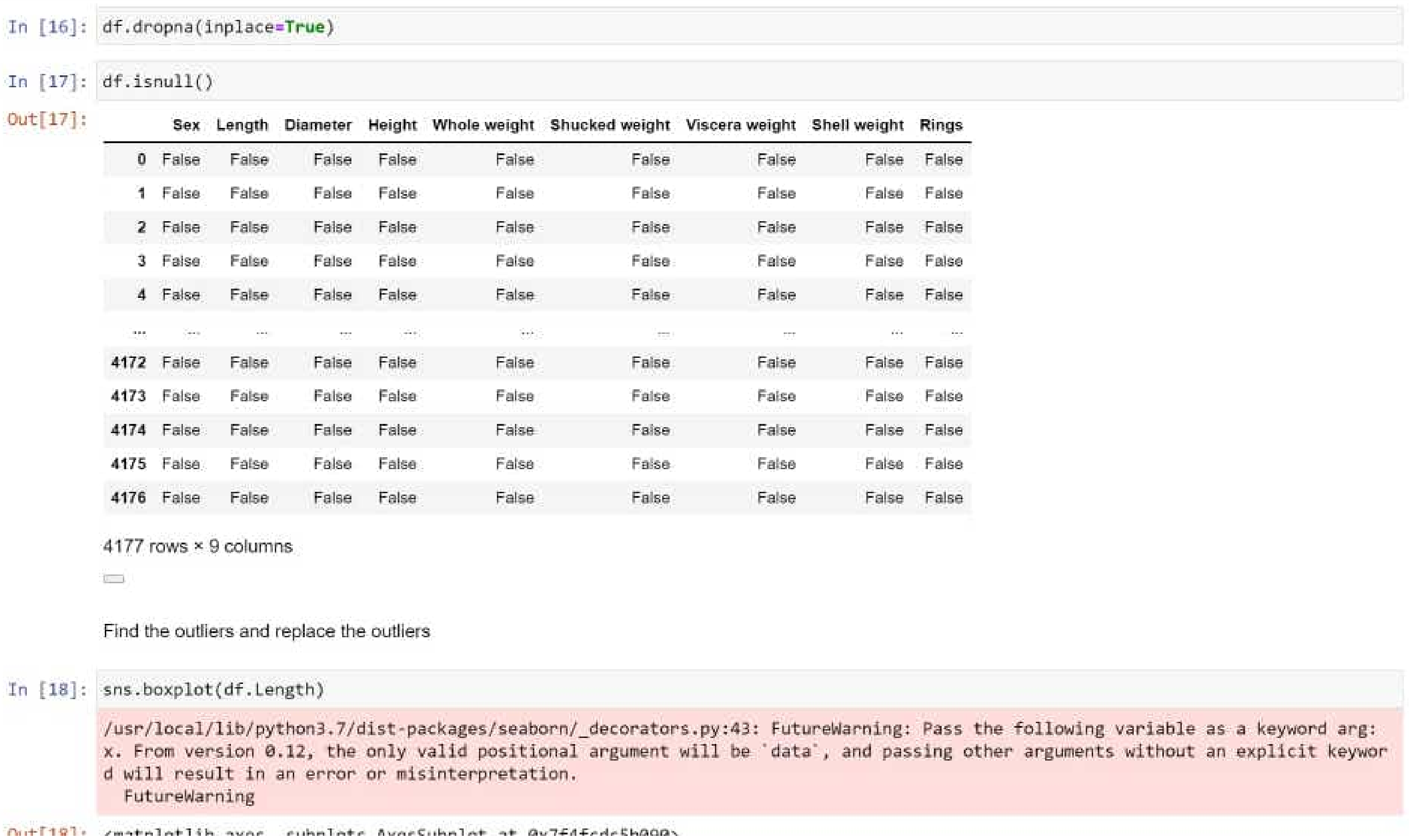
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False



iii) Multi - Variate Analysis

```
In [10]: sns.scatterplot(x=df.iloc[:200,:]["Shucked weight"],y=df.iloc[:200,:]["Viscera weight"],hue=df.iloc[:200,:]["Sex"])
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4fc8dc3b10>
```



```
In [16]: df.dropna(inplace=True)
```

```
In [17]: df.isnull()
```

Out[17]:

	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

Find the outliers and replace the outliers

```
In [18]: sns.boxplot(df.Length)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  FutureWarning
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4fcdc5b090>
```

In [35]:

```
x=df.drop(columns=['Length'],axis=1)
x.head()
```

Out[35]:

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

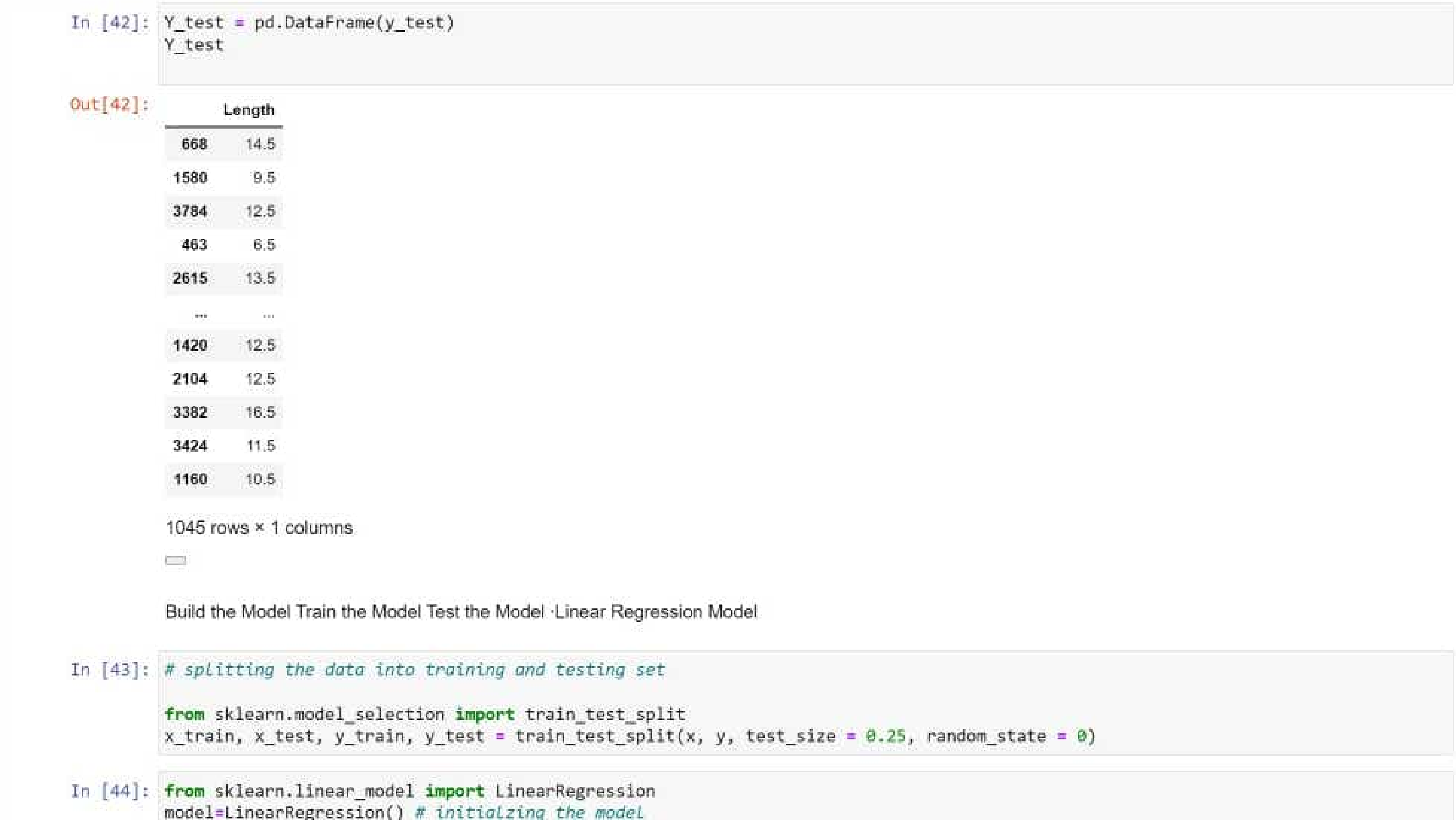
Scale the independent variables

In [36]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 20, test_size=0.4)

from sklearn.preprocessing import scale

x_scaled=pd.DataFrame(scale(x),columns=x.columns)
x_scaled.head()
```

Out[36]:

	Sex	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	1.151980	-0.432149	-1.064424	-0.641898	-0.607685	-0.726212	-0.638217	1.571544
1	1.151980	-1.439929	-1.183978	-1.230277	-1.170910	-1.205221	-1.212987	-0.910013
2	-1.280690	0.122130	-0.107991	-0.309469	-0.463500	-0.356690	-0.207139	-0.289624
3	1.151980	-0.432149	-0.347099	-0.637819	-0.648238	-0.607600	-0.602294	0.020571
4	-0.064355	-1.540707	-1.423087	-1.272086	-1.215968	-1.287337	-1.320757	-0.910013



```
In [42]: Y_test = pd.DataFrame(y_test)
         Y_test
```

Out[42]:

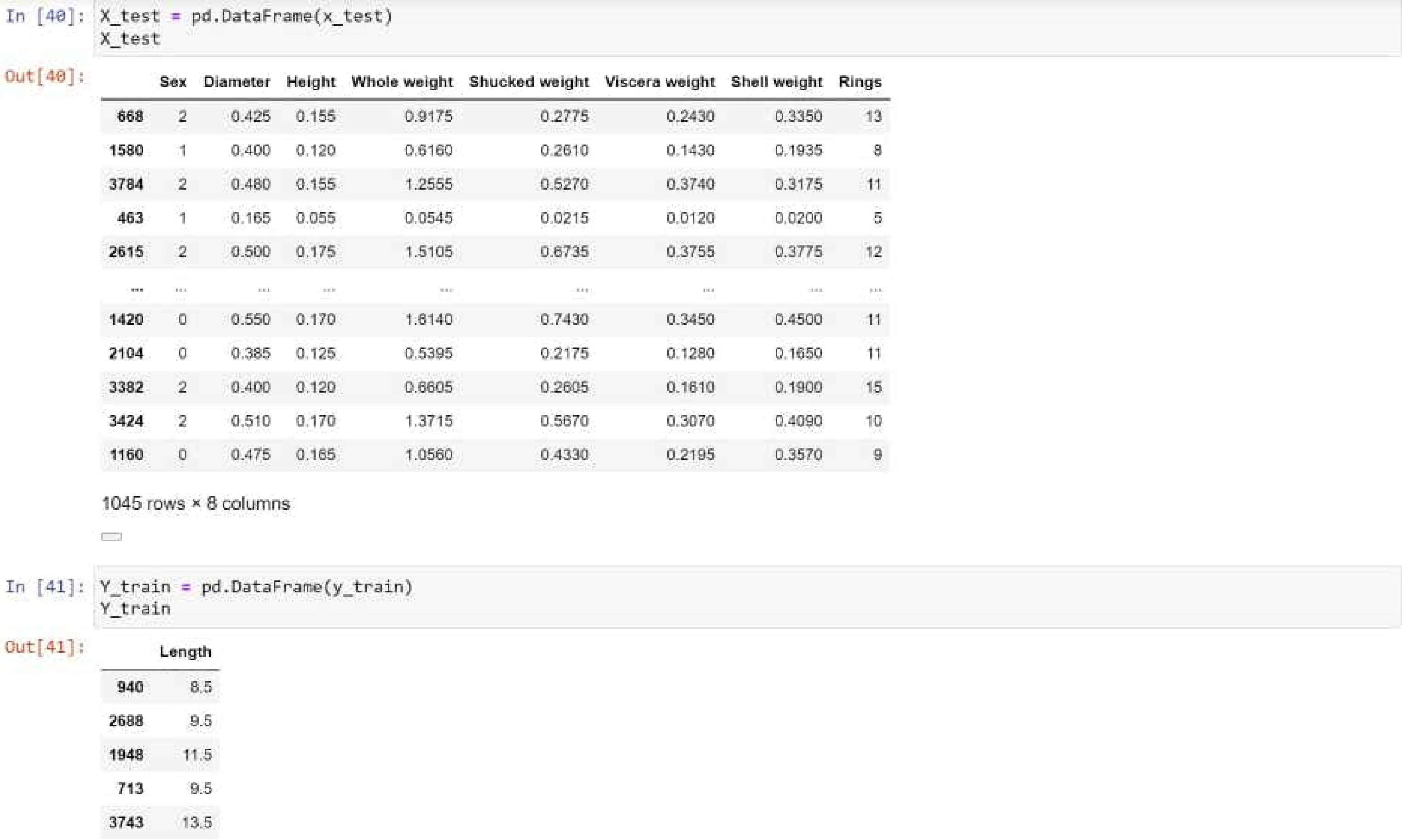
	Length
668	14.5
1580	9.5
3784	12.5
463	6.5
2615	13.5
...	...
1420	12.5
2104	12.5
3382	16.5
3424	11.5
1160	10.5

1045 rows × 1 columns

Build the Model Train the Model Test the Model ·Linear Regression Model

```
In [43]: # splitting the data into training and testing set

         from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
```

```
In [44]: from sklearn.linear_model import LinearRegression
         model=LinearRegression() # initialzing the model
```



In [40]:
```
X_test = pd.DataFrame(x_test)
X_test
```

Out[40]:

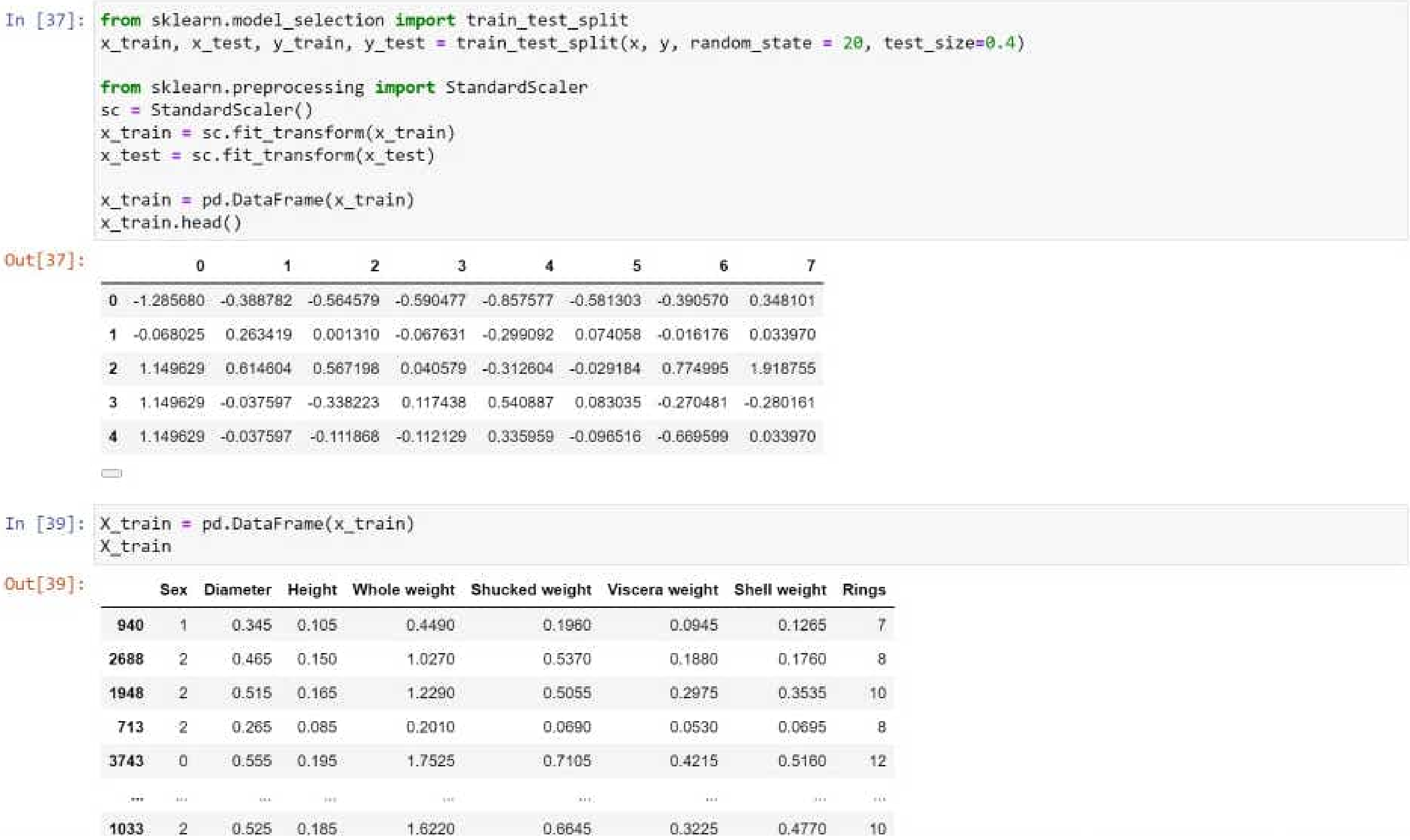
	Sex	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
668	2	0.425	0.155	0.9175	0.2775	0.2430	0.3350	13
1580	1	0.400	0.120	0.6160	0.2610	0.1430	0.1935	8
3784	2	0.480	0.155	1.2555	0.5270	0.3740	0.3175	11
463	1	0.165	0.055	0.0545	0.0215	0.0120	0.0200	5
2615	2	0.500	0.175	1.5105	0.6735	0.3755	0.3775	12
...	...	...	...	...	...	...	...	...
1420	0	0.550	0.170	1.6140	0.7430	0.3450	0.4500	11
2104	0	0.385	0.125	0.5395	0.2175	0.1280	0.1650	11
3382	2	0.400	0.120	0.6605	0.2605	0.1610	0.1900	15
3424	2	0.510	0.170	1.3715	0.5670	0.3070	0.4090	10
1160	0	0.475	0.165	1.0560	0.4330	0.2195	0.3570	9

1045 rows × 8 columns

In [41]:
```
Y_train = pd.DataFrame(y_train)
Y_train
```

Out[41]:

	Length
940	8.5
2688	9.5
1948	11.5
713	9.5
3743	13.5



In [37]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 20, test_size=0.4)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)

x_train = pd.DataFrame(x_train)
x_train.head()
```

Out[37]:

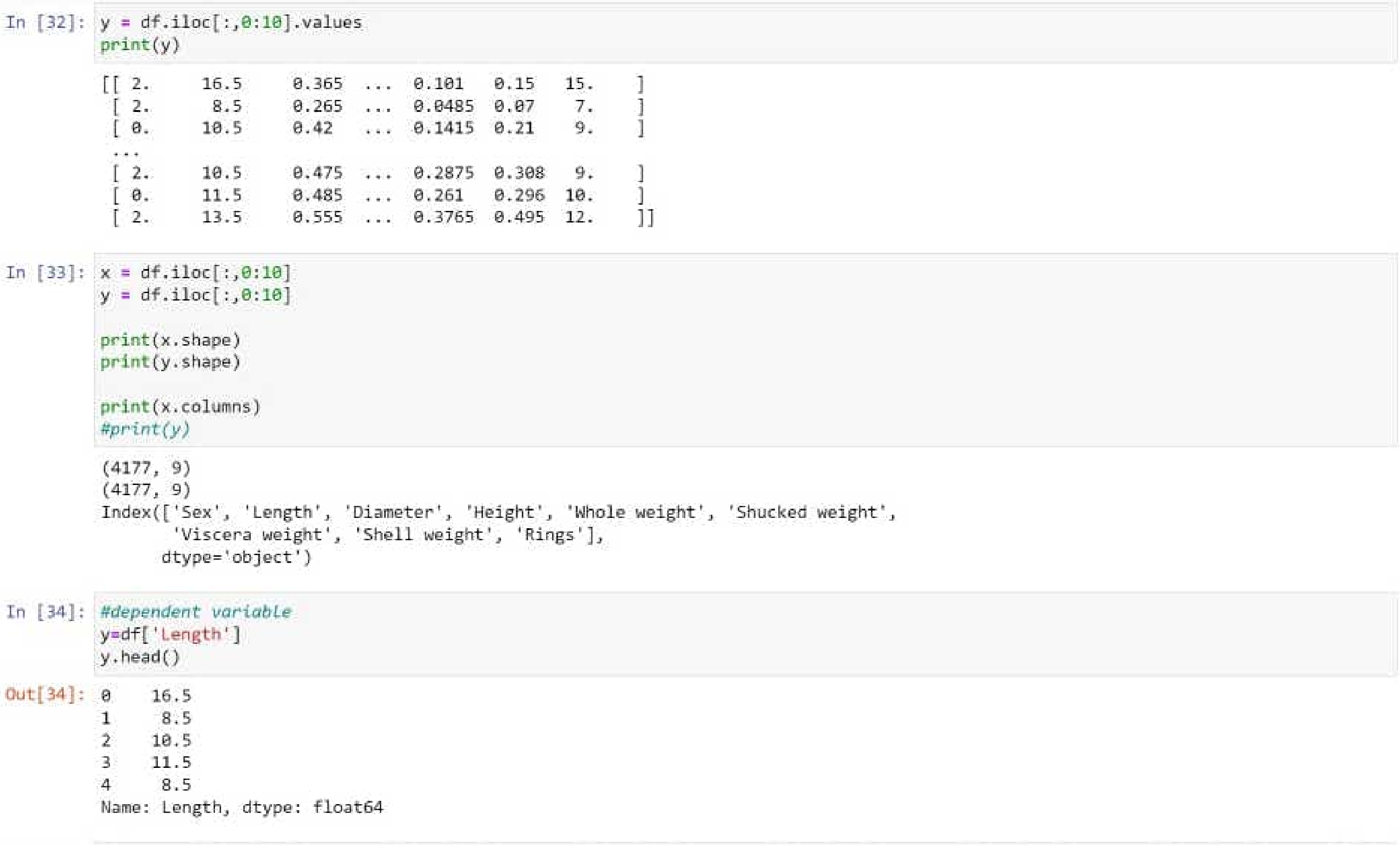
	0	1	2	3	4	5	6	7
0	-1.285680	-0.388782	-0.564579	-0.590477	-0.857577	-0.581303	-0.390570	0.348101
1	-0.068025	0.263419	0.001310	-0.067631	-0.299092	0.074058	-0.016176	0.033970
2	1.149629	0.614604	0.567198	0.040579	-0.312604	-0.029184	0.774995	1.918755
3	1.149629	-0.037597	-0.338223	0.117438	0.540887	0.083035	-0.270481	-0.280161
4	1.149629	-0.037597	-0.111868	-0.112129	0.335959	-0.096516	-0.669599	0.033970

In [39]:

```
X_train = pd.DataFrame(x_train)
X_train
```

Out[39]:

	Sex	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
940	1	0.345	0.105	0.4490	0.1960	0.0945	0.1265	7
2688	2	0.465	0.150	1.0270	0.5370	0.1880	0.1760	8
1948	2	0.515	0.165	1.2290	0.5055	0.2975	0.3535	10
713	2	0.265	0.085	0.2010	0.0690	0.0530	0.0695	8
3743	0	0.555	0.195	1.7525	0.7105	0.4215	0.5160	12
...	...	...	...	...	...	...	...	...
1033	2	0.525	0.185	1.6220	0.6645	0.3225	0.4770	10



In [32]:
```
y = df.iloc[:,0:10].values
print(y)
```

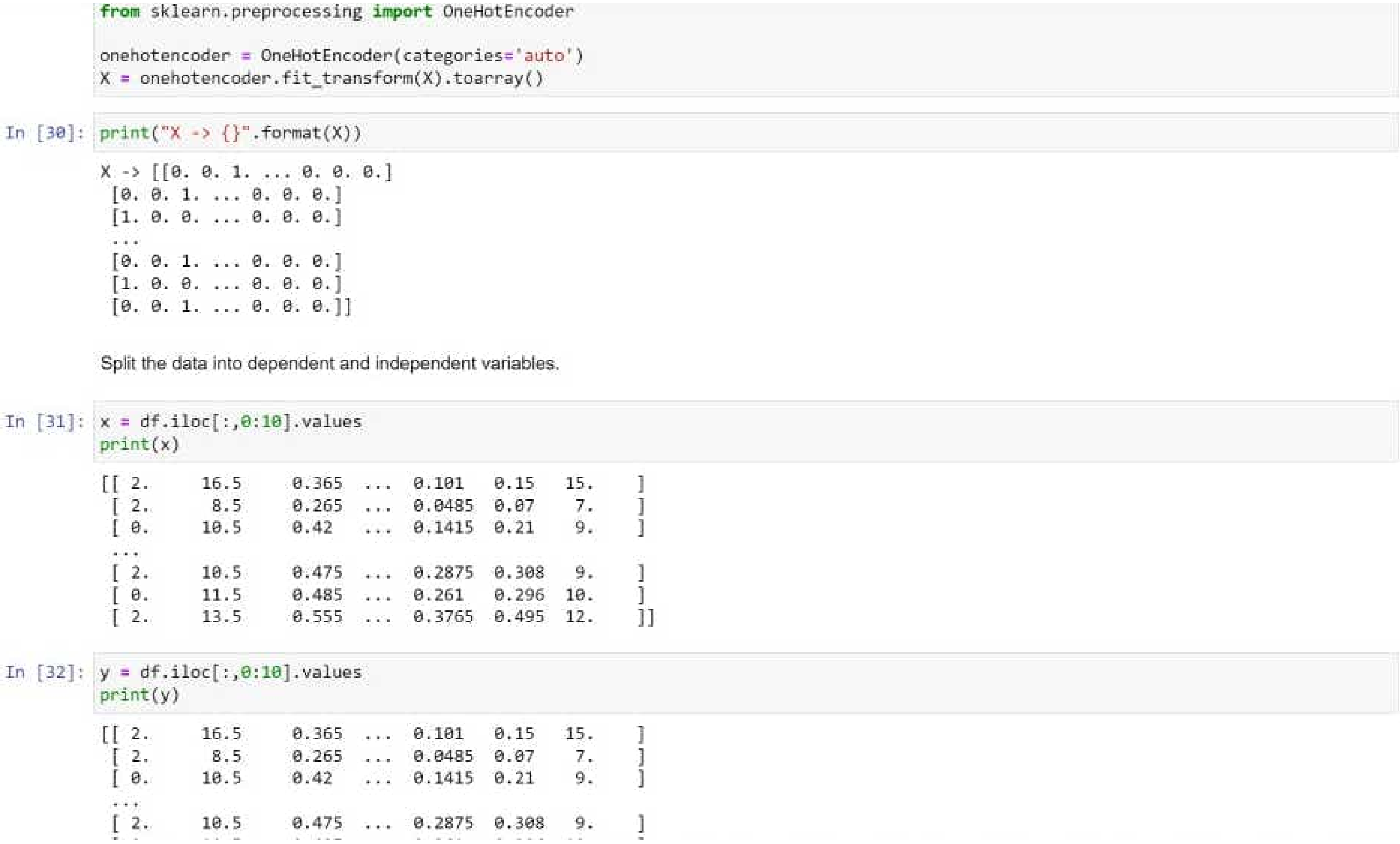
```
[[ 2.     16.5     0.365  ...  0.101   0.15   15.     ]
 [ 2.      8.5     0.265  ...  0.0485  0.07    7.     ]
 [ 0.     10.5     0.42   ...  0.1415  0.21    9.     ]
 ...
 [ 2.     10.5     0.475  ...  0.2875  0.308   9.     ]
 [ 0.     11.5     0.485  ...  0.261   0.296  10.     ]
 [ 2.     13.5     0.555  ...  0.3765  0.495  12.     ]]
```

In [33]:
```
x = df.iloc[:,0:10]
y = df.iloc[:,0:10]

print(x.shape)
print(y.shape)

print(x.columns)
#print(y)
```

```
(4177, 9)
(4177, 9)
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'Rings'],
      dtype='object')
```

In [34]:
```
#dependent variable
y=df['Length']
y.head()
```

Out[34]:
```
0    16.5
1     8.5
2    10.5
3    11.5
4     8.5
Name: Length, dtype: float64
```



```
from sklearn.preprocessing import OneHotEncoder

onehotencoder = OneHotEncoder(categories='auto')
X = onehotencoder.fit_transform(X).toarray()
```

```
In [30]: print("X -> {}".format(X))
```

```
X -> [[0. 0. 1. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 1. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]]
```

Split the data into dependent and independent variables.

```
In [31]: x = df.iloc[:,0:10].values
         print(x)
```

```
[[ 2.     16.5     0.365   ...  0.101   0.15    15.     ]
 [ 2.      8.5     0.265   ...  0.0485  0.07     7.     ]
 [ 0.     10.5     0.42    ...  0.1415  0.21     9.     ]
 ...
 [ 2.     10.5     0.475   ...  0.2875  0.308    9.     ]
 [ 0.     11.5     0.485   ...  0.261   0.296   10.     ]
 [ 2.     13.5     0.555   ...  0.3765  0.495   12.     ]]
```

```
In [32]: y = df.iloc[:,0:10].values
         print(y)
```

```
[[ 2.     16.5     0.365   ...  0.101   0.15    15.     ]
 [ 2.      8.5     0.265   ...  0.0485  0.07     7.     ]
 [ 0.     10.5     0.42    ...  0.1415  0.21     9.     ]
 ...
 [ 2.     10.5     0.475   ...  0.2875  0.308    9.     ]
```

In [26]:
```
newDf.drop(['LengthIndex', 'Sex'], axis = 1, inplace = True)
y = LengthIndex.values

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

X = sc.fit_transform(newDf)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 20, test_size=0.4)
```

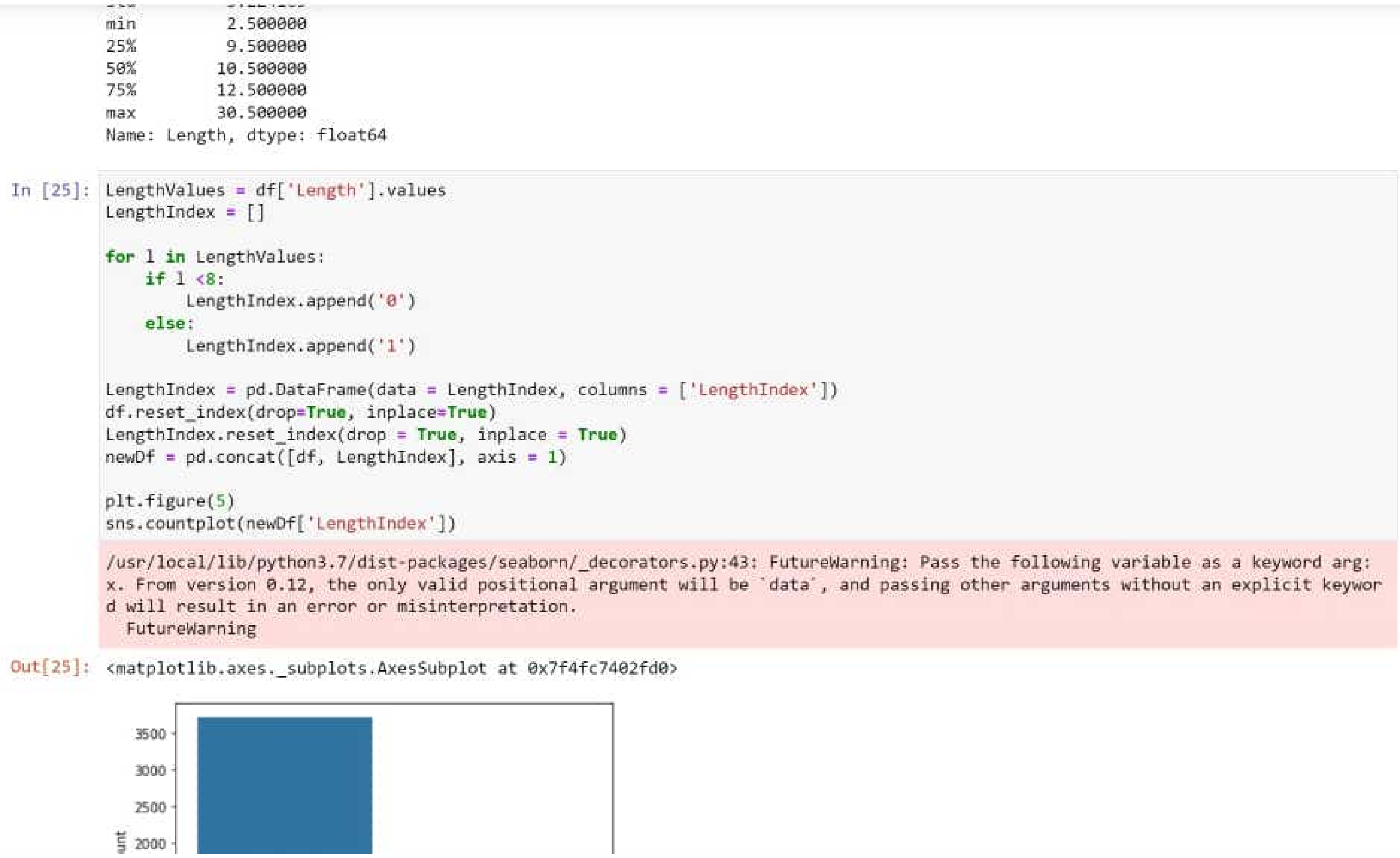
In [27]:
```
X = pd.get_dummies(df)

X.head()
```

Out[27]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	2	16.5	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	2	8.5	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0	10.5	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	2	11.5	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	1	8.5	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [28]:
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values = np.nan, strategy = 'mean',verbose=0)
imputer = imputer.fit(X.iloc[:, 1:3])
X.iloc[:, 1:3] = imputer.transform(X.iloc[:, 1:3])
```

In [29]:
```
from sklearn.preprocessing import LabelEncoder

labelencoder_X = LabelEncoder()
X.iloc[:,0] = labelencoder_X.fit_transform(X.iloc[:,0])
```



```
min         2.500000
25%         9.500000
50%        10.500000
75%        12.500000
max        30.500000
Name: Length, dtype: float64
```

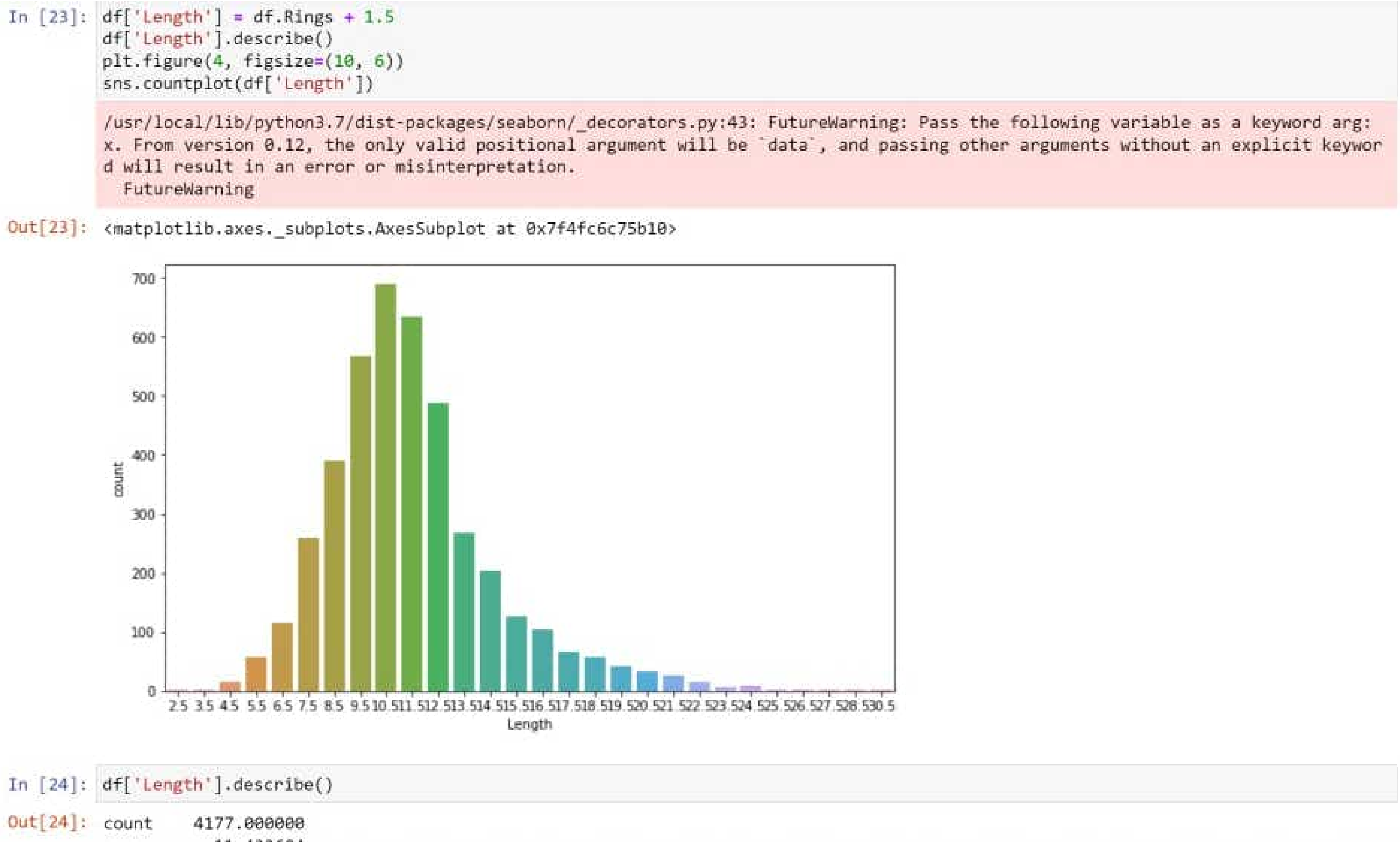
In [25]:

```
LengthValues = df['Length'].values
LengthIndex = []

for l in LengthValues:
    if l <8:
        LengthIndex.append('0')
    else:
        LengthIndex.append('1')

LengthIndex = pd.DataFrame(data = LengthIndex, columns = ['LengthIndex'])
df.reset_index(drop=True, inplace=True)
LengthIndex.reset_index(drop = True, inplace = True)
newDf = pd.concat([df, LengthIndex], axis = 1)

plt.figure(5)
sns.countplot(newDf['LengthIndex'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  FutureWarning
```

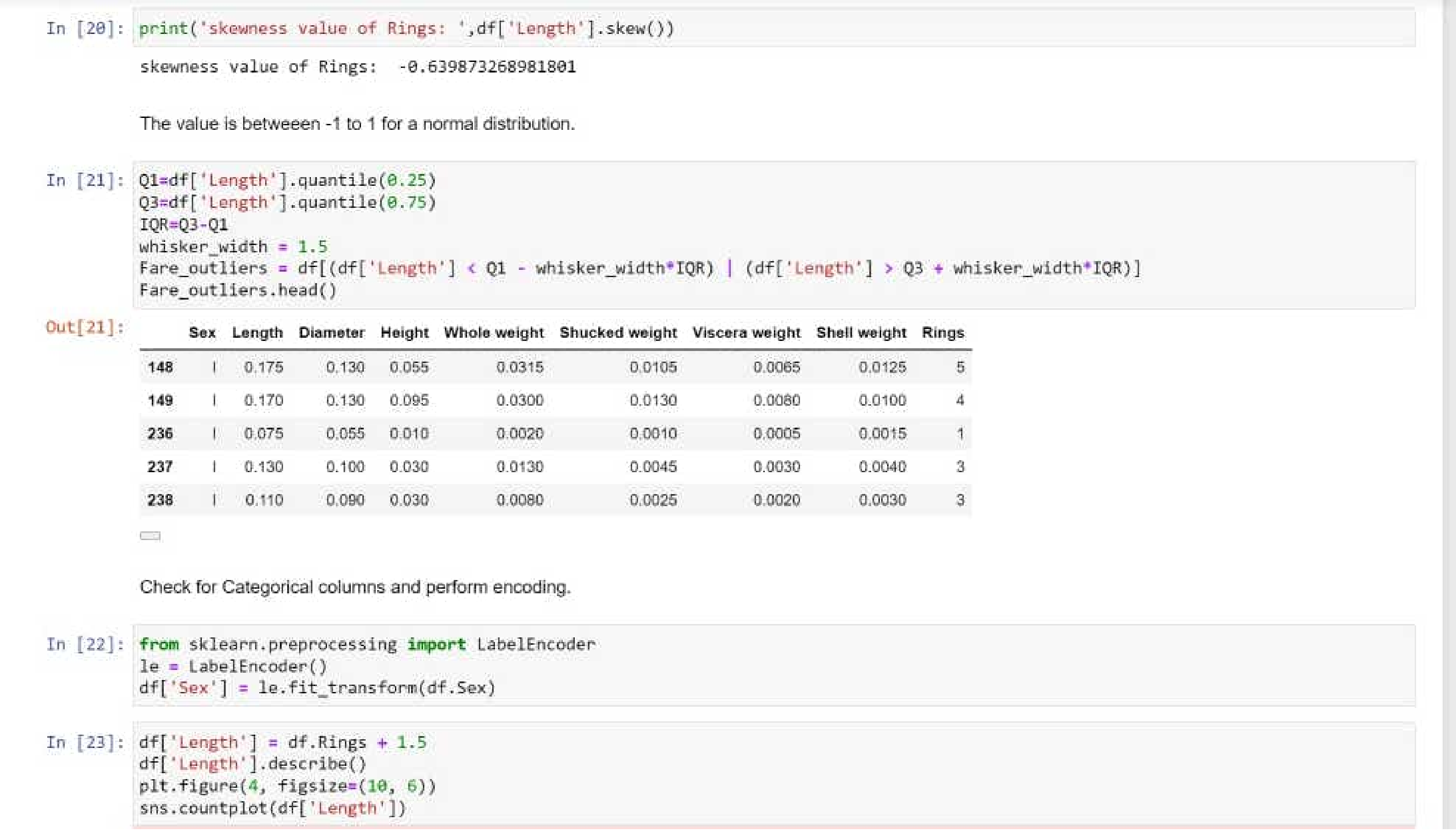
Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4fc7402fd0>
```



```
In [23]: df['Length'] = df.Rings + 1.5
         df['Length'].describe()
         plt.figure(4, figsize=(10, 6))
         sns.countplot(df['Length'])

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  FutureWarning

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4fc6c75b10>
```

```
In [24]: df['Length'].describe()

Out[24]: count    4177.000000
```



In [20]:
```
print('skewness value of Rings: ',df['Length'].skew())
```

```
skewness value of Rings:  -0.639873268981801
```

The value is betweeen -1 to 1 for a normal distribution.

In [21]:
```
Q1=df['Length'].quantile(0.25)
Q3=df['Length'].quantile(0.75)
IQR=Q3-Q1
whisker_width = 1.5
Fare_outliers = df[(df['Length'] < Q1 - whisker_width*IQR) | (df['Length'] > Q3 + whisker_width*IQR)]
Fare_outliers.head()
```

Out[21]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
148	I	0.175	0.130	0.055	0.0315	0.0105	0.0065	0.0125	5
149	I	0.170	0.130	0.095	0.0300	0.0130	0.0080	0.0100	4
236	I	0.075	0.055	0.010	0.0020	0.0010	0.0005	0.0015	1
237	I	0.130	0.100	0.030	0.0130	0.0045	0.0030	0.0040	3
238	I	0.110	0.090	0.030	0.0080	0.0025	0.0020	0.0030	3

Check for Categorical columns and perform encoding.

In [22]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Sex'] = le.fit_transform(df.Sex)
```

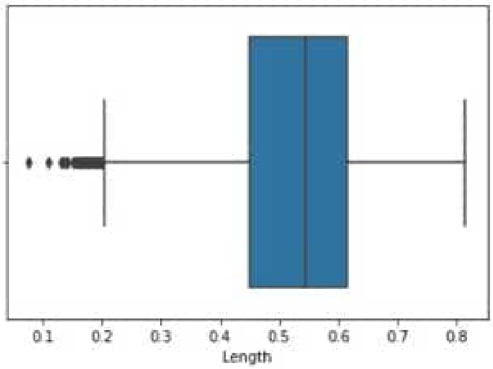
In [23]:
```
df['Length'] = df.Rings + 1.5
df['Length'].describe()
plt.figure(4, figsize=(10, 6))
sns.countplot(df['Length'])
```

Find the outliers and replace the outliers

```
In [18]: sns.boxplot(df.Length)
```

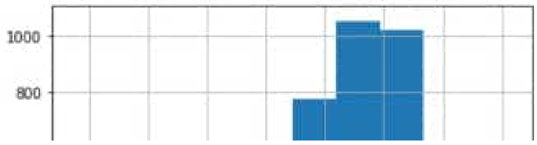
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  FutureWarning
```

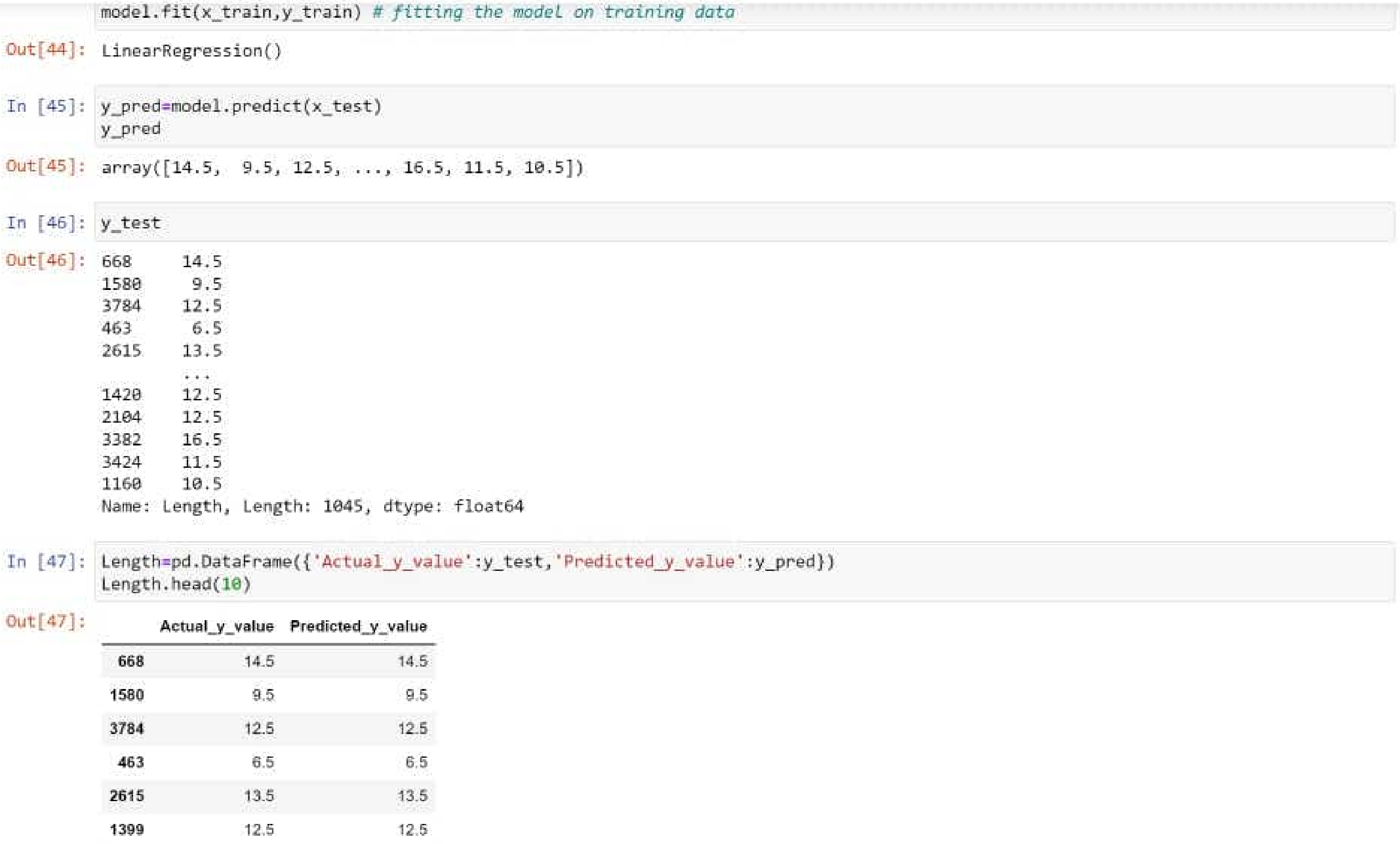
```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4fcdc5b090>
```



```
In [19]: df['Length'].hist()
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x7f4fc8e53090>
```





```
model.fit(x_train,y_train) # fitting the model on training data
```

Out[44]:
```
LinearRegression()
```

In [45]:
```
y_pred=model.predict(x_test)
y_pred
```

Out[45]:
```
array([14.5,  9.5, 12.5, ..., 16.5, 11.5, 10.5])
```

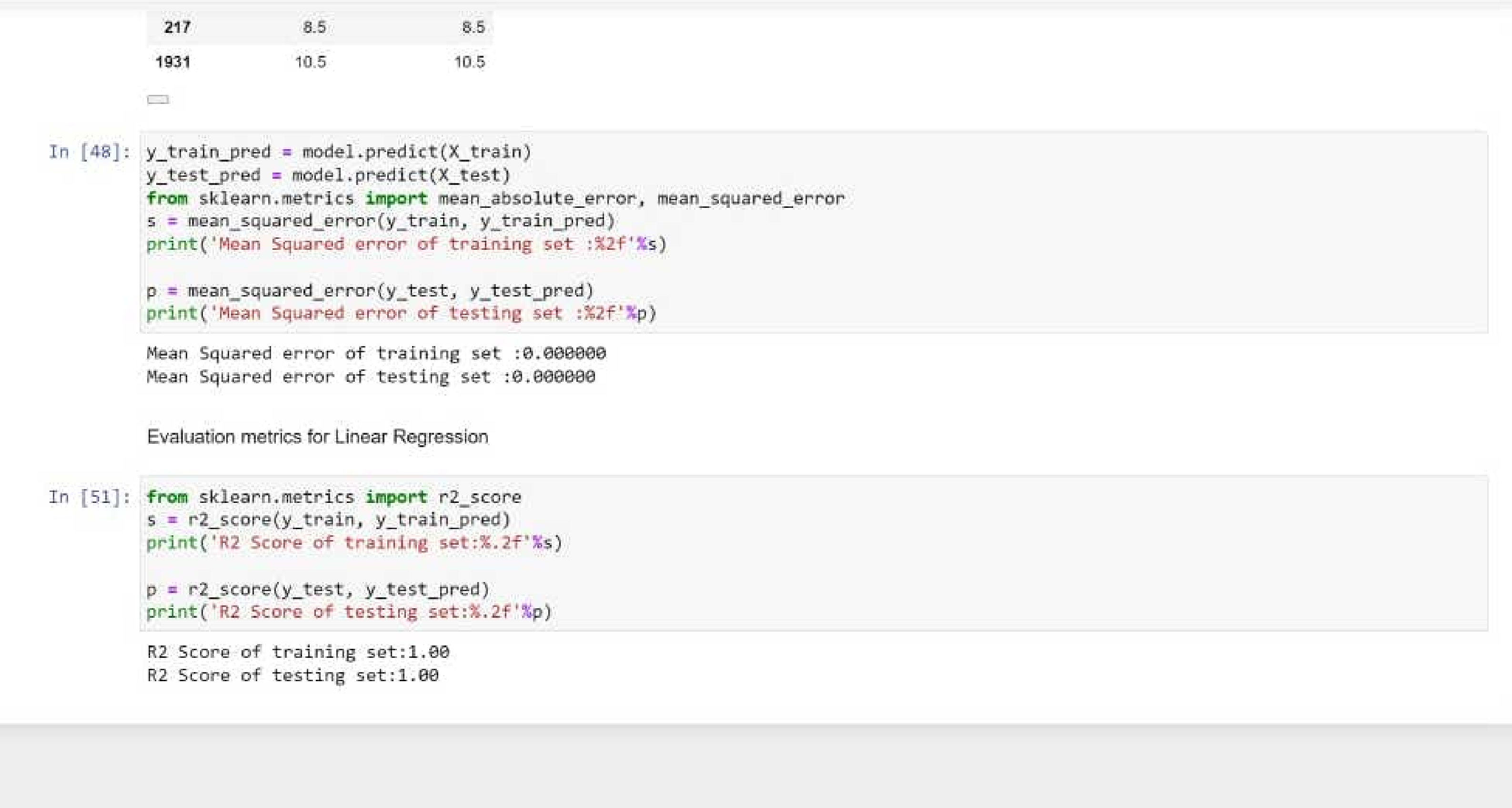
In [46]:
```
y_test
```

Out[46]:
```
668     14.5
1580     9.5
3784    12.5
463      6.5
2615    13.5
        ...
1420    12.5
2104    12.5
3382    16.5
3424    11.5
1160    10.5
Name: Length, Length: 1045, dtype: float64
```

In [47]:
```
Length=pd.DataFrame({'Actual_y_value':y_test,'Predicted_y_value':y_pred})
Length.head(10)
```

Out[47]:

	Actual_y_value	Predicted_y_value
668	14.5	14.5
1580	9.5	9.5
3784	12.5	12.5
463	6.5	6.5
2615	13.5	13.5
1399	12.5	12.5



217	8.5	8.5
1931	10.5	10.5

```
In [48]: y_train_pred = model.predict(X_train)
         y_test_pred = model.predict(X_test)
         from sklearn.metrics import mean_absolute_error, mean_squared_error
         s = mean_squared_error(y_train, y_train_pred)
         print('Mean Squared error of training set :%2f'%s)

         p = mean_squared_error(y_test, y_test_pred)
         print('Mean Squared error of testing set :%2f'%p)
```

```
Mean Squared error of training set :0.000000
Mean Squared error of testing set :0.000000
```

Evaluation metrics for Linear Regression

```
In [51]: from sklearn.metrics import r2_score
         s = r2_score(y_train, y_train_pred)
         print('R2 Score of training set:%.2f'%s)

         p = r2_score(y_test, y_test_pred)
         print('R2 Score of testing set:%.2f'%p)
```

```
R2 Score of training set:1.00
R2 Score of testing set:1.00
```