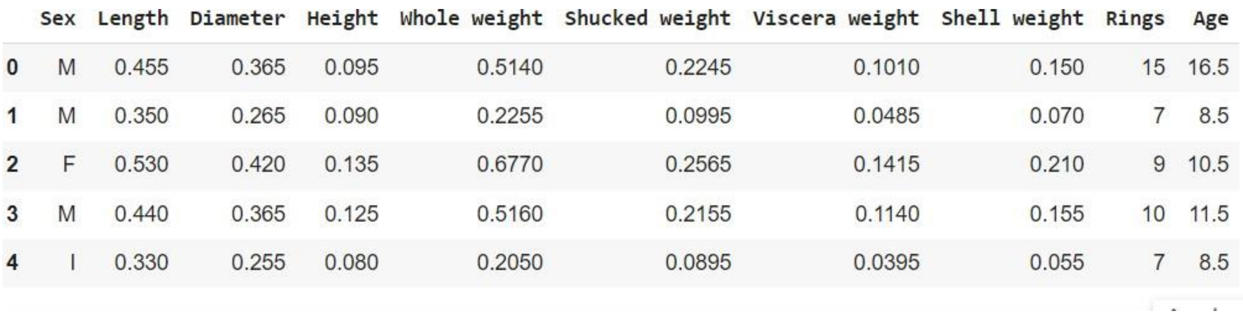
ASSESSMENT 3

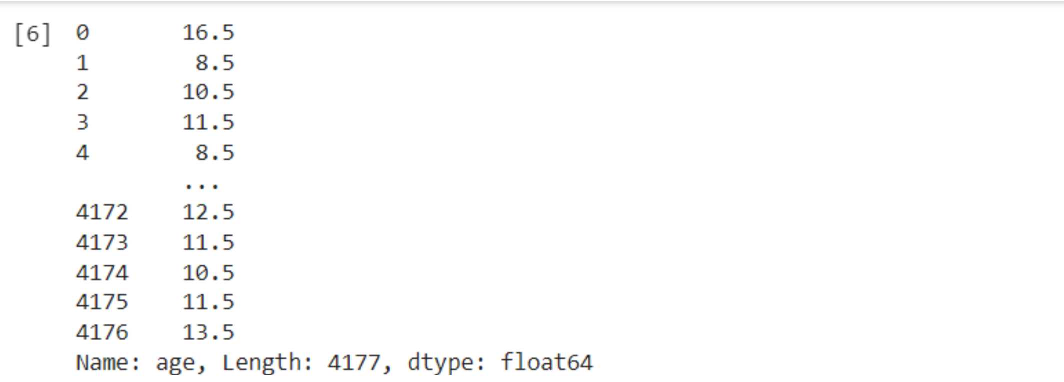
ASSESSMENT DATE	06-10-2022
STUDENT NAME	Saruthi.M
STUDENT ROLL NUMBER	713119205008
MAXIMUM MARKS	2 Marks

```
import pandas as pd
import numpy as np


#1. Download the dataset

from google.colab import files
uploaded = files.upload()

#2. Load the dataset into the tool.

df=pd.read_csv('abalone - abalone.csv')
df.head()
```



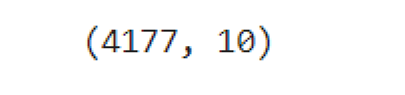
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	8.5

```
0s
df['age'] = df['Rings']+1.5
df.drop('Rings', axis = 1, inplace = True)
df.age
```



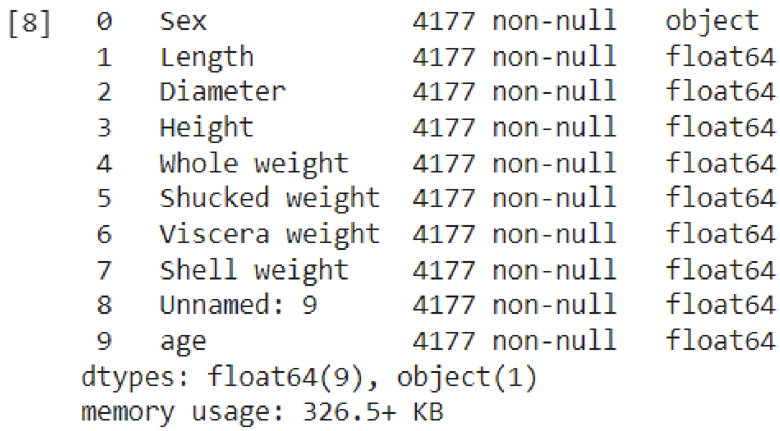
```
[6] 0       16.5
    1        8.5
    2       10.5
    3       11.5
    4        8.5
            ...
    4172    12.5
    4173    11.5
    4174    10.5
    4175    11.5
    4176    13.5
    Name: age, Length: 4177, dtype: float64
```

```
df.shape
```



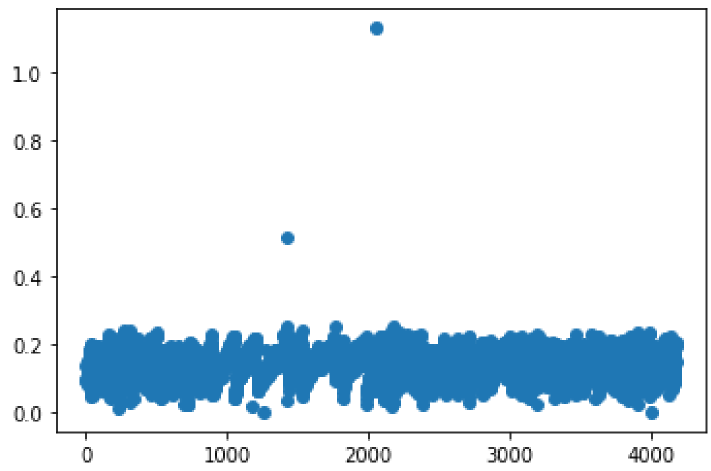
```
(4177, 10)
```

```
df.info()
```



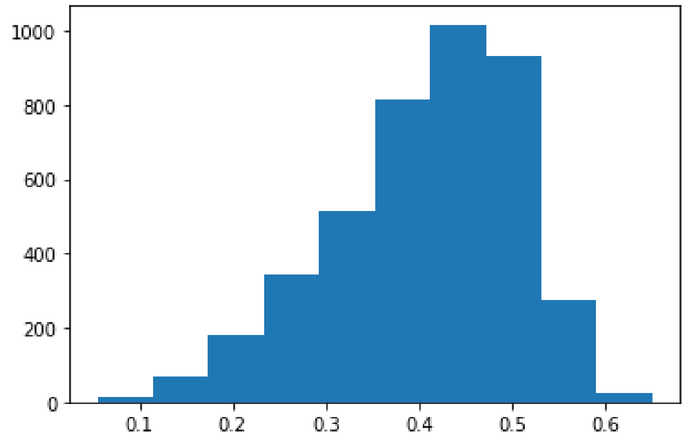
```
[8]  0   Sex             4177 non-null   object
     1   Length          4177 non-null   float64
     2   Diameter        4177 non-null   float64
     3   Height          4177 non-null   float64
     4   Whole weight    4177 non-null   float64
     5   Shucked weight  4177 non-null   float64
     6   Viscera weight  4177 non-null   float64
     7   Shell weight    4177 non-null   float64
     8   Unnamed: 9      4177 non-null   float64
     9   age             4177 non-null   float64
    dtypes: float64(9), object(1)
    memory usage: 326.5+ KB
```

```
# 3. Perform Below Visualizations.
#univariate analysis
import matplotlib.pyplot as plt
import seaborn as sns
plt.scatter(df.index,df['Height'])
plt.show()
```

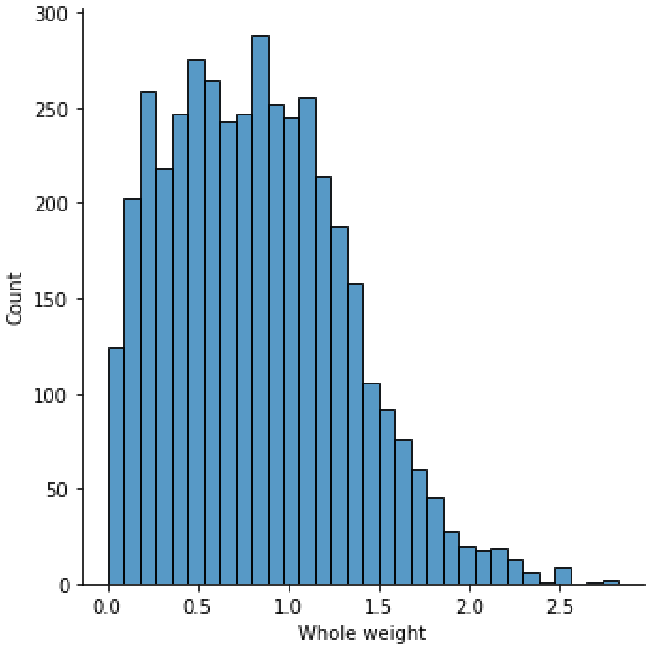


```
plt.hist(df['Diameter'])
```

```
(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>)
```

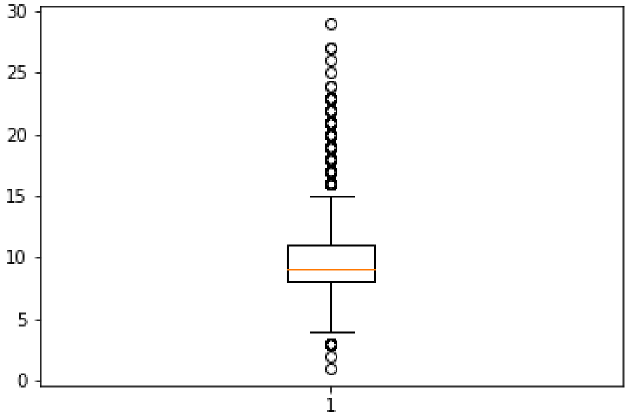


```
sns.displot(df['Whole weight'])
<seaborn.axisgrid.FacetGrid at 0x7f400117f050>
```

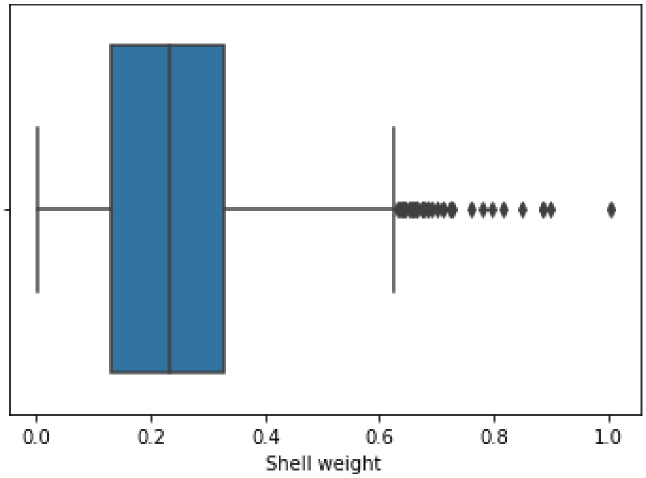


```
plt.boxplot(df['age'])
```

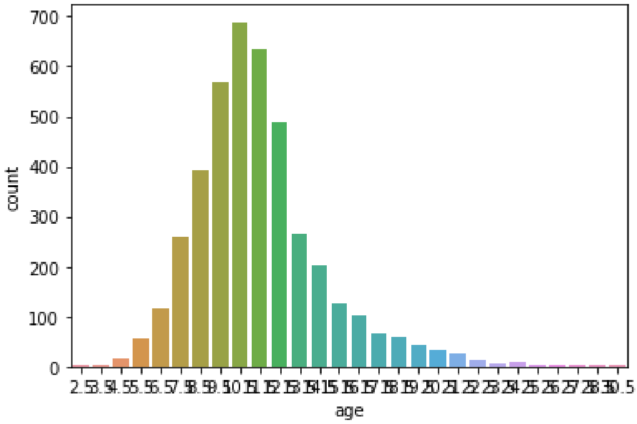
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f3ffe636e50>,
  <matplotlib.lines.Line2D at 0x7f3ffe636b50>],
 'caps': [<matplotlib.lines.Line2D at 0x7f3ffe6384d0>,
  <matplotlib.lines.Line2D at 0x7f3ffe638b50>],
 'boxes': [<matplotlib.lines.Line2D at 0x7f3ffe6368d0>],
 'medians': [<matplotlib.lines.Line2D at 0x7f3ffe5dff90>],
 'fliers': [<matplotlib.lines.Line2D at 0x7f3ffe5df790>],
 'means': []}
```



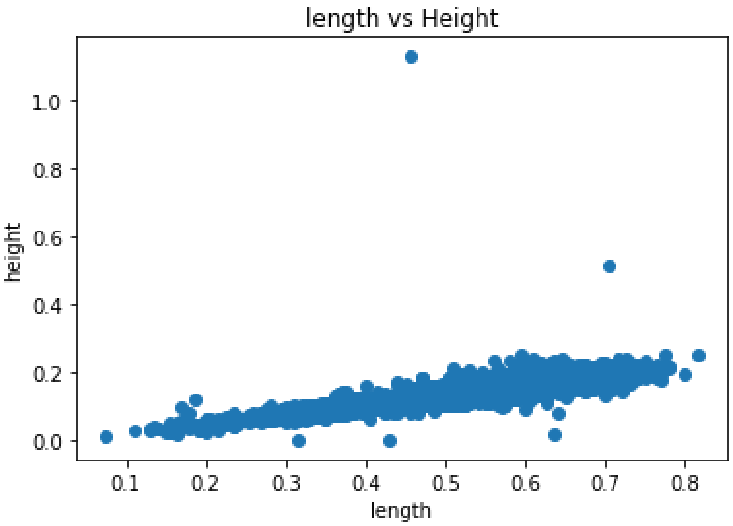
```
sns.boxplot(df['Shell weight'])
```



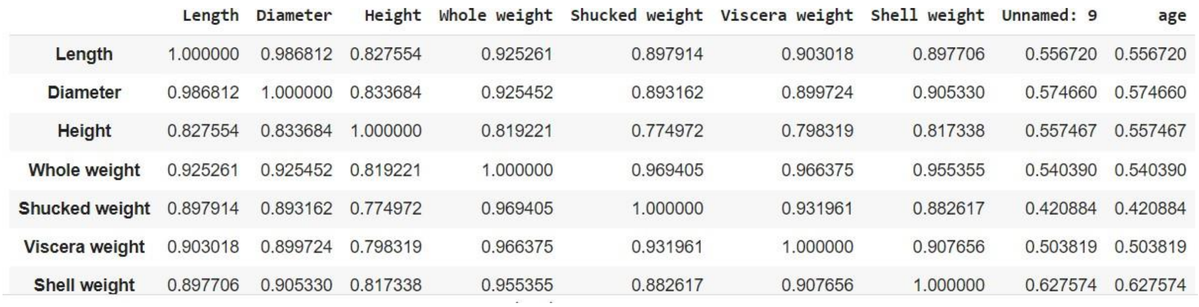
```
sns.countplot(df['age'])
```



```
#Bivariate analysis
plt.scatter(df.Length,df.Height)
plt.title('length vs Height')
plt.xlabel('length')
plt.ylabel('height')
```

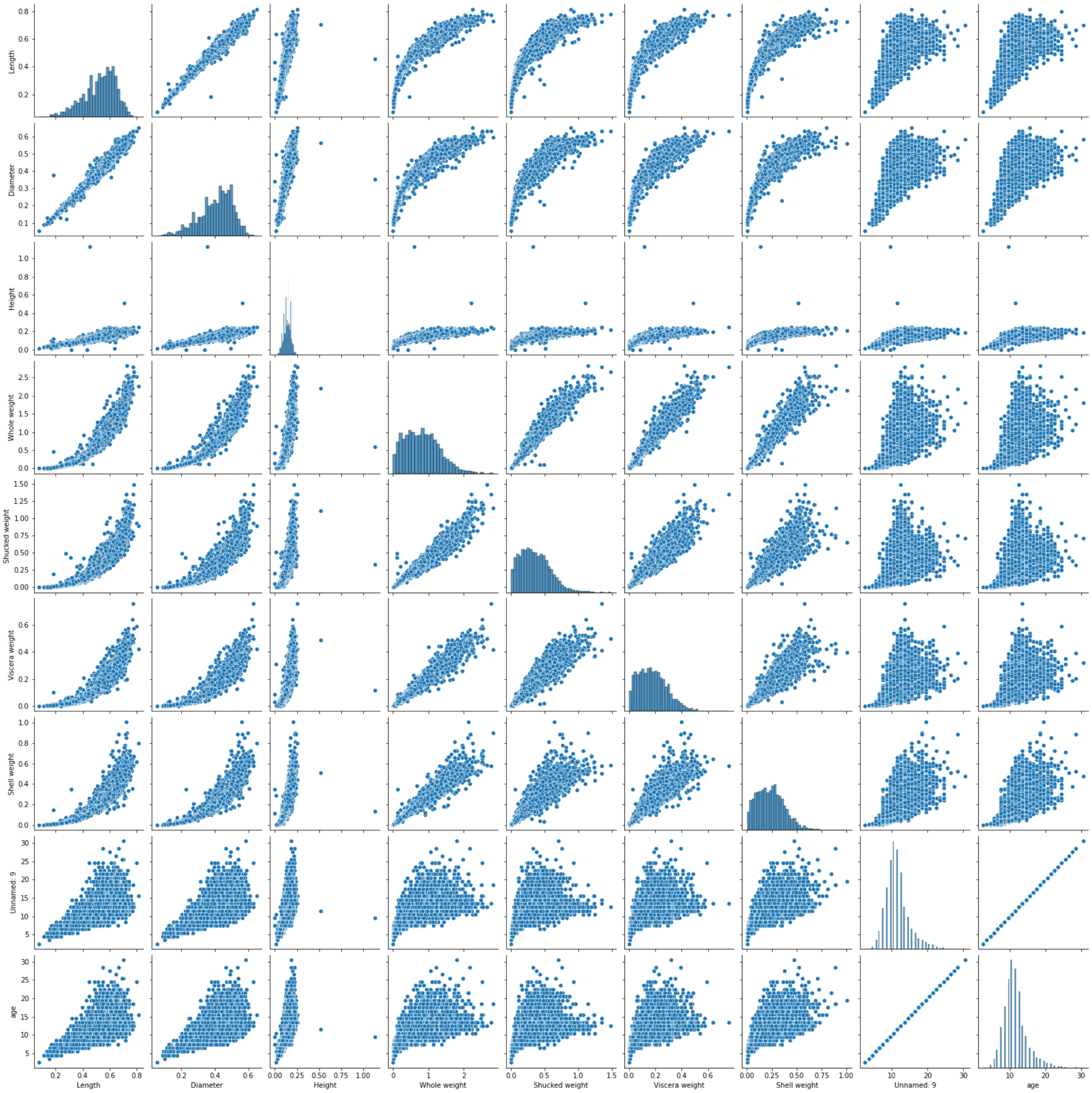


```
df.corr()
```

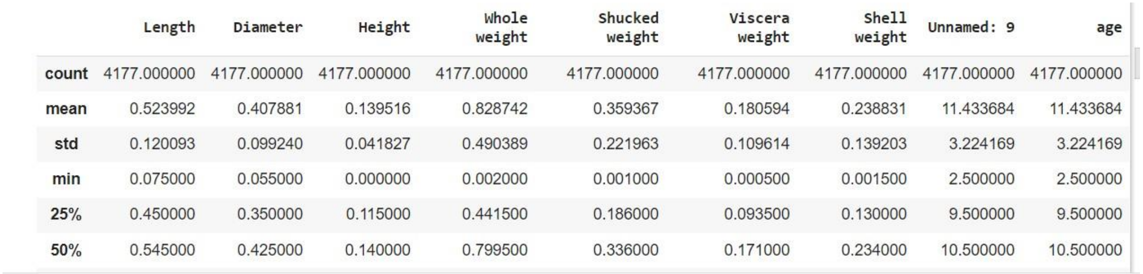


	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Unnamed: 9	age
Length	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720	0.556720
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660	0.574660
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467	0.557467
Whole weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390	0.540390
Shucked weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884	0.420884
Viscera weight	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819	0.503819
Shell weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574	0.627574

```
#multivariate analysis
sns.pairplot(df)
plt.show
```



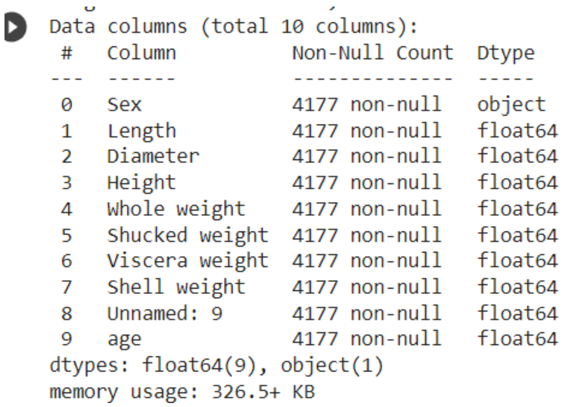
```
# 4.Perform descriptive statistics on the dataset.
df.describe()
```



	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Unnamed: 9	age
count	4177.000000	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	11.433684	11.433684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	2.500000	2.500000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	9.500000	9.500000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	10.500000	10.500000

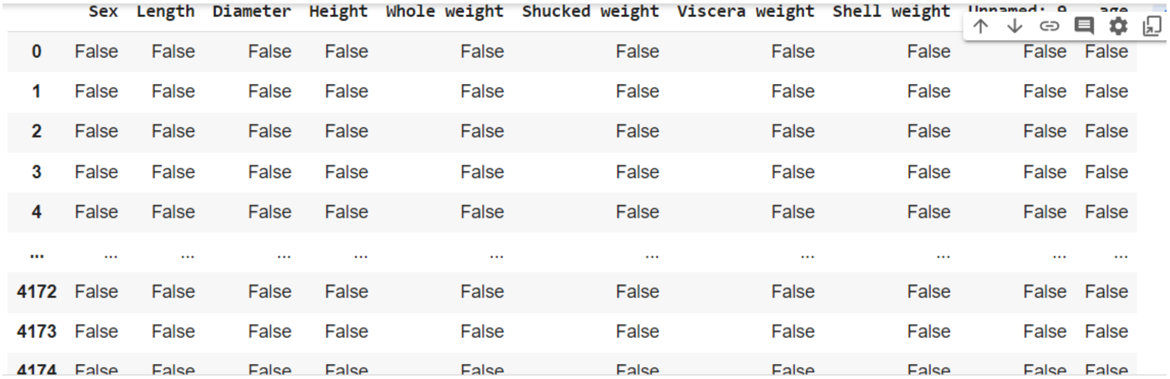
```
#5.Check for Missing values and deal with them.
```

df.info()



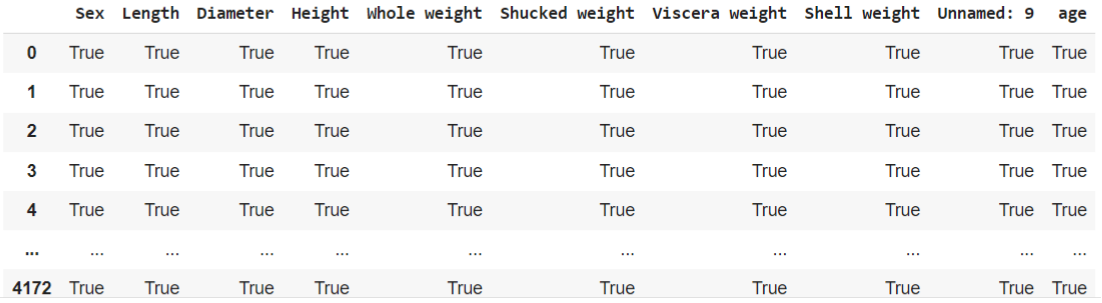
```
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4177 non-null   object 
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 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   Unnamed: 9      4177 non-null   float64
 9   age             4177 non-null   float64
dtypes: float64(9), object(1)
memory usage: 326.5+ KB
```

df.isnull()



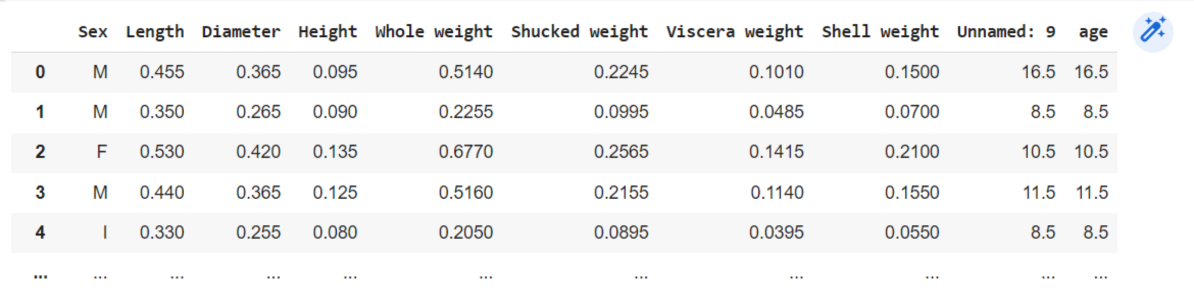
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Unnamed: 9	age
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
4172	False	False	False	False	False	False	False	False	False	False
4173	False	False	False	False	False	False	False	False	False	False
4174	False	False	False	False	False	False	False	False	False	False

df.notnull()



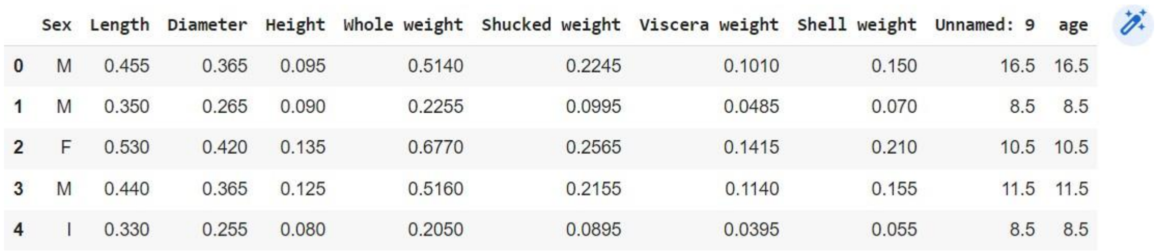
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Unnamed: 9	age
0	True	True	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...
4172	True	True	True	True	True	True	True	True	True	True

```
df.fillna(0)
```



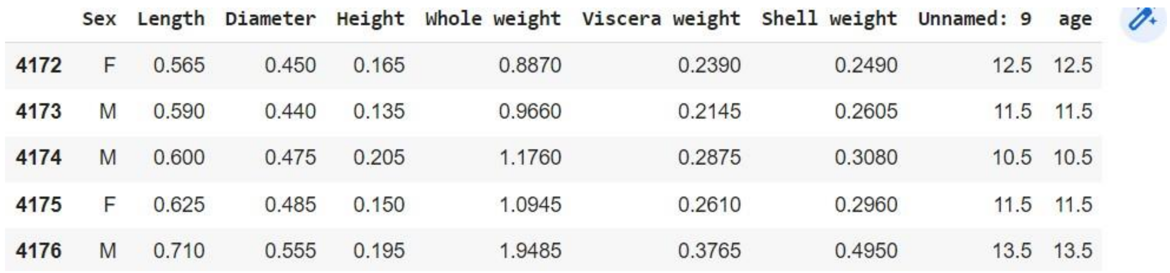
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Unnamed: 9	age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	16.5	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	8.5	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	10.5	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	11.5	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	8.5	8.5
...	...	...	...	...	...	...	...	...	...	...

```
df['Length'].fillna('No Length',inplace=True)
df.head()
```



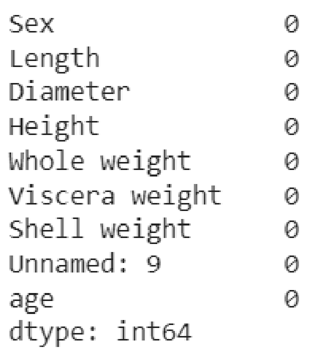
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Unnamed: 9	age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5	8.5

```
df.drop('Shucked weight',axis=1,inplace=True)
df.tail()
```



	Sex	Length	Diameter	Height	Whole weight	Viscera weight	Shell weight	Unnamed: 9	age
4172	F	0.565	0.450	0.165	0.8870	0.2390	0.2490	12.5	12.5
4173	M	0.590	0.440	0.135	0.9660	0.2145	0.2605	11.5	11.5
4174	M	0.600	0.475	0.205	1.1760	0.2875	0.3080	10.5	10.5
4175	F	0.625	0.485	0.150	1.0945	0.2610	0.2960	11.5	11.5
4176	M	0.710	0.555	0.195	1.9485	0.3765	0.4950	13.5	13.5

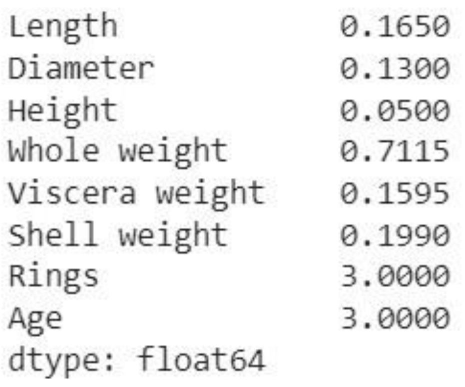
```
print(df.isnull().sum())
```



```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Viscera weight    0
Shell weight      0
Unnamed: 9        0
age               0
dtype: int64
```

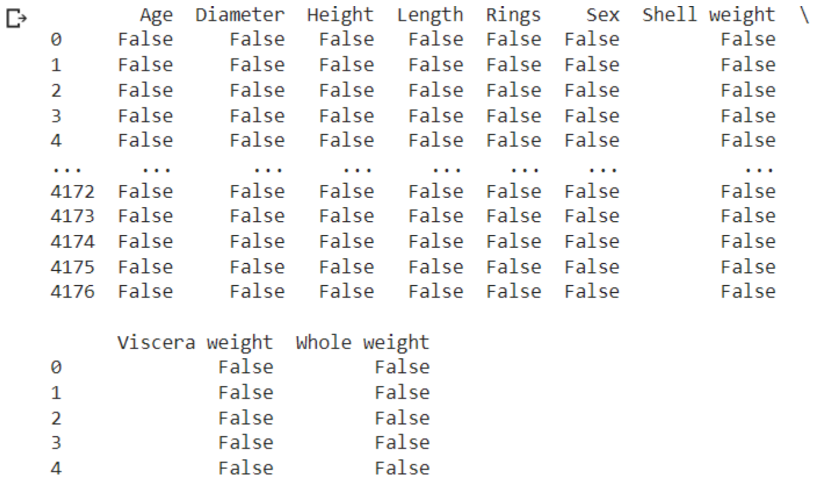
```
#6.Find the outliers and replace them outliers
Q1=df.quantile(0.25)
```

```
Q3=df.quantile(0.75)
IQR=Q3-Q1
print(IQR)
```



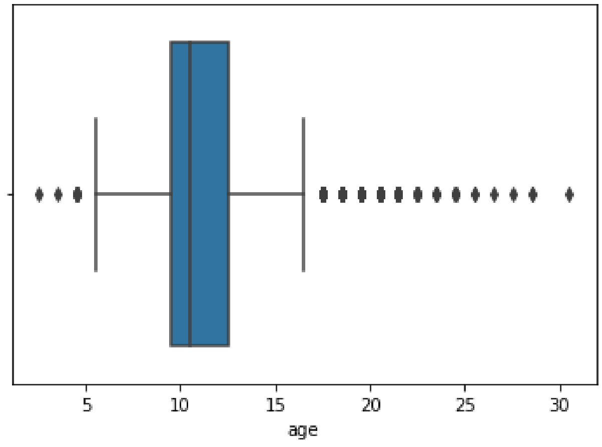
```
Length            0.1650
Diameter          0.1300
Height            0.0500
Whole weight      0.7115
Viscera weight    0.1595
Shell weight      0.1990
Rings             3.0000
Age               3.0000
dtype: float64
```

```
print(df<(Q1-1.5*IQR))
(df>(Q3+1.5*IQR))
```

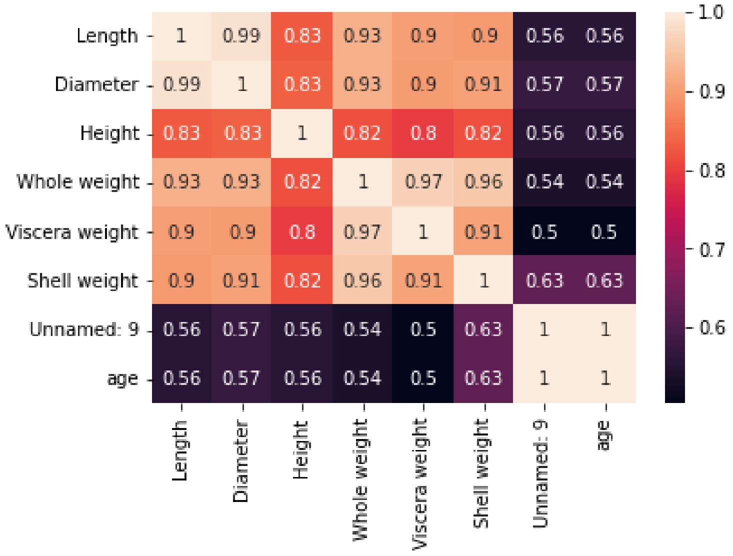


```
        Age  Diameter  Height  Length  Rings    Sex  Shell weight  \
0     False     False   False   False  False  False         False
1     False     False   False   False  False  False         False
2     False     False   False   False  False  False         False
3     False     False   False   False  False  False         False
4     False     False   False   False  False  False         False
...     ...       ...     ...     ...    ...    ...           ...
4172  False     False   False   False  False  False         False
4173  False     False   False   False  False  False         False
4174  False     False   False   False  False  False         False
4175  False     False   False   False  False  False         False
4176  False     False   False   False  False  False         False

      Viscera weight  Whole weight
0              False         False
1              False         False
2              False         False
3              False         False
4              False         False
```

```
sns.boxplot(df.Rings)
```



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

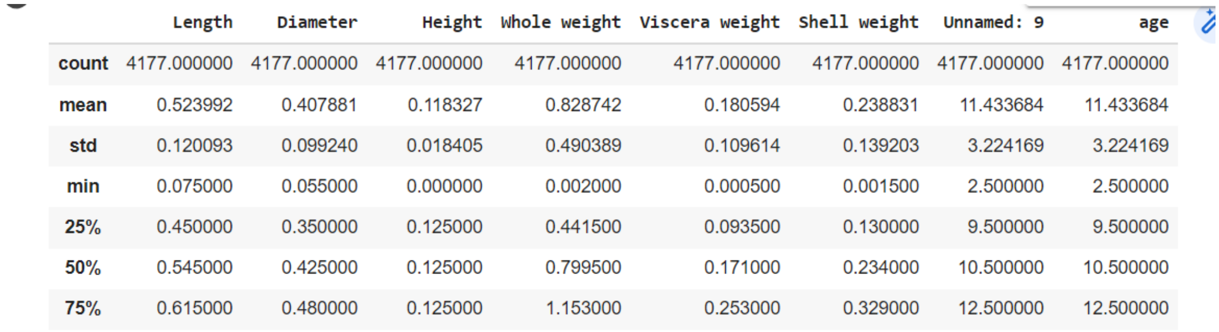


```
np.where(df.age>7,7,df.age)
```

output

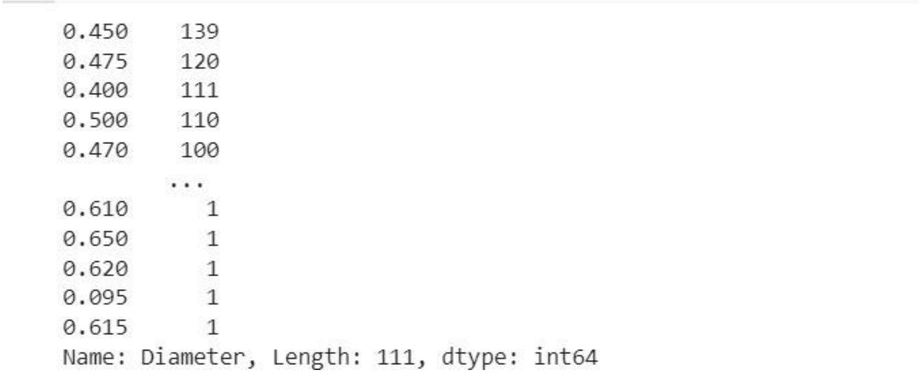
```
array([7, 7, 7, ..., 7, 7, 7])
```

```
print(df['Height'].quantile(0.25))
print(df['Height'].quantile(0.75))
df['Height']=np.where(df['Height']>0.090  ,0.125,df['Height'])
df.describe()
```



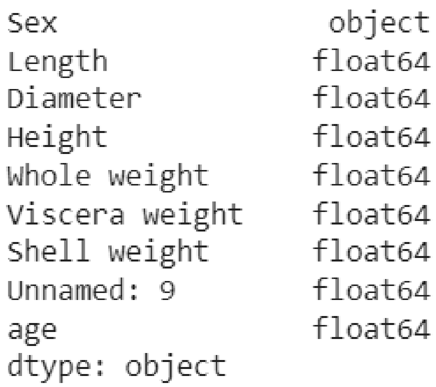
	Length	Diameter	Height	Whole weight	Viscera weight	Shell weight	Unnamed: 9	age
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.118327	0.828742	0.180594	0.238831	11.433684	11.433684
std	0.120093	0.099240	0.018405	0.490389	0.109614	0.139203	3.224169	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.000500	0.001500	2.500000	2.500000
25%	0.450000	0.350000	0.125000	0.441500	0.093500	0.130000	9.500000	9.500000
50%	0.545000	0.425000	0.125000	0.799500	0.171000	0.234000	10.500000	10.500000
75%	0.615000	0.480000	0.125000	1.153000	0.253000	0.329000	12.500000	12.500000

```
#  7.Check for Categorical columns and perform encoding.
df['Diameter'].value_counts()
```



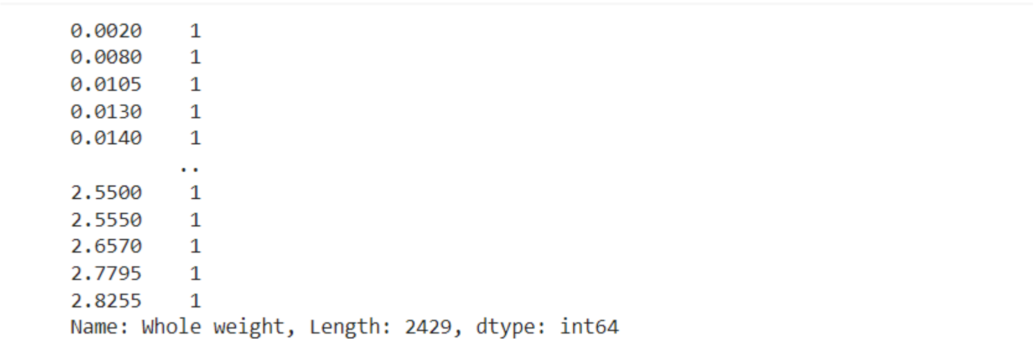
```
0.450    139
0.475    120
0.400    111
0.500    110
0.470    100
        ...
0.610      1
0.650      1
0.620      1
0.095      1
0.615      1
Name: Diameter, Length: 111, dtype: int64
```

```
df.dtypes
```



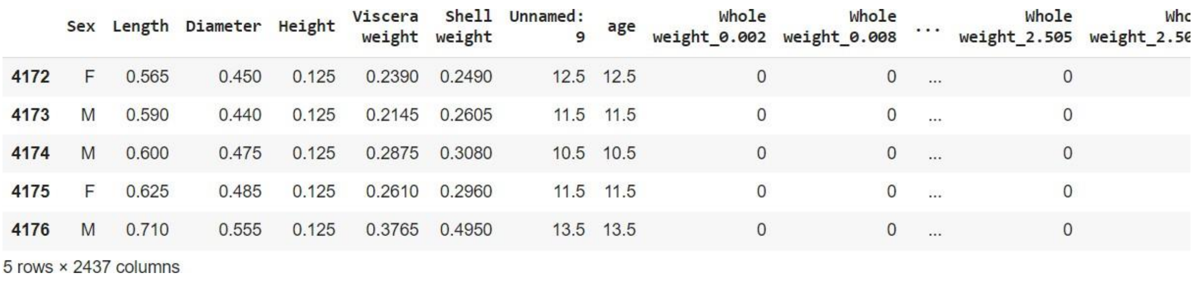
```
Sex                object
Length            float64
Diameter          float64
Height            float64
Whole weight      float64
Viscera weight    float64
Shell weight      float64
Unnamed: 9        float64
age               float64
dtype: object
```

```
df['Whole weight'].value_counts().sort_index()
```



```
0.0020    1
0.0080    1
0.0105    1
0.0130    1
0.0140    1
         ..
2.5500    1
2.5550    1
2.6570    1
2.7795    1
2.8255    1
Name: Whole weight, Length: 2429, dtype: int64
```

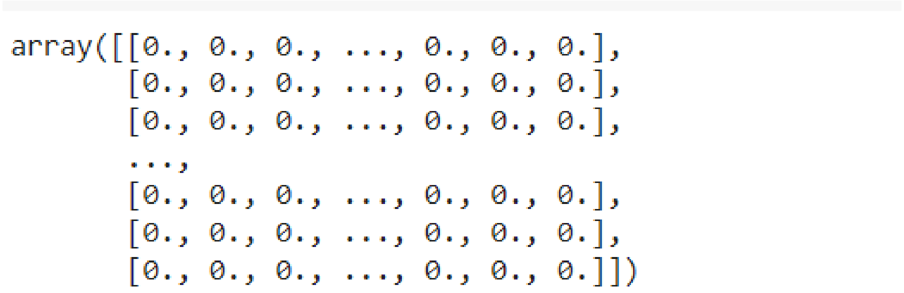
```
pd.get_dummies(df,columns=['Whole weight']).tail()
```



	Sex	Length	Diameter	Height	Viscera weight	Shell weight	Unnamed: 9	age	Whole weight_0.002	Whole weight_0.008	...	Whole weight_2.505	Whc weight_2.5
4172	F	0.565	0.450	0.125	0.2390	0.2490	12.5	12.5	0	0	...	0	
4173	M	0.590	0.440	0.125	0.2145	0.2605	11.5	11.5	0	0	...	0	
4174	M	0.600	0.475	0.125	0.2875	0.3080	10.5	10.5	0	0	...	0	
4175	F	0.625	0.485	0.125	0.2610	0.2960	11.5	11.5	0	0	...	0	
4176	M	0.710	0.555	0.125	0.3765	0.4950	13.5	13.5	0	0	...	0	

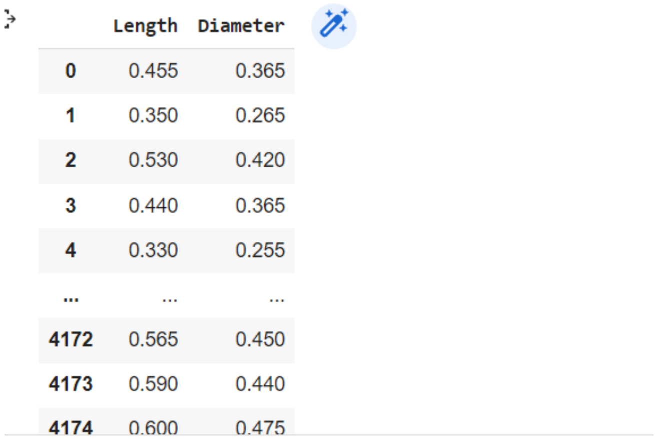
5 rows × 2437 columns

```
from sklearn.preprocessing import OneHotEncoder

one_encde= OneHotEncoder(sparse=False)
encoded_arr=one_encde.fit_transform(df[['Length','Diameter','Height','Visc
era weight']])
encoded_arr
```



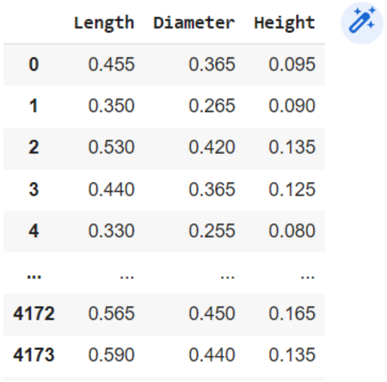
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

```
#8. Split the data into dependent and independent variables.
x=df.iloc[:,1:3]
x
```



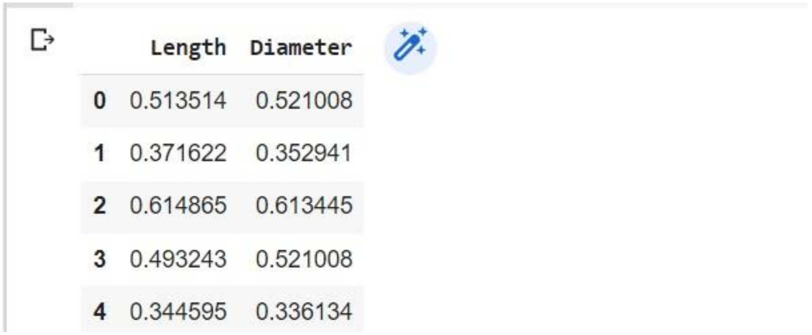
	Length	Diameter
0	0.455	0.365
1	0.350	0.265
2	0.530	0.420
3	0.440	0.365
4	0.330	0.255
...	...	...
4172	0.565	0.450
4173	0.590	0.440
4174	0.600	0.475

```
y=df.iloc[:,1:4]
y
```



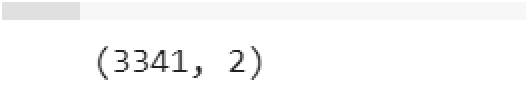
	Length	Diameter	Height
0	0.455	0.365	0.095
1	0.350	0.265	0.090
2	0.530	0.420	0.135
3	0.440	0.365	0.125
4	0.330	0.255	0.080
...	...	...	...
4172	0.565	0.450	0.165
4173	0.590	0.440	0.135

```
#9. Scale the independent variables.
from sklearn.preprocessing import MinMaxScaler
model=MinMaxScaler()
scaled_x=pd.DataFrame(model.fit_transform(x),columns=x.columns)
scaled_x.head()
```



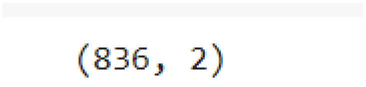
	Length	Diameter
0	0.513514	0.521008
1	0.371622	0.352941
2	0.614865	0.613445
3	0.493243	0.521008
4	0.344595	0.336134

```
#10. Split the data into 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_st
ate=0)
x_train.shape
```



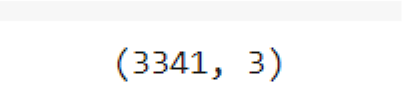
```
(3341, 2)
```

```
x_test.shape
```



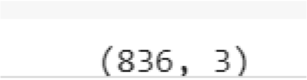
```
(836, 2)
```

```
y_train.shape
```



```
(3341, 3)
```

```
y_test.shape
```



```
(836, 3)
```

```
#11. Build the Model
from sklearn.linear_model import LinearRegression

model = LinearRegression()

model.fit(x,y)
```

output

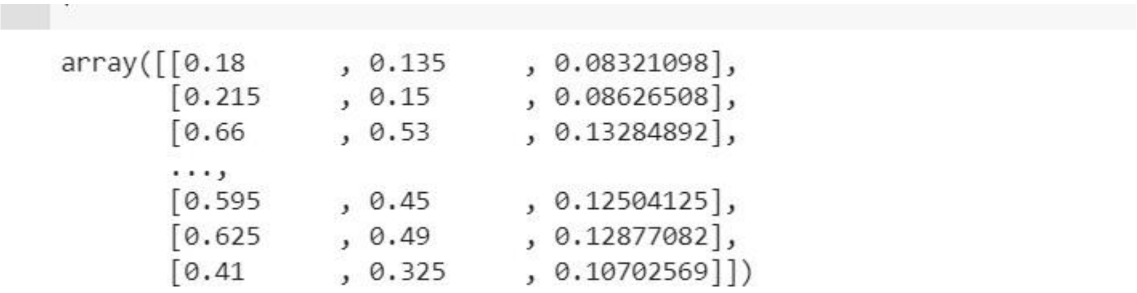
```
LinearRegression()
```

```
#12. Train the Model

model.fit(x_train, y_train)
LinearRegression()
```

output

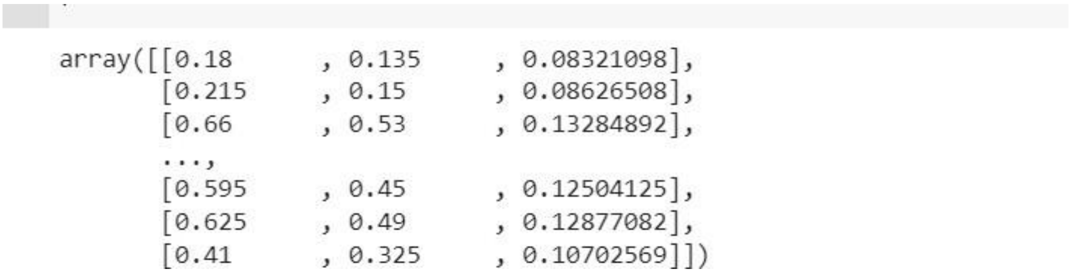
```
LinearRegression()
```

```
#13. Test the Model
pred1=model.predict(x_train)
pred1
```



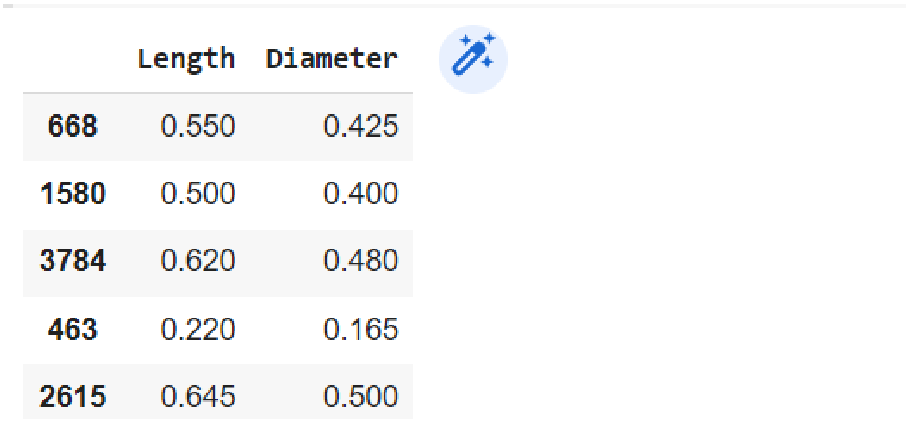
```
array([[0.18      , 0.135     , 0.08321098],
       [0.215     , 0.15      , 0.08626508],
       [0.66      , 0.53      , 0.13284892],
       ...,
       [0.595     , 0.45      , 0.12504125],
       [0.625     , 0.49      , 0.12877082],
       [0.41      , 0.325     , 0.10702569]])
```

```
predictions=model.predict(x_test)
predictions
pred=model.predict(x_test)
pred
```



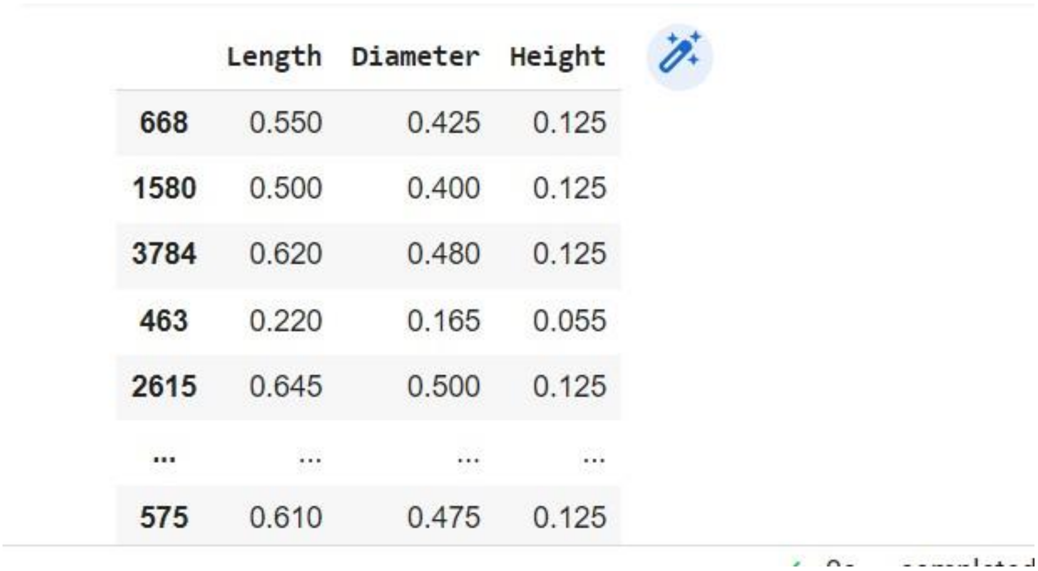
```
array([[0.18      , 0.135     , 0.08321098],
       [0.215     , 0.15      , 0.08626508],
       [0.66      , 0.53      , 0.13284892],
       ...,
       [0.595     , 0.45      , 0.12504125],
       [0.625     , 0.49      , 0.12877082],
       [0.41      , 0.325     , 0.10702569]])
```

```
pred=model.predict(x_test)
pred
y_pred=(x_test)
y_pred
```



	Length	Diameter
668	0.550	0.425
1580	0.500	0.400
3784	0.620	0.480
463	0.220	0.165
2615	0.645	0.500

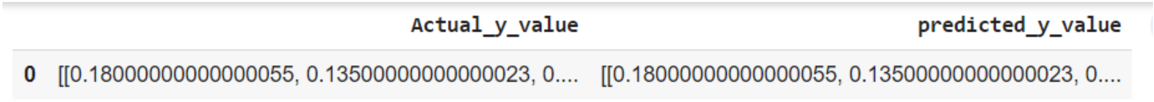
```
y_test
```



	Length	Diameter	Height
668	0.550	0.425	0.125
1580	0.500	0.400	0.125
3784	0.620	0.480	0.125
463	0.220	0.165	0.055
2615	0.645	0.500	0.125
...	...	...	...
575	0.610	0.475	0.125

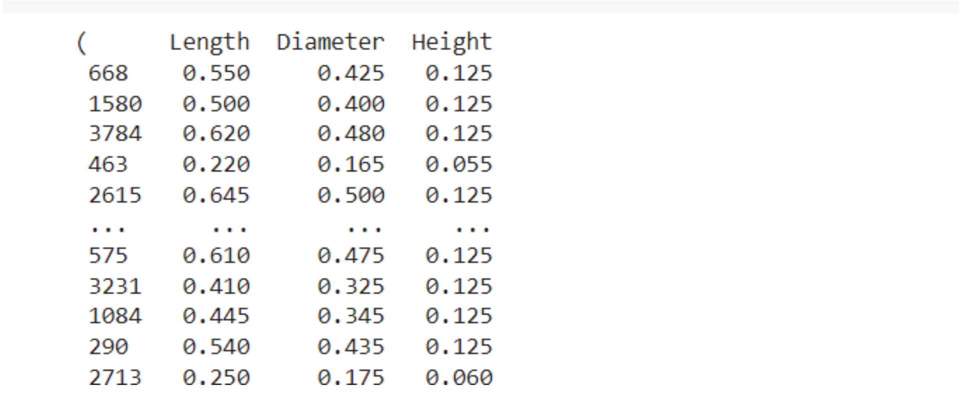
✓ 0s completed

```
length=pd.DataFrame({'Actual_y_value':[pred1],'predicted_y_value':[pred]})
length
```



	Actual_y_value	predicted_y_value
0	[[0.18000000000000055, 0.13500000000000023, 0....	[[0.18000000000000055, 0.13500000000000023, 0....

```
from sklearn import metrics
#Mean Absolute Error (MAE)
metrics.mean_absolute_error(y_test, predictions)
0.0038991947837602914
#Mean Squared Error (MSE)
metrics.mean_squared_error(y_test, predictions)
7.655875085238909e-05
#Root Mean Squared Error (RMSE)
np.sqrt(metrics.mean_squared_error(y_test, predictions))
0.008749785760370884
#14. Measure the performance using Metrics.
from sklearn.metrics import accuracy_score
accuracy_score=(y_test,y_pred)
accuracy_score
```



```
(      Length  Diameter  Height
 668    0.550     0.425   0.125
 1580   0.500     0.400   0.125
 3784   0.620     0.480   0.125
 463    0.220     0.165   0.055
 2615   0.645     0.500   0.125
 ...      ...       ...     ...
 575    0.610     0.475   0.125
 3231   0.410     0.325   0.125
 1084   0.445     0.345   0.125
 290    0.540     0.435   0.125
 2713   0.250     0.175   0.060
```

```
from sklearn.metrics import classification_report
classification_report(y_test,y_pred)
classification_report

<function sklearn.metrics._classification.classification_report(y_true,
y_pred, *, labels=None, target_names=None, sample_weight=None, digits=2,
output_dict=False, zero_division='warn')>
```