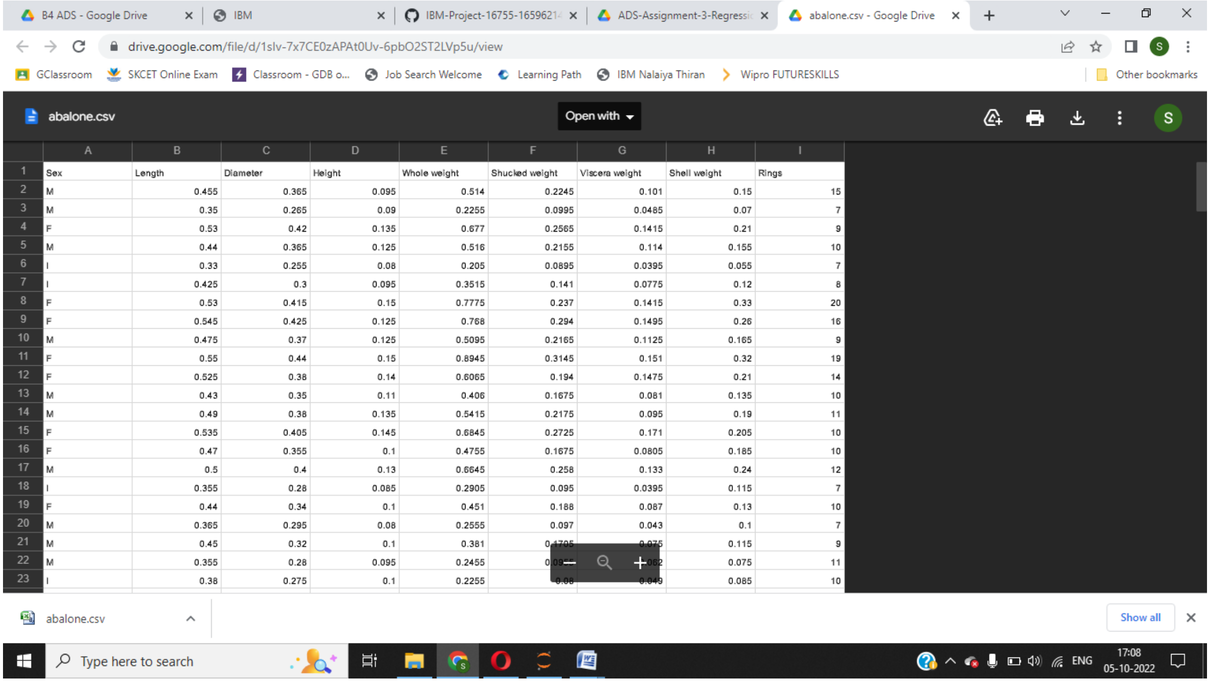
ASSIGNMENT -3
REGRESSION

Assignment Date	29 September 2022
Student Name	Manjunathan V
Student Roll Number	727719EUCS080
Maximum Marks	2 Marks

Question-1:

Download the dataset: Dataset



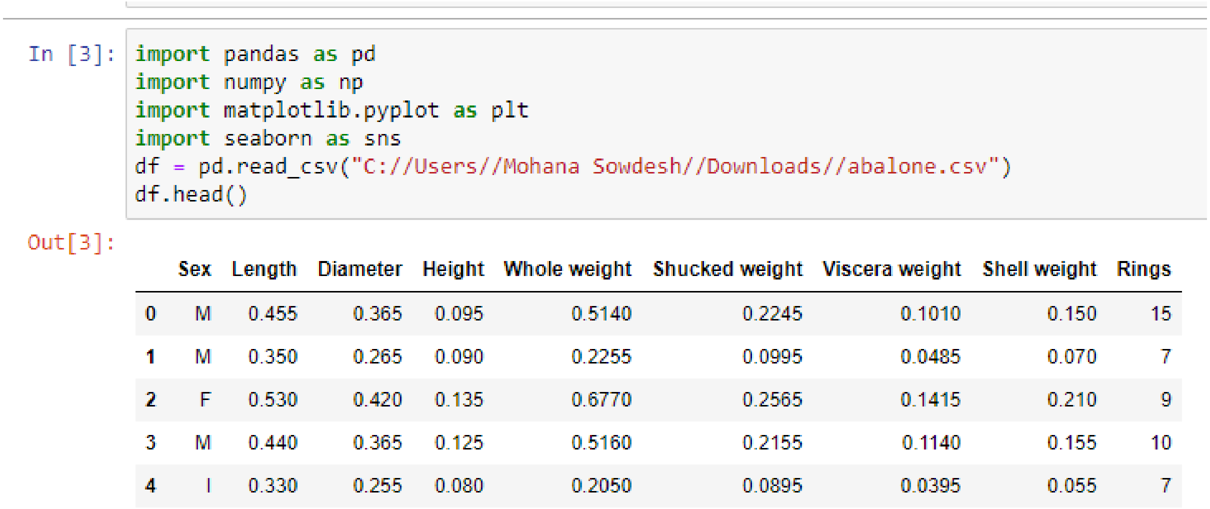
	A	B	C	D	E	F	G	H	I
1	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
2	M	0.455	0.365	0.095	0.514	0.2245	0.101	0.15	15
3	M	0.35	0.265	0.09	0.2255	0.0995	0.0485	0.07	7
4	F	0.53	0.42	0.135	0.677	0.2565	0.1415	0.21	9
5	M	0.44	0.365	0.125	0.516	0.2155	0.114	0.155	10
6	I	0.33	0.255	0.08	0.205	0.0895	0.0395	0.055	7
7	I	0.425	0.3	0.095	0.3515	0.141	0.0775	0.12	8
8	F	0.53	0.415	0.15	0.7775	0.237	0.1415	0.33	20
9	F	0.545	0.425	0.125	0.768	0.294	0.1495	0.26	16
10	M	0.475	0.37	0.125	0.5095	0.2165	0.1125	0.165	9
11	F	0.55	0.44	0.15	0.8945	0.3145	0.151	0.32	19
12	F	0.525	0.38	0.14	0.6065	0.194	0.1475	0.21	14
13	M	0.43	0.35	0.11	0.406	0.1675	0.081	0.135	10
14	M	0.49	0.38	0.135	0.5415	0.2175	0.095	0.19	11
15	F	0.535	0.405	0.145	0.6845	0.2725	0.171	0.205	10
16	F	0.47	0.355	0.1	0.4755	0.1675	0.0805	0.185	10
17	M	0.5	0.4	0.13	0.6645	0.258	0.133	0.24	12
18	I	0.355	0.28	0.085	0.2905	0.095	0.0395	0.115	7
19	F	0.44	0.34	0.1	0.451	0.188	0.087	0.13	10
20	M	0.365	0.295	0.08	0.2555	0.097	0.043	0.1	7
21	M	0.45	0.32	0.1	0.381	0.1705	0.075	0.115	9
22	M	0.355	0.28	0.095	0.2455	0.0955	0.062	0.075	11
23	I	0.38	0.275	0.1	0.2255	0.08	0.049	0.085	10

Question-2:

Load the dataset.

Solution:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

df = pd.read_csv("C://Users//Mohana Sowdesh//Downloads//abalone.csv")

df.head()
```



```
In [3]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        df = pd.read_csv("C://Users//Mohana Sowdesh//Downloads//abalone.csv")
        df.head()
```

Out[3]:

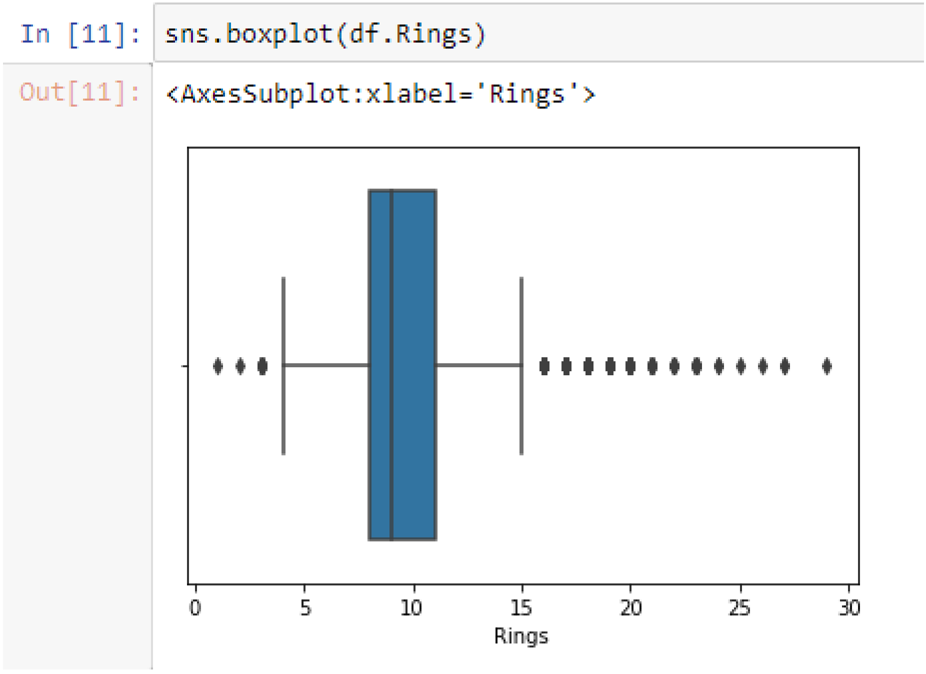
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330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

Question-3:

Perform Below Visualizations.

· Univariate Analysis

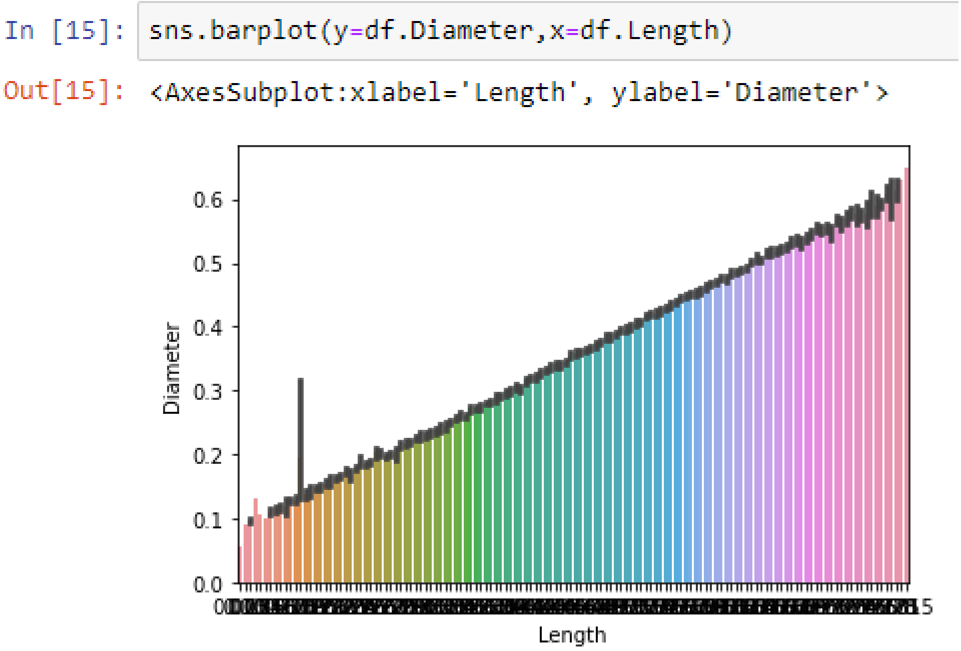
Solution:



```
In [11]: sns.boxplot(df.Rings)
Out[11]: <AxesSubplot:xlabel='Rings'>
```

· Bi-Variate Analysis

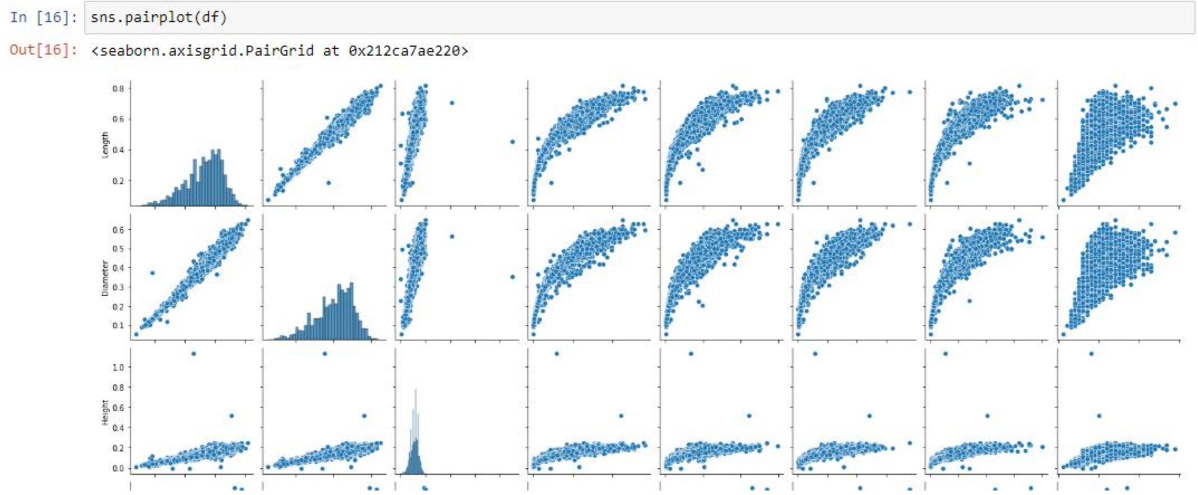
Solution:



```
In [15]: sns.barplot(y=df.Diameter,x=df.Length)

Out[15]: <AxesSubplot:xlabel='Length', ylabel='Diameter'>
```

· Multi-Variate Analysis

Solution:

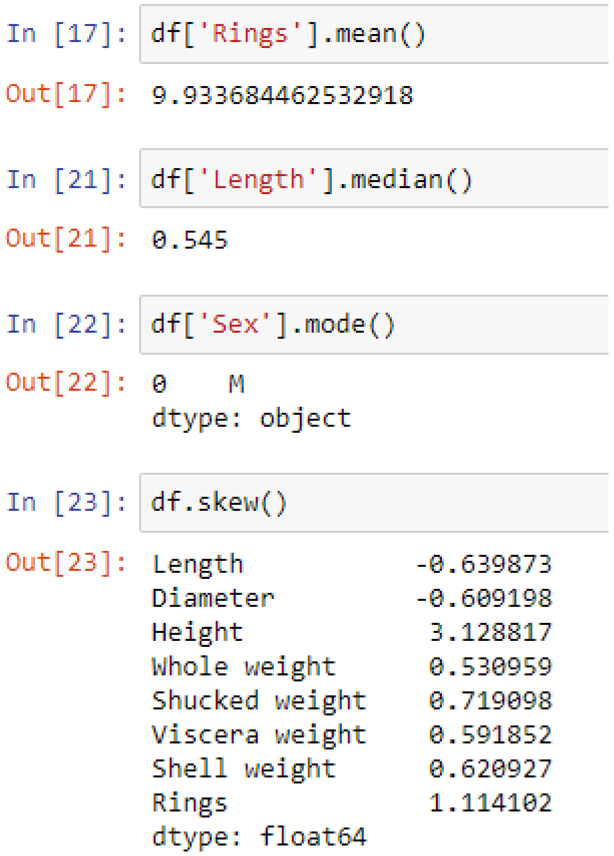


```
In [16]: sns.pairplot(df)

Out[16]: <seaborn.axisgrid.PairGrid at 0x212ca7ae220>
```

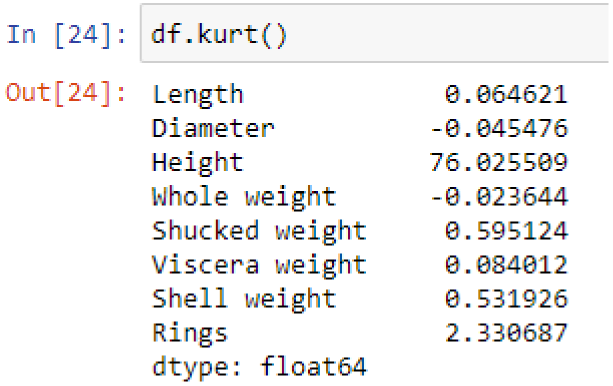
Question-4:

Perform descriptive statistics on the dataset.

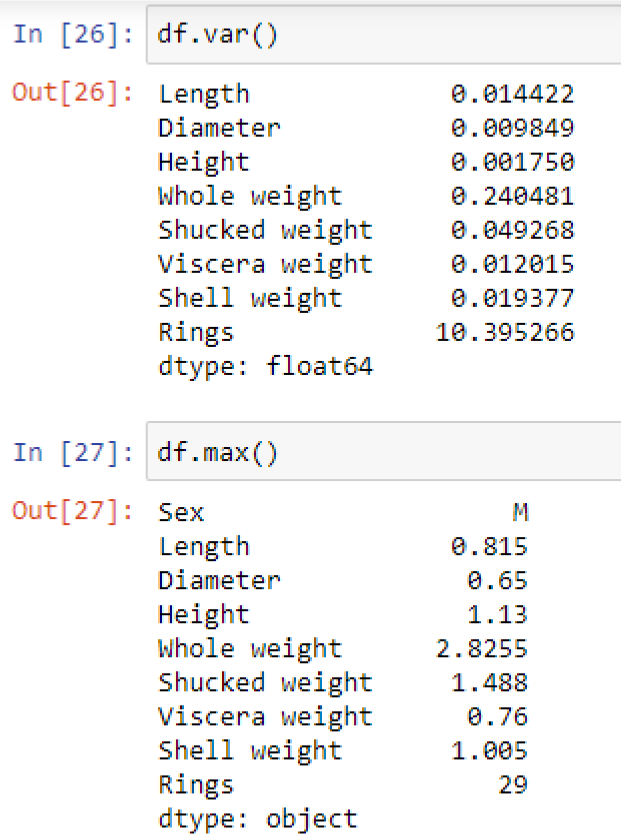
Solution:



```
In [17]: df['Rings'].mean()

Out[17]: 9.933684462532918

In [21]: df['Length'].median()

Out[21]: 0.545

In [22]: df['Sex'].mode()

Out[22]: 0    M
         dtype: object

In [23]: df.skew()

Out[23]: 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 [24]: df.kurt()

Out[24]: 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
```



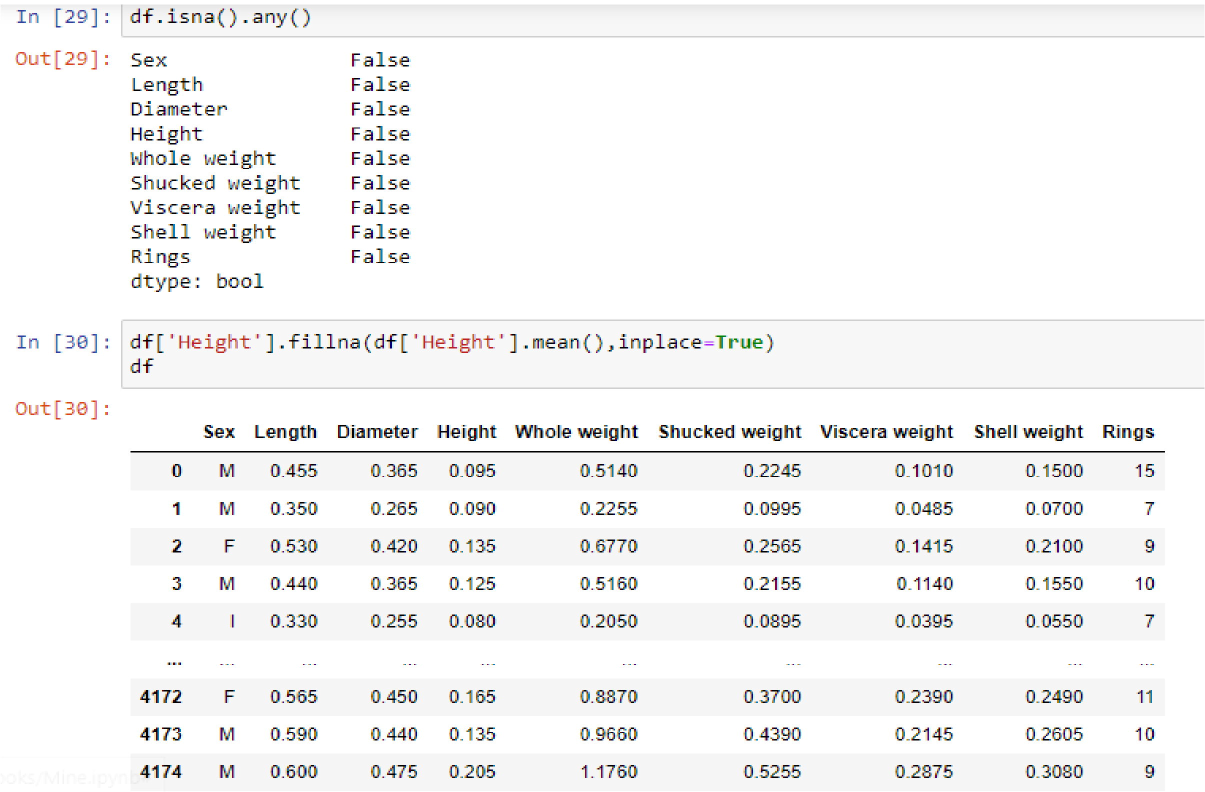
```
In [26]: df.var()

Out[26]: 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 [27]: df.max()

Out[27]: 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
```

Question-5:

Handle the Missing values.

Solution:



```
In [29]: df.isna().any()

Out[29]: Sex               False
         Length            False
         Diameter          False
         Height            False
         Whole weight      False
         Shucked weight    False
         Viscera weight    False
         Shell weight      False
         Rings             False
         dtype: bool
```

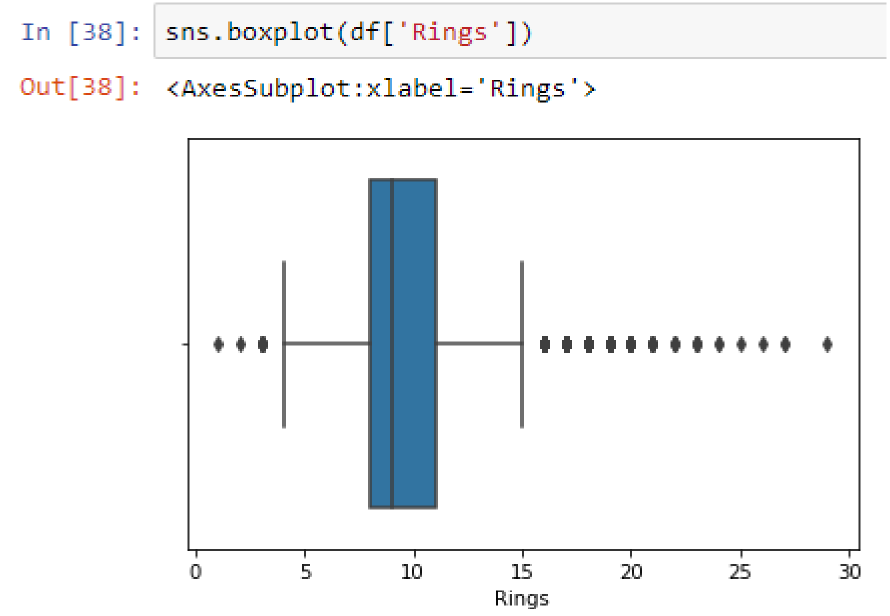
```
In [30]: df['Height'].fillna(df['Height'].mean(),inplace=True)
         df

Out[30]:
```

	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

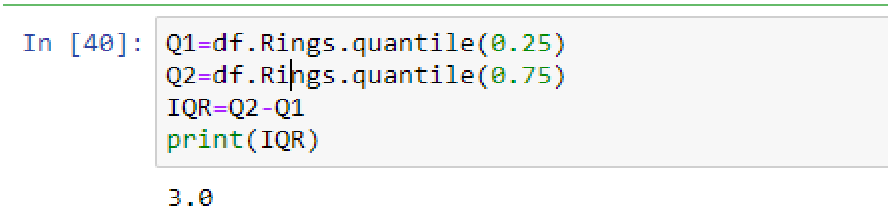
Question-6:

Find the outliers and replace the outliers

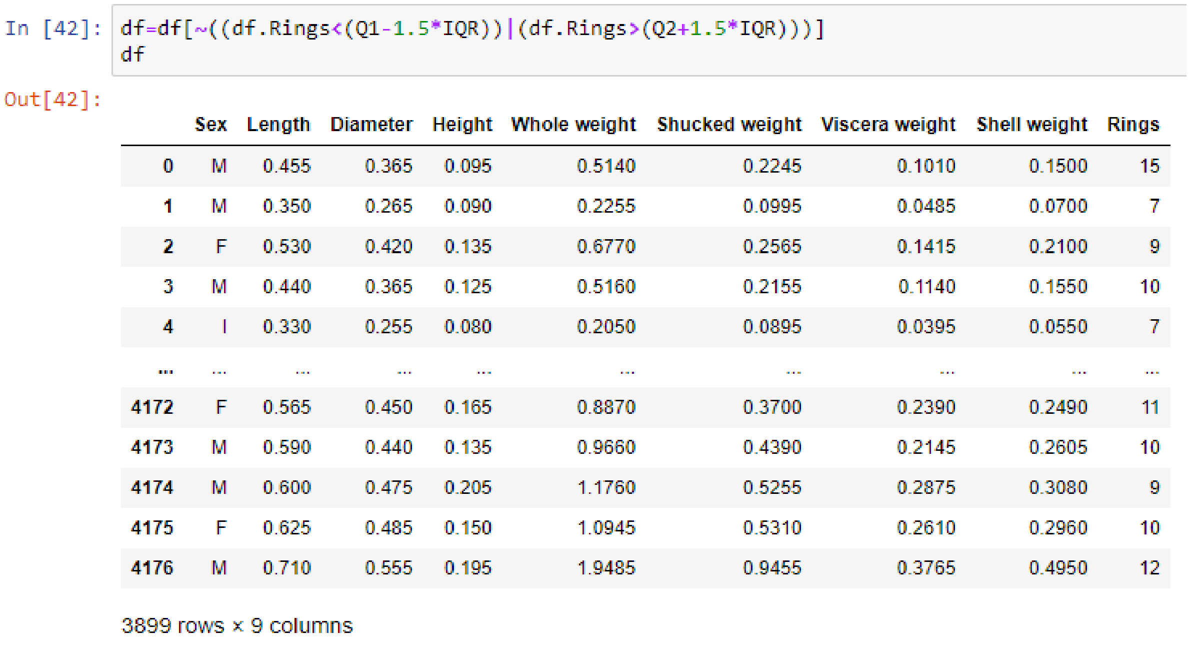
Solution:



```
In [38]: sns.boxplot(df['Rings'])
Out[38]: <AxesSubplot:xlabel='Rings'>
```



```
In [40]: Q1=df.Rings.quantile(0.25)
         Q2=df.Rings.quantile(0.75)
         IQR=Q2-Q1
         print(IQR)

         3.0
```



```
In [42]: df=df[~((df.Rings<(Q1-1.5*IQR))|(df.Rings>(Q2+1.5*IQR)))]
         df
Out[42]:
```

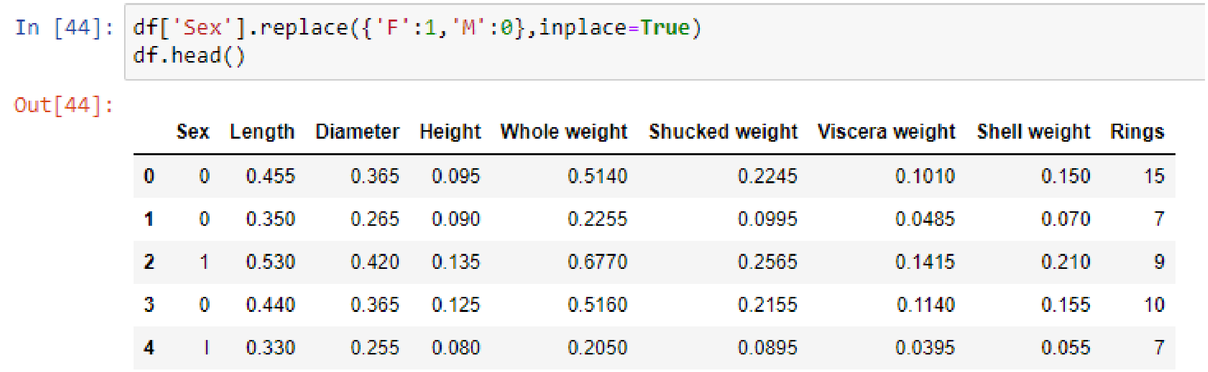
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	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

3899 rows × 9 columns

Question-7:

Check for Categorical columns and perform encoding.

Solution:



```
In [44]: df['Sex'].replace({'F':1,'M':0},inplace=True)
         df.head()
```

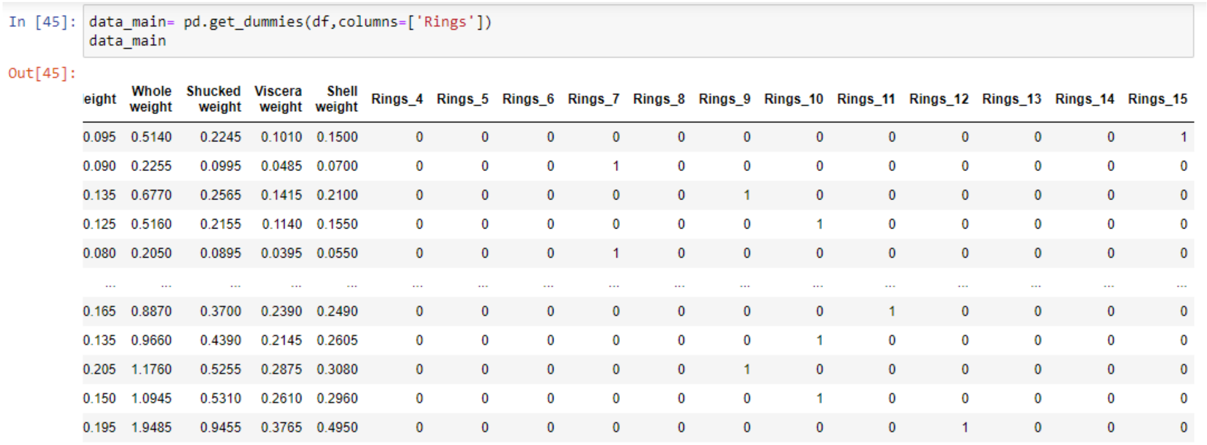
Out[44]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	0	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	1	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	0	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

Question-8:

Split the data into dependent and independent variables.

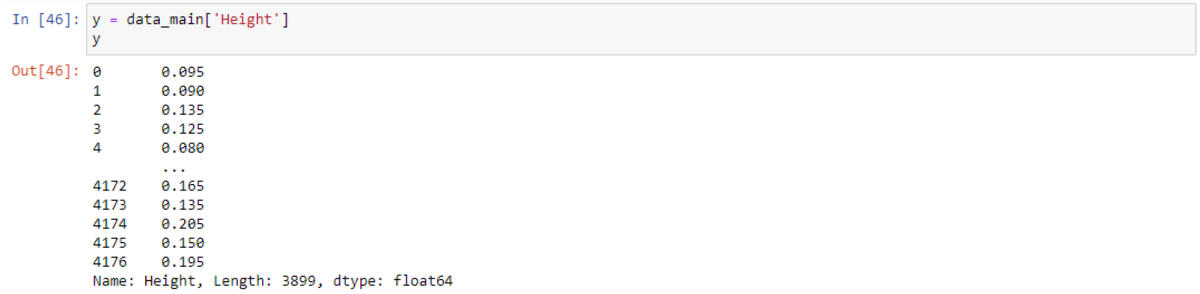
Solution:



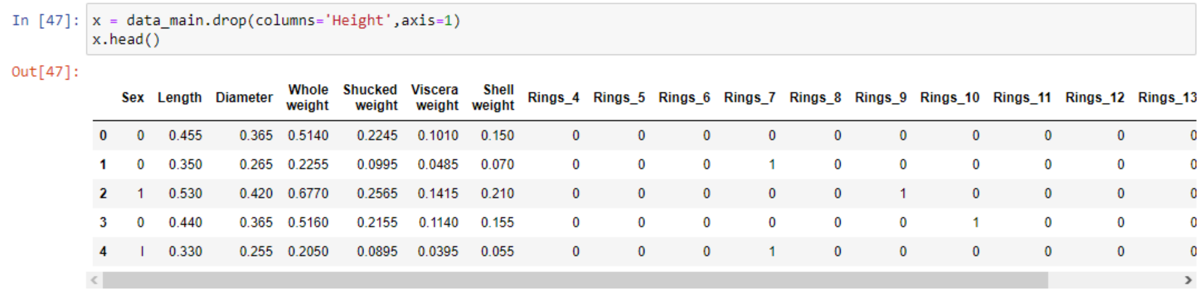
```
In [45]: data_main= pd.get_dummies(df,columns=['Rings'])
         data_main
```

Out[45]:

eight	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings_4	Rings_5	Rings_6	Rings_7	Rings_8	Rings_9	Rings_10	Rings_11	Rings_12	Rings_13	Rings_14	Rings_15
0.095	0.5140	0.2245	0.1010	0.1500	0	0	0	0	0	0	0	0	0	0	0	1
0.090	0.2255	0.0995	0.0485	0.0700	0	0	0	1	0	0	0	0	0	0	0	0
0.135	0.6770	0.2565	0.1415	0.2100	0	0	0	0	0	1	0	0	0	0	0	0
0.125	0.5160	0.2155	0.1140	0.1550	0	0	0	0	0	0	1	0	0	0	0	0
0.080	0.2050	0.0895	0.0395	0.0550	0	0	0	1	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
0.165	0.8870	0.3700	0.2390	0.2490	0	0	0	0	0	0	0	1	0	0	0	0
0.135	0.9660	0.4390	0.2145	0.2605	0	0	0	0	0	0	1	0	0	0	0	0
0.205	1.1760	0.5255	0.2875	0.3080	0	0	0	0	0	1	0	0	0	0	0	0
0.150	1.0945	0.5310	0.2610	0.2960	0	0	0	0	0	0	1	0	0	0	0	0
0.195	1.9485	0.9455	0.3765	0.4950	0	0	0	0	0	0	0	0	1	0	0	0



```
In [46]: y = data_main['Height']
         y

Out[46]: 0       0.095
         1       0.090
         2       0.135
         3       0.125
         4       0.080
                 ...
         4172    0.165
         4173    0.135
         4174    0.205
         4175    0.150
         4176    0.195
         Name: Height, Length: 3899, dtype: float64
```



```
In [47]: x = data_main.drop(columns='Height',axis=1)
         x.head()
```

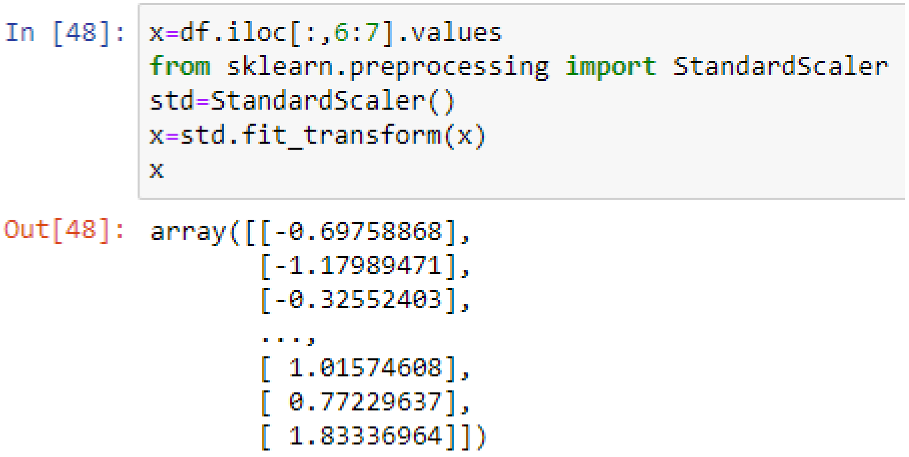
Out[47]:

	Sex	Length	Diameter	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings_4	Rings_5	Rings_6	Rings_7	Rings_8	Rings_9	Rings_10	Rings_11	Rings_12	Rings_13
0	0	0.455	0.365	0.5140	0.2245	0.1010	0.150	0	0	0	0	0	0	0	0	0	0
1	0	0.350	0.265	0.2255	0.0995	0.0485	0.070	0	0	0	1	0	0	0	0	0	0
2	1	0.530	0.420	0.6770	0.2565	0.1415	0.210	0	0	0	0	0	1	0	0	0	0
3	0	0.440	0.365	0.5160	0.2155	0.1140	0.155	0	0	0	0	0	0	1	0	0	0
4	I	0.330	0.255	0.2050	0.0895	0.0395	0.055	0	0	0	1	0	0	0	0	0	0

Question-9:

Scale the independent variables

Solution:

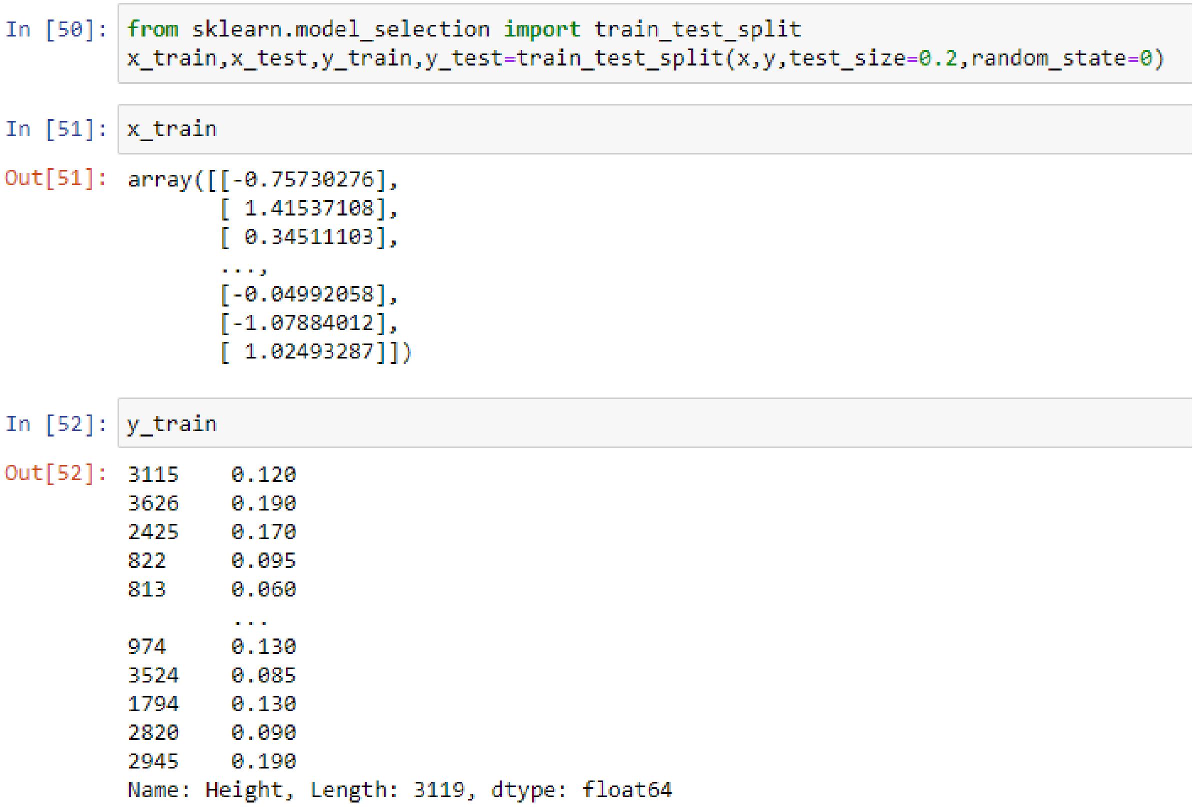


```
In [48]: x=df.iloc[:,6:7].values
         from sklearn.preprocessing import StandardScaler
         std=StandardScaler()
         x=std.fit_transform(x)
         x

Out[48]: array([[-0.69758868],
                [-1.17989471],
                [-0.32552403],
                ...,
                [ 1.01574608],
                [ 0.77229637],
                [ 1.83336964]])
```

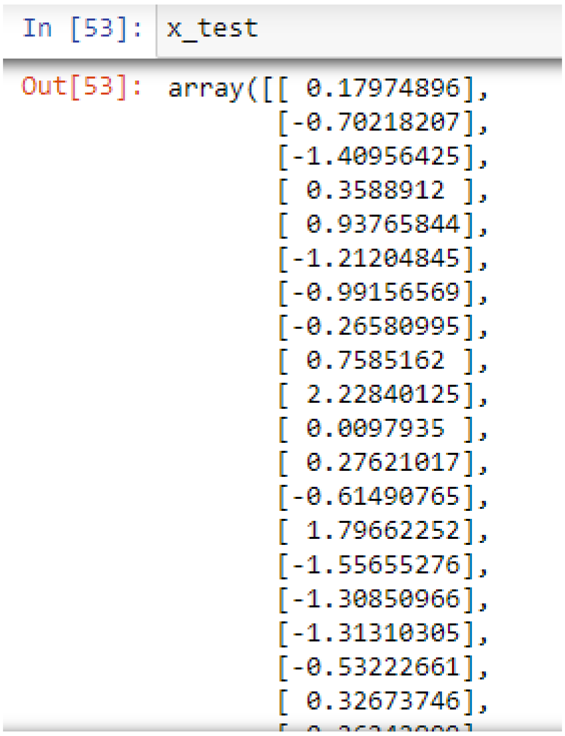
Question-10:

Split the data into training and testing

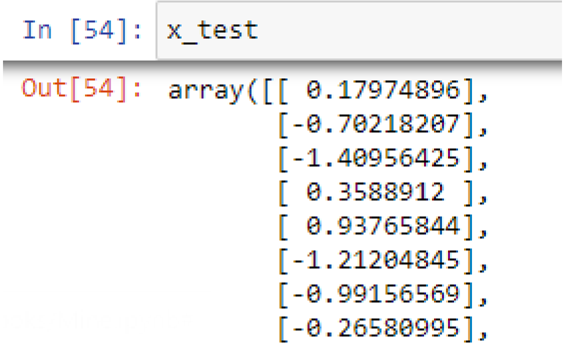
Solution:



```
In [50]: 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_state=0)

In [51]: x_train

Out[51]: array([[-0.75730276],
                [ 1.41537108],
                [ 0.34511103],
                ...,
                [-0.04992058],
                [-1.07884012],
                [ 1.02493287]])

In [52]: y_train

Out[52]: 3115    0.120
         3626    0.190
         2425    0.170
         822     0.095
         813     0.060
                 ...
         974     0.130
         3524    0.085
         1794    0.130
         2820    0.090
         2945    0.190
         Name: Height, Length: 3119, dtype: float64
```



```
In [53]: x_test

Out[53]: array([[ 0.17974896],
                [-0.70218207],
                [-1.40956425],
                [ 0.3588912 ],
                [ 0.93765844],
                [-1.21204845],
                [-0.99156569],
                [-0.26580995],
                [ 0.7585162 ],
                [ 2.22840125],
                [ 0.0097935 ],
                [ 0.27621017],
                [-0.61490765],
                [ 1.79662252],
                [-1.55655276],
                [-1.30850966],
                [-1.31310305],
                [-0.53222661],
                [ 0.32673746],
```

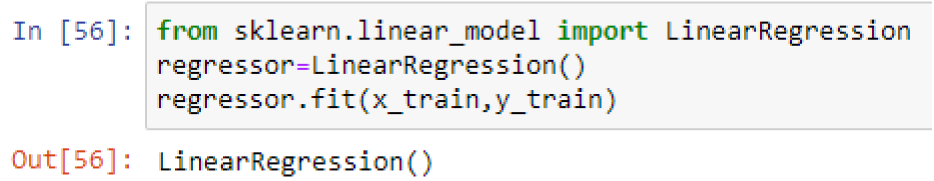


```
In [54]: x_test

Out[54]: array([[ 0.17974896],
                [-0.70218207],
                [-1.40956425],
                [ 0.3588912 ],
                [ 0.93765844],
                [-1.21204845],
                [-0.99156569],
                [-0.26580995],
```

Question-11:

Build the Model

Solution:

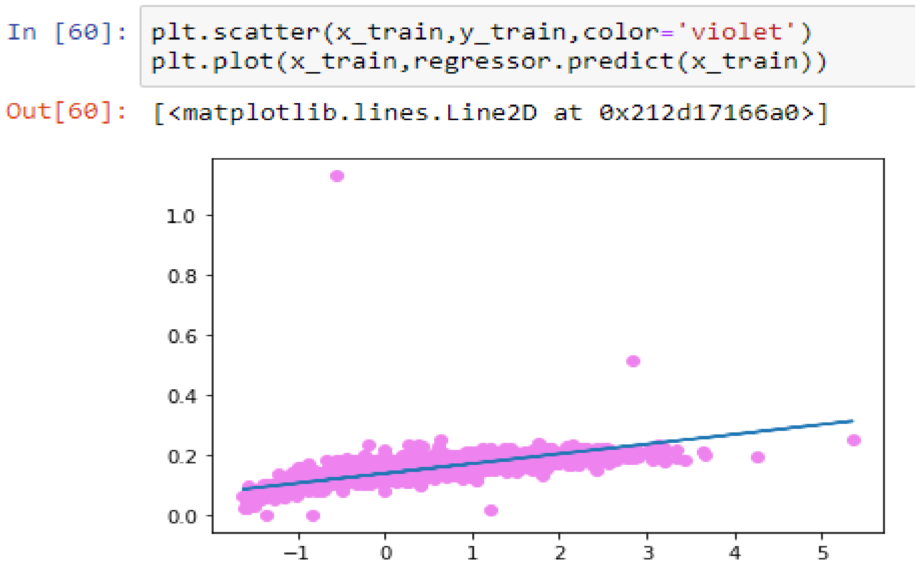


```
In [56]: from sklearn.linear_model import LinearRegression
         regressor=LinearRegression()
         regressor.fit(x_train,y_train)

Out[56]: LinearRegression()
```

Question-12:

Train the Model

Solution:

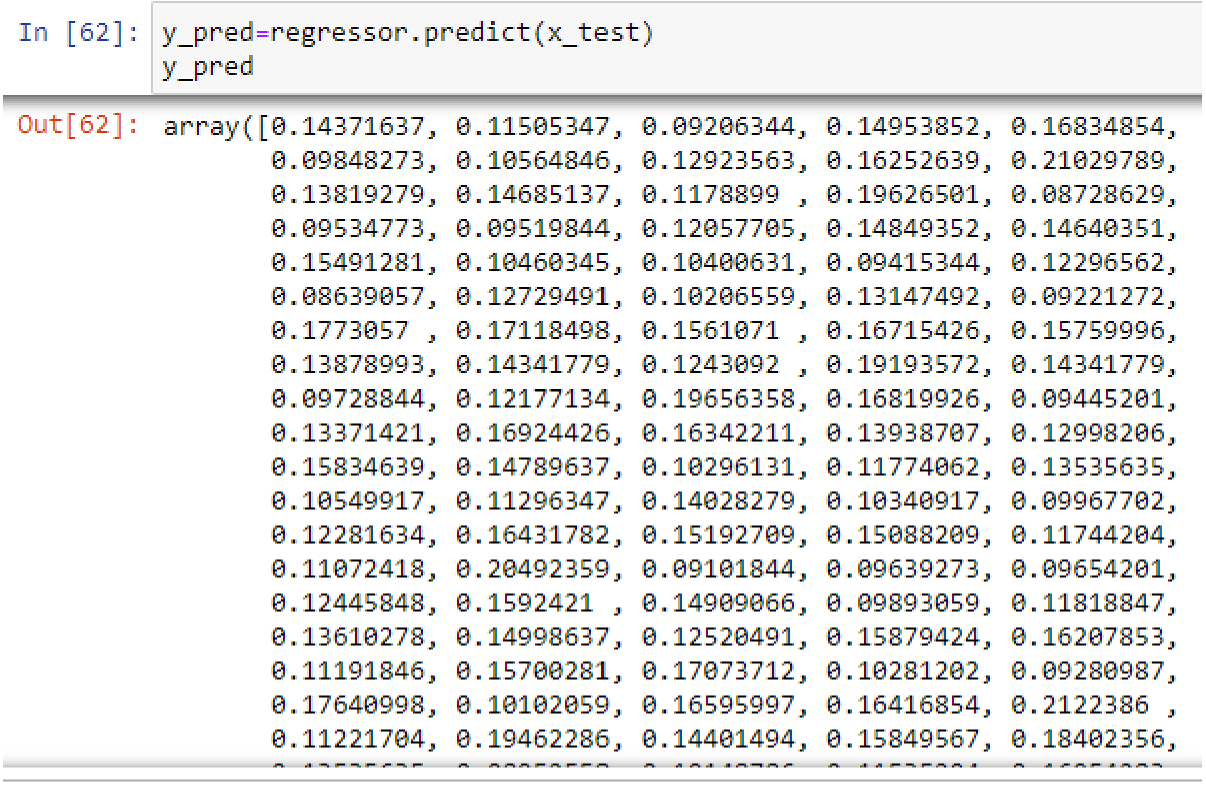


```
In [60]: plt.scatter(x_train,y_train,color='violet')
         plt.plot(x_train,regressor.predict(x_train))

Out[60]: [<matplotlib.lines.Line2D at 0x212d17166a0>]
```

Question-13:

Test the Model

Solution:

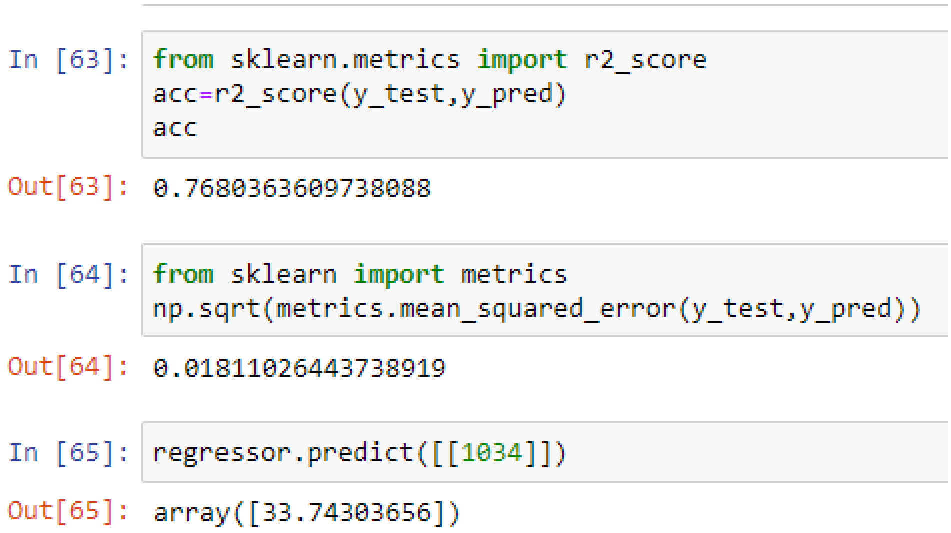


```
In [62]: y_pred=regressor.predict(x_test)
         y_pred

Out[62]: array([0.14371637, 0.11505347, 0.09206344, 0.14953852, 0.16834854,
                0.09848273, 0.10564846, 0.12923563, 0.16252639, 0.21029789,
                0.13819279, 0.14685137, 0.1178899 , 0.19626501, 0.08728629,
                0.09534773, 0.09519844, 0.12057705, 0.14849352, 0.14640351,
                0.15491281, 0.10460345, 0.10400631, 0.09415344, 0.12296562,
                0.08639057, 0.12729491, 0.10206559, 0.13147492, 0.09221272,
                0.1773057 , 0.17118498, 0.1561071 , 0.16715426, 0.15759996,
                0.13878993, 0.14341779, 0.1243092 , 0.19193572, 0.14341779,
                0.09728844, 0.12177134, 0.19656358, 0.16819926, 0.09445201,
                0.13371421, 0.16924426, 0.16342211, 0.13938707, 0.12998206,
                0.15834639, 0.14789637, 0.10296131, 0.11774062, 0.13535635,
                0.10549917, 0.11296347, 0.14028279, 0.10340917, 0.09967702,
                0.12281634, 0.16431782, 0.15192709, 0.15088209, 0.11744204,
                0.11072418, 0.20492359, 0.09101844, 0.09639273, 0.09654201,
                0.12445848, 0.1592421 , 0.14909066, 0.09893059, 0.11818847,
                0.13610278, 0.14998637, 0.12520491, 0.15879424, 0.16207853,
                0.11191846, 0.15700281, 0.17073712, 0.10281202, 0.09280987,
                0.17640998, 0.10102059, 0.16595997, 0.16416854, 0.2122386 ,
                0.11221704, 0.19462286, 0.14401494, 0.15849567, 0.18402356,
```

Question-14:

Measure the performance using Metrics.

Solution:



```
In [63]: from sklearn.metrics import r2_score
         acc=r2_score(y_test,y_pred)
         acc

Out[63]: 0.7680363609738088

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

Out[64]: 0.01811026443738919

In [65]: regressor.predict([[1034]])

Out[65]: array([33.74303656])
```