ASSIGNMENT -4

RETAIL STORE STOCK INVENTORY ANALYTICS

TEAM ID: PNT2022TMID08352
NAME: S.Navaneethakrishnan

Answer the questions or complete the tasks:

1. LOADING THE DATASET

1. LOADING THE DATASET

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

    ## 2.load the dataset
    data = pd.read_csv('abalone.csv')
    data.head()
```

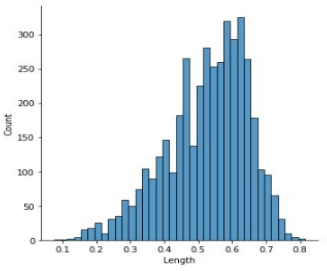
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330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

2. PERFORM THE VISUALIZATION

3.1.Univariate Analysis

```
[ ] df=pd.read_csv('abalone.csv')
    df.head()
    sns.displot(df.Length)
```

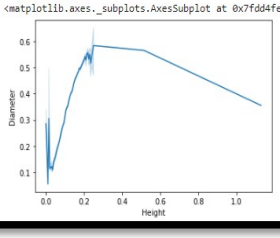
```
<seaborn.axisgrid.FacetGrid at 0x7fdd65e33b90>
```



3.2.Bivariate Analysis

```
df=pd.read_csv('abalone.csv')
df.head()
sns.lineplot(df.Height,df.Diameter)
```

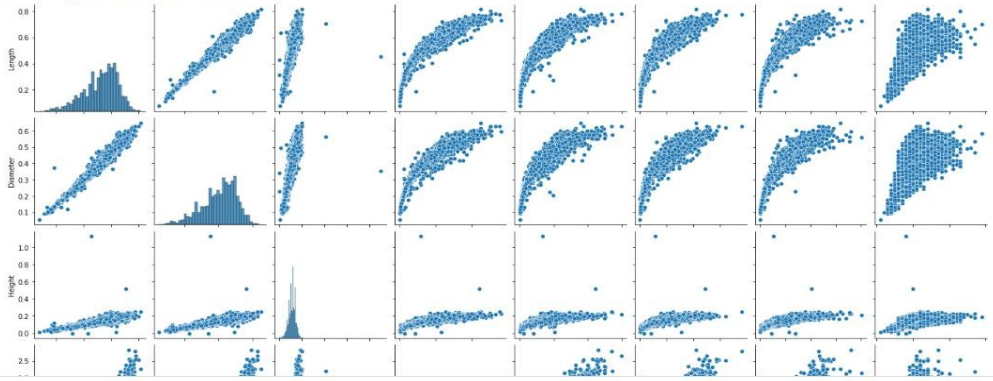
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid positional arg
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fdd4febd7d0>
```



3.3.Multi-variate Analysis

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

```
<seaborn.axisgrid.PairGrid at 0x7fdd4d138310>
```



3. PERFORM DESCRIPTIVE STATISTICS ON THE DATASET

4.PERFORM DESCRIPTIVE STATISTICS ON THE DATASET

```
[ ] df.describe()
```

	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

4. CHECK FOR MISSING VALUES AND DEAL WITH THEM

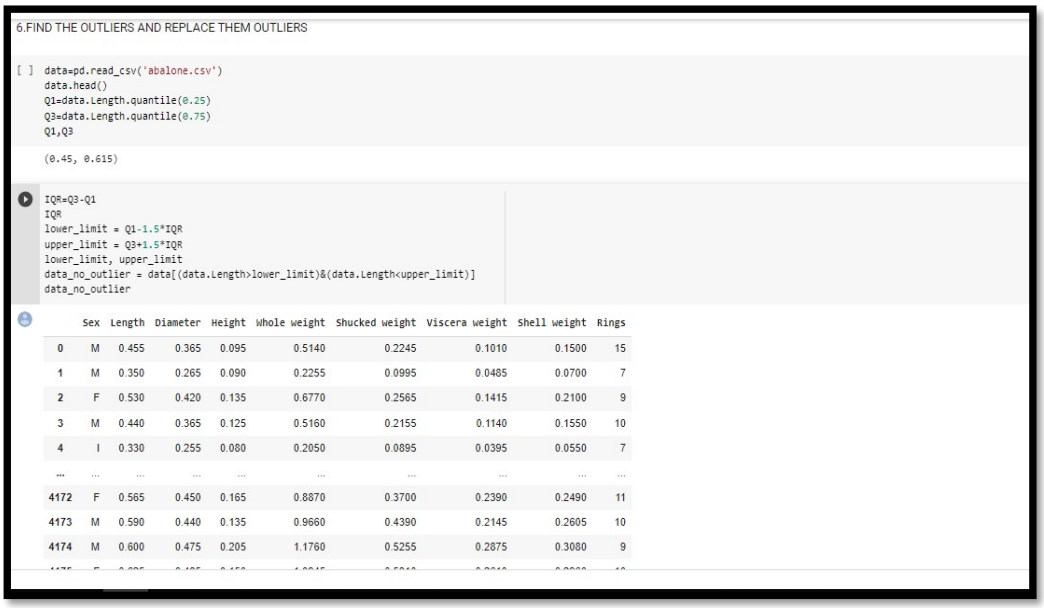
6.FIND THE OUTLIERS AND REPLACE THEM OUTLIERS

```
data=pd.read_csv('abalone.csv')
data.head()
Q1=data.Length.quantile(0.25)
Q3=data.Length.quantile(0.75)
Q1,Q3
```

```
(0.45, 0.615)
```

```
IQR=Q3-Q1
IQR
lower_limit = Q1-1.5*IQR
upper_limit = Q3+1.5*IQR
lower_limit, upper_limit
data_no_outlier = data[(data.Length>lower_limit)&(data.Length<upper_limit)]
data_no_outlier
```

5. FIND THE OUTLIERS AND REPLACE THEM OUTLIERS



6.FIND THE OUTLIERS AND REPLACE THEM OUTLIERS

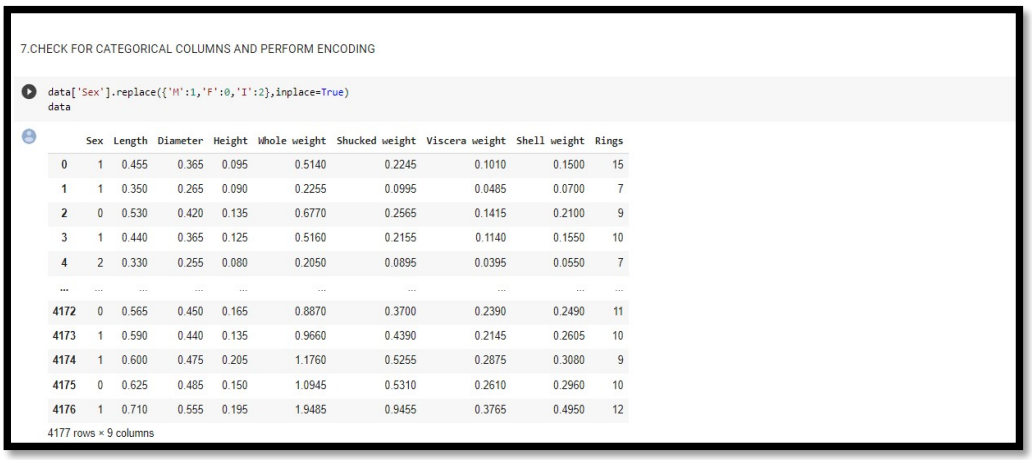
```
data=pd.read_csv('abalone.csv')
data.head()
Q1=data.Length.quantile(0.25)
Q3=data.Length.quantile(0.75)
Q1,Q3
```

```
(0.45, 0.615)
```

```
IQR=Q3-Q1
IQR
lower_limit = Q1-1.5*IQR
upper_limit = Q3+1.5*IQR
lower_limit, upper_limit
data_no_outlier = data[(data.Length>lower_limit)&(data.Length<upper_limit)]
data_no_outlier
```

	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

6. CHECK FOR CATEGORICAL COLUMNS AND PERFORM ENCODING



7.CHECK FOR CATEGORICAL COLUMNS AND PERFORM ENCODING

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

	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	0.9660	0.4390	0.2145	0.2605	10
4174	1	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	0	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	1	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 9 columns

7. SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

8.SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

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

	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	0.9660	0.4390	0.2145	0.2605
4174	1	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080
4175	0	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960
4176	1	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950

4177 rows × 8 columns

```
[ ] y
```

```
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

9.SCALE THE INDEPENDENT VARIABLES

```
from sklearn.preprocessing import scale
x = scale(x)
x
```

```
array([[-0.0105225 , -0.57455813, -0.43214879, ..., -0.60768536,
        -0.72621157, -0.63821689],
       [-0.0105225 , -1.44898585, -1.439929  , ..., -1.17090984,
        -1.20522124, -1.21298732],
       [-1.26630752,  0.05003309,  0.12213032, ..., -0.4634999 ,
        -0.35668983, -0.20713907],
       ...,
       [-0.0105225 ,  0.6329849 ,  0.67640943, ...,  0.74855917,
         0.97541324,  0.49695471],
       [-1.26630752,  0.84118198,  0.77718745, ...,  0.77334105,
         0.73362741,  0.41073914],
       [-0.0105225 ,  1.54905203,  1.48263359, ...,  2.64099341,
         1.78744868,  1.84048058]])
```

9. SPLIT THE DATA INTO TRAINING AND TESTING

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)
    print(x_train.shape, x_test.shape)
```

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

10. BUILD THE MODEL

11.BUILD THE MODEL

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

11. TRAIN THE MODEL

12.TRAIN THE MODEL

```
[ ] MLR.fit(x_train,y_train)

    LinearRegression()
```

12. TEST THE MODEL

13.TEST THE MODEL

```
y_pred=MLR.predict(x_test)
y_pred
```

```
array([14.44666767,  7.40745222, 10.78252097,  6.67673552,  8.36060517,
        9.93173721,  7.5379131 ,  8.27735969, 12.08608319, 12.29583434,
        8.29574382, 10.00474082, 10.12118388, 20.41663106, 18.40287657,
        9.01410145,  8.15149249, 11.17728349,  7.8817167 ,  9.23627153,
        7.22475815, 10.37839064,  9.63344681, 10.47074632, 11.53950746,
       10.95434036, 10.05460627,  8.66984804,  8.32478841,  8.9477472 ,
       11.28292888, 11.88502862,  7.77967711,  9.23899781,  8.48185467,
        6.59484556,  9.63756273,  9.01573937, 11.92456149,  9.52820927,
       10.14646238,  9.962085  ,  7.37530868, 11.06812272,  9.44867903,
        9.44418147,  7.34762047, 11.16558831, 13.92827424,  9.22663544,
       11.99434293, 10.43123452, 10.32767995,  6.11983576,  6.79521823,
       10.05528337,  8.53284124,  8.04305494,  9.92171955,  7.89210892,
        6.07840864,  9.9446442 ,  9.85338799,  8.68752504,  8.05979124,
       10.96333799,  7.27673389,  7.60319848,  6.64672865, 13.12749717,
        7.90163657, 12.47507082,  7.31744113,  6.50778142,  8.81198339,
        8.64057198,  6.53963896,  6.45664839,  8.75598234, 10.4890979 ,
        9.76947248,  8.90006123,  9.31761142,  9.35957542,  7.34115295,
       13.43047766,  7.39556291, 15.7252517 ,  6.65972367,  8.2367926 ,
        8.74726516, 10.15130117, 10.33317009,  8.890774  ,  7.56742459,
       11.28527425, 13.60151677, 13.60633873, 11.75750974,  9.79457312,
       10.48453644,  6.6401399 ,  8.25499507,  7.43644987,  7.91321069,
       11.3251922 ,  8.46296412,  4.64548299,  6.12181089,  8.25718621,
        7.49029784,  8.57641168,  7.99613531, 10.81766123,  9.42265379,
        9.08713187, 10.09224284,  9.50202631, 13.15222633,  7.80309981,
       10.17247184, 10.20195302, 12.61893172,  9.0512437 , 10.06718884,
```