Assignment -3

Python Programming

Team Id	PND2022TMID13322
Student Name	Gowsalya M
Maximum Marks	2 Marks

Question 1:

Download the dataset: Dataset

https://drive.google.com/file/d/1sIv-7x7CE0zAPAt0Uv-6pbO2ST2LVp5u/view?usp=sharing

Question 2:

Load the dataset.

In [26]:
```
import numpy as np
import pandas as pd
d=pd.read_csv("/abalone.csv")
d.head()
```

Out[26]:

	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

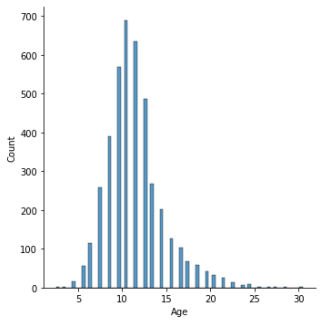
Question 3:

Perform Below Visualizations

- **Univariate Analysis**
- **Bivariate Analysis**
- **Multi-variate Analysis**

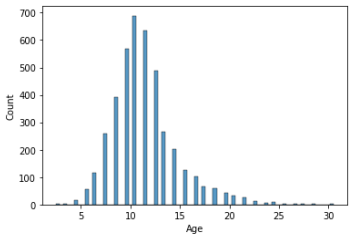
In [29]:
```
import seaborn as sns
sns.displot(df["Age"])
```

Out[29]:



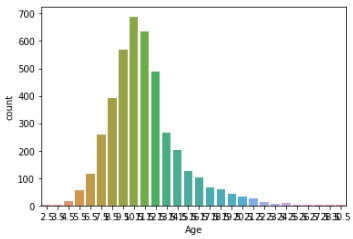
In [30]:
```
sns.histplot(x=df['Age'])
```

Out[30]:



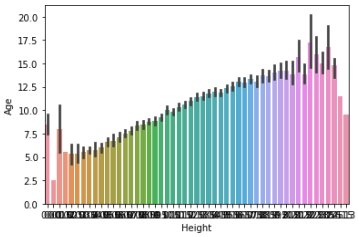
In [32]:
```
sns.countplot(x=df['Age'])
```

Out[32]:



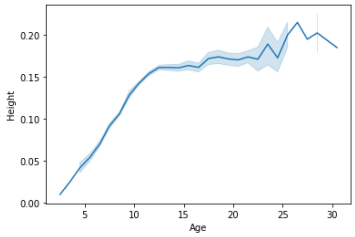
In [33]:
```
sns.barplot(x=df['Height'],y=df['Age'])
```

Out[33]:



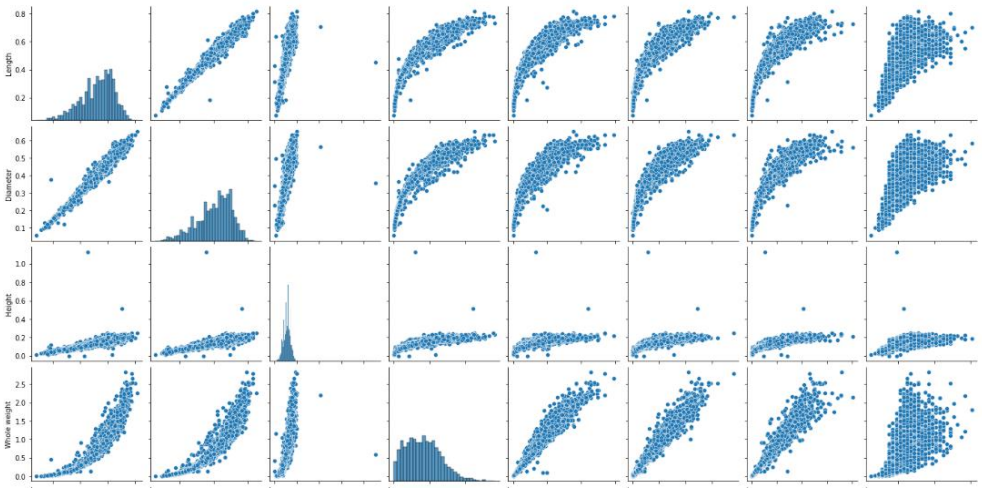
In [34]:
```
sns.lineplot(x=df['Age'],y=df['Height'])
```

Out[34]:



In [37]:
```
sns.pairplot(data=df)
```

Out[37]:



Question 4:

Perform descriptive statistics on the dataset.

In [38]:
```
df.describe()
```

Out[38]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Age
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	11.433684
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	2.500000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	9.500000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	10.500000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	12.500000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	30.500000

Question 5:

Handle the Missing values.

In [39]:
```
df.isnull().sum()
```

Out[39]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Age               0
dtype: int64
```

In [40]:
```
df.isna().any()
```

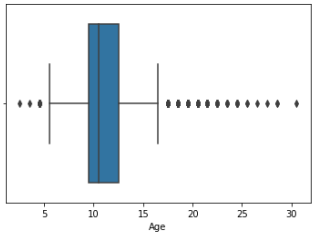
Out[40]:
```
Sex               False
Length            False
Diameter          False
Height            False
Whole weight      False
Shucked weight    False
Viscera weight    False
Shell weight      False
Age               False
dtype: bool
```

Question 6:

Find the outliers and replace the outliers

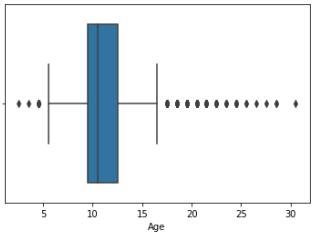
In [41]:
```
x = sns.boxplot(x=df["Age"])
x
```

Out[41]:



In [42]:
```
x = df.Age
sns.boxplot(x=x)
```

Out[42]:



In [43]:
```
x = np.where(df['Age']>57,39, df['Age'])
x
```

Out[43]:
```
array([16.5,  8.5, 10.5, ..., 10.5, 11.5, 13.5])
```

Question 7:

Check for Categorical columns and perform encoding.

In [45]:
```
import warnings
warnings.filterwarnings('ignore')
pd.Categorical(df["Whole weight"])
```

Out[45]:
```
[0.5140, 0.2255, 0.6770, 0.5160, 0.2050, ..., 0.8870, 0.9660, 1.1760, 1.0945, 1.9485]
Length: 4177
Categories (2429, float64): [0.0020, 0.0080, 0.0105, 0.0130, ..., 2.5550, 2.6570, 2.7795, 2.8255]
```

In [46]:
```
pd.get_dummies(df["Height"]).head()
```

Out[46]:

	0.000	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050	...	0.210	0.215	0.220	0.225	0.230	0.235	0.240	0.250	0.515	1.130
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 51 columns

In [47]:
```
pd.get_dummies(df).head()
```

Out[47]:

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

Question 8:

Split the data into dependent and independent variables.

In [48]:
```
X = df.iloc[:, :-1].values
X
```

Out[48]:
```
array([['M', 0.455, 0.365, ..., 0.2245, 0.101, 0.15],
       ['M', 0.35, 0.265, ..., 0.0995, 0.0485, 0.07],
       ['F', 0.53, 0.42, ..., 0.2565, 0.1415, 0.21],
       ...,
       ['M', 0.6, 0.475, ..., 0.5255, 0.2875, 0.308],
       ['F', 0.625, 0.485, ..., 0.531, 0.261, 0.296],
       ['M', 0.71, 0.555, ..., 0.9455, 0.3765, 0.495]], dtype=object)
```

In [50]:
```
Y = df.iloc[:, -1].values
Y
```

Out[50]:
```
array([16.5,  8.5, 10.5, ..., 10.5, 11.5, 13.5])
```

Question 9:

Scale the independent variables

In [52]:
```
from sklearn.preprocessing import scale
x = scale(df["Viscera weight"])
x
```

Out[52]:
```
array([-0.72621157, -1.20522124, -0.35668983, ...,  0.97541324,
        0.73362741,  1.78744868])
```

Question 10:

Split the data into training and testing

In [53]:
```
x = df.iloc[:, 1:7]
x
```

Out[53]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
0	0.455	0.365	0.095	0.5140	0.2245	0.1010
1	0.350	0.265	0.090	0.2255	0.0995	0.0485
2	0.530	0.420	0.135	0.6770	0.2565	0.1415
3	0.440	0.365	0.125	0.5160	0.2155	0.1140
4	0.330	0.255	0.080	0.2050	0.0895	0.0395
...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765

4177 rows × 6 columns

In [54]:
```
y = df.iloc[:, -1]
y
```

Out[54]:
```
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
```

In [55]:
```
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 =42)
x_train.shape
```

Out[55]:
```
(3132, 6)
```

In [56]:
```
y_test.shape
```

Out[56]:
```
(1045,)
```

In [57]:
```
x_train.head()
```

Out[57]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
3823	0.615	0.455	0.135	1.0590	0.4735	0.2630
3956	0.515	0.395	0.140	0.6860	0.2810	0.1255
3623	0.660	0.530	0.175	1.5830	0.7395	0.3505
0	0.455	0.365	0.095	0.5140	0.2245	0.1010
2183	0.495	0.400	0.155	0.8085	0.2345	0.1155

In [58]:
```
x_test.head()
```

Out[58]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
866	0.605	0.455	0.160	1.1035	0.4210	0.3015
1483	0.590	0.440	0.150	0.8725	0.3870	0.2150
599	0.560	0.445	0.195	0.9810	0.3050	0.2245
1702	0.635	0.490	0.170	1.2615	0.5385	0.2665
670	0.475	0.385	0.145	0.6175	0.2350	0.1080

In [59]:
```
y_train.head()
```

Out[59]:
```
3823    10.5
3956    13.5
3623    11.5
0       16.5
```

Question 11:

Build the model

In [61]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[61]:
```
LinearRegression()
```

Question 12:

Train the model

In [62]:
```
Y_predict_train = model.predict(x_train)
Y_predict_train
```

Out[62]:
```
array([11.25888828, 11.95379472, 12.33692259, ..., 11.12903068,
       10.71152746, 11.59516371])
```

Question 13:

Test the model

In [63]:
```
y_predict = model.predict(x_test)
y_predict
```

Out[63]:
```
array([13.0478407 , 11.43166184, 15.59825921, ..., 13.69440346,
       11.79279231, 10.83037939])
```

In [64]:
```
import math
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_predict))
print(math.sqrt(mean_squared_error(y_test, y_predict)))
```

```
4.862459933051861
2.205098622069285 4
```