1.The Dataset was successfully Downloaded

Import Libraries

In [12]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
from sklearn.model_selection  import train_test_split
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
```

2.Load the dataset

In [14]:
```python
df=pd.read_csv("D:/csv/abalone.csv")
```

In [16]:
```python
df.head()
```

Out[16]:

	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

In [17]:
```python
df.tail()
```

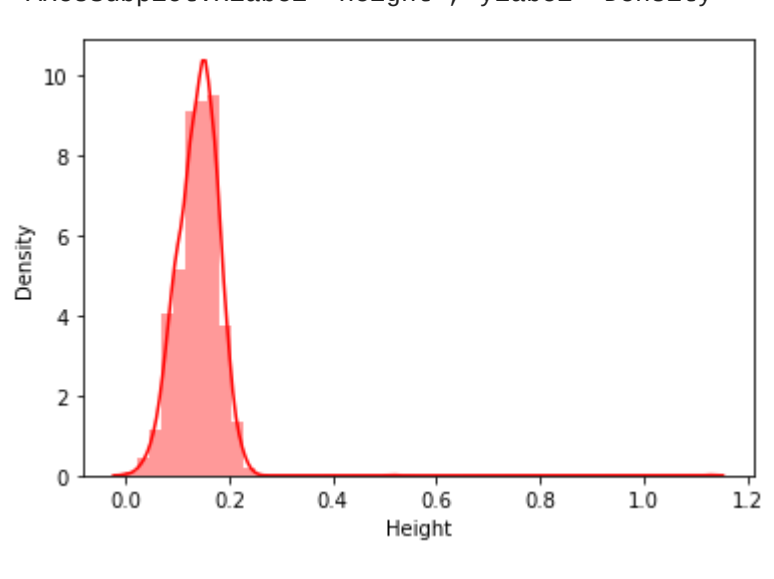
Out[17]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
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

In [19]:
```python
sns.distplot(df['Height'],color="red")
```

```
C:\Users\sss\anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future ver
sion. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

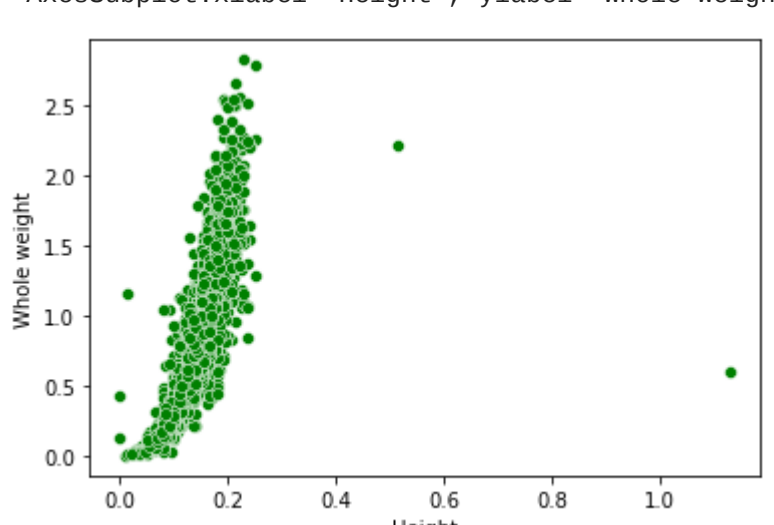
Out[19]: <AxesSubplot:xlabel='Height', ylabel='Density'>



In [28]:
```python
sns.scatterplot(df['Height'],df['Whole weight'],color="green")
```

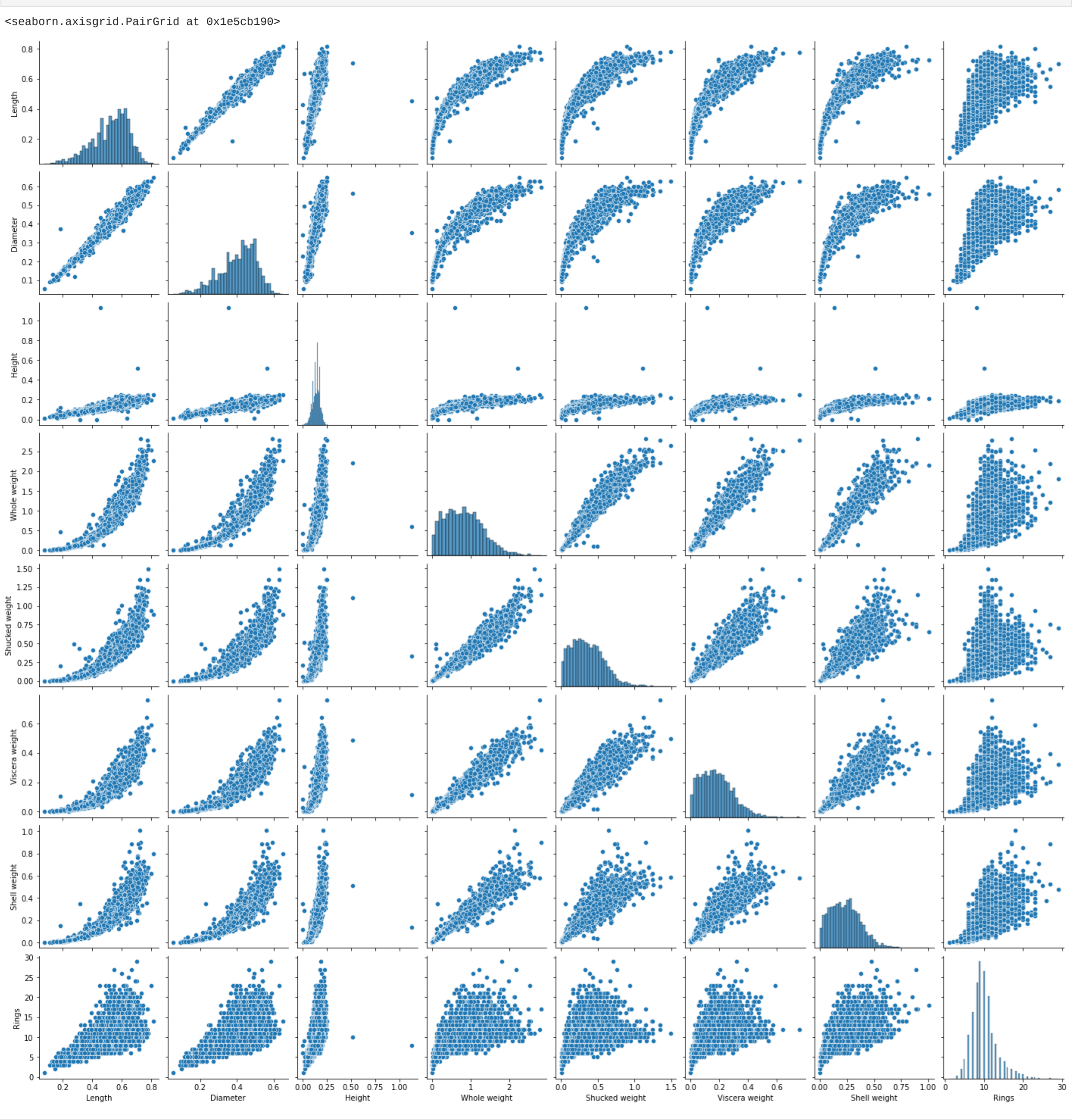
```
C:\Users\sss\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[28]: <AxesSubplot:xlabel='Height', ylabel='Whole weight'>



In [29]:
```python
sns.pairplot(df)
```

Out[29]: <seaborn.axisgrid.PairGrid at 0x1e5cb190>



In [30]:
```python
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: 277.4+ KB
```

In [31]:
```python
df.mean()
```

Out[31]:
```
Length            0.523992
Diameter          0.407881
Height            0.139516
Whole weight      0.828742
Shucked weight    0.359367
Viscera weight    0.180594
Shell weight      0.238831
Rings             9.933684
dtype: float64
```

In [33]:
```python
df.median()
```

Out[33]:
```
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 [34]:
```python
df.mode()
```

Out[34]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.550	0.45	0.15	0.2225	0.175	0.1715	0.275	9.0
1	NaN	0.625	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [35]:
```python
df.skew()
```

Out[35]:
```
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 [37]:
```python
df.kurt()
```

Out[37]:
```
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 [38]:
```python
df.var()
```

Out[38]:
```
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 [39]:
```python
df.std()
```

Out[39]:
```
Length            0.120093
Diameter          0.099240
Height            0.041827
Whole weight      0.490389
Shucked weight    0.221963
Viscera weight    0.109614
Shell weight      0.139203
Rings             3.224169
dtype: float64
```

In [40]:
```python
df.max()
```

Out[40]:
```
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
```

In [41]:
```python
df.min()
```

Out[41]:
```
Sex                    F
Length             0.075
Diameter           0.055
Height               0.0
Whole weight       0.002
Shucked weight     0.001
Viscera weight    0.0005
Shell weight      0.0015
Rings                  1
dtype: object
```

In [42]:
```python
df.describe()
```

Out[42]:

	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 [43]:
```python
df.isna().any()
```

Out[43]:
```
Sex               False
Length            False
Diameter          False
Height            False
Whole weight      False
Shucked weight    False
Viscera weight    False
Shell weight      False
Rings             False
dtype: bool
```

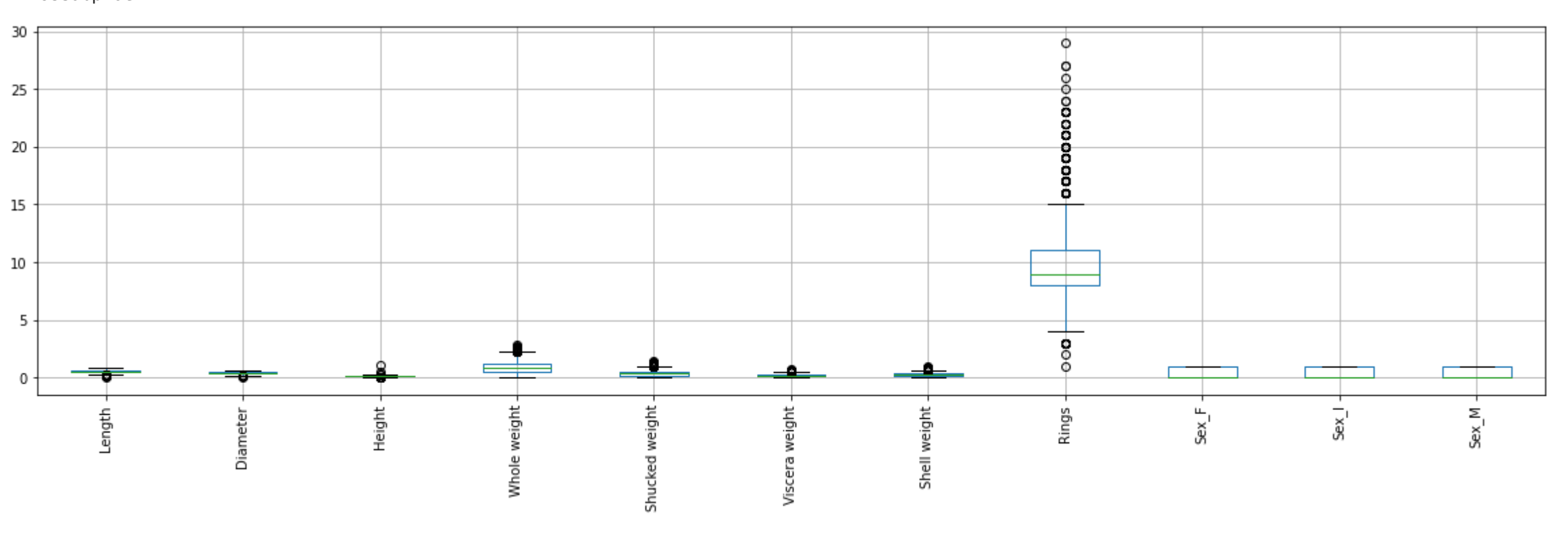
In [44]:
```python
df.isna().sum()
```

Out[44]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

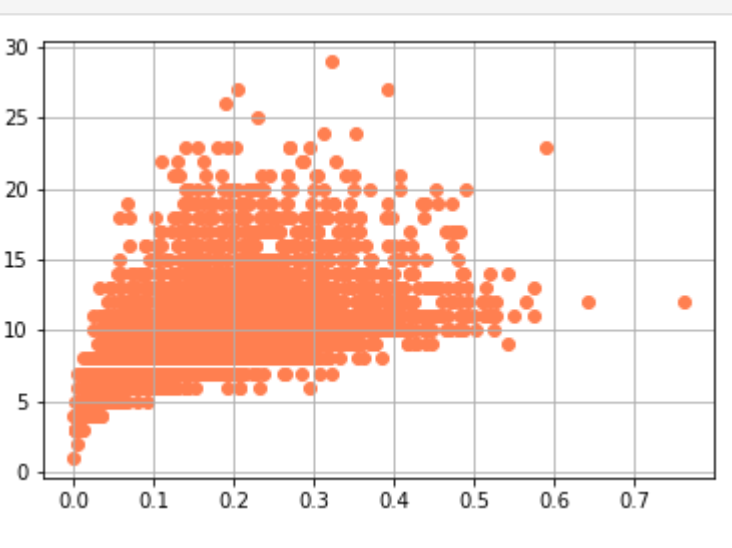
In [46]:
```python
df=pd.get_dummies(df)
dummy_df=df.copy()
```

In [47]:
```python
df.boxplot(rot=90,figsize=(20,5))
```

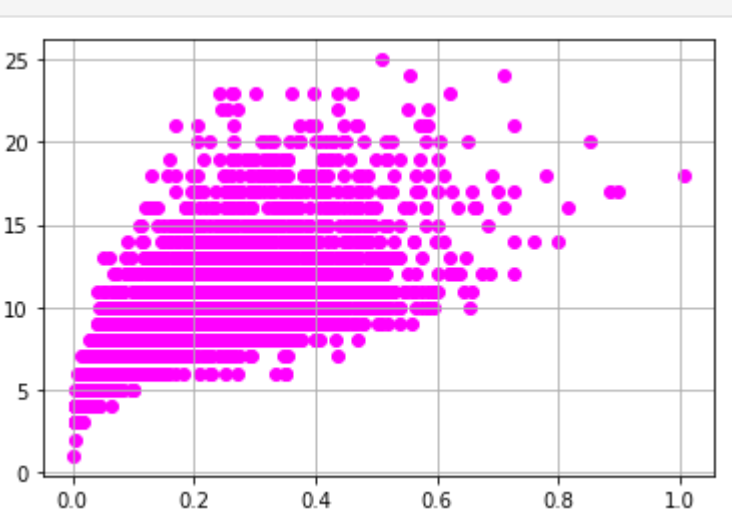
Out[47]: <AxesSubplot:>



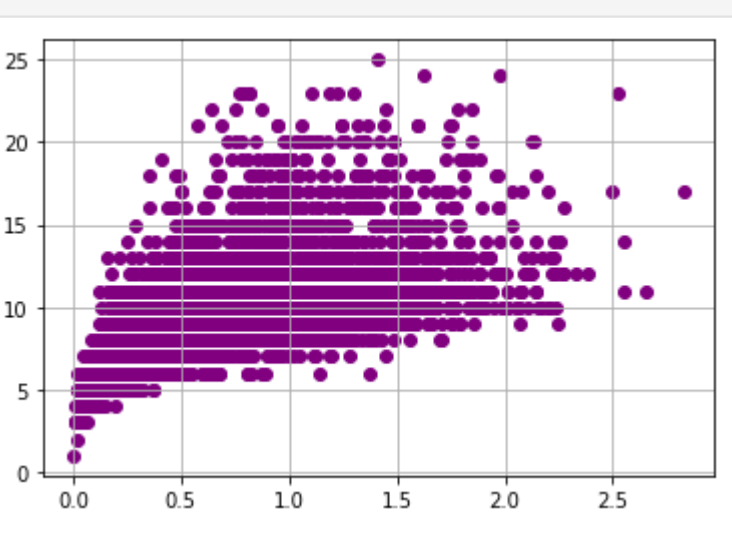
In [50]:
```python
var ='Viscera weight'
plt.scatter(x= df[var],y= df['Rings'],color="coral")
plt.grid(True)
```



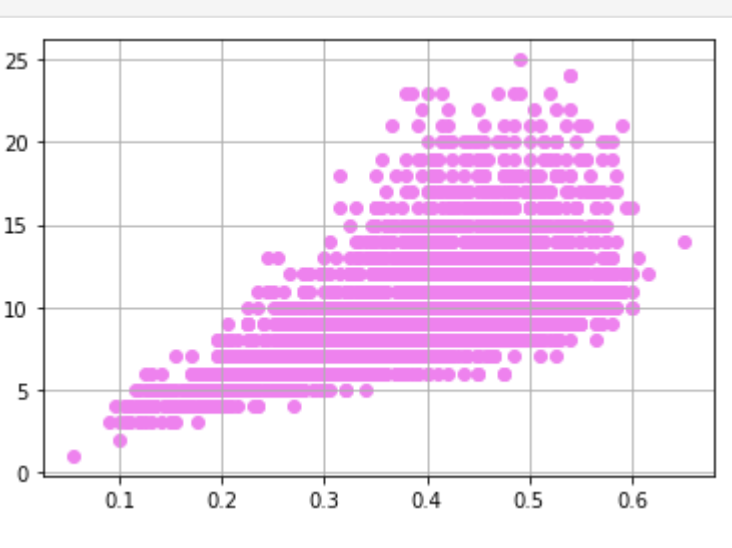
In [53]:
```python
df.drop(df[(df['Viscera weight']> 0.5) & (df['Rings'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight']<0.5) & (df['Rings'] > 25)].index, inplace=True)
var = 'Shell weight'
plt.scatter(x = df[var], y = df['Rings'],color="magenta")
plt.grid(True)
```



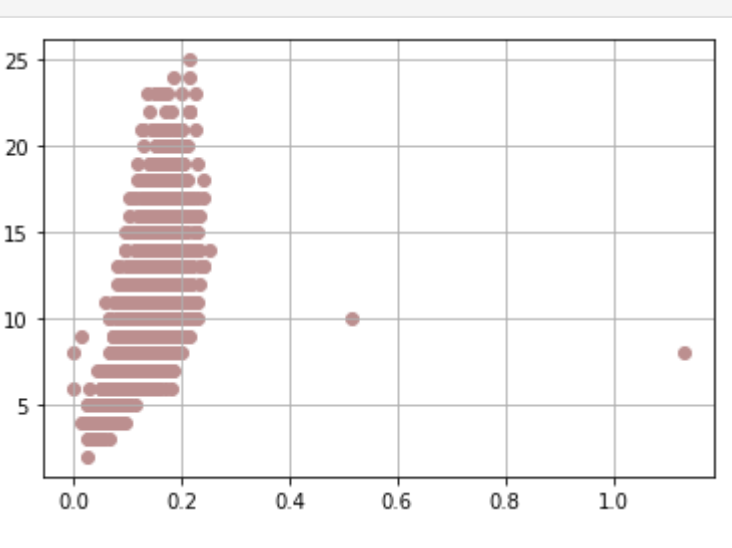
In [54]:
```python
var = 'Whole weight'
plt.scatter(x = df[var], y = df['Rings'],color="purple")
plt.grid(True)
```



In [57]:
```python
df.drop(df[(df['Whole weight']>= 2.5) & (df['Rings'] < 25)].index, inplace=True)
df.drop(df[(df['Whole weight']<2.5) & (df['Rings'] > 25)].index, inplace=True)
var = 'Diameter'
plt.scatter(x = df[var], y = df['Rings'],color="violet")
plt.grid(True)
```



In [58]:
```python
df.drop(df[(df['Diameter']<0.1) & (df['Rings'] < 5)].index, inplace=True)
df.drop(df[(df['Diameter']<0.6) & (df['Rings'] > 25)].index, inplace=True)
df.drop(df[(df['Diameter']>=0.6) & (df['Rings']< 25)].index, inplace=True)
var = 'Height'
plt.scatter(x = df[var], y = df['Rings'],color="rosybrown")
plt.grid(True)
```



In [59]:
```python
df.drop(df[(df['Height']>0.4) & (df['Rings'] < 15)].index, inplace=True)
df.drop(df[(df['Height']<0.4) & (df['Rings'] > 25)].index, inplace=True)
x = pd.get_dummies(df)
x.head()
```

Out[59]:

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

In [62]:
```python
x = df.iloc[:,0:10]
y = df.iloc[:,10]

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

```
(4132, 10)
(4132,)
```

In [66]:
```python
from sklearn.preprocessing import StandardScaler
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)
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[66]:

	0	1	2	3	4	5	6	7	8	9
0	1.665936	1.569336	0.963746	1.924895	1.327806	1.547603	2.136692	2.240426	1.502889	-0.694820
1	0.909325	0.856351	0.438113	0.925563	1.295177	0.647487	0.397621	-0.275866	-0.665385	-0.694820
2	0.068646	0.245221	-0.087521	-0.396338	-0.499429	-0.299756	-0.158734	0.038670	-0.665385	1.439221
3	1.077461	1.161916	0.306704	0.894993	0.919941	1.118752	0.703432	0.038670	-0.665385	1.439221
4	-3.041866	-2.912285	-3.109913	-1.677128	-1.606491	-1.628723	-1.695159	-1.848549	-0.665385	1.439221

In [68]:
```python
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)
print(' x_train.shape : ',x_train.shape)
print(' y_train.shape : ',y_train.shape)
print(' x_test.shape : ',x_test.shape)
print(' y_test.shape : ',y_test.shape)
```

```
 x_train.shape :  (3099, 10)
 y_train.shape :  (3099,)
 x_test.shape :  (1033, 10)
 y_test.shape :  (1033,)
```

In [69]:
```python
from sklearn.linear_model import LinearRegression
abc = LinearRegression()
abc.fit(x_train, y_train)
```

Out[69]: LinearRegression()

In [71]:
```python
y_train_pred = abc.predict(x_train)
```

In [72]:
```python
y_test_pred = abc.predict(x_test)
```

In [73]:
```python
from sklearn.metrics import mean_absolute_error, mean_squared_error
sq = mean_squared_error(y_train, y_train_pred)
print('Mean Squared error of training set :%2f'%sq)

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

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

In [74]:
```python
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
```

In [ ]: