IMPORTING LIBRARIES

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [ ]:
```
import os
os.chdir("C:/Datasets")
```

In [ ]:
```
df = pd.read_csv('abalone.csv')
```

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

Out[ ]:

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

Out[ ]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shel weigh
count	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	0.23883
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000

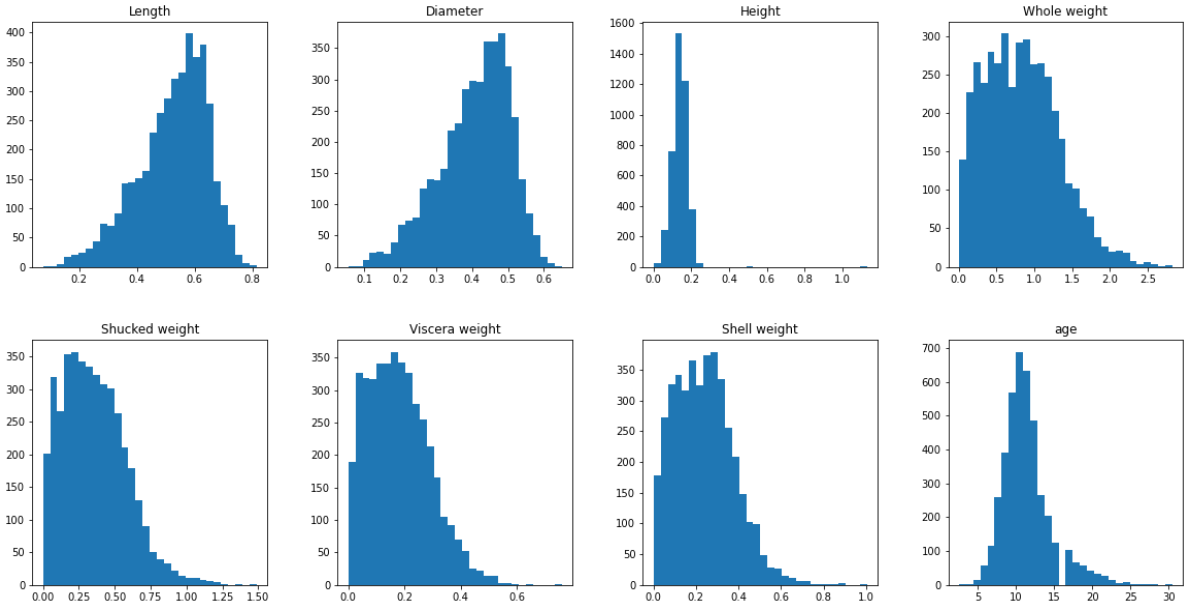
In [ ]:
```
df['age'] = df['Rings']+1.5
df = df.drop('Rings', axis = 1)
```

In [ ]:
```
from sklearn.preprocessing import  StandardScaler
from sklearn.model_selection import  train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest
from sklearn.metrics import r2_score, mean_squared_error
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

UNIVARIATE ANALYSIS

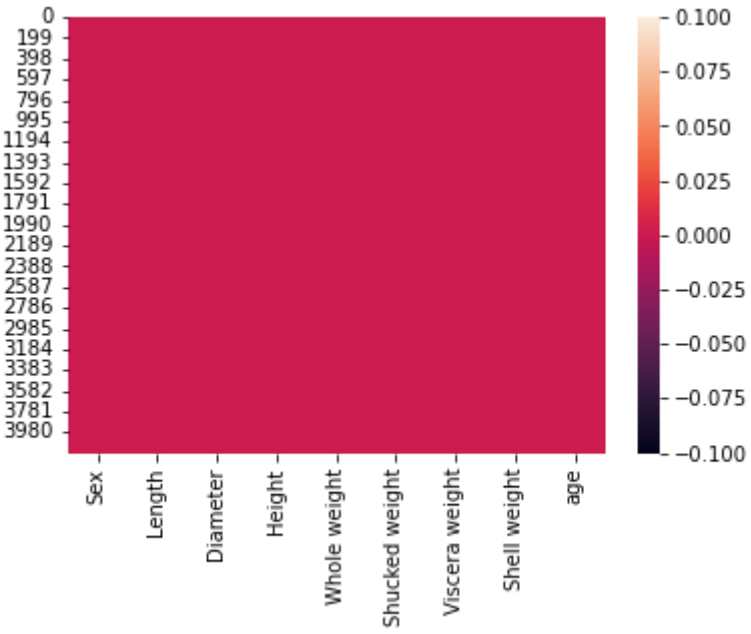
In [ ]:
```
df.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
```

Out[ ]:
```
array([[<AxesSubplot:title={'center':'Length'}>,
        <AxesSubplot:title={'center':'Diameter'}>,
        <AxesSubplot:title={'center':'Height'}>,
        <AxesSubplot:title={'center':'Whole weight'}>],
       [<AxesSubplot:title={'center':'Shucked weight'}>,
        <AxesSubplot:title={'center':'Viscera weight'}>,
        <AxesSubplot:title={'center':'Shell weight'}>,
        <AxesSubplot:title={'center':'age'}>]], dtype=object)
```



In [ ]:
```
sns.heatmap(df.isnull())
```

Out[ ]:
```
<AxesSubplot:>
```

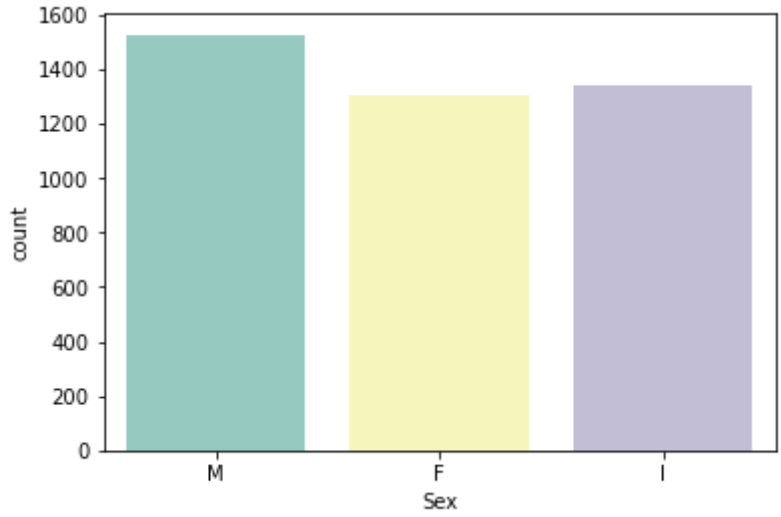


In [ ]:
```
df.info()
```

```
<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   age             4177 non-null   float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
```

In [ ]:
```
sns.countplot(x = 'Sex', data = df, palette = 'Set3')
```

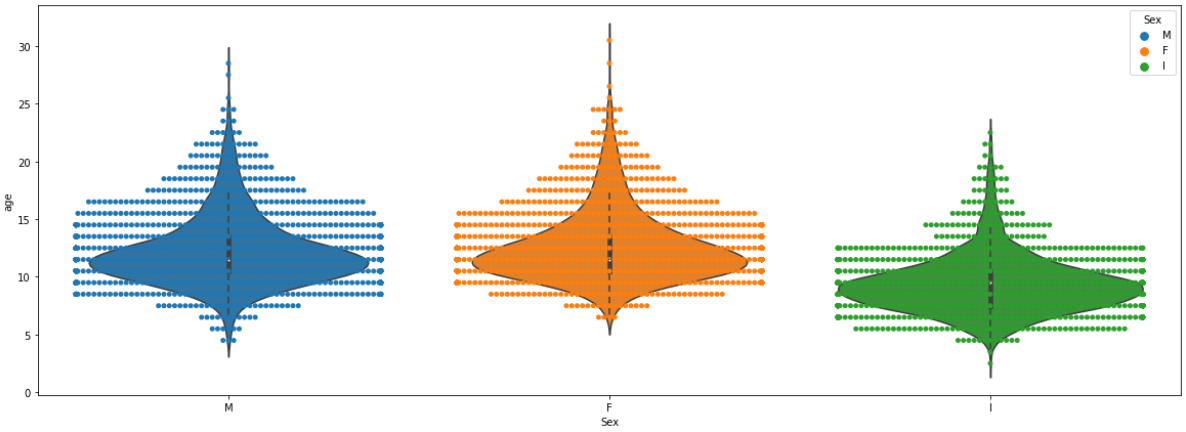
Out[ ]:
```
<AxesSubplot:xlabel='Sex', ylabel='count'>
```



In [ ]:
```
plt.figure(figsize = (20,7))
sns.swarmplot(x = 'Sex', y = 'age', data = df, hue = 'Sex')
sns.violinplot(x = 'Sex', y = 'age',data = df)
```

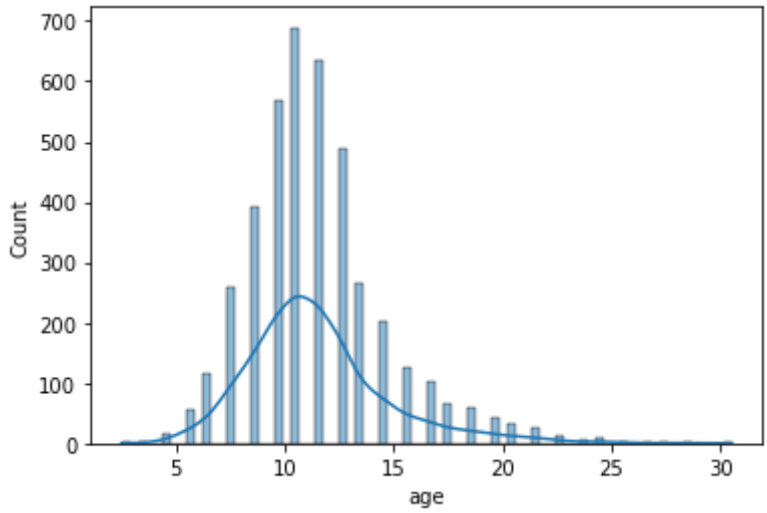
```
C:\Users\Harini S\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWar
ning: 56.2% of the points cannot be placed; you may want to decrease the size of t
he markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\Harini S\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWar
ning: 52.2% of the points cannot be placed; you may want to decrease the size of t
he markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\Harini S\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWar
ning: 58.5% of the points cannot be placed; you may want to decrease the size of t
he markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

Out[ ]:
```
<AxesSubplot:xlabel='Sex', ylabel='age'>
```



In [ ]:
```
sns.histplot(df.age,kde=True)
```

Out[ ]:
```
<AxesSubplot:xlabel='age', ylabel='Count'>
```



BIVARIATE ANALYSIS

In [ ]:
```
num_fea = df.select_dtypes(include = [np.number]).columns
ctg_fea = df.select_dtypes(include = [np.object]).columns
```

In [ ]:
```
num_fea
```

Out[ ]:
```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age'],
      dtype='object')
```

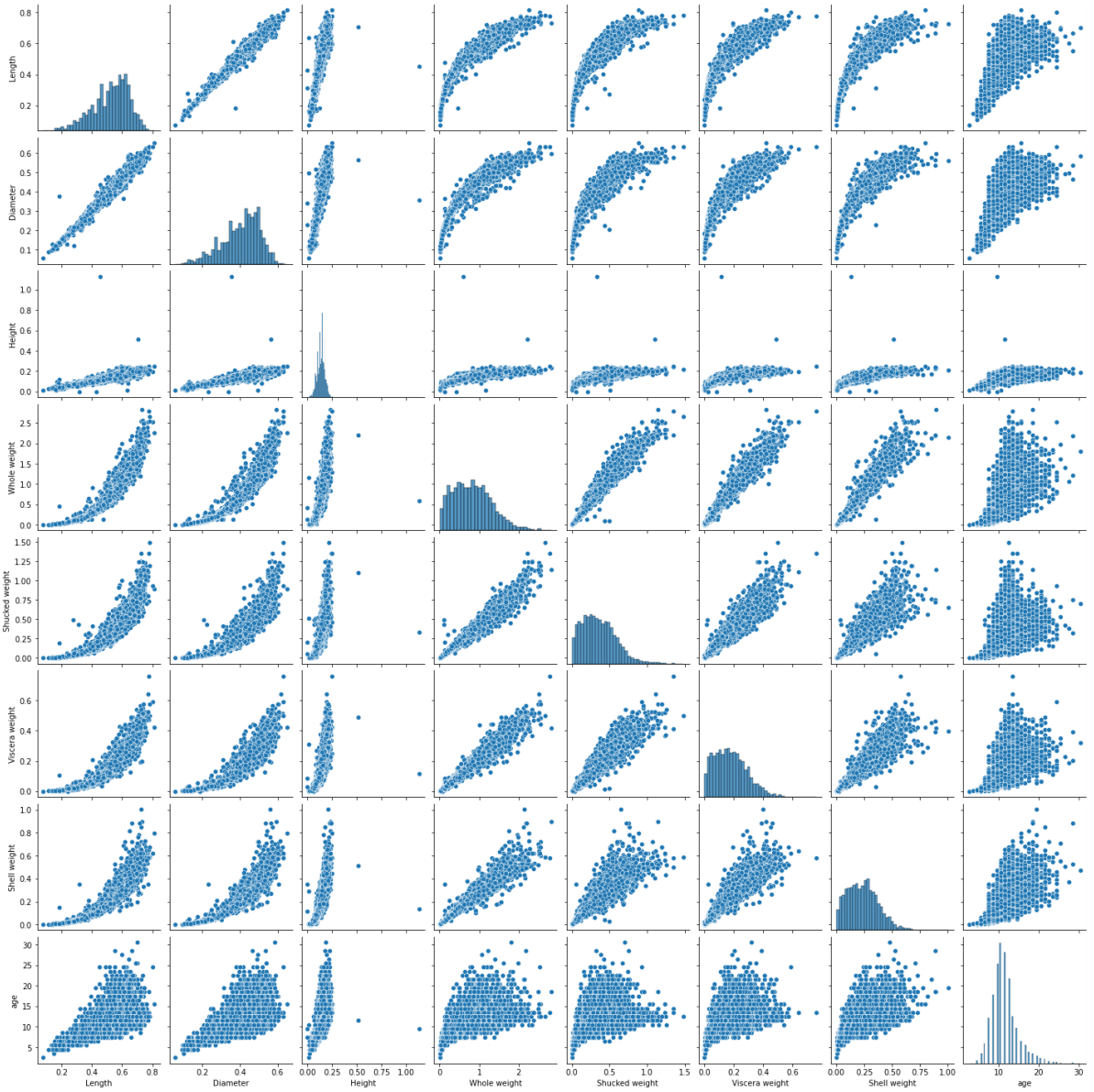
In [ ]:
```
plt.figure(figsize = (20,7))
sns.heatmap(df[num_fea].corr(),annot = True)
```

Out[ ]:
```
<AxesSubplot:>
```

MULTI VARIATE ANALYSIS

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

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x28913a93fa0>
```



MISSING VALUES

In [ ]:
```
missing_values = df.isnull().sum()
```

In [ ]:
```
missing_values
```

Out[ ]:
```
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 [ ]:
```
missing_values = df.isnull().sum().sort_values(ascending = False)
percentage_missing_values = (missing_values/len(df))*100
pd.concat([missing_values, percentage_missing_values], axis = 1, keys= ['Missing va
```

Out[ ]:

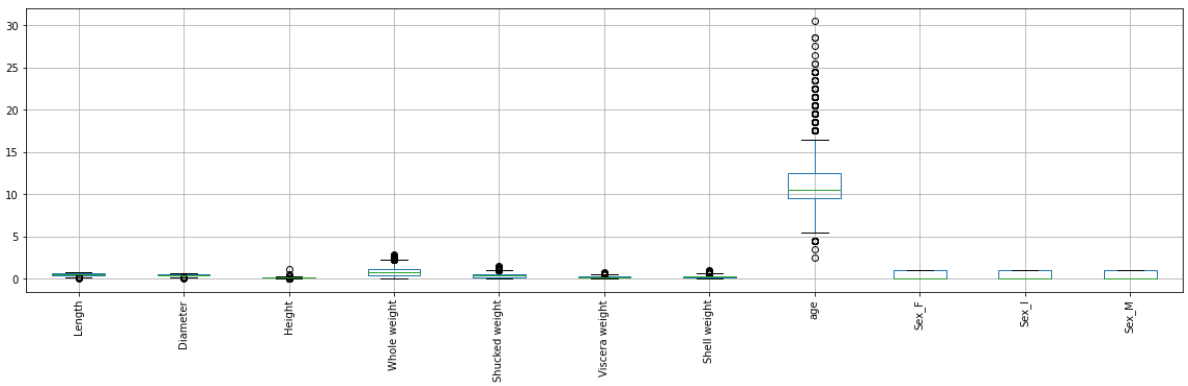
	Missing values	% Missing
Sex	0	0.0
Length	0	0.0
Diameter	0	0.0
Height	0	0.0
Whole weight	0	0.0
Shucked weight	0	0.0
Viscera weight	0	0.0
Shell weight	0	0.0
age	0	0.0

OUTLIERS

In [ ]:
```
df = pd.get_dummies(df)
dummy_data = df.copy()
```

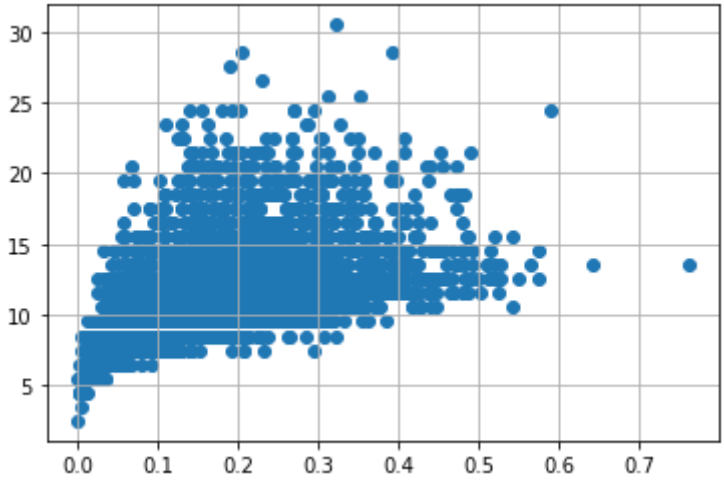
In [ ]:
```
df.boxplot( rot = 90, figsize=(20,5))
```

Out[ ]:
```
<AxesSubplot:>
```



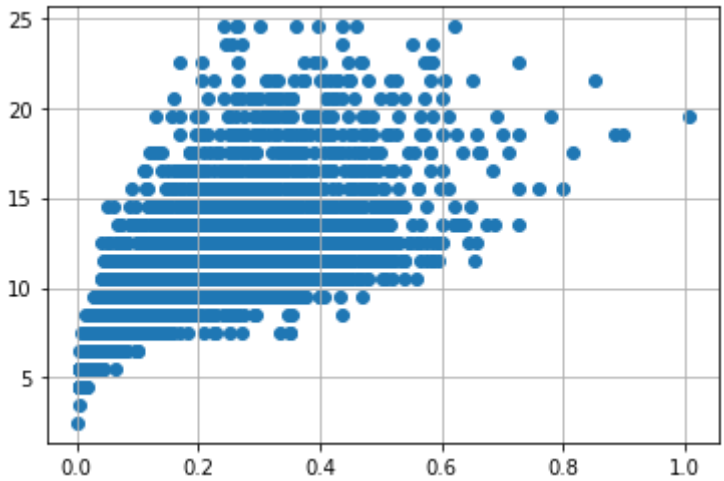
In [ ]:
```
var = 'Viscera weight'
plt.scatter(x = df[var], y = df['age'],)
```

```
plt.grid(True)
```



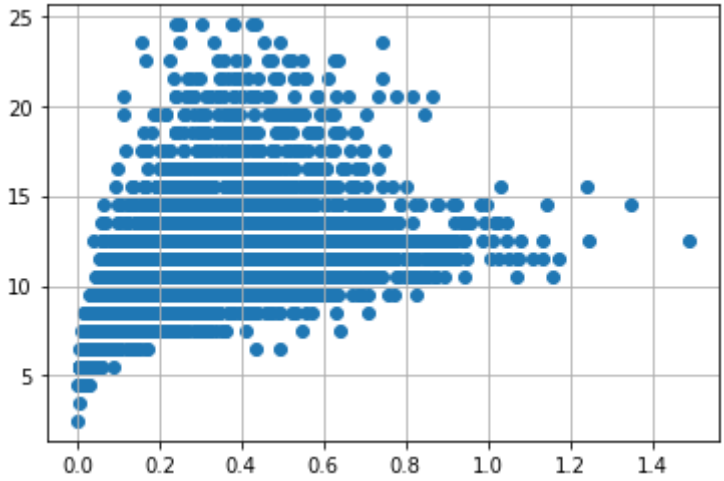
In [ ]:
```
# outliers removal
df.drop(df[(df['Viscera weight']> 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight']<0.5) & (df['age'] > 25)].index, inplace=True)
```

In [ ]:
```
var = 'Shell weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
```



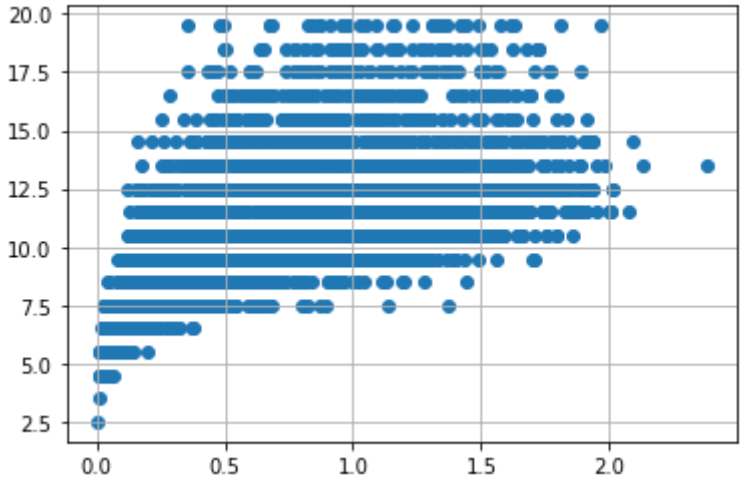
In [ ]:
```
df.drop(df[(df['Shell weight']> 0.6) & (df['age'] < 25)].index, inplace=True)
df.drop(df[(df['Shell weight']<0.8) & (df['age'] > 25)].index, inplace=True)
```

In [ ]:
```
var = 'Shucked weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
```



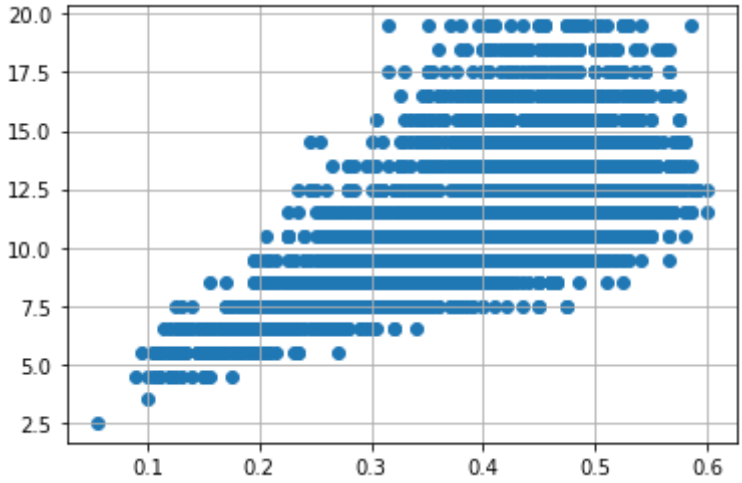
In [ ]:
```
df.drop(df[(df['Shucked weight']>= 1) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Shucked weight']<1) & (df['age'] > 20)].index, inplace=True)
```

In [ ]:
```
var = 'Whole weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
```



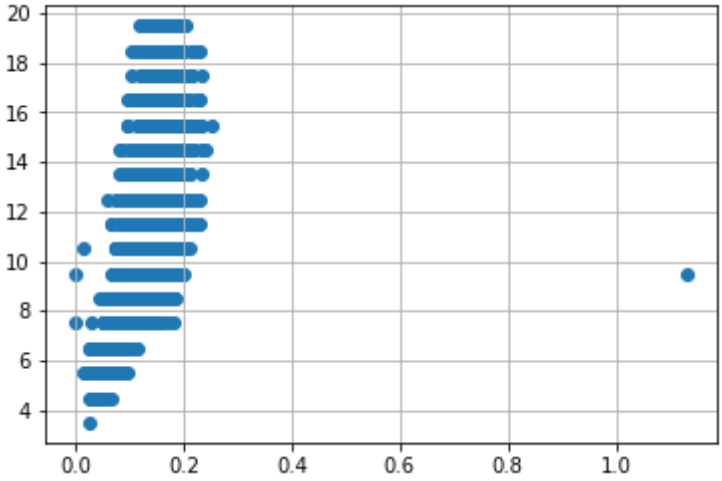
In [ ]:
```
df.drop(df[(df['Whole weight']>= 2.5) & (df['age'] < 25)].index, inplace=True)
df.drop(df[(df['Whole weight']<2.5) & (df['age'] > 25)].index, inplace=True)
```

In [ ]:
```
var = 'Diameter'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
```



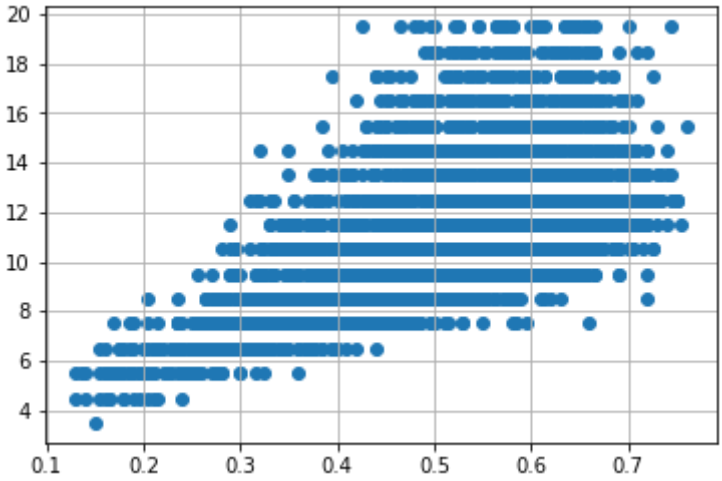
In [ ]:
```
df.drop(df[(df['Diameter']<0.1) & (df['age'] < 5)].index, inplace=True)
df.drop(df[(df['Diameter']<0.6) & (df['age'] > 25)].index, inplace=True)
df.drop(df[(df['Diameter']>=0.6) & (df['age']< 25)].index, inplace=True)
```

In [ ]:
```
var = 'Height'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
```



In [ ]:
```
df.drop(df[(df['Height']>0.4) & (df['age'] < 15)].index, inplace=True)
df.drop(df[(df['Height']<0.4) & (df['age'] > 25)].index, inplace=True)
```

In [ ]:
```
var = 'Length'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
```



In [ ]:
```
df.drop(df[(df['Length']<0.1) & (df['age'] < 5)].index, inplace=True)
df.drop(df[(df['Length']<0.8) & (df['age'] > 25)].index, inplace=True)
df.drop(df[(df['Length']>=0.8) & (df['age']< 25)].index, inplace=True)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 3995 entries, 0 to 4176
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Length          3995 non-null   float64
 1   Diameter        3995 non-null   float64
 2   Height          3995 non-null   float64
 3   Whole weight    3995 non-null   float64
 4   Shucked weight  3995 non-null   float64
 5   Viscera weight  3995 non-null   float64
 6   Shell weight    3995 non-null   float64
 7   age             3995 non-null   float64
 8   Sex_F           3995 non-null   uint8  
 9   Sex_I           3995 non-null   uint8  
 10  Sex_M           3995 non-null   uint8  
dtypes: float64(8), uint8(3)
memory usage: 292.6 KB
```

In [ ]:
```
df
```

Out[ ]:

	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	0.1500	16.5	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	8.5	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	10.5	1	0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	11.5	0	0	1
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	8.5	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	12.5	1	0	0
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	11.5	0	0	1
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	10.5	0	0	1
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	11.5	1	0	0
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	13.5	0	0	1

3995 rows × 11 columns

CATEGORICAL COLUMNS

In [ ]:
```
cat
```

Out[ ]:
```
pandas.core.arrays.categorical.Categorical
```

In [ ]:
```
num_fea = df.select_dtypes(include = [np.number]).columns
ctg_fea = df.select_dtypes(include = [np.object]).columns
```

In [ ]:
```
num_fea
```

Out[ ]:
```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I', 'Sex_M'],
      dtype='object')
```

In [ ]:
```
df_numeric = df[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
```

In [ ]:
```
df_numeric.head()
```

Out[ ]:

	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

In [ ]:
```
ctg_fea
```

Out[ ]:
```
Index([], dtype='object')
```

INDEPENDENT AND DEPENDENT VARIABLE

In [ ]:
```
x = df.iloc[:, 0:1].values
```

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

In [ ]:
```
x
```

Out[ ]:
```
array([[0.455],
       [0.35 ],
       [0.53 ],
       ...,
       [0.6  ],
       [0.625],
       [0.71 ]])
```

In [ ]:
```
y
```

Out[ ]:
```
0       0.365
1       0.265
2       0.420
3       0.365
4       0.255
        ...
4172    0.450
4173    0.440
4174    0.475
4175    0.485
4176    0.555
Name: Diameter, Length: 3995, dtype: float64
```

SCALING THE INDEPENDENT VARIABLE

In [ ]:
```
print ("\n ORIGIONAL VALUES: \n\n", x,y)
```

```
 ORIGIONAL VALUES:

 [[0.455]
 [0.35 ]
 [0.53 ]
 ...
 [0.6  ]
 [0.625]
 [0.71 ]] 0       0.365
1       0.265
2       0.420
3       0.365
4       0.255
        ...
4172    0.450
4173    0.440
4174    0.475
4175    0.485
4176    0.555
Name: Diameter, Length: 3995, dtype: float64
```

In [ ]:
```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))
new_y= min_max_scaler.fit_transform(x,y)
print ("\n VALUES AFTER MIN MAX SCALING: \n\n", new_y)
```

```
 VALUES AFTER MIN MAX SCALING:

 [[0.51587302]
 [0.34920635]
 [0.63492063]
 ...
 [0.74603175]
 [0.78571429]
 [0.92063492]]
```

SPLITING THE DATA

In [ ]:
```
X = df.drop('age', axis = 1)
y = df['age']
```

In [ ]:
```
standardScale = StandardScaler()
standardScale.fit_transform(X)

selectkBest = SelectKBest()
X_new = selectkBest.fit_transform(X, y)

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size = 0.25)
```

BUILDING MODEL

LINEAR REGRESSION

In [ ]:
```
from sklearn.linear_model import LinearRegression
```

In [ ]:
```
lm = LinearRegression()
lm.fit(X_train, y_train)
```

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

In [ ]:
```
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
```

TRAINING THE MODEL

In [ ]:
```
X_train
```

Out[ ]:
```
array([[0.35 , 0.265, 0.095, ..., 0.   , 1.   , 0.   ],
       [0.465, 0.37 , 0.12 , ..., 0.   , 1.   , 0.   ],
       [0.435, 0.335, 0.1  , ..., 0.   , 1.   , 0.   ],
       ...,
       [0.515, 0.395, 0.125, ..., 0.   , 1.   , 0.   ],
       [0.515, 0.38 , 0.12 , ..., 0.   , 1.   , 0.   ],
       [0.37 , 0.275, 0.1  , ..., 0.   , 1.   , 0.   ]])
```

In [ ]:
```
y_train
```

Out[ ]:
```
3431     6.5
1566    10.5
1559     8.5
1284    10.5
41      15.5
        ...
2896     9.5
2       10.5
1465     9.5
1290     8.5
3107     6.5
Name: age, Length: 2996, dtype: float64
```

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

```
Mean Squared error of training set :3.499447
```

TESTING THE MODEL

In [ ]:
```
X_test
```

Out[ ]:
```
array([[0.575, 0.45 , 0.16 , ..., 1.   , 0.   , 0.   ],
       [0.255, 0.195, 0.07 , ..., 0.   , 1.   , 0.   ],
       [0.41 , 0.33 , 0.105, ..., 0.   , 1.   , 0.   ],
       ...,
       [0.415, 0.325, 0.105, ..., 1.   , 0.   , 0.   ],
       [0.495, 0.385, 0.125, ..., 0.   , 1.   , 0.   ],
       [0.36 , 0.265, 0.075, ..., 0.   , 1.   , 0.   ]])
```

In [ ]:
```
y_test
```

Out[ ]:
```
1137    11.5
464      7.5
3885     8.5
80      10.5
3213    14.5
        ...
248      8.5
813      6.5
212     13.5
2299     9.5
3529     7.5
Name: age, Length: 999, dtype: float64
```

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

```
Mean Squared error of testing set :3.738376
```

In [ ]:
```
from sklearn.metrics import r2_score
s = r2_score(y_train, y_train_pred)
print('R2 Score of training set:%.2f'%s)
```

```
R2 Score of training set:0.54
```

In [ ]:
```
from sklearn.metrics import r2_score
p = r2_score(y_test, y_test_pred)
print('R2 Score of testing set:%.2f'%p)
```

```
R2 Score of testing set:0.51
```