1. Download the dataset

```
# Dataset Downloaded
```

2. Load the dataset

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

```
df = pd.read_csv('abalone.csv')
```

```
df.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

```
#Modifying the given dataset
Age=1.5+df.Rings
df["Age"]=Age
```

```
df=df.rename(columns = {'Whole weight':'Whole_weight','Shucked weight': 'Shucked_weight','Viscera weight': 'Viscera_weight',
                        'Shell weight': 'Shell_weight'})
df=df.drop(columns=["Rings"],axis=1)
df.head()
```

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

```
df.tail()
```

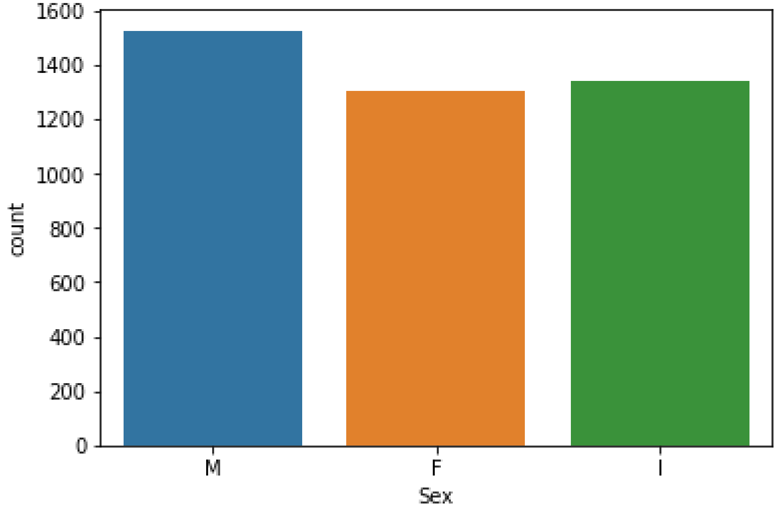
	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	12.5
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	11.5
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	10.5
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	11.5
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	13.5

3. Perform Below Visualizations

Univariate Analysis

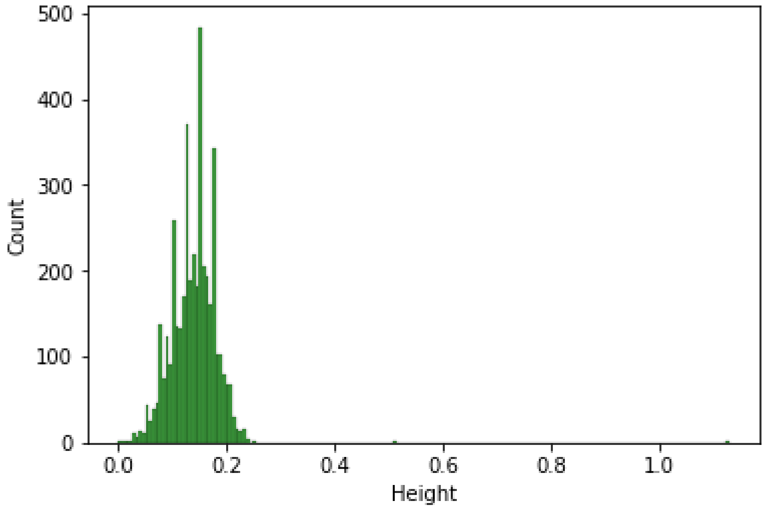
```
sns.countplot(x='Sex',data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f774f8e6510>



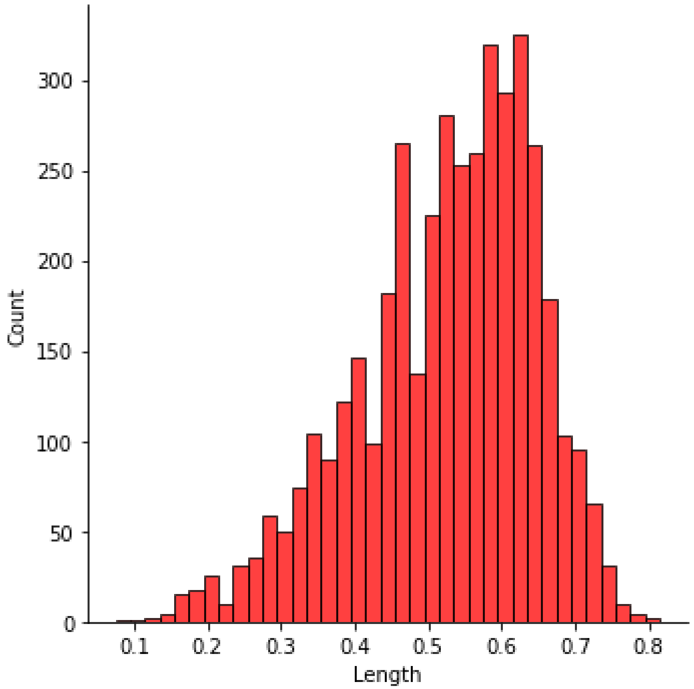

```
sns.histplot(df["Height"],color='green')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f774f6b56d0>




```
sns.displot(df["Length"],color='red')
```

```
<seaborn.axisgrid.FacetGrid at 0x7f774f6ee690>
```

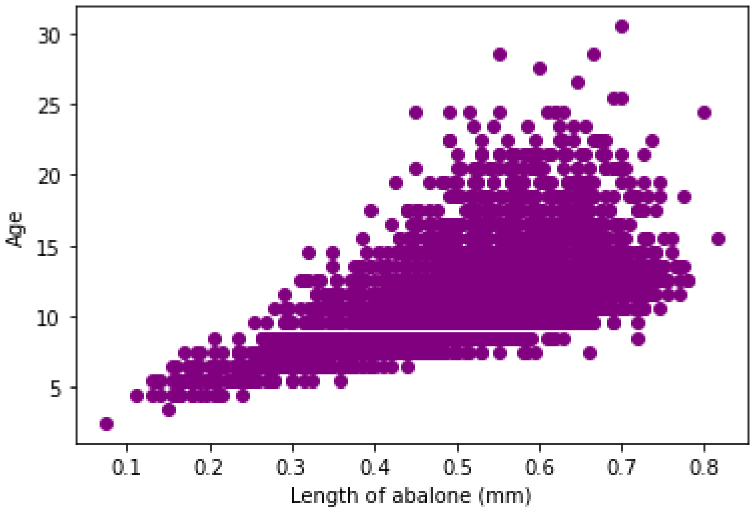



```
sns.boxplot(x=df["Length"],color='orange')
```

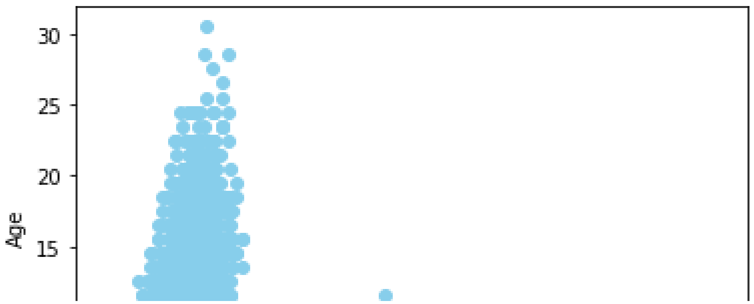
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f774c69d090>
```

Bi-Variate Analysis

```
plt.scatter(df['Length'], df['Age'], c='purple')
plt.xlabel('Length of abalone (mm)')
plt.ylabel('Age')
plt.show()
```

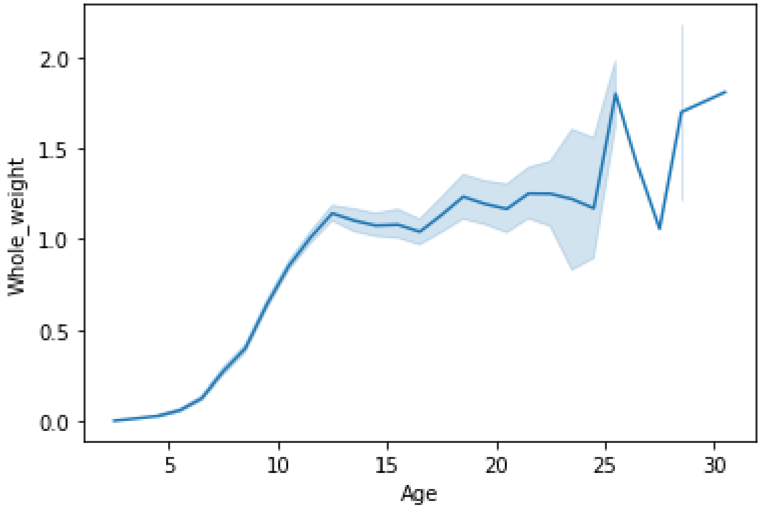



```
plt.scatter(df['Height'], df['Age'], c='skyblue')
plt.xlabel('Height of abalone (mm)')
plt.ylabel('Age')
plt.show()
```

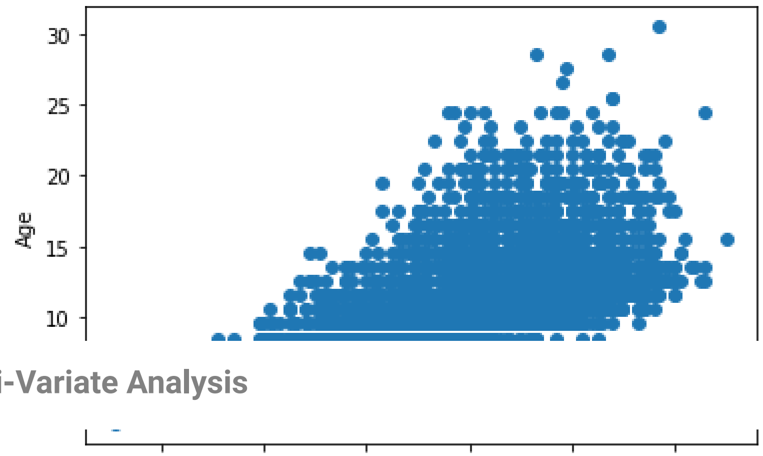



```
sns.lineplot(x=df["Age"],y=df["Whole_weight"])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f774c4d9dd0>



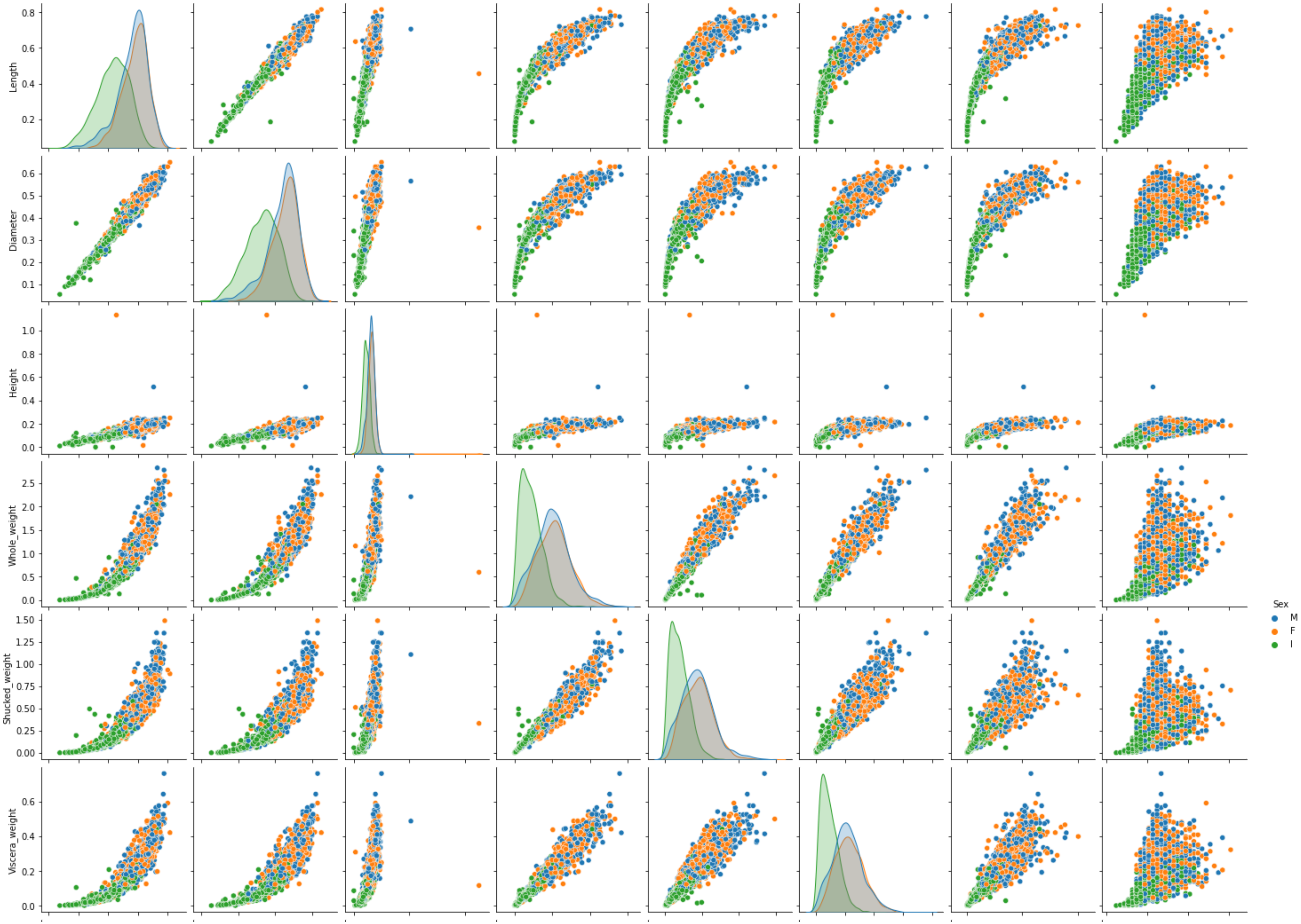

```
plt.scatter(df['Diameter'], df['Age'])
plt.xlabel('Height of abalone (mm)')
plt.ylabel('Age')
plt.show()
```

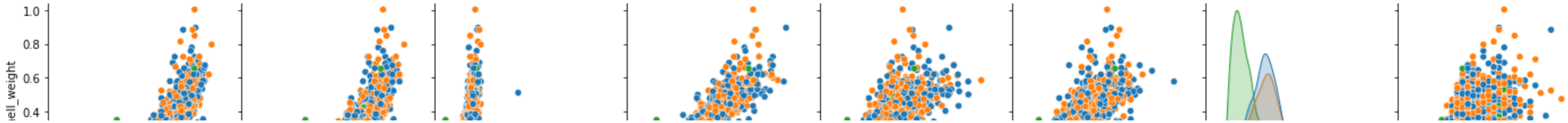


Multi-Variate Analysis

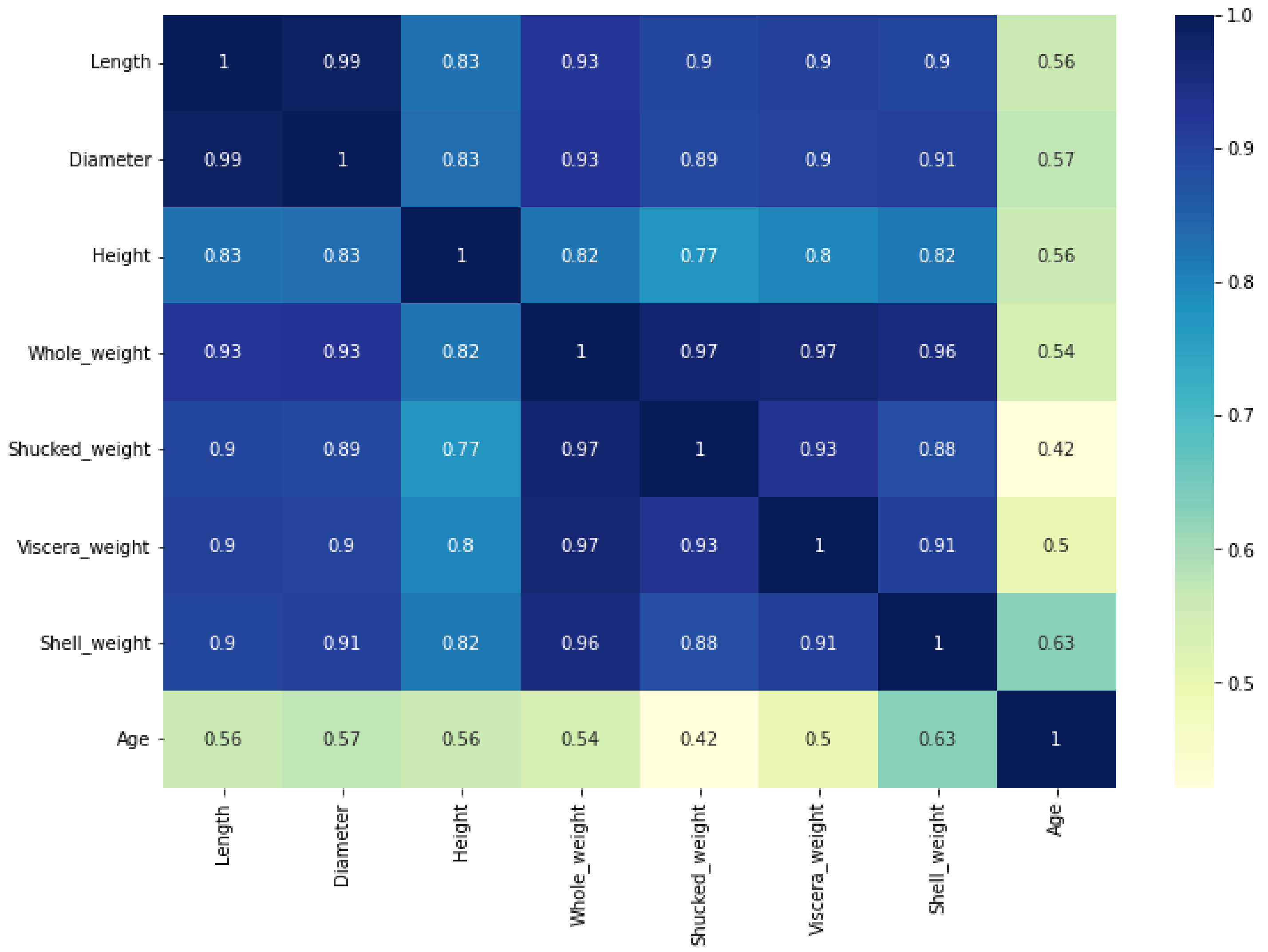
```
sns.pairplot(df,hue='Sex')
```

<seaborn.axisgrid.PairGrid at 0x7f774c49aed0>






```
plt.figure(figsize=(12,8));
sns.heatmap(df.corr(), cmap="YlGnBu",annot=True);
```



4. Descriptive statistics

```
df.describe()
```

	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

5. Check for Missing values and deal with them

```
df.isnull().sum()
```

```
Sex               0
Length            0
Diameter          0
Height            0
Whole_weight      0
Shucked_weight    0
Viscera_weight    0
Shell_weight      0
```

```
    Age                 0
    dtype: int64
```

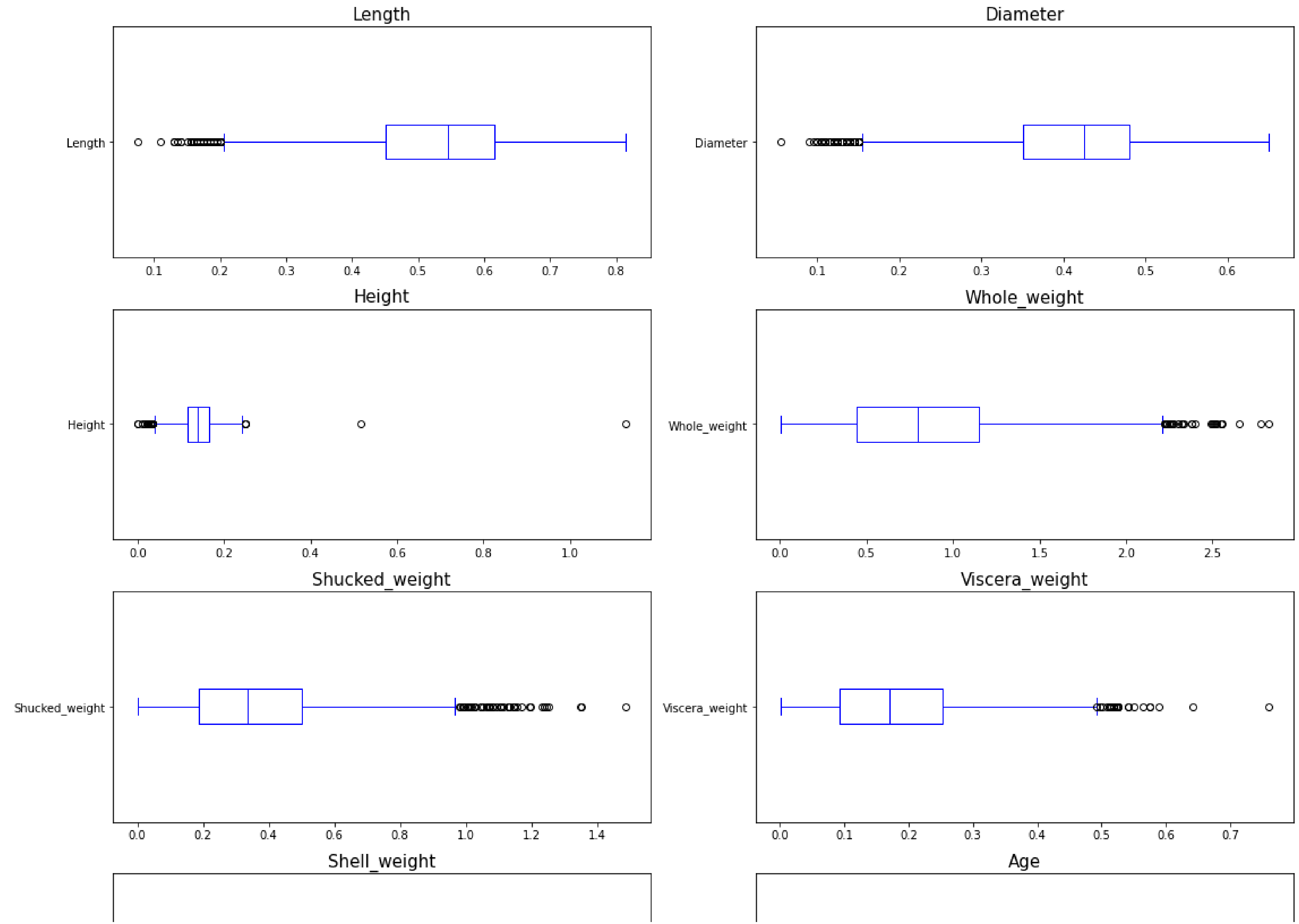
6. Find the outliers and Replace their outliers

```
col =['Length', 'Diameter', 'Height', 'Whole_weight', 'Shucked_weight',
       'Viscera_weight', 'Shell_weight', 'Age']
```

```
figfig, axes = plt.subplots(4,2,figsize=(16, 14))
axes = np.ravel(axes)

for i, c in enumerate(col):
    hist = df[c].plot(kind = 'box', ax=axes[i],color='blue', vert=False)
    axes[i].set_title(c, fontsize=15)

plt.tight_layout()
plt.show()
```


Length
Length
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Diameter
Diameter
0.1
0.2
0.3
0.4
0.5
0.6
Height
Height
0.0
0.2
0.4
0.6
0.8
1.0
Whole_weight
Whole_weight
0.0
0.5
1.0
1.5
2.0
2.5
Shucked_weight
Shucked_weight
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Viscera_weight
Viscera_weight
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Shell_weight
Age

```
qnt = df.quantile(q=(0.75,0.25))
qnt
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329	12.5
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	9.5

```
IQR = qnt.loc[0.75] - qnt.loc[0.25]
IQR
```

```
Length            0.1650
Diameter          0.1300
Height            0.0500
Whole_weight      0.7115
Shucked_weight    0.3160
Viscera_weight    0.1595
Shell_weight      0.1990
Age               3.0000
dtype: float64
```

```
lower = qnt.loc[0.25] - 1.5 * IQR
lower
```

```
Length            0.20250
Diameter          0.15500
Height            0.04000
Whole_weight     -0.62575
Shucked_weight   -0.28800
Viscera_weight   -0.14575
Shell_weight     -0.16850
Age               5.00000
dtype: float64
```

```
upper = qnt.loc[0.75] + 1.5 * IQR
upper
```

```
Length            0.86250
Diameter          0.67500
Height            0.24000
Whole_weight      2.22025
Shucked_weight    0.97600
Viscera_weight    0.49225
Shell_weight      0.62750
Age              17.00000
dtype: float64
```

```
df.mean()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Dropping of nuisance columns in DataFrame reduct
  """Entry point for launching an IPython kernel.
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
Age              11.433684
dtype: float64
```

```
df['Length']=np.where(df['Length']<0.22,0.52,df['Length'])
```

```
df['Diameter']=np.where(df['Diameter']<0.155,0.407,df['Diameter'])
```

```
df['Height']=np.where(df['Height']<0.04,0.13,df['Height'])
```

```
df['Height']=np.where(df['Height']>0.24,0.13,df['Height'])
```

```
df['Whole_weight']=np.where(df['Whole_weight']>2.18,0.83,df['Whole_weight'])
```

```
df['Shucked_weight']=np.where(df['Shucked_weight']>0.958,0.359367,df['Shucked_weight'])
```

```
df['Viscera_weight']=np.where(df['Viscera_weight']>0.478,0.18,df['Viscera_weight'])
```

```
df['Shell_weight']=np.where(df['Shell_weight']>0.61,0.238831,df['Shell_weight'])
```

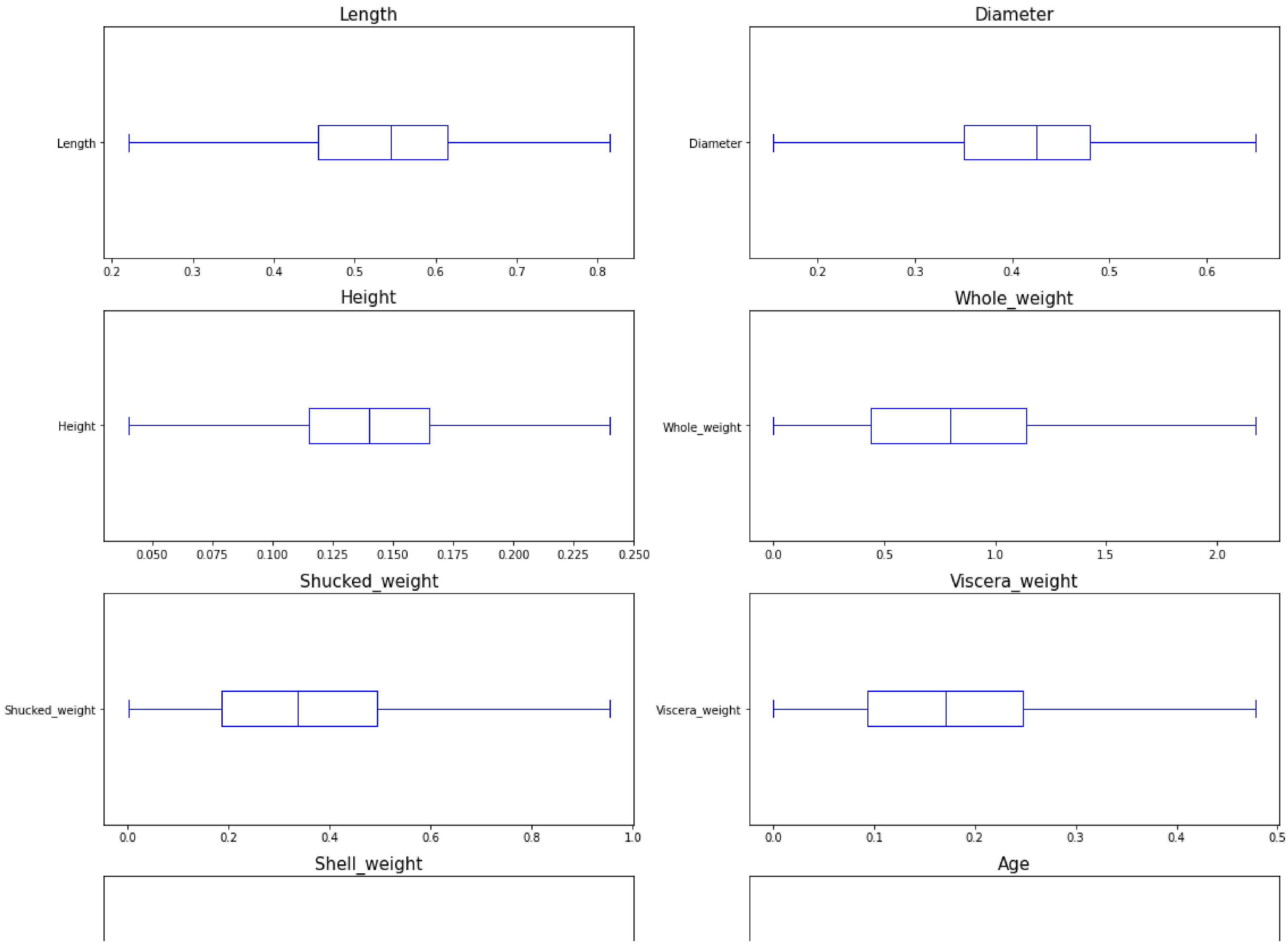
```
df['Age']=np.where(df['Age']<5.0,11.43,df['Age'])
```

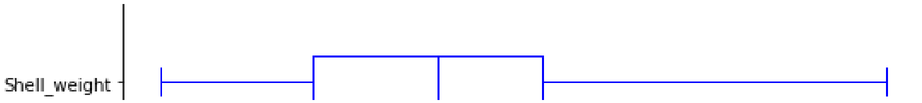
```
df['Age']=np.where(df['Age']>17.0,11.43,df['Age'])
```

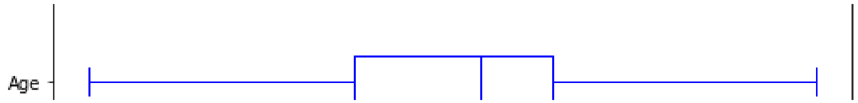
```
figfig, axes = plt.subplots(4,2,figsize=(16, 14))
axes = np.ravel(axes)

for i, c in enumerate(col):
    hist = df[c].plot(kind = 'box', ax=axes[i],color='blue', vert=False)
    axes[i].set_title(c, fontsize=15)

plt.tight_layout()
plt.show()
```


Length
Length
0.2 0.3 0.4 0.5 0.6 0.7 0.8
Diameter
Diameter
0.2 0.3 0.4 0.5 0.6
Height
Height
0.050 0.075 0.100 0.125 0.150 0.175 0.200 0.225 0.250
Whole_weight
Whole_weight
0.0 0.5 1.0 1.5 2.0
Shucked_weight
Shucked_weight
0.0 0.2 0.4 0.6 0.8 1.0
Viscera_weight
Viscera_weight
0.0 0.1 0.2 0.3 0.4 0.5
Shell_weight
Age






```
df.shape
```

```
(4177, 9)
```

7. Check for Categorical columns and perform encoding

```
df['Sex'].unique()
```

```
array(['M', 'F', 'I'], dtype=object)
```

```
x = pd.get_dummies(df)
```

```
x.head()
```

	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

8. Split the data into dependent and independent variables

```
x.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Length          4177 non-null   float64
 1   Diameter        4177 non-null   float64
 2   Height          4177 non-null   float64
 3   Whole_weight    4177 non-null   float64
 4   Shucked_weight  4177 non-null   float64
 5   Viscera_weight  4177 non-null   float64
 6   Shell_weight    4177 non-null   float64
 7   Age             4177 non-null   float64
 8   Sex_F           4177 non-null   uint8  
 9   Sex_I           4177 non-null   uint8  
 10  Sex_M           4177 non-null   uint8  
dtypes: float64(8), uint8(3)
memory usage: 273.4 KB
```

```
X = x.drop(['Age'], axis = 1)
```

```
X.head(2)
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Sex_F	Sex_I	Sex_M
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.15	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.07	0	0	1

```
y = x['Age']
```

```
y.head(2)
```

```
0    16.5
```

```
1     8.5
Name: Age, dtype: float64
```

9. Scale the independent variables

```
from sklearn.preprocessing import StandardScaler
```

```
X_columns = X.select_dtypes(include=np.number).columns.tolist()
X_columns
```

```
['Length',
 'Diameter',
 'Height',
 'Whole_weight',
 'Shucked_weight',
 'Viscera_weight',
 'Shell_weight',
 'Sex_F',
 'Sex_I',
 'Sex_M']
```

```
scaler = StandardScaler()
```

```
X[X_columns] = scaler.fit_transform(X[X_columns])
```

```
X.head(2)
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Sex_F	Sex_I	Sex_M
0	-0.663474	-0.501673	-1.196422	-0.643390	-0.611770	-0.732343	-0.643590	-0.674834	-0.688018	1.316677
1	-1.601273	-1.572915	-1.330241	-1.259765	-1.219694	-1.236126	-1.257424	-0.674834	-0.688018	1.316677

10. Split the data into training and testing

```
X.shape, y.shape
```

```
((4177, 10), (4177,))
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42)
```

```
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 :  (2923, 10)
 y_train.shape :  (2923,)
 x_test.shape :  (1254, 10)
 y_test.shape :  (1254,)
```

11. Build the Model

```
#Random Forest
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error,make_scorer
from sklearn.model_selection import RandomizedSearchCV

rf = RandomForestRegressor()

param = {
```

```
    'max_depth':[3,6,9,12,15],
    'n_estimators' : [10,50,100,150,200]
}

rf_search = RandomizedSearchCV(rf,param_distributions=param,n_iter=5,scoring=make_scorer(mean_squared_error),
                               n_jobs=-1,cv=5,verbose=3)

rf_search.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 5 candidates, totalling 25 fits
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_iter=5, n_jobs=-1,
                   param_distributions={'max_depth': [3, 6, 9, 12, 15],
                                        'n_estimators': [10, 50, 100, 150,
                                                         200]},
                   scoring=make_scorer(mean_squared_error), verbose=3)
```

12. Train the Model

```
means = rf_search.cv_results_['mean_test_score']
params = rf_search.cv_results_['params']
for mean, param in zip(means, params):
    print("%f with: %r" % (mean, param))
    if mean == min(means):
        print('Best parameters with the minimum Mean Square Error are:',param)
```

```
2.736678 with: {'n_estimators': 50, 'max_depth': 12}
2.648226 with: {'n_estimators': 50, 'max_depth': 6}
Best parameters with the minimum Mean Square Error are: {'n_estimators': 50, 'max_depth': 6}
2.742811 with: {'n_estimators': 150, 'max_depth': 15}
2.704251 with: {'n_estimators': 150, 'max_depth': 12}
2.732051 with: {'n_estimators': 200, 'max_depth': 15}
```

13. Test the Model

```
rf = RandomForestRegressor(n_estimators=200, max_depth=6)
rf.fit(x_train,y_train)

rf_pred =  rf.predict(x_test)
```

14. Measure the performance using Metrics

```
from sklearn import metrics
print('Random Forest Contains:')
print('-------------------')
print('MAE:', metrics.mean_absolute_error(y_test, rf_pred))
print('MSE:', metrics.mean_squared_error(y_test, rf_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, rf_pred)))
print('R2 Score :',metrics.r2_score(y_test,rf_pred))
```

```
Random Forest Contains:
-------------------
MAE: 1.2337570234045656
MSE: 2.5269930689816285
RMSE: 1.5896518703733935
R2 Score : 0.5338096549517406
```

0s completed at 11:08 AM