In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
fish_data = pd.read_csv("fish.csv")
```

In [3]:

```
fish_data.head()
```

Out[3]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

In [4]:

```
fish_data.tail()
```

Out[4]:

	Species	Weight	Length1	Length2	Length3	Height	Width
154	Smelt	12.2	11.5	12.2	13.4	2.0904	1.3936
155	Smelt	13.4	11.7	12.4	13.5	2.4300	1.2690
156	Smelt	12.2	12.1	13.0	13.8	2.2770	1.2558
157	Smelt	19.7	13.2	14.3	15.2	2.8728	2.0672
158	Smelt	19.9	13.8	15.0	16.2	2.9322	1.8792

In [5]:

```
fish_data.shape
```

Out[5]:

```
(159, 7)
```

In [6]:

```
fish_data.columns
```

Out[6]:

```
Index(['Species', 'Weight', 'Length1', 'Length2', 'Length3', 'Height',
       'Width'],
      dtype='object')
```

In [7]:

```
fish_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 7 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   Species  159 non-null    object 
 1   Weight   159 non-null    float64
 2   Length1  159 non-null    float64
 3   Length2  159 non-null    float64
 4   Length3  159 non-null    float64
 5   Height   159 non-null    float64
 6   Width    159 non-null    float64
dtypes: float64(6), object(1)
memory usage: 8.8+ KB
```

In [8]:

```
fish_data.describe()
```

Out[8]:

	Weight	Length1	Length2	Length3	Height	Width
count	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000
mean	398.326415	26.247170	28.415723	31.227044	8.970994	4.417486
std	357.978317	9.996441	10.716328	11.610246	4.286208	1.685804
min	0.000000	7.500000	8.400000	8.800000	1.728400	1.047600
25%	120.000000	19.050000	21.000000	23.150000	5.944800	3.385650
50%	273.000000	25.200000	27.300000	29.400000	7.786000	4.248500
75%	650.000000	32.700000	35.500000	39.650000	12.365900	5.584500
max	1650.000000	59.000000	63.400000	68.000000	18.957000	8.142000

In [9]:

```
fish_data.isnull().sum()
```

Out[9]:

```
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64
```

In [10]:

```
fish_data.nunique()
```

Out[10]:

```
Species      7
Weight     101
Length1    116
Length2     93
Length3    124
Height     154
Width      152
dtype: int64
```

In [11]:

```
fish_data['Species'].unique()
```

Out[11]:

```
array(['Bream', 'Roach', 'Whitefish', 'Parkki', 'Perch', 'Pike', 'Smelt'],
      dtype=object)
```

In [12]:

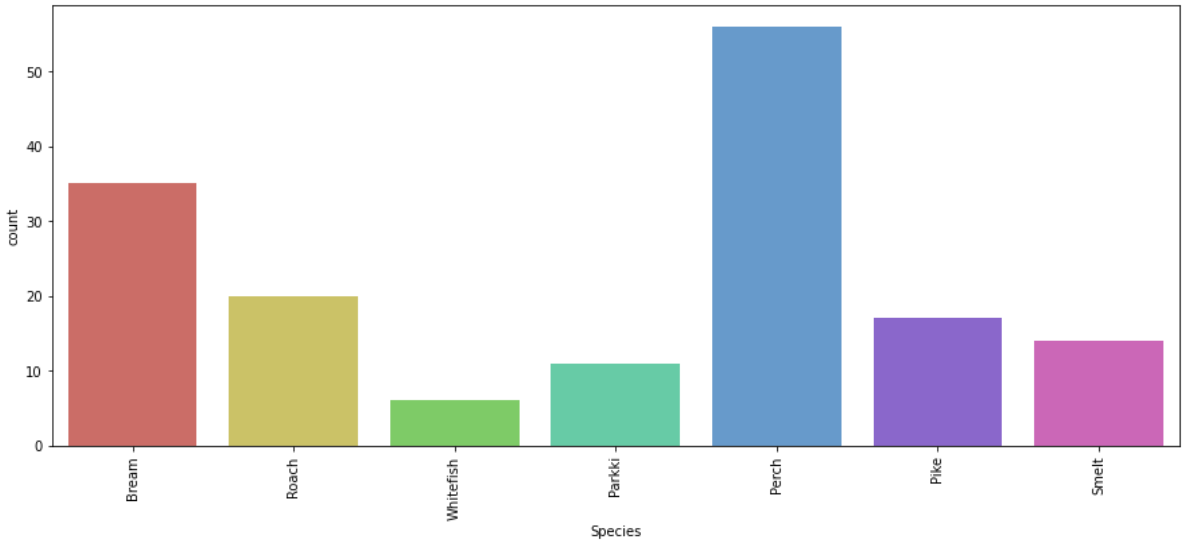
```
fish_data['Species'].value_counts()
```

Out[12]:

```
Perch        56
Bream        35
Roach        20
Pike         17
Smelt        14
Parkki       11
Whitefish     6
Name: Species, dtype: int64
```

In [13]:

```
plt.figure(figsize=(15,6))
sns.countplot('Species', data = fish_data, palette='hls')
plt.xticks(rotation = 90)
plt.show()
```

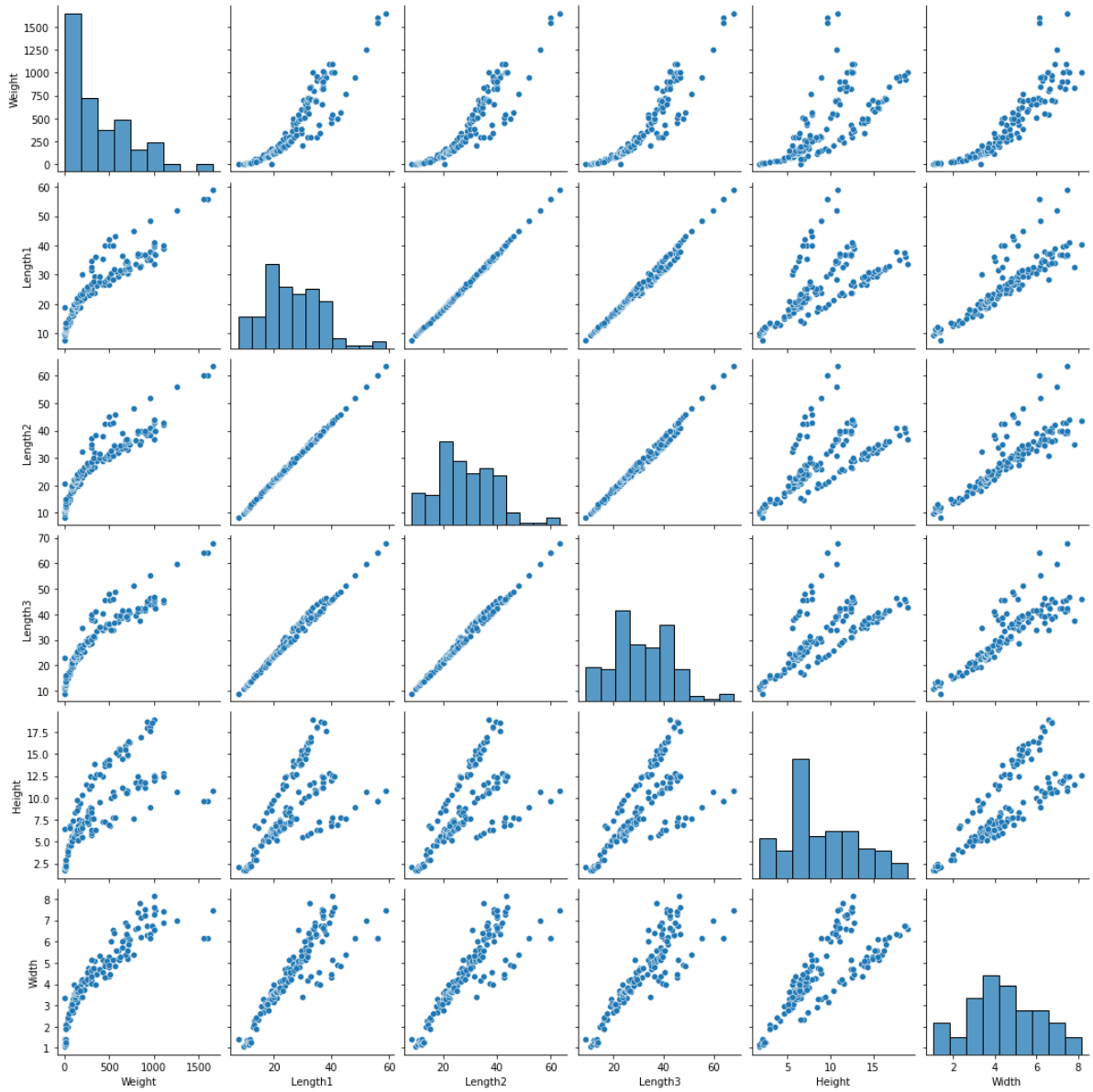


In [14]:

```
sns.pairplot(fish_data)
```

Out[14]:

```
<seaborn.axisgrid.PairGrid at 0xb9d5c460d0>
```



In [15]:

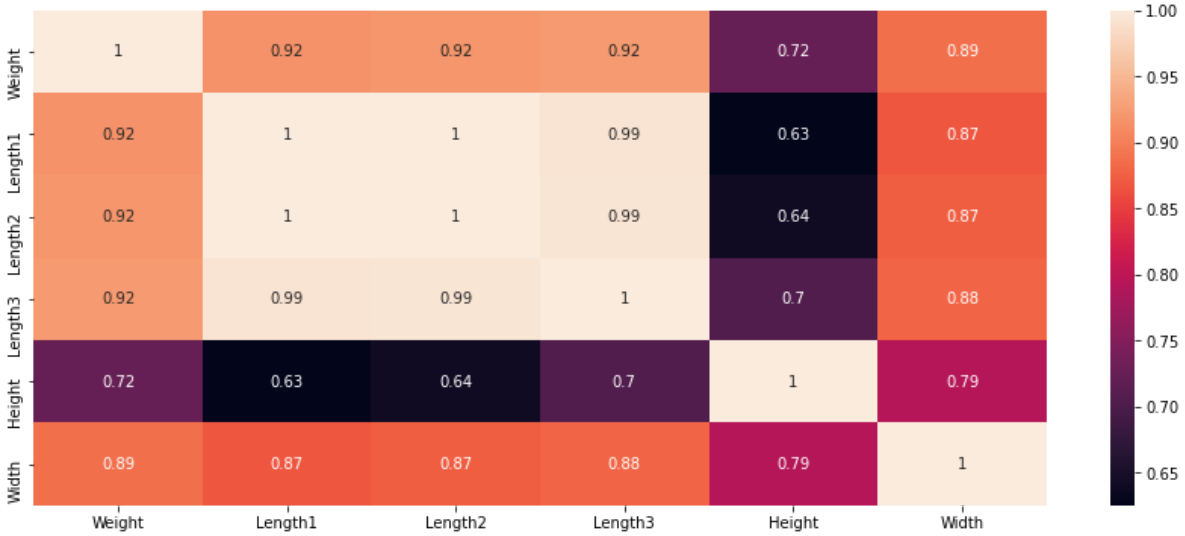
```
fish_data.corr()
```

Out[15]:

	Weight	Length1	Length2	Length3	Height	Width
Weight	1.000000	0.915712	0.918618	0.923044	0.724345	0.886507
Length1	0.915712	1.000000	0.999517	0.992031	0.625378	0.867050
Length2	0.918618	0.999517	1.000000	0.994103	0.640441	0.873547
Length3	0.923044	0.992031	0.994103	1.000000	0.703409	0.878520
Height	0.724345	0.625378	0.640441	0.703409	1.000000	0.792881
Width	0.886507	0.867050	0.873547	0.878520	0.792881	1.000000

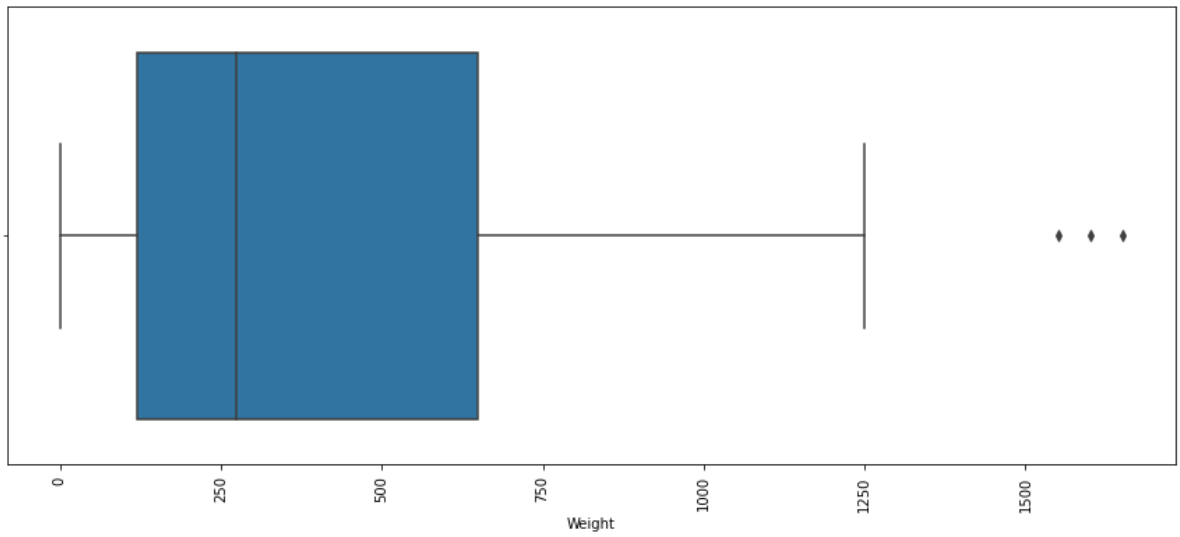
In [17]:

```
plt.figure(figsize=(15,6))
sns.heatmap(fish_data.corr(), annot = True)
plt.show()
```



In [18]:

```
plt.figure(figsize=(15,6))
sns.boxplot(fish_data['Weight'])
plt.xticks(rotation = 90)
plt.show()
```



In [19]:

```
fish_weight = fish_data['Weight']
Q3 = fish_weight.quantile(0.75)
Q1 = fish_weight.quantile(0.25)
IQR = Q3-Q1
lower_limit = Q1 -(1.5*IQR)
upper_limit = Q3 +(1.5*IQR)
```

In [20]:

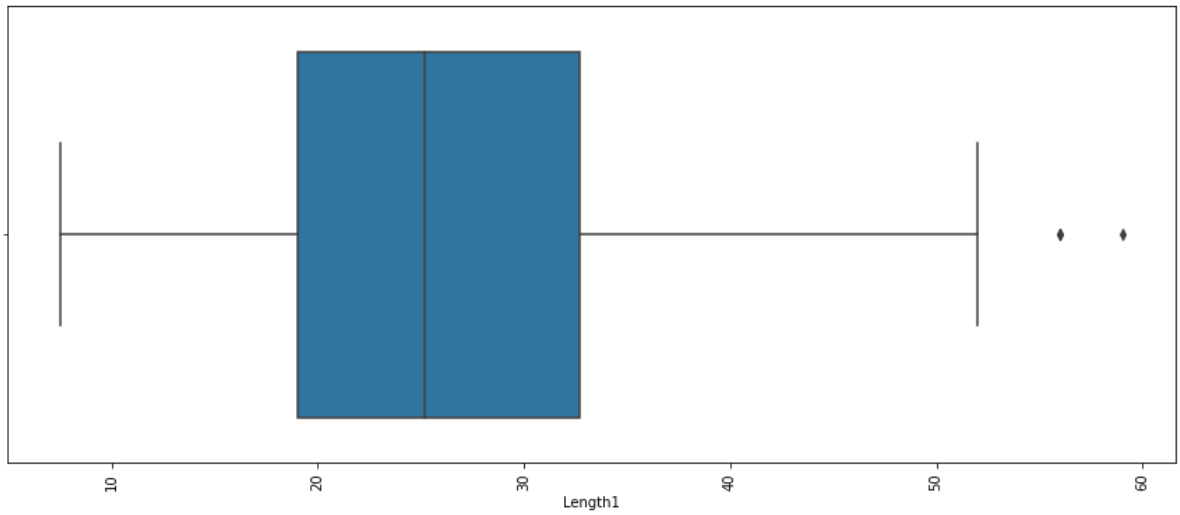
```
weight_outliers = fish_weight[(fish_weight <lower_limit) | (fish_weight >upper_limit)]
weight_outliers
```

Out[20]:

```
142    1600.0
143    1550.0
144    1650.0
Name: Weight, dtype: float64
```

In [21]:

```
plt.figure(figsize=(15,6))
sns.boxplot(fish_data['Length1'])
plt.xticks(rotation = 90)
plt.show()
```



In [22]:

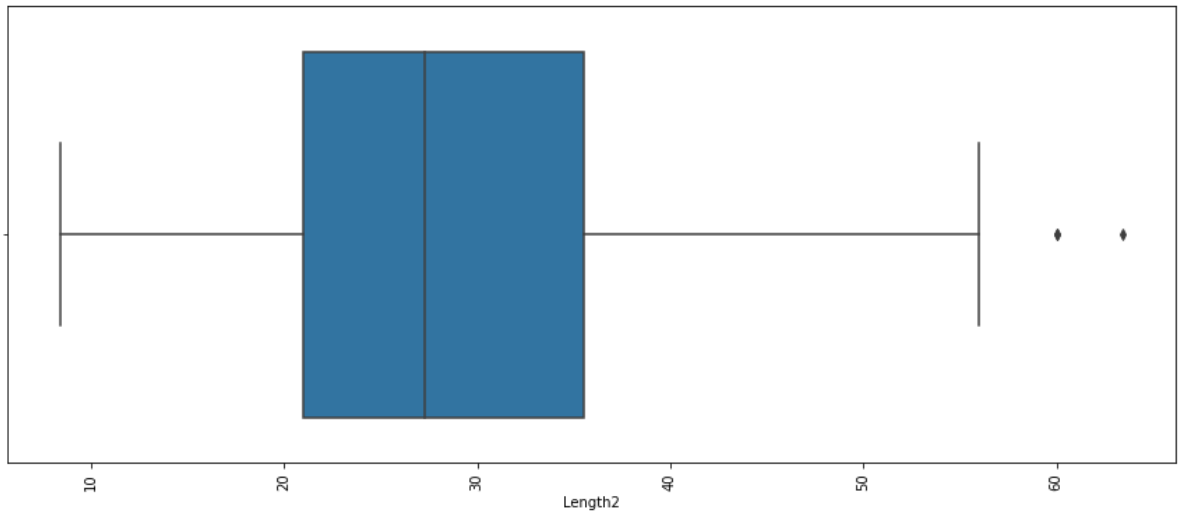
```
fish_Length1 = fish_data['Length1']
Q3 = fish_Length1.quantile(0.75)
Q1 = fish_Length1.quantile(0.25)
IQR = Q3-Q1
lower_limit = Q1 -(1.5*IQR)
upper_limit = Q3 +(1.5*IQR)
length1_outliers = fish_Length1[(fish_Length1 <lower_limit) | (fish_Length1 >upper_limi
length1_outliers
```

Out[22]:

```
142    56.0
143    56.0
144    59.0
Name: Length1, dtype: float64
```

In [23]:

```
plt.figure(figsize=(15,6))
sns.boxplot(fish_data['Length2'])
plt.xticks(rotation = 90)
plt.show()
```



In [24]:

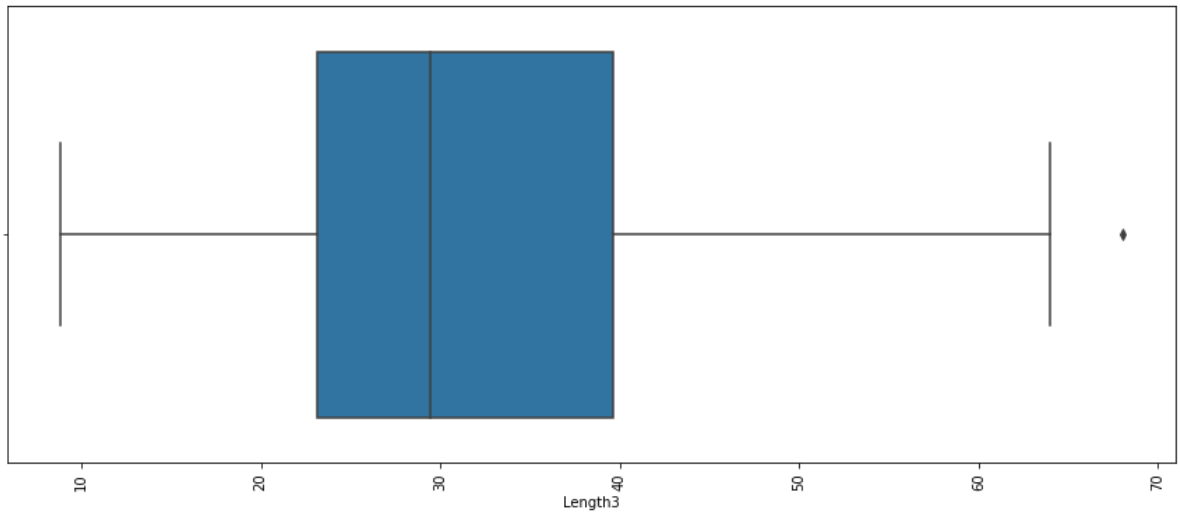
```
fish_Length2 = fish_data['Length2']
Q3 = fish_Length2.quantile(0.75)
Q1 = fish_Length2.quantile(0.25)
IQR = Q3-Q1
lower_limit = Q1 -(1.5*IQR)
upper_limit = Q3 +(1.5*IQR)
length2_outliers = fish_Length2[(fish_Length2 <lower_limit) | (fish_Length2 >upper_limi
length2_outliers
```

Out[24]:

```
142    60.0
143    60.0
144    63.4
Name: Length2, dtype: float64
```

In [25]:

```
plt.figure(figsize=(15,6))
sns.boxplot(fish_data['Length3'])
plt.xticks(rotation = 90)
plt.show()
```



In [26]:

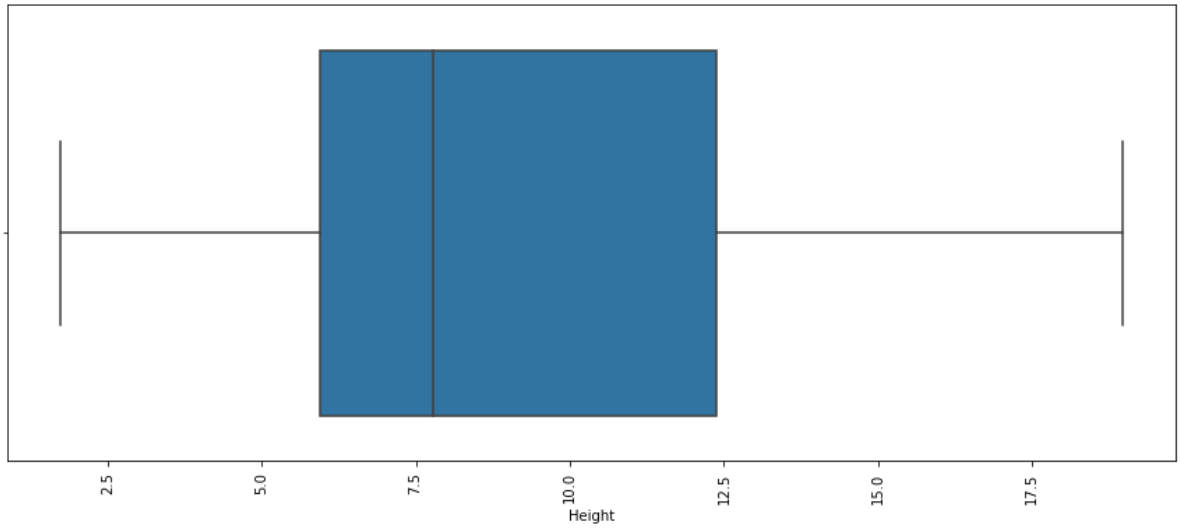
```
fish_Length3 = fish_data['Length3']
Q3 = fish_Length3.quantile(0.75)
Q1 = fish_Length3.quantile(0.25)
IQR = Q3-Q1
lower_limit = Q1 -(1.5*IQR)
upper_limit = Q3 +(1.5*IQR)
length3_outliers = fish_Length3[(fish_Length3 <lower_limit) | (fish_Length3 >upper_limi
length3_outliers
```

Out[26]:

```
144    68.0
Name: Length3, dtype: float64
```

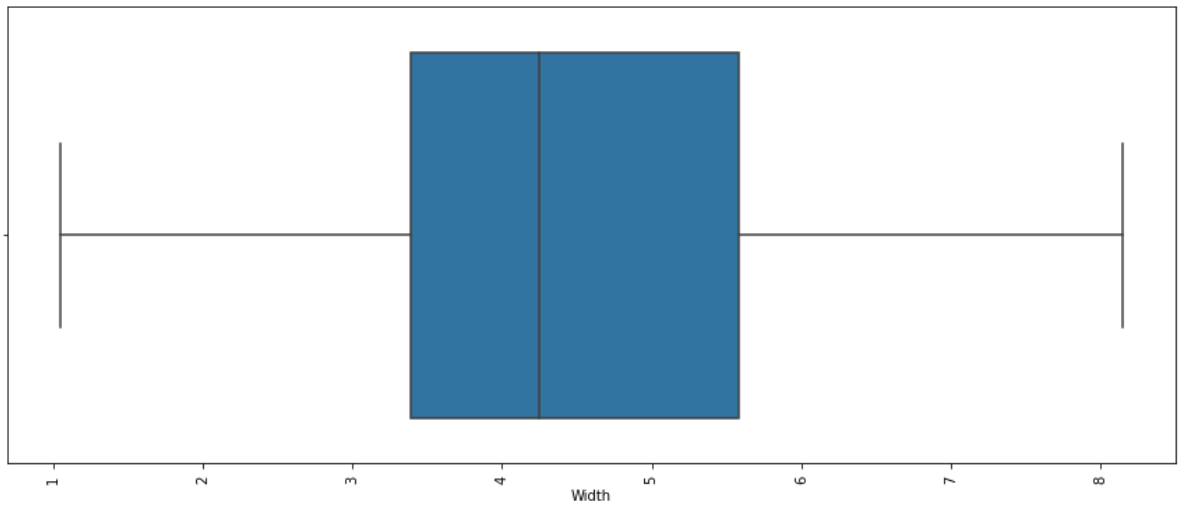
In [27]:

```
plt.figure(figsize=(15,6))
sns.boxplot(fish_data['Height'])
plt.xticks(rotation = 90)
plt.show()
```



In [28]:

```
plt.figure(figsize=(15,6))
sns.boxplot(fish_data['Width'])
plt.xticks(rotation = 90)
plt.show()
```



In [29]:

```
fish_data[142:145]
```

Out[29]:

	Species	Weight	Length1	Length2	Length3	Height	Width
142	Pike	1600.0	56.0	60.0	64.0	9.600	6.144
143	Pike	1550.0	56.0	60.0	64.0	9.600	6.144
144	Pike	1650.0	59.0	63.4	68.0	10.812	7.480

In [47]:

```
fish_data_new = fish_data.drop([142,143,144])
```

In [48]:

```
fish_data_new.head()
```

Out[48]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

In [49]:

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

In [50]:

```
scaler = StandardScaler()
```

In [51]:

```
scaling_columns = ['Weight', 'Length1','Length2','Length3','Height','Width']
fish_data_new[scaling_columns] = scaler.fit_transform(fish_data_new[scaling_columns])
fish_data_new.describe()
```

Out[51]:

	Weight	Length1	Length2	Length3	Height	Wid
count	1.560000e+02	1.560000e+02	1.560000e+02	1.560000e+02	1.560000e+02	1.560000e+(
mean	-5.693451e-17	1.281027e-17	-4.896368e-16	-2.049643e-16	1.366428e-16	1.480297e-
std	1.003221e+00	1.003221e+00	1.003221e+00	1.003221e+00	1.003221e+00	1.003221e+(
min	-1.177998e+00	-1.997257e+00	-1.986079e+00	-2.042126e+00	-1.675635e+00	-1.996688e+(
25%	-8.012570e-01	-7.321794e-01	-6.952556e-01	-7.078263e-01	-7.004973e-01	-6.036162e-(
50%	-3.271912e-01	-7.213903e-02	-1.061895e-01	-1.239237e-01	-3.023653e-01	-7.944425e-(
75%	7.449512e-01	7.116589e-01	7.389923e-01	8.304874e-01	7.951438e-01	6.291357e-(
max	2.746388e+00	2.898043e+00	2.890364e+00	2.732275e+00	2.321310e+00	2.259561e+(

In [52]:

```
data_cleaned = fish_data_new.drop("Weight", axis=1)
y = fish_data_new['Weight']
```

In [53]:

```
from sklearn.model_selection import train_test_split
```

In [54]:

```
x_train, x_test, y_train, y_test = train_test_split(data_cleaned,y,
                                                    test_size=0.2,
                                                    random_state=42)
```

In [55]:

```
from sklearn.preprocessing import  LabelEncoder
```

In [56]:

```
label_encoder = LabelEncoder()
```

In [57]:

```
x_train['Species'] = label_encoder.fit_transform(x_train['Species'].values)
x_test['Species'] = label_encoder.transform(x_test['Species'].values)
```

In [59]:

```
pip install hyperopt
```

```
Collecting hyperopt
  Downloading hyperopt-0.2.7-py2.py3-none-any.whl (1.6 MB)
     ---------------------------------------- 1.6/1.6 MB 1.3 MB/s eta 0:0
0:00
Requirement already satisfied: tqdm in c:\python\lib\site-packages (from h
yperopt) (4.51.0)
Requirement already satisfied: six in c:\python\lib\site-packages (from hy
peropt) (1.12.0)
Collecting cloudpickle
  Downloading cloudpickle-2.1.0-py3-none-any.whl (25 kB)
Requirement already satisfied: numpy in c:\python\lib\site-packages (from
hyperopt) (1.22.3)
Requirement already satisfied: future in c:\python\lib\site-packages (from
hyperopt) (0.18.2)
Collecting py4j
  Downloading py4j-0.10.9.5-py2.py3-none-any.whl (199 kB)
     -------------------------------------- 199.7/199.7 KB 1.7 MB/s eta 0:
00:00
Requirement already satisfied: networkx>=2.2 in c:\python\lib\site-package
s (from hyperopt) (2.5)
Requirement already satisfied: scipy in c:\python\lib\site-packages (from
hyperopt) (1.5.4)
Requirement already satisfied: decorator>=4.3.0 in c:\python\lib\site-pack
ages (from networkx>=2.2->hyperopt) (4.4.2)
Installing collected packages: py4j, cloudpickle, hyperopt
Successfully installed cloudpickle-2.1.0 hyperopt-0.2.7 py4j-0.10.9.5
Note: you may need to restart the kernel to use updated packages.


WARNING: Ignoring invalid distribution -illow (c:\python\lib\site-package
s)
WARNING: Ignoring invalid distribution -atplotlib (c:\python\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -illow (c:\python\lib\site-package
s)
WARNING: Ignoring invalid distribution -atplotlib (c:\python\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -illow (c:\python\lib\site-package
s)
WARNING: Ignoring invalid distribution -atplotlib (c:\python\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -illow (c:\python\lib\site-package
s)
WARNING: Ignoring invalid distribution -atplotlib (c:\python\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -illow (c:\python\lib\site-package
s)
WARNING: Ignoring invalid distribution -atplotlib (c:\python\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -illow (c:\python\lib\site-package
s)
WARNING: Ignoring invalid distribution -atplotlib (c:\python\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -illow (c:\python\lib\site-package
s)
WARNING: Ignoring invalid distribution -atplotlib (c:\python\lib\site-pack
```

```
ages)
WARNING: Ignoring invalid distribution -illow (c:\python\lib\site-package
s)
WARNING: Ignoring invalid distribution -atplotlib (c:\python\lib\site-pack
ages)
WARNING: Ignoring invalid distribution -illow (c:\python\lib\site-package
s)
WARNING: Ignoring invalid distribution -atplotlib (c:\python\lib\site-pack
ages)
WARNING: You are using pip version 22.0.4; however, version 22.1.1 is avai
lable.
You should consider upgrading via the 'c:\python\python.exe -m pip install
--upgrade pip' command.
```

In [60]:

```
from itertools import combinations
from hyperopt import hp
from hyperopt import fmin, tpe, STATUS_OK, STATUS_FAIL, Trials
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

In [63]:

```
def evauation_model(pred, y_val):
  score_MSE = round(mean_squared_error(pred, y_val),2)
  score_MAE = round(mean_absolute_error(pred, y_val),2)
  score_r2score = round(r2_score(pred, y_val),2)
  return score_MSE, score_MAE, score_r2score
```

In [61]:

```
def models_score(model_name, train_data, y_train, val_data, y_val):
    model_list = ["Decision_Tree", "Random_Forest", "XGboost_Regressor"]
    # model_1
    if model_name == "Decision_Tree":
        reg = DecisionTreeRegressor(random_state=42)
    # model_2
    elif model_name == "Random_Forest":
        reg = RandomForestRegressor(random_state=42)

    # model_3
    elif model_name == "XGboost_Regressor":
        reg = xgb.XGBRegressor(objective="reg:squarederror", random_state=42, )
    else:
        print("please enter correct regressor name")

    if model_name in model_list:
        reg.fit(train_data, y_train)
        pred = reg.predict(val_data)

        score_MSE, score_MAE, score_r2score = evauation_model(pred, y_val)
        return round(score_MSE, 2), round(score_MAE, 2), round(score_r2score, 2)
```

In [64]:

```
model_list = ["Decision_Tree","Random_Forest","XGboost_Regressor"]
result_scores = []
for model in model_list:
    score = models_score(model, x_train, y_train, x_test, y_test)
    result_scores.append((model, score[0], score[1],score[2]))
    print(model,score)
```

```
Decision_Tree (0.05, 0.16, 0.94)
Random_Forest (0.03, 0.11, 0.97)
XGboost_Regressor (0.04, 0.14, 0.96)
```

In [65]:

```
df_result_scores = pd.DataFrame(result_scores,columns=["model","mse","mae","r2score"])
df_result_scores
```

Out[65]:

	model	mse	mae	r2score
0	Decision_Tree	0.05	0.16	0.94
1	Random_Forest	0.03	0.11	0.97
2	XGboost_Regressor	0.04	0.14	0.96

In [66]:

```
num_estimator = [100, 150, 200, 250]
```

In [67]:

```
space = {'max_depth': hp.quniform("max_depth", 3, 18, 1),
         'gamma': hp.uniform('gamma', 1, 9),
         'reg_alpha': hp.quniform('reg_alpha', 30, 180, 1),
         'reg_lambda': hp.uniform('reg_lambda', 0, 1),
         'colsample_bytree': hp.uniform('colsample_bytree', 0.5, 1),
         'min_child_weight': hp.quniform('min_child_weight', 0, 10, 1),
         'n_estimators': hp.choice("n_estimators", num_estimator),
         }
```

In [68]:

```
def hyperparameter_tuning(space):
    model = xgb.XGBRegressor(n_estimators=space['n_estimators'], max_depth=int(space['ma
                             gamma=space['gamma'],
                             reg_alpha=int(space['reg_alpha']), min_child_weight=space[
                             colsample_bytree=space['colsample_bytree'], objective="reg

    score_cv = cross_val_score(model, x_train, y_train, cv=5, scoring="neg_mean_absolute
    return {'loss': -score_cv, 'status': STATUS_OK, 'model': model}
```

In [69]:

```
trials = Trials()
best = fmin(fn=hyperparameter_tuning,
            space=space,
            algo=tpe.suggest,
            max_evals=200,
            trials=trials)

print(best)
```

```
100%|██████| 200/200 [02:00<00:00,  1.66trial/s, best loss: 0.658911115941
0673]
{'colsample_bytree': 0.810503648295446, 'gamma': 1.0010030640720304, 'max_
depth': 16.0, 'min_child_weight': 5.0, 'n_estimators': 2, 'reg_alpha': 30.
0, 'reg_lambda': 0.2825700829452797}
```

In [70]:

```
best['max_depth'] = int(best['max_depth']) # convert to int
best["n_estimators"] = num_estimator[best["n_estimators"]] # assing n_estimator because
best_xgboost_model = xgb.XGBRegressor(**best)
best_xgboost_model.fit(x_train,y_train)
pred = best_xgboost_model.predict(x_test)
score_MSE, score_MAE, score_r2score = evauation_model(pred,y_test)
to_append = ["XGboost_hyper_tuned",score_MSE, score_MAE, score_r2score]
df_result_scores.loc[len(df_result_scores)] = to_append
```

In [71]:

```
best_xgboost_model.save_model("best_model.json")
```