SLR MID EXAM

Data_set:

This dataset is a record of 7 common different fish species in fish market sales. With this dataset, a predictive model can be performed using machine friendly data and estimate the weight of fish can be predicted.

1. **Species: Species name of fish**
2. **Weight: Weight of fish in gram**
3. **Length1: Vertical length in cm**
4. **Length2: Diagonal length in cm**
5. **Length3: Cross length in cm**
6. **Height: Height in cm**
7. **Width: Diagonal width in cm**

Our dependent variable is 'Weight'. Independent variables are 'species', different lengths, 'height' and 'width'.

In [1]:

```
# Kindly change the below cells from markdown to code and execute it
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
%matplotlib inline
```

In [2]:

```
import pandas as pd

import csv

with open("data_set.csv","r")as file:

    reader=csv.reader(file)
df=pd.read_csv("data_set.csv")

df.head()
```

Out[2]:

	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

1. Data Understanding (5 marks)

a. Read the dataset (tab, csv, xls, txt, inbuilt dataset). What are the number of rows and no. of cols & types of variables (continuous, categorical etc.)? (1 MARK)

b. Calculate five-point summary for numerical variables (1 MARK)

c. Summarize observations for categorical variables – no. of categories, % observations in each category. (1 mark)

d. Check for defects in the data such as missing values, null, outliers, etc. (2 marks)

In [3]:

```
# a. Read the dataset (tab, csv, xls, txt, inbuilt dataset).
# What are the number of rows and no. of cols & types of variables (continuous, categoric
al etc.)? (1 MARK)

print('number of rows:',df.shape[0])
print(' ')
print('number of columns:',df.shape[1])
print(' ')
print('Type of Variables:')
print(' ')
print(df.info())
```

```
number of rows: 159

number of columns: 7

Type of Variables:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 159 entries, 0 to 158
Data columns (total 7 columns):
Species    159 non-null object
Weight     159 non-null float64
Length1    159 non-null float64
Length2    159 non-null float64
Length3    159 non-null float64
Height     159 non-null float64
Width      159 non-null float64
dtypes: float64(6), object(1)
memory usage: 8.8+ KB
None
```

In [4]:

```
# b. Calculate five-point summary for numerical variables (1 MARK)

df.describe()
```

Out[4]:

	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 [5]:

```
# c. Summarize observations for categorical variables - no. of categories, % observations
in each category. (1 mark)

df_cat=df['Species'].value_counts()
df_cat=pd.DataFrame(df_cat)
df_cat['%']=df_cat['Species'].apply(lambda x:round((x/159)*100,2))
df_cat
```

Out[5]:

	Species	%
Perch	56	35.22
Bream	35	22.01
Roach	20	12.58
Pike	17	10.69
Smelt	14	8.81
Parkki	11	6.92
Whitefish	6	3.77

In [6]:

```
# d. Check for defects in the data such as missing values, null, outliers, etc. (2 marks)


print('Missing Values in the data :')
print(df.isnull().sum())
```

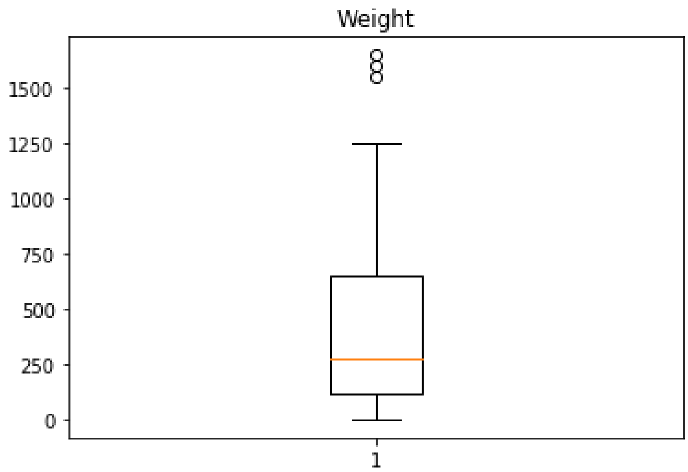
```
Missing Values in the data :
Species     0
Weight      0
Length1     0
Length2     0
Length3     0
Height      0
Width       0
dtype: int64
```

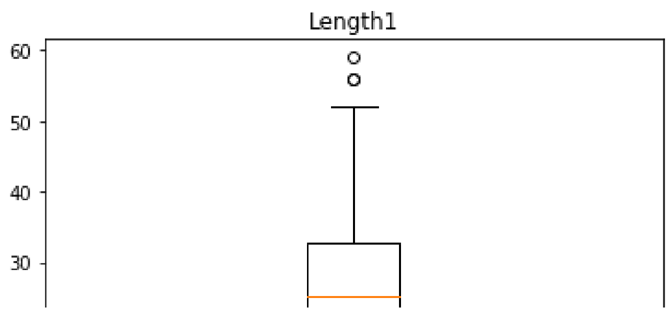
- **There are no missing values in the data**

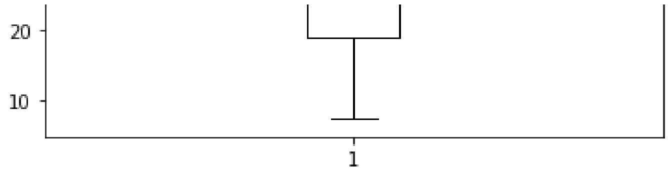
In [7]:

```
df_num=df.select_dtypes(np.number)
print('Outliers in the data :')
for features in df_num.columns:
    plt.boxplot(df_num[features])
    plt.title(features)
    plt.show()
```

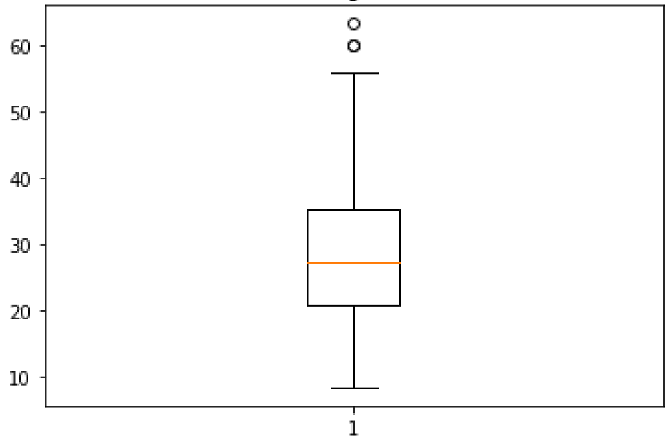
```
Outliers in the data :
```



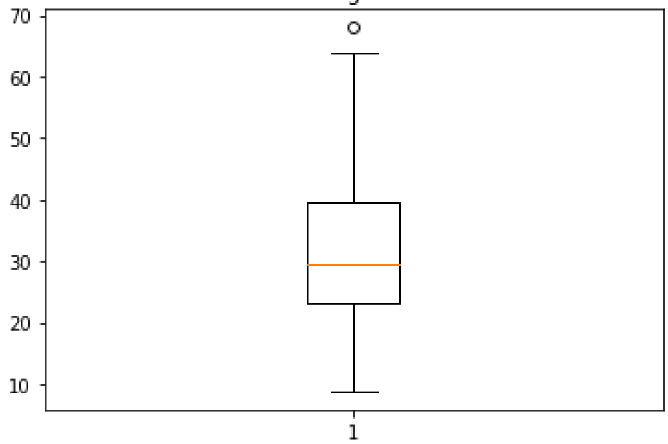




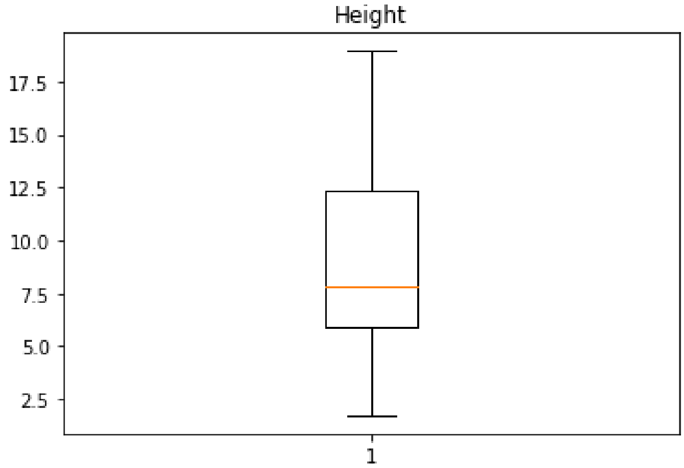
Length2



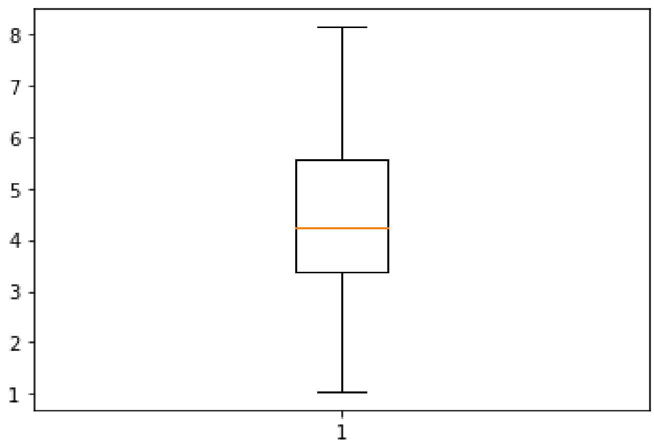
Length3



Height



Width



- **We could see Outliers in weight,length1,length2 and length3 columns in the data with very less number of outliers**

2. Data Preparation (10 marks)

a. Fix the defects found above and do appropriate treatment if any. (3 marks)

b. Visualize the data using relevant plots. Find out the variables which are highly correlated with target variable? (3 marks)

c. Do you want to exclude some variables from the model based on this analysis? What other actions will you take? (2 marks)

d. Split dataset into train and test (70:30). Are both train and test representative of the overall data? How would you ascertain this statistically? (2 marks)

In [8]:

```
# a. Fix the defects found above and do appropriate treatment if any. (3 marks)

# Since we could not see any null value/missing values in the data we dont need to handle it.

# We just have to remove the outliers in the data using IQR method

q1=df['Weight'].quantile(0.25)
q3=df['Weight'].quantile(0.75)
iqr=q3-q1
df= df[(df['Weight']>q1-(1.5*iqr)) & (df['Weight']<q3+(1.5*iqr))]
```

In [9]:

```
q1=df['Length1'].quantile(0.25)
q3=df['Length1'].quantile(0.75)
iqr=q3-q1
df= df[(df['Length1']>q1-(1.5*iqr)) & (df['Length1']<q3+(1.5*iqr))]
```

In [10]:

```
q1=df['Length2'].quantile(0.25)
q3=df['Length2'].quantile(0.75)
iqr=q3-q1
df= df[(df['Length2']>q1-(1.5*iqr)) & (df['Length2']<q3+(1.5*iqr))]
```

In [11]:

```
q1=df['Length3'].quantile(0.25)
q3=df['Length3'].quantile(0.75)
iqr=q3-q1
df= df[(df['Length3']>q1-(1.5*iqr)) & (df['Length3']<q3+(1.5*iqr))]
```

In [12]:

```
df.shape
```

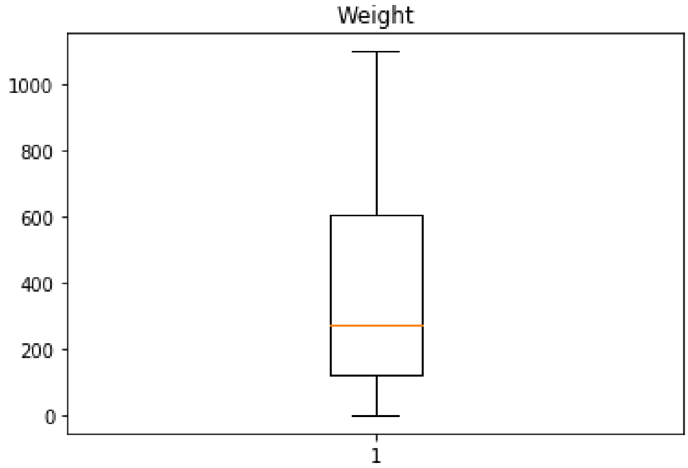
Out[12]:

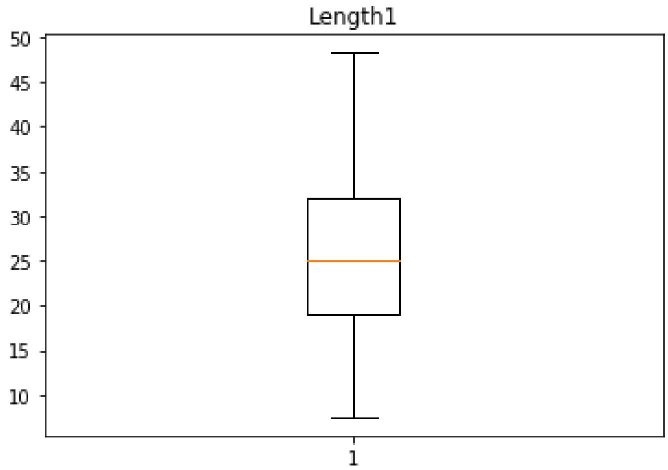
```
(155, 7)
```

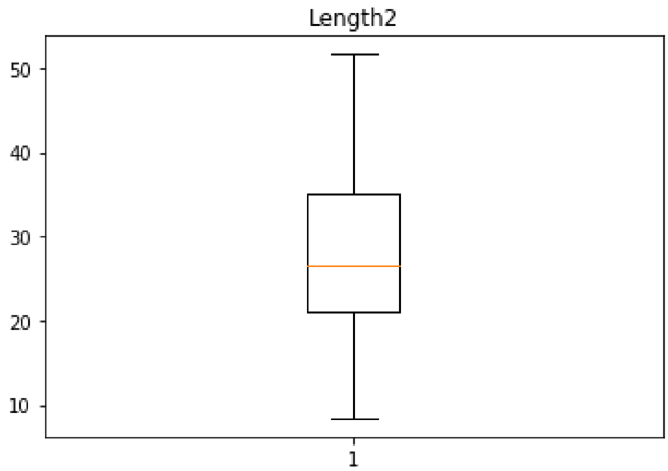
In [13]:

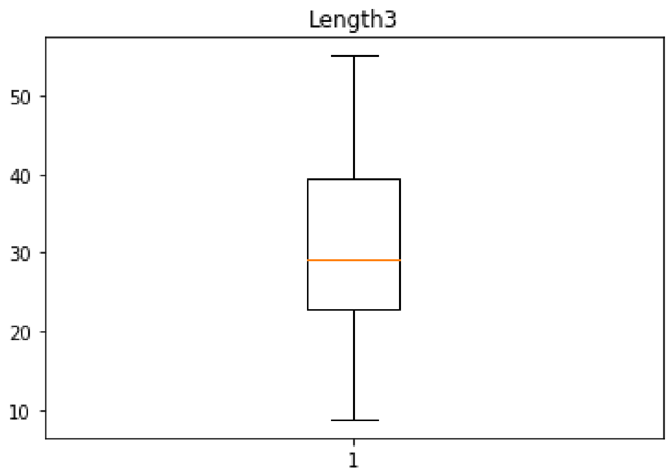
```
df_num=df.select_dtypes(np.number)
print('Outliers in the data :')
for features in df_num.columns:
    plt.boxplot(df_num[features])
    plt.title(features)
    plt.show()
```

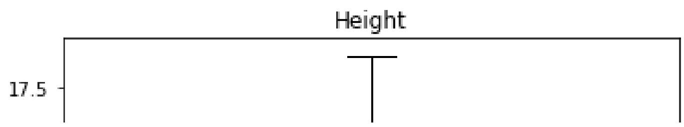
```
Outliers in the data :
```

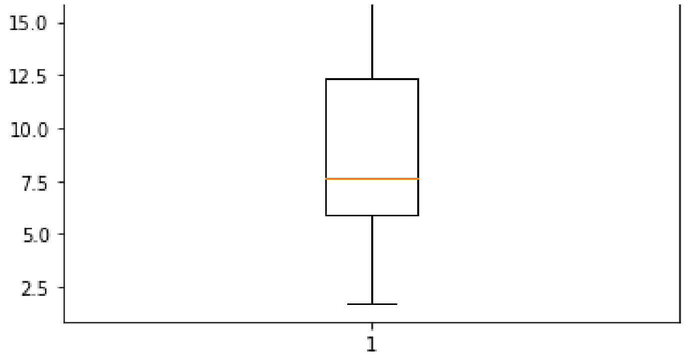


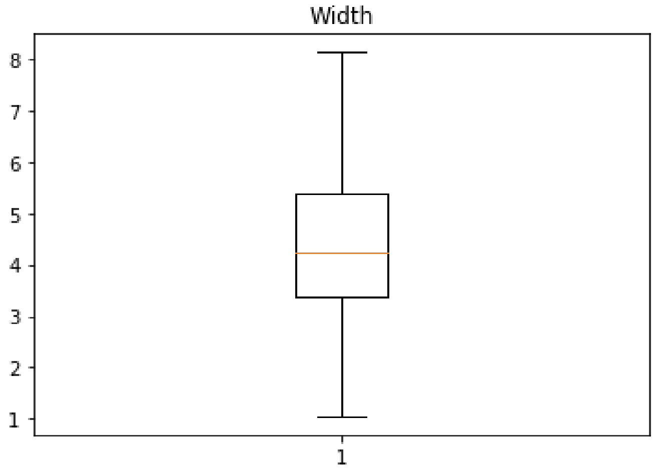












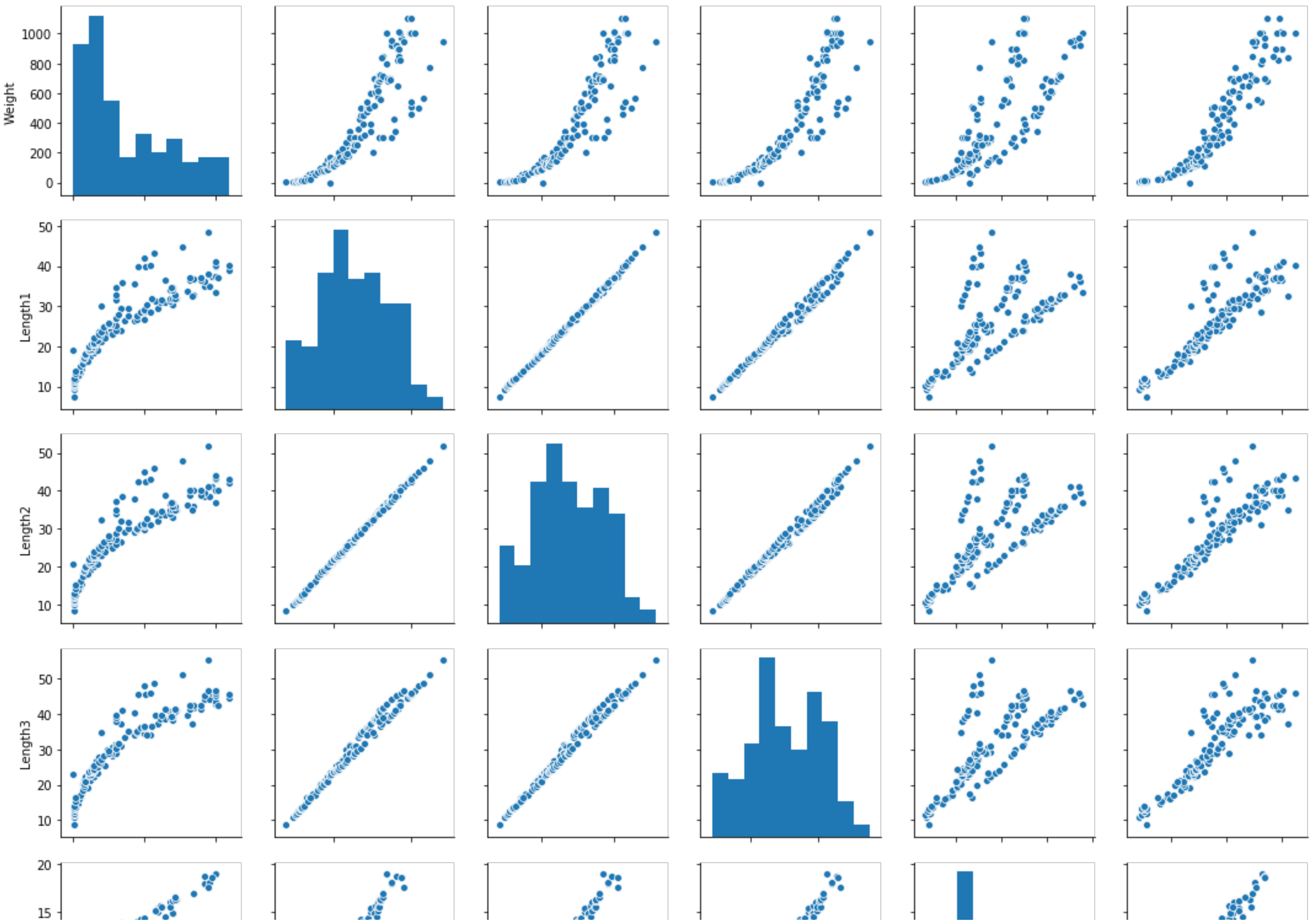
In [14]:

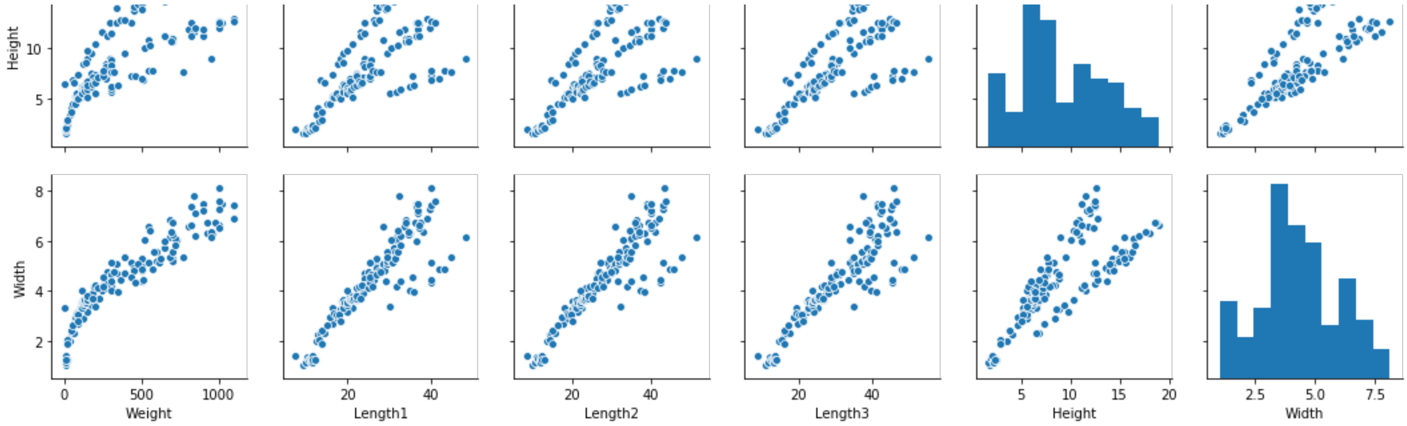
```
# b. Visualize the data using relevant plots.
# Find out the variables which are highly correlated with target variable? (3 marks)

sns.pairplot(df)
```

Out[14]:

```
<seaborn.axisgrid.PairGrid at 0x7f4de3e44978>
```





In [15]:

```
df.corr()
```

Out[15]:

	Weight	Length1	Length2	Length3	Height	Width
Weight	1.000000	0.890173	0.894415	0.902576	0.815237	0.925664
Length1	0.890173	1.000000	0.999385	0.990205	0.688050	0.891260
Length2	0.894415	0.999385	1.000000	0.992765	0.704041	0.897855
Length3	0.902576	0.990205	0.992765	1.000000	0.768038	0.898315
Height	0.815237	0.688050	0.704041	0.768038	1.000000	0.803268
Width	0.925664	0.891260	0.897855	0.898315	0.803268	1.000000

In [17]:

```
sns.heatmap(df.corr(),annot=True)
```

Out[17]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f4de05f8c50>

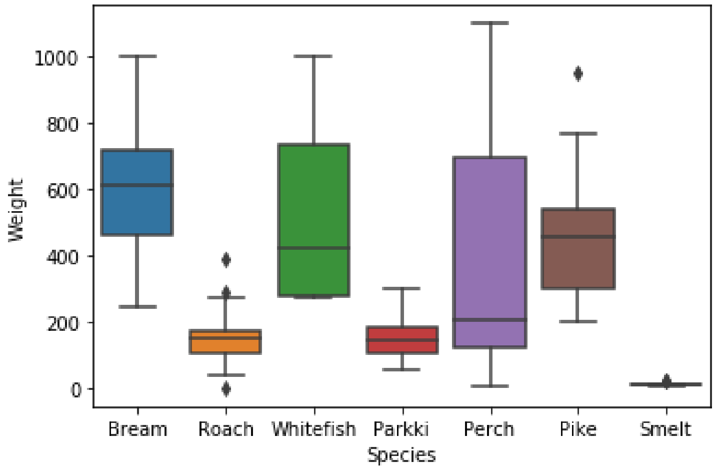
- **From the above analysis, the below variables are highly correlated withe target variable**
- **Length1**
- **Length2**
- **Length3**
- **Height**
- **Width**

In [21]:

```
sns.boxplot(df['Species'],df['Weight'])
```

Out[21]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f4de0517358>



- **From the above boxplot on categorical column we could clearly see the variation in the mean weight of different categories, which shows that the Species columns contributes to the weight of the fish.**

In [22]:

```
# c. Do you want to exclude some variables from the model based on this analysis?
# What other actions will you take? (2 marks)
```

- **From the above plots and correlation matrix it is clearly visible that length 1,length 2 , length 3 are highly correlated and could tend to have Multicollinearity with a correlation of close to 1 and adding redundancy to the data.**
- **Hence removing the redundant columns of length1 and length2 will be appropriate to reduce the redundancy**

Checking Multicoolinearity

In [33]:

```
df_samp=df_num.drop(columns=['Weight'])
```

In [34]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif['features']=df_samp.columns
vif['vif']=[variance_inflation_factor(df_samp.values,i)  for i in range(df_samp.shape[1]
)]
vif
```

Out[34]:

	features	vif
0	Length1	11640.943653
1	Length2	15183.972890
2	Length3	3093.805839
3	Height	75.546906
4	Width	96.285071

- **From VIF factor it is clearly visible that length 1,length2,length3 are having high multicollinearity with each other.**

In [37]:

```
# d. Split dataset into train and test (70:30). Are both train and test representative of
```

```
the overall data?
# How would you ascertain this statistically? (2 marks)

df_dummy=pd.get_dummies(data=df,columns=['Species'],drop_first=True)
```

In [42]:

```
x=df_dummy.drop(columns='Weight')
y=df_dummy['Weight']
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=0)
```

In [43]:

```
x_train.mean()
```

Out[43]:

```
Length1              25.294444
Length2              27.389815
Length3              30.183333
Height                8.805746
Width                 4.288494
Species_Parkki        0.083333
Species_Perch         0.342593
Species_Pike          0.101852
Species_Roach         0.129630
Species_Smelt         0.092593
Species_Whitefish     0.027778
dtype: float64
```

In [45]:

```
x_test.mean()
```

Out[45]:

```
Length1              25.925532
Length2              28.097872
Length3              30.842553
Height                9.248279
Width                 4.520638
Species_Parkki        0.042553
Species_Perch         0.404255
Species_Pike          0.042553
Species_Roach         0.127660
Species_Smelt         0.085106
Species_Whitefish     0.063830
dtype: float64
```

- **From the above mean analysis it is clear that the mean of all the training variables is almost close to the mean of all the test variables.**
- **Hence we can conclude that the splitted data is the correct representative of the original data.**

3. Model Building (15 marks)

a. Fit a base model and observe the overall R- Squared, RMSE and MAPE values of the model. Please comment on whether it is good or not. (3 marks)

b. Check for multi-collinearity and treat the same. (2 marks)

c. How would you improve the model? Write clearly the changes that you will make before refitting the model. Fit the final model. (6 marks)

d. Write down a business interpretation/explanation of the model – which variables are affecting the target the most and explain the relationship. Feel free to use charts or graphs to explain. (2 marks)

e. What changes from the base model had the most effect on model performance? (2 marks)

In [46]:

```
# a. Fit a base model and observe the overall R- Squared, RMSE and MAPE values of the mod
el.
# Please comment on whether it is good or not.   (3 marks)

from sklearn.linear_model import LinearRegression
```

In [47]:

```
base_model=LinearRegression()
```

In [48]:

```
base_model.fit(x_train,y_train)
```

Out[48]:

```
LinearRegression()
```

In [49]:

```
y_pred=base_model.predict(x_test)
```

In [50]:

```
from sklearn.metrics import r2_score,mean_squared_error
print('R2 score of the base model is',r2_score(y_test,y_pred))
```

```
R2 score of the base model is 0.9290256329220582
```

In [51]:

```
mse=mean_squared_error(y_test,y_pred)
rmse=np.sqrt(mse)
print('RMSE score of the base model is',rmse)
```

```
RMSE score of the base model is 81.25878675299707
```

In [100]:

```
y_pred_train=base_model.predict(x_train)
```

In [102]:

```
mse=mean_squared_error(y_train,y_pred_train)
rmse=np.sqrt(mse)
print('RMSE score of the base model is',rmse)
```

```
RMSE score of the base model is 71.37047100437793
```

In [104]:

```
mape=np.mean(np.abs((y_test-y_pred)/y_test))*100
print('MAPE score of the base model is',mape)
```

```
MAPE score of the base model is inf
```

- **From the above analysis the r2 score is close to 92% which is good enough**
- **the rmse score is 81.25 which is the variance error and 71.37 is the bias error which not shows much of overfitting in the model**

In [54]:

```
# b. Check for multi-collinearity and treat the same. (2 marks)
```

In [55]:

```
df_samp=df_num.drop(columns=['Weight'])
```

In [56]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif['features']=df_samp.columns
vif['vif']=[variance_inflation_factor(df_samp.values,i)  for i in range(df_samp.shape[1]
)]
vif
```

Out[56]:

	features	vif
0	Length1	11640.943653
1	Length2	15183.972890
2	Length3	3093.805839
3	Height	75.546906
4	Width	96.285071

In [59]:

```
df_samp=df_num.drop(columns=['Weight','Length1','Length2'])
```

In [60]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif['features']=df_samp.columns
vif['vif']=[variance_inflation_factor(df_samp.values,i)  for i in range(df_samp.shape[1]
)]
vif
```

Out[60]:

	features	vif
0	Length3	41.909181
1	Height	15.220064
2	Width	48.769166

In [61]:

```
df_samp=df_num.drop(columns=['Weight','Length1','Length2','Length3'])
```

In [62]:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif['features']=df_samp.columns
vif['vif']=[variance_inflation_factor(df_samp.values,i)  for i in range(df_samp.shape[1]
)]
vif
```

Out[62]:

	features	vif
0	Height	14.869484
1	Width	14.869484

- **Removed all the multicollinearity columns and still multicollinearity can be seen hence we can go with Lasso to reduce the effect of Multicollinearity.**

c. How would you improve the model?

Write clearly the changes that you will make before refitting the model. Fit the final model. (6 marks)

- **We can use the feature selection technique to find the significant features in the model and refit the model to improve the performance of the model**

In [72]:

```
from sklearn.feature_selection import RFE
```

In [80]:

```
lr=LinearRegression()
rfe_feat=RFE(estimator=lr,n_features_to_select=5)
```

In [81]:

```
rfe_sel_feat=rfe_feat.fit(x,y)
```

In [84]:

```
sel_feat=rfe_sel_feat.get_support()
```

In [87]:

```
sel_feat=list(x.columns[sel_feat])
```

In [88]:

```
print('the selected features are : ',sel_feat)
```

```
the selected features are :  ['Width', 'Species_Perch', 'Species_Roach', 'Species_Smelt', 'Species_Whitefish']
```

- **the feature selection technique removed the insignificant features and the multicollinear features from the model and now the model can be built with the selected features**

In [105]:

```
final_model=LinearRegression()
```

In [106]:

```
final_model.fit(x_train,y_train)
```

Out[106]:

```
LinearRegression()
```

In [107]:

```
y_pred_final=final_model.predict(x_test)
```

In [108]:

```
print('the R2 score of the final model is:',r2_score(y_test,y_pred_final))
```

```
the R2 score of the final model is: 0.9290256329220582
```

In [109]:

```
mse=mean_squared_error(y_test,y_pred_final)
rmse=np.sqrt(mse)
print('RMSE score of the base model is',rmse)
```

```
RMSE score of the base model is 81.25878675299707
```

In [95]:

```
# d. Write down a business interpretation/explanation of the model -
# which variables are affecting the target the most and explain the relationship.
# Feel free to use charts or graphs to explain. (2 marks)

import statsmodels.api as sm
xc=sm.add_constant(x)
model=sm.OLS(y,x).fit()
model.summary()
```

```
/home/deploy/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2389: Future
Warning: Method .ptp is deprecated and will be removed in a future version. Use numpy.ptp
instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[95]:

OLS Regression Results

Dep. Variable:	Weight	**R-squared (uncentered):**	0.970
Model:	OLS	**Adj. R-squared (uncentered):**	0.968
Method:	Least Squares	**F-statistic:**	427.4
Date:	Fri, 18 Sep 2020	**Prob (F-statistic):**	5.13e-104
Time:	10:13:23	**Log-Likelihood:**	-905.49
No. Observations:	155	**AIC:**	1833.
Df Residuals:	144	**BIC:**	1866.
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Length1	62.4023	34.681	1.799	0.074	-6.147	130.952
Length2	103.2937	41.019	2.518	0.013	22.216	184.372
Length3	-147.6639	18.715	-7.890	0.000	-184.655	-110.673
Height	22.5274	12.551	1.795	0.075	-2.280	47.335
Width	117.7837	20.385	5.778	0.000	77.491	158.077
Species_Parkki	-331.2196	29.444	-11.249	0.000	-389.417	-273.022
Species_Perch	-474.3285	41.545	-11.417	0.000	-556.446	-392.211
Species_Pike	-357.5681	102.967	-3.473	0.001	-561.090	-154.046
Species_Roach	-331.5962	37.179	-8.919	0.000	-405.083	-258.110
Species_Smelt	-205.4095	36.447	-5.636	0.000	-277.449	-133.370
Species_Whitefish	-304.1895	51.376	-5.921	0.000	-405.738	-202.641

Omnibus:	9.252	**Durbin-Watson:**	0.827
Prob(Omnibus):	0.010	**Jarque-Bera (JB):**	10.599
Skew:	0.432	**Prob(JB):**	0.00499
Kurtosis:	3.946	**Cond. No.**	905.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

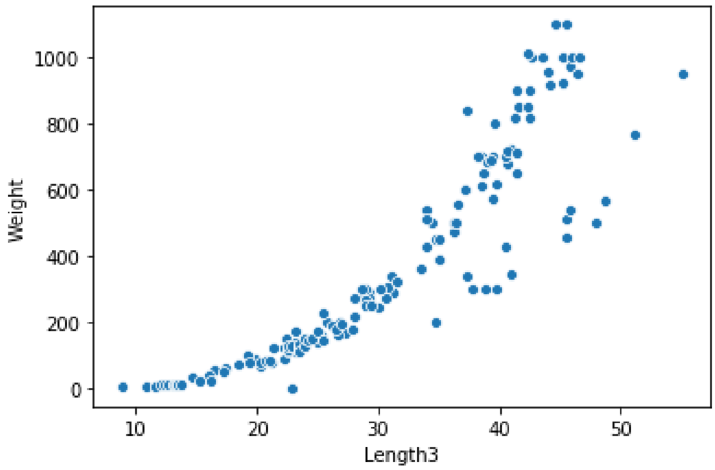
- **the weight of the fish increases by 62.4 grams from the average weight for a unit change of 1 cm in length.**
- **the width can have a high high correlation as weight increases by 117.7 grams for a unit change of 1 cm in length**

In [97]:

```
sns.scatterplot(x['Length3'],y)
```

Out[97]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f4d7dee89e8>

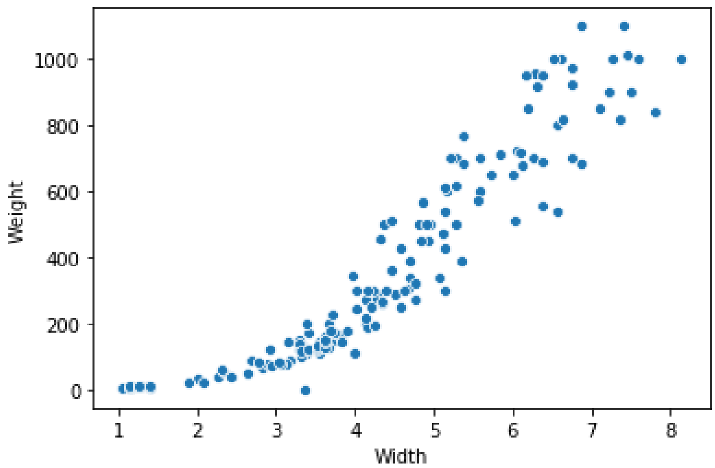


In [98]:

```
sns.scatterplot(x['Width'],y)
```

Out[98]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f4d7deadb38>

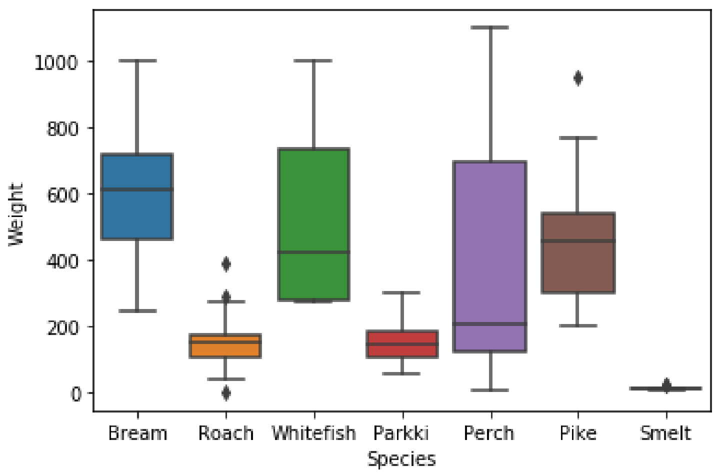


In [99]:

```
sns.boxplot(df['Species'],df['Weight'])
```

Out[99]:

<matplotlib.axes._subplots.AxesSubplot at 0x7f4d7de747b8>



- **From the above boxplot on categorical column we could clearly see the variation in the mean weight of different categories, which shows that the Species columns contributes to the weight of the fish.**

e. What changes from the base model had the most effect on model performance? (2 marks)

- **choosing the significant predictors from the model and significant features from the model is increasing the performance of the model.**
- **removing the multicollinearity columns from the model will give a generalized and reliable model to predict the weigth of the fish.**

In [ ]:

```
```