2440016804 - Rio Pramana - LA01 - Assignment 4

1. Do Exploratory Data Analysis for Fish Market dataset

Import libraries and load dataset

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

In [2]:
```
# Importing the dataset, downloaded file is in the same folder
csv_path = "fish.csv"
fish_df = pd.read_csv(csv_path)
```

Check dataset (Shape, Info)

In [3]:
```
fish_df.head(5)
```

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_df.shape
```

Out[4]:
```
(159, 7)
```

In [5]:
```
fish_df.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
```

No anomaly in any of the data type

Check missing value

In [6]:
```
fish_df.isnull().sum()
```

Out[6]:
```
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64
```

There is no missing value

Data Summarization

In [7]:
```
fish_df.describe()
```

Out[7]:

	Weight	Length1	Length2	Length3	Height	Width
count	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000

	Weight	Length1	Length2	Length3	Height	Width
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

There is a **weird value** on the **Weight** column. There is a fish with 0 Weight, that shouldn't make sense. To handle this, we will remove the outlier (row with Weight == 0)

Checking which row has Weight == 0 :

In [8]:
```
fish_df.loc[fish_df['Weight'] == 0]
```

Out[8]:

	Species	Weight	Length1	Length2	Length3	Height	Width
40	Roach	0.0	19.0	20.5	22.8	6.4752	3.3516

Only row with index 40 has this weird value of Weight, so we will remove it.

In [9]:
```
# Removing the outlier
i = fish_df[fish_df.Weight == 0].index #Take the index
new_fish_df = fish_df.drop(i) #Drop the row
```

In [10]:
```
new_fish_df.shape
```

Out[10]:
```
(158, 7)
```

In [11]:
```
new_fish_df.describe()
```

Out[11]:

	Weight	Length1	Length2	Length3	Height	Width
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000
mean	400.847468	26.293038	28.465823	31.280380	8.986790	4.424232
std	357.697796	10.011427	10.731707	11.627605	4.295191	1.689010
min	5.900000	7.500000	8.400000	8.800000	1.728400	1.047600
25%	121.250000	19.150000	21.000000	23.200000	5.940600	3.398650
50%	281.500000	25.300000	27.400000	29.700000	7.789000	4.277050
75%	650.000000	32.700000	35.750000	39.675000	12.371850	5.586750
max	1650.000000	59.000000	63.400000	68.000000	18.957000	8.142000

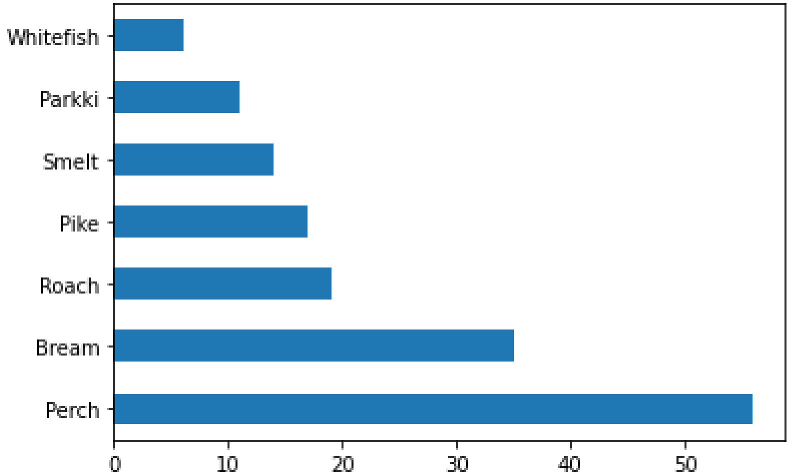
The outlier has been removed and the dataset is more normal

Plotting number of fish for each species on a bar graph

In [12]:

```
new_fish_df.Species.value_counts().plot(kind = "barh")
```

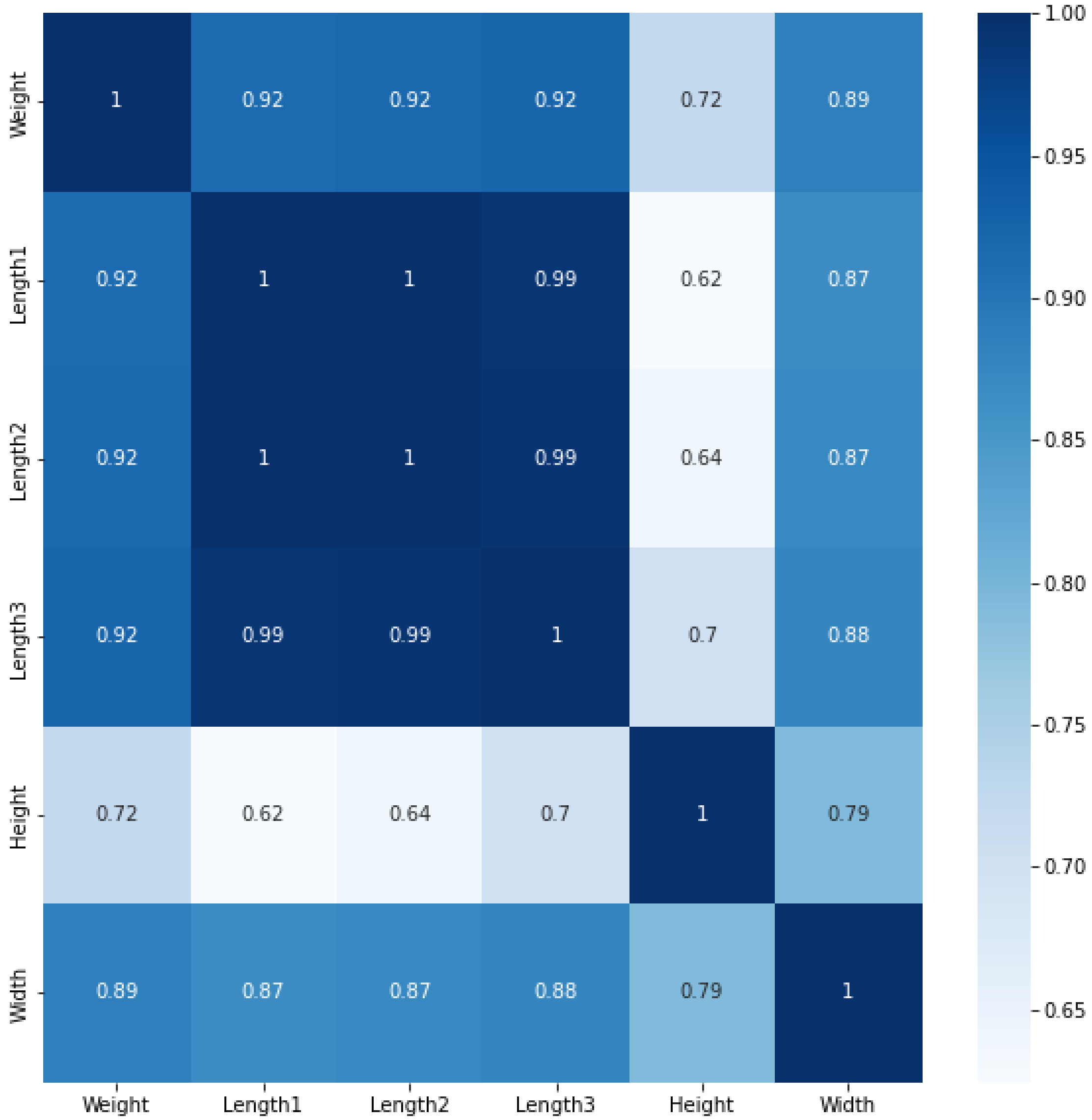
Out[12]: <AxesSubplot:>



Checking the correlation between data

In [13]:
```
plt.figure(figsize=(10,10))
sns.heatmap(new_fish_df.corr(),cbar=True,annot=True,cmap='Blues')
```

Out[13]: <AxesSubplot:>

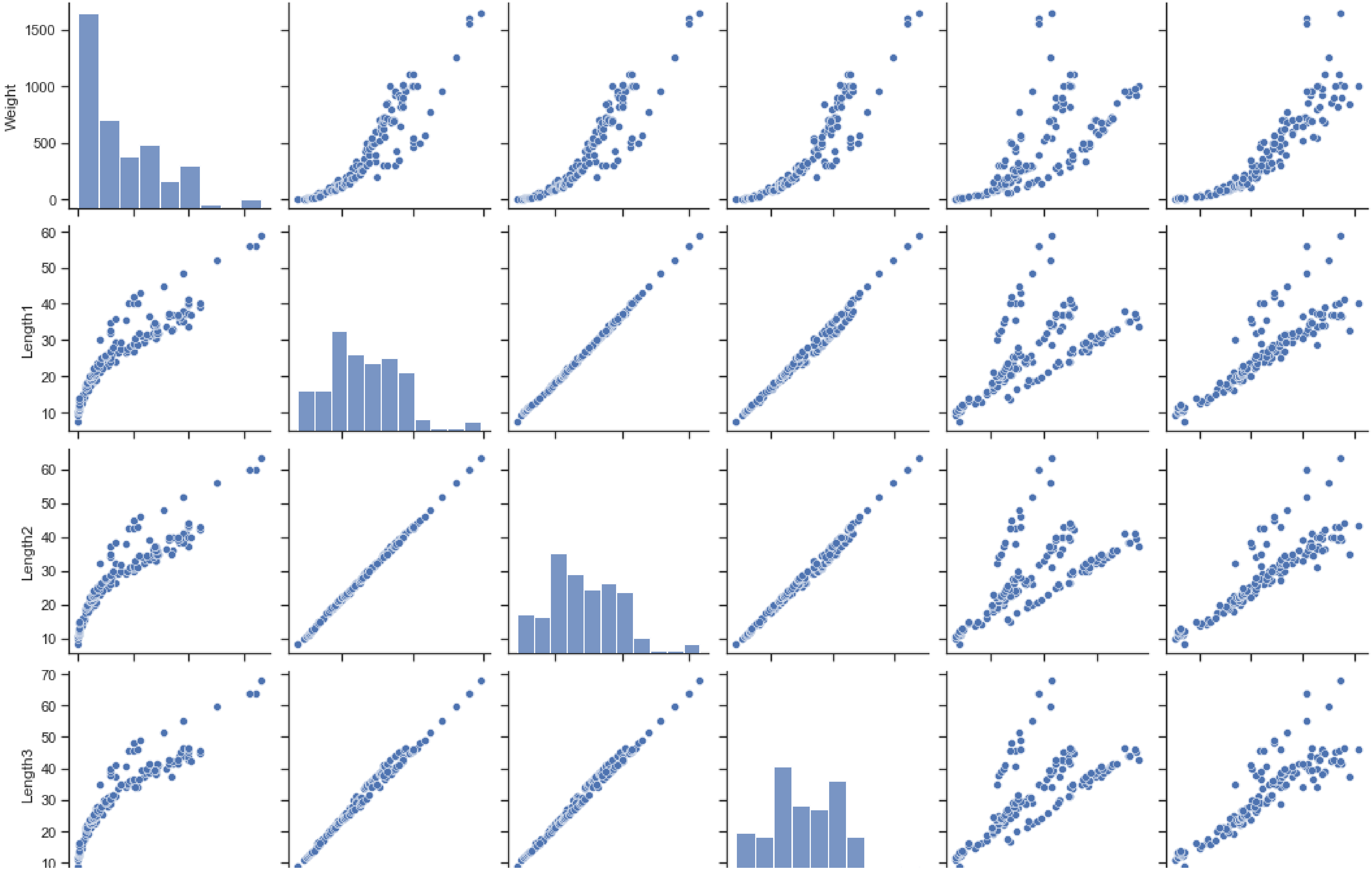


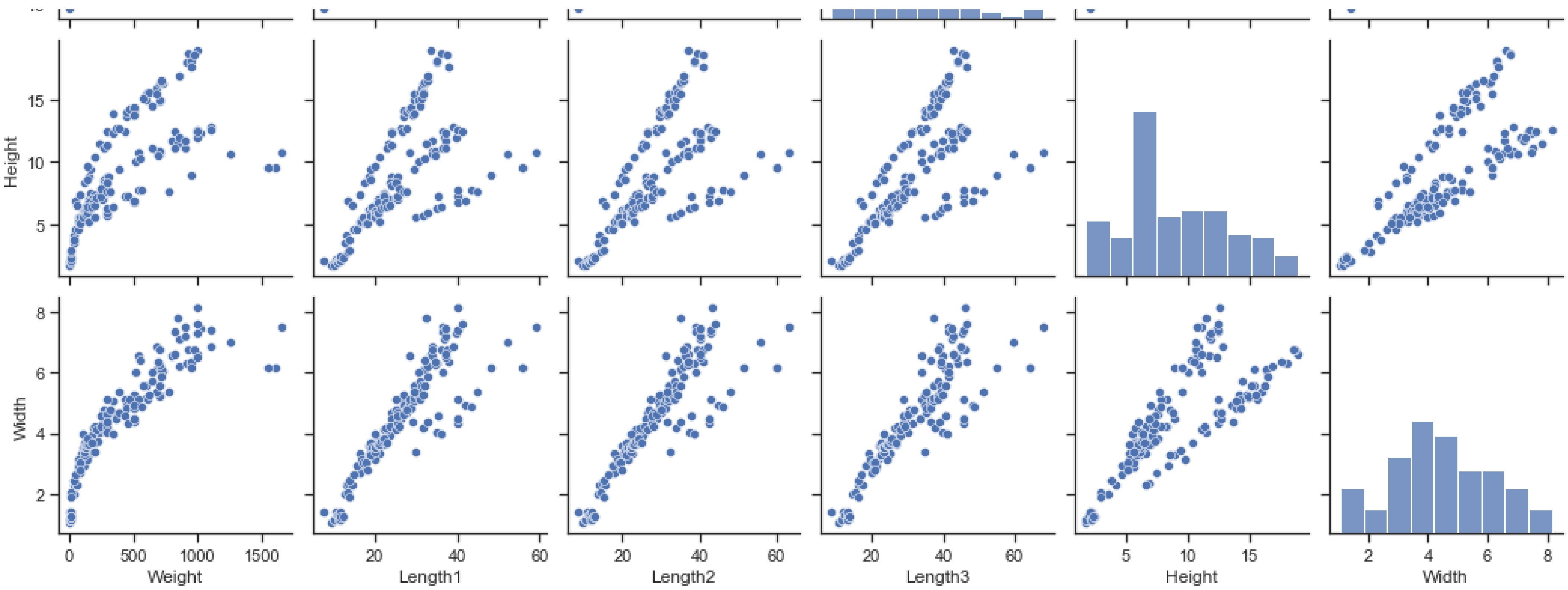
From the correlation matrix above, we can see that Weight have positive correlations with every other variable

2. Please check Pairwise Relationships in a dataset using Pairplot

In [14]:

```
sns.set(style="ticks", color_codes=True)
g = sns.pairplot(new_fish_df)
plt.show()
```





From the correlation matrix and pairwise relationships shown, we can see the positive correlation where the Weight will increase as the other variable increases

3. Prepared Training and Test Dataset

To make feature extraction easier, we will move the dependent variable (Weight) to be the last column

In [15]:

```
fish_list = new_fish_df.copy()
cols_at_end = ['Weight']
fish_list = fish_list[[c for c in fish_list if c not in cols_at_end]
        + [c for c in cols_at_end if c in fish_list]]
fish_list.head(5)
```

Out[15]:

	Species	Length1	Length2	Length3	Height	Width	Weight
0	Bream	23.2	25.4	30.0	11.5200	4.0200	242.0
1	Bream	24.0	26.3	31.2	12.4800	4.3056	290.0
2	Bream	23.9	26.5	31.1	12.3778	4.6961	340.0

	Species	Length1	Length2	Length3	Height	Width	Weight
3	Bream	26.3	29.0	33.5	12.7300	4.4555	363.0
4	Bream	26.5	29.0	34.0	12.4440	5.1340	430.0

Extracting independent and dependent variables

In [16]:

```
#Extracting independent variables:
x = fish_list.iloc[:, :-1].values #Extract semua kolom kecuali kolom terakhir
print(x)
```

```
[['Bream' 23.2 25.4 30.0 11.52 4.02]
 ['Bream' 24.0 26.3 31.2 12.48 4.3056]
 ['Bream' 23.9 26.5 31.1 12.3778 4.6961]
 ['Bream' 26.3 29.0 33.5 12.73 4.4555]
 ['Bream' 26.5 29.0 34.0 12.444 5.134]
 ['Bream' 26.8 29.7 34.7 13.6024 4.9274]
 ['Bream' 26.8 29.7 34.5 14.1795 5.2785]
 ['Bream' 27.6 30.0 35.0 12.67 4.69]
 ['Bream' 27.6 30.0 35.1 14.0049 4.8438]
 ['Bream' 28.5 30.7 36.2 14.2266 4.9594]
 ['Bream' 28.4 31.0 36.2 14.2628 5.1042]
 ['Bream' 28.7 31.0 36.2 14.3714 4.8146]
 ['Bream' 29.1 31.5 36.4 13.7592 4.368]
 ['Bream' 29.5 32.0 37.3 13.9129 5.0728]
 ['Bream' 29.4 32.0 37.2 14.9544 5.1708]
 ['Bream' 29.4 32.0 37.2 15.438 5.58]
 ['Bream' 30.4 33.0 38.3 14.8604 5.2854]
 ['Bream' 30.4 33.0 38.5 14.938 5.1975]
 ['Bream' 30.9 33.5 38.6 15.633 5.1338]
 ['Bream' 31.0 33.5 38.7 14.4738 5.7276]
 ['Bream' 31.3 34.0 39.5 15.1285 5.5695]
 ['Bream' 31.4 34.0 39.2 15.9936 5.3704]
 ['Bream' 31.5 34.5 39.7 15.5227 5.2801]
 ['Bream' 31.8 35.0 40.6 15.4686 6.1306]
 ['Bream' 31.9 35.0 40.5 16.2405 5.589]
 ['Bream' 31.8 35.0 40.9 16.36 6.0532]
 ['Bream' 32.0 35.0 40.6 16.3618 6.09]
 ['Bream' 32.7 36.0 41.5 16.517 5.8515]
 ['Bream' 32.8 36.0 41.6 16.8896 6.1984]
 ['Bream' 33.5 37.0 42.6 18.957 6.603]
```

```
['Bream' 35.0 38.5 44.1 18.0369 6.3063]
['Bream' 35.0 38.5 44.0 18.084 6.292]
['Bream' 36.2 39.5 45.3 18.7542 6.7497]
['Bream' 37.4 41.0 45.9 18.6354 6.7473]
['Bream' 38.0 41.0 46.5 17.6235 6.3705]
['Roach' 12.9 14.1 16.2 4.1472 2.268]
['Roach' 16.5 18.2 20.3 5.2983 2.8217]
['Roach' 17.5 18.8 21.2 5.5756 2.9044]
['Roach' 18.2 19.8 22.2 5.6166 3.1746]
['Roach' 18.6 20.0 22.2 6.216 3.5742]
['Roach' 19.1 20.8 23.1 6.1677 3.3957]
['Roach' 19.4 21.0 23.7 6.1146 3.2943]
['Roach' 20.4 22.0 24.7 5.8045 3.7544]
['Roach' 20.5 22.0 24.3 6.6339 3.5478]
['Roach' 20.5 22.5 25.3 7.0334 3.8203]
['Roach' 21.0 22.5 25.0 6.55 3.325]
['Roach' 21.1 22.5 25.0 6.4 3.8]
['Roach' 22.0 24.0 27.2 7.5344 3.8352]
['Roach' 22.0 23.4 26.7 6.9153 3.6312]
['Roach' 22.1 23.5 26.8 7.3968 4.1272]
['Roach' 23.6 25.2 27.9 7.0866 3.906]
['Roach' 24.0 26.0 29.2 8.8768 4.4968]
['Roach' 25.0 27.0 30.6 8.568 4.7736]
['Roach' 29.5 31.7 35.0 9.485 5.355]
['Whitefish' 23.6 26.0 28.7 8.3804 4.2476]
['Whitefish' 24.1 26.5 29.3 8.1454 4.2485]
['Whitefish' 25.6 28.0 30.8 8.778 4.6816]
['Whitefish' 28.5 31.0 34.0 10.744 6.562]
['Whitefish' 33.7 36.4 39.6 11.7612 6.5736]
['Whitefish' 37.3 40.0 43.5 12.354 6.525]
['Parkki' 13.5 14.7 16.5 6.8475 2.3265]
['Parkki' 14.3 15.5 17.4 6.5772 2.3142]
['Parkki' 16.3 17.7 19.8 7.4052 2.673]
['Parkki' 17.5 19.0 21.3 8.3922 2.9181]
['Parkki' 18.4 20.0 22.4 8.8928 3.2928]
['Parkki' 19.0 20.7 23.2 8.5376 3.2944]
['Parkki' 19.0 20.7 23.2 9.396 3.4104]
['Parkki' 19.8 21.5 24.1 9.7364 3.1571]
['Parkki' 21.2 23.0 25.8 10.3458 3.6636]
['Parkki' 23.0 25.0 28.0 11.088 4.144]
['Parkki' 24.0 26.0 29.0 11.368 4.234]
['Perch' 7.5 8.4 8.8 2.112 1.408]
['Perch' 12.5 13.7 14.7 3.528 1.9992]
['Perch' 13.8 15.0 16.0 3.824 2.432]
```

```
['Perch' 15.0 16.2 17.2 4.5924 2.6316]
['Perch' 15.7 17.4 18.5 4.588 2.9415]
['Perch' 16.2 18.0 19.2 5.2224 3.3216]
['Perch' 16.8 18.7 19.4 5.1992 3.1234]
['Perch' 17.2 19.0 20.2 5.6358 3.0502]
['Perch' 17.8 19.6 20.8 5.1376 3.0368]
['Perch' 18.2 20.0 21.0 5.082 2.772]
['Perch' 19.0 21.0 22.5 5.6925 3.555]
['Perch' 19.0 21.0 22.5 5.9175 3.3075]
['Perch' 19.0 21.0 22.5 5.6925 3.6675]
['Perch' 19.3 21.3 22.8 6.384 3.534]
['Perch' 20.0 22.0 23.5 6.11 3.4075]
['Perch' 20.0 22.0 23.5 5.64 3.525]
['Perch' 20.0 22.0 23.5 6.11 3.525]
['Perch' 20.0 22.0 23.5 5.875 3.525]
['Perch' 20.0 22.0 23.5 5.5225 3.995]
['Perch' 20.5 22.5 24.0 5.856 3.624]
['Perch' 20.5 22.5 24.0 6.792 3.624]
['Perch' 20.7 22.7 24.2 5.9532 3.63]
['Perch' 21.0 23.0 24.5 5.2185 3.626]
['Perch' 21.5 23.5 25.0 6.275 3.725]
['Perch' 22.0 24.0 25.5 7.293 3.723]
['Perch' 22.0 24.0 25.5 6.375 3.825]
['Perch' 22.6 24.6 26.2 6.7334 4.1658]
['Perch' 23.0 25.0 26.5 6.4395 3.6835]
['Perch' 23.5 25.6 27.0 6.561 4.239]
['Perch' 25.0 26.5 28.0 7.168 4.144]
['Perch' 25.2 27.3 28.7 8.323 5.1373]
['Perch' 25.4 27.5 28.9 7.1672 4.335]
['Perch' 25.4 27.5 28.9 7.0516 4.335]
['Perch' 25.4 27.5 28.9 7.2828 4.5662]
['Perch' 25.9 28.0 29.4 7.8204 4.2042]
['Perch' 26.9 28.7 30.1 7.5852 4.6354]
['Perch' 27.8 30.0 31.6 7.6156 4.7716]
['Perch' 30.5 32.8 34.0 10.03 6.018]
['Perch' 32.0 34.5 36.5 10.2565 6.3875]
['Perch' 32.5 35.0 37.3 11.4884 7.7957]
['Perch' 34.0 36.5 39.0 10.881 6.864]
['Perch' 34.0 36.0 38.3 10.6091 6.7408]
['Perch' 34.5 37.0 39.4 10.835 6.2646]
['Perch' 34.6 37.0 39.3 10.5717 6.3666]
['Perch' 36.5 39.0 41.4 11.1366 7.4934]
['Perch' 36.5 39.0 41.4 11.1366 6.003]
['Perch' 36.6 39.0 41.3 12.4313 7.3514]
```

```
['Perch' 36.9 40.0 42.3 11.9286 7.1064]
['Perch' 37.0 40.0 42.5 11.73 7.225]
['Perch' 37.0 40.0 42.4 12.3808 7.4624]
['Perch' 37.1 40.0 42.5 11.135 6.63]
['Perch' 39.0 42.0 44.6 12.8002 6.8684]
['Perch' 39.8 43.0 45.2 11.9328 7.2772]
['Perch' 40.1 43.0 45.5 12.5125 7.4165]
['Perch' 40.2 43.5 46.0 12.604 8.142]
['Perch' 41.1 44.0 46.6 12.4888 7.5958]
['Pike' 30.0 32.3 34.8 5.568 3.3756]
['Pike' 31.7 34.0 37.8 5.7078 4.158]
['Pike' 32.7 35.0 38.8 5.9364 4.3844]
['Pike' 34.8 37.3 39.8 6.2884 4.0198]
['Pike' 35.5 38.0 40.5 7.29 4.5765]
['Pike' 36.0 38.5 41.0 6.396 3.977]
['Pike' 40.0 42.5 45.5 7.28 4.3225]
['Pike' 40.0 42.5 45.5 6.825 4.459]
['Pike' 40.1 43.0 45.8 7.786 5.1296]
['Pike' 42.0 45.0 48.0 6.96 4.896]
['Pike' 43.2 46.0 48.7 7.792 4.87]
['Pike' 44.8 48.0 51.2 7.68 5.376]
['Pike' 48.3 51.7 55.1 8.9262 6.1712]
['Pike' 52.0 56.0 59.7 10.6863 6.9849]
['Pike' 56.0 60.0 64.0 9.6 6.144]
['Pike' 56.0 60.0 64.0 9.6 6.144]
['Pike' 59.0 63.4 68.0 10.812 7.48]
['Smelt' 9.3 9.8 10.8 1.7388 1.0476]
['Smelt' 10.0 10.5 11.6 1.972 1.16]
['Smelt' 10.1 10.6 11.6 1.7284 1.1484]
['Smelt' 10.4 11.0 12.0 2.196 1.38]
['Smelt' 10.7 11.2 12.4 2.0832 1.2772]
['Smelt' 10.8 11.3 12.6 1.9782 1.2852]
['Smelt' 11.3 11.8 13.1 2.2139 1.2838]
['Smelt' 11.3 11.8 13.1 2.2139 1.1659]
['Smelt' 11.4 12.0 13.2 2.2044 1.1484]
['Smelt' 11.5 12.2 13.4 2.0904 1.3936]
['Smelt' 11.7 12.4 13.5 2.43 1.269]
['Smelt' 12.1 13.0 13.8 2.277 1.2558]
['Smelt' 13.2 14.3 15.2 2.8728 2.0672]
['Smelt' 13.8 15.0 16.2 2.9322 1.8792]]
```

In [17]:

```
#Extracting dependent variable:
y = fish_list.iloc[:, 6].values #Extract kolom terakhir
```

```
print(y)
```

```
[ 242.   290.   340.   363.   430.   450.   500.   390.   450.   500.
  475.   500.   500.   340.   600.   600.   700.   700.   610.   650.
  575.   685.   620.   680.   700.   725.   720.   714.   850.  1000.
  920.   955.   925.   975.   950.    40.    69.    78.    87.   120.
  110.   120.   150.   145.   160.   140.   160.   169.   161.   200.
  180.   290.   272.   390.   270.   270.   306.   540.   800.  1000.
   55.    60.    90.   120.   150.   140.   170.   145.   200.   273.
  300.     5.9   32.    40.    51.5   70.   100.    78.    80.    85.
   85.   110.   115.   125.   130.   120.   120.   130.   135.   110.
  130.   150.   145.   150.   170.   225.   145.   188.   180.   197.
  218.   300.   260.   265.   250.   250.   300.   320.   514.   556.
  840.   685.   700.   700.   690.   900.   650.   820.   850.   900.
 1015.   820.  1100.  1000.  1100.  1000.  1000.   200.   300.   300.
  300.   430.   345.   456.   510.   540.   500.   567.   770.   950.
 1250.  1600.  1550.  1650.     6.7    7.5    7.     9.7    9.8    8.7
   10.     9.9    9.8   12.2   13.4   12.2   19.7   19.9]
```

From the independent variables shown, we can see that there is a categorical data (Species). So, we will have to encode it using One Hot Encoding

Encoding categorical data (Species)

In [18]:

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

In [19]:

```
ct = ColumnTransformer([("Species", OneHotEncoder(), [0])], remainder = 'passthrough')
#[0] menunjukkan kolom yang diencode
x = ct.fit_transform(x)
print(x)
```

```
[[1.0 0.0 0.0 ... 30.0 11.52 4.02]
 [1.0 0.0 0.0 ... 31.2 12.48 4.3056]
 [1.0 0.0 0.0 ... 31.1 12.3778 4.6961]
 ...
 [0.0 0.0 0.0 ... 13.8 2.277 1.2558]
 [0.0 0.0 0.0 ... 15.2 2.8728 2.0672]
 [0.0 0.0 0.0 ... 16.2 2.9322 1.8792]]
```

Split dataset menjadi training set dan test set

In [20]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.2, random_state=0)
```

We choose to split the dataset into 80/20 set because usually that's the standard. Another reason is that the dataset only contains 159 rows (158 after removing outlier), so taking 20% as the test size is good because we get a good amount of data for testing (around 30-31 data). Taking only a small amount of data for testing (Say 10-15 data) could be risky because if these 10-15 data points are from the most abnormal regions of the dataset, the model will perform worse.

5. Predict Weight Fish each Species

In [21]:
```
#Fitting the MLR model to the training set:
from sklearn.linear_model import LinearRegression
regressor= LinearRegression()
regressor.fit(x_train, y_train)
```

Out[21]:
```
LinearRegression()
```

In [22]:
```
#Predicting the Test set and Training set result;
y_test_pred= regressor.predict(x_test)
y_train_pred= regressor.predict(x_train)
```

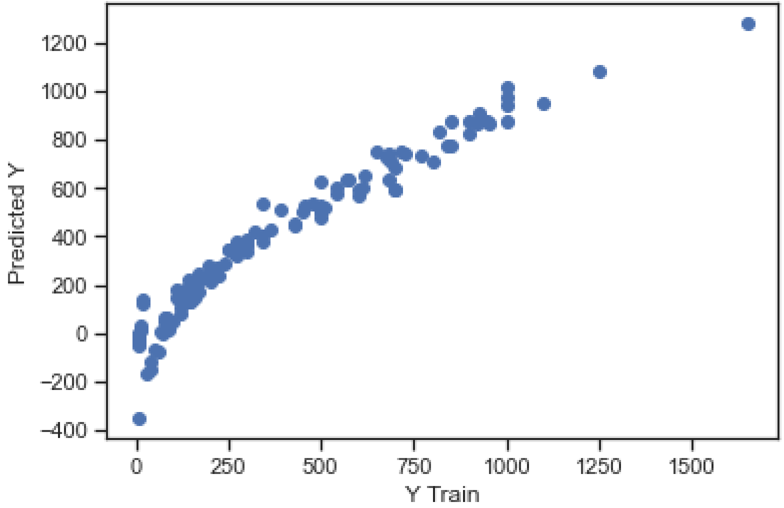
In [23]:
```
#check the score for training dataset and test dataset
print('Train Score: ', regressor.score(x_train, y_train))
print('Test Score: ', regressor.score(x_test, y_test))
```

```
Train Score:  0.9377625177306101
Test Score:  0.885683370845778
```

6. Plot with scatter of predict result

Scatter plot for y_training

In [24]:
```
plt.scatter(y_train, y_train_pred)
plt.xlabel('Y Train')
plt.ylabel('Predicted Y')
plt.show()
```



Scatter plot for y_test

In [25]:

```
plt.scatter(y_test, y_test_pred)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.show()
```