INSTRUCTIONS

Every learner should submit his/her own homework solutions. However, you are allowed to discuss the homework with each other– but everyone must submit his/her own solution; you may not copy someone else's solution.

The homework consists of two parts: 1. Data manipulation 2. Exploratory Data Analysis

Follow the prompts in the attached jupyter notebook. Download the data and place it in your working directory, or modify the path to upload it to your notebook. Add markdown cells to your analysis to include your solutions, comments, answers. Add as many cells as you need, for easy readability comment when possible.Make sure that you run your whole notebook before saving and sending it to us.

Hopefully this homework will help you develop skills, make you understand the flow of an EDA, get you ready for individual work.

Submission: Send in both a ipynb and a pdf file of your work.

Good luck!

Part1: Cleaning, wrangling data

Data cleaning focuses on removing inaccurate data from your data set whereas data wrangling focuses on transforming the data's format, typically by converting "raw" data into another format more suitable for use. This excersize uses the traffic_cameras file. Your task is to follow prompts to change, modify your data. Try your best!

In [1]:
```
#read in libraries
import numpy as np
from sklearn.datasets import load_iris
from sklearn import preprocessing
import pandas as pd
```

In [2]:
```
# Reading the CSV file
df = pd.read_csv("traffic_cameras.csv")

# Printing top 5 rows
df.head()
```

Out[2]:

	Camera ID	Location Name	Camera Status	Turn on Date	Camera Manufacturer	ATD Location ID	Landmark	Sig Engine Ar
0	370	PLEASANT VALLEY RD / NUCKOLS CROSSING RD	TURNED_ON	5/24/2018	Advidia	LOC16-003180	NaN	SOUTHEA
1	379	BARTON SPRINGS RD / KINNEY AVE	TURNED_ON	5/21/2018	Advidia	LOC16-000640	NaN	SOUTHWE
2	404	SPRINGDALE RD / OAK SPRINGS DR	TURNED_ON	6/7/2018	Advidia	LOC16-000800	NaN	NORTHEA
3	447	BRAKER LN / STONELAKE BLVD	TURNED_ON	9/9/2016	Advidia	LOC16-003740	NaN	NORTHWE
4	552	EXPOSITION BLVD / WESTOVER RD	TURNED_ON	2/24/2020	Advidia	LOC16-003710	NaN	CENTR

5 rows × 28 columns

1. How many rows and columns does your data have?

In [3]:
```
### Your code goes here ###
print("Number of rows: " + str(df.shape[0]))
print("Number of columns: " + str(df.shape[1]))
```

```
Number of rows: 802
Number of columns: 28
```

2. What can you tell us about the type of variables we have?

In [4]:
```
### Your code goes here ###
print(df.dtypes)
```

```
Camera ID                          int64
Location Name                     object
Camera Status                     object
Turn on Date                      object
Camera Manufacturer               object
ATD Location ID                   object
Landmark                          object
Signal Engineer Area              object
Council District                  object
Jurisdiction                      object
Location Type                     object
Primary St Segment ID            float64
Cross St Segment ID              float64
Primary Street Block             float64
Primary Street                    object
PRIMARY_ST_AKA                   float64
Cross Street Block               float64
Cross Street                      object
CROSS_ST_AKA                     float64
COA Intersection ID              float64
Modified Date                     object
IP Comm Status                    object
IP Comm Status Date and Time      object
Published Screenshots            float64
Screenshot Address                object
Funding                           object
ID                                object
Location                          object
dtype: object
```

In [5]:
```
### Your asnwer should go in here ### change the cell to markdown
```

3. Delete only the columns that have all null values, name it df1 (nothing else, but null)

In [6]:
```
### Your code goes here ###
df1 = df
for column in df1.columns:
    if df1[column].isnull().all():
        del df1[column]
```

In [7]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 802 entries, 0 to 801
Data columns (total 23 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Camera ID                   802 non-null    int64
 1   Location Name               802 non-null    object
 2   Camera Status               802 non-null    object
 3   Turn on Date                442 non-null    object
 4   Camera Manufacturer         646 non-null    object
 5   ATD Location ID             802 non-null    object
 6   Landmark                    94 non-null     object
 7   Signal Engineer Area        799 non-null    object
 8   Council District            790 non-null    object
 9   Jurisdiction                799 non-null    object
 10  Location Type               802 non-null    object
 11  Primary Street Block        800 non-null    float64
 12  Primary Street              801 non-null    object
 13  Cross Street Block          757 non-null    float64
```

4. Dropp columns that have (any) null values name it df2

In [8]:
```
### Your code goes here ###
df2 = df.dropna(how="any", axis=1)
```

In [9]:
```
df2.info()
df2.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 802 entries, 0 to 801
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype 
---  ------                          --------------  ----- 
 0   Camera ID                       802 non-null    int64 
 1   Location Name                   802 non-null    object
 2   Camera Status                   802 non-null    object
 3   ATD Location ID                 802 non-null    object
 4   Location Type                   802 non-null    object
 5   Modified Date                   802 non-null    object
 6   IP Comm Status                  802 non-null    object
 7   IP Comm Status Date and Time    802 non-null    object
 8   Screenshot Address              802 non-null    object
 9   ID                              802 non-null    object
 10  Location                        802 non-null    object
dtypes: int64(1), object(10)
memory usage: 69.0+ KB
```

Out[9]: (802, 11)

5. Rename column names in df2 so they are more usable (name the new dataframe df3) to the followings: cam_id, loc_name, cam_stat, atd_loc_id, loc_type, date, comm_stat, comm_stat_date, screen_addr, id, location

In [10]:
```
### Your code goes here ###
df3 = df2.rename(columns={'Camera ID': 'cam_id', 'Location Name':'loc_name','C
```

In [11]:
```
df3.head()
```

Out[11]:

	cam_id	loc_name	cam_stat	atd_loc_id	loc_type	date	comm_stat	comm_sta
0	370	PLEASANT VALLEY RD / NUCKOLS CROSSING RD	TURNED_ON	LOC16-003180	ROADWAY	10/28/2021 08:40:00 AM +0000	ONLINE	10/2 08:30
1	379	BARTON SPRINGS RD / KINNEY AVE	TURNED_ON	LOC16-000640	ROADWAY	10/29/2021 08:45:00 AM +0000	ONLINE	10/2 08:35
2	404	SPRINGDALE RD / OAK SPRINGS DR	TURNED_ON	LOC16-000800	ROADWAY	10/29/2021 07:38:00 PM +0000	ONLINE	10/2 08:35
3	447	BRAKER LN / STONELAKE BLVD	TURNED_ON	LOC16-003740	ROADWAY	10/29/2021 07:49:00 PM +0000	ONLINE	10/2 08:35
4	552	EXPOSITION BLVD / WESTOVER RD	TURNED_ON	LOC16-003710	ROADWAY	10/29/2021 07:47:00 PM +0000	ONLINE	10/2 08:35

6. Split "date" column into two new columns within df3 ('Dates' and 'Time') /modify df3 data/

In [12]:
```
### Your code goes here ###
df3[['Dates', 'Time']] = df3["date"].apply(lambda x: pd.Series(str(x).split("
del df3['date']
```

In [13]:
```
df3.head()
```

Out[13]:

	cam_id	loc_name	cam_stat	atd_loc_id	loc_type	comm_stat	comm_stat_date	
0	370	PLEASANT VALLEY RD / NUCKOLS CROSSING RD	TURNED_ON	LOC16-003180	ROADWAY	ONLINE	10/28/2021 08:30:00 AM +0000	https
1	379	BARTON SPRINGS RD / KINNEY AVE	TURNED_ON	LOC16-000640	ROADWAY	ONLINE	10/29/2021 08:35:00 AM +0000	https
2	404	SPRINGDALE RD / OAK SPRINGS DR	TURNED_ON	LOC16-000800	ROADWAY	ONLINE	10/28/2021 08:35:00 AM +0000	https
3	447	BRAKER LN / STONELAKE BLVD	TURNED_ON	LOC16-003740	ROADWAY	ONLINE	10/23/2021 08:35:00 AM +0000	https
4	552	EXPOSITION BLVD / WESTOVER RD	TURNED_ON	LOC16-003710	ROADWAY	ONLINE	10/20/2021 08:35:00 AM +0000	https

7. Split atd_loc into two new columns 'Loc' and 'code' within df3

In [14]:
```
### Your code goes here ###
df3[['Loc', 'code']] = df3["atd_loc_id"].apply(lambda x: pd.Series(str(x).spli
del df3['atd_loc_id']
```

In [15]:
```
df3.head()
```

Out[15]:

	cam_id	loc_name	cam_stat	loc_type	comm_stat	comm_stat_date	
0	370	PLEASANT VALLEY RD / NUCKOLS CROSSING RD	TURNED_ON	ROADWAY	ONLINE	10/28/2021 08:30:00 AM +0000	https://cctv.aus
1	379	BARTON SPRINGS RD / KINNEY AVE	TURNED_ON	ROADWAY	ONLINE	10/29/2021 08:35:00 AM +0000	https://cctv.aus
2	404	SPRINGDALE RD / OAK SPRINGS DR	TURNED_ON	ROADWAY	ONLINE	10/28/2021 08:35:00 AM +0000	https://cctv.aus
3	447	BRAKER LN / STONELAKE BLVD	TURNED_ON	ROADWAY	ONLINE	10/23/2021 08:35:00 AM +0000	https://cctv.aus
4	552	EXPOSITION BLVD /	TURNED_ON	ROADWAY	ONLINE	10/20/2021	

8. What are the unique values in loc_type?

In [16]:
```
### Your code goes here ###
uniques = df3['loc_type'].unique()
print(uniques)
```

```
['ROADWAY' 'BUILDING']
```

9. Replace 'ROADWAY' to '0', 'BUILDING' to '1' in the loc_type column within df3

In [17]:
```
### Your code goes here ###
df3.loc[df3['loc_type']=='ROADWAY','loc_type'] = '0'
df3.loc[df3['loc_type']=='BUILDING','loc_type'] = '1'
```

In [18]:
```
df3.head()
```

Out[18]:

	cam_id	loc_name	cam_stat	loc_type	comm_stat	comm_stat_date	
0	370	PLEASANT VALLEY RD / NUCKOLS CROSSING RD	TURNED_ON	0	ONLINE	10/28/2021 08:30:00 AM +0000	https://cctv.austinmc
1	379	BARTON SPRINGS RD / KINNEY AVE	TURNED_ON	0	ONLINE	10/29/2021 08:35:00 AM +0000	https://cctv.austinmc
2	404	SPRINGDALE RD / OAK SPRINGS DR	TURNED_ON	0	ONLINE	10/28/2021 08:35:00 AM +0000	https://cctv.austinmc
3	447	BRAKER LN / STONELAKE BLVD	TURNED_ON	0	ONLINE	10/23/2021 08:35:00 AM +0000	https://cctv.austinmc
4	552	EXPOSITION BLVD / WESTOVER RD	TURNED_ON	0	ONLINE	10/20/2021 08:35:00 AM +0000	https://cctv.austinmc

In [19]:
```
df3.loc_type.unique()
```

Out[19]:
```
array(['0', '1'], dtype=object)
```

10. Split on on '/' the loc_name column into two new variables 'corner1', 'corner2'

In [20]:
```
### Your code goes here ###
df3[['corner1', 'corner2']] = df3["loc_name"].apply(lambda x: pd.Series(str(x)
del df3['loc_name']
```

In [21]:
```
df3.head()
```

Out[21]:

	cam_id	cam_stat	loc_type	comm_stat	comm_stat_date	screen_
0	370	TURNED_ON	0	ONLINE	10/28/2021 08:30:00 AM +0000	https://cctv.austinmobility.io/image/37
1	379	TURNED_ON	0	ONLINE	10/29/2021 08:35:00 AM +0000	https://cctv.austinmobility.io/image/37
2	404	TURNED_ON	0	ONLINE	10/28/2021 08:35:00 AM +0000	https://cctv.austinmobility.io/image/40
3	447	TURNED_ON	0	ONLINE	10/23/2021 08:35:00 AM +0000	https://cctv.austinmobility.io/image/44
4	552	TURNED_ON	0	ONLINE	10/20/2021 08:35:00 AM +0000	https://cctv.austinmobility.io/image/55

Part2: Exploratory Data Analysis (EDA)

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

Follow the lecture notes for ideas of how to perform EDA on your dataset. For help, here are the steps we talked about:

Steps in EDA:

1. Provide descriptions of your sample and features
2. Check for missing data
3. Identify the shape of your data
4. Identify significant correlations
5. Spot/deal with outliers in the dataset

These steps are a guidline. Try different things and share your insights about the dataset.

Don't forget to add "markdown" cells to include your findings or to explain what you are doing

In [22]:
```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [23]:
```
# Reading the CSV file
df_fish = pd.read_csv("Fish.csv")

# Printing top 5 rows
df_fish.head()
```

Out[23]:

	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 [24]:
```
#1. Provide descriptions of your sample and features
#Find row and columns
print("Rows/Observations: " + str(df_fish.shape[0]))
print("Columns/Features: " + str(df_fish.shape[1]))
print(df_fish.info())

#Check how many variables are categorical, rest are likely numeric
categorical_fish = 0
for i in df_fish.dtypes:
    if i == object:
        categorical_fish += 1
print("The data frame has " + str(categorical_fish) + " features that are categ

#print quick description
df_fish.describe()
```

```
Rows/Observations: 159
Columns/Features: 7
<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
None
The data frame has 1 features that are categorical
```

Out[24]:

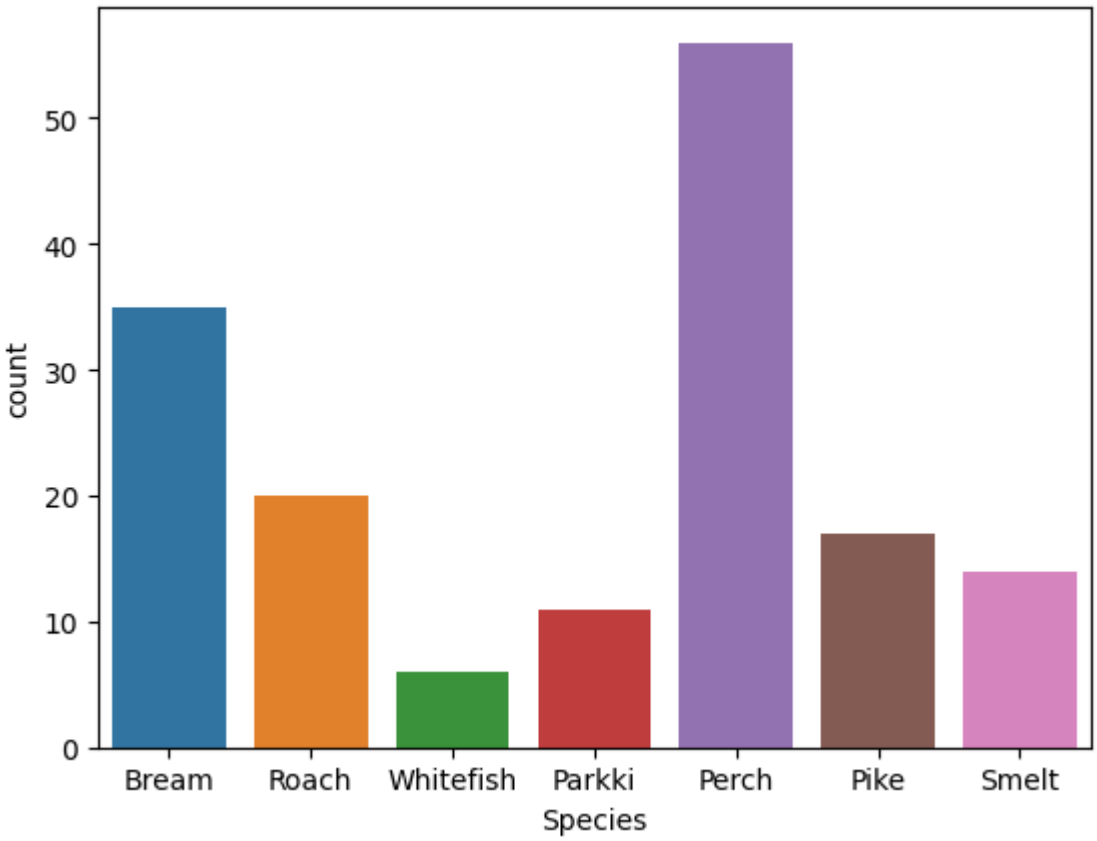
	Weight	Length1	Length2	Length3	Height	Width
count	159.000000	159.000000	159.000000	159.000000	159.000000	159.000000
mean	398.326415	26.247170	28.415723	31.227044	8.970994	4.417486
std	357.978317	9.996441	10.716328	11.610246	4.286208	1.685804
min	0.000000	7.500000	8.400000	8.800000	1.728400	1.047600
25%	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 [25]:
```
#2 Check for missing values
missing_values = False
for i in df_fish.isnull().sum():
    if i != 0:
        print("There are missing values")
        missing_values = True
if(not missing_values):
    print("There are no missing values")

print(df_fish.isnull().sum())
print("\n\n")
print(df_fish.value_counts("Species"))
print("We can see our data is not balanced, since there are different amounts o
```

```
There are no missing values
Species    0
Weight     0
Length1    0
Length2    0
Length3    0
Height     0
Width      0
dtype: int64



Species
Perch        56
Bream        35
Roach        20
Pike         17
Smelt        14
Parkki       11
Whitefish     6
dtype: int64
We can see our data is not balanced, since there are different amounts of eac
h species of fish
```

In [26]:
```
#3 Find shape of data
sns.countplot(x='Species', data=df_fish)
```

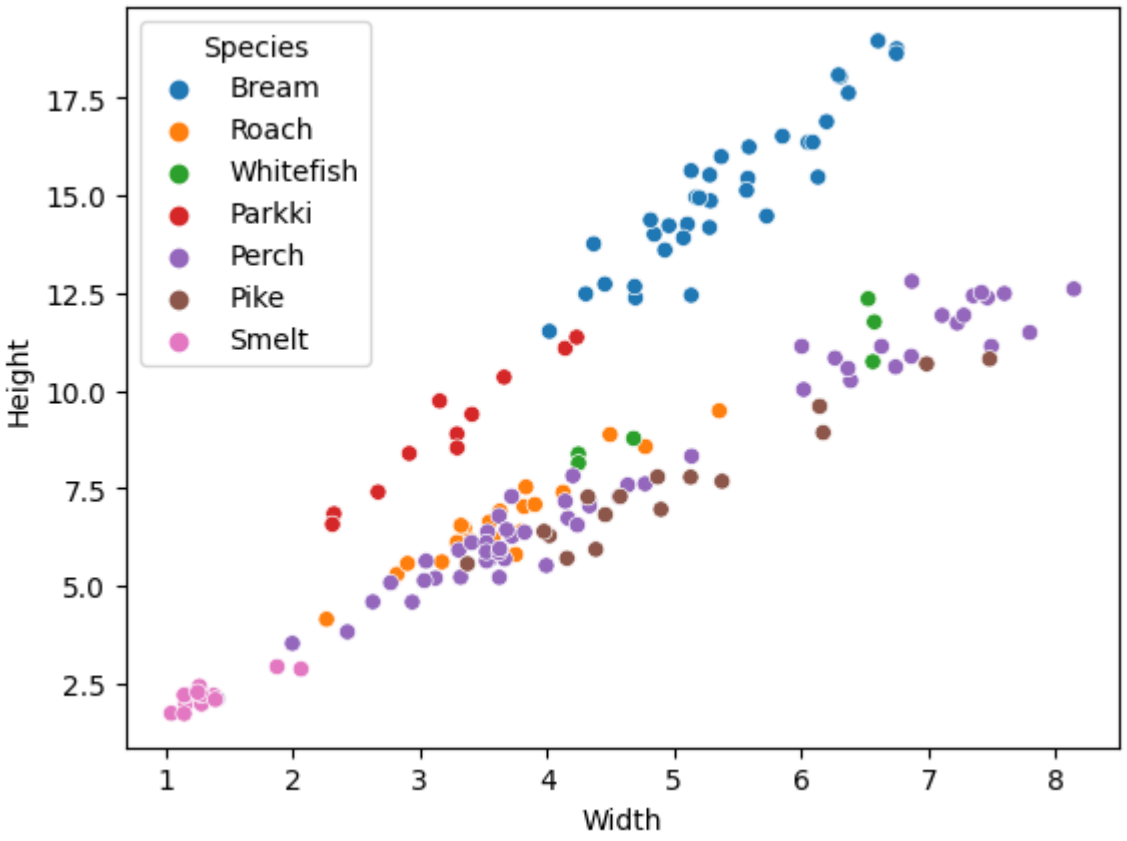
Out[26]: <Axes: xlabel='Species', ylabel='count'>



In [27]:
```
#From the countplot above, we can see our data consists mainly of the species F
#This will affect are conclusion in the future
```

In [28]:
```
sns.scatterplot(data=df_fish, x="Width", y="Height", hue='Species')
```

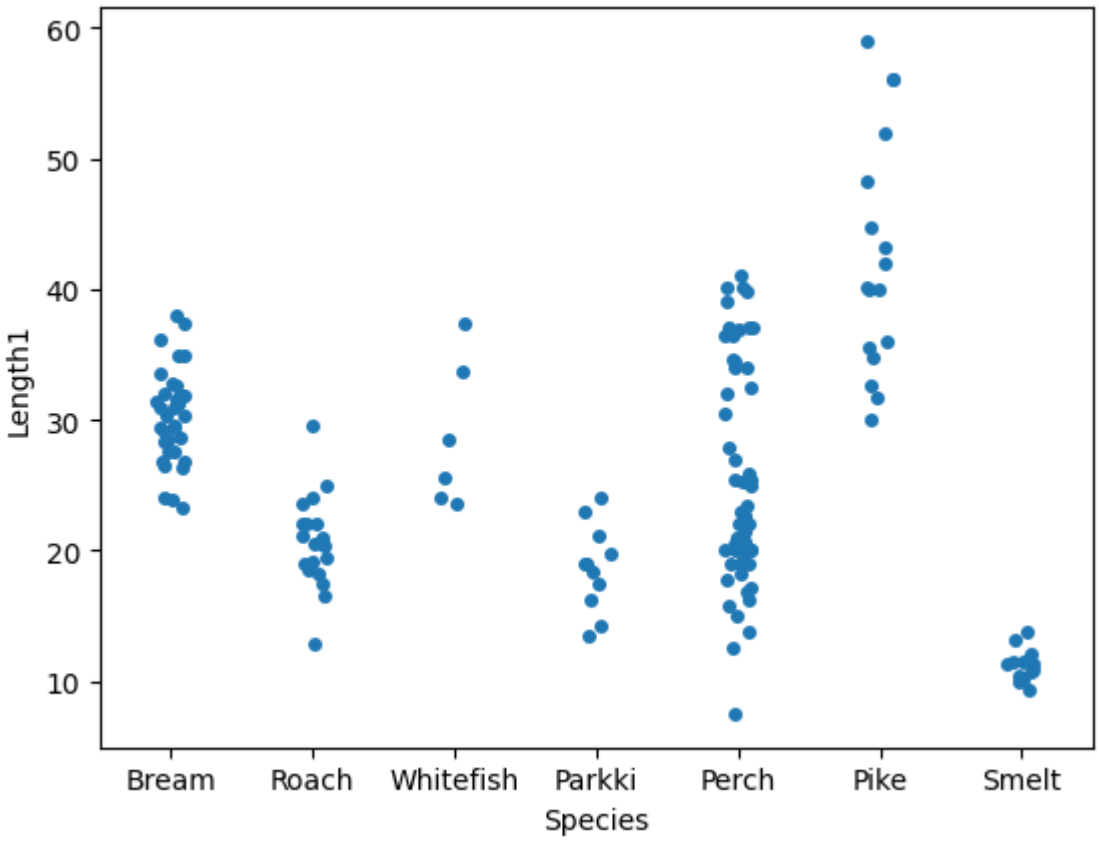
Out[28]:
```
<Axes: xlabel='Width', ylabel='Height'>
```



In [29]:
```
#Here, we can see the type of species has a relationship with the fishes size,
#since many points of the same species are grouped together

#Species Bream Seems on average to have the highest height and width, while sme
#smallest in both.
#The others seem to lie in the middle, however their width seems to vary.
```

In [30]:
```
sns.stripplot(y ='Length1', x = 'Species', data = df_fish)
```

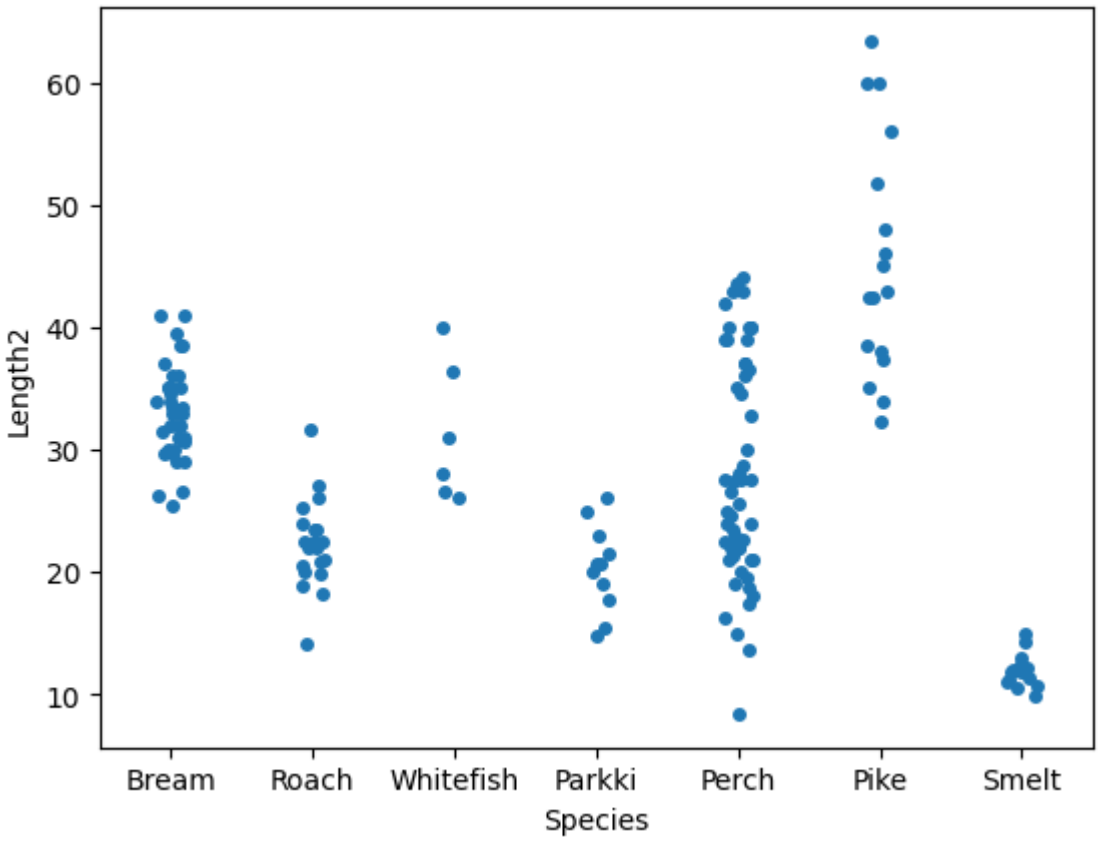
Out[30]:
```
<Axes: xlabel='Species', ylabel='Length1'>
```



In [31]:
```
#Here, we see Smelt has the smallest length1, while Pike has some with the high
#Perch seems to be in betwen 10-40, which is a large variance, however there ar
#compared to the others.
```

In [32]:
```
sns.stripplot(y ='Length2', x = 'Species', data = df_fish)
```

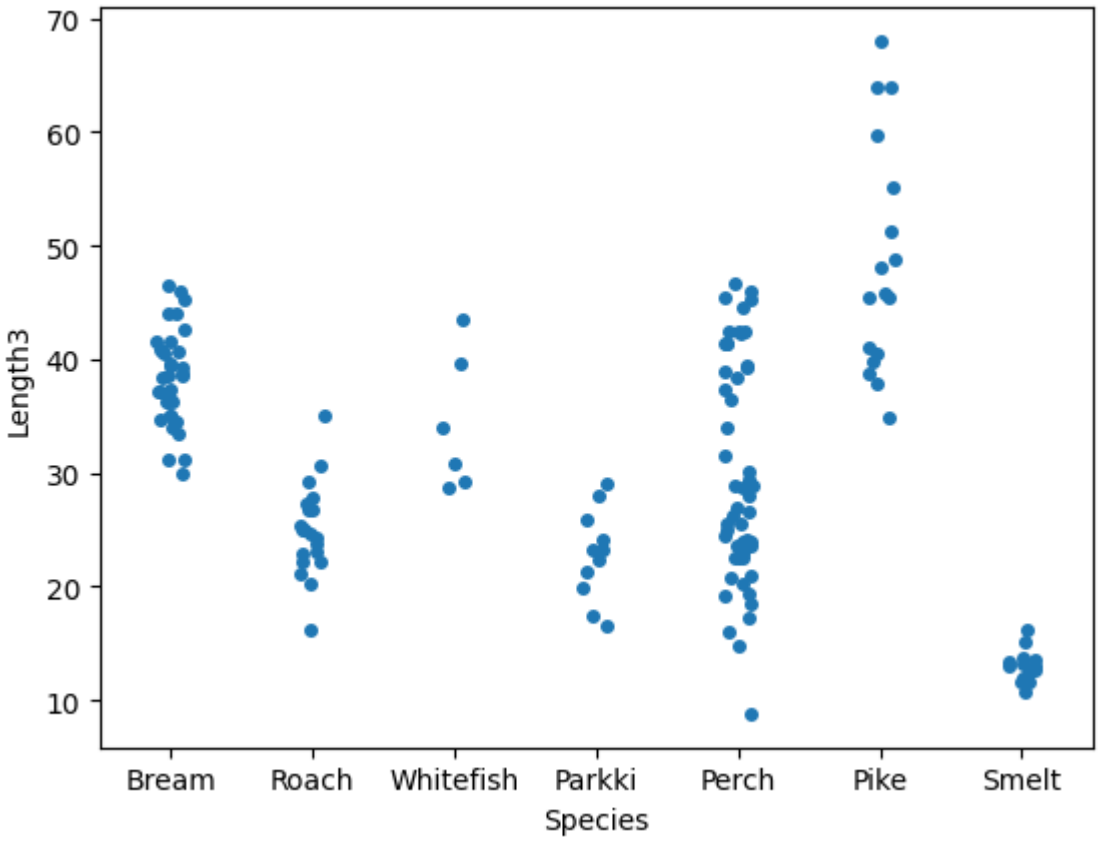
Out[32]: <Axes: xlabel='Species', ylabel='Length2'>



In [33]:
```
#Plot of Length2 seems similar to plot of Length1
```

In [34]:
```
sns.stripplot(y ='Length3', x = 'Species', data = df_fish)
```

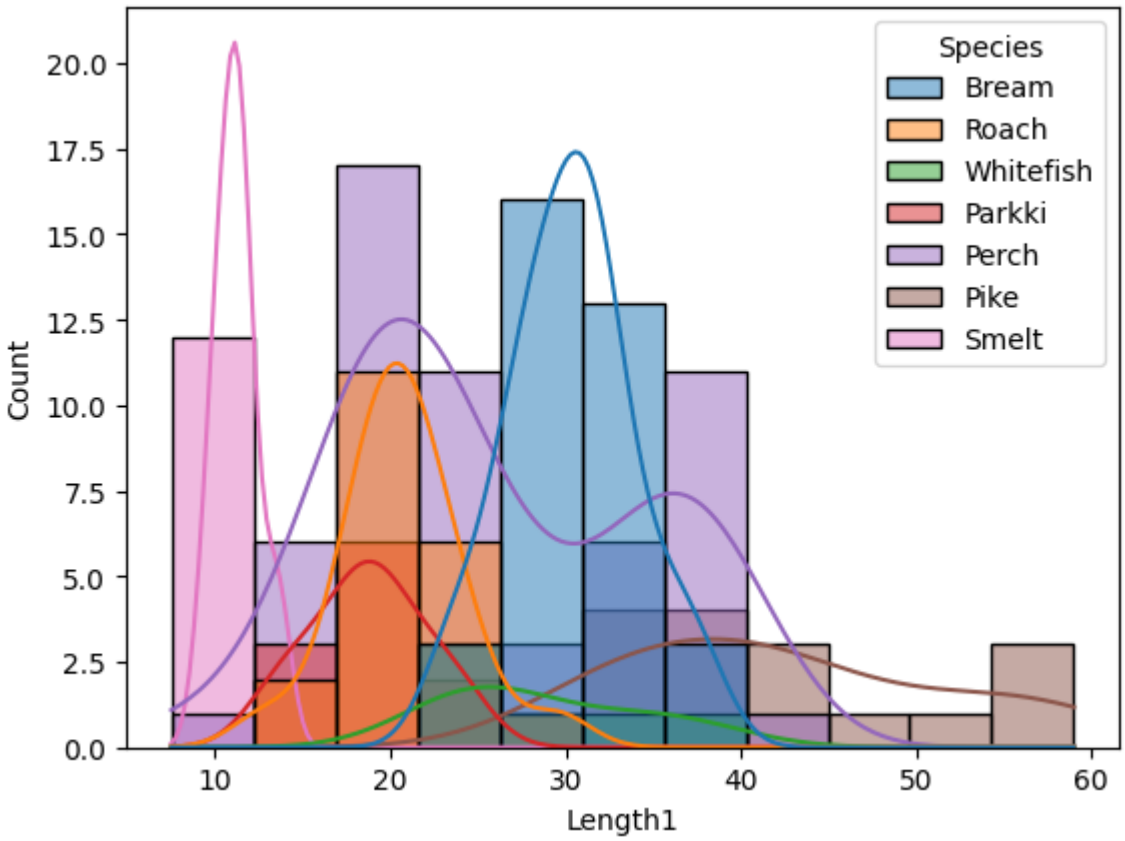
Out[34]:
```
<Axes: xlabel='Species', ylabel='Length3'>
```



In [35]:
```
#Same for length 3, except now Pikes reach up to 70 instead of 60 units.
```

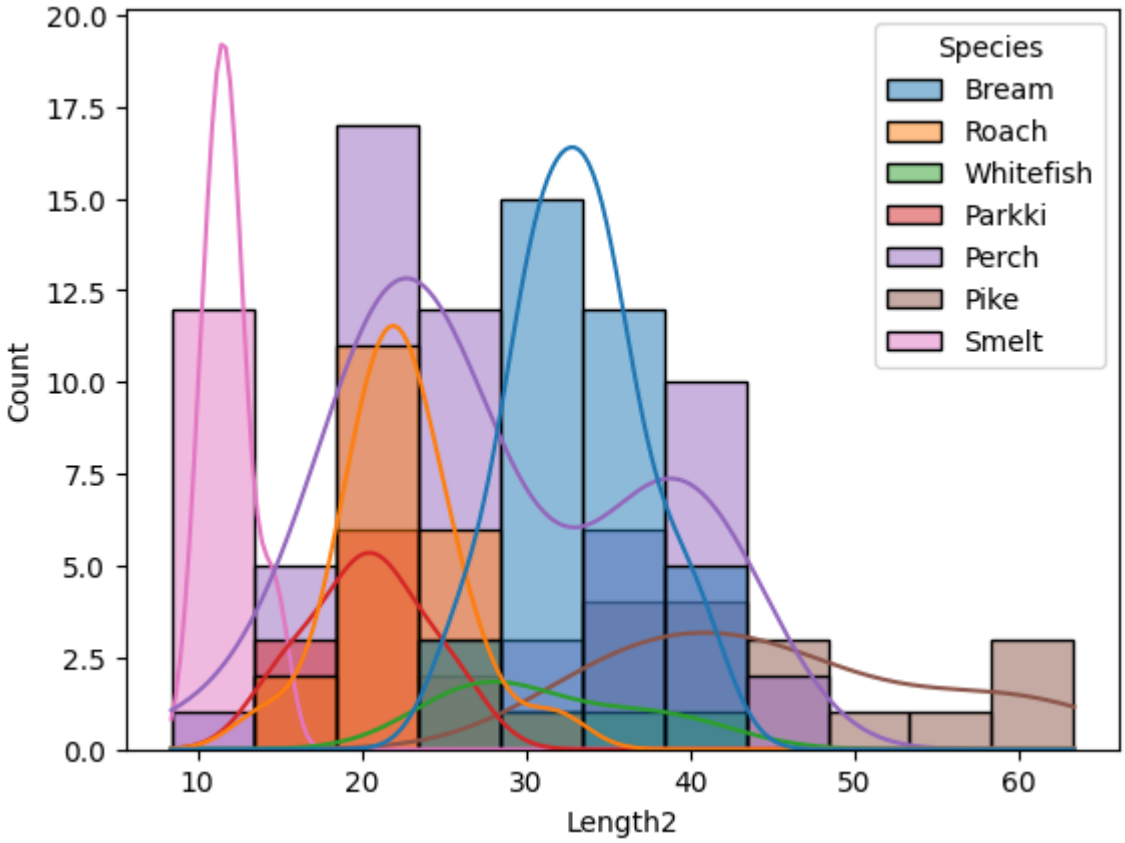
In [36]:
```
sns.histplot(x='Length1', data=df_fish, hue='Species', kde=True)
```

Out[36]:
```
<Axes: xlabel='Length1', ylabel='Count'>
```



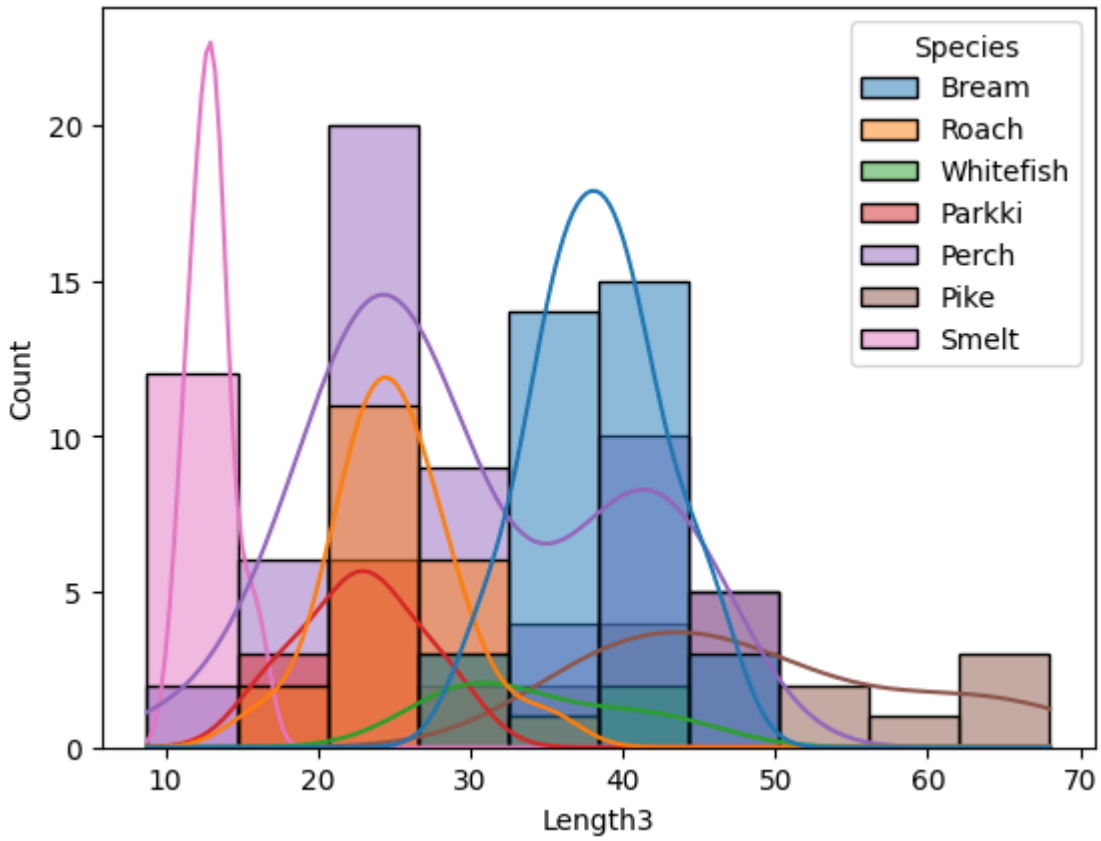
In [37]:
```
sns.histplot(x='Length2', data=df_fish, hue='Species', kde=True)
```

Out[37]:
```
<Axes: xlabel='Length2', ylabel='Count'>
```



In [38]:
```
sns.histplot(x='Length3', data=df_fish, hue='Species', kde=True)
```

Out[38]:
```
<Axes: xlabel='Length3', ylabel='Count'>
```



In [39]:
```
#For the 3 lengthsm, we can see Smelt, Bream, and Pike tend to skew away from t
#While Roach,Parkki, and Perch, and Whitefish tend to overlap
```

In [40]:
```
#4 Identify Significant Correlation
df_fish.corr(method="pearson")
```

```
C:\Users\gamer\AppData\Local\Temp\ipykernel_22744\1020334001.py:2: FutureWarn
ing: The default value of numeric_only in DataFrame.corr is deprecated. In a
future version, it will default to False. Select only valid columns or specif
y the value of numeric_only to silence this warning.
  df_fish.corr(method="pearson")
```

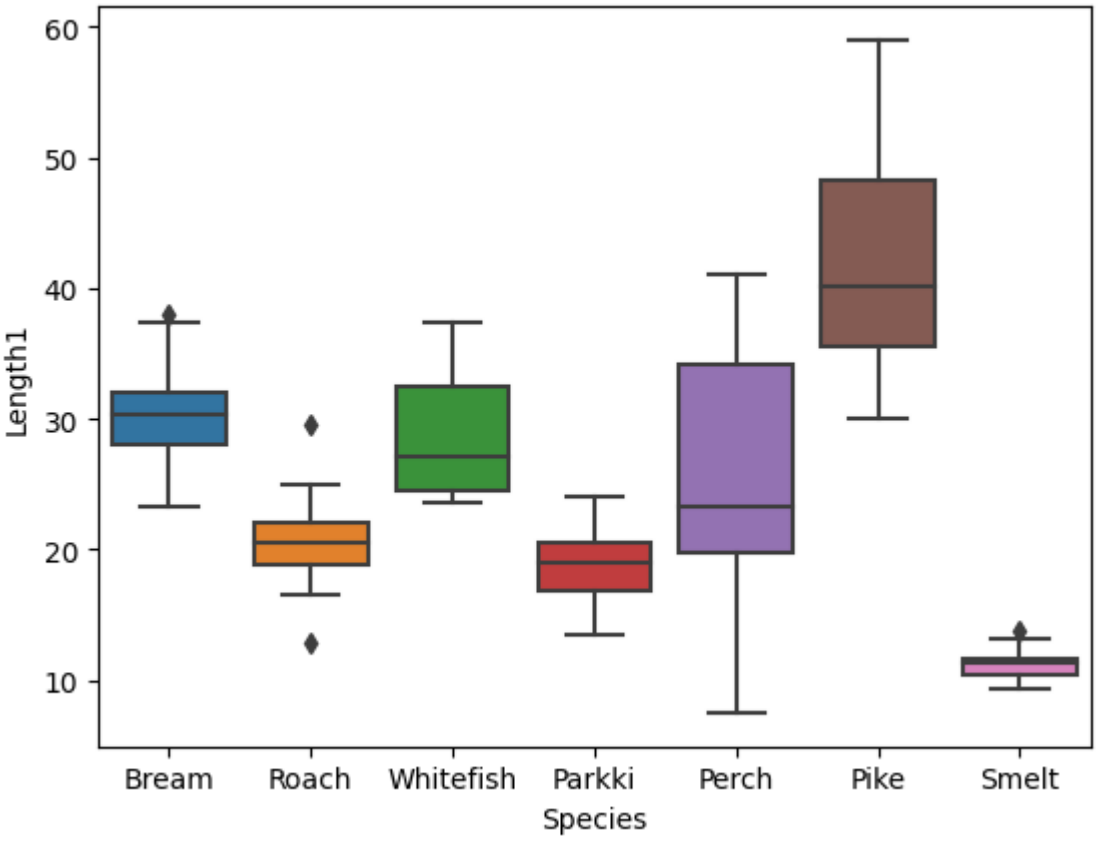
Out[40]:

	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 [41]:
```
#There appears to be significant correlations between the lengths, and some co
```

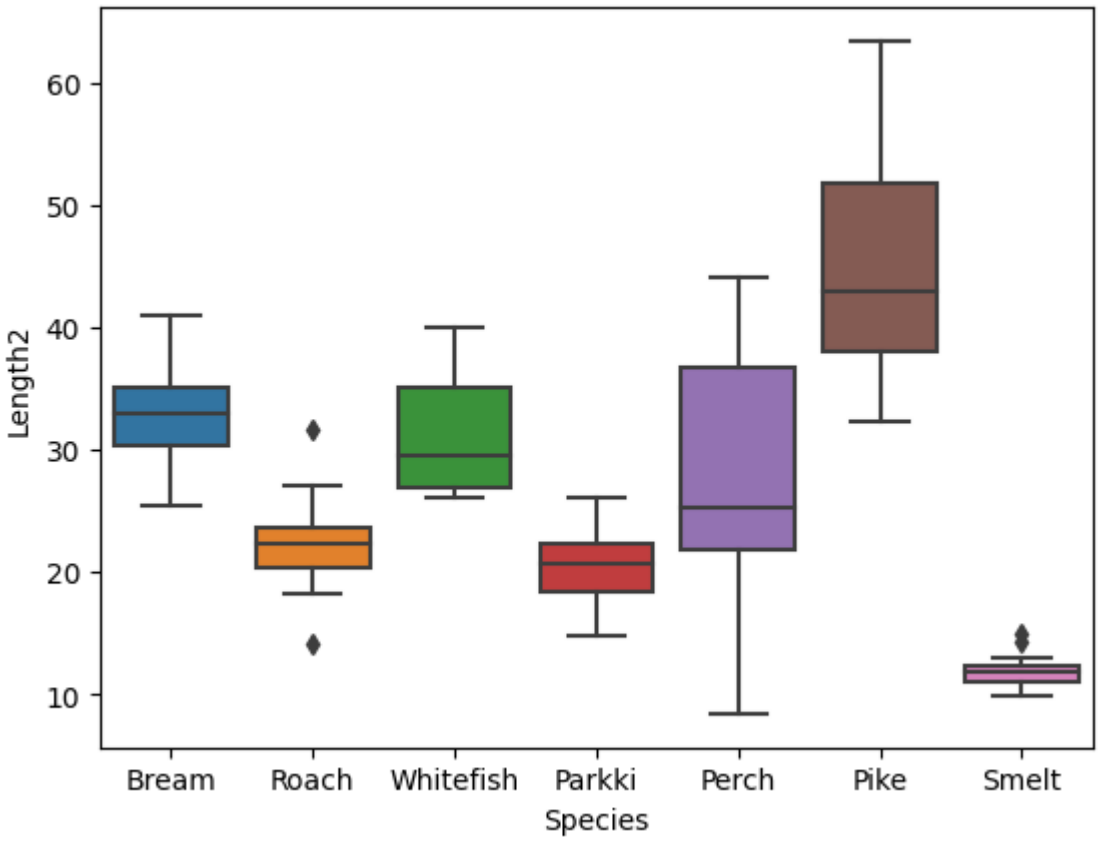
In [42]:
```
#5 Detect and Handle Outliers
sns.boxplot(x="Species", y="Length1", data=df_fish)
```

Out[42]:
```
<Axes: xlabel='Species', ylabel='Length1'>
```



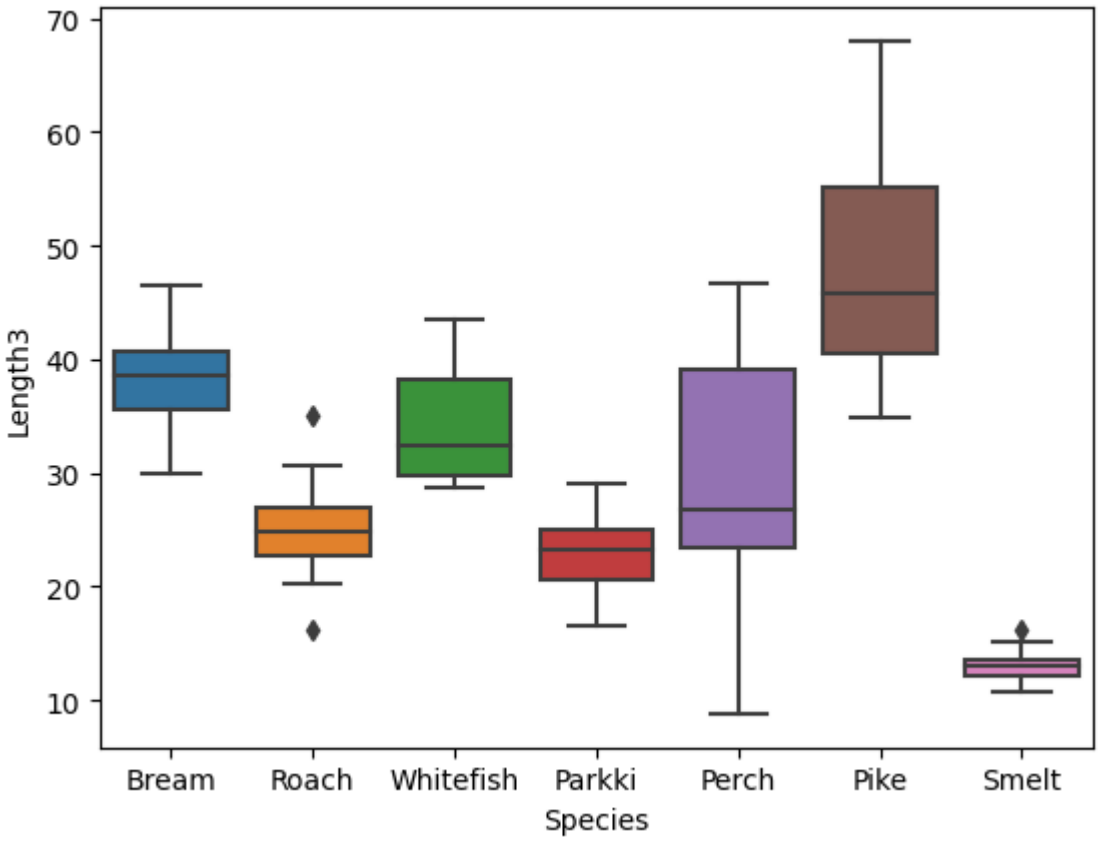
In [43]:
```
sns.boxplot(x="Species", y="Length2", data=df_fish)
```

Out[43]:
```
<Axes: xlabel='Species', ylabel='Length2'>
```



In [44]:
```
sns.boxplot(x="Species", y="Length3", data=df_fish)
```

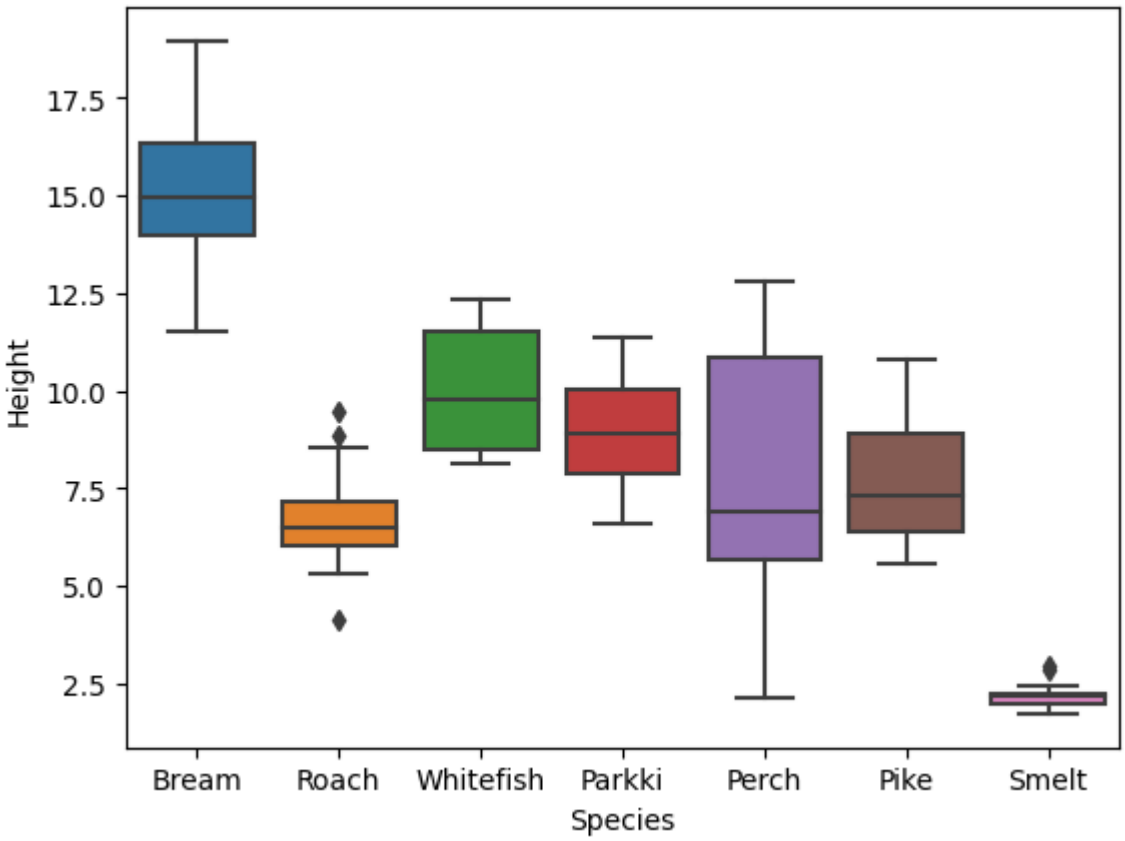
Out[44]:
```
<Axes: xlabel='Species', ylabel='Length3'>
```



In [45]:
```
#Here we see when it comes to the lengths of the fish, Roach and Smelt both hav
#Bream also has outliers within its species when it comes to Length1

#It is possible the outliers were fed more than other fish, or grew up in diffe
#There is also the possibility that there were errors made when the data was to
```

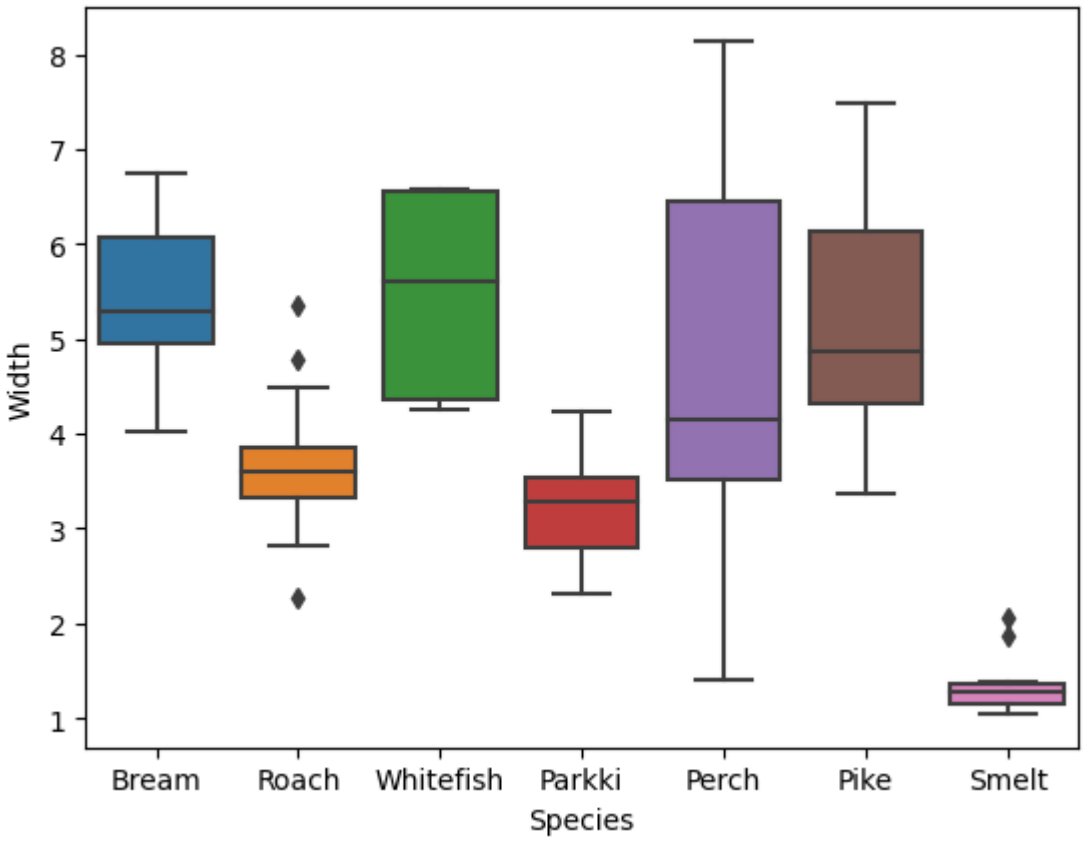
In [46]:
```
sns.boxplot(x="Species", y="Height", data=df_fish)
```

Out[46]:
```
<Axes: xlabel='Species', ylabel='Height'>
```



In [47]:
```
sns.boxplot(x="Species", y="Width", data=df_fish)
```

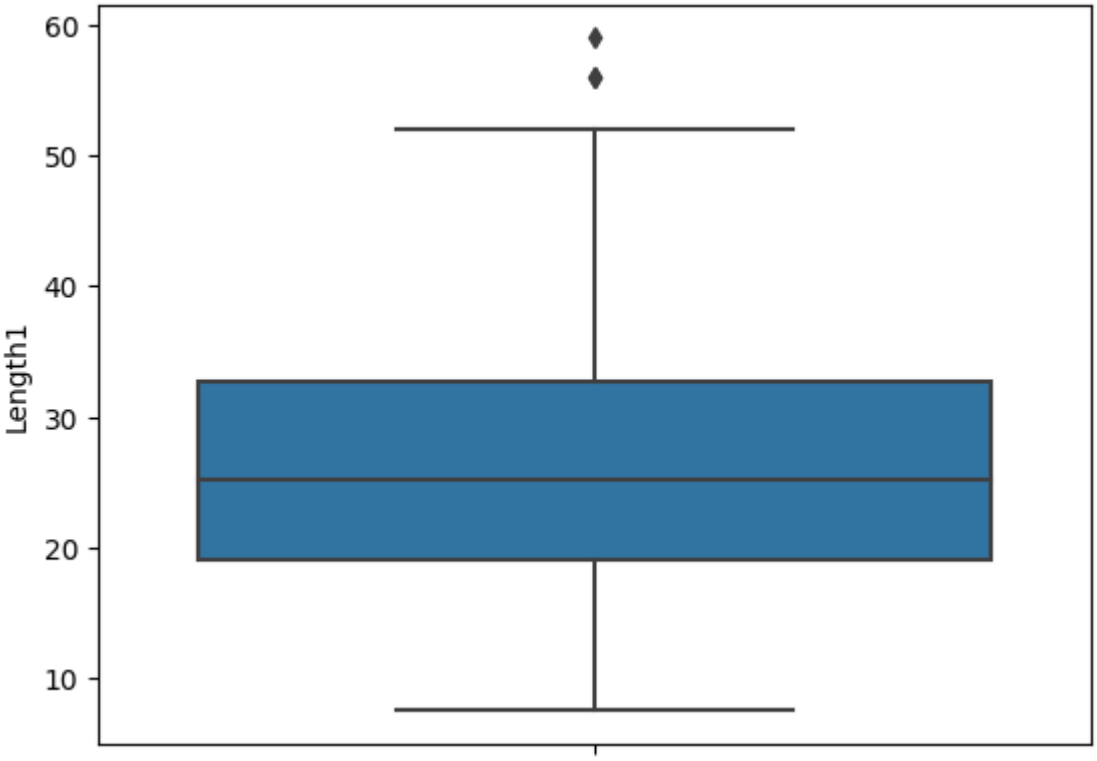
Out[47]:
```
<Axes: xlabel='Species', ylabel='Width'>
```



In [48]:
```
#When looking at height and width of fish with their species, we see
#its similar to the lengths, where Roach and Smelt both have outliers.
#However Bream appears to have none.
```

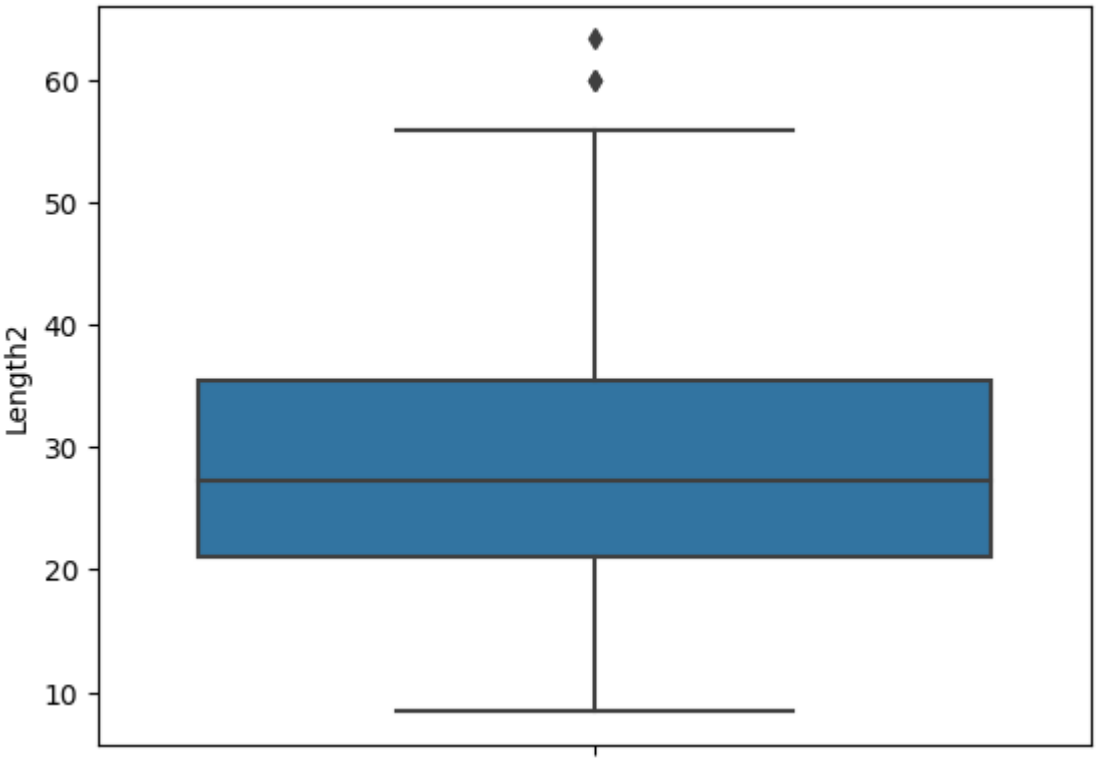
In [49]:
```
sns.boxplot(y="Length1", data=df_fish)
```

Out[49]:
```
<Axes: ylabel='Length1'>
```



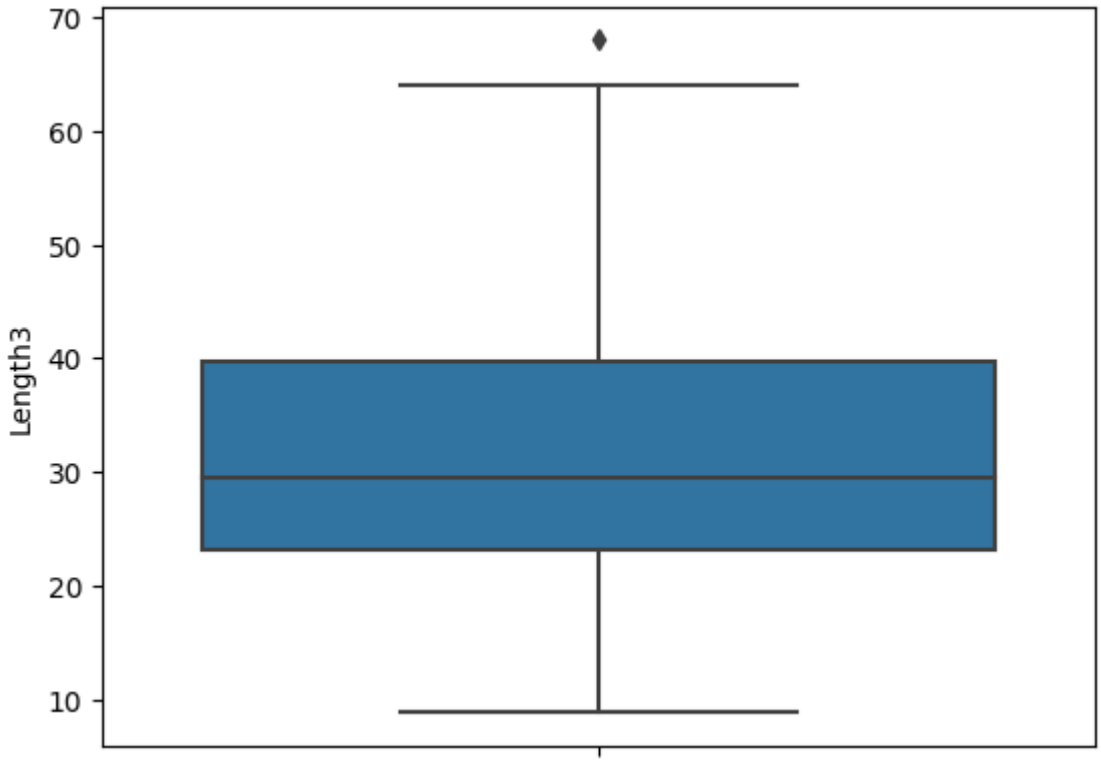
In [50]:
```
sns.boxplot(y="Length2", data=df_fish)
```

Out[50]:
```
<Axes: ylabel='Length2'>
```



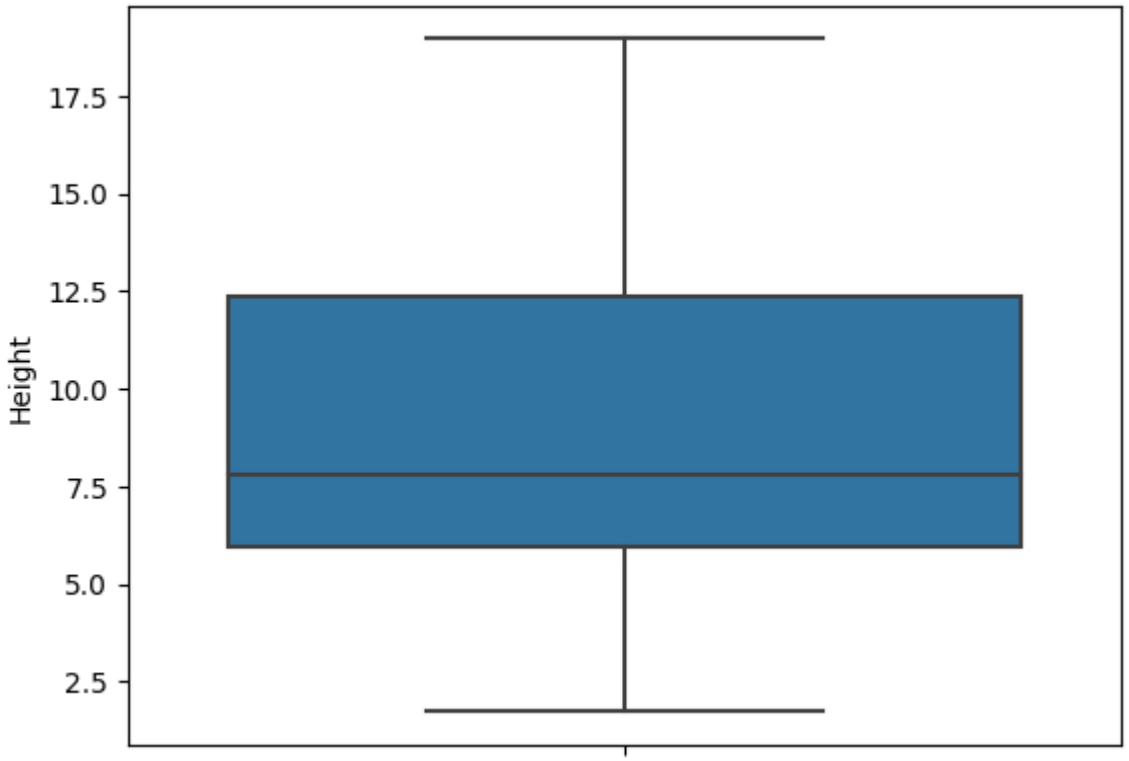
In [51]:
```
sns.boxplot(y="Length3", data=df_fish)
```

Out[51]:
```
<Axes: ylabel='Length3'>
```



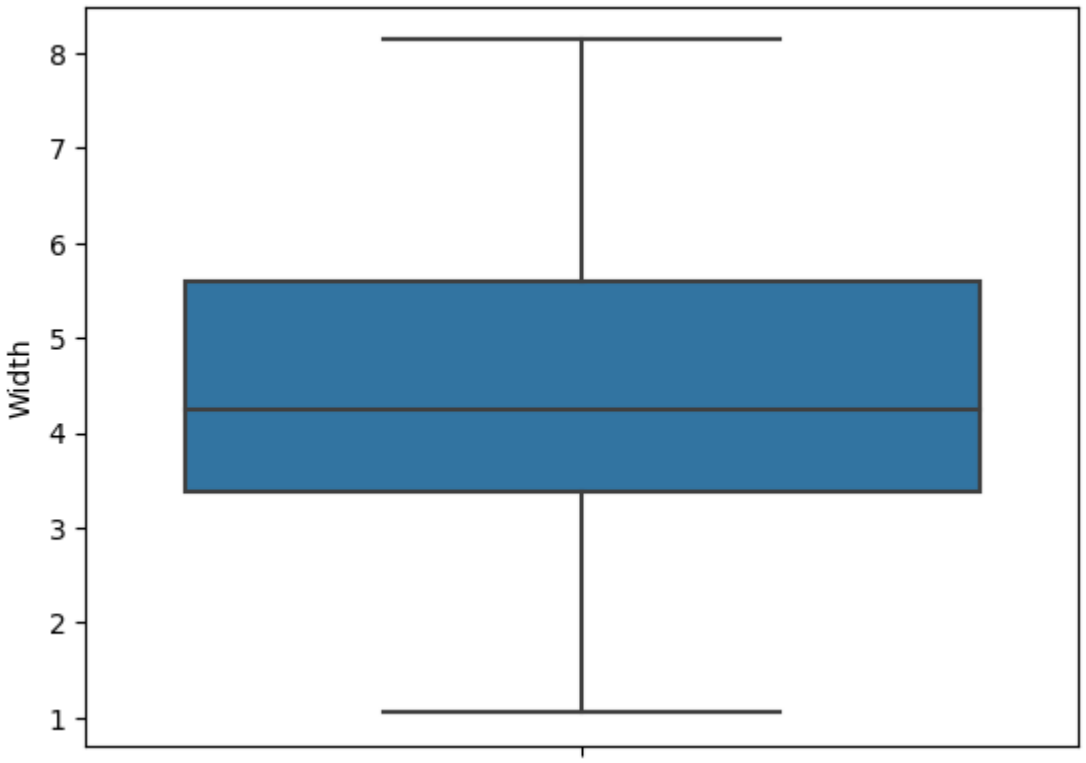
In [52]:
```
sns.boxplot(y="Height", data=df_fish)
```

Out[52]:
```
<Axes: ylabel='Height'>
```



In [53]:
```
sns.boxplot(y="Width", data=df_fish)
```

Out[53]: <Axes: ylabel='Width'>



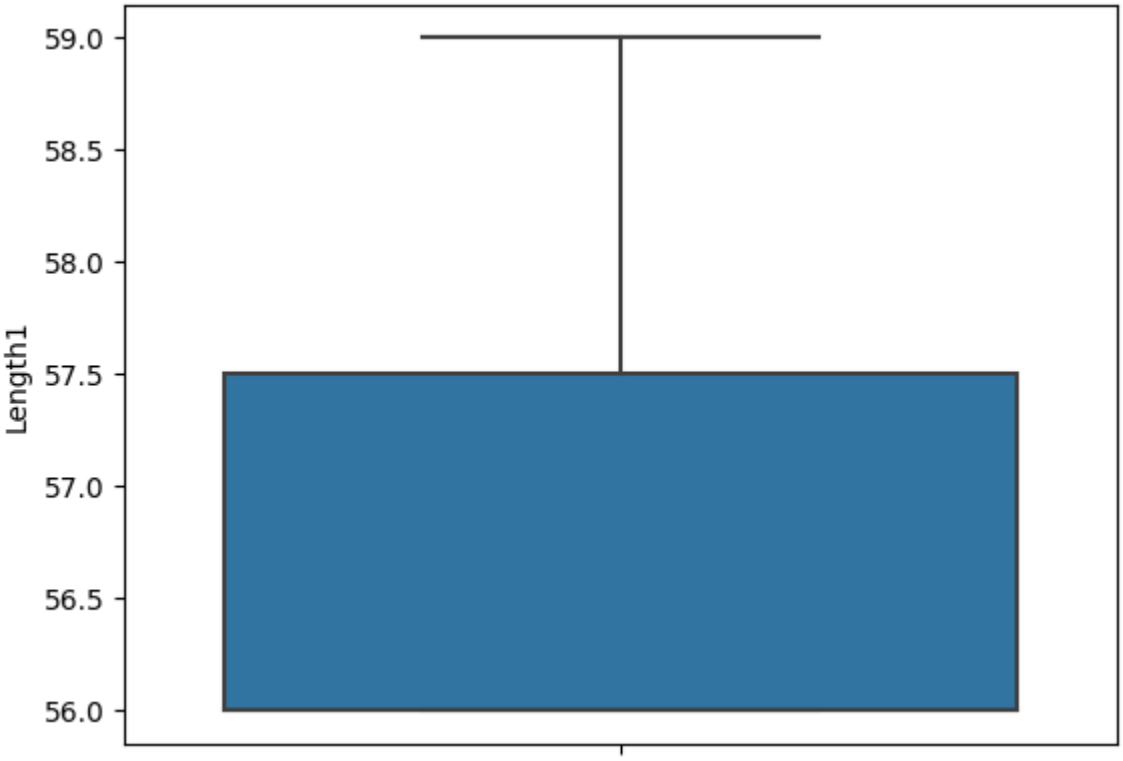
In [54]:
```
#When comparing the lengths between all of the fish, we can see some outliers
#When just comparing height and width of all fish, we can see no outliers
```

In [55]:
```
#Remove outliers between all species of fish

#Find quartiles of data
Q1 = df_fish['Length1'].quantile(0.25)
Q3 = df_fish['Length1'].quantile(0.75)
print("Q1 = " + str(Q1))
print("Q3 = " + str(Q3))

IQR = Q3-Q1
print("IQR = " + str(IQR))

u_fence = Q3 + (1.5 * IQR)
print("Upper fence = " + str(u_fence))
```

```
Q1 = 19.05
Q3 = 32.7
IQR = 13.650000000000002
Upper fence = 53.175000000000004
```

In [56]:
```
#Find outliers above upper fence
df_fish_filtered = df_fish[df_fish['Length1'] >= u_fence]
sns.boxplot(y="Length1", data=df_fish_filtered)
```

Out[56]: <Axes: ylabel='Length1'>



In [ ]: