HRISHIKESH G KULKARNI - 21BAI1660

hrishikesh.gkulkarni2021@vitstudent.ac.in

ASSIGNMENT - 03

1. Downloaded the dataset from the given link!

import needed libraries

In [ ]:

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

2. Load the Dataset

In [ ]:

```
df = pd.read_csv('penguins_size.csv')
df.head()
```

Out[ ]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_
0	Adelie	Torgersen	39.1	18.7	181.0	
1	Adelie	Torgersen	39.5	17.4	186.0	
2	Adelie	Torgersen	40.3	18.0	195.0	
3	Adelie	Torgersen	NaN	NaN	NaN	
4	Adelie	Torgersen	36.7	19.3	193.0	

In [ ]:

```
df.shape
```

Out[ ]:

```
(344, 7)
```

In [ ]:

```
df.species.value_counts()
```

Out[ ]:

```
Adelie       152
Gentoo       124
Chinstrap     68
Name: species, dtype: int64
```

In [ ]:

```
df.island.value_counts()
```

Out[ ]:

```
Biscoe       168
Dream        124
Torgersen     52
Name: island, dtype: int64
```

In [ ]:
```
df.sex.value_counts()
```

Out[ ]:
```
MALE      168
FEMALE    165
.           1
Name: sex, dtype: int64
```

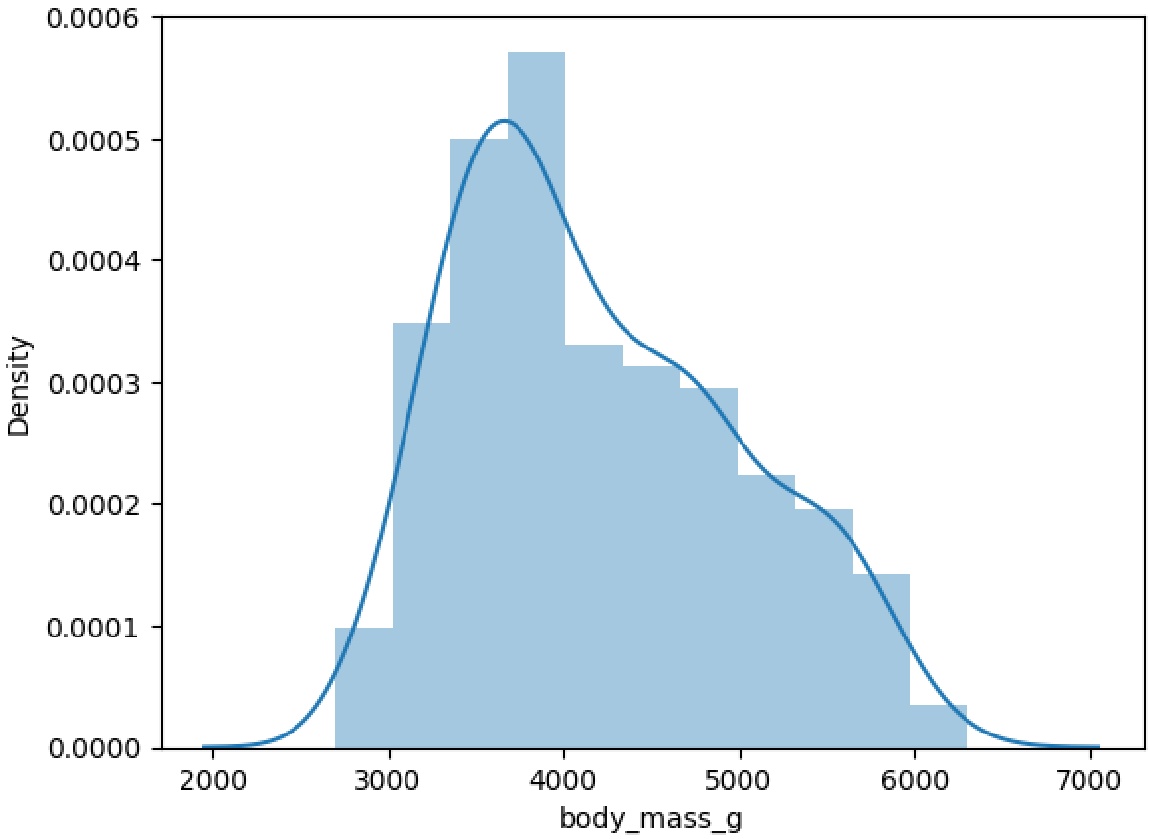
3. Visualizations

Univariate

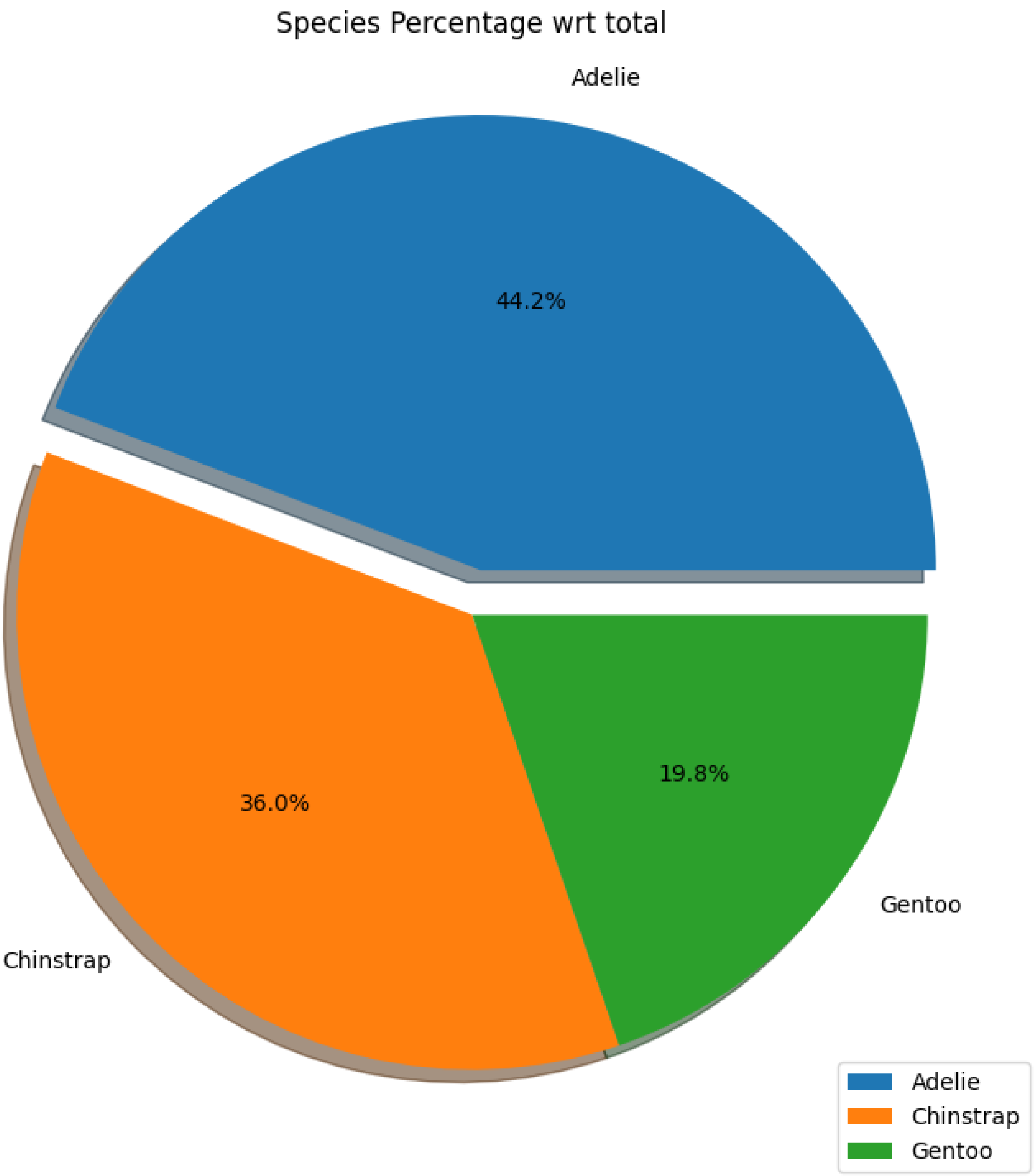
In [ ]:
```
sns.distplot(df["body_mass_g"])
```

```
C:\Users\hrish\AppData\Local\Temp\ipykernel_15604\3012059868.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df["body_mass_g"])
```

Out[ ]:
```
<AxesSubplot: xlabel='body_mass_g', ylabel='Density'>
```



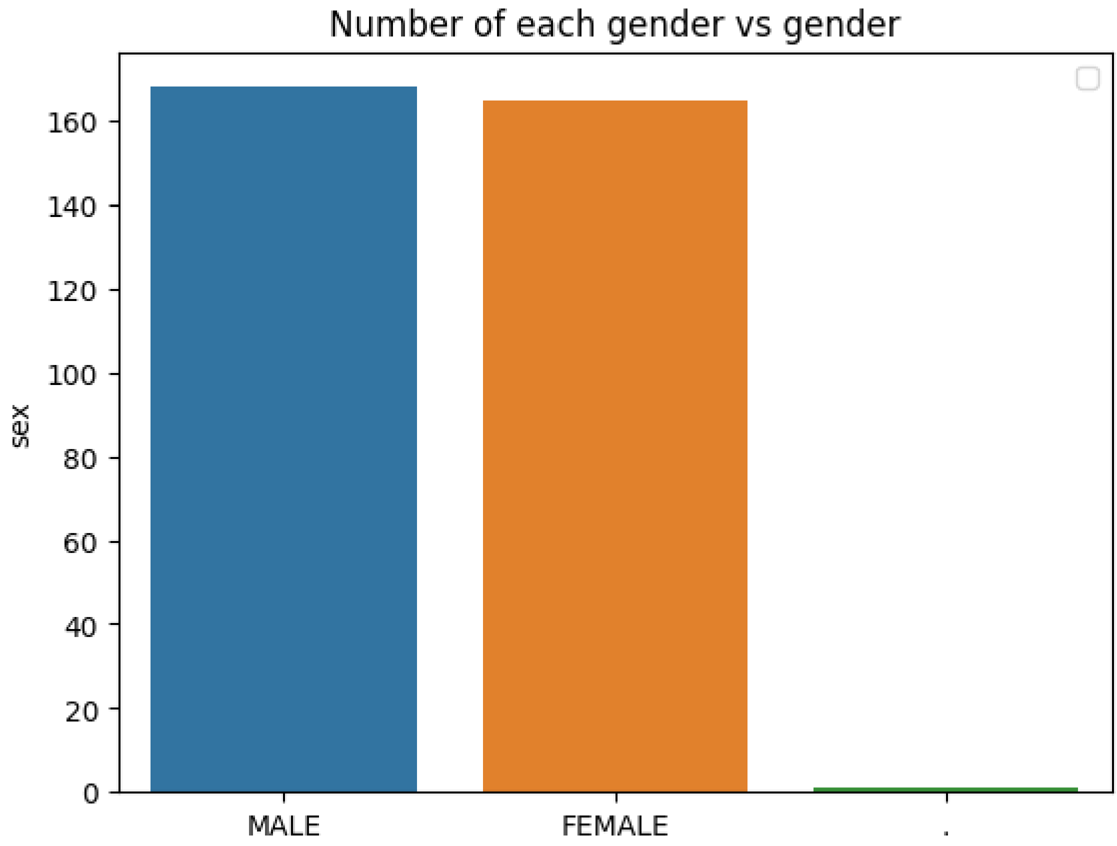
In [ ]:
```
plt.figure(figsize=(9,9))
plt.pie(df["species"].value_counts(),[0.1,0,0], labels = df["species"].unique(),
plt.title('Species Percentage wrt total')
plt.legend()
plt.show()
```



In [ ]:

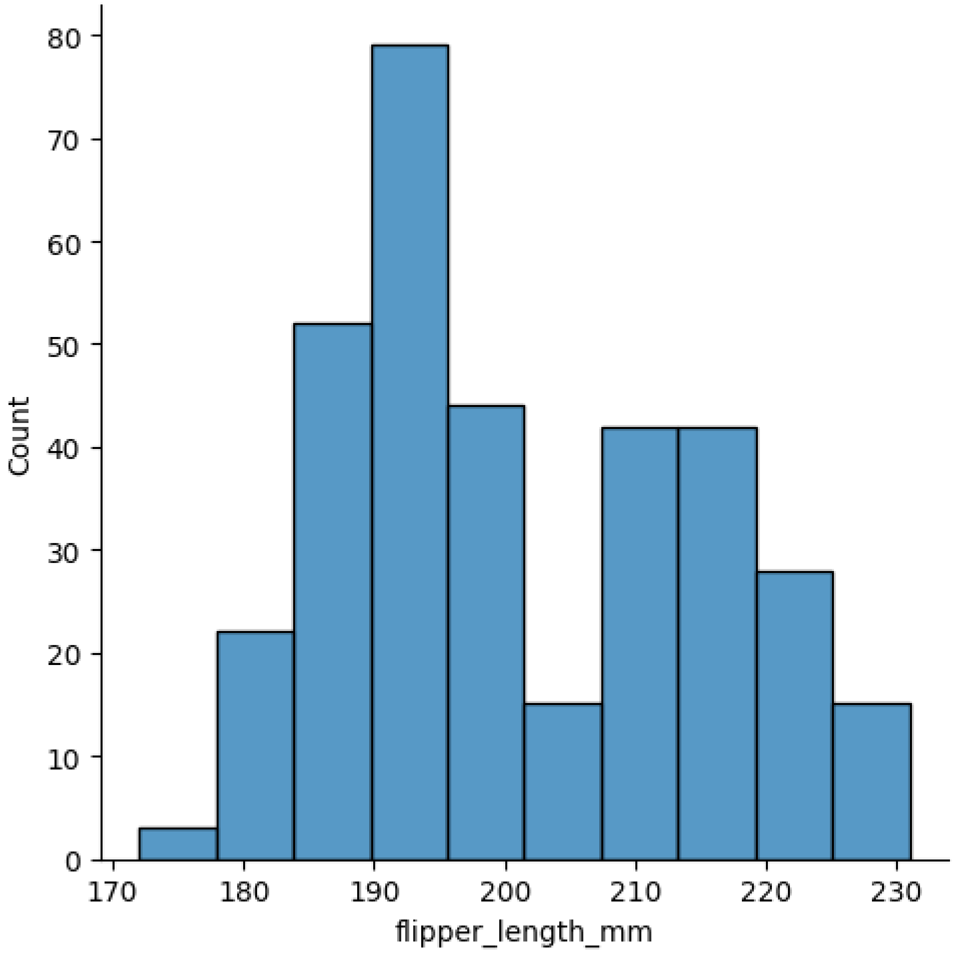
```
sns.barplot(x =df["sex"].value_counts().index,y =df["sex"].value_counts())
plt.title('Number of each gender vs gender')
plt.legend()
plt.show()
```

```
No artists with labels found to put in legend.  Note that artists whose label sta
rt with an underscore are ignored when legend() is called with no argument.
```



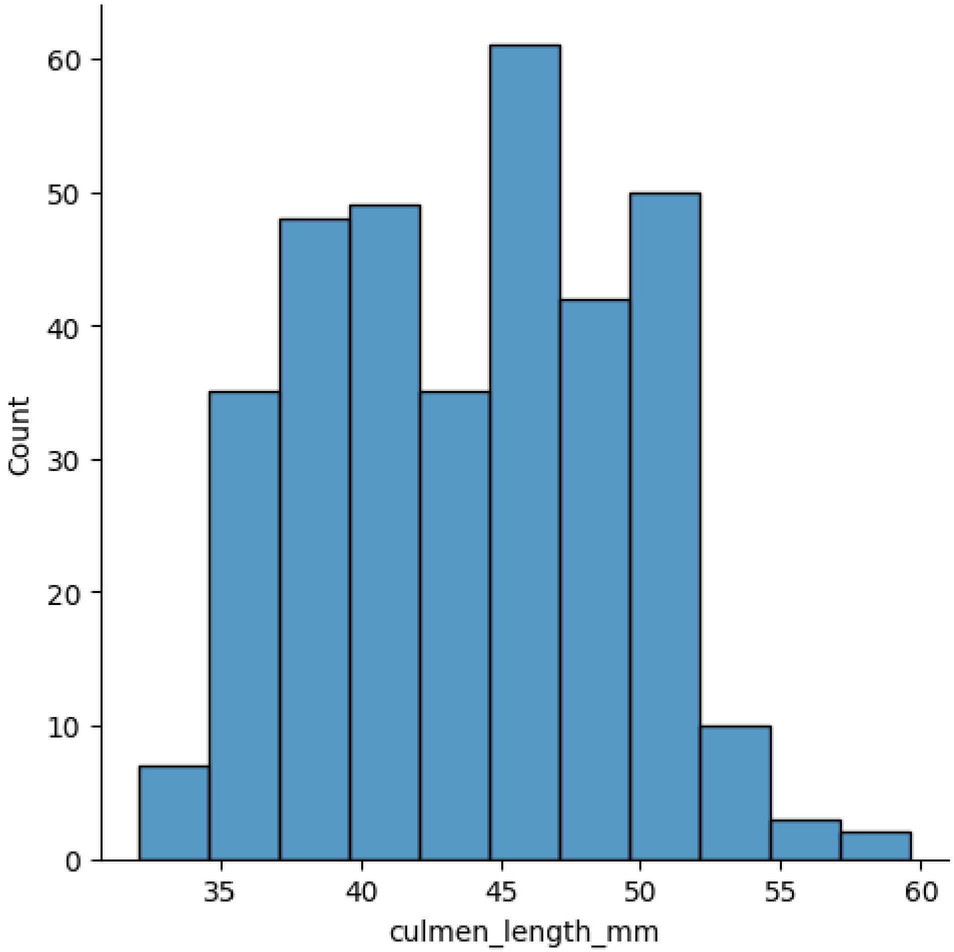
In [ ]:
```
sns.displot(df["flipper_length_mm"])
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1e087923070>
```



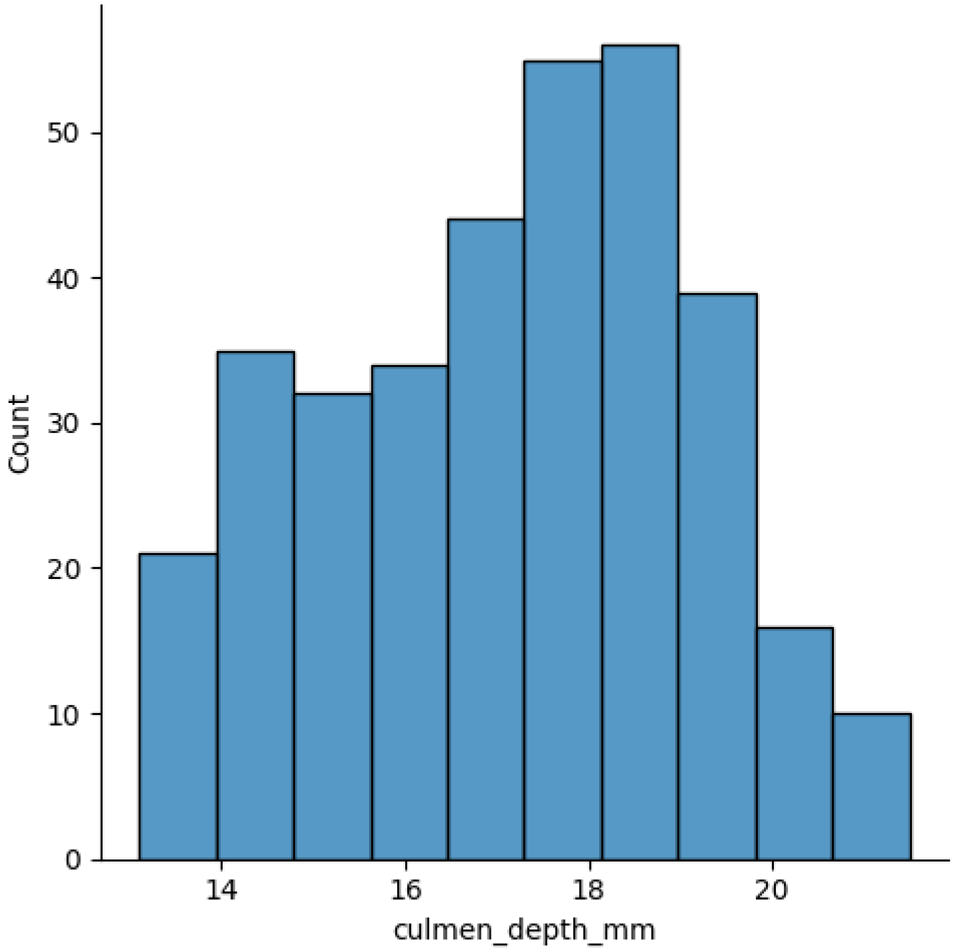
```
In [ ]: sns.displot(df["culmen_length_mm"])
```

```
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x1e088636e00>
```



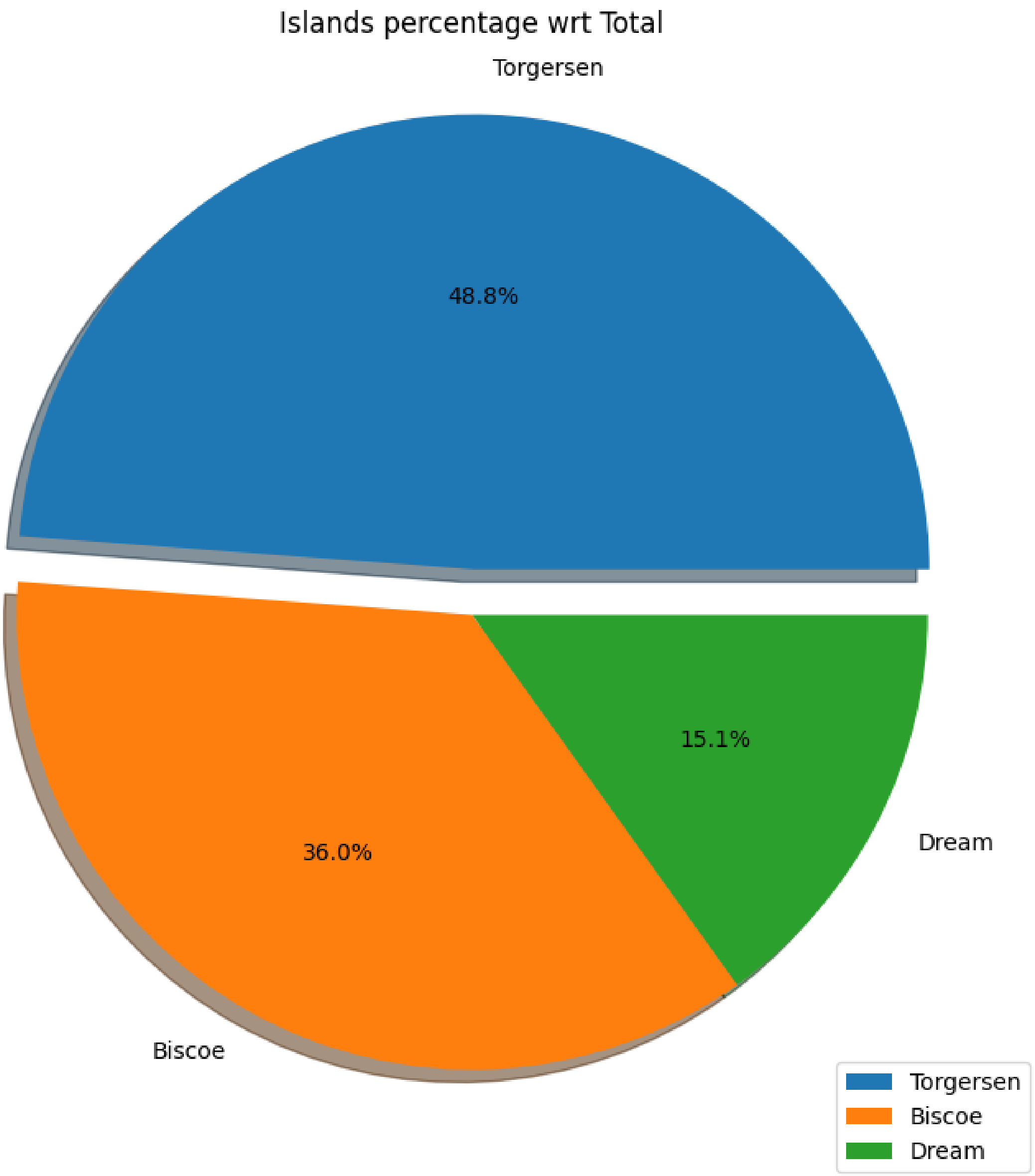
```
In [ ]: sns.displot(df["culmen_depth_mm"])
```

```
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x1e09f8b2590>
```



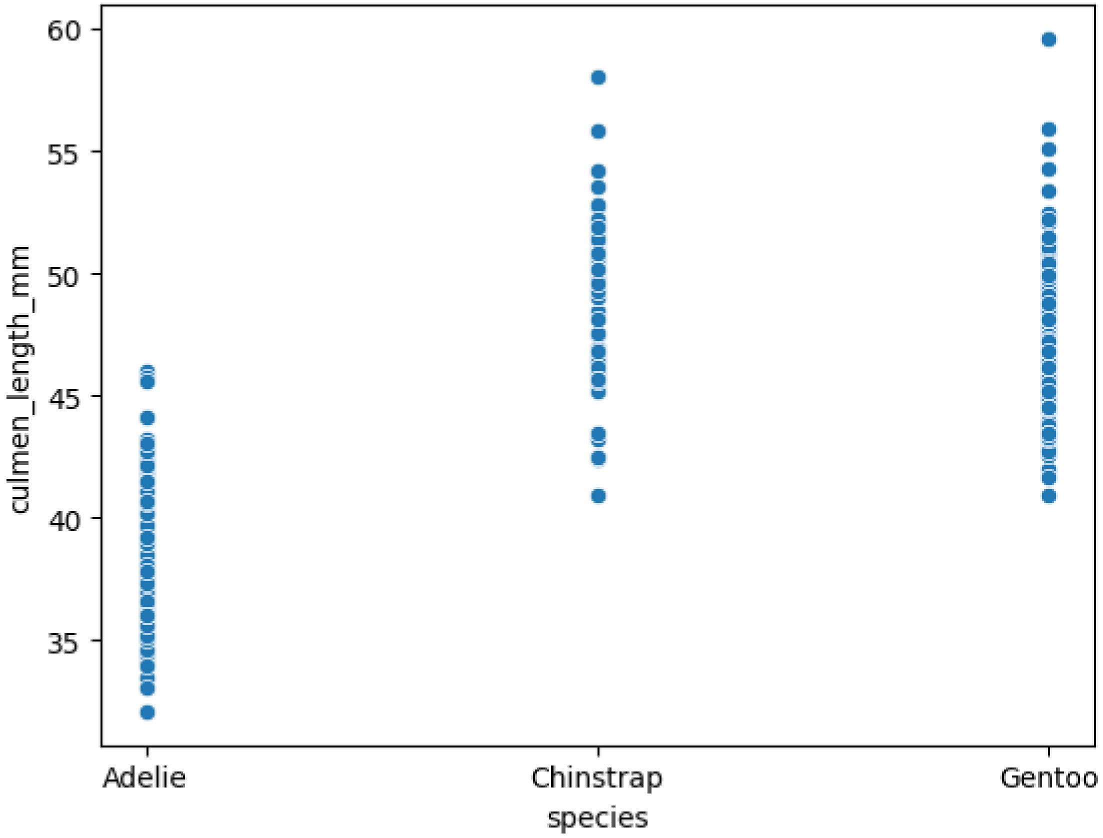
In [ ]:

```
plt.figure(figsize=(9,9))
plt.pie(df["island"].value_counts(),[0.1,0,0], labels = df["island"].unique(),au
plt.title('Islands percentage wrt Total')
plt.legend()
plt.show()
```

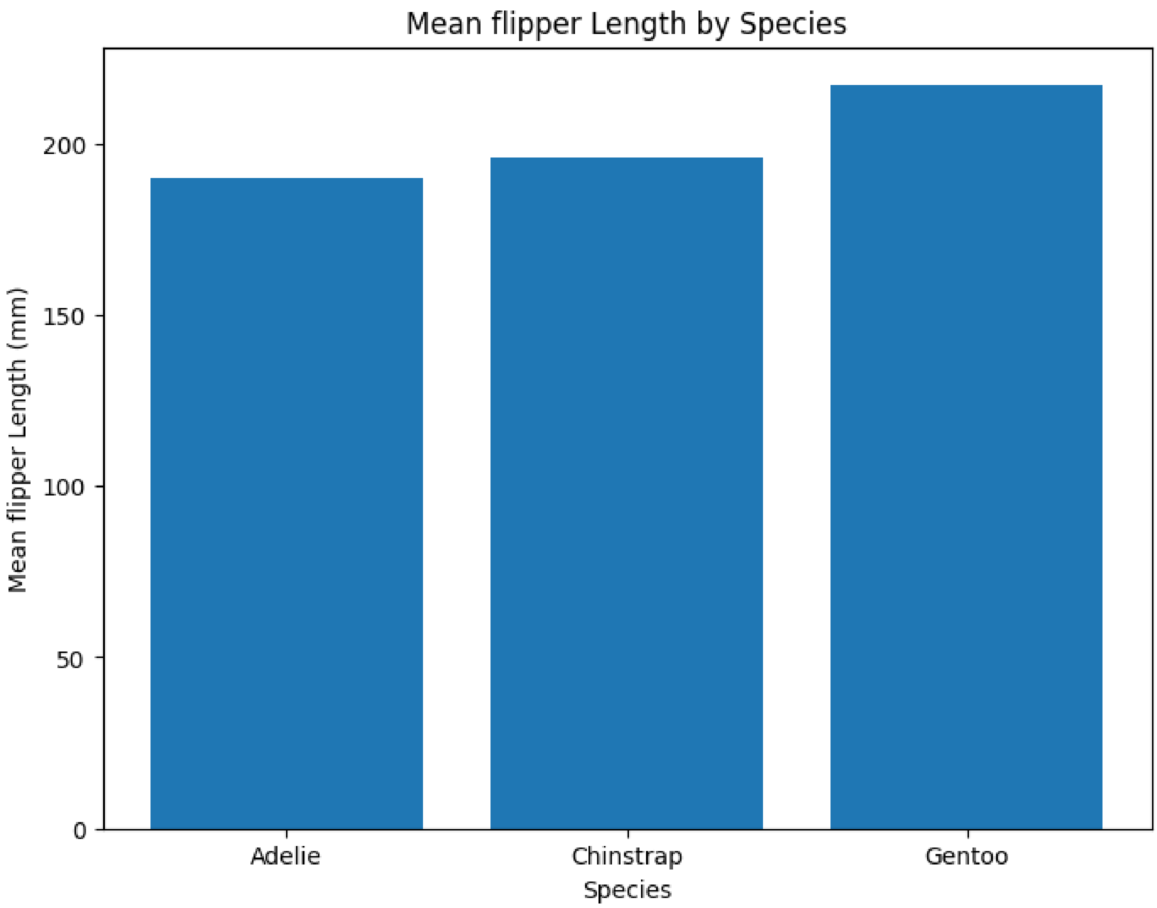


Bivariate

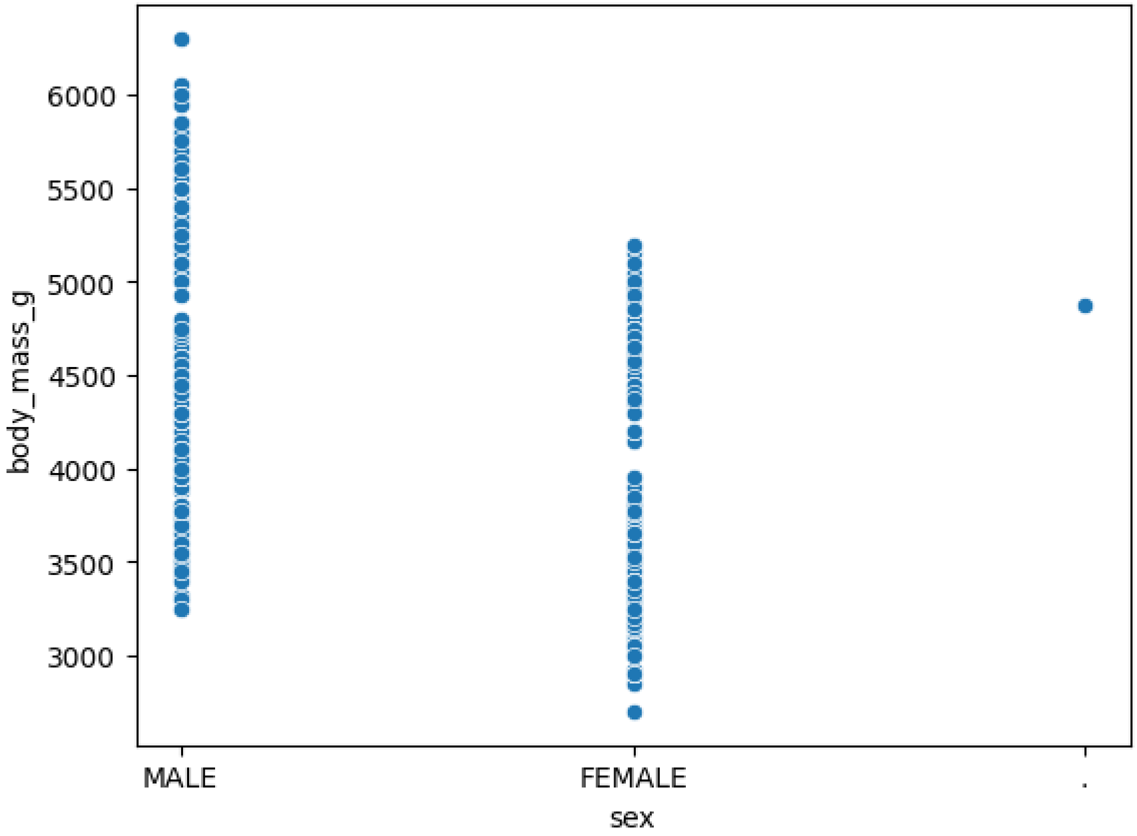
```
In [ ]: sns.scatterplot(x='species', y='culmen_length_mm', data=df)
        plt.show()
```



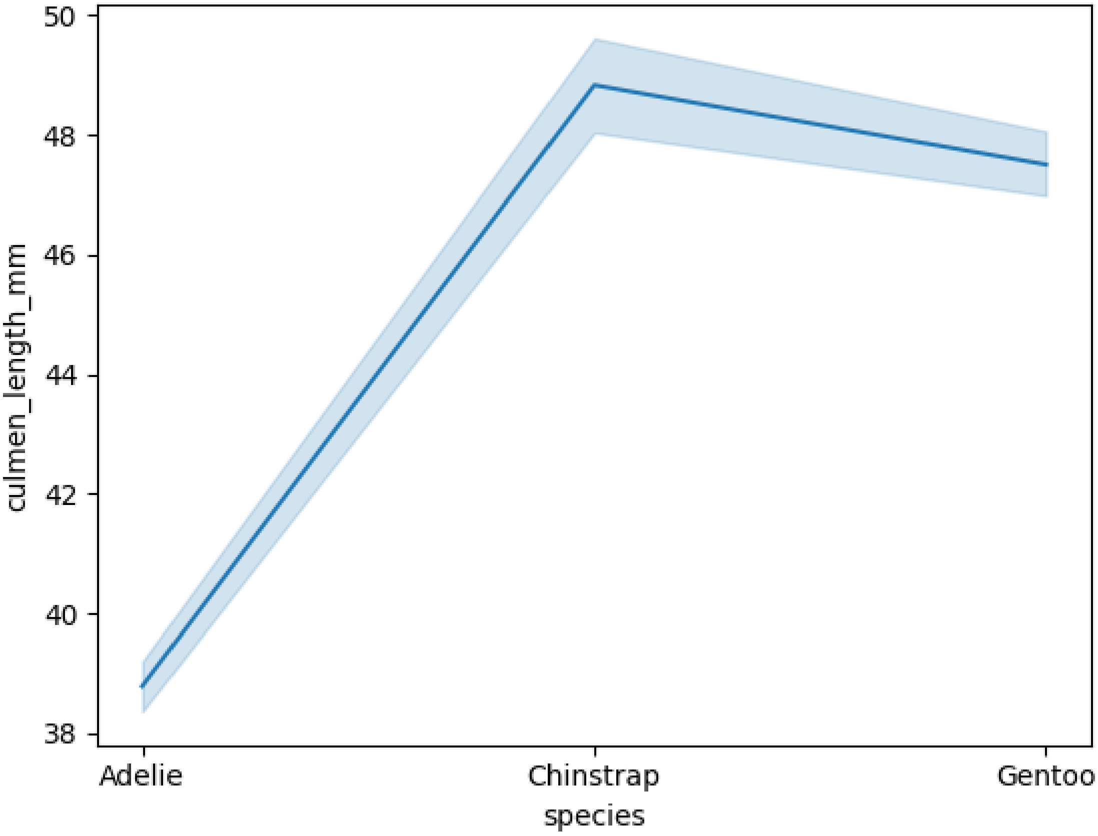
```
In [ ]: mean_flipper_length = df.groupby('species')['flipper_length_mm'].mean()
        plt.figure(figsize=(8, 6))
        plt.bar(mean_flipper_length.index, mean_flipper_length.values)
        plt.xlabel('Species')
        plt.ylabel('Mean flipper Length (mm)')
        plt.title('Mean flipper Length by Species')
        plt.show()
```



In [ ]:
```
sns.scatterplot(x='sex', y='body_mass_g', data=df)
plt.show()
```

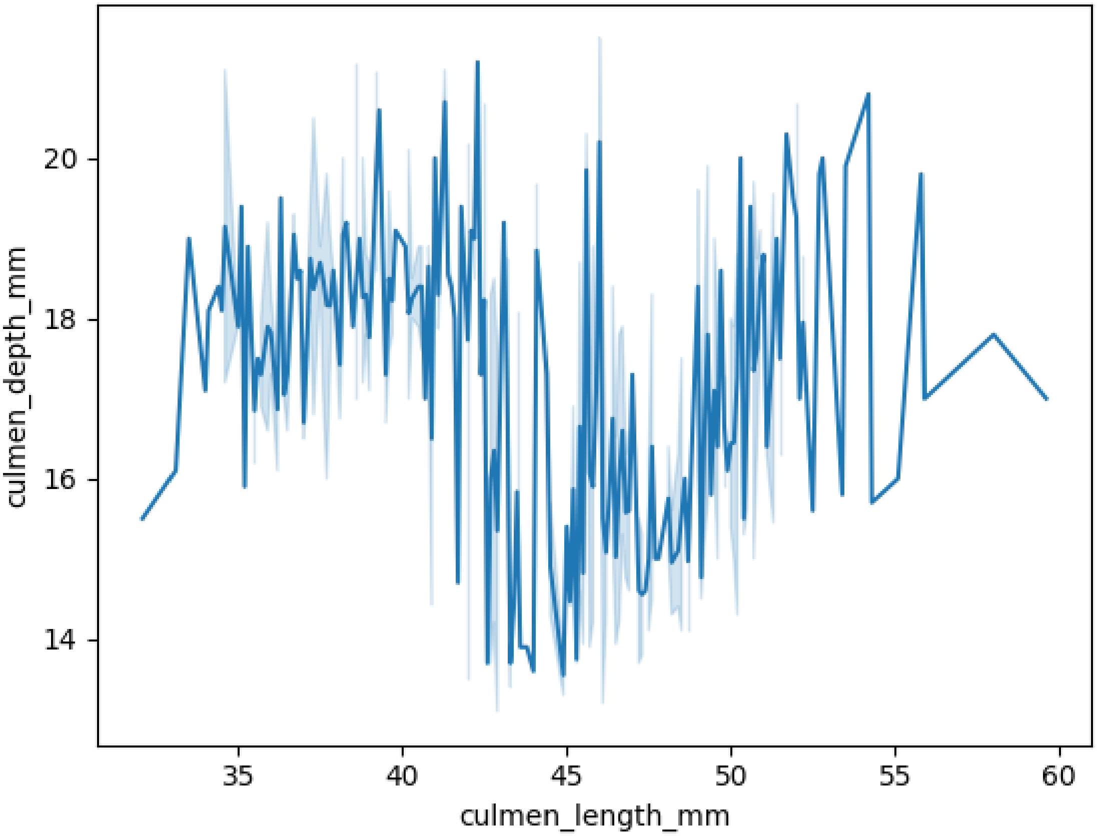


In [ ]:
```
sns.lineplot(x='species', y='culmen_length_mm', data=df)
plt.show()
```



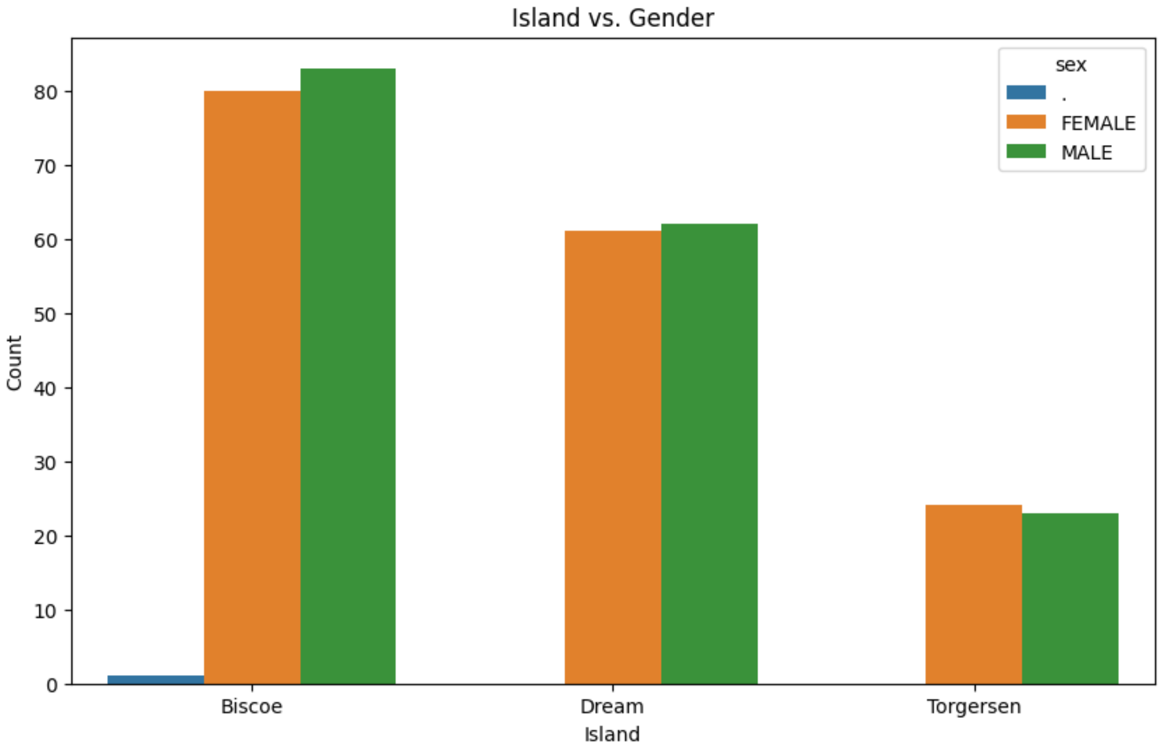
In [ ]:

```
sns.lineplot(x='culmen_length_mm', y='culmen_depth_mm', data=df)
plt.show()
```



In [ ]:

```
island_gender_counts = df.groupby(['island', 'sex']).size().reset_index(name='co
plt.figure(figsize=(10, 6))
sns.barplot(x='island', y='counts', hue='sex', data=island_gender_counts)
plt.xlabel('Island')
plt.ylabel('Count')
plt.title('Island vs. Gender')
plt.show()
```



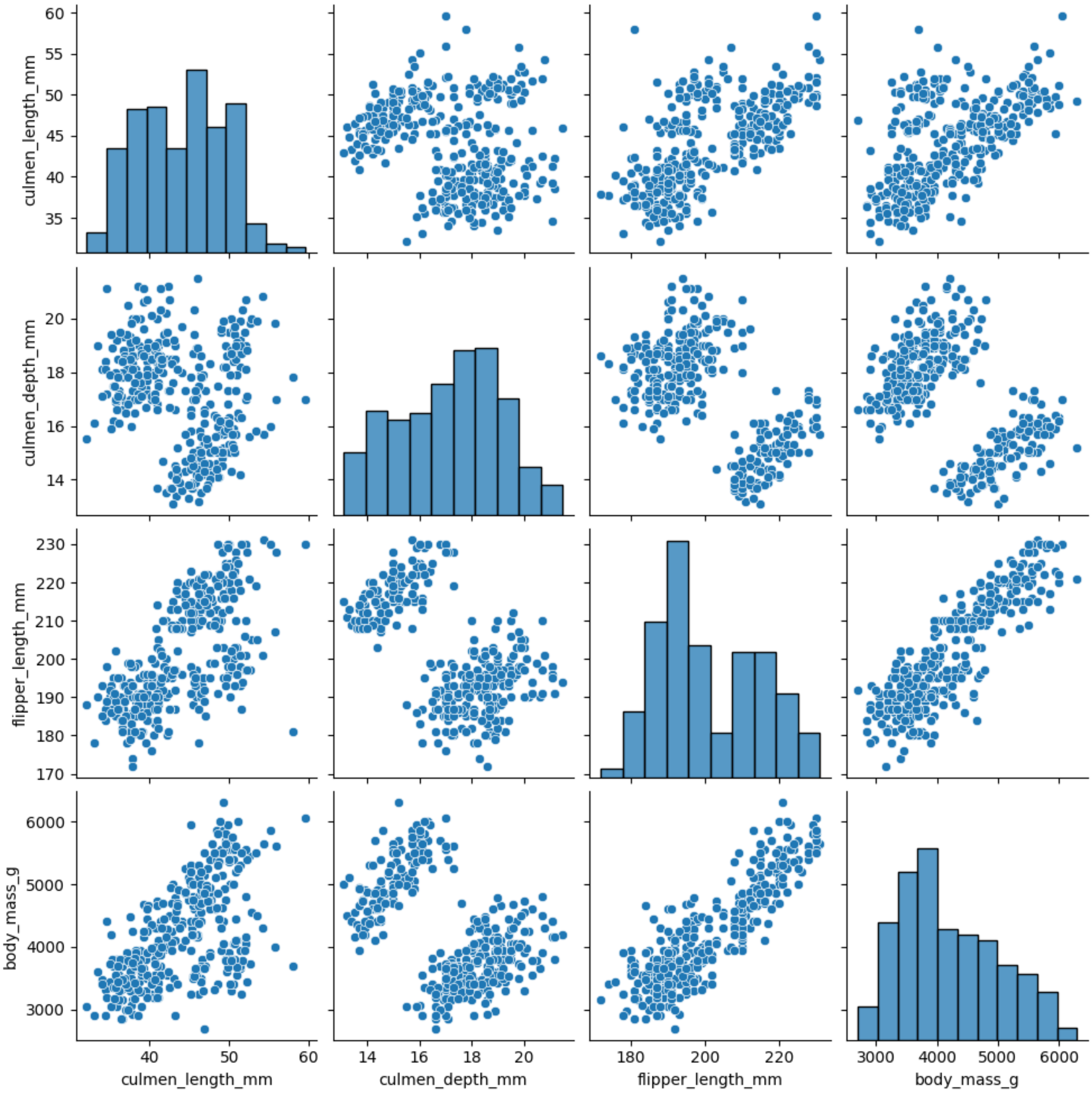
Multivariate

In [ ]:

```
sns.pairplot(df)
```

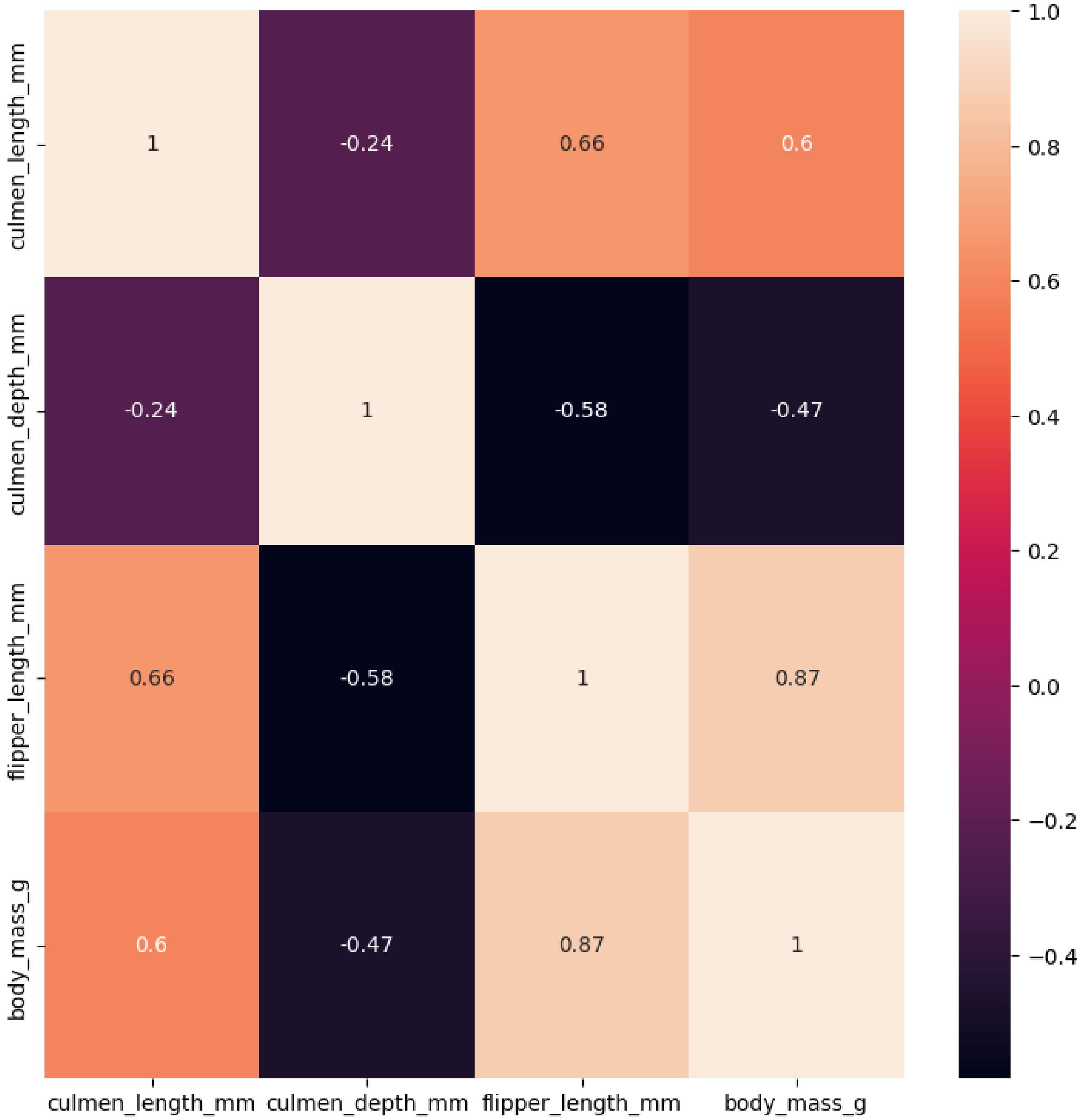
Out[ ]:

```
<seaborn.axisgrid.PairGrid at 0x1e0a12cc280>
```



In [ ]:
```
plt.figure(figsize=(9,9))
sns.heatmap(df.corr(), annot=True)
```

Out[ ]:
```
<AxesSubplot: >
```



4. Perform Descriptive statistics of the dataset

In [ ]:
```
df.describe()
```

Out[ ]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
count	342.000000	342.000000	342.000000	342.000000
mean	43.921930	17.151170	200.915205	4201.754386
std	5.459584	1.974793	14.061714	801.954536
min	32.100000	13.100000	172.000000	2700.000000
25%	39.225000	15.600000	190.000000	3550.000000
50%	44.450000	17.300000	197.000000	4050.000000
75%	48.500000	18.700000	213.000000	4750.000000
max	59.600000	21.500000	231.000000	6300.000000

5. Check for missing values and deal with them

```
In [ ]: df.isnull()
```

Out[ ]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_
0	False	False	False	False	False	
1	False	False	False	False	False	
2	False	False	False	False	False	
3	False	False	True	True	True	
4	False	False	False	False	False	
...	...	...	...	...	...	
339	False	False	True	True	True	
340	False	False	False	False	False	
341	False	False	False	False	False	
342	False	False	False	False	False	
343	False	False	False	False	False	

344 rows × 7 columns

```
In [ ]: df.isnull().sum()
```

```
Out[ ]: species              0
        island               0
        culmen_length_mm     2
        culmen_depth_mm      2
        flipper_length_mm    2
        body_mass_g          2
        sex                 10
        dtype: int64
```

```
In [ ]: df.isnull().any()
```

```
Out[ ]: species             False
        island              False
        culmen_length_mm     True
        culmen_depth_mm      True
        flipper_length_mm    True
        body_mass_g          True
        sex                  True
        dtype: bool
```

```
In [ ]: df.sex = df.sex.fillna(df.sex.mode()[0])
        df.culmen_length_mm = df.culmen_length_mm.fillna(df.culmen_length_mm.median())
        df.culmen_depth_mm = df.culmen_depth_mm.fillna(df.culmen_depth_mm.median())
        df.flipper_length_mm = df.flipper_length_mm.fillna(df.flipper_length_mm.median()
        df.body_mass_g = df.body_mass_g.fillna(df.body_mass_g.median())
```

In [ ]:

```
df.head()
```

Out[ ]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_
0	Adelie	Torgersen	39.10	18.7	181.0	
1	Adelie	Torgersen	39.50	17.4	186.0	
2	Adelie	Torgersen	40.30	18.0	195.0	
3	Adelie	Torgersen	44.45	17.3	197.0	
4	Adelie	Torgersen	36.70	19.3	193.0	

In [ ]:

```
df.isnull().any()
```

Out[ ]:

```
species              False
island               False
culmen_length_mm     False
culmen_depth_mm      False
flipper_length_mm    False
body_mass_g          False
sex                  False
dtype: bool
```

In [ ]:

```
df.head()
```

Out[ ]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_
0	Adelie	Torgersen	39.10	18.7	181.0	
1	Adelie	Torgersen	39.50	17.4	186.0	
2	Adelie	Torgersen	40.30	18.0	195.0	
3	Adelie	Torgersen	44.45	17.3	197.0	
4	Adelie	Torgersen	36.70	19.3	193.0	

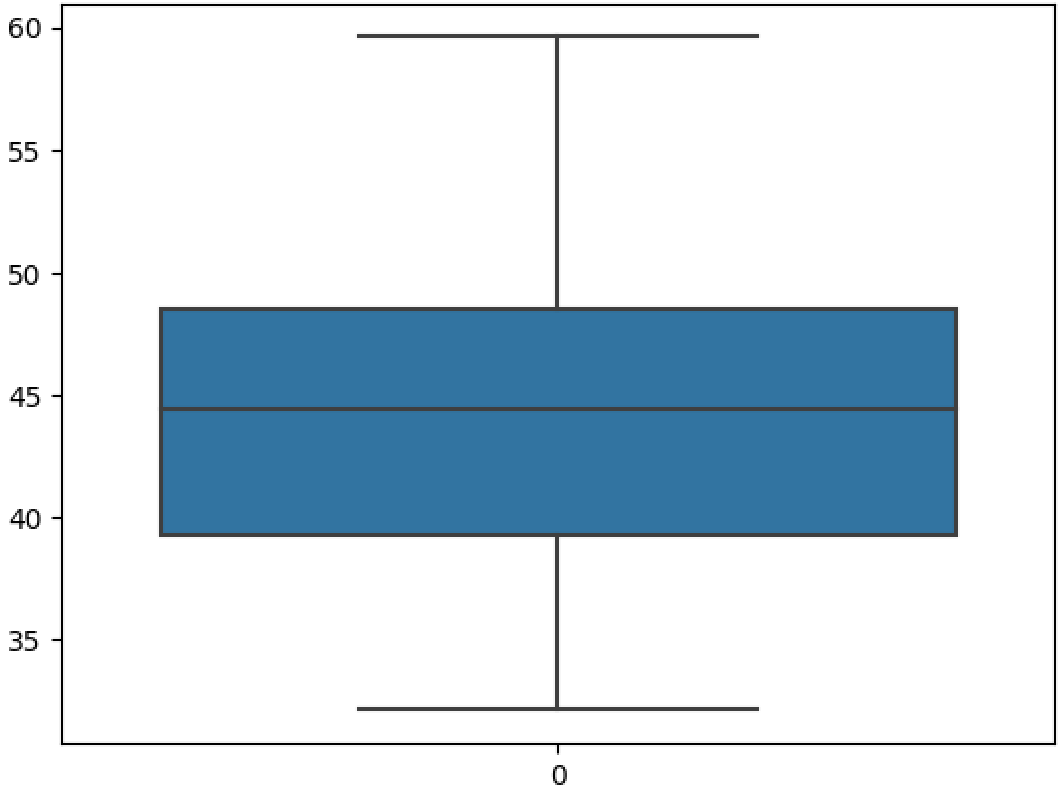
6. Find out the outliers and replace them

In [ ]:

```
sns.boxplot(df.culmen_length_mm)
```

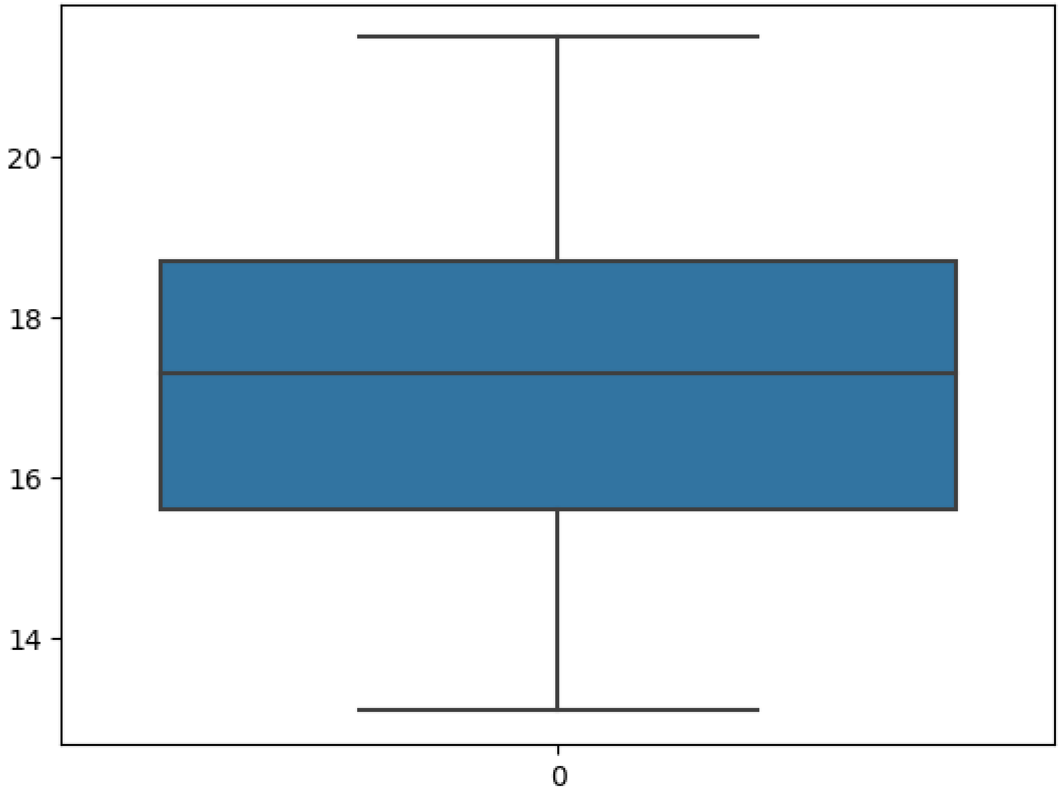
Out[ ]:

```
<AxesSubplot: >
```



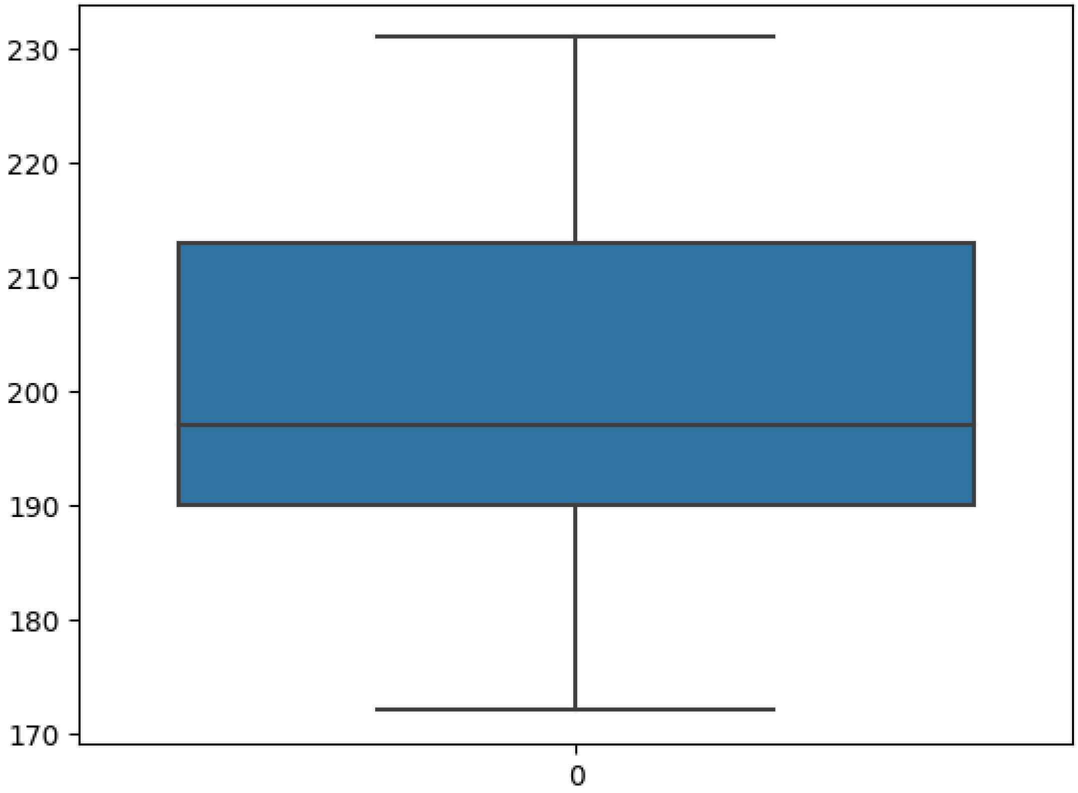
In [ ]:
```
sns.boxplot(df.culmen_depth_mm)
```

Out[ ]:
```
<AxesSubplot: >
```



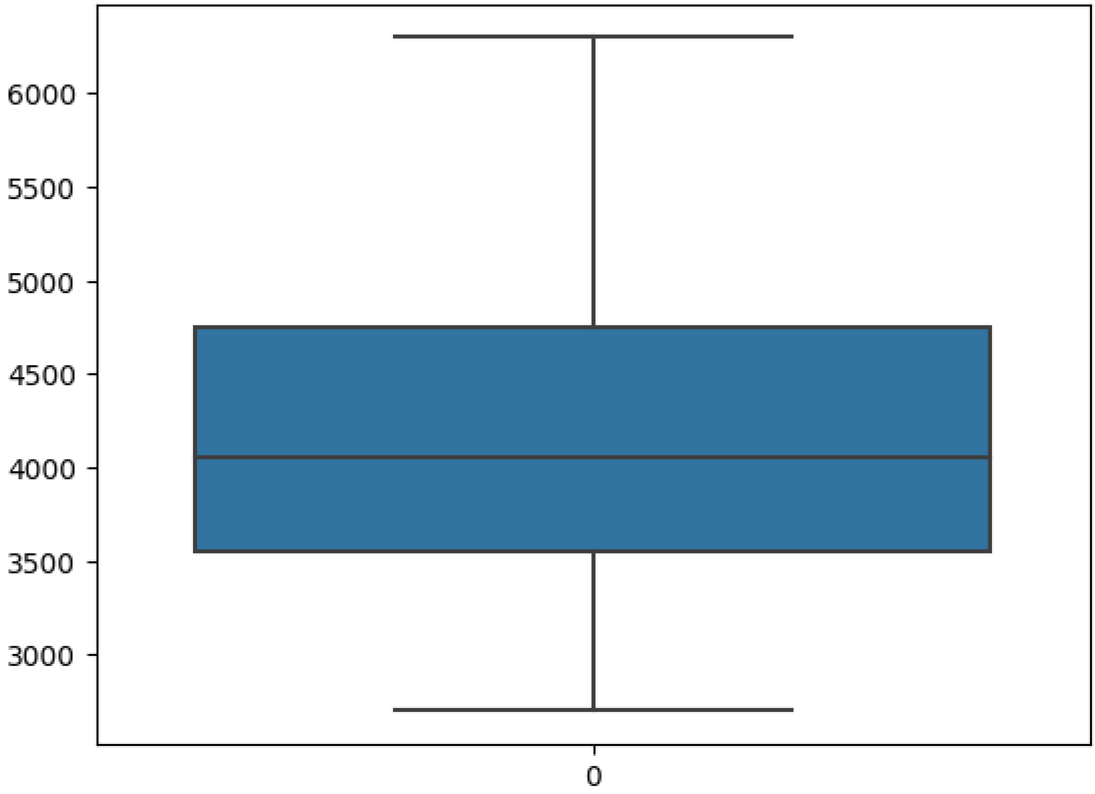
In [ ]:
```
sns.boxplot(df.flipper_length_mm)
```

Out[ ]:
```
<AxesSubplot: >
```



```
In [ ]: sns.boxplot(df.body_mass_g)
```

```
Out[ ]: <AxesSubplot: >
```



No outliers in any of the numerical columns. No need to do anything

7. and 8.

target is 'species' which has a dtype of 'object, which is categorical so we must first perform encoding and then only we can check correlation of independent variables with the target

In [ ]:

```
print(df['species'].dtype)
```

```
object
```

So first, we will check for categorical columns and perform encoding. Secondly, we will check the correlation of independent variables with the target.

target varibles --> species

In [ ]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 344 entries, 0 to 343
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   species            344 non-null    object 
 1   island             344 non-null    object 
 2   culmen_length_mm   344 non-null    float64
 3   culmen_depth_mm    344 non-null    float64
 4   flipper_length_mm  344 non-null    float64
 5   body_mass_g        344 non-null    float64
 6   sex                344 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df.sex = le.fit_transform(df.sex)
df.island = le.fit_transform(df.island)
df.species = le.fit_transform(df.species)
```

In [ ]:

```
df.head()
```

Out[ ]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_ma
0	0	2	39.10	18.7	181.0	37
1	0	2	39.50	17.4	186.0	38
2	0	2	40.30	18.0	195.0	32
3	0	2	44.45	17.3	197.0	40
4	0	2	36.70	19.3	193.0	34

In [ ]:

```
df.corr().species.sort_values(ascending=False)
```

Out[ ]:

```
species              1.000000
flipper_length_mm    0.850819
body_mass_g          0.747547
culmen_length_mm     0.728706
sex                 -0.003823
island              -0.635659
culmen_depth_mm     -0.741282
Name: species, dtype: float64
```

9. Split the data into independent and dependent variables

In [ ]:

```
X = df.drop(columns=['species'], axis=1)
y = df['species']
```

In [ ]:

```
X
```

Out[ ]:

	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
0	2	39.10	18.7	181.0	3750.0
1	2	39.50	17.4	186.0	3800.0
2	2	40.30	18.0	195.0	3250.0
3	2	44.45	17.3	197.0	4050.0
4	2	36.70	19.3	193.0	3450.0
...	...	...	...	...	...
339	0	44.45	17.3	197.0	4050.0
340	0	46.80	14.3	215.0	4850.0
341	0	50.40	15.7	222.0	5750.0
342	0	45.20	14.8	212.0	5200.0
343	0	49.90	16.1	213.0	5400.0

344 rows × 6 columns

In [ ]:

```
y
```

Out[ ]:

```
0      0
1      0
2      0
3      0
4      0
      ..
339    2
340    2
341    2
342    2
343    2
Name: species, Length: 344, dtype: int32
```

10. Scaling the data

In [ ]:
```
from sklearn.preprocessing import MinMaxScaler
scale =MinMaxScaler()
X_scaled= pd.DataFrame(scale.fit_transform(X),columns =X.columns)
X_scaled.head()
```

Out[ ]:

	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g	sex
0	1.0	0.254545	0.666667	0.152542	0.291667	1.0
1	1.0	0.269091	0.511905	0.237288	0.305556	0.5
2	1.0	0.298182	0.583333	0.389831	0.152778	0.5
3	1.0	0.449091	0.500000	0.423729	0.375000	1.0
4	1.0	0.167273	0.738095	0.355932	0.208333	0.5

11. Split the data into training and testing

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.3,random
```

In [ ]:
```
X_train
```

Out[ ]:

	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
258	0.0	0.432727	0.059524	0.610169	0.458333
332	0.0	0.414545	0.250000	0.694915	0.541667
121	1.0	0.203636	0.797619	0.440678	0.222222
61	0.0	0.334545	0.952381	0.389831	0.472222
70	1.0	0.050909	0.702381	0.305085	0.250000
...	...	...	...	...	...
123	1.0	0.338182	0.642857	0.508475	0.326389
320	0.0	0.596364	0.226190	0.796610	0.597222
15	1.0	0.163636	0.559524	0.220339	0.277778
125	1.0	0.309091	0.702381	0.457627	0.361111
265	0.0	0.418182	0.095238	0.762712	0.611111

240 rows × 6 columns

In [ ]:
```
X_test
```

Out[ ]:

	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
229	0.0	0.534545	0.273810	0.728814	0.680556
80	1.0	0.090909	0.488095	0.288136	0.138889
327	0.0	0.774545	0.321429	0.796610	0.777778
6	1.0	0.247273	0.559524	0.152542	0.256944
309	0.0	0.727273	0.464286	0.983051	0.791667
...	...	...	...	...	...
211	0.5	0.490909	0.750000	0.372881	0.229167
311	0.0	0.730909	0.476190	0.949153	0.750000
19	1.0	0.505455	1.000000	0.372881	0.416667
270	0.0	0.527273	0.130952	0.644068	0.597222
194	0.5	0.683636	0.714286	0.406780	0.236111

104 rows × 6 columns

In [ ]:

```
y_train
```

Out[ ]:

```
258    2
332    2
121    0
61     0
70     0
      ..
123    0
320    2
15     0
125    0
265    2
Name: species, Length: 240, dtype: int32
```

In [ ]:

```
y_test
```

Out[ ]:

```
229    2
80     0
327    2
6      0
309    2
      ..
211    1
311    2
19     0
270    2
194    1
Name: species, Length: 104, dtype: int32
```

12. Check the training and test data shape

```
In [ ]: print("X_train shape -> ",X_train.shape)
        print("X_test shape -> ", X_test.shape)
        print("y_train shape -> ", y_train.shape)
        print("y_test shape -> ", y_test.shape)
```

```
X_train shape ->  (240, 6)
X_test shape ->  (104, 6)
y_train shape ->  (240,)
y_test shape ->  (104,)
```

Completed - HRISHIKESH G KULKARNI (21BAI1660)

assignment03 completed

hrishikesh.gkulkarni2021@vitstudent.ac.in

==X=======