SYAM KRISHNA REDDY PULAGAM

REG NO: 21BAI1725 VIT CHENNAI CAMPUS

AI&ML ASSIGNMENT-3

1. IMPORTING ALL THE NECCESARY LIBRARIES AND DOWNLOADING THE DATASET

In [2]:

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

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm

from sklearn.metrics import accuracy_score
from sklearn import metrics
from sklearn.metrics import confusion_matrix
```

2. Loading the dataset

In [10]:

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

Out[10]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass
0	Adelie	Torgersen	39.1	18.7	181.0	375(
1	Adelie	Torgersen	39.5	17.4	186.0	380(
2	Adelie	Torgersen	40.3	18.0	195.0	325(
3	Adelie	Torgersen	NaN	NaN	NaN	N
4	Adelie	Torgersen	36.7	19.3	193.0	345(

In [11]:

```
df.dtypes
```

Out[11]:

```
species              object
island               object
culmen_length_mm    float64
culmen_depth_mm     float64
flipper_length_mm   float64
body_mass_g         float64
sex                  object
dtype: object
```

3. VISUALIZATIONS

- Univariate Analysis
- Bi- Variate Analysis
- Multi-Variate Analysis

Univariate Analysis

In [12]:

```
df['species'].value_counts()
```

Out[12]:

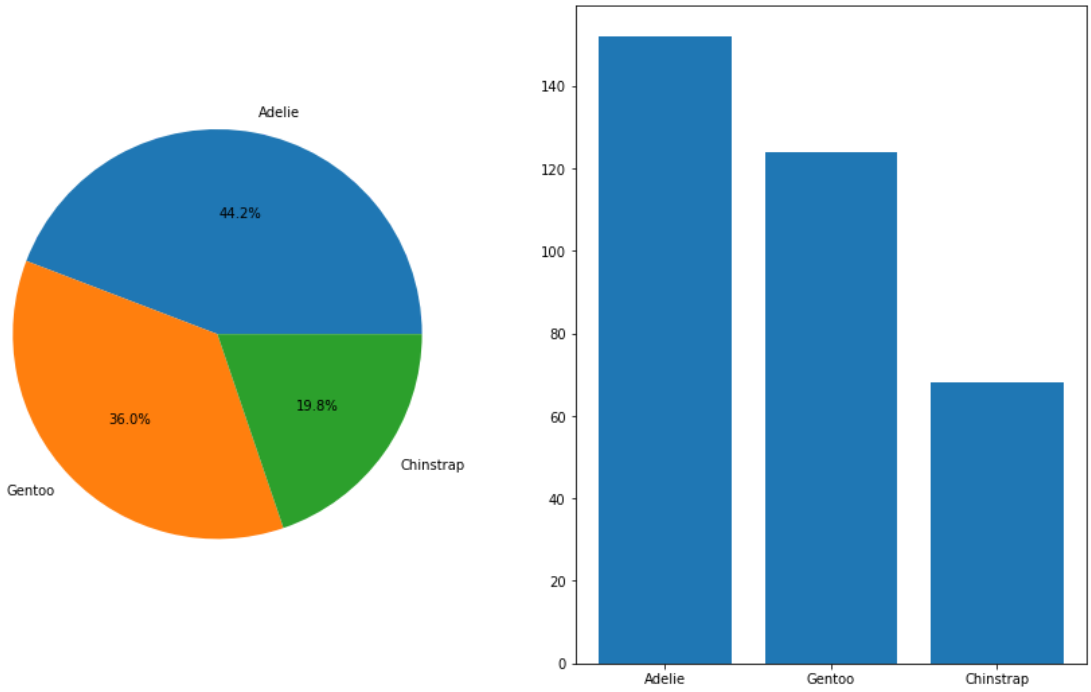
```
Adelie       152
Gentoo       124
Chinstrap     68
Name: species, dtype: int64
```

In [254]:

```
a=df['species'].value_counts().index
fig,ax = plt.subplots(1,2, figsize=(15,9))
ax[0].pie(df['species'].value_counts(),labels=a,autopct = "%1.1f%%")
ax[1].bar(a,df['species'].value_counts())
```

Out[254]:

```
<BarContainer object of 3 artists>
```



In [255]:

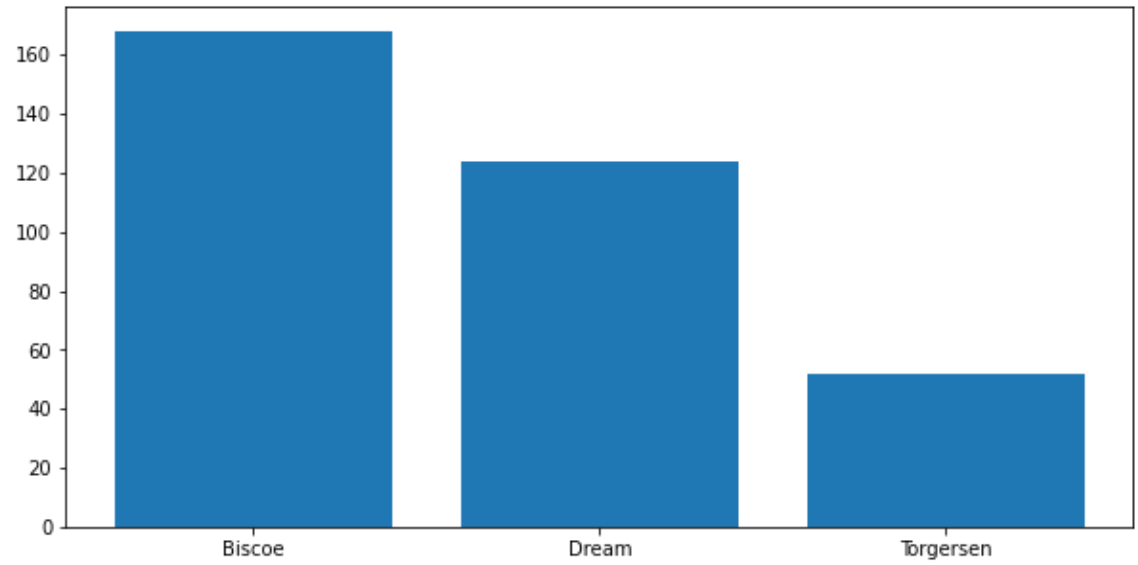
```
df['island'].value_counts()
```

Out[255]:

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

In [256]:

```
plt.figure(figsize=(10,5))
a=df['island'].value_counts().index
plt.bar(a,df['island'].value_counts())
plt.show()
```

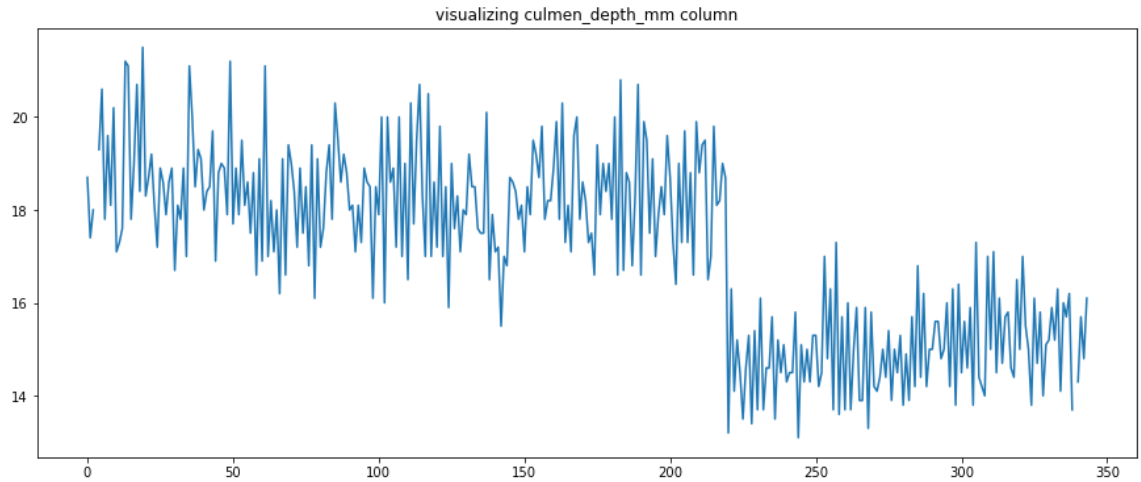


In [257]:

```
plt.figure(figsize=(15,6))
plt.plot(df['culmen_depth_mm'])
plt.title("visualizing culmen_depth_mm column")
```

Out[257]:

```
Text(0.5, 1.0, 'visualizing culmen_depth_mm column')
```



In [258]:

```
df1=df.sex.dropna()
```

In [259]:

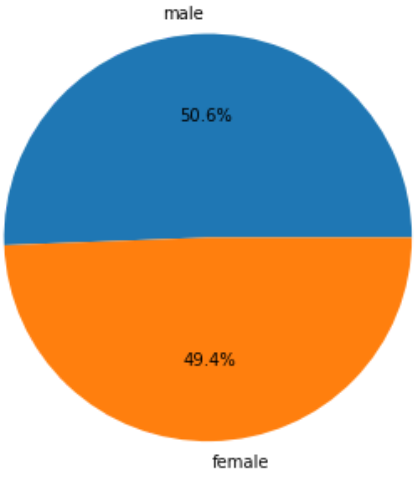
```
df1=df1.replace(".","MALE")
df1.value_counts()
```

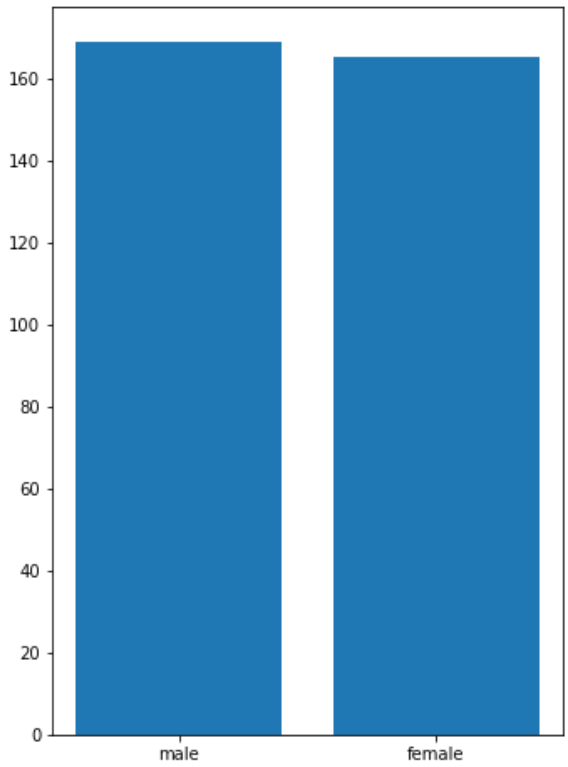
Out[259]:

```
MALE      169
FEMALE    165
Name: sex, dtype: int64
```

In [260]:

```
fig,ax = plt.subplots(1,2, figsize=(12,8))
a=['male','female']
ax[0].pie(df1.value_counts(),labels=a,autopct = "%1.1f%%")
ax[1].bar(a,df1.value_counts())
plt.show()
```





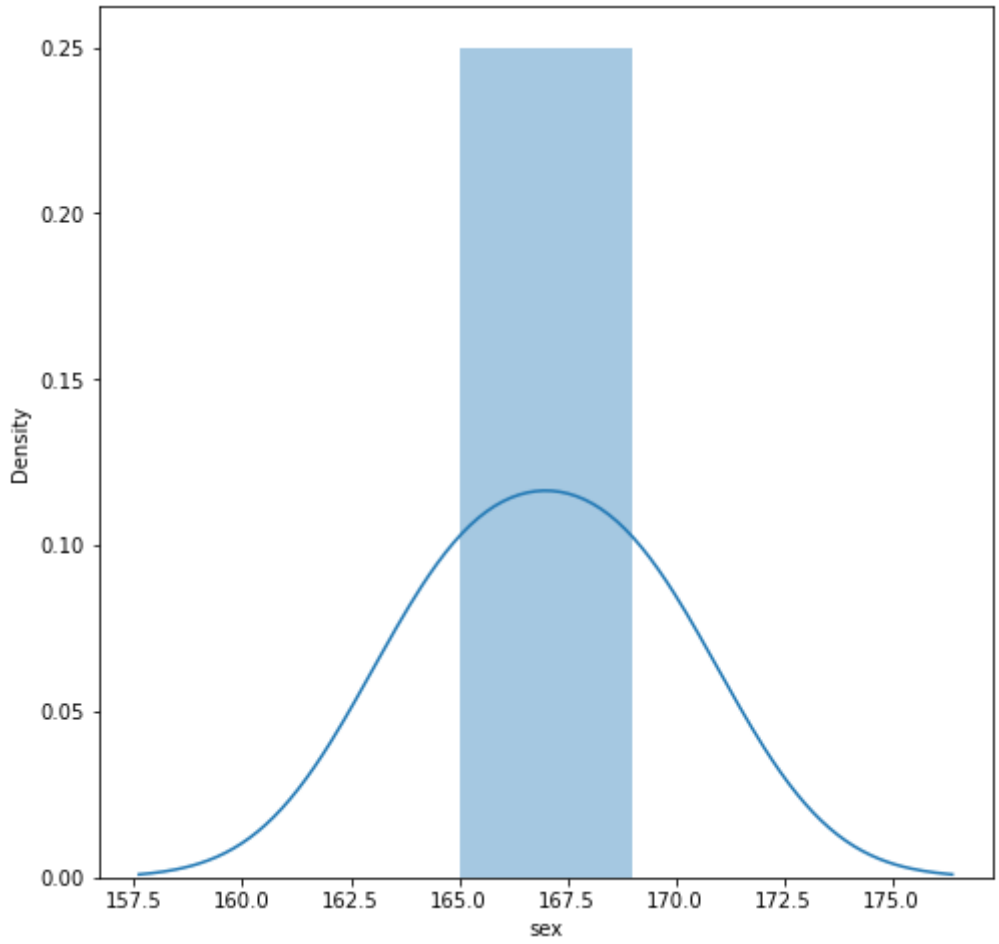
In [261]:

```
sns.distplot(df1.value_counts())
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a f
uture version. Please adapt your code to use either `displot` (a figure-le
vel function with similar flexibility) or `histplot` (an axes-level functi
on for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[261]:

```
<AxesSubplot:xlabel='sex', ylabel='Density'>
```



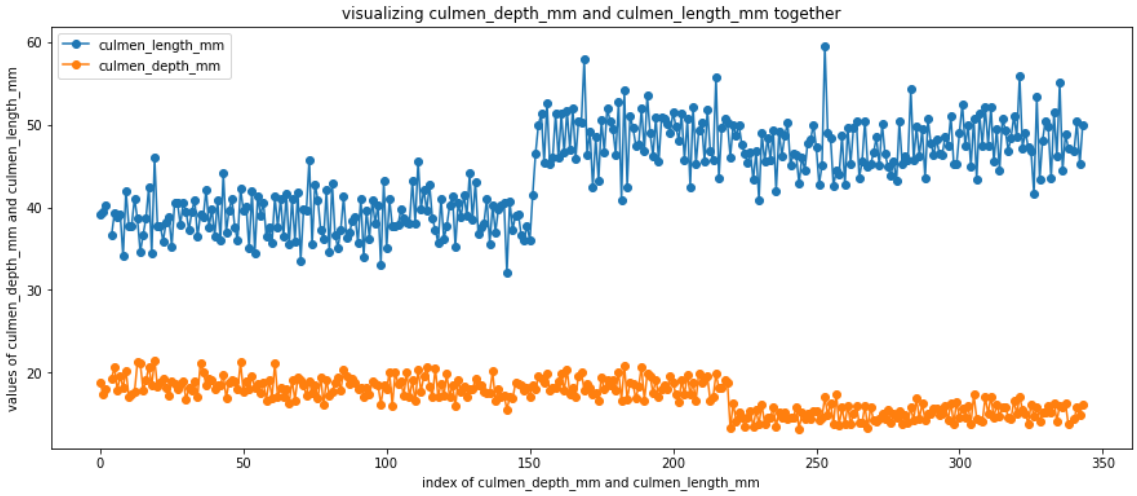
Bi- Variate Analysis

In [262]:

```
plt.figure(figsize=(15,6))
plt.plot(df['culmen_length_mm'],'o-')
plt.plot(df['culmen_depth_mm'],'o-')
plt.title("visualizing culmen_depth_mm and culmen_length_mm together")
plt.xlabel("index of culmen_depth_mm and culmen_length_mm")
plt.ylabel("values of culmen_depth_mm and culmen_length_mm")
plt.legend(['culmen_length_mm','culmen_depth_mm'])
```

Out[262]:

```
<matplotlib.legend.Legend at 0x1c1d4aad850>
```

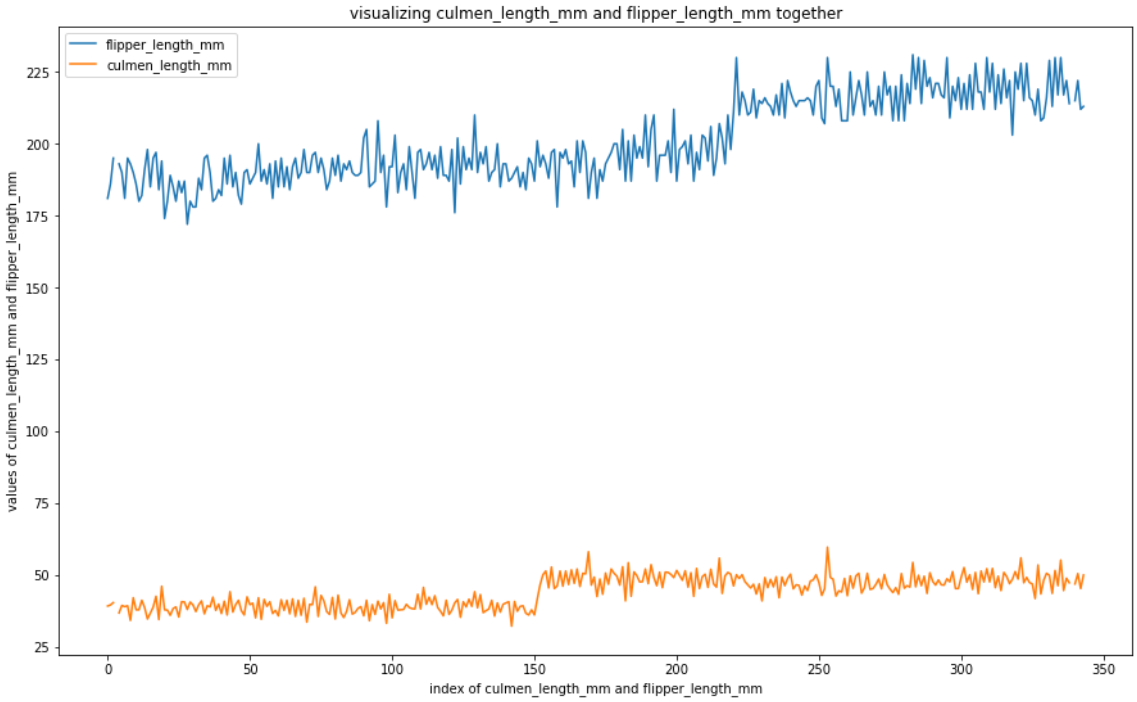


In [263]:

```
plt.figure(figsize=(15,9))
plt.plot(df['flipper_length_mm'])
plt.plot(df['culmen_length_mm'])
plt.title("visualizing culmen_length_mm and flipper_length_mm together")
plt.xlabel("index of culmen_length_mm and flipper_length_mm")
plt.ylabel("values of culmen_length_mm and flipper_length_mm")
plt.legend(['flipper_length_mm','culmen_length_mm'])
```

Out[263]:

```
<matplotlib.legend.Legend at 0x1c1d5094fa0>
```

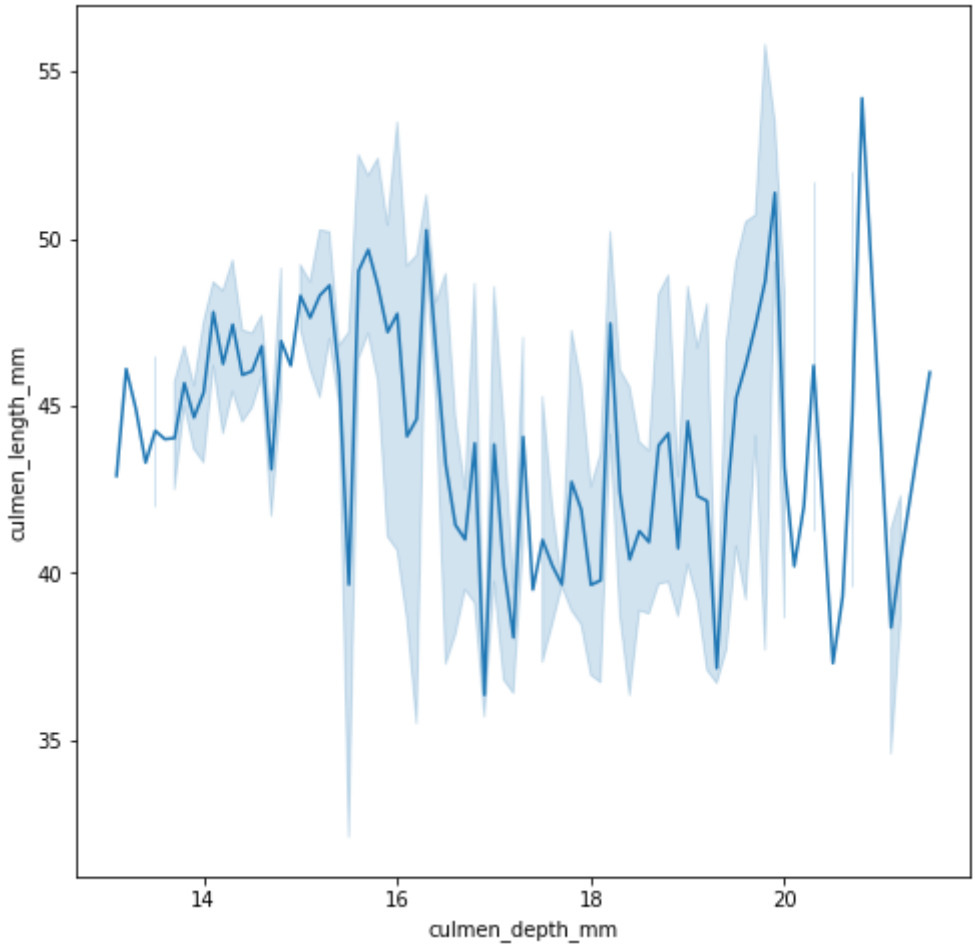


In [264]:

```
sns.lineplot(x = df.culmen_depth_mm,y=df.culmen_length_mm)
```

Out[264]:

```
<AxesSubplot:xlabel='culmen_depth_mm', ylabel='culmen_length_mm'>
```

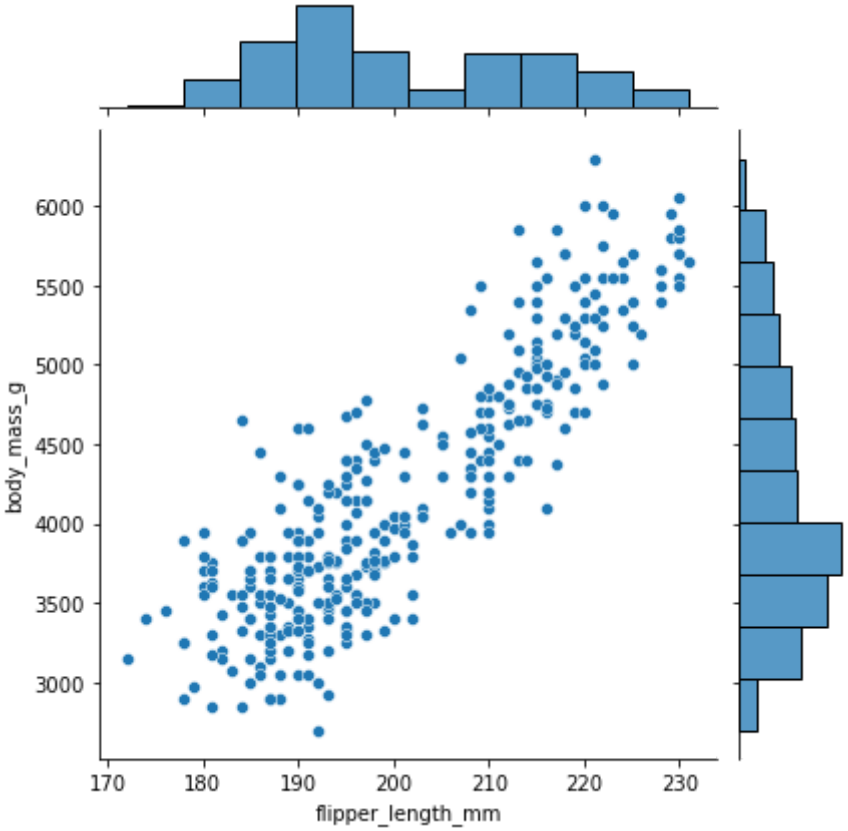


In [265]:

```
sns.jointplot(x= 'flipper_length_mm',y ='body_mass_g',data=df)
```

Out[265]:

```
<seaborn.axisgrid.JointGrid at 0x1c1d5142970>
```



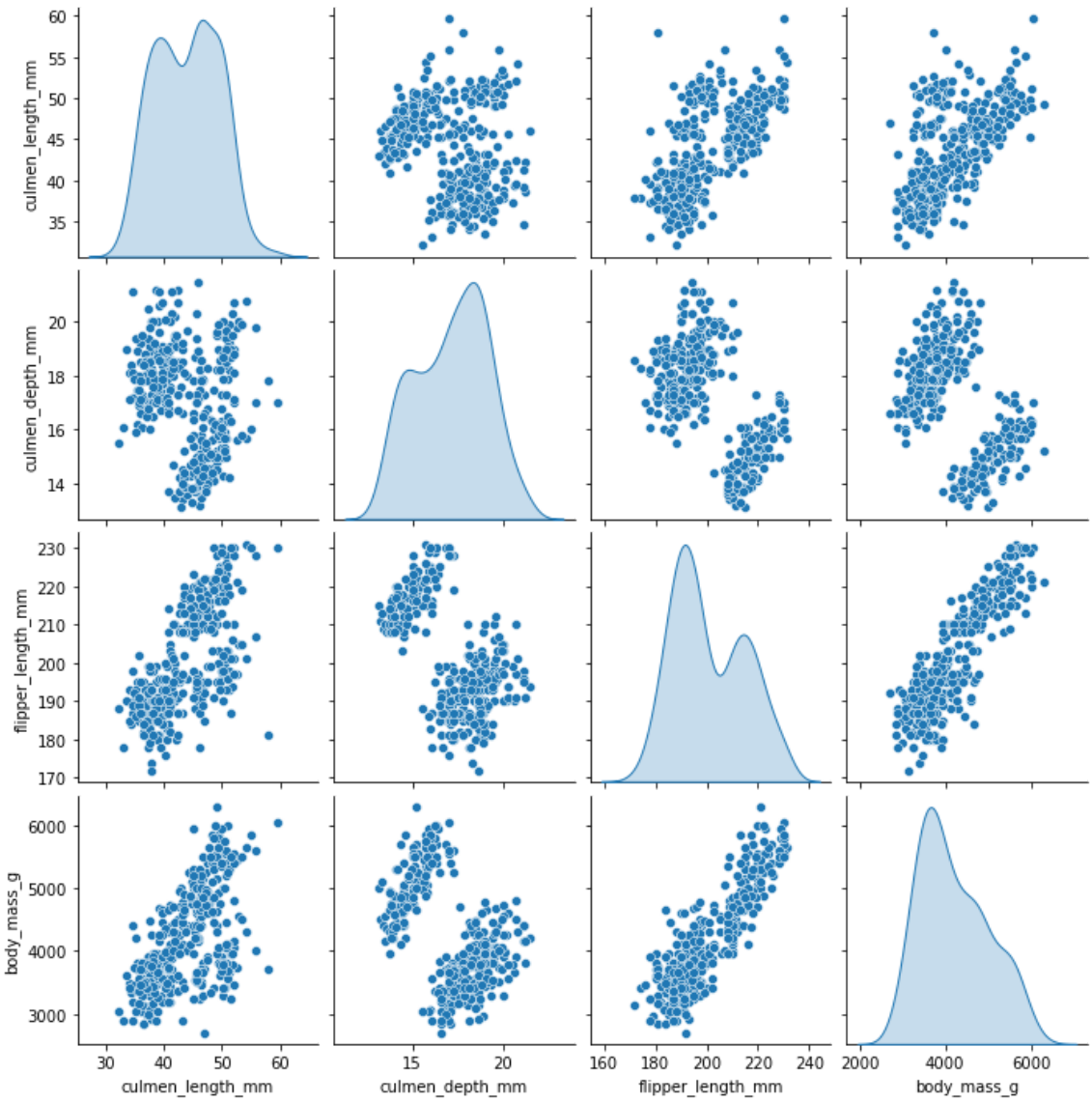
Multi-Variate Analysis

In [266]:

```
sns.pairplot(df,diag_kind='kde')
```

Out[266]:

```
<seaborn.axisgrid.PairGrid at 0x1c1d5231a30>
```

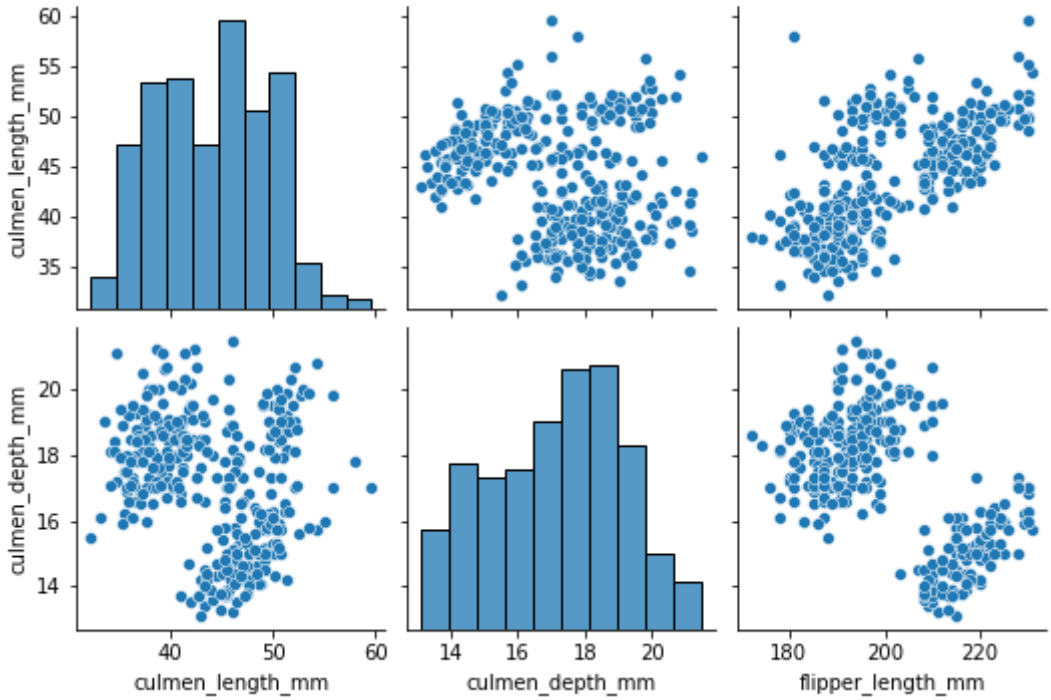


In [267]:

```
sns.pairplot(
 df,
 x_vars=['culmen_length_mm', 'culmen_depth_mm','flipper_length_mm'],
 y_vars=['culmen_length_mm', 'culmen_depth_mm']
)
```

Out[267]:

```
<seaborn.axisgrid.PairGrid at 0x1c1d6d6bf70>
```



In [268]:

```
df.corr()
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_15880/1134722465.py:1: 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 s
pecify the value of numeric_only to silence this warning.
  df.corr()
```

Out[268]:

	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
culmen_length_mm	1.000000	-0.235053	0.656181	0.595110
culmen_depth_mm	-0.235053	1.000000	-0.583851	-0.471916
flipper_length_mm	0.656181	-0.583851	1.000000	0.871202
body_mass_g	0.595110	-0.471916	0.871202	1.000000

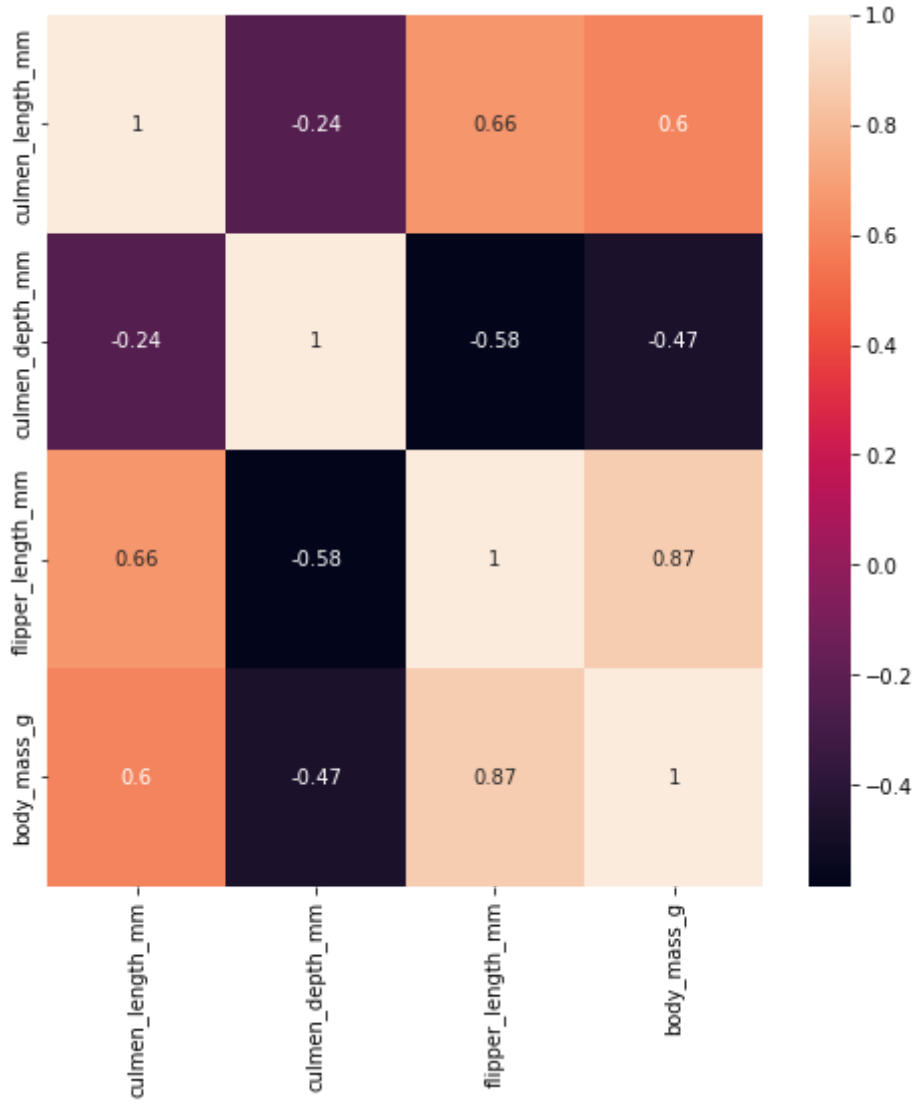
In [269]:

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

```
C:\Users\HP\AppData\Local\Temp/ipykernel_15880/4277794465.py:1: 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 s
pecify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[269]:

```
<AxesSubplot:>
```

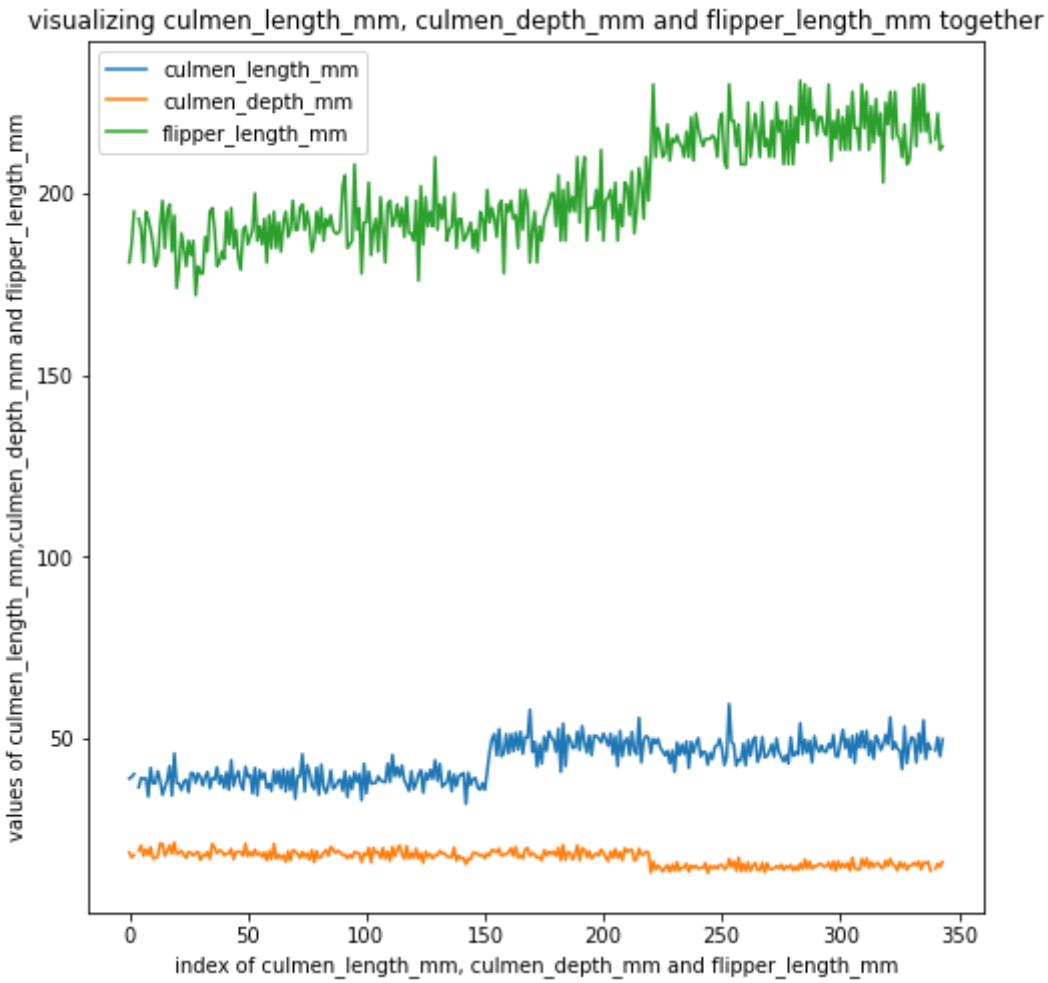


In [270]:

```
plt.plot(df['culmen_length_mm'])
plt.plot(df['culmen_depth_mm'])
plt.plot(df['flipper_length_mm'])
plt.title("visualizing culmen_length_mm, culmen_depth_mm and flipper_length_mm together"
plt.xlabel("index of culmen_length_mm, culmen_depth_mm and flipper_length_mm")
plt.ylabel("values of culmen_length_mm,culmen_depth_mm and flipper_length_mm")
plt.legend(['culmen_length_mm','culmen_depth_mm','flipper_length_mm'])
```

Out[270]:

```
<matplotlib.legend.Legend at 0x1c1d7595fd0>
```



4. DESCRIPTIVE STATISTICS ON OUR DATASET

In [271]:

```
df.columns
```

Out[271]:

```
Index(['species', 'island', 'culmen_length_mm', 'culmen_depth_mm',
       'flipper_length_mm', 'body_mass_g', 'sex'],
      dtype='object')
```

In [272]:

```
df.shape
```

Out[272]:

```
(344, 7)
```

In [273]:

```
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   342 non-null    float64
 3   culmen_depth_mm    342 non-null    float64
 4   flipper_length_mm  342 non-null    float64
 5   body_mass_g        342 non-null    float64
 6   sex                334 non-null    object 
dtypes: float64(4), object(3)
memory usage: 18.9+ KB
```

In [274]:

```
df.dtypes
```

Out[274]:

```
species               object
island                object
culmen_length_mm     float64
culmen_depth_mm      float64
flipper_length_mm    float64
body_mass_g          float64
sex                   object
dtype: object
```

In [275]:

```
df.describe()
```

Out[275]:

	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. CHECKING FOR MISSING VALUES AND IMPUTING THEM WITH STATISTICAL METHODS

(MEAN FOR NUMERIC DATA AND MODE FOR CATEGORICAL DATA)

In [276]:

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

Out[276]:

```
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 [277]:

```
df.isnull().sum() # columns except species and island are having null values.
#except sex columns remaining columns are of type float
```

Out[277]:

```
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 [4]:

```
x=['culmen_length_mm', 'culmen_depth_mm','flipper_length_mm', 'body_mass_g']
for column in x:
    df[column].fillna(df[column].mean(),inplace =True)
df['sex'].fillna(df['sex'].mode()[0],inplace =True)
```

In [279]:

```
#again checking for null values
df.isnull().sum()
```

Out[279]:

```
species              0
island               0
culmen_length_mm     0
culmen_depth_mm      0
flipper_length_mm    0
body_mass_g          0
sex                  0
dtype: int64
```

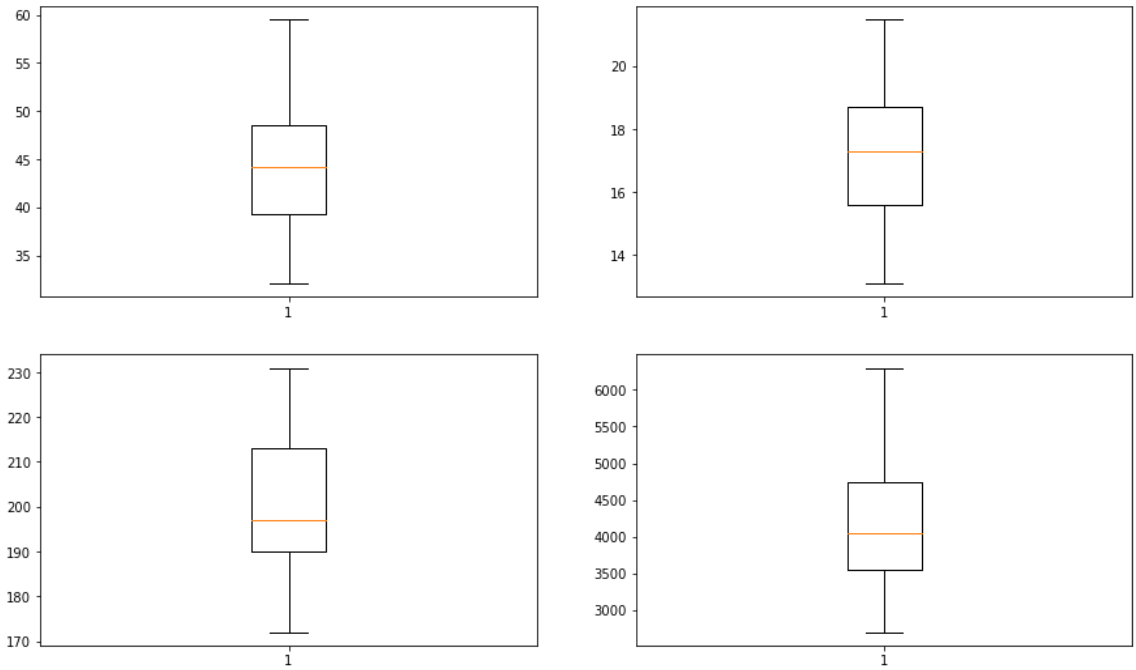
6. FINDING THE OUTLIERS AND REPLACING THEM

In [280]:

```
fig,ax = plt.subplots(2,2, figsize=(15,9))
ax[0][0].boxplot(df['culmen_length_mm'])
ax[0][1].boxplot(df['culmen_depth_mm'])
ax[1][0].boxplot(df['flipper_length_mm'])
ax[1][1].boxplot(df['body_mass_g']) #no outliers in following columns
```

Out[280]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1c1d75d5eb0>,
  <matplotlib.lines.Line2D at 0x1c1d75e2280>],
 'caps': [<matplotlib.lines.Line2D at 0x1c1d75e2610>,
  <matplotlib.lines.Line2D at 0x1c1d75e29a0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1c1d75d5b20>],
 'medians': [<matplotlib.lines.Line2D at 0x1c1d75e2d30>],
 'fliers': [<matplotlib.lines.Line2D at 0x1c1d75ef100>],
 'means': []}
```

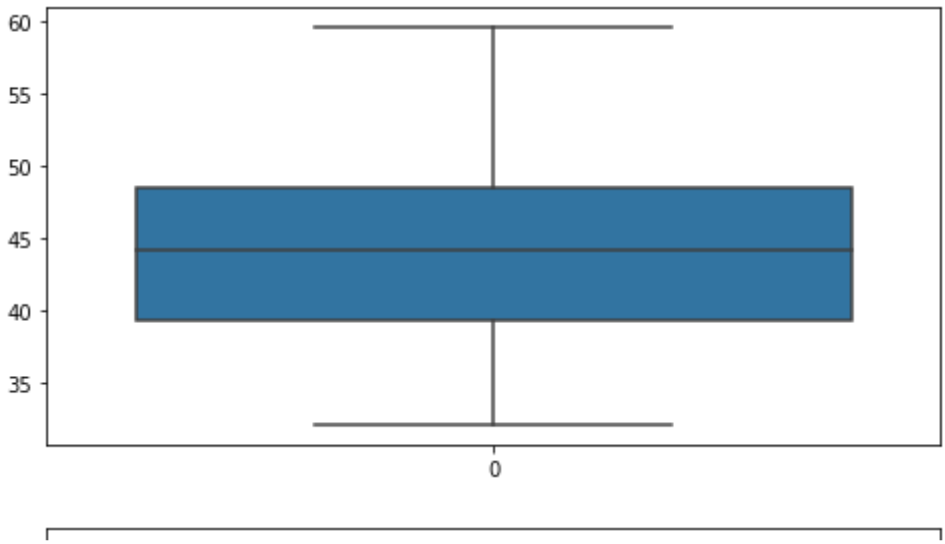


In [281]:

```
plt.figure(figsize=(8,4))
sns.boxplot(data=df['culmen_length_mm'])
plt.figure(figsize=(8,4))
sns.boxplot(data=df['culmen_depth_mm'])
plt.figure(figsize=(8,4))
sns.boxplot(data=df['flipper_length_mm'])
plt.figure(figsize=(8,4))
sns.boxplot(data=df['body_mass_g'])
```

Out[281]:

<AxesSubplot:>



In [282]:

```
def funct(col):
    print(col +"\n")
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    print(f" First quartile of {col} is q1= {q1} \n Second quartile of {col} is q3= {q3}
    iqr=q3-q1
    print(f" IQR OF {col} is {iqr}")
    upper_limit = q3+1.5*iqr
    lower_limit =q1-1.5*iqr
    print(f" Upper limit of {col} is: {upper_limit} \n Lower limit of {col} is: {lower_l
    print()
```

In [283]:

```
z=['culmen_length_mm', 'culmen_depth_mm','flipper_length_mm', 'body_mass_g']
for col in z:
    funct(col)
```

```
culmen_length_mm

 First quartile of culmen_length_mm is q1= 39.275
 Second quartile of culmen_length_mm is q3= 48.5
 IQR OF culmen_length_mm is 9.225000000000001
 Upper limit of culmen_length_mm is: 62.337500000000006
 Lower limit of culmen_length_mm is: 25.437499999999996

culmen_depth_mm

 First quartile of culmen_depth_mm is q1= 15.6
 Second quartile of culmen_depth_mm is q3= 18.7
 IQR OF culmen_depth_mm is 3.0999999999999996
 Upper limit of culmen_depth_mm is: 23.349999999999998
 Lower limit of culmen_depth_mm is: 10.95

flipper_length_mm

 First quartile of flipper_length_mm is q1= 190.0
 Second quartile of flipper_length_mm is q3= 213.0
 IQR OF flipper_length_mm is 23.0
 Upper limit of flipper_length_mm is: 247.5
 Lower limit of flipper_length_mm is: 155.5

body_mass_g

 First quartile of body_mass_g is q1= 3550.0
 Second quartile of body_mass_g is q3= 4750.0
 IQR OF body_mass_g is 1200.0
 Upper limit of body_mass_g is: 6550.0
 Lower limit of body_mass_g is: 1750.0
```

7. CHECKING THE CORRELATION OF INDEPENDENT VARIABLES WITH TARGET

In [18]:

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

Out[18]:

```
species              1.000000
flipper_length_mm    0.854307
body_mass_g          0.750491
culmen_length_mm     0.731369
sex                  0.002262
island              -0.635659
culmen_depth_mm     -0.744076
Name: species, dtype: float64
```

8. CHECKING FOR CATEGORICAL COLUMNS AND PERFORMING ENCODING

In [13]:

```
x=df.dtypes
x[1]
for i in range(len(x)):
    if x[i]=='O':
        print(x.index[i])
```

```
species
island
sex
```

In [14]:

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

Out[14]:

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

In [15]:

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

Out[15]:

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

In [16]:

```
df['sex']=df['sex'].replace(".","MALE")
df.sex.value_counts()
```

Out[16]:

```
MALE      169
FEMALE    165
Name: sex, dtype: int64
```

In [17]:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

df.species= le.fit_transform(df.species)
df.island= le.fit_transform(df.island)
df.sex = le.fit_transform(df.sex)
```

In [290]:

```
df.head()
```

Out[290]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
0	0	2	39.10000	18.70000	181.000000	3750.000000
1	0	2	39.50000	17.40000	186.000000	3800.000000
2	0	2	40.30000	18.00000	195.000000	3250.000000
3	0	2	43.92193	17.15117	200.915205	4201.754386
4	0	2	36.70000	19.30000	193.000000	3450.000000

9. SPLIT THE DATA INTO DEPENDENT AND INDEPENDENT VARIABLES.

In [291]:

```
X=df.drop(columns =['sex'],axis =1) #dependent variables
y=df.sex #independent variables
```

10. SCALING THE DATA

In [292]:

```
from sklearn.preprocessing import MinMaxScaler
scale =MinMaxScaler()
```

In [293]:

```
X_scaled= pd.DataFrame(scale.fit_transform(X),columns =X.columns)
X_scaled.head()
```

Out[293]:

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
0	0.0	1.0	0.254545	0.666667	0.152542	0.291667
1	0.0	1.0	0.269091	0.511905	0.237288	0.305556
2	0.0	1.0	0.298182	0.583333	0.389831	0.152778
3	0.0	1.0	0.429888	0.482282	0.490088	0.417154
4	0.0	1.0	0.167273	0.738095	0.355932	0.208333

11. SPLIT THE DATA INTO TRAINING AND TESTING

In [294]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.2,random_state=4
```

12.CHECK THE TRAINING AND TESTING DATA SHAPE.

In [295]:

```
X_train.shape
```

Out[295]:

```
(275, 6)
```

In [296]:

```
X_test.shape
```

Out[296]:

```
(69, 6)
```

In [297]:

```
y_train.shape
```

Out[297]:

```
(275,)
```

In [298]:

```
y_test.shape
```

Out[298]:

```
(69,)
```

13. BUILDING THE CLASSIFICATION MODEL

LOGISTIC REGRESSION

In [299]:

```
model_1=LogisticRegression()
model_1.fit(X_train,y_train)

y_train_pred=model_1.predict(X_train)
accuracy_train= accuracy_score(y_train, y_train_pred)
accuracy_train
```

Out[299]:

```
0.8690909090909091
```

In [300]:

```
y_test_pred=model_1.predict(X_test)
accuracy_test= accuracy_score(y_test, y_test_pred)
accuracy_test
```

Out[300]:

```
0.855072463768116
```

In [301]:

```
print(confusion_matrix(y_test_pred, y_test))
```

```
[[23  3]
 [ 7 36]]
```

DECISION TREES

In [302]:

```
model_2 = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
model_2.fit(X_train, y_train)
```

Out[302]:

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [303]:

```
y_pred_d=model_2.predict(X_test)
accuracy_d=accuracy_score(y_test,y_pred_d)
accuracy_d
```

Out[303]:

```
0.8695652173913043
```

In [304]:

```
confusion_matrix(y_pred_d,y_test)
```

Out[304]:

```
array([[25,  4],
       [ 5, 35]], dtype=int64)
```

RANDOM FOREST CLASSIFIERS

In [305]:

```
model_3 = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state
model_3.fit(X_train, y_train)
```

Out[305]:

```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=
42)
```

In [306]:

```
y_pred_r=model_3.predict(X_test)
accuracy_r=accuracy_score(y_test,y_pred_d)
accuracy_r
```

Out[306]:

```
0.8695652173913043
```

In [307]:

```
confusion_matrix(y_pred_r,y_test)
```

Out[307]:

```
array([[26,  4],
       [ 4, 35]], dtype=int64)
```

SUPPORT VECTOR MACHINE

In [308]:

```
model_4= svm.SVC(kernel='linear')
model_4.fit(X_train, y_train)
```

Out[308]:

```
SVC(kernel='linear')
```

In [309]:

```
y_pred_s=model_4.predict(X_test)
accuracy_s=accuracy_score(y_pred_s,y_test)
accuracy_s
```

Out[309]:

```
0.8840579710144928
```

OUT OF ALL THE MODELS SVM GIVES BETTER ACCURACY

In [310]:

```
import pickle
pickle.dump(model_4,open('Syam.pkl','wb'))
```

```
C:\Users\HP\AppData\Local\Temp/ipykernel_15880/2735532712.py:2: ResourceWa
rning: unclosed file <_io.BufferedWriter name='Syam.pkl'>
  pickle.dump(model_4,open('Syam.pkl','wb'))
ResourceWarning: Enable tracemalloc to get the object allocation traceback
```