```
import numpy as nmpy
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sbn
```

```
df = pd.read_csv('sample_data/penguins_size.csv')
```

```
df.head()
```

	species	island	culmen_length_mm	culmen_depth_mm	flipper_length_mm	body_mass_g
0	Adelie	Torgersen	39.1	18.7	181.0	3750.0
1	Adelie	Torgersen	39.5	17.4	186.0	3800.0
2	Adelie	Torgersen	40.3	18.0	195.0	3250.0
3	Adelie	Torgersen	NaN	NaN	NaN	NaN
4	Adelie	Torgersen	36.7	19.3	193.0	3450.0

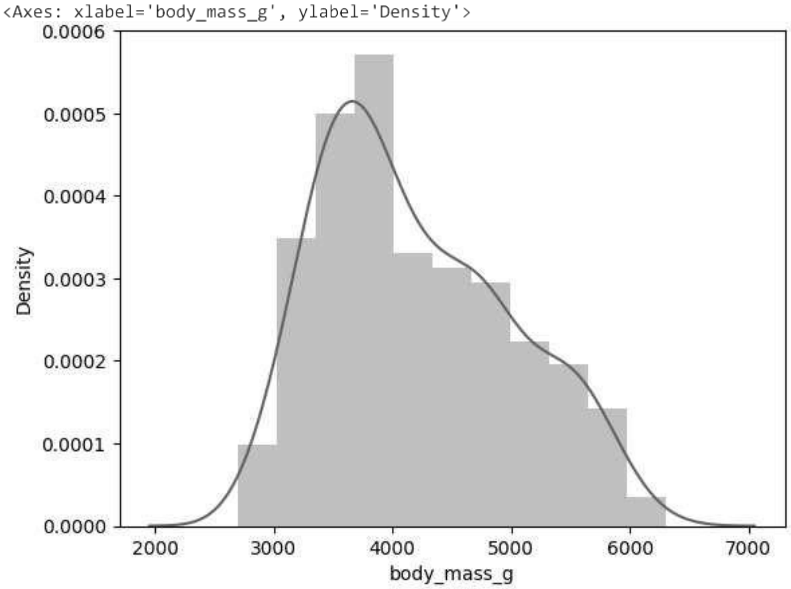
```
df.shape
```

```
    (344, 7)
```

1.Univariate Analysis

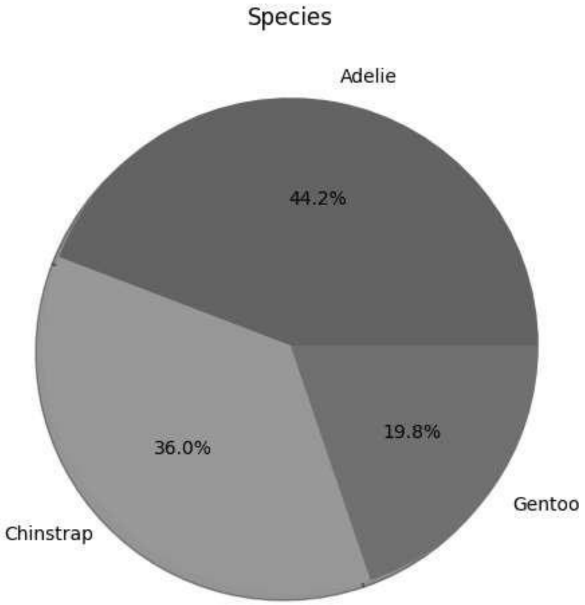
```
sbn.distplot(df["body_mass_g"])
```

```
    <ipython-input-9-9a0d592a49b0>: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

      sbn.distplot(df["body_mass_g"])
    <Axes: xlabel='body_mass_g', ylabel='Density'>
```



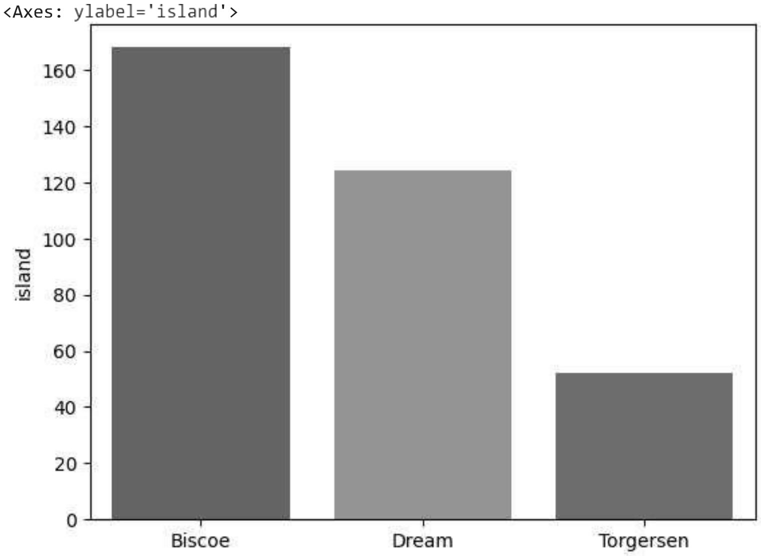
```
plt.figure(figsize=(6,6))
plt.pie(df["species"].value_counts(),labels=df["species"].unique(),autopct='%1.1f%%',shadow = True)
plt.title("Species")
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



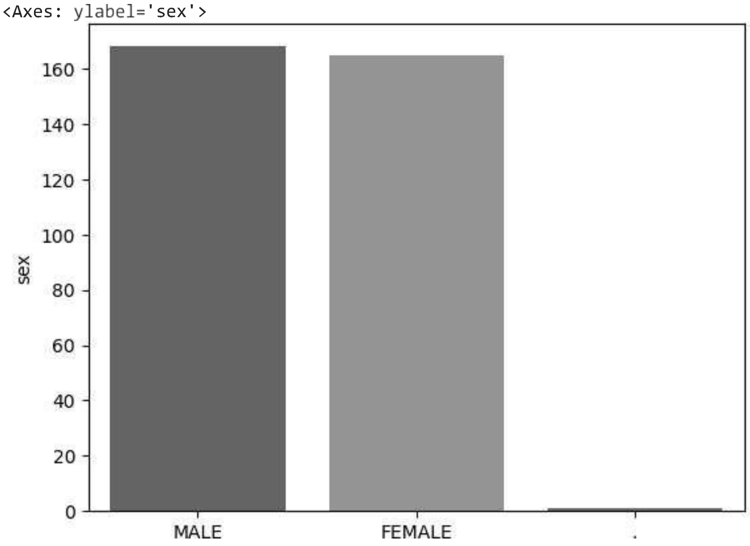
```
sbn.barplot(x=df["island"].value_counts().index,y=df["island"].value_counts())
```

```
<Axes: ylabel='island'>
```



```
sbn.barplot(x=df["sex"].value_counts().index,y=df["sex"].value_counts())
```

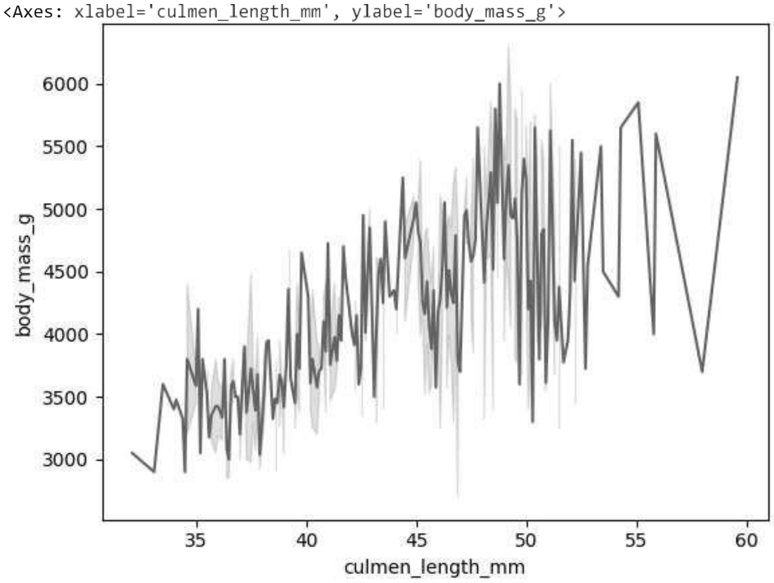
```
<Axes: ylabel='sex'>
```



2.Bivariate Analysis

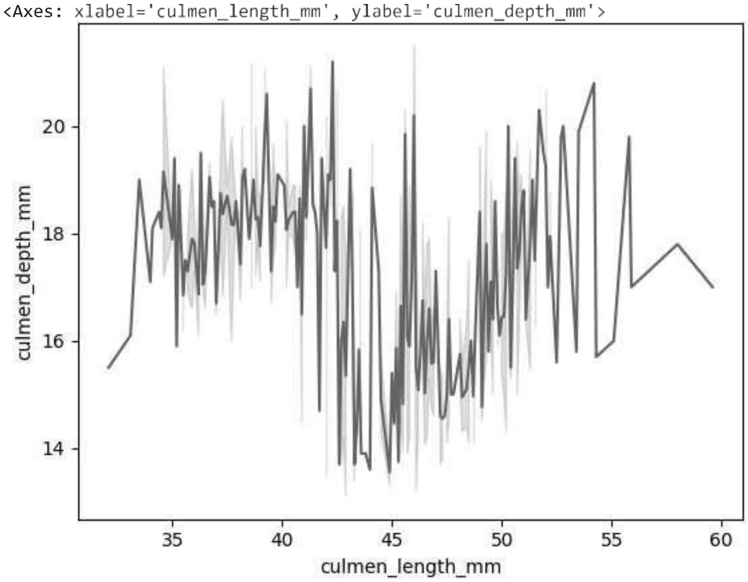
```
sbn.lineplot(x=df["culmen_length_mm"],y=df["body_mass_g"])
```

```
<Axes: xlabel='culmen_length_mm', ylabel='body_mass_g'>
```



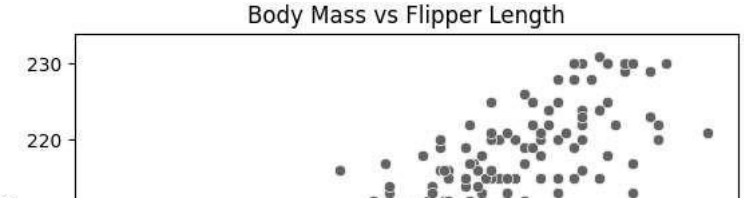
```
sbn.lineplot(x=df['culmen_length_mm'],y=df['culmen_depth_mm'])
```

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



```
plt.title("Body Mass vs Flipper Length")
sbn.scatterplot(x=df["body_mass_g"],y=df['flipper_length_mm'])
```

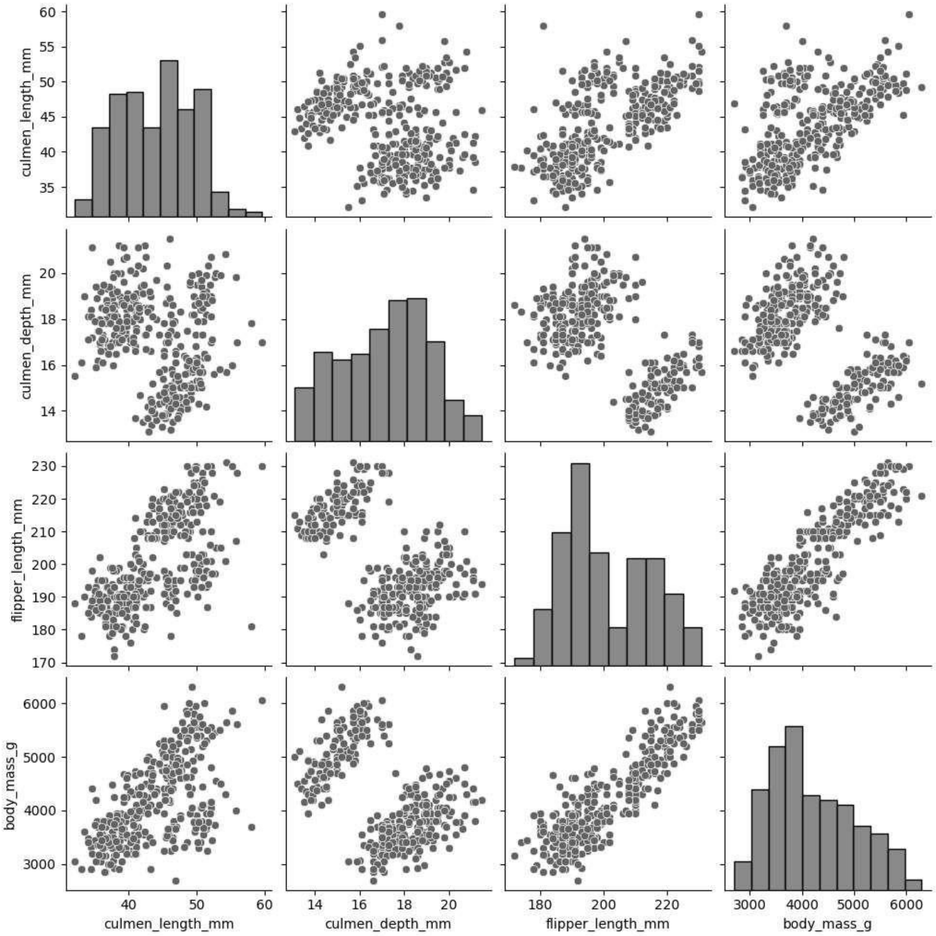
```
<Axes: title={'center': 'Body Mass vs Flipper Length'}, xlabel='body_mass_g',
ylabel='flipper_length_mm'>
```



3.Multivariate Analysis

```
sbn.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7aac8d553880>
```



4.Descriptive Statistics

```
dstats = df.describe()
print(dstats)
```

```
       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

```
#THE MISSING VALUES
missing_values = df.isnull().sum()
print(missing_values)
```

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

Dealing with missing values

```
#Input values using mean for numerical value based
df['culmen_length_mm'].fillna(df['culmen_length_mm'].median(), inplace=True)
df['culmen_depth_mm'].fillna(df['culmen_depth_mm'].median(),inplace=True)
df['flipper_length_mm'].fillna(df['flipper_length_mm'].median(), inplace=True)
df['body_mass_g'].fillna(df['body_mass_g'].median(),inplace=True)
df['sex'].fillna(df['sex'].mode()[0],inplace=True)
```

Checking for missing values again:

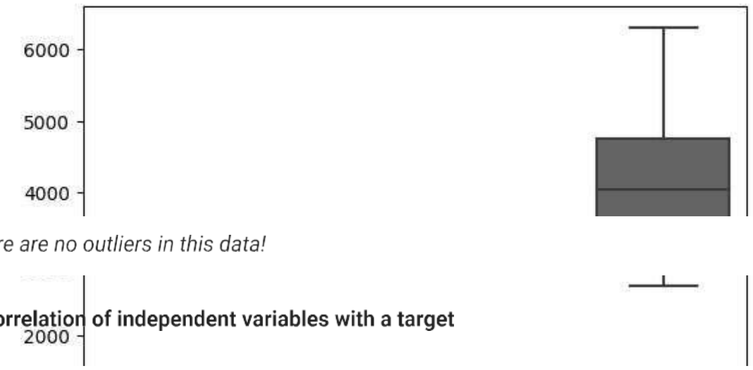
```
#THE MISSING VALUES
missing_values = df.isnull().sum()
print(missing_values)
```

```
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.Outlier Removal

```
#viewing the outliers
sbn.boxplot(data=df[['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']])
```

```
<Axes: >
```



There are no outliers in this data!

7.Correlation of independent variables with a target

```
cmatrix = df[['culmen_length_mm', 'culmen_depth_mm', 'flipper_length_mm', 'body_mass_g']].corr()

# correlation with the target variable 'body_mass_g'
target = cmatrix['body_mass_g']
print(target)
```

```
culmen_length_mm     0.595110
culmen_depth_mm     -0.471916
flipper_length_mm    0.871202
body_mass_g          1.000000
Name: body_mass_g, dtype: float64
```

8.Check for categorical columns and encode them

```
#checking for categorical columns
ccolumn = df.select_dtypes(include=['object']).columns
print(ccolumn)
```

```
Index(['species', 'island', 'sex'], dtype='object')
```

```
#encoding the columns
df_encode = pd.get_dummies(df, columns=ccolumn, drop_first=True)
df_encode.head()
```

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

9.Splitting of data to Independent and Dependent

```
X = df_encode.iloc[:, :-1].values #taking the independent variable species
Y = df_encode.iloc[:, 9] # our dependent variables
print(X)
print(Y)
```

```
[[ 39.1  18.7 181.  ...   0.    1.    0.
                                          ]
 [ 39.5  17.4 186.
                    ...   0.    1.    1. ]
 [ 40.3  18.  195.
                    ...   0.    1.    1. ]
 ...
 [ 50.4  15.7 222.  ...   0.    0.    0.
                                          ]
 [ 45.2  14.8 212.
                    ...   0.    0.    1. ]
 [ 49.9  16.1 213.
0      1            ...   0.    0.    0. ]]
1      0
2      0
3      1
4      0
      ..

339    1
340    0
341    1
342    0
```

```
343    1
Name: sex_MALE, Length: 344, dtype: uint8
```

10.Scaling the Data

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

11.Split the data into training and testing

```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled,  Y, test_size=0.2, random_state=42)
```

12.Check Shape

```
print("Training data shape:", X_train.shape, y_train.shape)
print("Testing data shape:", X_test.shape, y_test.shape)
```

```
Training data shape: (275, 9) (275,)
Testing data shape: (69, 9) (69,)
```

✓ 0s completed at 8:45 PM ● ✕