Part 1 Penguins

Task1

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

# Loading dataset
df = pd.read_csv('penguins.csv')
print("Dataset Loaded")
```

```
Dataset Loaded
```

Task2

```
# Printing the main statistics here
print("\n -------DATASET OVERVIEW-------")
print("Dataset Info:")
print(df.info())
print("\n -------DATASET DESCRIPTION-------")
print(df.describe(include='all'))
print("\n -------MISSING VALUES-------")
print(df.isnull().sum())
```

```
 8   gender                  327 non-null    object
 9   year                    342 non-null    float64
dtypes: float64(5), int64(2), object(3)
memory usage: 27.0+ KB
None

 -------DATASET DESCRIPTION-------
       species  island  calorie requirement  average sleep duration  \
count      333     334           344.000000              344.000000
unique       8       8                  NaN                     NaN
top     Adelie  Biscoe                  NaN                     NaN
freq       145     160                  NaN                     NaN
mean       NaN     NaN          5270.002907               10.447674
std        NaN     NaN          1067.959116                2.265895
min        NaN     NaN          3504.000000                7.000000
25%        NaN     NaN          4403.000000                9.000000
50%        NaN     NaN          5106.500000               10.000000
75%        NaN     NaN          6212.750000               12.000000
max        NaN     NaN          7197.000000               14.000000

        bill_length_mm  bill_depth_mm  flipper_length_mm  body_mass_g gender  \
count       337.000000     333.000000         336.000000   339.000000    327
unique             NaN            NaN                NaN          NaN      4
top                NaN            NaN                NaN          NaN   male
freq               NaN            NaN                NaN          NaN    164
mean         45.494214      18.018318         197.764881  4175.463127    NaN
std          10.815787       9.241384          27.764491   858.713267    NaN
min          32.100000      13.100000          10.000000   882.000000    NaN
25%          39.500000      15.700000         190.000000  3550.000000    NaN
50%          45.100000      17.300000         197.000000  4050.000000    NaN
75%          49.000000      18.700000         213.000000  4750.000000    NaN
max         124.300000     127.260000         231.000000  6300.000000    NaN
```

```
calorie requirement        0
average sleep duration     0
bill_length_mm             7
bill_depth_mm             11
flipper_length_mm          8
body_mass_g                5
gender                    17
year                       2
dtype: int64
```

```
# Displaying distinct values for each column to get better understanding
print("\n ------Distinct Values in Each Column------")
for column in df.columns:
    distinct_values = df[column].unique()
    print(f"\nColumn: {column}")
    print(f"Distinct Values ({len(distinct_values)}): {distinct_values}")
```

```
Column: average sleep duration
Distinct Values (8): [11 14  8 13  9  7 10 12]

Column: bill_length_mm
Distinct Values (173): [ 39.1   39.5   40.3     nan  36.7   39.3   38.9   39.2   34.1   42.
  37.8   41.1   38.6   34.6   36.6   38.7   82.4   34.4   46.    82.47
  35.9   38.2   38.8   35.3   40.6   40.5   37.9   37.2   40.9   36.4
  42.2   37.6   39.8   36.5   40.8   36.    44.1   37.    39.6   37.5
  42.3   40.1   35.    34.5   41.4   39.    35.7   41.3   41.6   35.5
  41.8   33.5   39.7  119.89  45.8   42.8   36.2   42.1   42.9   35.1
  37.3   36.3   36.9   38.3   81.1   38.1   33.1   43.2   41.    37.7
  45.6   42.7   40.2   35.2   41.5   38.5   43.1   36.8   35.6   32.1
  40.7  124.3   46.1   50.    48.7   47.6   46.5   45.4   46.7   43.3
  46.8   49.    45.5   48.4   49.3   49.2   46.2   50.2   46.3   44.5
  47.8   48.2   47.3   45.1   59.6   49.1   42.6   44.4   44.    49.6
  45.3   50.5   43.6   44.9   45.2   46.6   48.5   50.1   45.    43.8
  90.3   50.4   45.7   54.3   49.8   49.5   43.5   50.7   47.7   46.4
  48.6   47.5   51.1   52.5   47.4   50.8   43.4   51.3   52.1   52.2
  49.4   46.9   55.9   47.2   41.7   53.4   48.1   51.5   55.1   48.8
  49.9  116.67  52.7   51.7   47.    52.    45.9   50.3   58.    42.4
  50.6   52.8   54.2   42.5   51.    49.7   53.5   50.9   83.27  51.4
 112.75  51.9   55.8 ]

Column: bill_depth_mm
Distinct Values (84): [ 18.7   17.4   18.      nan  19.3   20.6   17.8   19.6   18.1   20.2
  17.1   17.3   17.6   21.2   21.1   19.    20.7   18.4   21.5   18.3
  19.2   17.2   18.9   18.6   17.9   16.7   17.    89.21  18.5   19.1
  19.7   16.9   18.8   17.7   19.5   17.5   16.6   18.2   16.2   19.4
  16.8   16.1   20.3   20.    16.    16.5  117.23  20.5   15.9   20.1
  15.5   13.2   16.3   15.2   14.5   13.5   14.6   15.3   13.4  127.26
  13.7   15.7   15.1   14.3   15.8   13.1   15.    14.2   14.8   13.6
  13.9   13.3   14.1   14.4   15.4   13.8   14.9   15.6   16.4   14.
  14.7   19.8   19.9   20.8 ]

Column: flipper_length_mm
Distinct Values (61): [181. 186. 195.  nan 193. 190. 180. 182. 191. 198. 185. 197. 184. 194.
 174. 189. 187. 183. 172. 178. 188. 196. 179. 200. 192. 202. 205.  27.
 208. 203. 199. 176. 210. 201. 211. 230. 218. 215. 219. 209. 214. 216.
 213. 217. 221. 222.  23. 220. 207. 225. 224. 231. 229. 223. 212. 228.
  17. 226.  10.  19. 206.]

Column: body_mass_g
Distinct Values (98): [3750. 3800. 3250.  992. 3450. 3650. 3625. 4675. 3475. 4250. 3300. 3700.
 3200. 4400. 4500. 3325. 4200. 3400. 3600. 3950. 3550. 3150. 3900. 4150.
 4650. 3100. 3000. 4600. 3425. 2975. 3500. 4300. 4050. 2900. 2850. 3350.
 4100. 3050. 4450.   nan 4000. 4700. 4350. 3725. 4725. 3075. 2925. 3175.
 4775. 3825. 4275. 4075. 3775. 3875. 3275. 4475. 3975. 5700. 5400. 4550.
 4800. 5200. 5150. 5550. 5850. 6300. 5350. 5050. 5000. 5100. 5650. 5250.
 6050. 4950. 4750. 4900. 5300. 4850. 5800. 6000. 5950. 4625. 5450. 5600.
 4875. 4925. 4975. 5500. 4575. 4375. 5750. 3525.  882. 3575. 3850. 2700.
  908. 3675.]

Column: gender
Distinct Values (5): ['male' 'female' nan 'FEMALE' 'MALE']

Column: year
Distinct Values (4): [2007. 2008. 2009.   nan]
```

Task3

```
# Task 3: Handling missing values
df = df.dropna()
```

Task4

```
# Task 4: Handling inconsistent string formatting
df['species'] = df['species'].str.strip().str.capitalize()
df['island'] = df['island'].str.strip().str.capitalize()
df['gender'] = df['gender'].str.strip().str.lower()
print("String Formatting Done")
```

```
String Formatting Done
<ipython-input-5-98e4644e4fd7>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['species'] = df['species'].str.strip().str.capitalize()
<ipython-input-5-98e4644e4fd7>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['island'] = df['island'].str.strip().str.capitalize()
<ipython-input-5-98e4644e4fd7>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df['gender'] = df['gender'].str.strip().str.lower()
```

```
# Displaying distinct values for each column to get better understanding
print("\n ------Distinct Values in Each Column After Cleaning------")
for column in df.columns:
    distinct_values = df[column].unique()
    print(f"\nColumn: {column}")
    print(f"Distinct Values ({len(distinct_values)}): {distinct_values}")
```

```
 3620 6922 5454 4746 4980 3816 4579 6533 6385 3574 7197 3802 4261 3922
 3690 5535 7008 5924 3528 4284 6280 6564 5405 7195 6700 4037 4111 7049
 4705 6098]

Column: average sleep duration
Distinct Values (8): [11 14  8 13  9 12 10  7]

Column: bill_length_mm
Distinct Values (162): [ 39.1   39.5   40.3   36.7   39.3   38.9   39.2   41.1   38.6   34.6
  36.6   38.7   34.4   46.    37.8   35.9   38.2   38.8   35.3   40.6
  40.5   37.9   37.2   40.9   36.4   42.2   37.6   39.8   36.5   40.8
  36.    44.1   37.    39.6   42.3   40.1   35.    42.    34.5   41.4
  39.    35.7   41.3   41.6   35.5   41.8   33.5   39.7   45.8   42.8
  36.2   42.1   42.9   37.3   36.3   36.9   38.3   38.1   33.1   43.2
  41.    37.7   45.6   42.7   40.2   35.2   41.5   38.5   43.1   36.8
  37.5   35.6   32.1   40.7   46.1   50.    47.6   46.5   45.4   46.7
  43.3   46.8   49.    45.5   48.4   49.3   49.2   46.2   50.2   46.3
  47.8   48.2   47.3   45.1   59.6   49.1   42.6   44.4   48.7   45.3
```

```
 4700. 4350. 3725. 4725. 3075. 2925. 3175. 4775. 3825. 4275. 4075. 3775.
 3875. 3275. 4475. 3975. 3475. 4500. 5700. 5400. 4550. 4800. 5200. 5150.
 5550. 5850. 6300. 5050. 5000. 5100. 5650. 5250. 6050. 4950. 5350. 4750.
 4900. 5300. 4850. 5800. 6000. 5950. 4625. 5450. 5600. 4875. 4925. 4975.
 5500. 4575. 4375. 3575. 2700. 3675. 3525.]

Column: gender
Distinct Values (2): ['male' 'female']

Column: year
Distinct Values (3): [2007. 2008. 2009.]
```

Task5

```
# Task 5: Handling outliers using Z-score
def remove_outliers_zscore(df, columns, threshold=3):
    z_scores = stats.zscore(df[columns])
    abs_z_scores = np.abs(z_scores)
    filtered_entries = (abs_z_scores < threshold).all(axis=1)
    return df[filtered_entries]

numeric_columns = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g', 'calorie requirement', 'avera
df = remove_outliers_zscore(df, numeric_columns)
print("Outliers Removed Using Z-Score")
```

```
Outliers Removed Using Z-Score
```

```
df
```

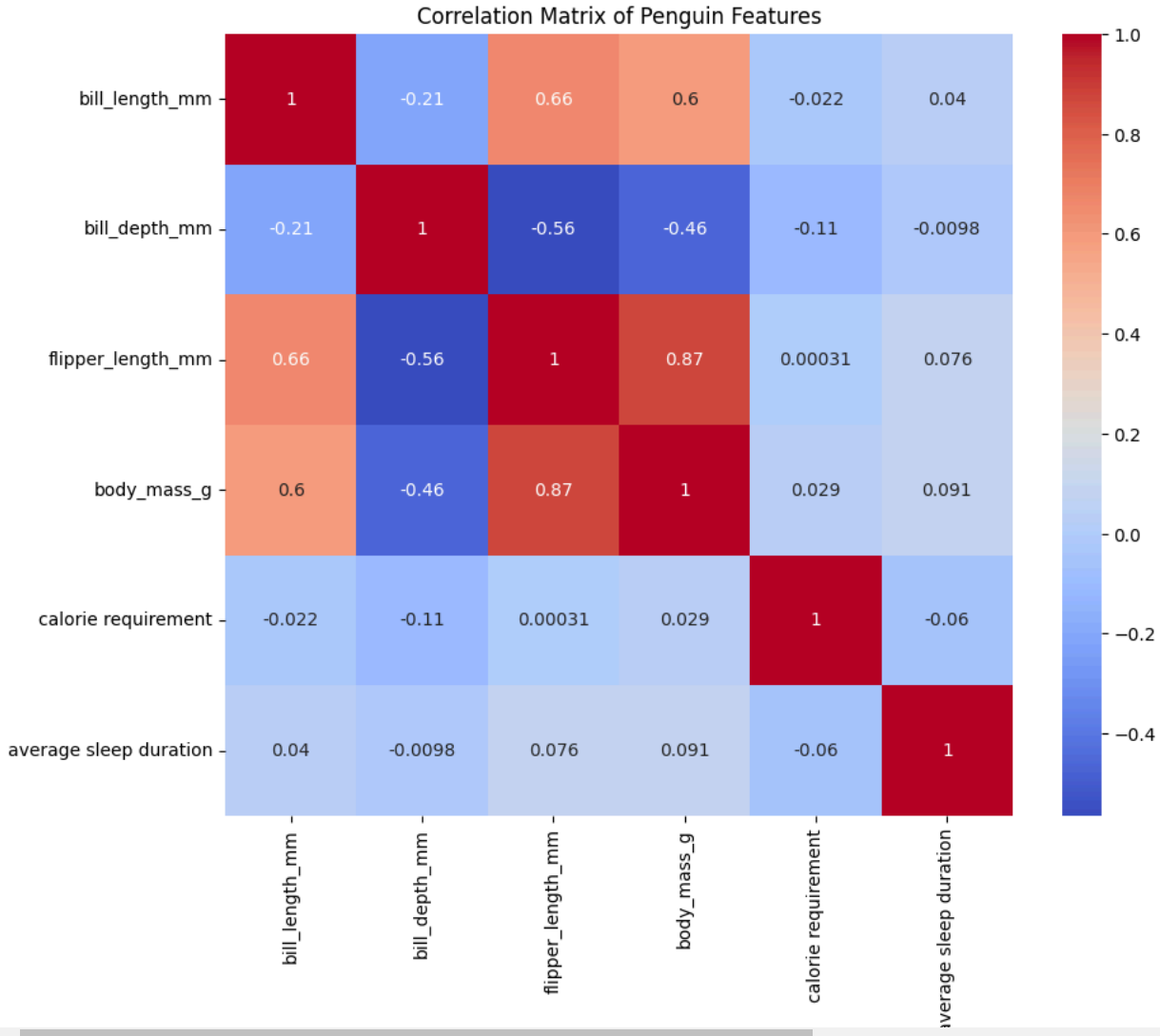
	species	island	calorie requirement	average sleep duration	bill_length_mm	bill_depth_mm	flipper_length_mm	body_mass_g	gender	year
0	Adelie	Torgersen	6563	11	39.1	18.7	181.0	3750.0	male	2007.0
1	Adelie	Torgersen	4890	14	39.5	17.4	186.0	3800.0	female	2007.0
2	Adelie	Torgersen	7184	11	40.3	18.0	195.0	3250.0	female	2007.0
4	Adelie	Torgersen	4774	8	36.7	19.3	193.0	3450.0	female	2007.0
5	Adelie	Torgersen	4403	13	39.3	20.6	190.0	3650.0	male	2007.0
...	...	...	...	...	...	...	...	...	...	...
339	Chinstrap	Dream	4826	11	55.8	19.8	207.0	4000.0	male	2009.0
340	Chinstrap	Dream	4111	9	43.5	18.1	202.0	3400.0	female	2009.0
341	Chinstrap	Dream	7049	10	49.6	18.2	193.0	3775.0	male	2009.0
342	Chinstrap	Dream	4705	7	50.8	19.0	210.0	4100.0	male	2009.0

Next steps: Generate code with df | View recommended plots | New interactive sheet

Task6

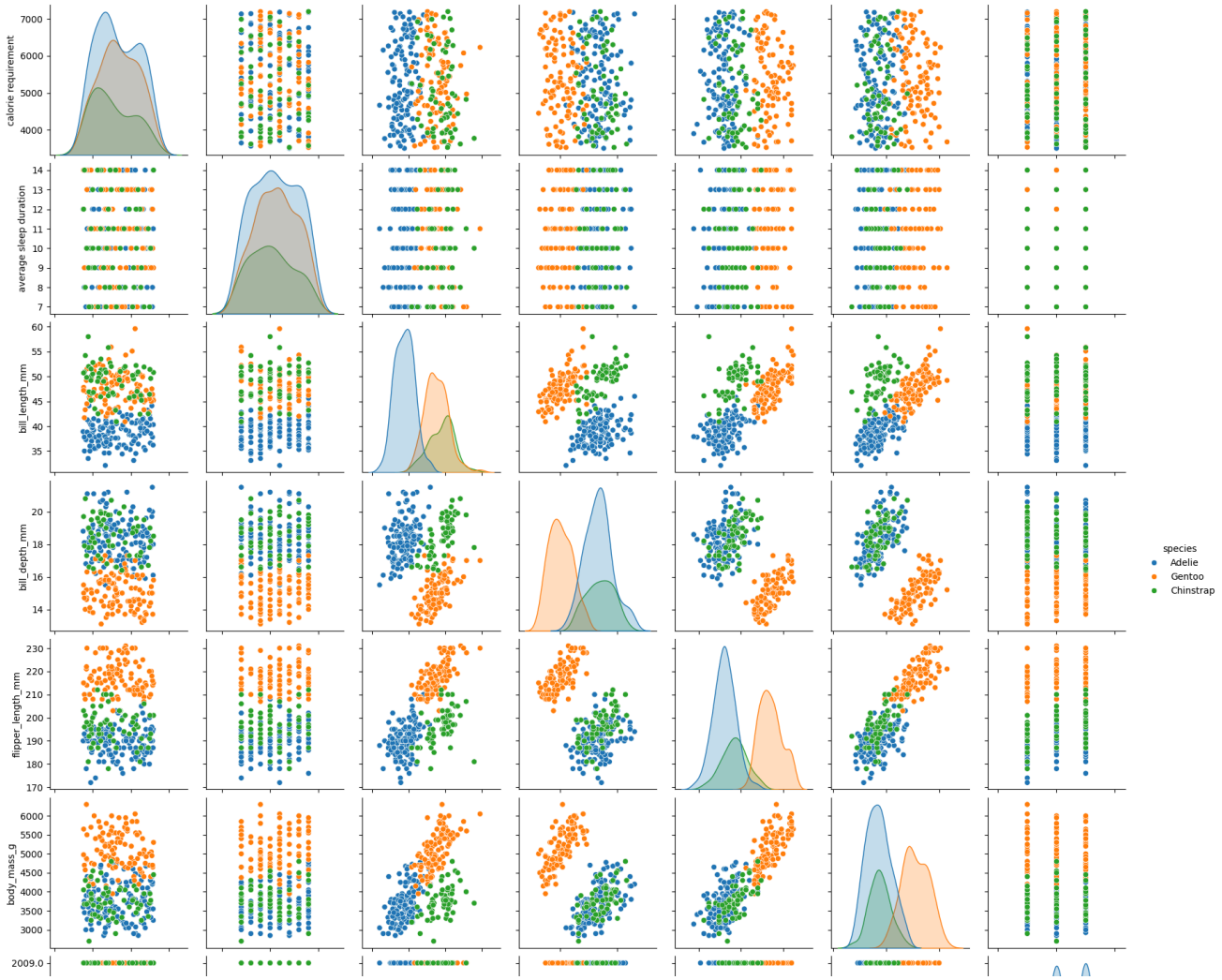
```
#Data Visualization here - Correlation Matrix
print("\n ------Visualizing Correlation Matrix------")
plt.figure(figsize=(10, 8))
sns.heatmap(df[numeric_columns].corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix of Penguin Features')
plt.show()
plt.close()
```

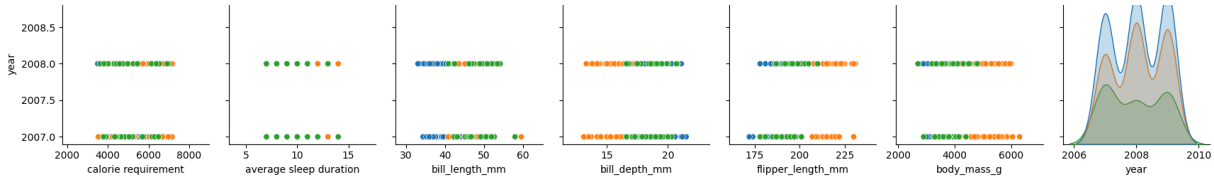
```
------Visualizing Correlation Matrix------
```




```
# Using the pairplot to visualize the relationship between the species
print("\n ------Visualizing Pair Relationships------\n")
sns.pairplot(df, hue='species')
plt.show()
plt.close()
```

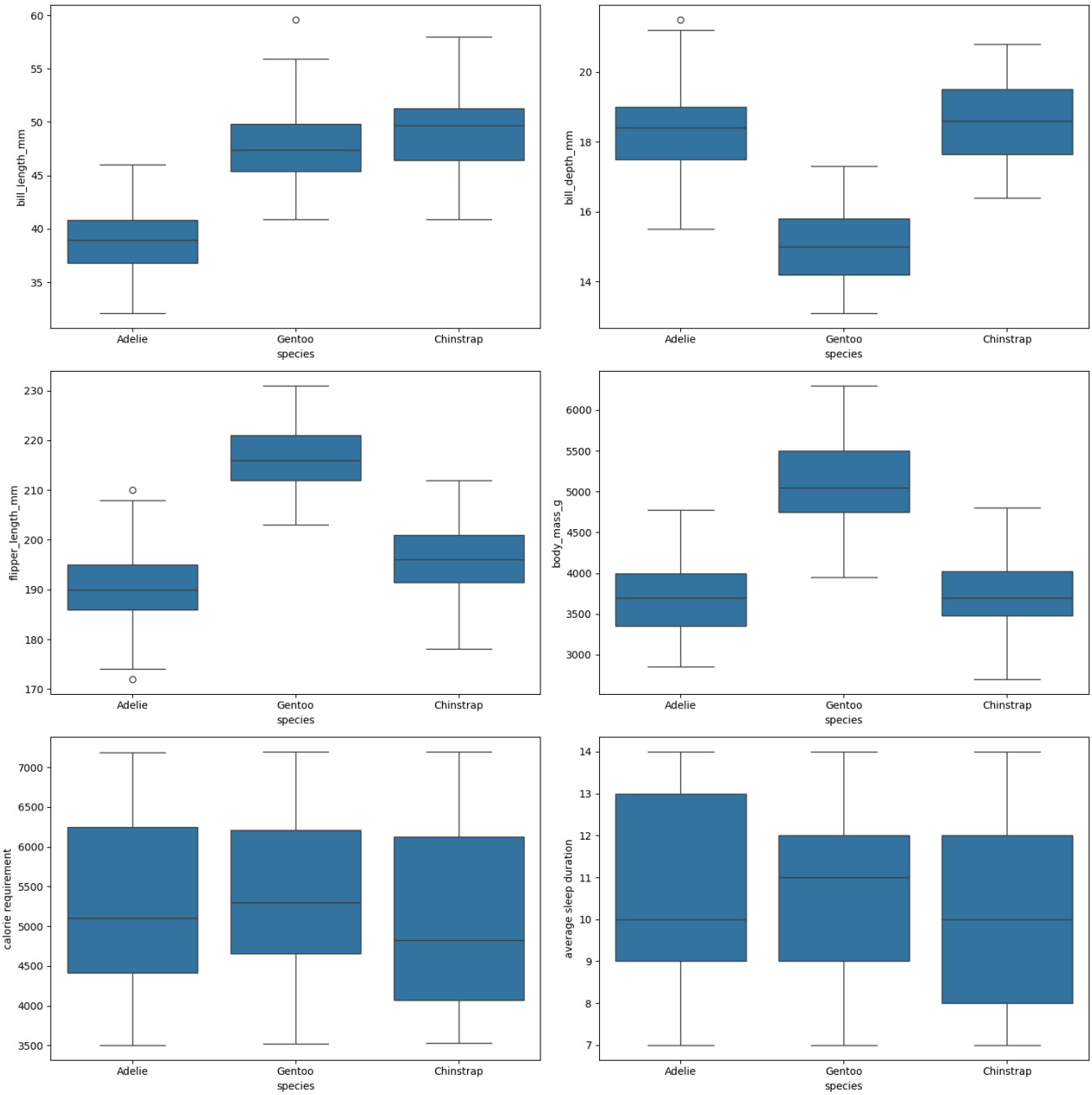
```
------Visualizing Pair Relationships------

/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will n
  data_subset = grouped_data.get_group(pd_key)
```





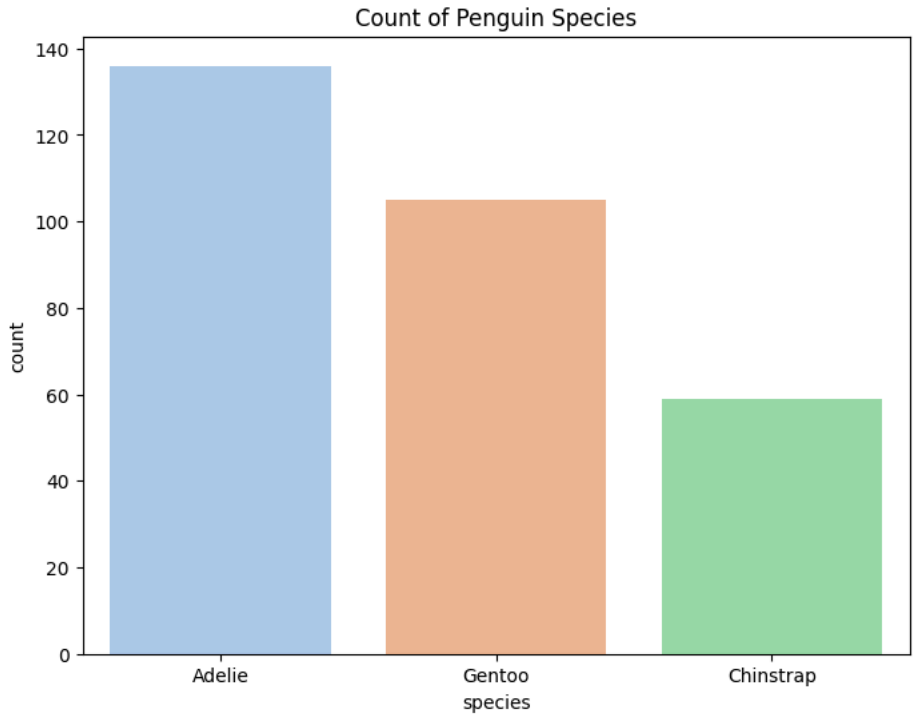

```
# Using Box plots to Visualize the Distributions
print("\n ------Box Plots of Features------\n")
fig, axes = plt.subplots(3, 2, figsize=(15, 15))
sns.boxplot(x='species', y='bill_length_mm', data=df, ax=axes[0, 0])
sns.boxplot(x='species', y='bill_depth_mm', data=df, ax=axes[0, 1])
sns.boxplot(x='species', y='flipper_length_mm', data=df, ax=axes[1, 0])
sns.boxplot(x='species', y='body_mass_g', data=df, ax=axes[1, 1])
sns.boxplot(x='species', y='calorie requirement', data=df, ax=axes[2, 0])
sns.boxplot(x='species', y='average sleep duration', data=df, ax=axes[2, 1])
plt.tight_layout()
plt.show()
plt.close()
```

```
------Box Plots of Features------

/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:640: FutureWarning: SeriesGroupBy.grouper is deprecated and will
  positions = grouped.grouper.result_index.to_numpy(dtype=float)
```



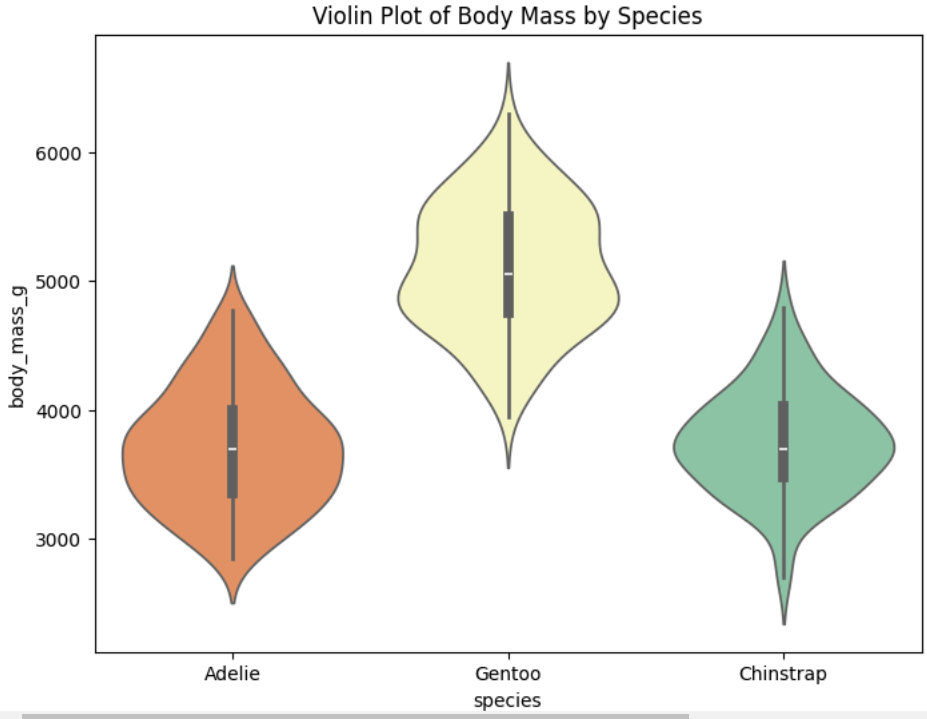
```
#using countplot to plot the count of species
print("\n ------Count Plot of Species------\n")
plt.figure(figsize=(8, 6))
sns.countplot(x='species', data=df, palette='pastel')
plt.title('Count of Penguin Species')
plt.show()
plt.close()
```

```
 ------Count Plot of Species------

<ipython-input-12-2307981c3ee7>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.countplot(x='species', data=df, palette='pastel')
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
/usr/local/lib/python3.10/dist-packages/seaborn/_base.py:949: FutureWarning: When grouping with a length-1 list-like, you will need
  data_subset = grouped_data.get_group(pd_key)
```



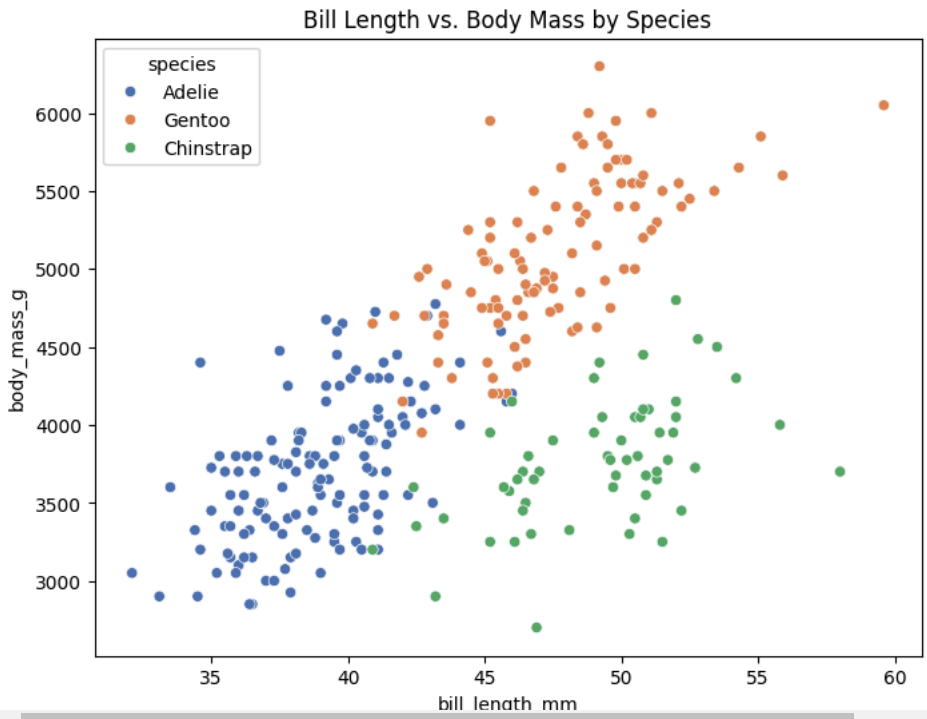

```
#Violin plot for body mass by species
print("\n ------Violin Plot of Body Mass by Species------\n")
plt.figure(figsize=(8, 6))
sns.violinplot(x='species', y='body_mass_g', data=df, palette='Spectral')
plt.title('Violin Plot of Body Mass by Species')
plt.show()
plt.close()
```

```
------Violin Plot of Body Mass by Species------

<ipython-input-13-a3c5ebefefe2>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.violinplot(x='species', y='body_mass_g', data=df, palette='Spectral')
```




```
#scatter plot of bill length vs body mass
print("\n ------Scatter Plot of Bill Length vs. Body Mass------\n")
plt.figure(figsize=(8, 6))
sns.scatterplot(x='bill_length_mm', y='body_mass_g', hue='species', data=df, palette='deep')
plt.title('Bill Length vs. Body Mass by Species')
plt.show()
plt.close()
```

```
------Scatter Plot of Bill Length vs. Body Mass------
```




```
print(df['bill_length_mm'])
```

```
0      39.1
1      39.5
2      40.3
4      36.7
5      39.3
       ...
339    55.8
340    43.5
341    49.6
342    50.8
343    50.2
Name: bill_length_mm, Length: 300, dtype: float64
```

Task7

```
# Encoding the target variable (gender)
df['gender']=df['gender'].str.strip().str.title()
df['gender'] = df['gender'].replace({'Male': 1, 'Female': 0})

# Selecting the numeric columns
numeric_columns = ['bill_length_mm', 'bill_depth_mm', 'flipper_length_mm', 'body_mass_g','calorie requirement','average
features = numeric_columns + ['gender']

# Calculating the correlation matrix
correlation_matrix = df[features].corr()

print("\n------Identifying Uncorrelated Features------")
target_correlation = correlation_matrix['gender']
print("\nCorrelation with Gender:")
print(target_correlation)

# Setting a threshold of 0.1 for low correlation
uncorrelated=target_correlation[target_correlation < 0.1].index.tolist()


print("\nFeatures with lower correlation:")
print(uncorrelated)

# Droping the uncorrelated features
df = df.drop(columns=uncorrelated)
```

```
------Identifying Uncorrelated Features------

Correlation with Gender:
bill_length_mm            0.368319
bill_depth_mm             0.375190
flipper_length_mm         0.269758
body_mass_g               0.436215
calorie requirement      -0.007381
average sleep duration    0.078985
year                      0.015449
gender                    1.000000
Name: gender, dtype: float64

Features with lower correlation:
['calorie requirement', 'average sleep duration', 'year']
<ipython-input-16-db2f37e5b69d>:3: FutureWarning: Downcasting behavior in `replace` is deprecated and will be removed in a future ve
  df['gender'] = df['gender'].replace({'Male': 1, 'Female': 0})
```

Task8

```
def one_hot_encode(df, column):
    unique_values = df[column].unique()
    for value in unique_values:
        df[f"{column}_{value}"] = (df[column] == value).astype(int
    return df.drop(columns=[column])

# Applying one-hot encoding to 'species' and 'island'
df = one_hot_encode(df, 'species')
df = one_hot_encode(df, 'island')

print("\n------One-Hot Encoded Columns------")
```