Name : Gauri V Korgaonkar FDS 02

Aim: To explore the Titanic, Tips, and Penguins datasets using Seaborn and Matplotlib libraries to create basic visualizations and gain insights from the data.

Objectives:

1. Introduce students to the Seaborn and Matplotlib libraries for data visualization.
2. Perform exploratory data analysis (EDA) on the Titanic, Tips, and Penguins datasets.
3. Enable students to create basic visualizations such as bar plots, histograms, box plots, scatter plots, heatmaps, pie charts, timeline charts, and bubble plots.
4. Help students interpret the visualizations to understand relationships, distributions, and patterns in the data.

Datasets:

- **Titanic dataset**: Passenger data including survival, class, gender, and age.
- **Tips dataset**: Data on tips received by waiters, including bill amount, tip amount, and customer attributes.
- **Penguins dataset**: Data on different species of penguins, including flipper length, body mass, and island of habitat.

Steps:

In [1]:
```
import seaborn as sns
```

In [2]:
```
import matplotlib.pyplot as plt
```

In [3]:
```
import pandas as pd
```

In [34]:
```
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [8]:
```
titanic = sns.load_dataset("titanic")
```

In [9]:
```
tips = sns.load_dataset("tips")
```

In [10]:
```
penguins = sns.load_dataset("penguins")
```

In [28]:
```
# Titanic Dataset Overview
print("Titanic Dataset- First 5 rows")
# View first 5 rows
print(titanic.head())
```

```
Titanic Dataset- First 5 rows
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who  adult_male deck  embark_town alive  alone
0    man        True  NaN  Southampton    no  False
1  woman       False    C    Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False    C  Southampton   yes  False
4    man        True  NaN  Southampton    no   True
```

In [26]:
```
# View data types and missing values
print("\nData types and missing values:\n")
print(titanic.info())
```

```
Data types and missing values:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
None
```

In [25]:
```
print("\nMissing Values Summary:\n")
print(titanic.isnull().sum())
```

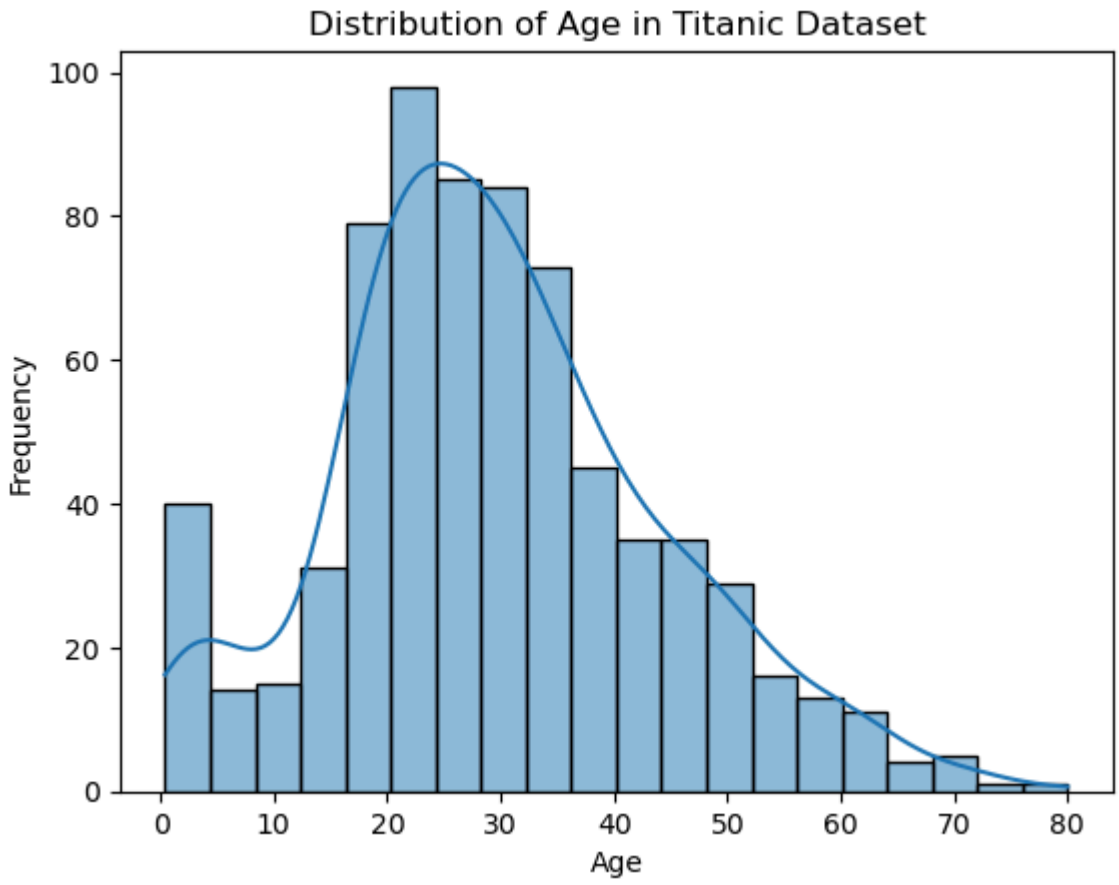
```
Missing Values Summary:

survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64
```

In [29]:
```
# View summary statistics
print("\nSummary stats:\n")
print(titanic.describe())
```

```
Summary stats:

         survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

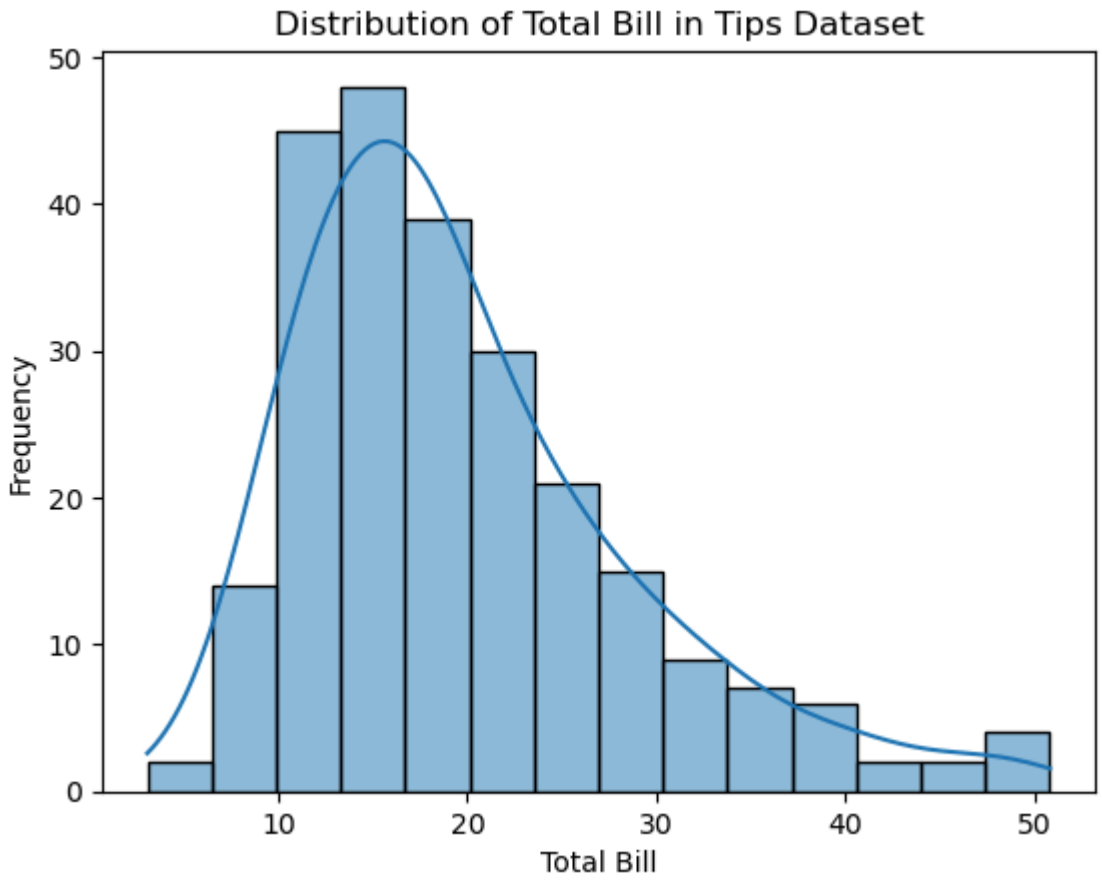
In [35]:
```
print("\nPlot histogram for the 'age' column in the Titanic dataset\n")
sns.histplot(titanic['age'], kde=True )
plt.title('Distribution of Age in Titanic Dataset')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

```
Plot histogram for the 'age' column in the Titanic dataset
```



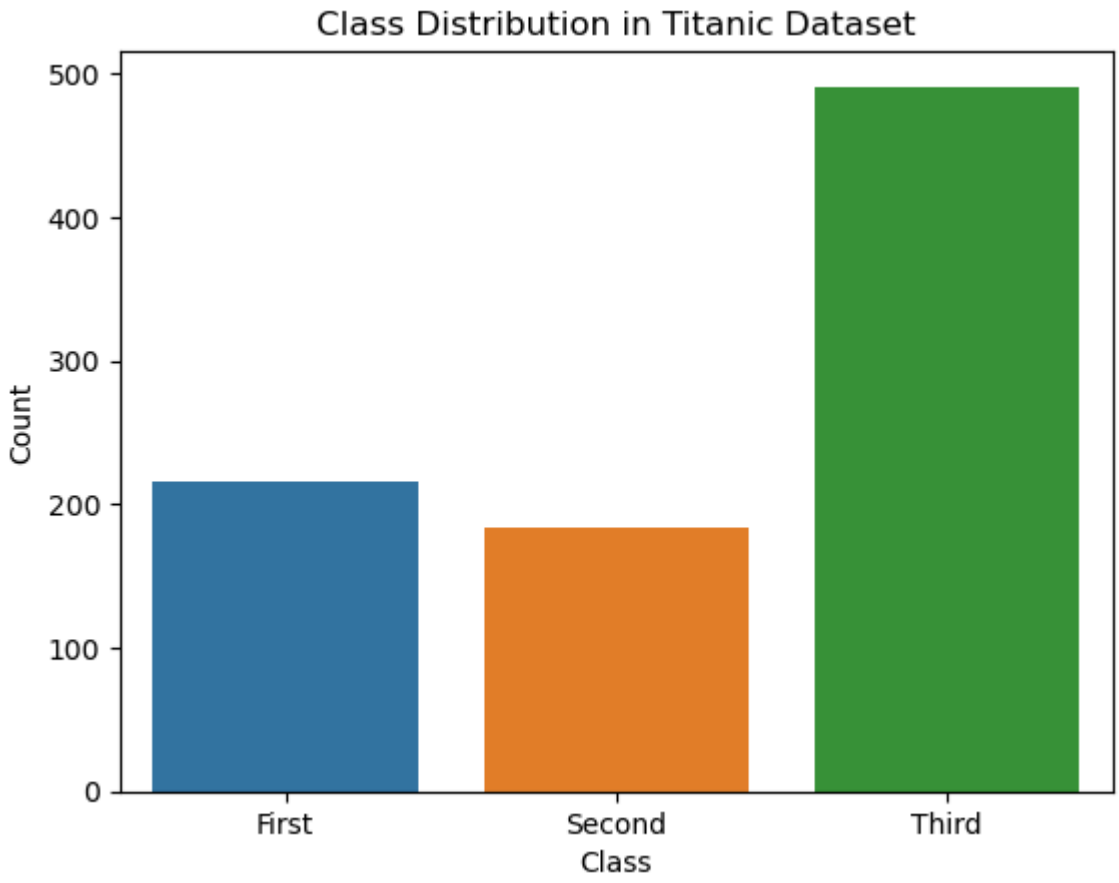
In [36]:
```
print("\nPlot histogram for the 'total_bill' column in the Tips dataset\n")
sns.histplot(tips['total_bill'], kde=True)
plt.title('Distribution of Total Bill in Tips Dataset')
plt.xlabel('Total Bill')
plt.ylabel('Frequency')
plt.show()
```

```
Plot histogram for the 'total_bill' column in the Tips dataset
```



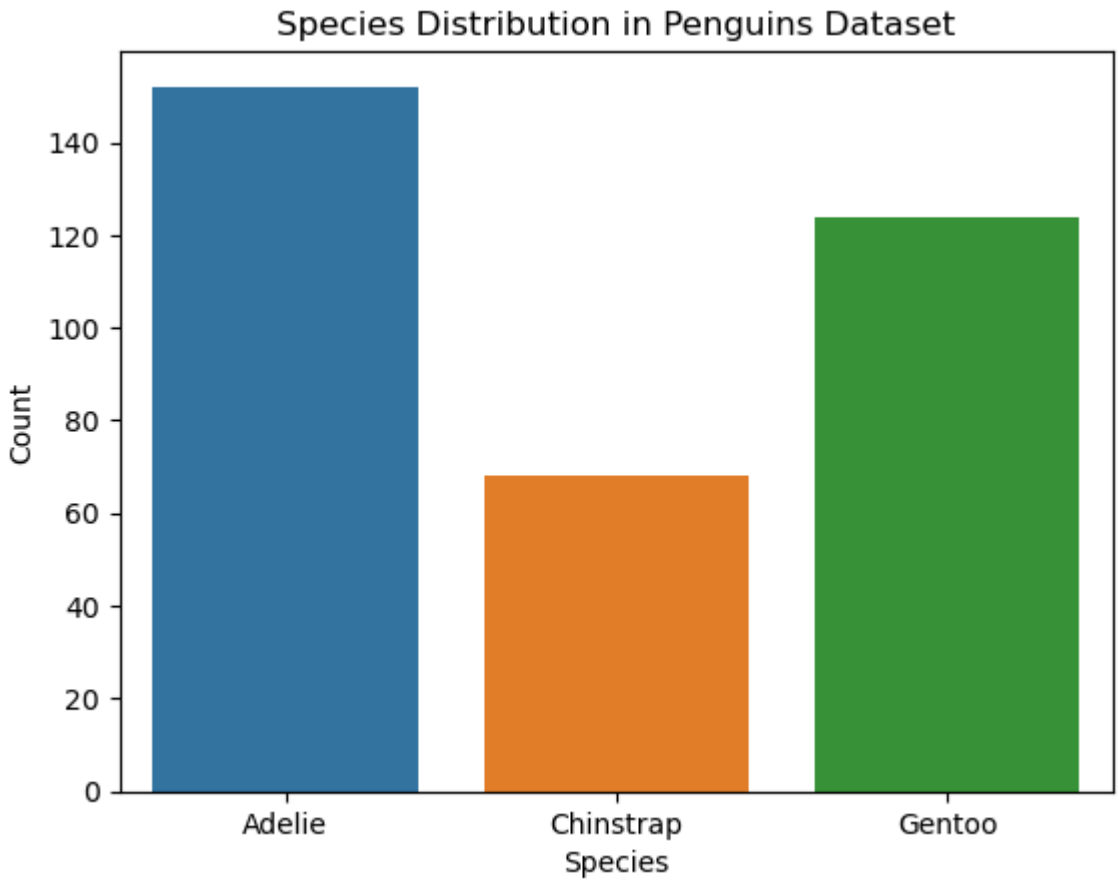
In [37]:
```
print("\nBar plot for the 'class' column in Titanic dataset\n")
sns.countplot(x='class', data=titanic)
plt.title('Class Distribution in Titanic Dataset')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```

```
Bar plot for the 'class' column in Titanic dataset
```



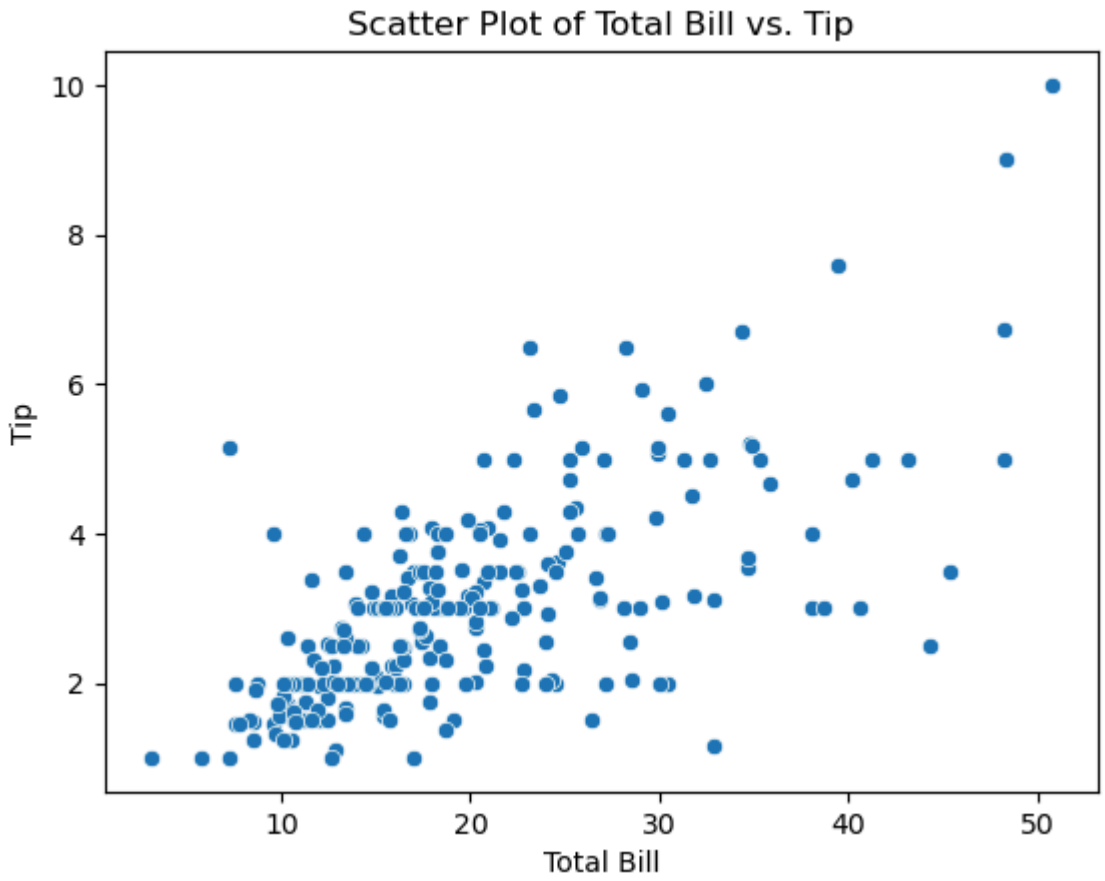
In [38]:
```
print("\nBar plot for the 'species' column in Penguins dataset\n")
sns.countplot(x='species', data=penguins)
plt.title('Species Distribution in Penguins Dataset')
plt.xlabel('Species')
plt.ylabel('Count')
plt.show()
```

```
Bar plot for the 'species' column in Penguins dataset
```



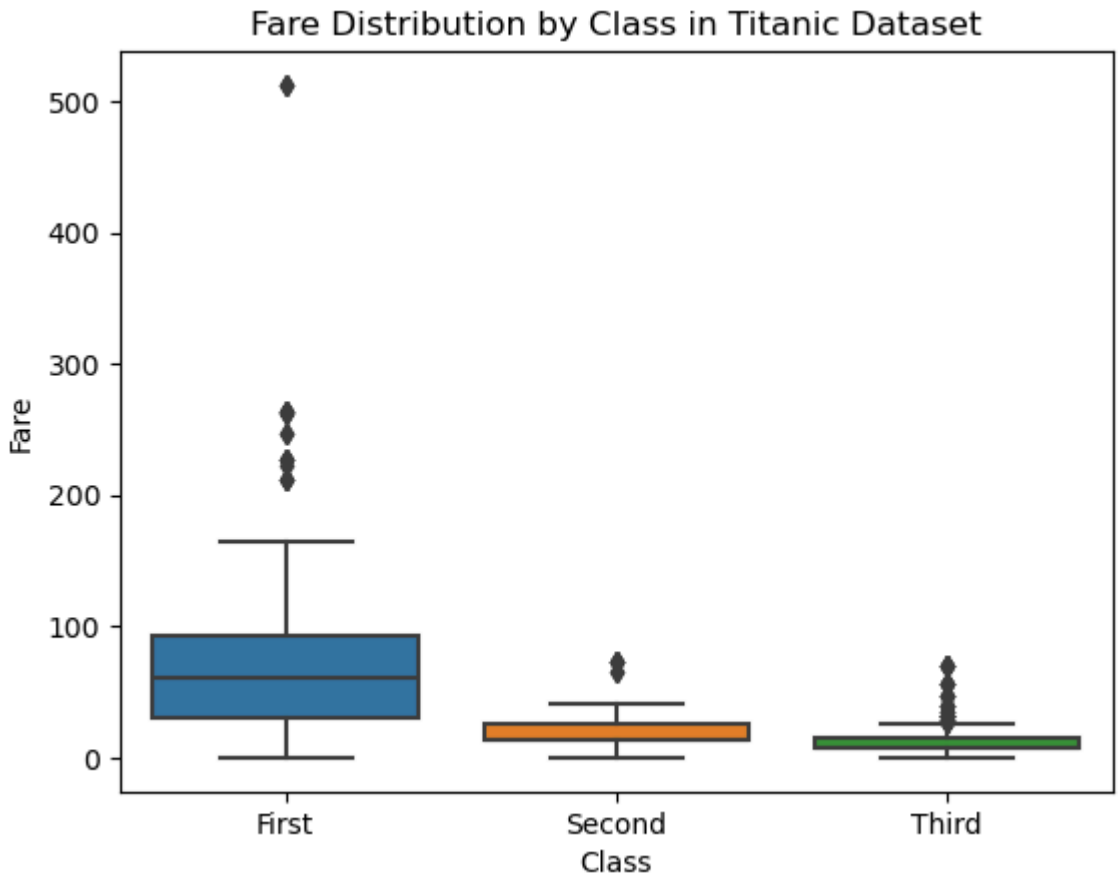
In [39]:
```
print("\nScatter plot for the relationship between total_bill and tip in Tips da
sns.scatterplot(x='total_bill', y='tip', data=tips)
plt.title('Scatter Plot of Total Bill vs. Tip')
plt.xlabel('Total Bill')
plt.ylabel('Tip')
plt.show()
```

```
Scatter plot for the relationship between total_bill and tip in Tips dataset
```



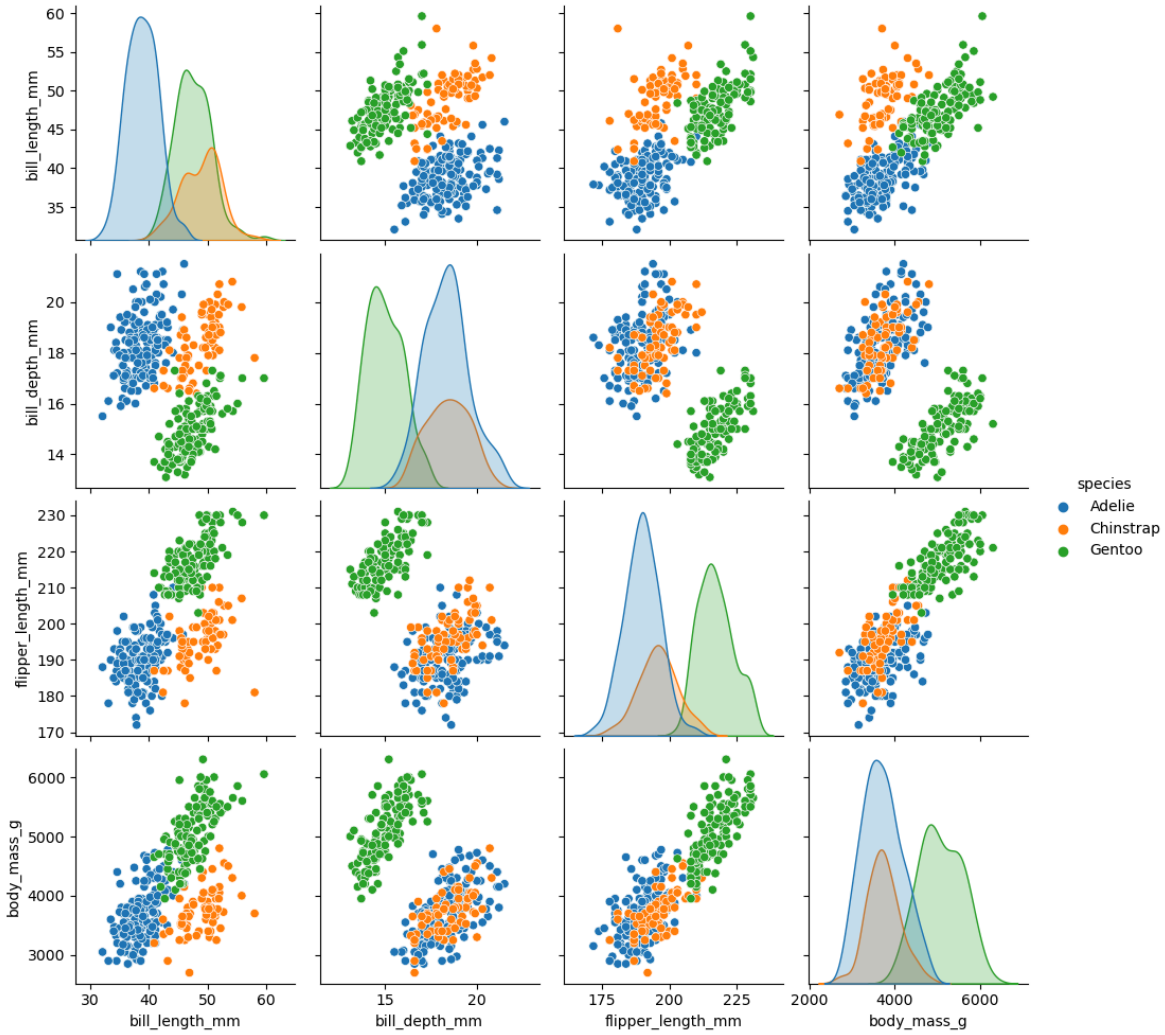
In [40]:
```
print("\nBox plot for fare by class in the Titanic dataset\n")
sns.boxplot(x='class', y='fare', data=titanic)
plt.title('Fare Distribution by Class in Titanic Dataset')
plt.xlabel('Class')
plt.ylabel('Fare')
plt.show()
```

```
Box plot for fare by class in the Titanic dataset
```



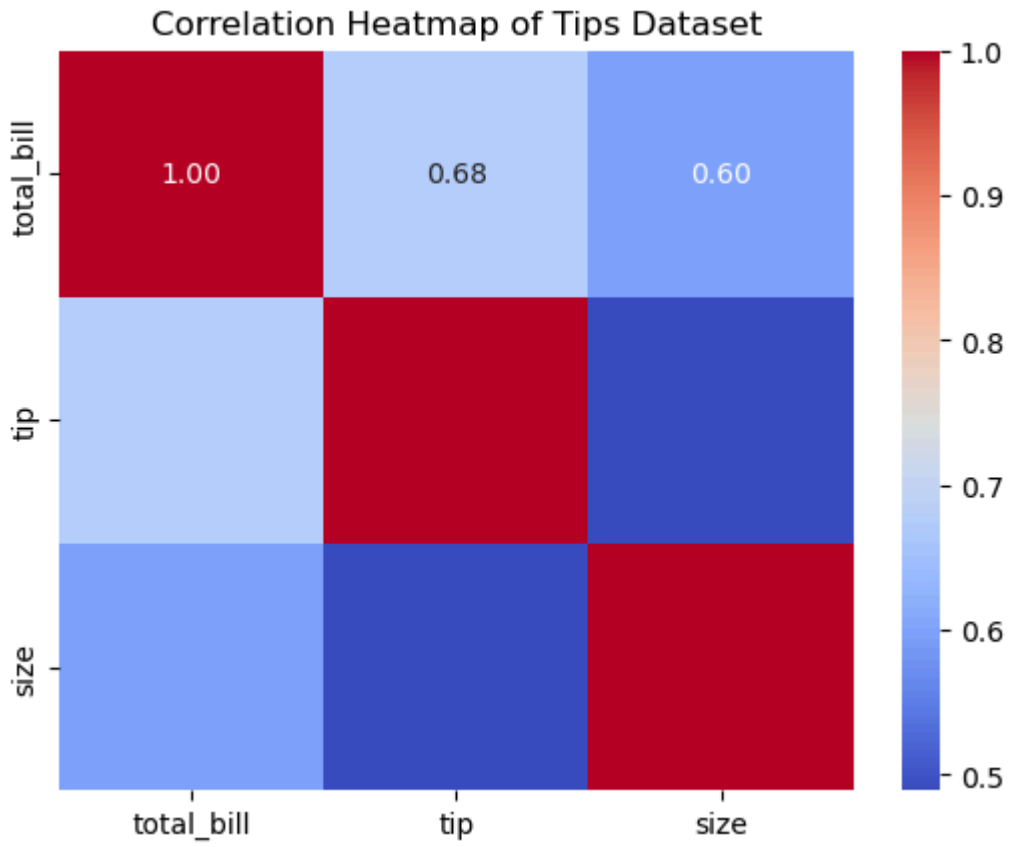
In [41]:
```
print("\nPair plot for the Penguins dataset\n")
sns.pairplot(penguins, hue='species')  # 'hue' adds color coding based on specie
plt.show()
```

```
Pair plot for the Penguins dataset
```



In [50]:
```
# Select only numerical columns for correlation
numerical_columns = tips.select_dtypes(include=['float64', 'int64'])
# Calculate the correlation matrix
correlation_matrix = numerical_columns.corr()
print("\n Heatmap for the correlation matrix\n")
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title('Correlation Heatmap of Tips Dataset')
plt.show()
```

```
Heatmap for the correlation matrix
```

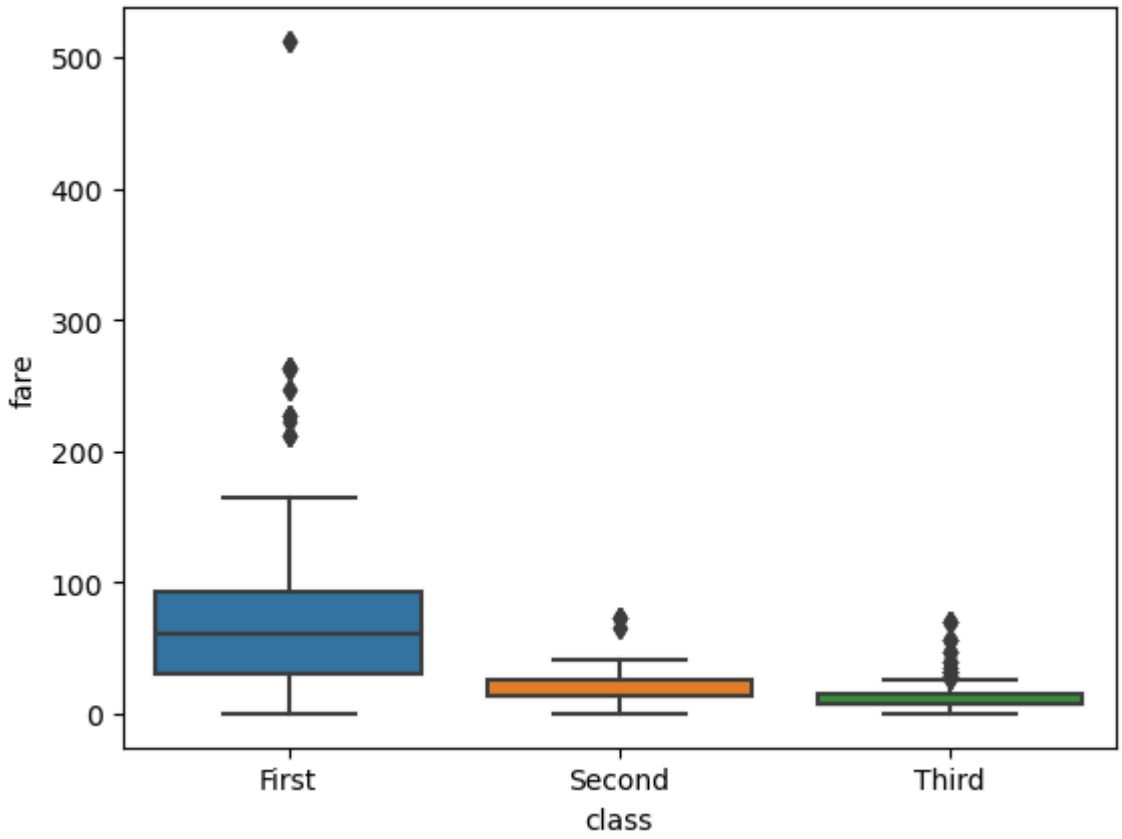


In [ ]:
```
#Tips dataset that cannot be converted to a float when calculating the correlati
#select only the numerical columns before computing the correlation matrix.
```

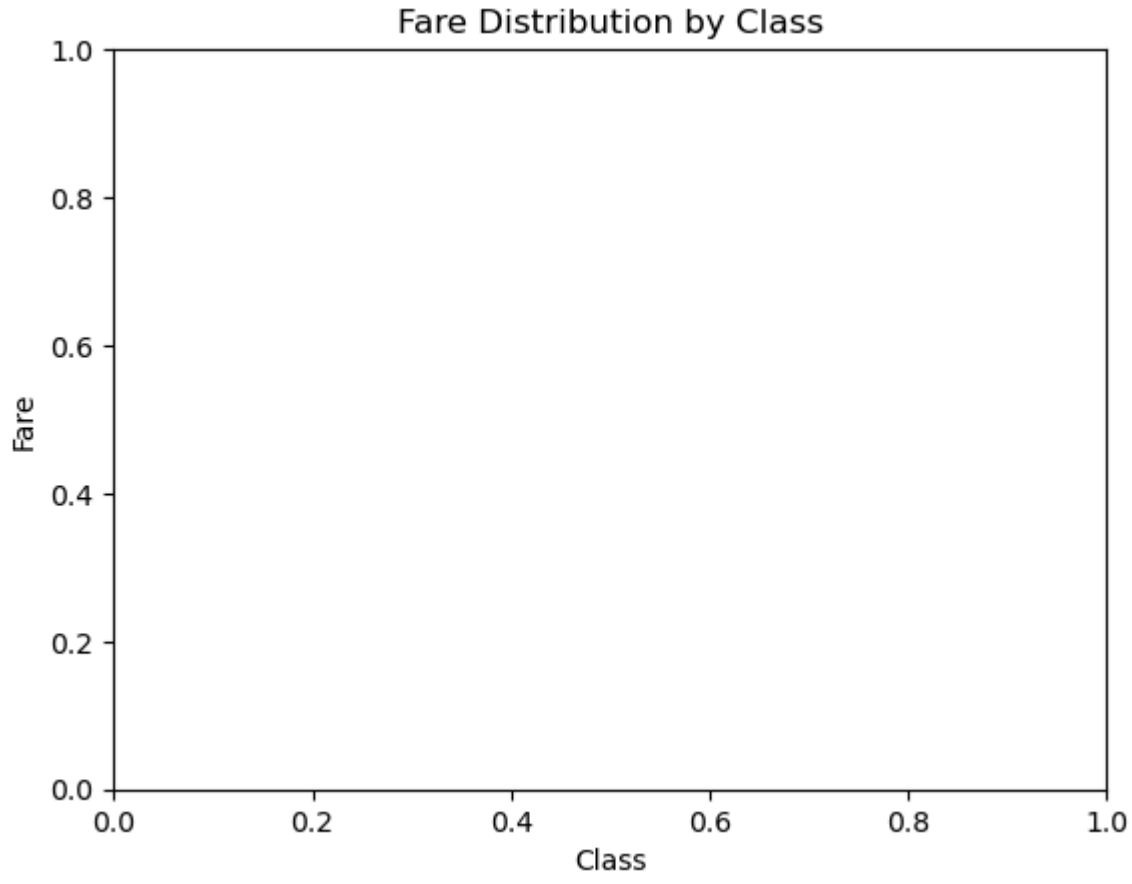
In [51]:
```
print("\n Box plot for fare by class in the Titanic dataset\n")
sns.boxplot(x='class', y='fare', data=titanic)
```

```
Box plot for fare by class in the Titanic dataset
```

Out[51]:
```
<Axes: xlabel='class', ylabel='fare'>
```

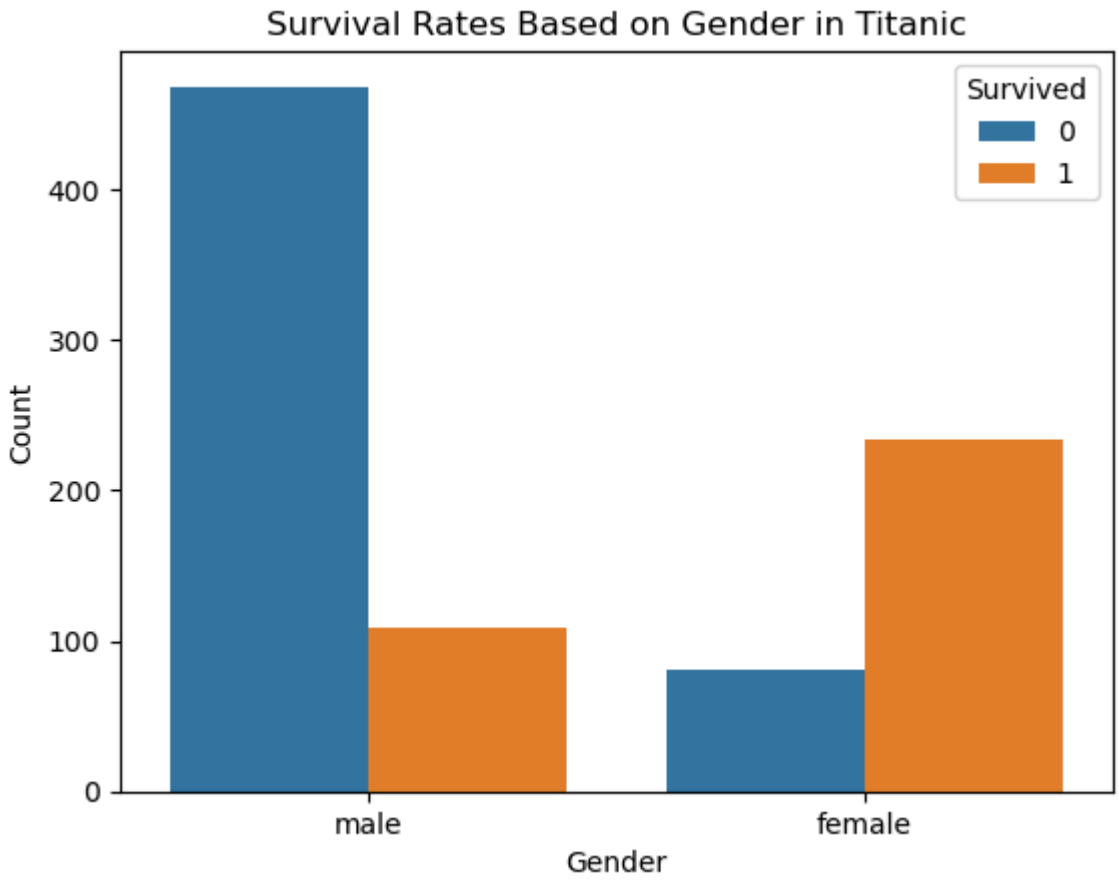


In [52]:
```
# Customizing the plot
plt.title('Fare Distribution by Class')
plt.xlabel('Class')
plt.ylabel('Fare')
plt.show()
```



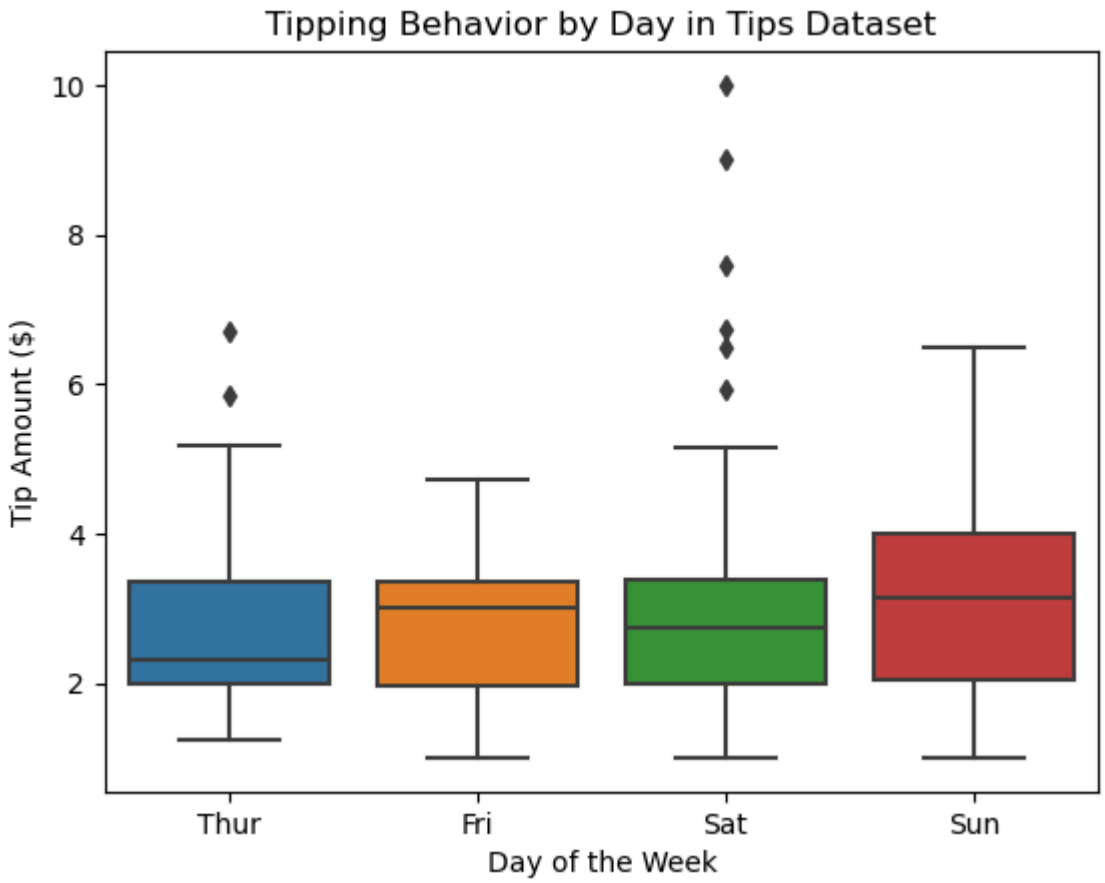
In [74]:
```
print("\nTitanic Dataset: Survival Rates Based on Gender\n")
#Visualization: A bar plot showing the count of survivors by gender.


# Print data types and check the first few rows of the DataFrame
print(titanic.dtypes)

# Convert 'survived' to a categorical type explicitly
titanic['survived'] = titanic['survived'].astype(str)

print(titanic['survived'].dtype)  # Check the data type

# Create the count plot with hue as string
sns.countplot(x='sex', hue='survived', data=titanic)
plt.title('Survival Rates Based on Gender in Titanic')
plt.xlabel('Gender')
plt.ylabel('Count')
plt.legend(title='Survived', labels=['0', '1'])  # Update legend labels
plt.show()
```

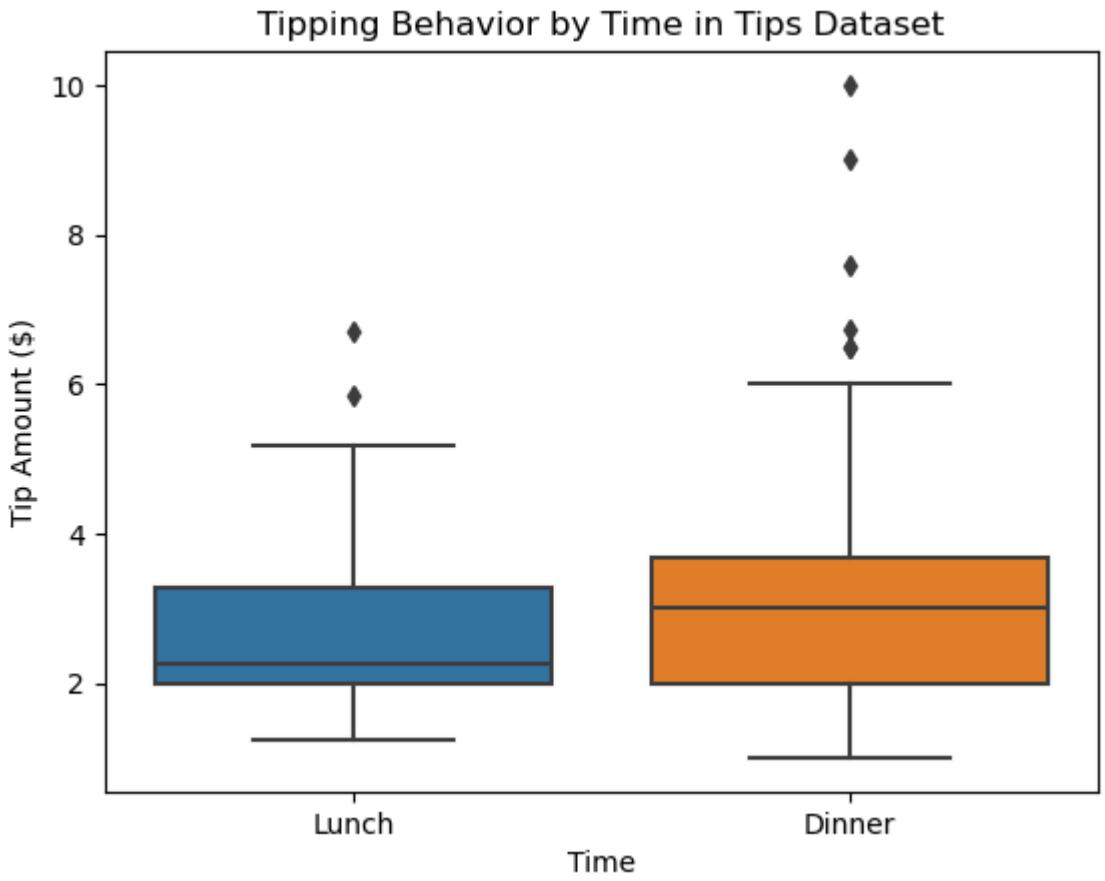
```
Titanic Dataset: Survival Rates Based on Gender

survived         object
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
object
```



In [77]:
```
print("\nTips Dataset: Tipping Behavior Based on Day or Time\n")
#Visualization: A box plot or bar plot showing tips by day or time.

# Box plot for tips by day
sns.boxplot(x='day', y='tip', data=tips)
plt.title('Tipping Behavior by Day in Tips Dataset')
plt.xlabel('Day of the Week')
plt.ylabel('Tip Amount ($)')
plt.show()

# Alternatively, for time of day (if available)
sns.boxplot(x='time', y='tip', data=tips)
plt.title('Tipping Behavior by Time in Tips Dataset')
plt.xlabel('Time')
plt.ylabel('Tip Amount ($)')
plt.show()
```

```
Tips Dataset: Tipping Behavior Based on Day or Time
```

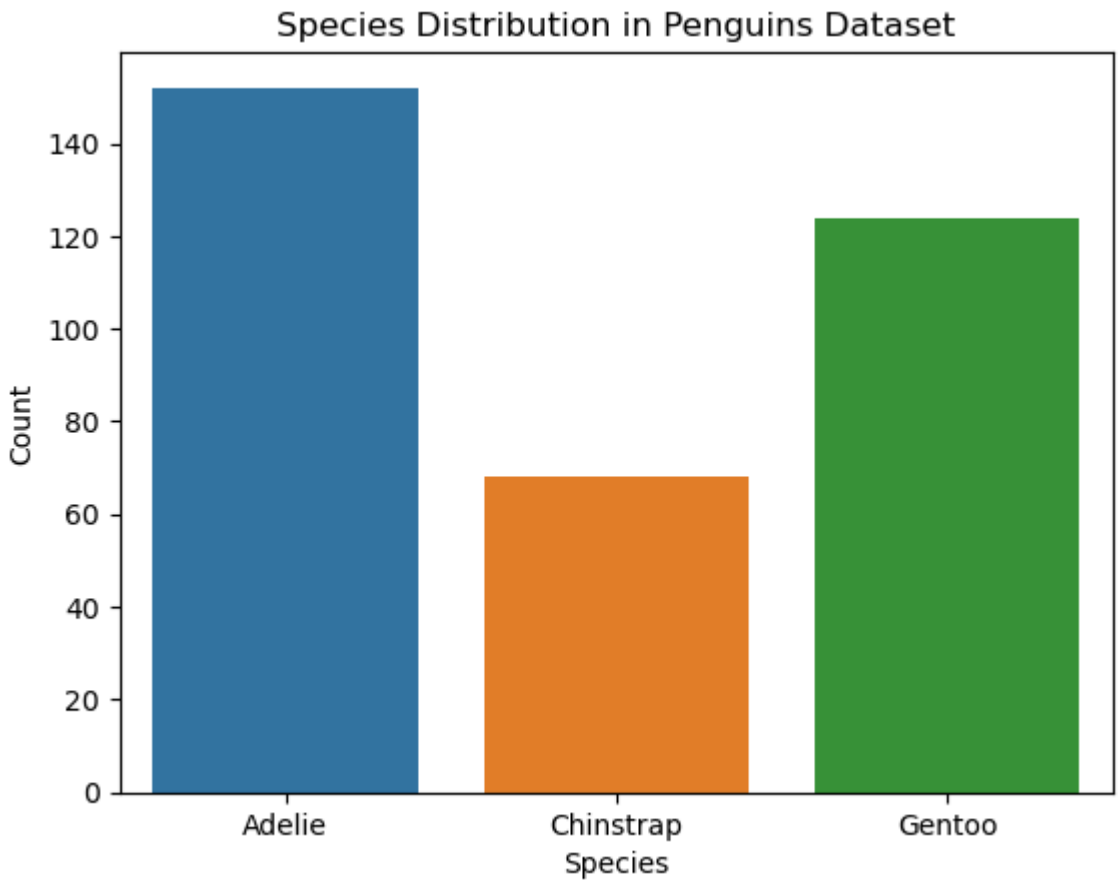




In [78]:
```
print("\nPenguins Dataset: Species Distribution\n")
#Visualization: A bar plot showing the count of penguins by species.

sns.countplot(x='species', data=penguins)
plt.title('Species Distribution in Penguins Dataset')
```

```
plt.xlabel('Species')
plt.ylabel('Count')
plt.show()
```

```
Penguins Dataset: Species Distribution
```



Conclusion

By performing **exploratory data analysis** using Seaborn and Matplotlib on the Titanic, Tips, and Penguins datasets, I learnee to develop skills in **creating and interpreting basic visualizations**. This helped me uncover patterns, relationships, and distributions in real-world data, laying a foundation for **deeper analysis and decision-making** based on visual insights.