shadowfox-beginner

August 30, 2025

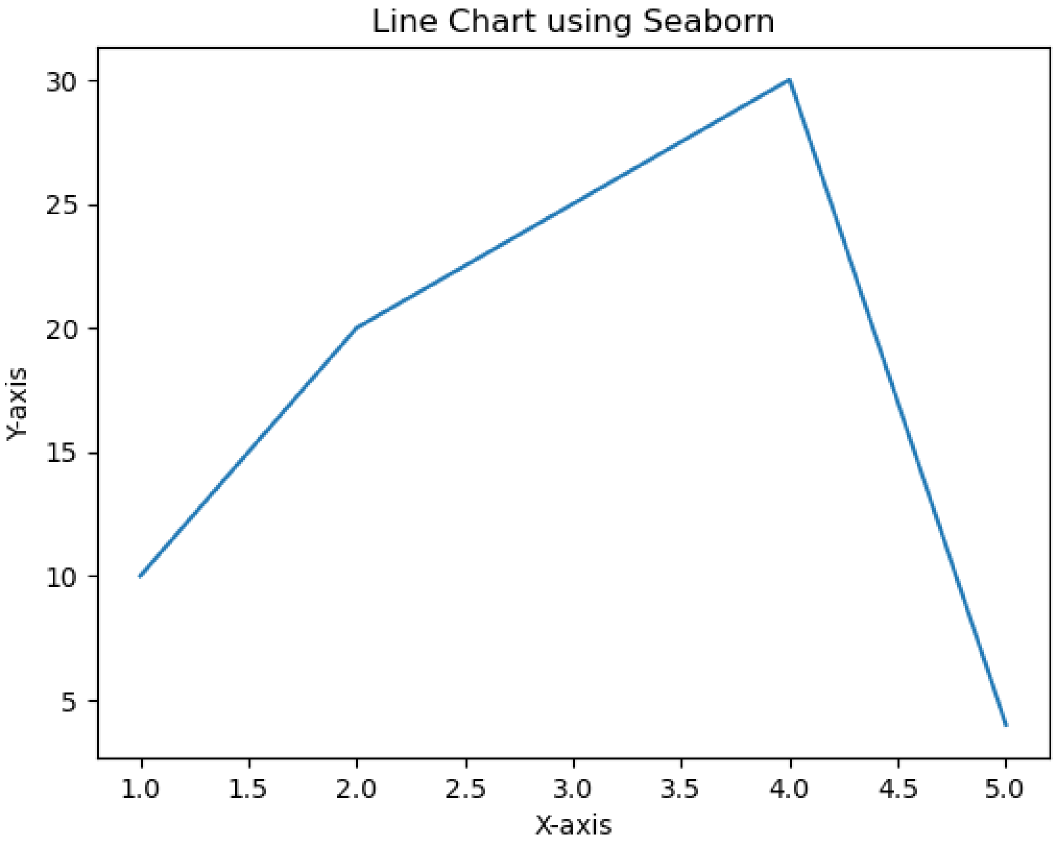
[19]:

```
#This line imports the pandas library and renames it as pd for convenience.
#Pandas is a data analysis and manipulation library used to work with␣
 ↪structured data like Excel, CSV, SQL, etc.
#It provides powerful tools like DataFrame (2D table) and Series (1D array) to␣
 ↪store and process data.
import pandas as pd
df = pd.read_excel("IPL sample data.xlsx")
print(df.head())
```

```
    Pick                                                Y->  Clean Pick  \
0  Throw                                                Y->  Good Throw
1   Runs  "+" stands for runs saved "-" stands for runs …           NaN
2    NaN                                                NaN         NaN
3    NaN                                          Match No.     Innings
4    NaN                                            IPL2367           1

              N->         Fumble        C->          Catch  DC->  \
0             N->      Bad throw       DH->      Dirct Hit  RO->
1             NaN            NaN        NaN            NaN   NaN
2             NaN            NaN        NaN            NaN   NaN
3           Teams    Player Name  BallCount       Position  Pick
4  Delhi Capitals  Rilee russouw        0.1  Short mid wicket     n

  Dropped Catch   S->       Stumping Unnamed: 11          Unnamed: 12
0       Run Out  MR->  Missed Runout         NaN                  NaN
1           NaN   NaN            NaN         NaN                  NaN
2           NaN   NaN            NaN         NaN                  NaN
3         Throw  Runs      Overcount       Venue              Stadium
4           NaN     1              1       Delhi  Arun Jaitly Stadium
```

[23]:

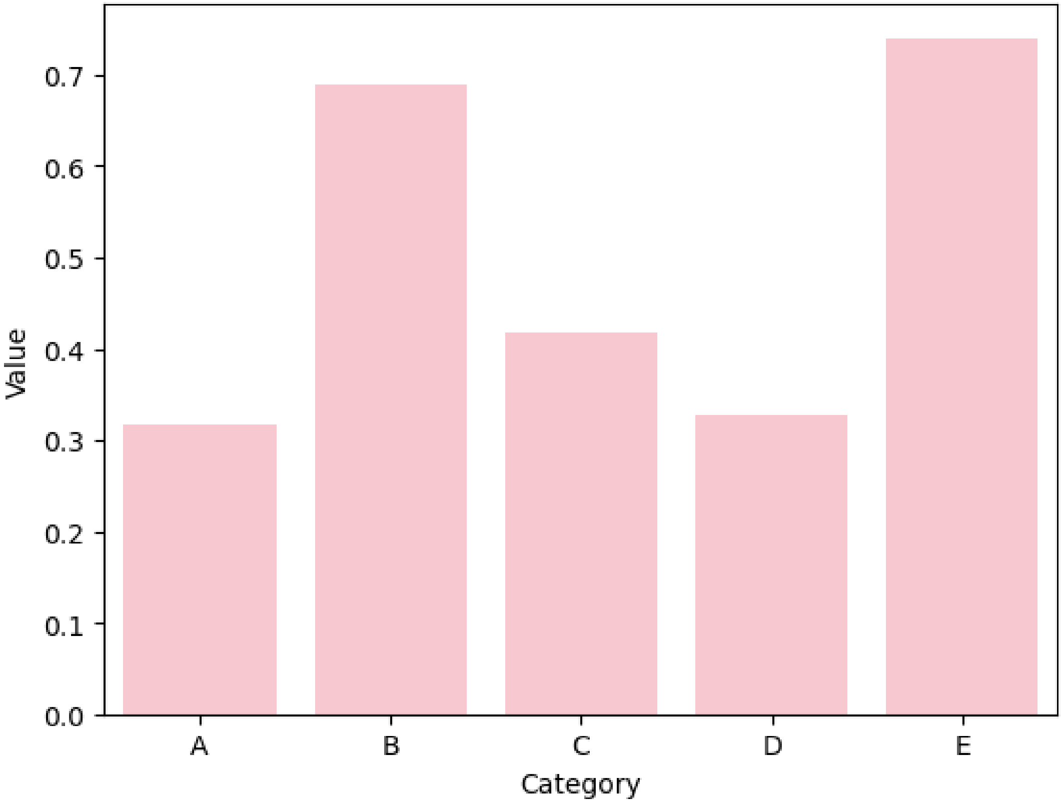
```
# Imports the Seaborn library for creating beautiful and easy statistical plots.
#Imports the NumPy library to create numerical arrays for the data.
# Imports the pandas library to create and manage structured tabular data using␣
 ↪DataFrame.
# Creates a NumPy array x which stores the values for the X-axis.
import seaborn as sns
import numpy as np
import pandas as pd

x = np.array([1, 2, 3, 4, 5])
y = np.array([10, 20, 25, 30, 4])
df = pd.DataFrame({
    'X': x,
    'Y': y
})
sns.lineplot(x='X', y='Y', data=df)
plt.title('Line Chart using Seaborn')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```



[27]:
```
# Creates a NumPy array x containing categorical labels ('A', 'B', 'C', 'D',␣
↪'E') - these will be used on the X-axis.
# Creates a NumPy array y of 5 random float values between 0 and 1, which will␣
↪be used on the Y-axis.
#Converts the x and y arrays into a pandas DataFrame with two columns:
#'Category' (for labels)
```

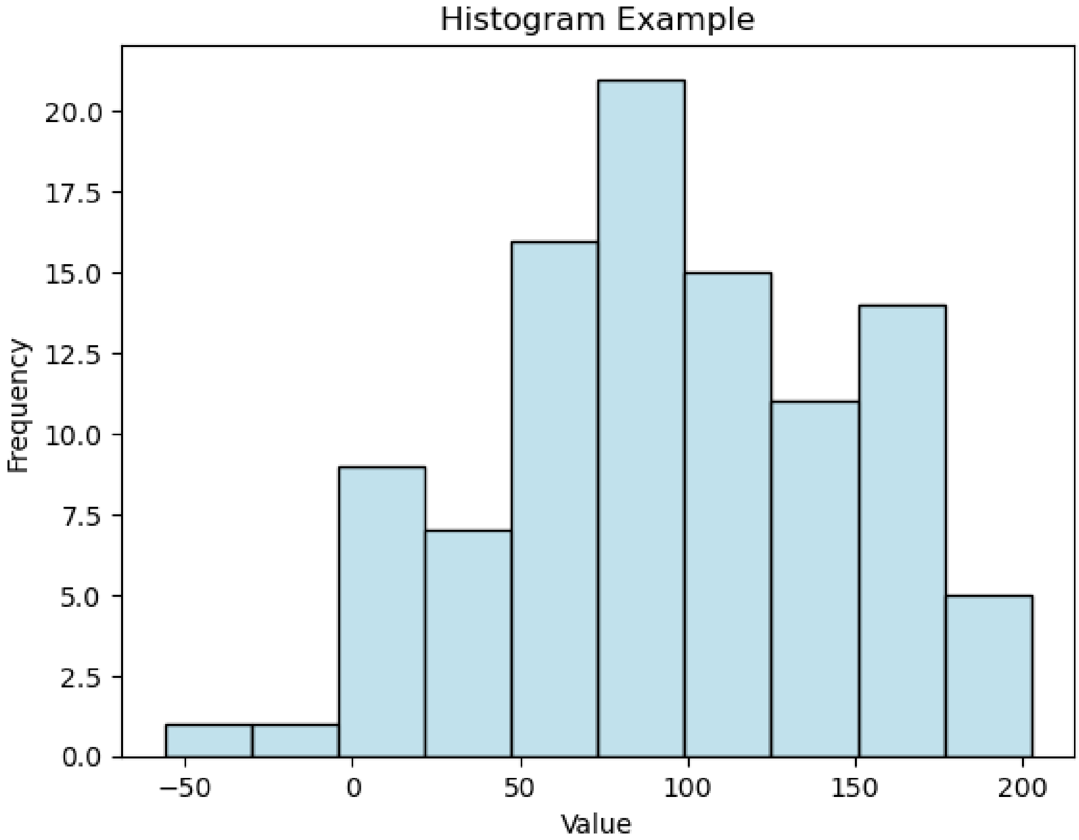
```
#'Value' (for numerical values)
x = np.array(['A', 'B', 'C', 'D', 'E'])
y = np.random.rand(5)
df = pd.DataFrame({'Category': x, 'Value': y})
sns.barplot(data=df, x='Category', y='Value', color='pink')
```

```
[27]: <Axes: xlabel='Category', ylabel='Value'>
```

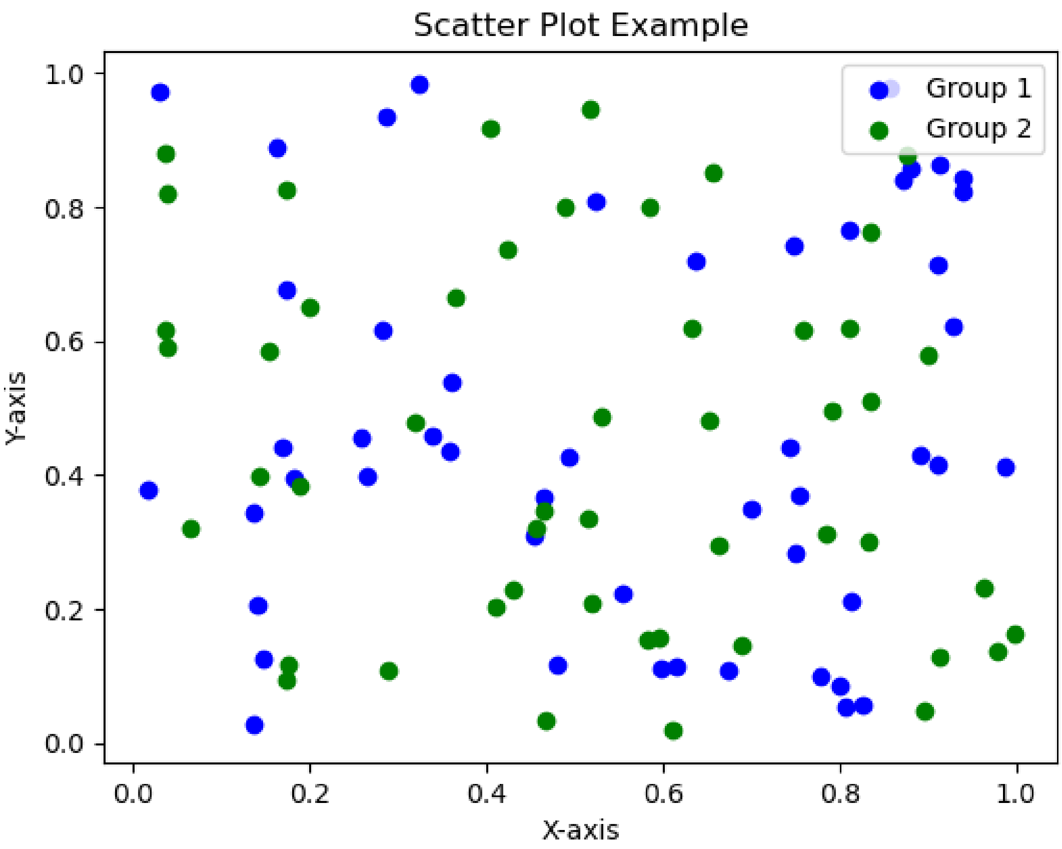


```
[29]: #A histogram is used to display the distribution of continuous numerical data.
#Seaborn provides the **histplot()** function to create histograms easily.
#It divides data into bins (intervals) and shows the frequency of data points␣
 ↪in each bin.
#Helps to understand the shape, spread, and central tendency of the data.
data = np.random.normal(100, 50, 100)
df = pd.DataFrame({'Value': data})
ax = sns.histplot(data=df, x='Value', bins=10, color='lightblue',␣
 ↪edgecolor='black')
ax.set_title('Histogram Example')
ax.set_xlabel('Value')
ax.set_ylabel('Frequency')
```

[29]: Text(0, 0.5, 'Frequency')

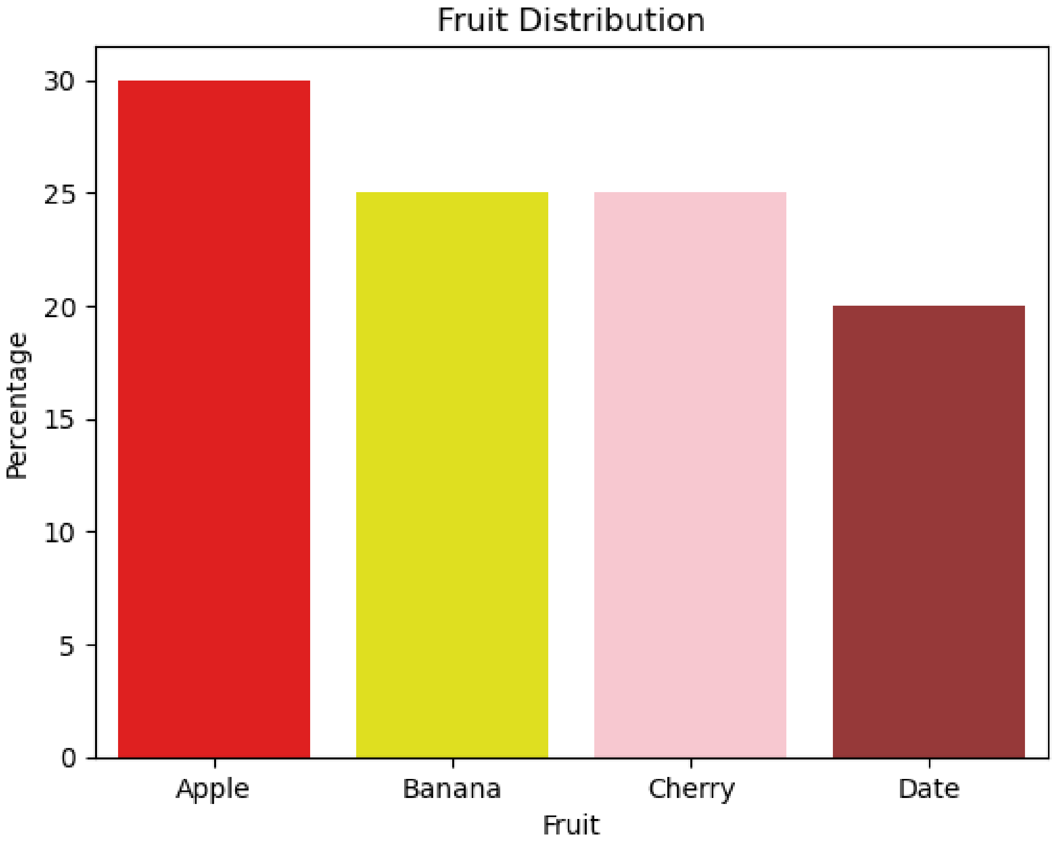


[20]:
```
x1 = np.random.rand(50)
y1 = np.random.rand(50)
x2 = np.random.rand(50)
y2 = np.random.rand(50)

plt.scatter(x1, y1, c='blue', label='Group 1')
plt.scatter(x2, y2, c='green', label='Group 2')
plt.title('Scatter Plot Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.show()
```



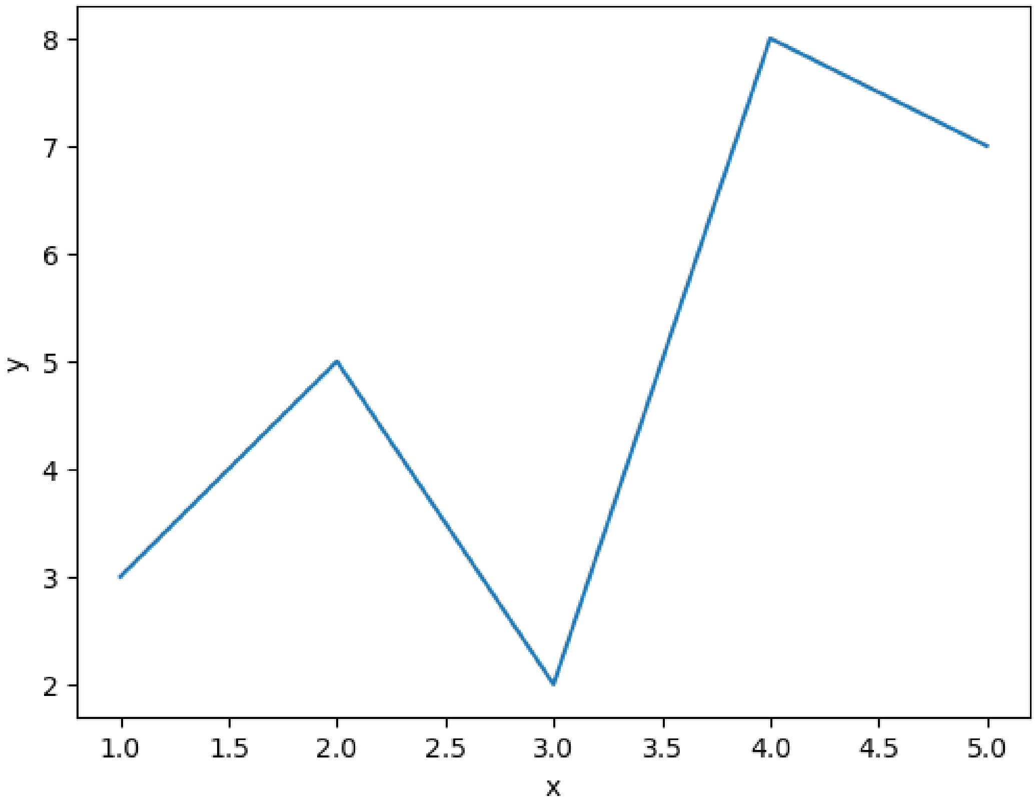
```
[31]: #Seaborn's barplot() is used to create categorical bar charts that show the␣
      ↪relationship between a category and a numerical value.
      #A DataFrame is created with fruit names and their corresponding percentage␣
      ↪values.
      #The hue parameter is set to the same categorical column ('Fruit') to assign␣
      ↪individual colors to each bar using the palette.
      labels = ['Apple', 'Banana', 'Cherry', 'Date']
      sizes = [30, 25, 25, 20]
      colors = ['red', 'yellow', 'pink', 'brown']
      df = pd.DataFrame({'Fruit': labels, 'Percentage': sizes})
      ax = sns.barplot(data=df, x='Fruit', y='Percentage', hue='Fruit',␣
      ↪palette=colors, legend=False)
      ax.set_title('Fruit Distribution')
```

```
[31]: Text(0.5, 1.0, 'Fruit Distribution')
```

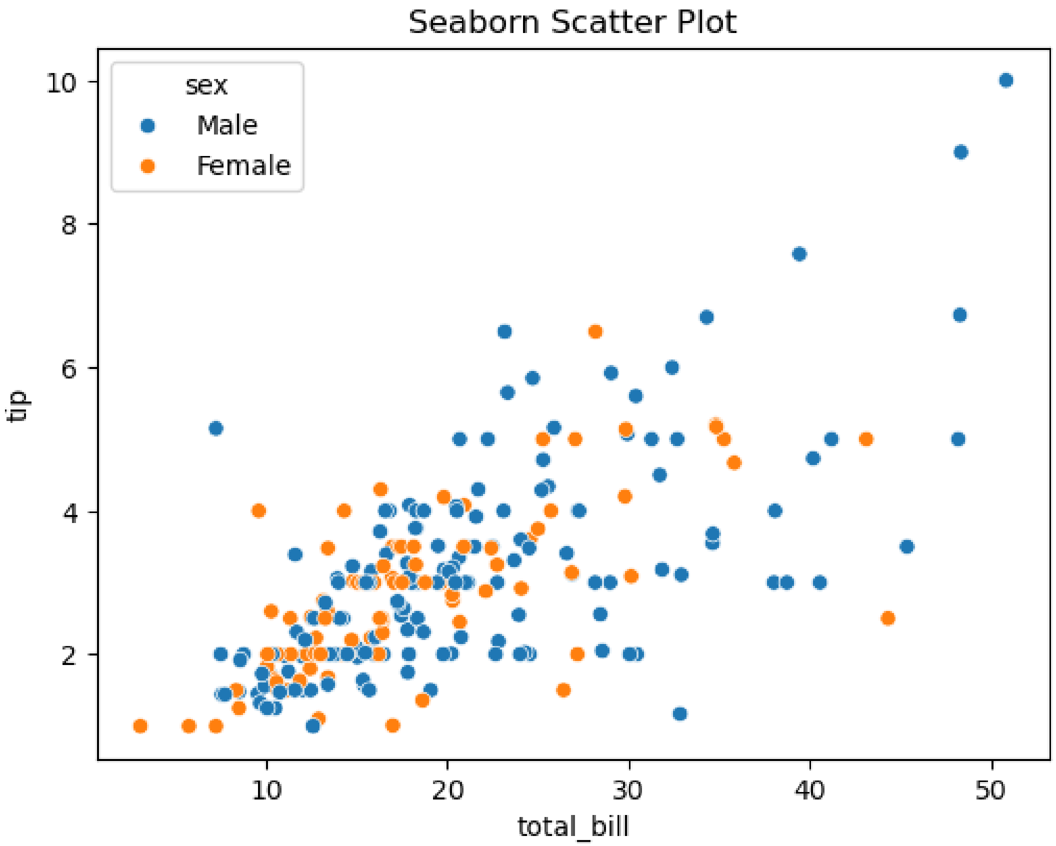


```
[33]: #seaborn’s lineplot() is used to create a line chart that shows the␣
      ↪relationship between two continuous variables.
      #A pandas DataFrame is created with two columns: 'x' (independent variable) and␣
      ↪'y' (dependent variable).
      #The lineplot() function takes the DataFrame as input and plots 'x' on the␣
      ↪X-axis and 'y' on the Y-axis.
      df = pd.DataFrame({
          'x': [1, 2, 3, 4, 5],
          'y': [3, 5, 2, 8, 7]
      })
      sns.lineplot(data=df, x='x', y='y')
```

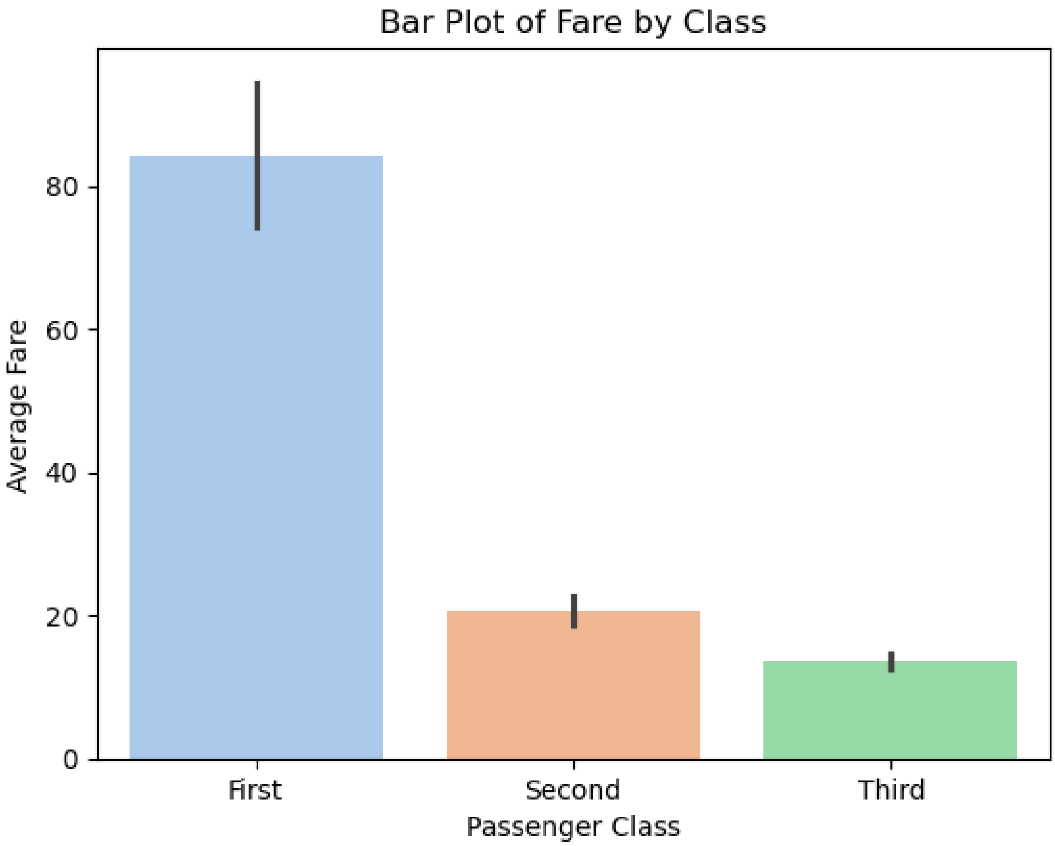
```
[33]: <Axes: xlabel='x', ylabel='y'>
```



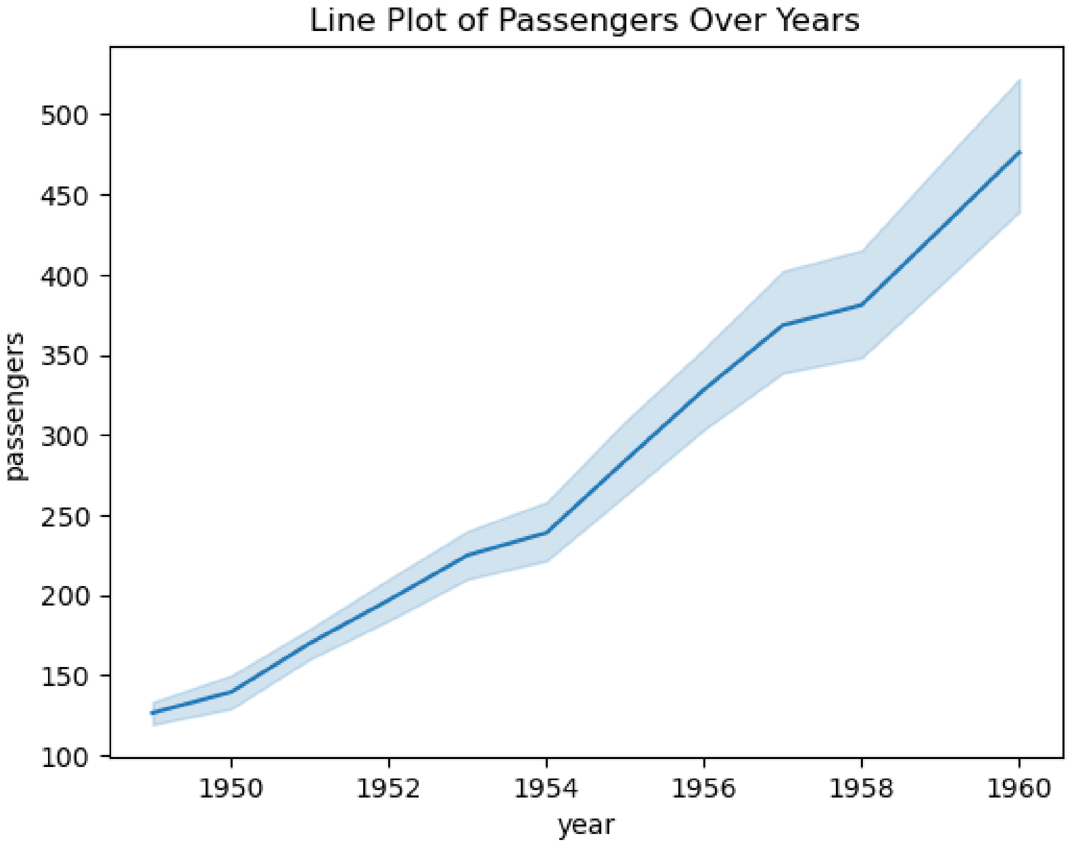
```
[35]: #Seaborn's scatterplot() is used to create a scatter plot, which shows the␣
       ↪relationship between two continuous variables.
      #The built-in "tips" dataset is loaded using sns.load_dataset(), which contains␣
       ↪restaurant bill and tip data.
      #The x and y parameters are set to 'total_bill' and 'tip', respectively, to␣
       ↪plot each data point.
      import seaborn as sns
      df = sns.load_dataset("tips")
      sns.scatterplot(x='total_bill', y='tip', hue='sex', data=df)
      plt.title('Seaborn Scatter Plot')
      plt.show()
```



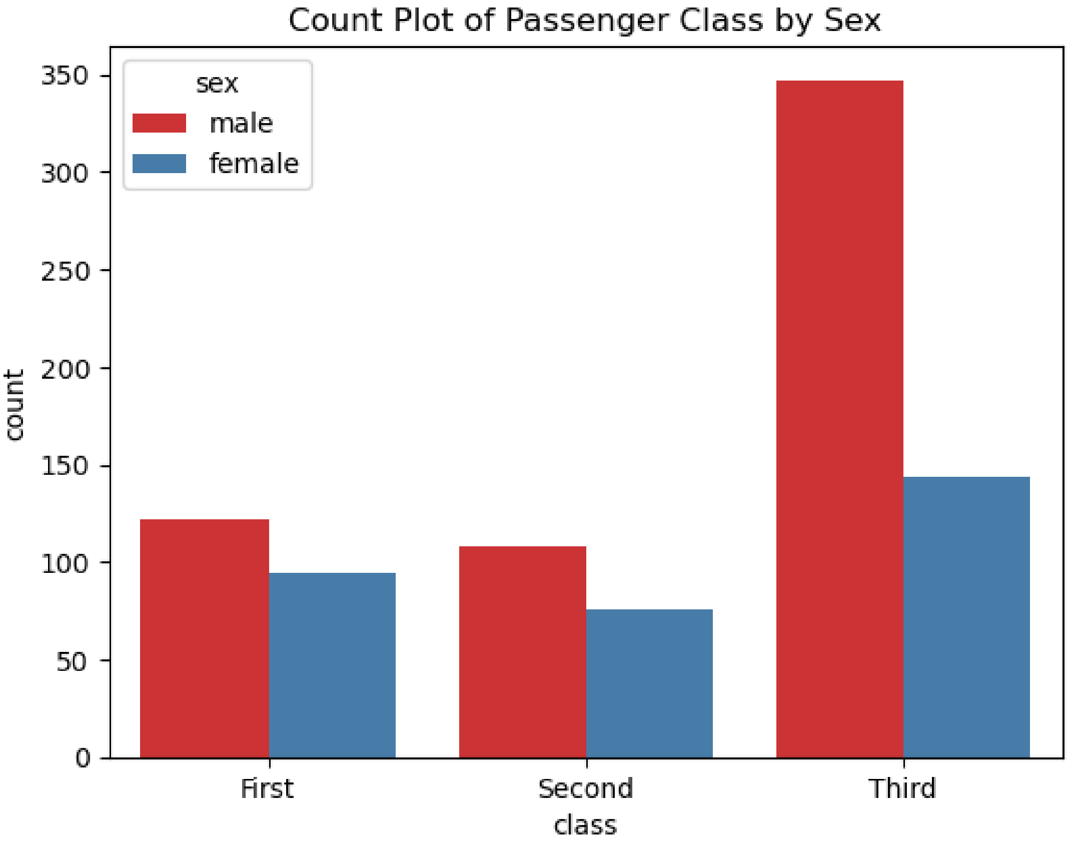
[119]:
```
# Seaborn’s barplot() is used to create a bar chart that shows the average fare␣
 ↪paid by passengers in each class from the Titanic dataset.
# The built-in Titanic dataset is loaded using sns.load_dataset("titanic").
# The x and y parameters are set to 'class' and 'fare', which plot passenger␣
 ↪classes on the X-axis and their corresponding average fares on the Y-axis.
# The hue='class' parameter assigns different colors to each bar based on␣
 ↪class, while palette='pastel' applies soft, pleasant colors.
import seaborn as sns
df = sns.load_dataset("titanic")
sns.barplot(x='class', y='fare', hue='class', data=df, palette='pastel',␣
 ↪legend=False)
plt.title('Bar Plot of Fare by Class')
plt.xlabel('Passenger Class')
plt.ylabel('Average Fare')
plt.show()
```



[57]:
```
# Seaborn's lineplot() is used to visualize trends over time or continuous␣
↪values.
# The built-in 'flights' dataset is loaded using sns.load_dataset("flights"),
# which contains data about monthly airline passengers over years.
# The x parameter is set to 'year' and the y to 'passengers',
# showing how passenger numbers changed over the years.
df = sns.load_dataset("flights")
sns.lineplot(x='year', y='passengers', data=df)
plt.title('Line Plot of Passengers Over Years')
plt.show()
```

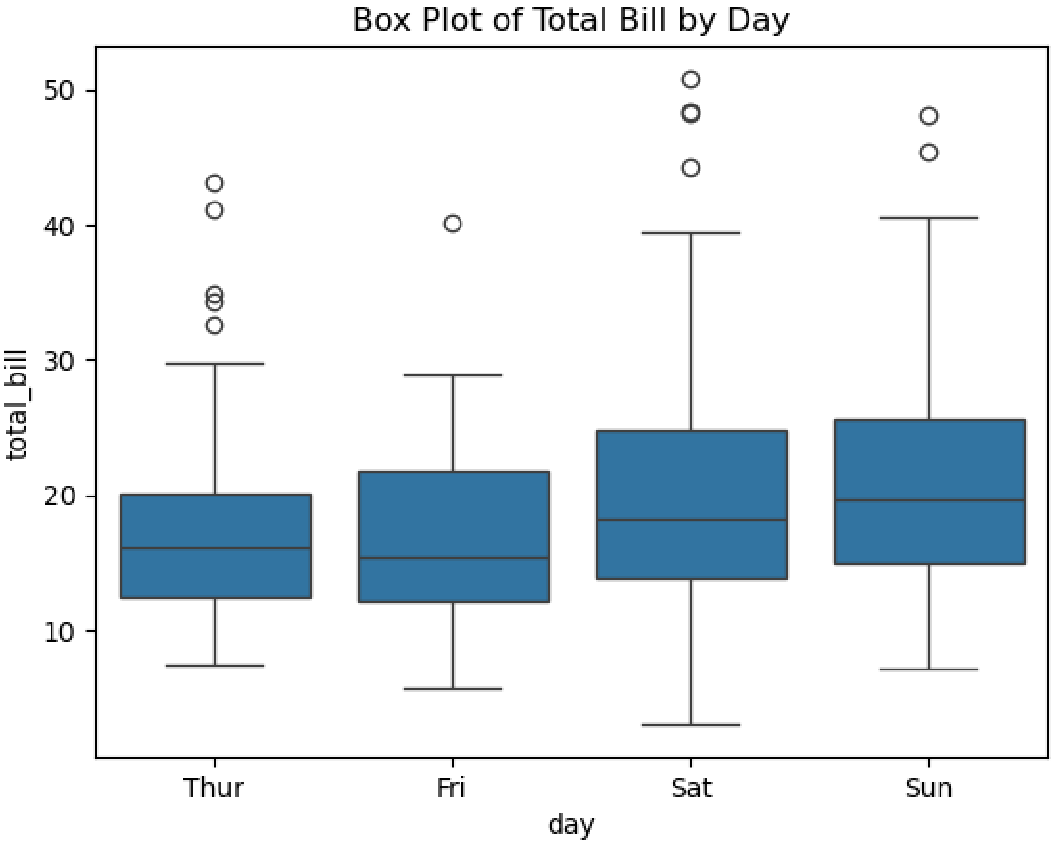


```
[59]: #Seaborn's countplot() is used to display the count of observations in each␣
       ↪categorical bin using bars.
      #The titanic dataset is loaded using sns.load_dataset("titanic"), which␣
       ↪contains passenger data.
      #The x='class' parameter sets the passenger class as the category to group by.
      #The hue='sex' parameter further separates bars based on passenger sex (male/
       ↪female).
      #The palette='Set1' applies a predefined color scheme for better visual␣
       ↪distinction.
      df = sns.load_dataset("titanic")
      sns.countplot(x='class', hue='sex', palette='Set1', data=df)
      plt.title('Count Plot of Passenger Class by Sex')
      plt.show()
```



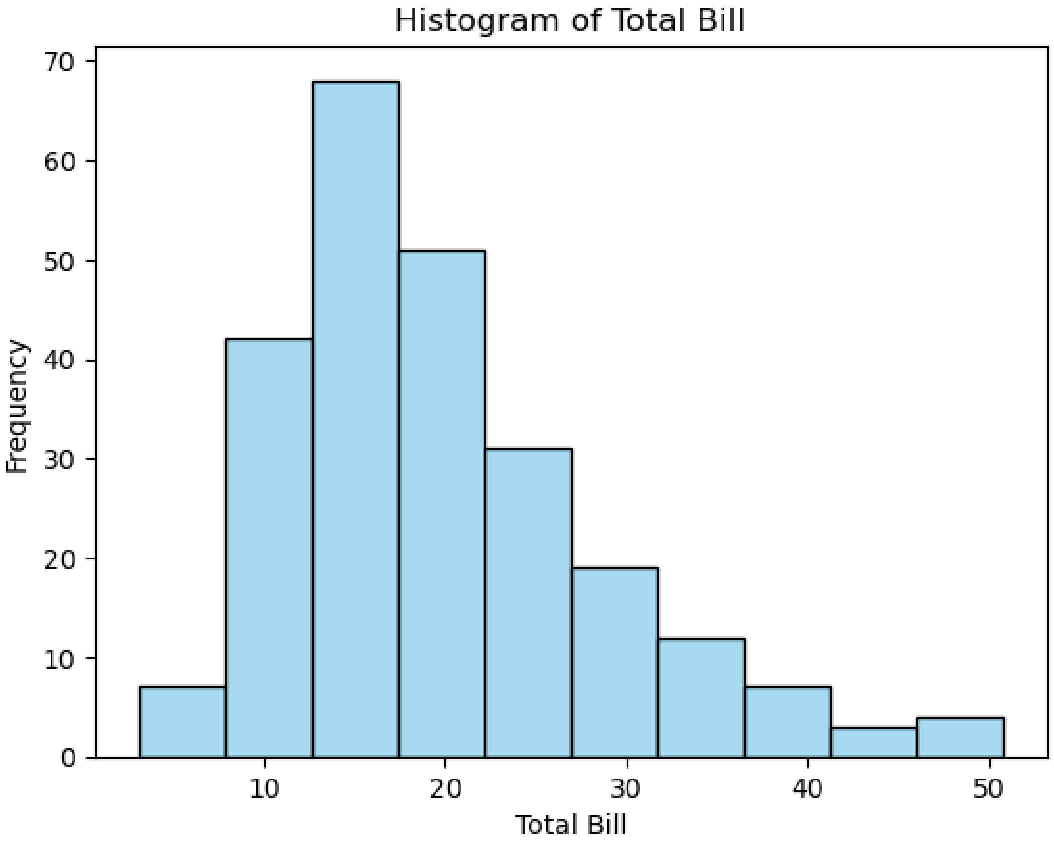
[61]:

```
#Seaborn’s boxplot() is used to visualize the distribution, spread, and outliers of numerical data across different categories.
#The tips dataset is loaded using sns.load_dataset("tips"), which contains data about restaurant bills, tips, gender, smoking habits, etc.
#The x='day' and y='total_bill' parameters indicate that the plot will compare total bills for each day of the week.
#This helps identify central tendency, spread, and any outliers in the total_bill amounts for each day.

df = sns.load_dataset("tips")
sns.boxplot(x='day', y='total_bill', data=df)
plt.title('Box Plot of Total Bill by Day')
plt.show()
```

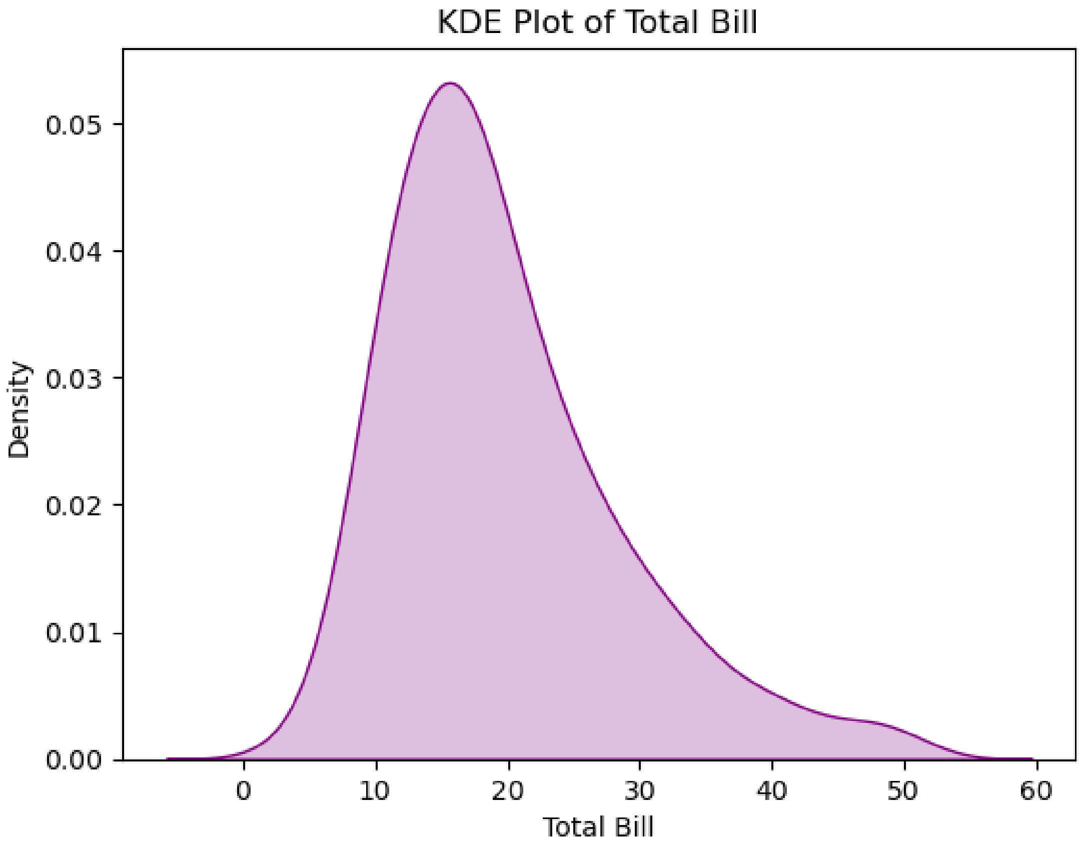


[67]:

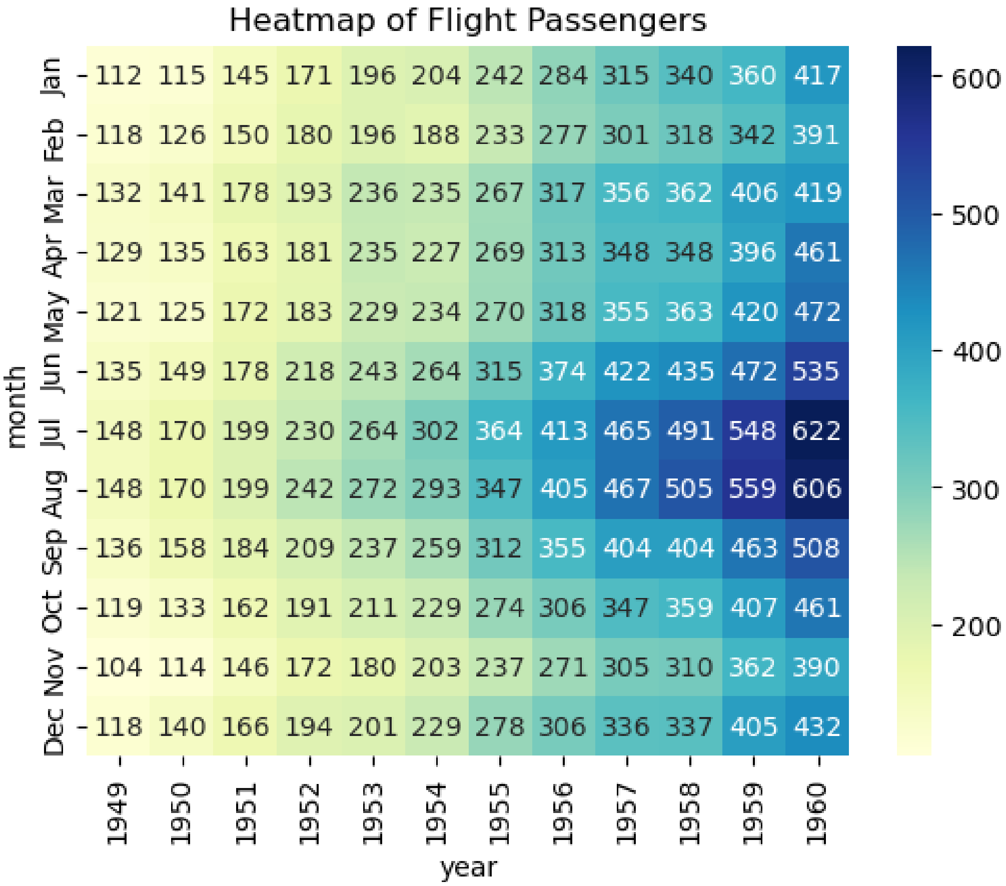
```
#The tips dataset is loaded using sns.load_dataset("tips"), which includes␣
  ↪information about restaurant bills and tips.
#sns.histplot() is used to create a histogram of the total_bill column.
#bins=10 divides the total bill values into 10 equal intervals.
#kde=False disables the kernel density estimate line, showing only the␣
  ↪histogram bars.
#color='sky blue' sets the bar color to sky blue.
df = sns.load_dataset("tips")
sns.histplot(df['total_bill'], bins=10, kde=False, color='skyblue',␣
  ↪edgecolor='black')
plt.title('Histogram of Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Frequency')
plt.show()
```



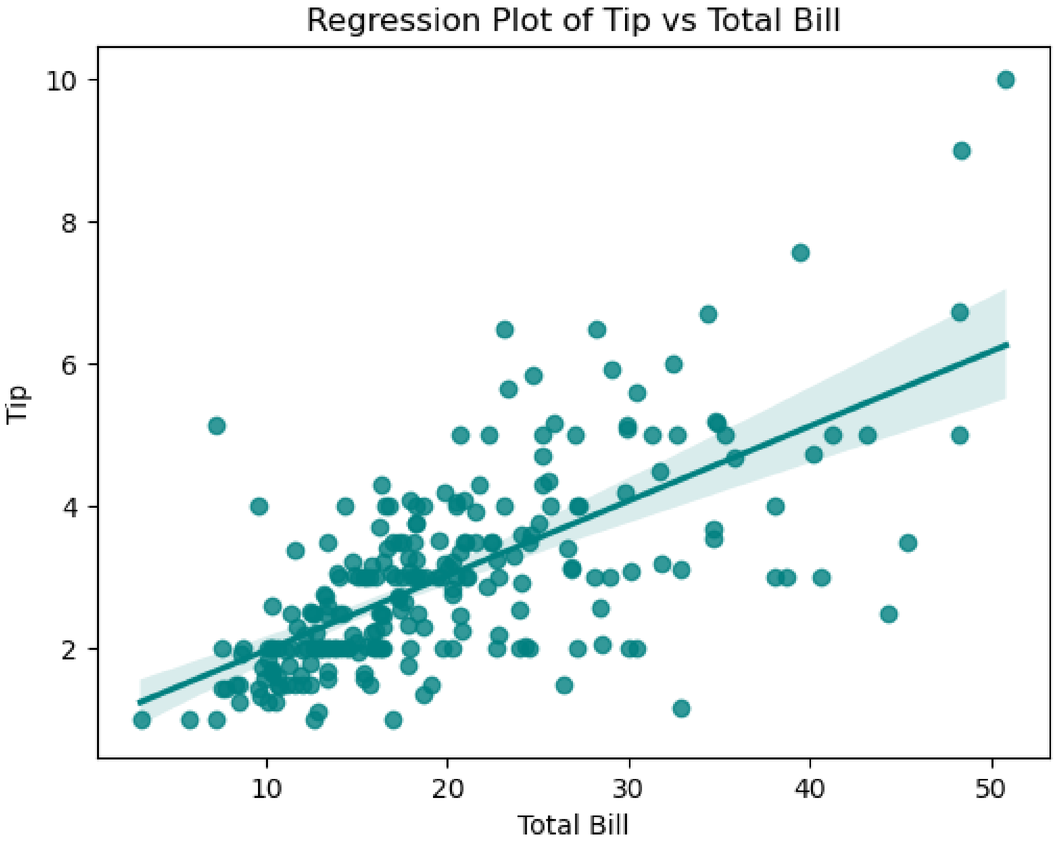
[69]:
```
#The tips dataset is loaded using sns.load_dataset("tips"), which includes␣
↪restaurant billing and tipping data.
#sns.kdeplot() is used to draw the Kernel Density Estimate (KDE) plot of the␣
↪total_bill column.
#fill=True fills the area under the KDE curve for better visualization.
#color='purple' sets the color of the KDE curve and filled area.
#plt.title(), plt.xlabel(), and plt.ylabel() add a title and axis labels to␣
↪describe what the plot represents.
df = sns.load_dataset("tips")
sns.kdeplot(df['total_bill'], fill=True, color='purple')
plt.title('KDE Plot of Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Density')
plt.show()
```



[73]:
```
#df = sns.load_dataset("flights"): Loads the flights dataset, which contains␣
 ↪monthly passenger counts for different years.
#df.pivot(index="month", columns="year", values="passengers"): Transforms the␣
 ↪dataset into a pivot table where months are rows, years are columns, and␣
 ↪values are passenger counts.
#sns.heatmap(...): Creates a heatmap where each cell's color represents the␣
 ↪number of passengers.
#data: the pivot table is used as the heatmap input.
#cmap="YlGnBu": applies a yellow-green-blue color gradient.
#annot=True: shows the exact passenger values inside each cell.
#fmt="d": formats annotations as integers.

df = sns.load_dataset("flights")
data = df.pivot(index="month", columns="year", values="passengers")
sns.heatmap(data, cmap="YlGnBu", annot=True, fmt="d")
plt.title("Heatmap of Flight Passengers")
plt.show()
```



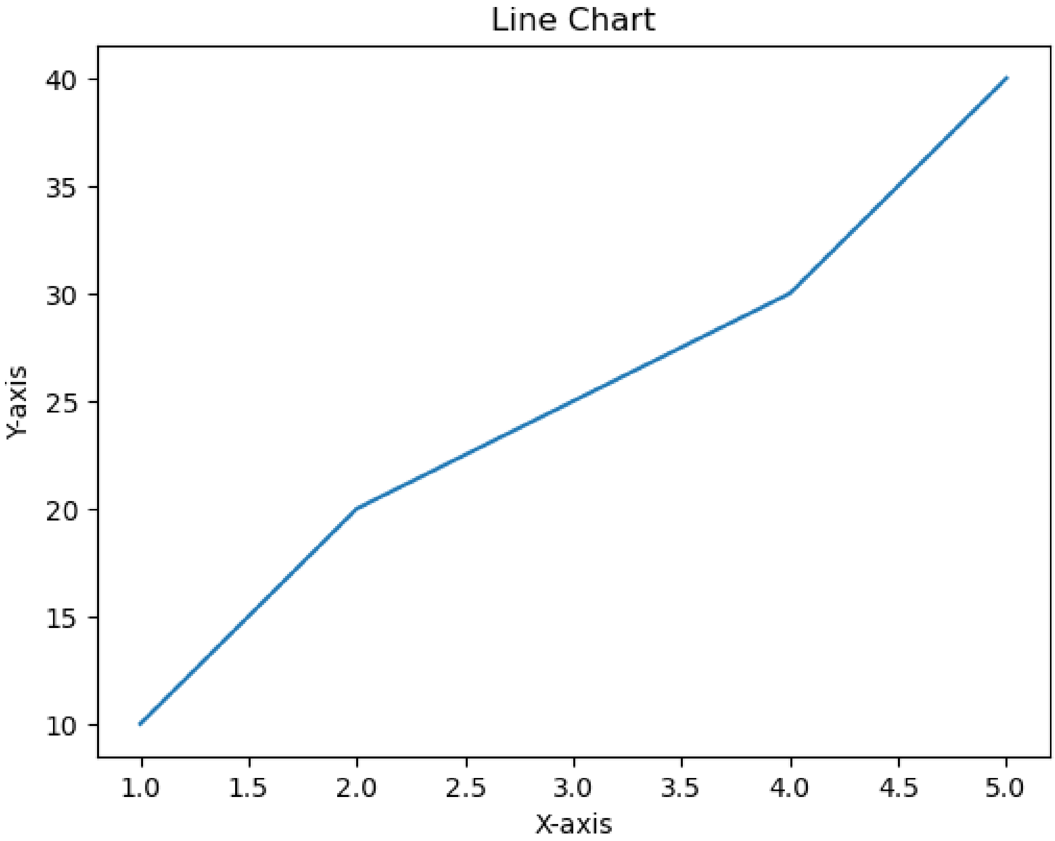
```
[77]: #df = sns.load_dataset("tips"): Loads the built-in tips dataset, which contains␣
       ↪data about restaurant bills and tips.
      #sns.regplot(x='total_bill', y='tip', data=df, color='teal')
      #Creates a regression plot to visualize the relationship between the total bill␣
       ↪and the tip amount.
      #Plots individual data points and fits a linear regression line.
      #color='teal': Sets the color of the plot to teal.
      #plt.title('Regression Plot of Tip vs Total Bill'): Adds a title to the plot.
      df = sns.load_dataset("tips")
      sns.regplot(x='total_bill', y='tip', data=df, color='teal')
      plt.title('Regression Plot of Tip vs Total Bill')
      plt.xlabel('Total Bill')
      plt.ylabel('Tip')
      plt.show()
```



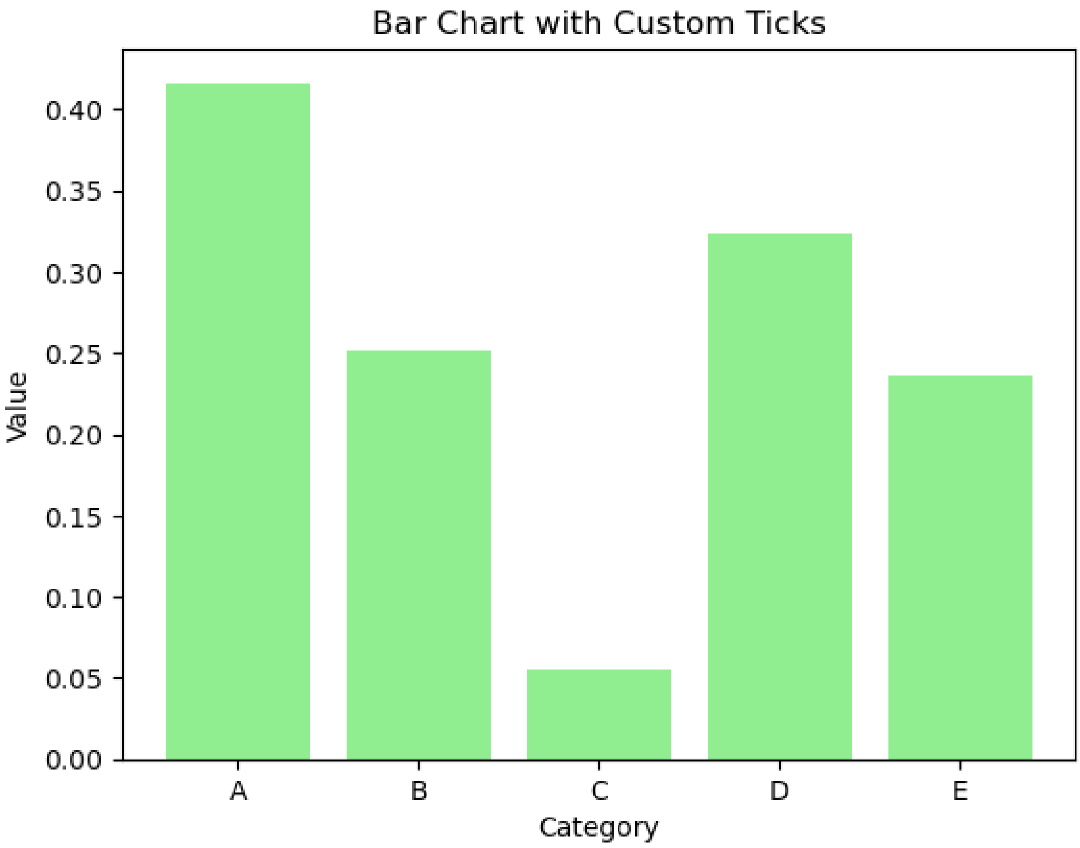
1 Matplotlib

[85]:

```
#import matplotlib.pyplot as plt:
#Imports the Matplotlib plotting library and assigns it the alias plt.
#import numpy as np:
#Imports NumPy, a library used for numerical operations.
#x = np.array([1, 2, 3, 4, 5]):
#Creates a NumPy array x with values [1, 2, 3, 4, 5] representing the x-axis␣
 ↪data.
#y = np.array([10, 20, 25, 30, 40]):
#Creates a NumPy array y with values [10, 20, 25, 30, 40] representing the␣
 ↪y-axis data.
import matplotlib.pyplot as plt
import numpy as np

x = np.array([1, 2, 3, 4, 5])
y = np.array([10, 20, 25, 30, 40])

plt.plot(x, y)
plt.title('Line Chart')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

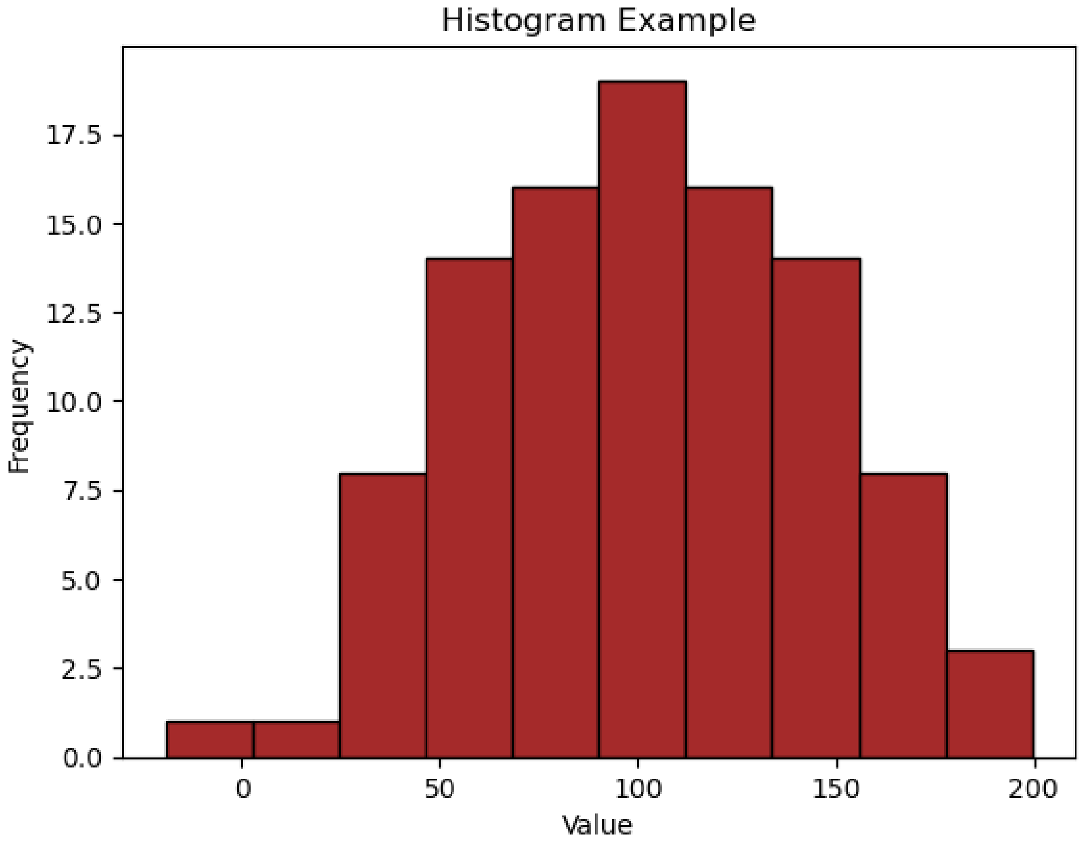


[101]:
```
#Imports Matplotlib for creating visualizations.
#Imports NumPy for generating random numerical data.
#Imports Pandas for handling data in DataFrame format.
#x = np.array(['A', 'B', 'C', 'D', 'E'])
#Defines the category labels for the x-axis.
#y = np.random.rand(5):
#Generates 5 random values between 0 and 1 for the y-axis.
#df = pd.DataFrame({'Category': x, 'Value': y}):
#Creates a DataFrame with two columns: 'Category' and 'Value'.
indices = np.arange(len(df))
plt.bar(indices, df['Value'], color='lightgreen')
plt.xticks(indices, df['Category'])
plt.title('Bar Chart with Custom Ticks')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()
```

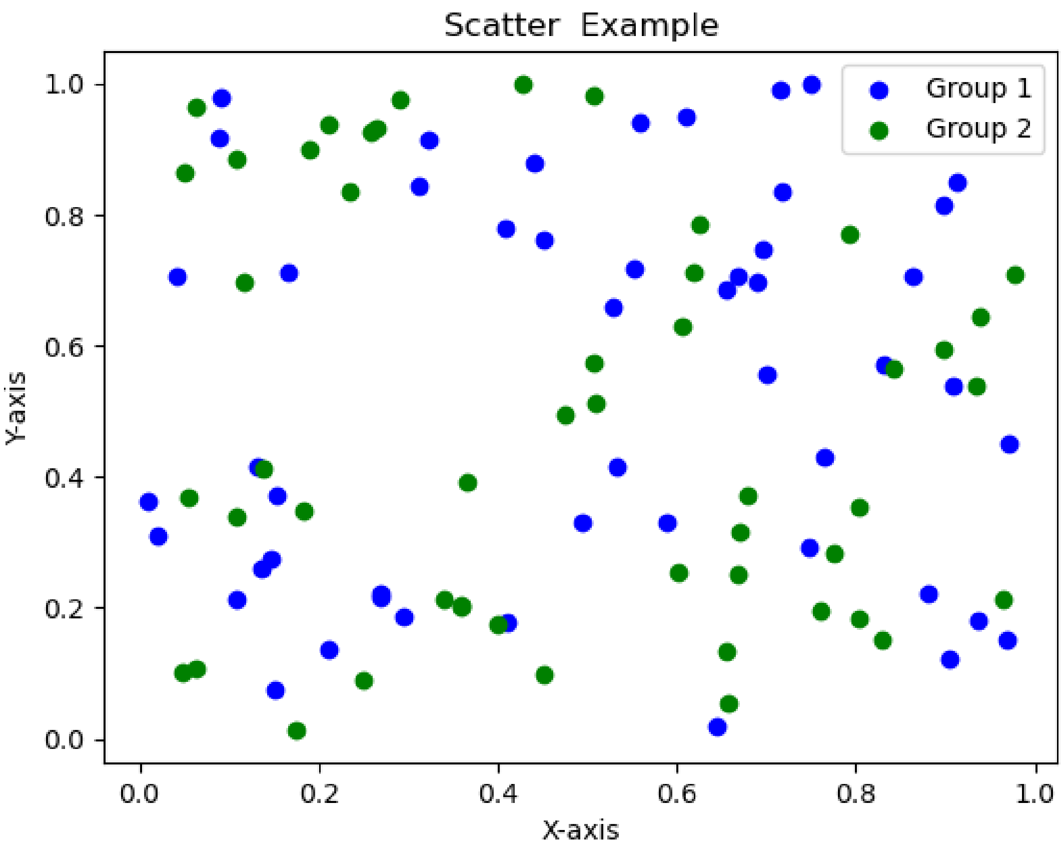


[109]:

```
#Matplotlib is used for creating visualizations in Python.
#A histogram shows the distribution of numerical data.
#First, data is generated or collected (e.g., random numbers).
#The data is grouped into bins (ranges of values).
#plt.hist() is used to plot the histogram.
#Bars represent how many data points fall into each bin.
data = np.random.normal(100, 50, 100)
df = pd.DataFrame({'Value': data})
# Plot histogram using Matplotlib
plt.hist(df['Value'], bins=10, color='brown', edgecolor='black')
plt.title('Histogram Example')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.show()
```

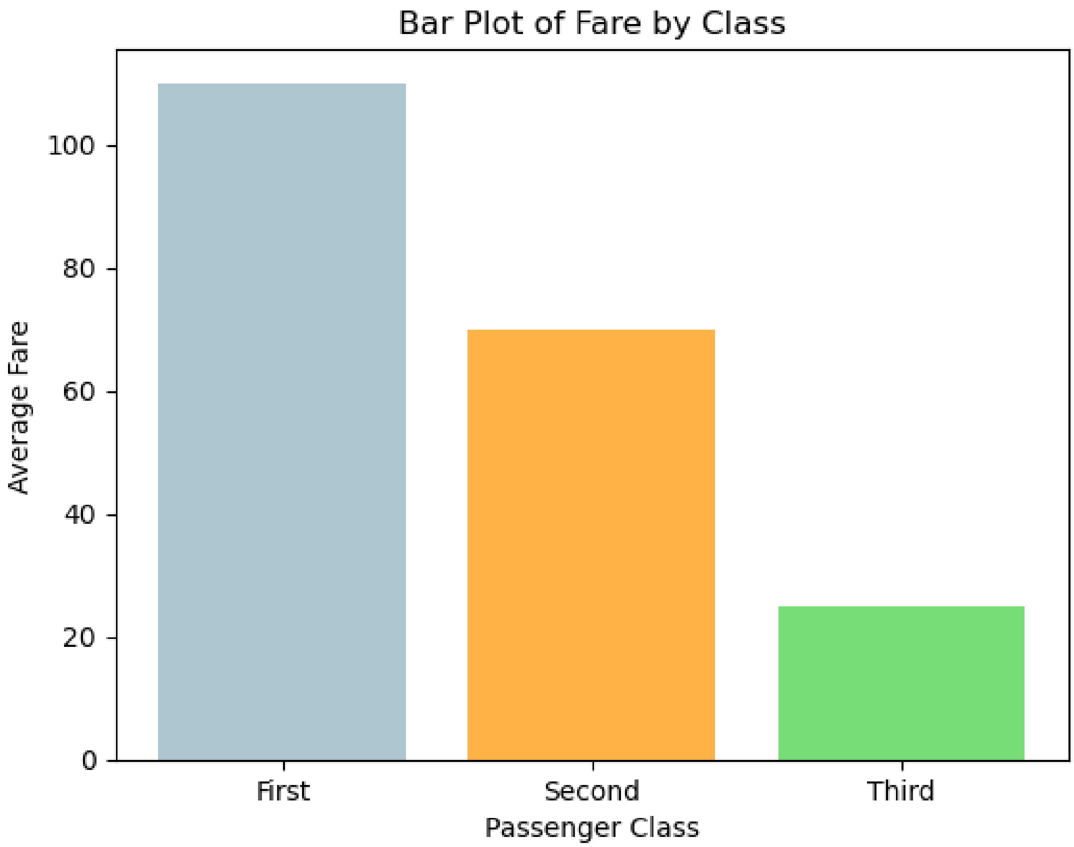


[117]:
```
#Import matplotlib.pyplot and numpy.
#Generate random data using np.random.rand().
#Use plt.scatter() to plot data points.
#Set different colors for different groups.
#Add title, xlabel, and ylabel for context.
#Use plt.legend() to show group labels.
#Use plt.show() to display the plot.
import matplotlib.pyplot as plt
import numpy as np
x1 = np.random.rand(50)
y1 = np.random.rand(50)
x2 = np.random.rand(50)
y2 = np.random.rand(50)
plt.scatter(x1, y1, c='blue', label='Group 1')
plt.scatter(x2, y2, c='green', label='Group 2')
plt.title('Scatter  Example')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.legend()
plt.show()
```



[5]:

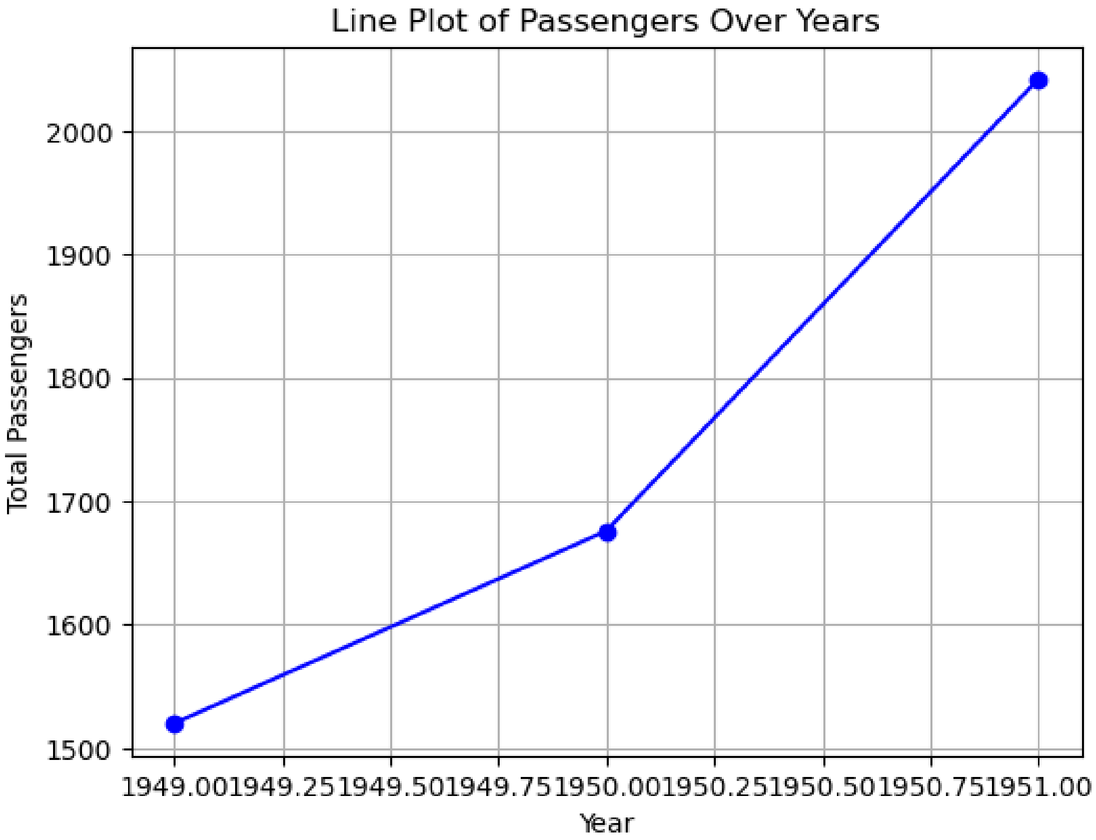
```
#A bar plot is used to represent categorical data using rectangular bars, where␣
↪the height of each bar represents the value associated with that category.
#We manually create a dataset with two columns: class and fare.
#Then we calculate the average fare for each class using groupby() and mean().
#Finally, we plot a bar chart using plt.bar().
import matplotlib.pyplot as plt
import pandas as pd
data = {
    'class': ['First', 'Second', 'Third', 'First', 'Second', 'Third', 'First',␣
↪'Second', 'Third'],
    'fare': [100, 70, 30, 120, 60, 25, 110, 80, 20]
}
df = pd.DataFrame(data)
class_fare = df.groupby('class')['fare'].mean().sort_index()
# Plot using Matplotlib
plt.bar(class_fare.index, class_fare.values, color=['#AEC6CF', '#FFB347',␣
↪'#77DD77'])
plt.title('Bar Plot of Fare by Class')
plt.xlabel('Passenger Class')
plt.ylabel('Average Fare')
plt.show()
```



[11]:

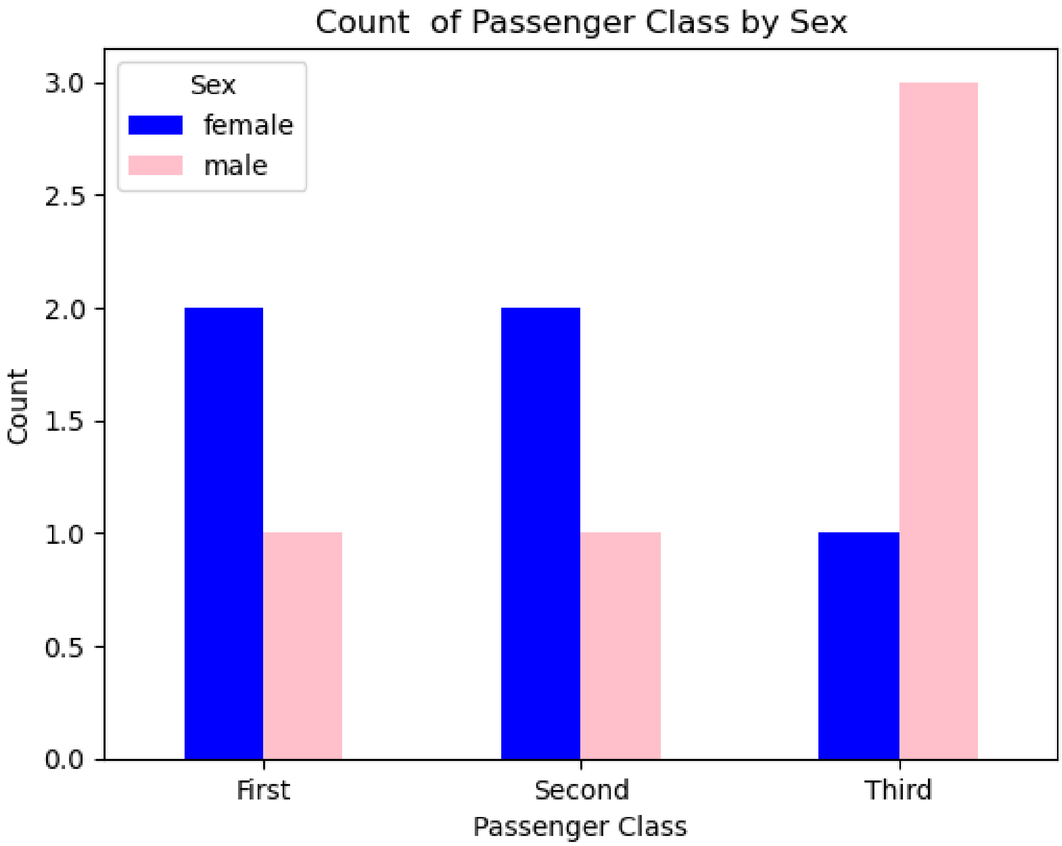
```
#A line plot is used to visualize trends over time by connecting data points␣
↪with straight lines.
#In this example, we are plotting the number of passengers over the years using␣
↪only Matplotlib and Pandas
#The dataset includes the number of airline passengers per month across years.
#We group the data by year and calculate the total passengers per year.
import matplotlib.pyplot as plt
import pandas as pd
data = {
    'year': [1949]*12 + [1950]*12 + [1951]*12,
    'month': ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
              'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'] * 3,
    'passengers': [
        112, 118, 132, 129, 121, 135, 148, 148, 136, 119, 104, 118,
        115, 126, 141, 135, 125, 149, 170, 170, 158, 133, 114, 140,
        145, 150, 178, 163, 172, 178, 199, 199, 184, 162, 146, 166
    ]
}
```

```
df = pd.DataFrame(data)
yearly_data = df.groupby('year')['passengers'].sum()
plt.plot(yearly_data.index, yearly_data.values, marker='o', color='blue')
plt.title('Line Plot of Passengers Over Years')
plt.xlabel('Year')
plt.ylabel('Total Passengers')
plt.grid(True)
plt.show()
```



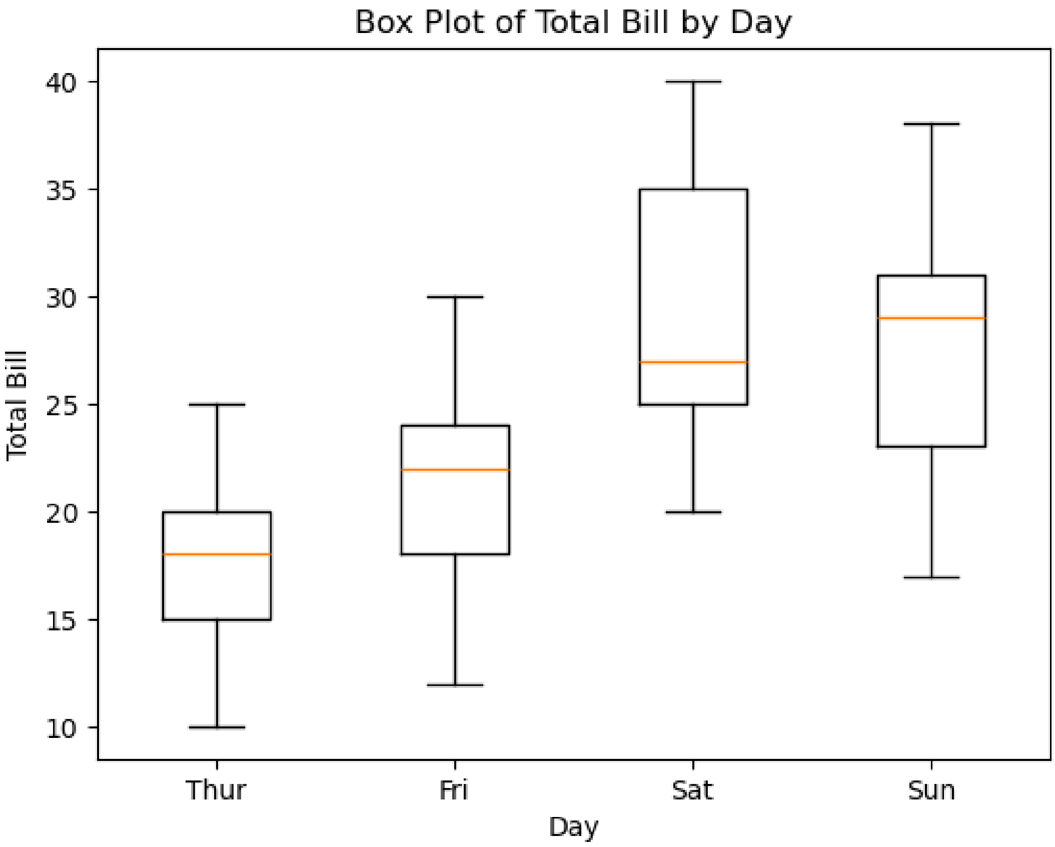
[25]:
```
#plt.bar() is used to plot two bars (male and female) side by side for each␣
↪class.
#X-axis shows the passenger classes, and Y-axis shows the count.
#legend() is used to label male and female bars.
#This simple version uses hardcoded counts to demonstrate grouped bar plotting.
data = {
    'class': ['First', 'Second', 'Third', 'First', 'Second', 'Third', 'Third',␣
↪'Second', 'First', 'Third'],
    'sex':   ['male', 'female', 'male', 'female', 'female', 'male', 'female',␣
↪'male', 'female', 'male']
}
```

```
df = pd.DataFrame(data)
counts = df.groupby(['class', 'sex']).size().unstack(fill_value=0)
counts.plot(kind='bar', stacked=False, color=['blue', 'pink'])
plt.title('Count  of Passenger Class by Sex')
plt.xlabel('Passenger Class')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.legend(title='Sex')
plt.show()
```



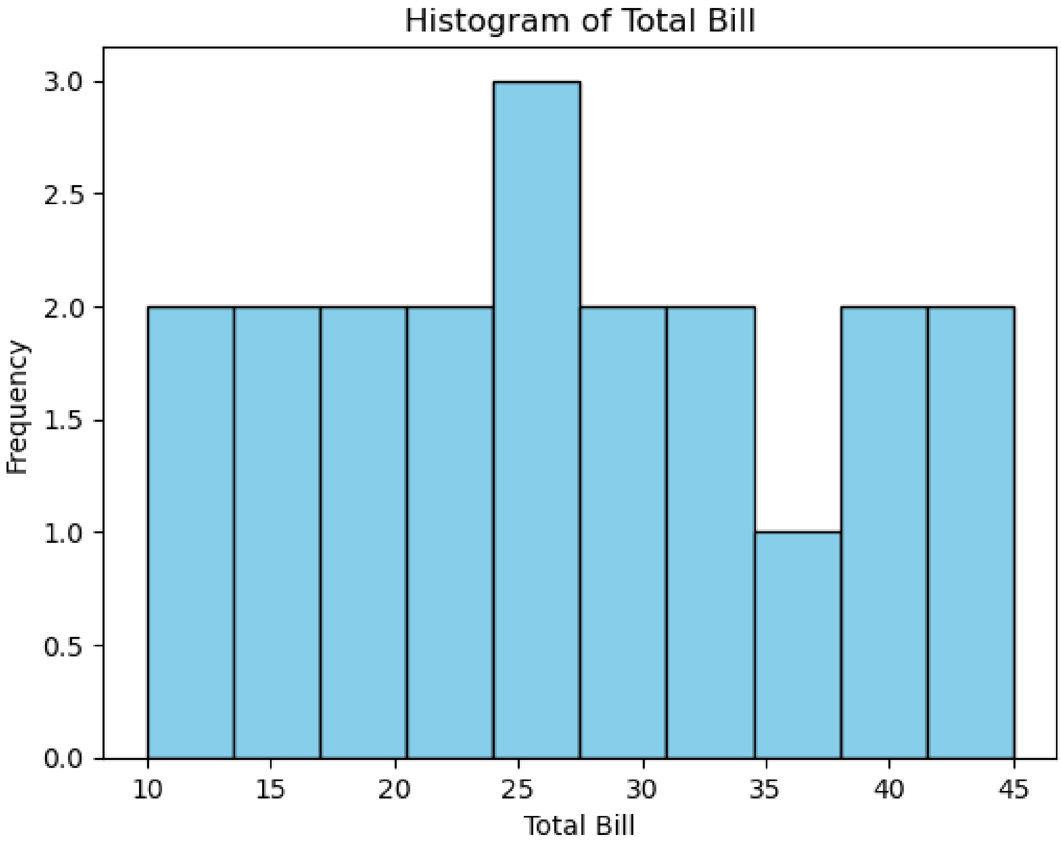
[35]:

```
#A box plot is used to visualize the spread and distribution of numerical data.
#It highlights the median, quartiles, and potential outliers in the dataset.
#We compare the total bill amounts across different days (Thur, Fri, Sat, Sun)␣
 ↪using Matplotlib only.
#Each box represents the distribution of bills on that day.

data = {
    'Thur': [10, 15, 20, 25, 18],
    'Fri': [12, 18, 22, 24, 30],
    'Sat': [20, 25, 27, 35, 40],
    'Sun': [17, 23, 29, 31, 38]
}
days = list(data.keys())
values = [data[day] for day in days]
plt.boxplot(values, tick_labels=days)
plt.title('Box Plot of Total Bill by Day')
plt.xlabel('Day')
plt.ylabel('Total Bill')
plt.show()
```



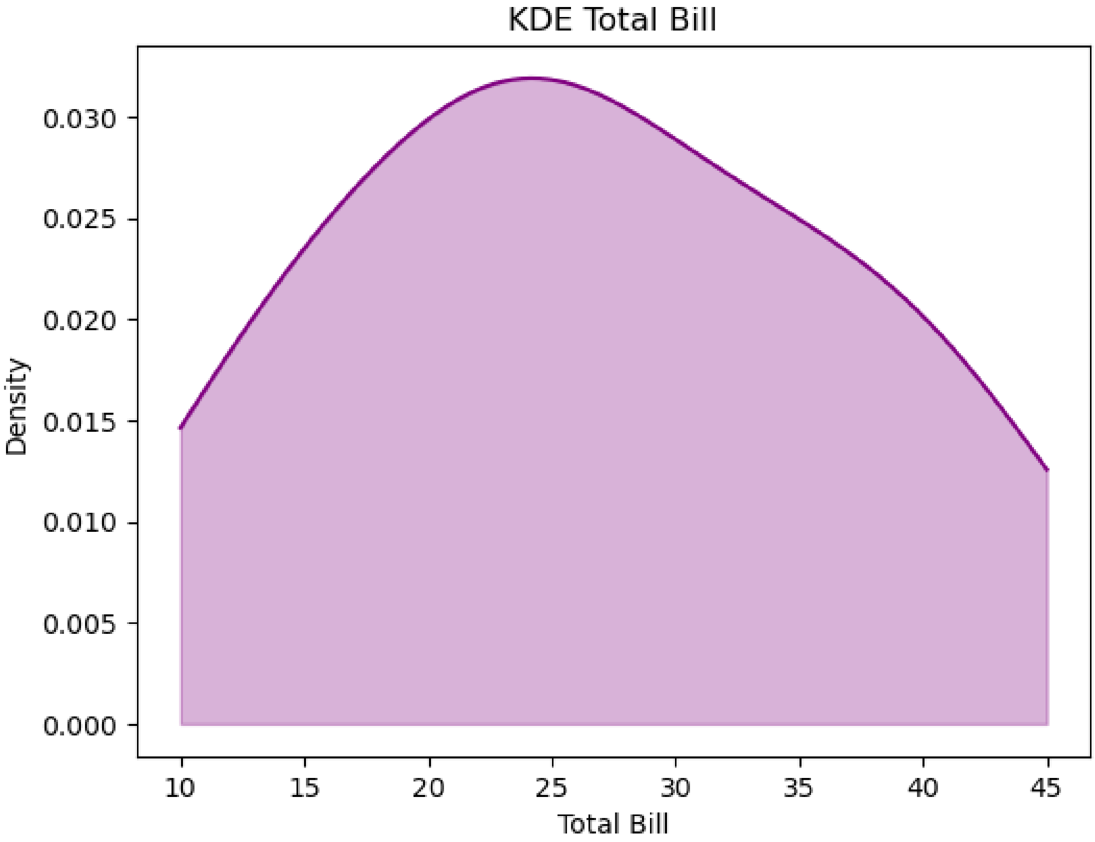
[42]:

```
#A histogram displays the frequency distribution of a numerical variable by␣
 ↪grouping data into bins.
#This example shows how the total_bill values are distributed using Matplotlib␣
 ↪only.
#plt.hist() creates the histogram with 10 bins.
#color sets the bar color and edgecolor outlines the bars.
total_bill = [10, 12, 14, 16, 18, 20, 22, 23, 24, 25,
              26, 28, 30, 32, 34, 36, 38, 40, 42, 45]
plt.hist(total_bill, bins=10, color='skyblue', edgecolor='black')
plt.title('Histogram of Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Frequency')
plt.show()
```

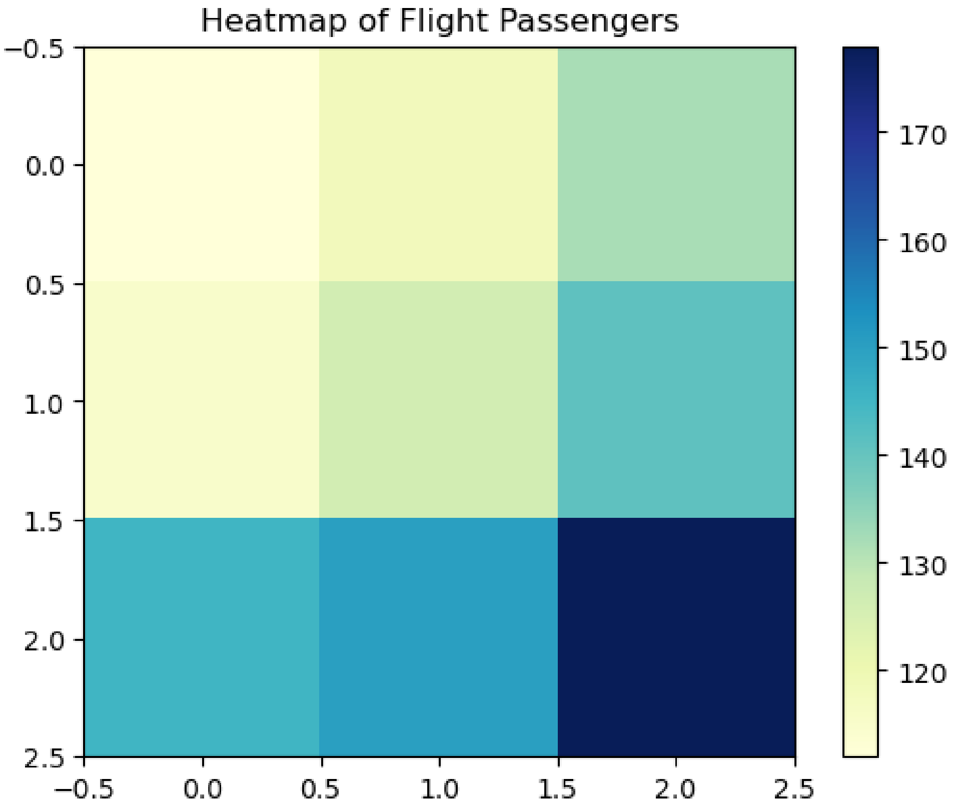


[48]:

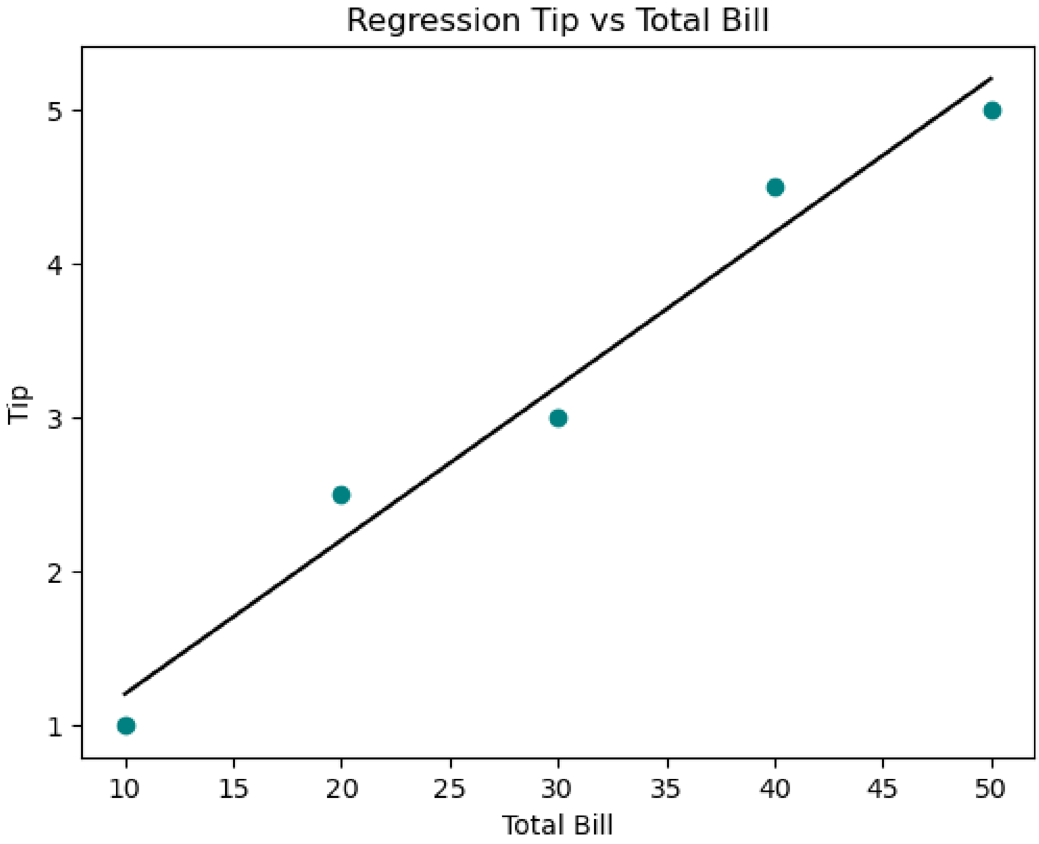
```
#A KDE plot (Kernel Density Estimate) is a smooth curve that estimates the␣
 ↪probability density of a variable.
#In this code, we use gaussian_kde() from scipy.stats to compute KDE for the␣
 ↪total_bill data.
#plt.plot() draws the smooth density curve
#plt.fill_between() shades the area under the curve (like fill=True in Seaborn).
#This plot helps us visualize where values are concentrated.
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import gaussian_kde
total_bill = [10, 12, 14, 16, 18, 20, 22, 23, 24, 25,
              26, 28, 30, 32, 34, 36, 38, 40, 42, 45]
kde = gaussian_kde(total_bill)
x = np.linspace(min(total_bill), max(total_bill), 100)
y = kde(x)
plt.plot(x, y, color='purple')
plt.fill_between(x, y, color='purple', alpha=0.3)
plt.title('KDE Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Density')
plt.show()
```



[52]:
```
#A heatmap visualizes 2D data using color intensity.
#plt.imshow() is used in Matplotlib to create a heatmap.
#Each cell's color represents the magnitude of the value.
#plt.colorbar() adds a color scale beside the heatmap.
#Useful for showing patterns, trends, and high/low values in a table-like␣
 ↪format.
#Commonly used in correlation matrices, flight data, and confusion matrices.
import matplotlib.pyplot as plt

data = [[112, 118, 132],
        [115, 126, 141],
        [145, 150, 178]]

plt.imshow(data, cmap='YlGnBu')
plt.title("Heatmap of Flight Passengers")
plt.colorbar()
plt.show()
```



[60]:
```
#A regression plot shows the relationship between two variables with a best-fit␣
↪line.
#plt.scatter() is used to plot the data points.
#np.polyfit() fits a linear regression line.
#plt.plot() draws the regression line.
#Helps to identify correlation and trend between total_bill and tip.
total_bill = np.array([10, 20, 30, 40, 50])
tip = np.array([1, 2.5, 3, 4.5, 5])
coeffs = np.polyfit(total_bill, tip, deg=1)
reg_line = np.poly1d(coeffs)
plt.scatter(total_bill, tip, color='teal')
plt.plot(total_bill, reg_line(total_bill), color='black')
plt.title('Regression Tip vs Total Bill')
plt.xlabel('Total Bill')
plt.ylabel('Tip')
plt.show()
```


Regression Tip vs Total Bill
Tip
5
4
3
2
1
10
15
20
25
30
35
40
45
50
Total Bill

```
[ ]:
```