SHADOWFOX DATA SCIENCE INTERNSHIP TASK 1 :

Visualization Library Documentation

VISUALIZATION:

Visualization refers to the graphical representation of data or information. It involves transforming raw data into visual formats like charts, graphs, maps, or plots, making it easier to identify patterns, trends, relationships, and outliers. The goal is to present data in a way that allows users to quickly grasp insights and make informed decisions.

MATPLOTLIB:

Matplotlib is a widely used data visualization library in Python that provides a flexible framework for creating static, interactive, and animated plots. It is particularly well-suited for creating 2D graphs but can also support limited 3D plotting

Matplotlib is often used for exploratory data analysis, presenting insights, and producing publication-quality figures in various formats.

Key Features of Matplotlib:

1. Wide Range of Plot Types
2. Customization
3. Integration with NumPy and Pandas
4. Interactive Plots
5. 3D Plotting Support

Installation:

You can install both Matplotlib and Seaborn using `pip` (Python's package installer). Open a terminal or command prompt and run the following commands

```
pip install matplotlib
```

Importing:

Once installed, you can import Matplotlib in your Python code like this:

```
import matplotlib.pyplot as plt
```

Graph Types:

Line Plot:

Description: Displays data points connected by lines, showing trends over time or continuous data.

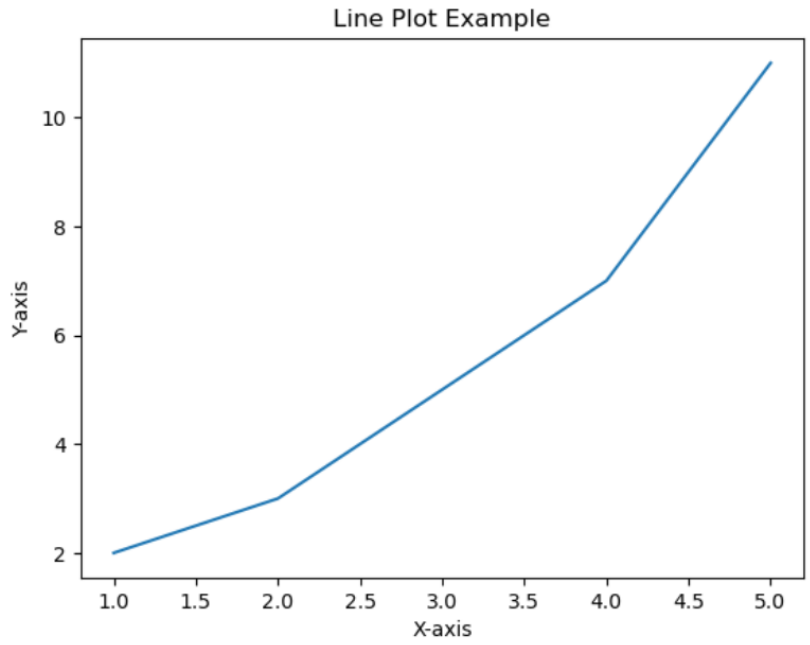
Use Case: Analysing trends in stock prices over time.

Code Snippet:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
plt.plot(x, y)
plt.title("Line Plot Example")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Output:



Bar Chart:

Description: Represents categorical data with rectangular bars.

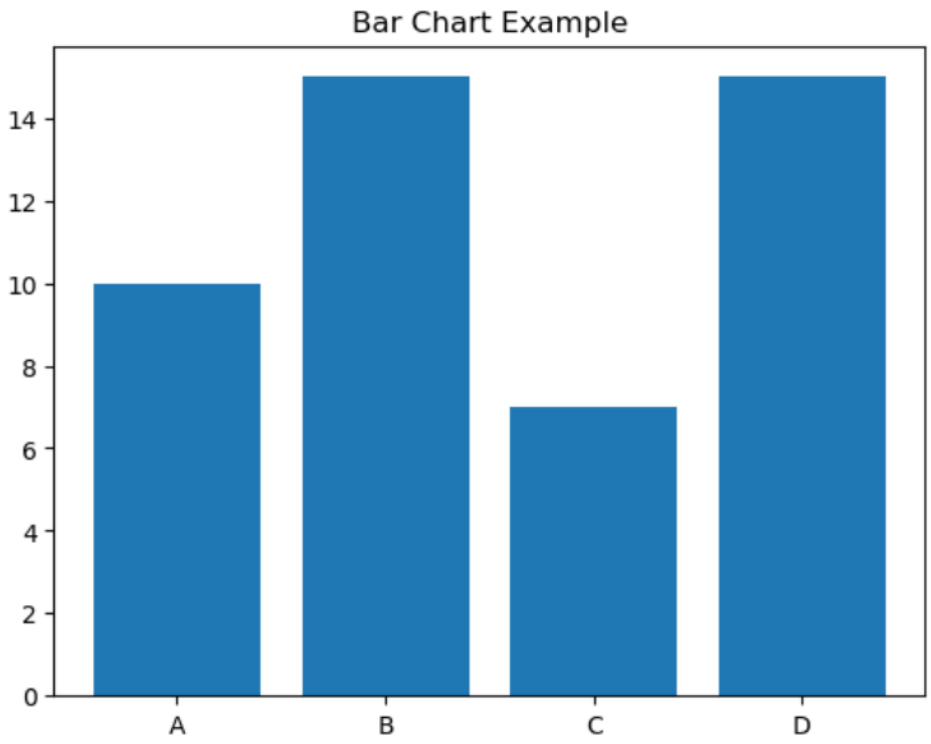
Use Case: Comparing sales data across different regions.

Code Snippet:

```
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C','D']
values = [10, 15, 7, 15]
plt.bar(categories, values)
plt.title("Bar Chart Example")
plt.show()
```

Output:



Histogram

Description: Displays the distribution of numerical data by dividing it into bins.

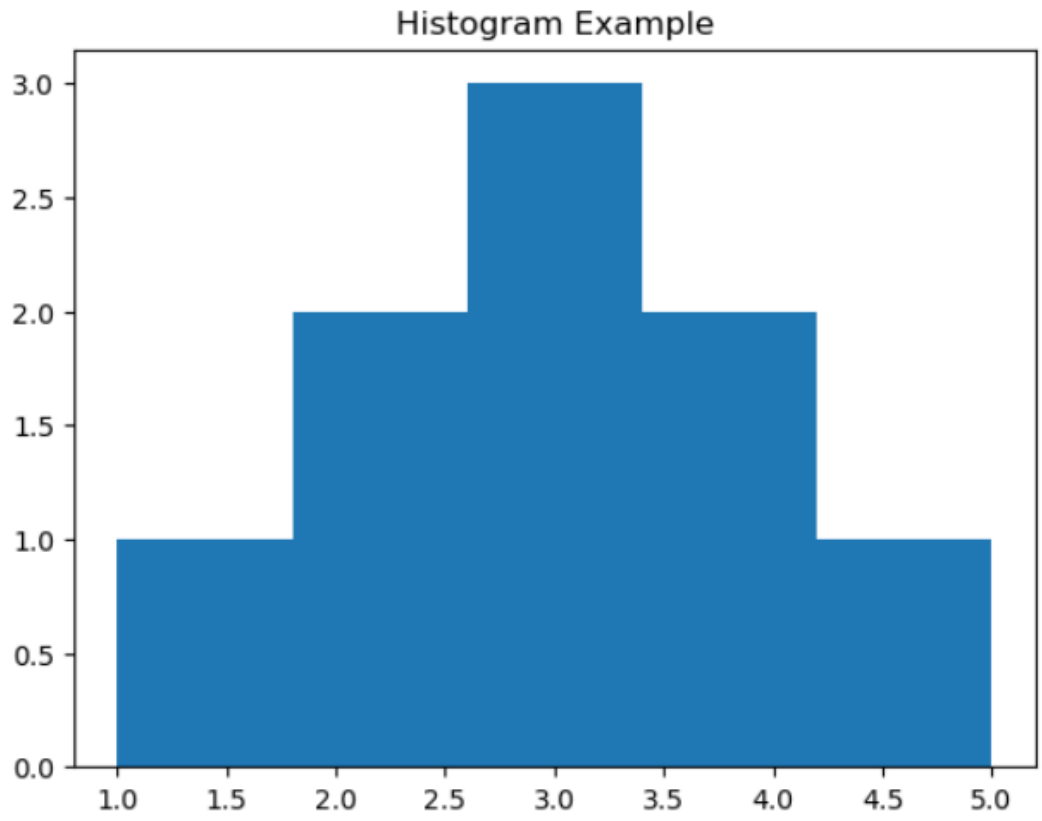
Use Case: Understanding the distribution of exam scores.

Code Snippet:

```
import matplotlib.pyplot as plt

data = [1, 2, 2, 3, 3, 3, 4, 4, 5]
plt.hist(data, bins=5)
plt.title("Histogram Example")
plt.show()
```

Output:



Scatter Plot

Description: Displays relationships between variables with enhanced aesthetics and options for color coding.

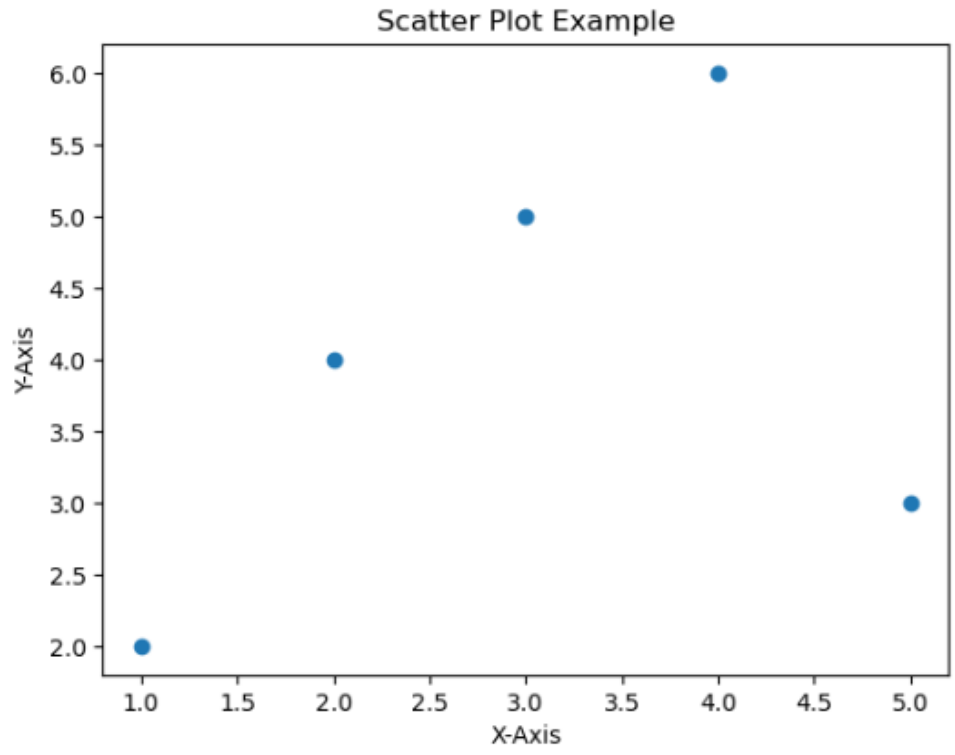
Use Case: Analysing the relationship between two continuous variables.

Code Snippet:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 4, 5, 6, 3]
plt.scatter(x, y)
plt.title("Scatter Plot Example")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.show()
```

Output:



Pie Chart

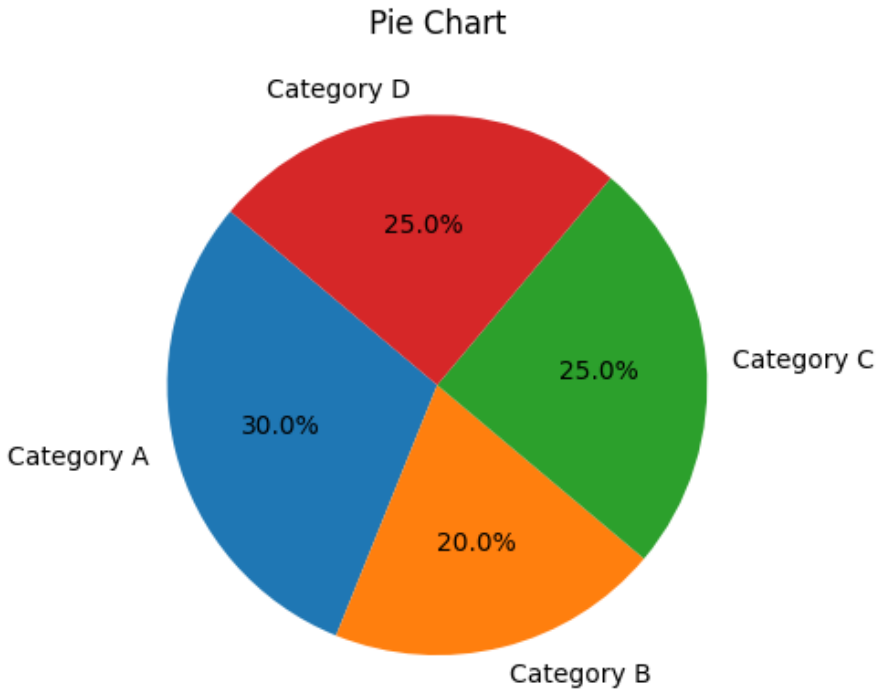
Description: Represents data as slices of a circle, showing proportions of a whole.

Use Case: Visualizing market share of different companies.

Code Snippet:

```
import matplotlib.pyplot as plt
labels = ['Category A', 'Category B', 'Category C', 'Category D']
sizes = [30, 20, 25, 25]
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=140)
plt.title('Pie Chart')
plt.show()
```

Output:



3D Plot

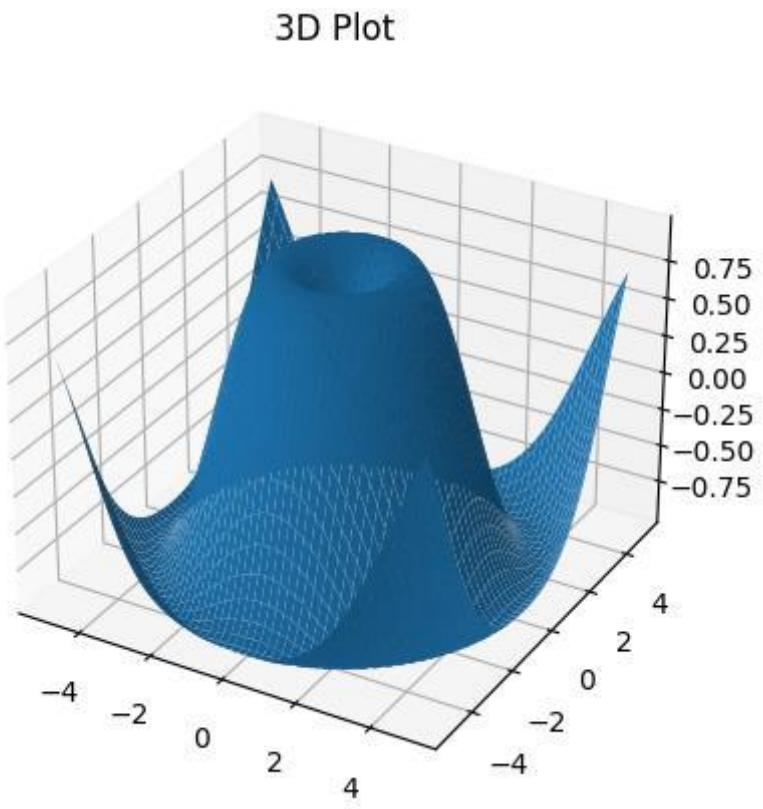
Description: Matplotlib provides 3D plotting capabilities to visualize data with three dimensions. The Axes3D module allows for plotting 3D scatter plots, surface plots, and wireframes.

Use Case: 3D plots are used to visualize three-dimensional data, allowing for the representation of functions or datasets in three dimensions. They are useful for understanding the relationships between variables in three-dimensional space and identifying patterns or trends.

Code Snippet:

```
x = np.linspace(-5, 5, 100)
y = np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = np.sin(np.sqrt(X**2 + Y**2))
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z)
plt.title('3D Plot')
plt.show()
```

Output:



Heatmap:

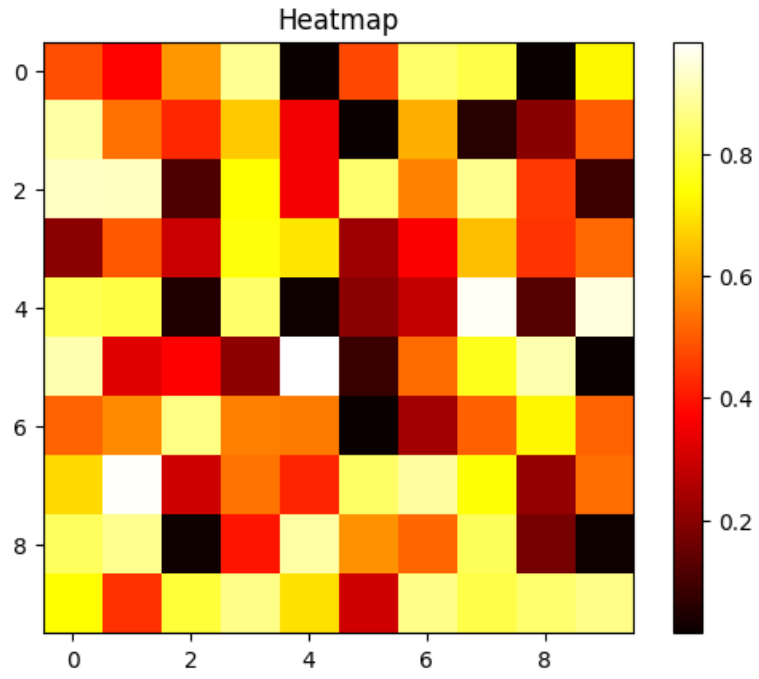
Description: A heatmap is a 2D graphical representation of data where individual values are represented as colors. It's useful for visualizing matrix data or correlation between variables.

Use Case: Heatmaps are used to represent data values as colors in a matrix. They provide a visual representation of the magnitude or intensity of values in a two-dimensional space. Heatmaps are useful for visualizing correlation matrices, confusion matrices, and other tabular data.

Code Snippet:

```
import matplotlib.pyplot as plt
import numpy as np
data = np.random.rand(10, 10)
plt.imshow(data, cmap='hot', interpolation='nearest')
plt.title('Heatmap')
plt.colorbar()
plt.show()
```

Output:



Area Plot:

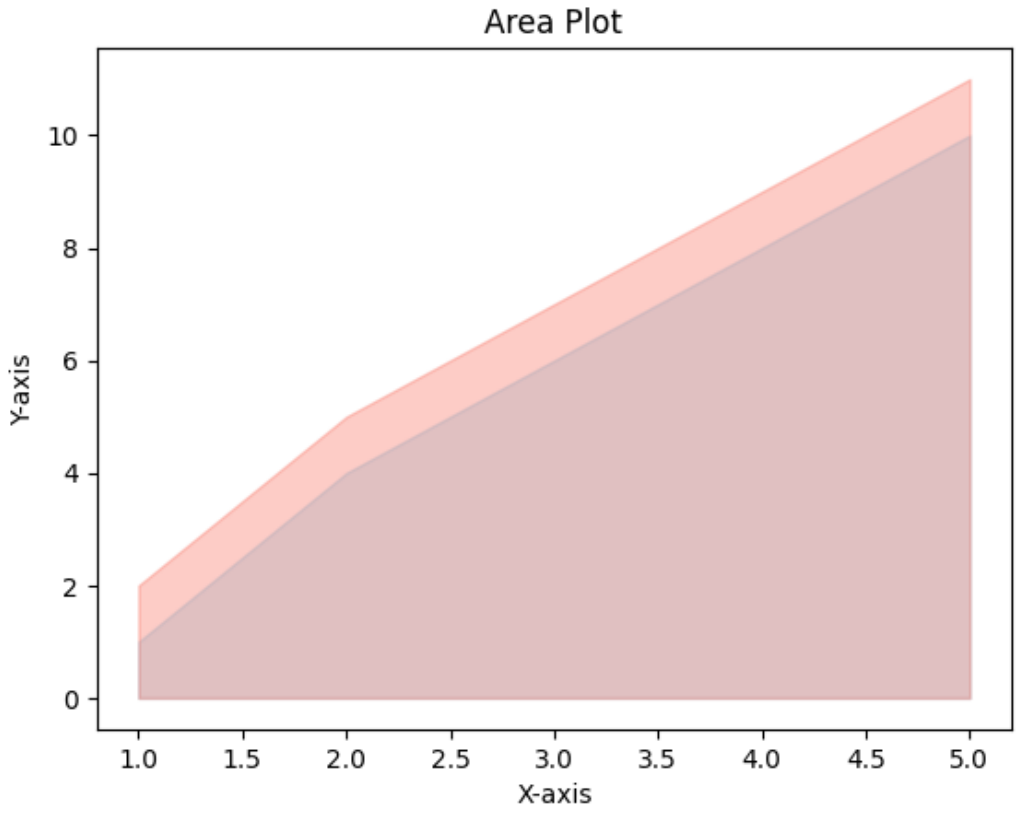
Description: An area plot is similar to a line plot but with the area under the line filled. It's useful for visualizing cumulative data or comparing magnitudes over time.

Use Case: Area plots, also known as filled line plots, are used to represent data with continuous values as filled areas between the data points and the x-axis. They are useful for visualizing cumulative data or highlighting the magnitude of changes over time.

Code Snippet:

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y1 = [1, 4, 6, 8, 10]
y2 = [2, 5, 7, 9, 11]
plt.fill_between(x, y1, color="skyblue", alpha=0.4)
plt.fill_between(x, y2, color="salmon", alpha=0.4)
plt.title('Area Plot')
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.show()
```

Output:



Stem Plot:

Description: A stem plot displays data points with lines extending from a baseline, often used to emphasize individual values in time series data.

Use Case: Stem plots, also known as stem-and-leaf plots, are used to represent numerical data in a visually appealing and compact format. They are particularly useful for visualizing the distribution of small datasets and identifying patterns or outliers.

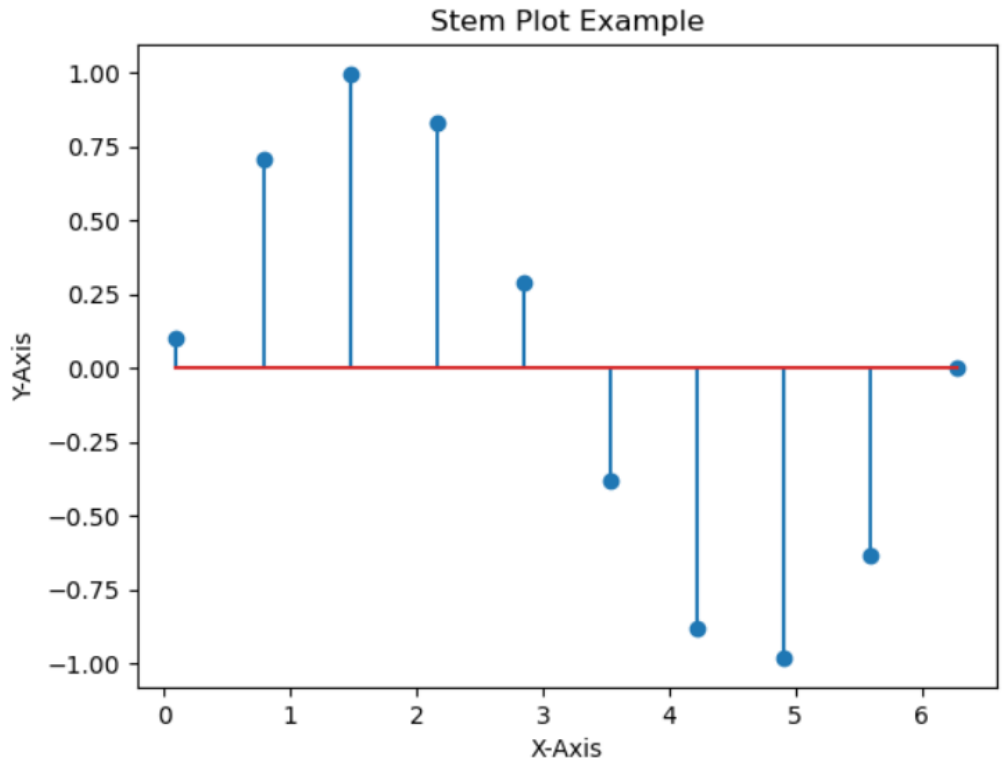
Code Snippet:

```
import matplotlib.pyplot as plt
import numpy as np

# Sample data
x = np.linspace(0.1, 2 * np.pi, 10)
y = np.sin(x)

# Create stem plot
plt.stem(x, y)
plt.title("Stem Plot Example")
plt.xlabel("X-Axis")
plt.ylabel("Y-Axis")
plt.show()
```

Output:



SEABORN:

Seaborn is a powerful Python data visualization library built on top of Matplotlib that simplifies creating attractive and informative statistical graphics.

t provides a high-level interface for drawing graphs, making complex visualizations easier to generate with fewer lines of code.

Key Features of Seaborn:

1. Built on Matplotlib
2. Integrated with Pandas
3. Advanced Statistical Plots
4. Automatic Aesthetics
5. Faceting and Grid-based Plotting

Installation:

You can install both Matplotlib and Seaborn using `pip` (Python's package installer). Open a terminal or command prompt and run the following commands

```
pip install seaborn
```

Importing:

Once installed, you can import Matplotlib in your Python code like this:

```
import seaborn as sns
```

Graph Types:

Line Plot:

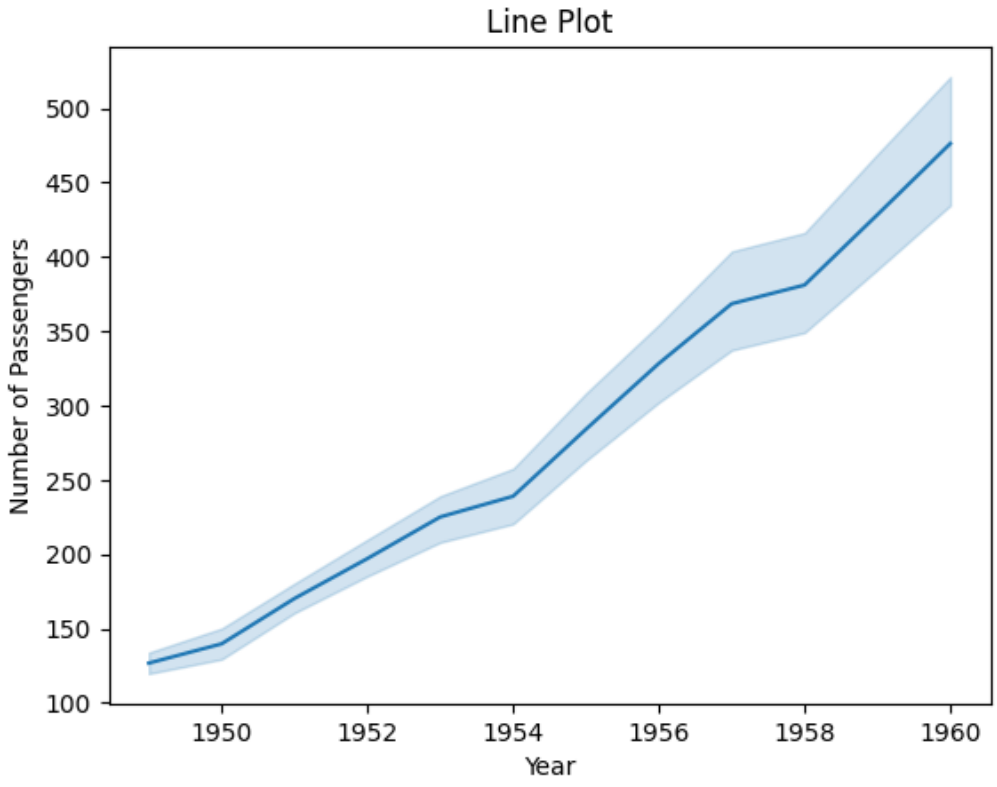
Description: A line plot displays data points connected by lines, showing trends over a continuous variable (usually time). Seaborn's line plot comes with built-in features like confidence intervals.

Use Case: Used for visualizing trends over time, such as stock prices, temperature changes, or sales over a period.

Code Snippet:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset('flights')
sns.lineplot(x='year', y='passengers', data=data)
plt.title('Line Plot')
plt.xlabel('Year')
plt.ylabel('Number of Passengers')
plt.show()
```

Output:



Bar Plot:

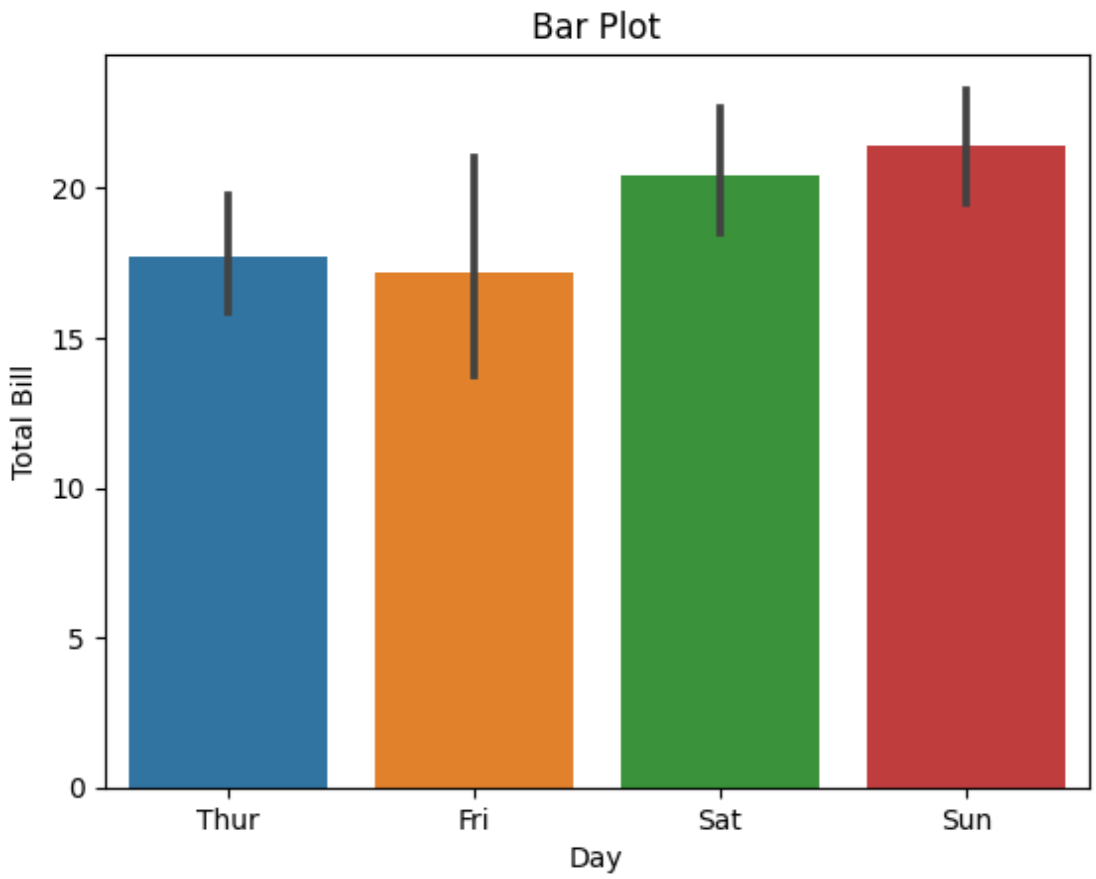
Description: A bar plot shows the relationship between a categorical variable and a continuous variable, with rectangular bars representing the data.

Use Case: Commonly used for comparing quantities across different categories, such as sales across regions or average performance across groups.

Code Snippet:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset('tips')
sns.barplot(x='day', y='total_bill', data=data)
plt.title('Bar Plot')
plt.xlabel('Day')
plt.ylabel('Total Bill')
plt.show()
```

Output:



Histogram

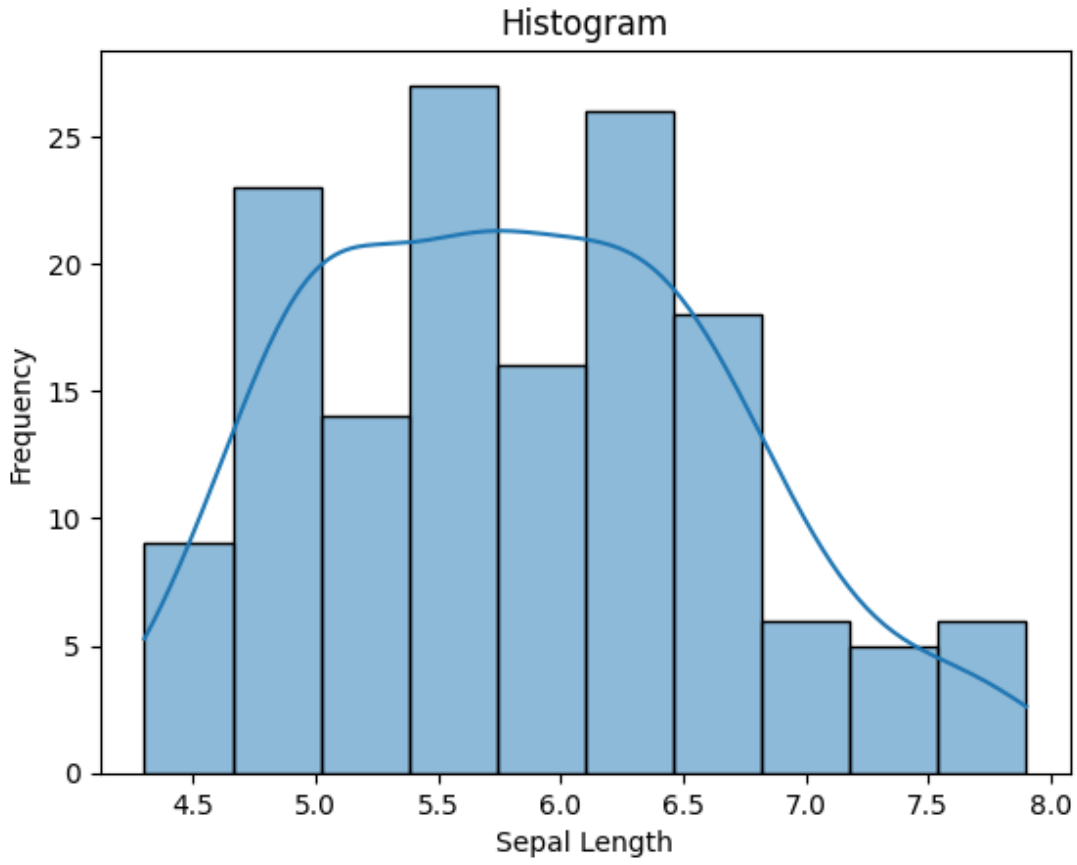
Description: A **h**istogram is a plot that shows the distribution of a numerical variable by dividing the data into bins and displaying the count or frequency for each bin

Use Case: Used to understand the distribution of data, such as the distribution of age, income, or scores.

Code Snippet:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset('iris')
sns.histplot(data['sepal_length'], bins=10, kde=True)
plt.title('Histogram')
plt.xlabel('Sepal Length')
plt.ylabel('Frequency')
plt.show()
```

Output:



Scatter Plot:

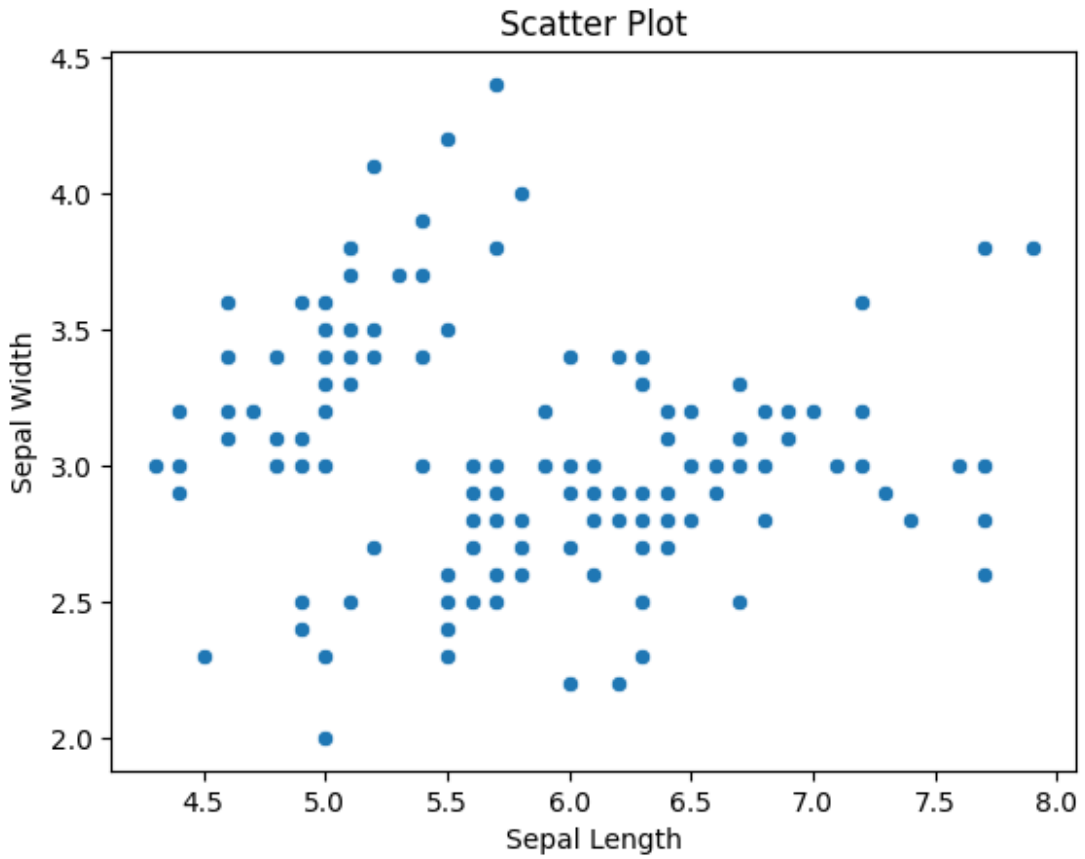
Description: A scatter plot visualizes the relationship between two continuous variables, where each data point represents an observation in the dataset.

Use Case: Used to analyse the correlation or relationship between two variables, such as height vs. weight, or age vs. income.

Code Snippet:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset('iris')
sns.scatterplot(x='sepal_length', y='sepal_width', data=data)
plt.title('Scatter Plot')
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.show()
```

Output:



Heatmap:

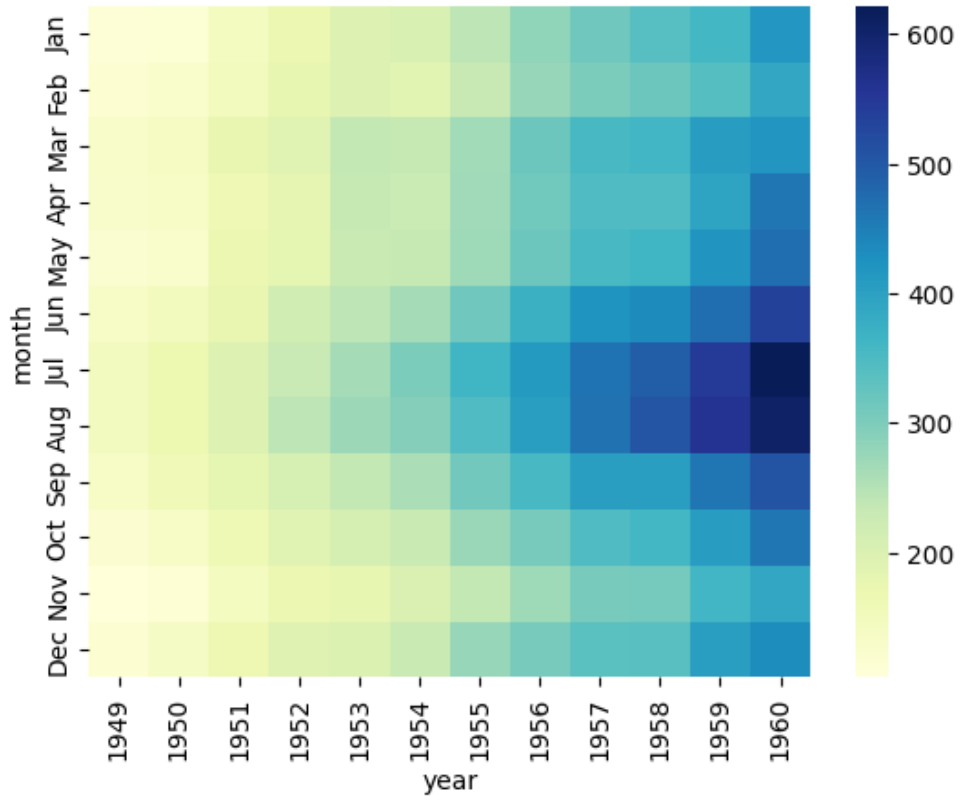
Description: A heatmap is a 2D visualization where individual values in a matrix are represented as colors. It's excellent for visualizing relationships in matrix data or correlations between variables.

Use Case: Used in machine learning to display correlation matrices, confusion matrices, or to visualize any tabular data (e.g., visualizing a pivot table, population density, etc.).

Code Snippet:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset('flights')
pivot_data = data.pivot_table(index='month', columns='year', values='passengers')
sns.heatmap(pivot_data, cmap='YlGnBu', annot=True, fmt='d')
plt.title('Heatmap')
plt.xlabel('Year')
plt.ylabel('Month')
plt.show()
```

Output:



Box Plot

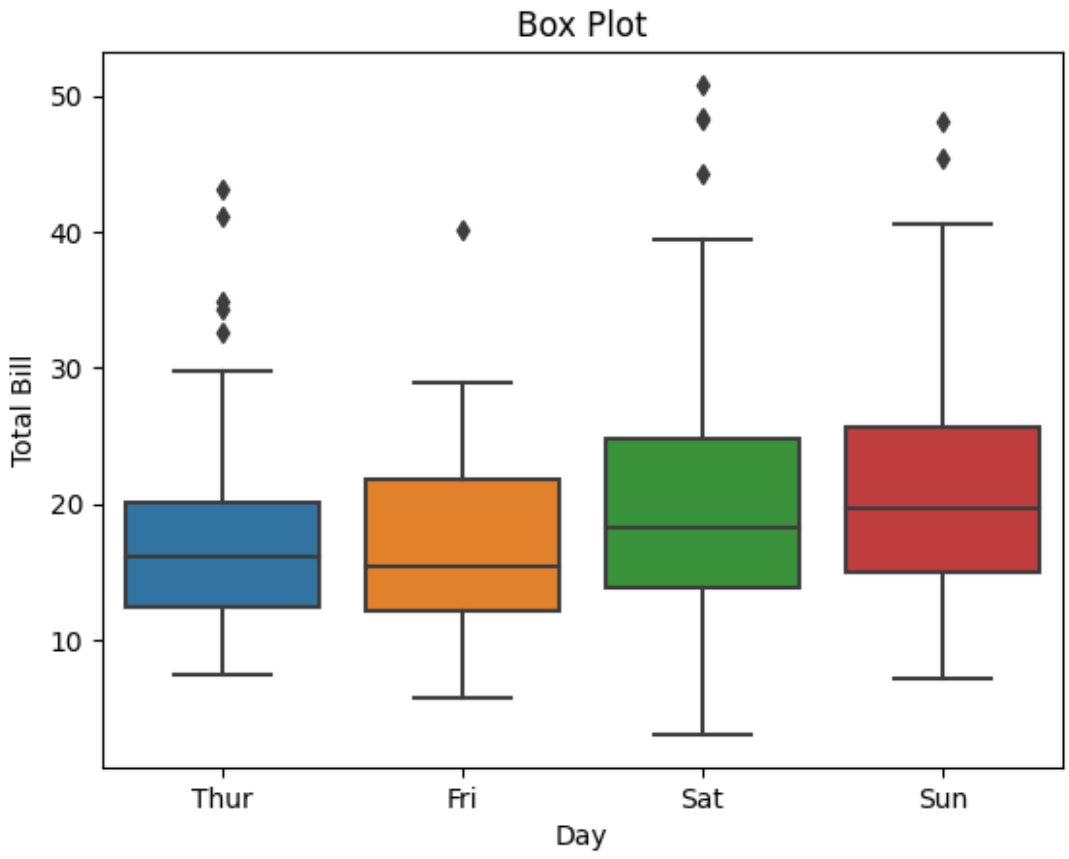
Description: A box plot shows the distribution of a continuous variable through five summary statistics: minimum, first quartile (Q1), median (Q2), third quartile (Q3), and maximum.

Use Case: Used to compare the distribution of multiple groups or detect outliers in the data.

Code Snippet:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset('tips')
sns.boxplot(x='day', y='total_bill', data=data)
plt.title('Box Plot')
plt.xlabel('Day')
plt.ylabel('Total Bill')
plt.show()
```

Output:



Violin Plot:

Description: A violin plot is similar to a box plot but also shows the kernel density estimate of the underlying data distribution, providing more insight into the data's distribution shape.

Use Case: Ideal for visualizing the distribution of the data while comparing multiple groups.

Code Snippet:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset('tips')
sns.violinplot(x='day', y='total_bill', data=data)
plt.title('Violin Plot')
plt.xlabel('Day')
plt.ylabel('Total Bill')
plt.show()
```

Output:

Point Plot:

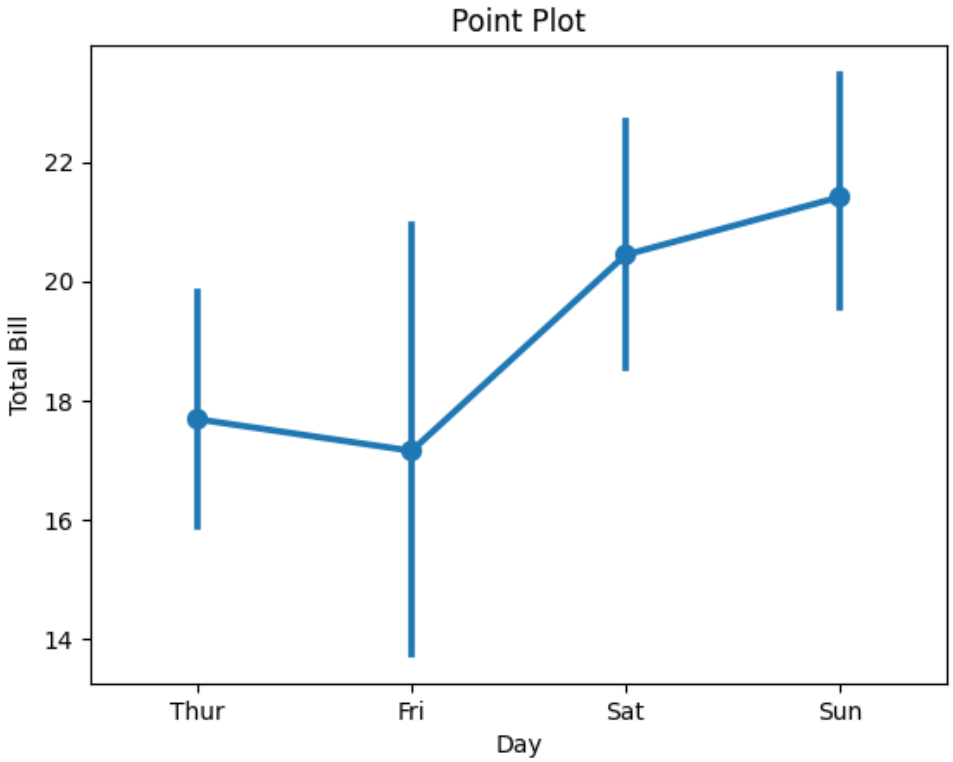
Description: A point plot in Seaborn displays the mean of a numerical variable for different categories, with points connected by lines and optional error bars representing confidence intervals or standard deviations. It is ideal for comparing trends across categorical variables and understanding interactions between groups.

Use Case: Comparing average values (e.g., sales, scores) across categories like time periods, regions, or groups. Visualizing changes and trends in grouped data, such as tracking performance differences between categories

Code Snippet:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset('tips')
sns.pointplot(x='day', y='total_bill', data=data)
plt.title('Point Plot')
plt.xlabel('Day')
plt.ylabel('Total Bill')
plt.show()
```

Output:



Swarm Plot:

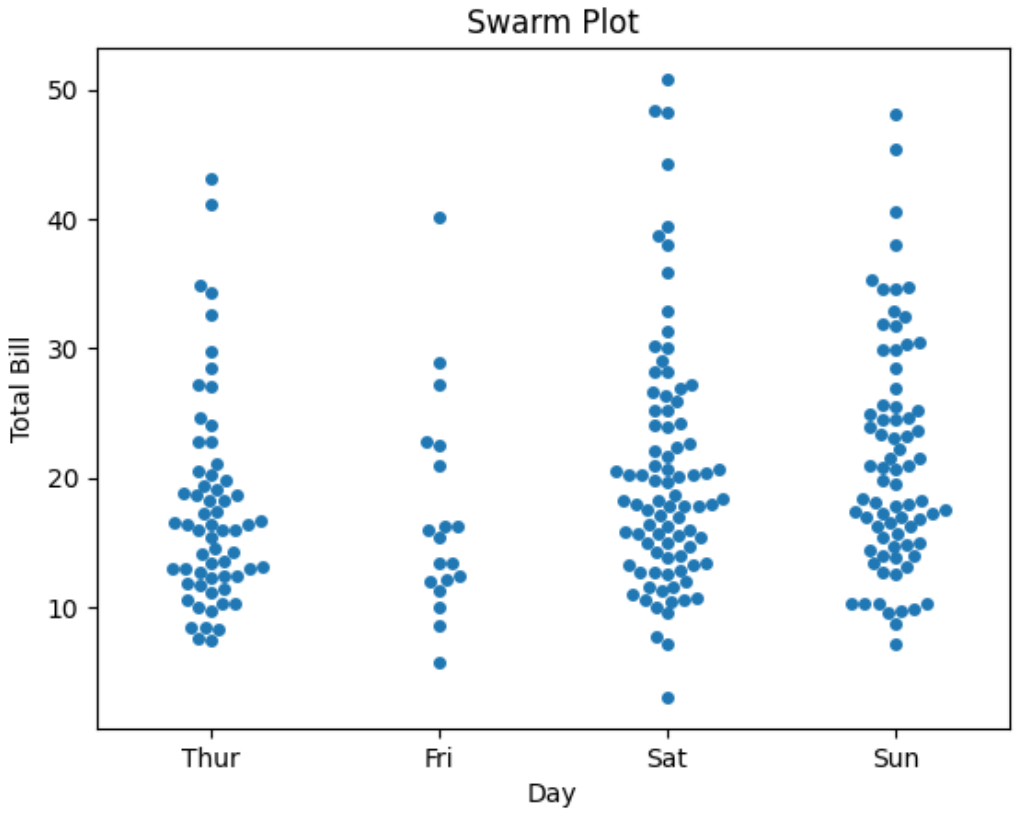
Description: A swarm plot is a categorical scatter plot where data points are plotted along a categorical axis, and the points are spread out (or "swarmed") to avoid overlap. It provides a good sense of the distribution of the data while preserving individual data points.

Use Case: Ideal for displaying the distribution of data across categories while showing each individual observation. Useful for identifying outliers and variations within categories, such as displaying the distribution of scores or measurements across different groups.

Code Snippet:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset('tips')
sns.swarmplot(x='day', y='total_bill', data=data)
plt.title('Swarm Plot')
plt.xlabel('Day')
plt.ylabel('Total Bill')
plt.show()
```

Output:



Comparison Between Matplotlib and Seaborn.

Ease of Use

- Matplotlib: While powerful, it often requires more lines of code to achieve aesthetically pleasing results, especially for complex visualizations.
- Seaborn: Offers a more straightforward and user-friendly interface, making it easier for users to create complex statistical visualizations with minimal code.

Customization Options

- Matplotlib: Provides extensive customization capabilities for every aspect of a plot, allowing for high flexibility.
- Seaborn: Focuses on providing aesthetically pleasing defaults, but it may not offer as many low-level customization options as Matplotlib.

Interactivity

- Matplotlib: Offers basic interactivity in Jupyter Notebooks, but its primary focus is on static plots.
- Seaborn: While it can produce interactive plots when used in combination with other libraries (like Plotly), its primary strength lies in static visualizations.

Performance with Large Datasets

- Matplotlib: Generally handles large datasets well, but performance can decrease with highly complex plots.
- Seaborn: Built on top of Matplotlib, it inherits similar performance characteristics. However, for extremely large datasets, users may need to be cautious about the complexity of visualizations and the number of data points displayed.