Shadow fox

Beginner Task

NAME:-Kalluri Harshith

1)Matplot library:-

Overview

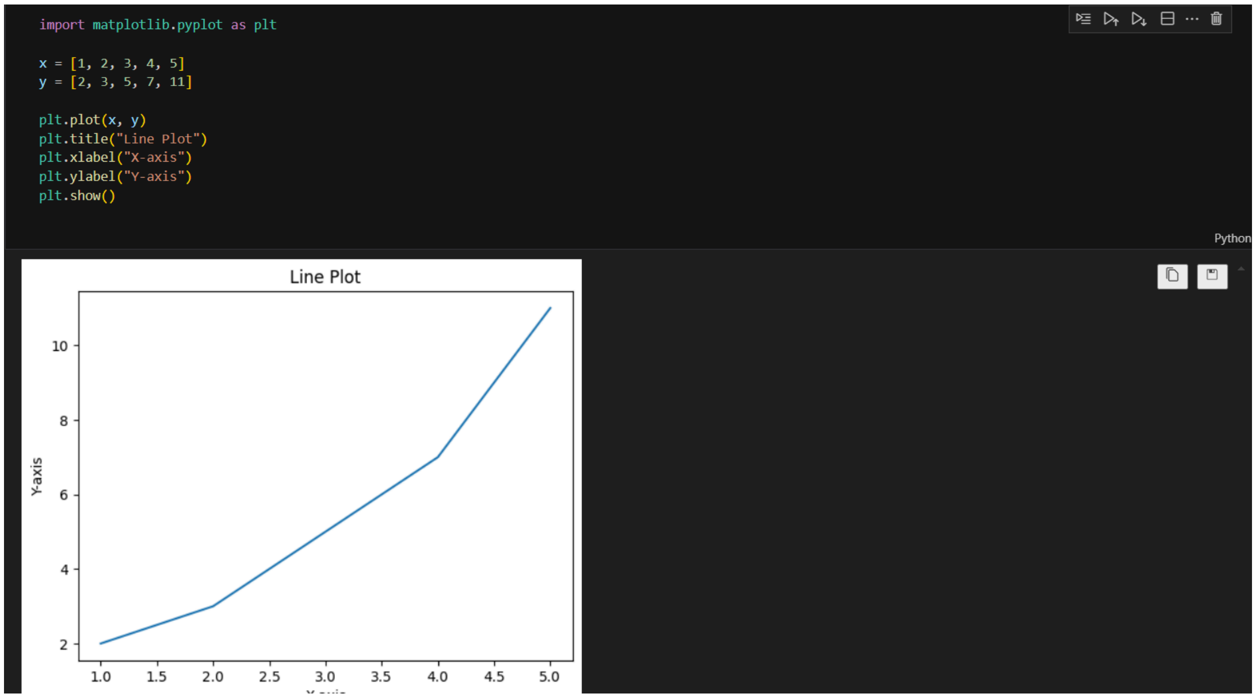
Matplotlib is a comprehensive library in Python for creating static, animated, and interactive visualizations. It is particularly effective for producing publication-quality figures and has extensive support for various plot types.

Key Features of Matplotlib:
1)versatile plotting
2)coustamisation
3)interactive plots
4)integration
5)export options
6)subplots and layouts

<u>Graph Types</u>

i) Line Plot

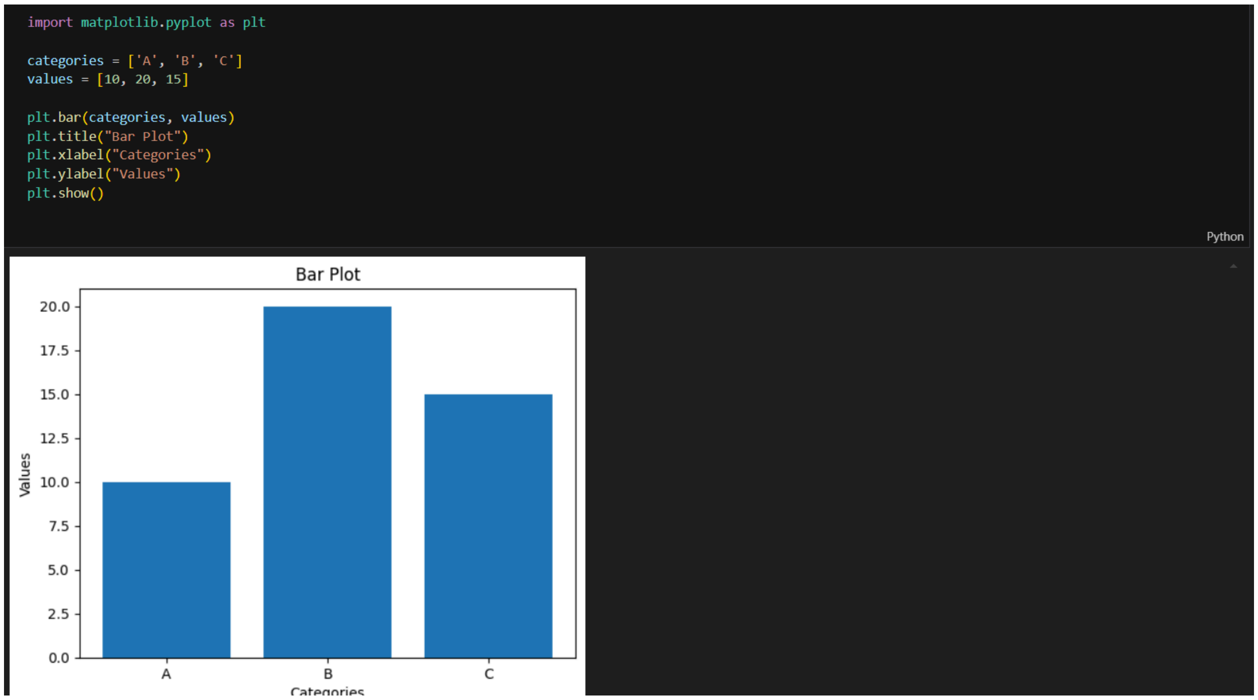
Line plots are one of the most commonly used types of plots in data visualization. They are particularly useful for showing trends over time or relationships between variables. Here's an overview of how to create and customize line plots using Matplotlib.



```python
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")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

ii) Bar charts

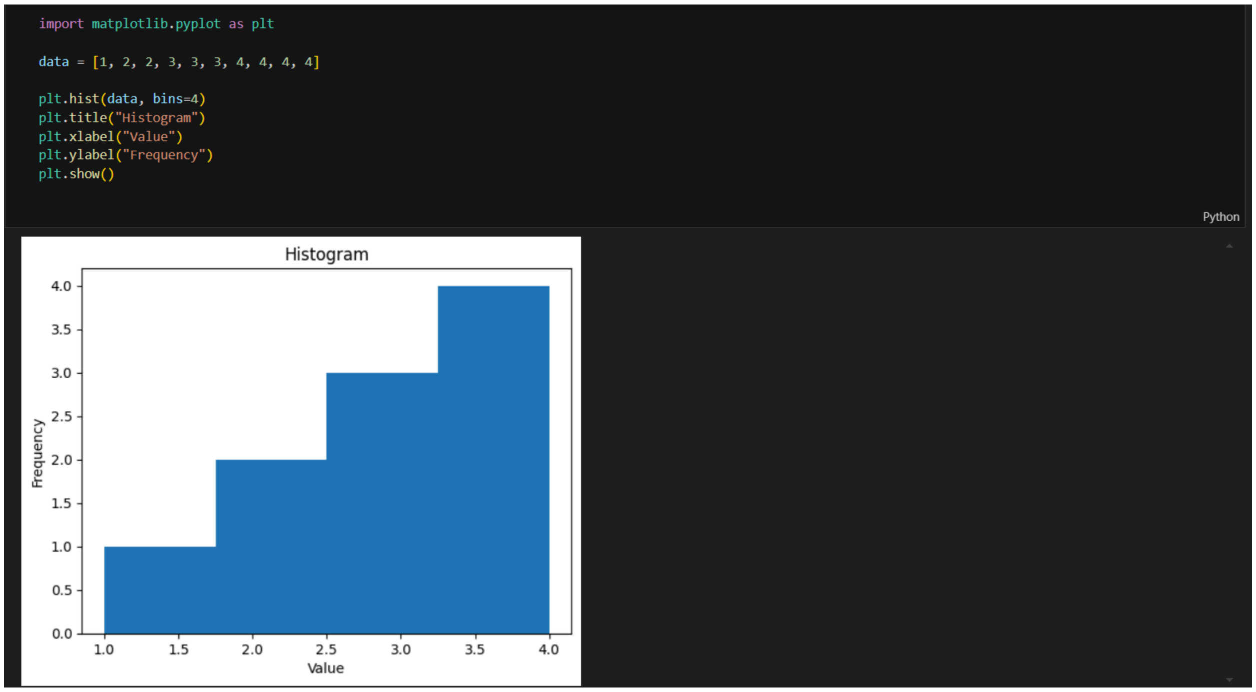
Bar plots are used to display and compare the number, frequency, or other measures for different discrete categories. They are useful for visualizing categorical data.



```python
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C']
values = [10, 20, 15]

plt.bar(categories, values)
plt.title("Bar Plot")
plt.xlabel("Categories")
plt.ylabel("Values")
plt.show()
```

iii)Histogram

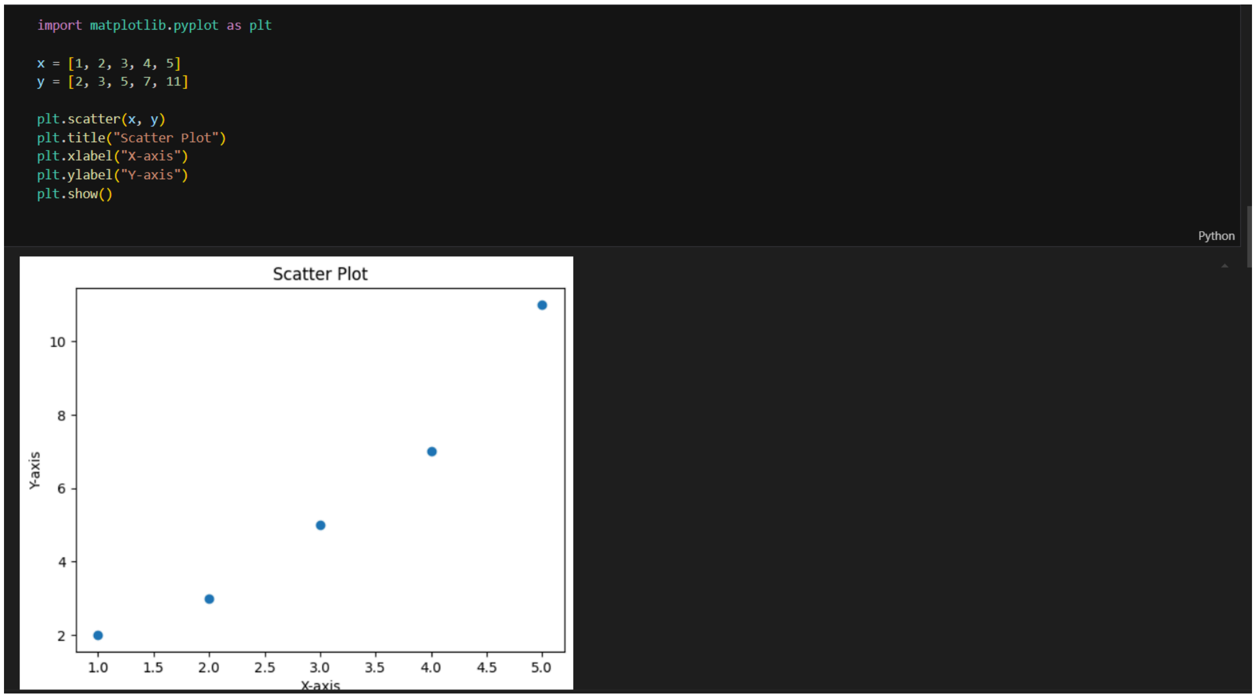
Histograms are used to represent the distribution of a dataset by dividing the data into bins (intervals) and counting the number of observations in each bin. They are useful for visualizing the underlying frequency distribution (shape) of a set of continuous data.



```python
import matplotlib.pyplot as plt

data = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]

plt.hist(data, bins=4)
plt.title("Histogram")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.show()
```

iv)Scatter plot

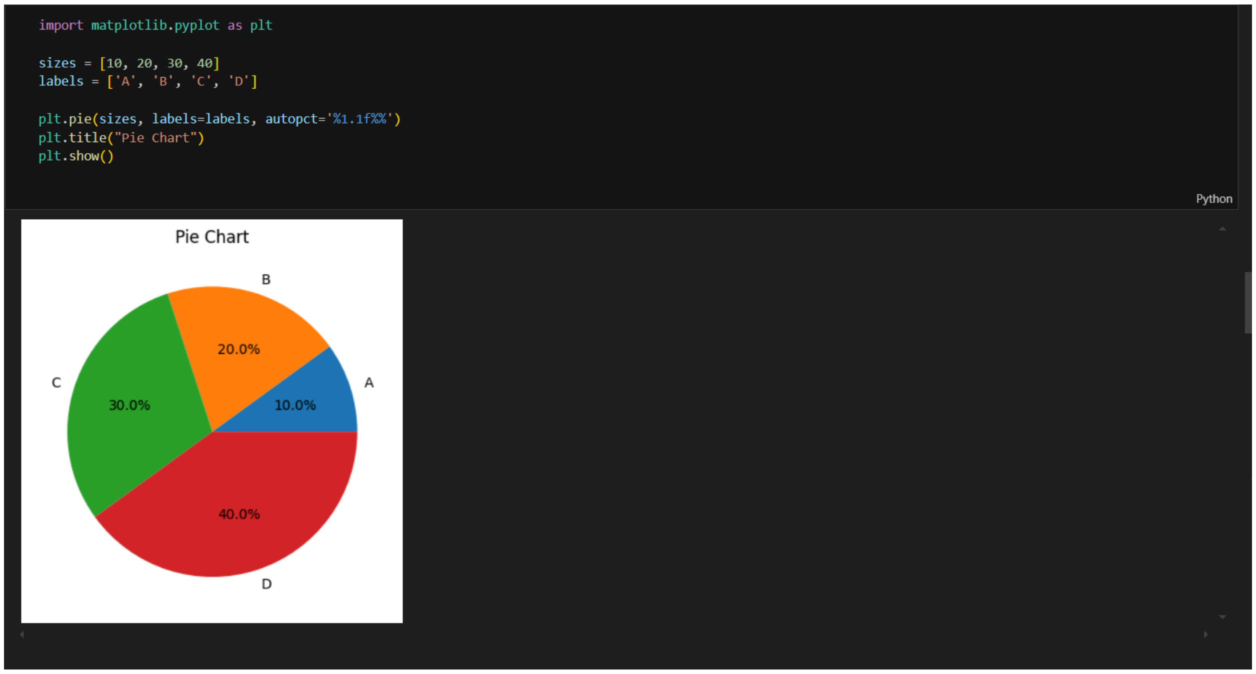
Scatter plots are used to display values for typically two variables for a set of data. They are useful for identifying relationships, correlations, and distributions between the two variables.



```python
import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]

plt.scatter(x, y)
plt.title("Scatter Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

v)Pie chart

Pie charts are used to represent the proportions of different categories in a dataset, displayed as slices of a circle. Each slice represents a category and its size is proportional to its percentage of the whole.



```python
import matplotlib.pyplot as plt

sizes = [10, 20, 30, 40]
labels = ['A', 'B', 'C', 'D']

plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title("Pie Chart")
plt.show()
```

2)Seaborn library:-

<u>Overview</u>

Seaborn is a Python visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. It is particularly well-suited for visualizing complex datasets and making statistical plots.

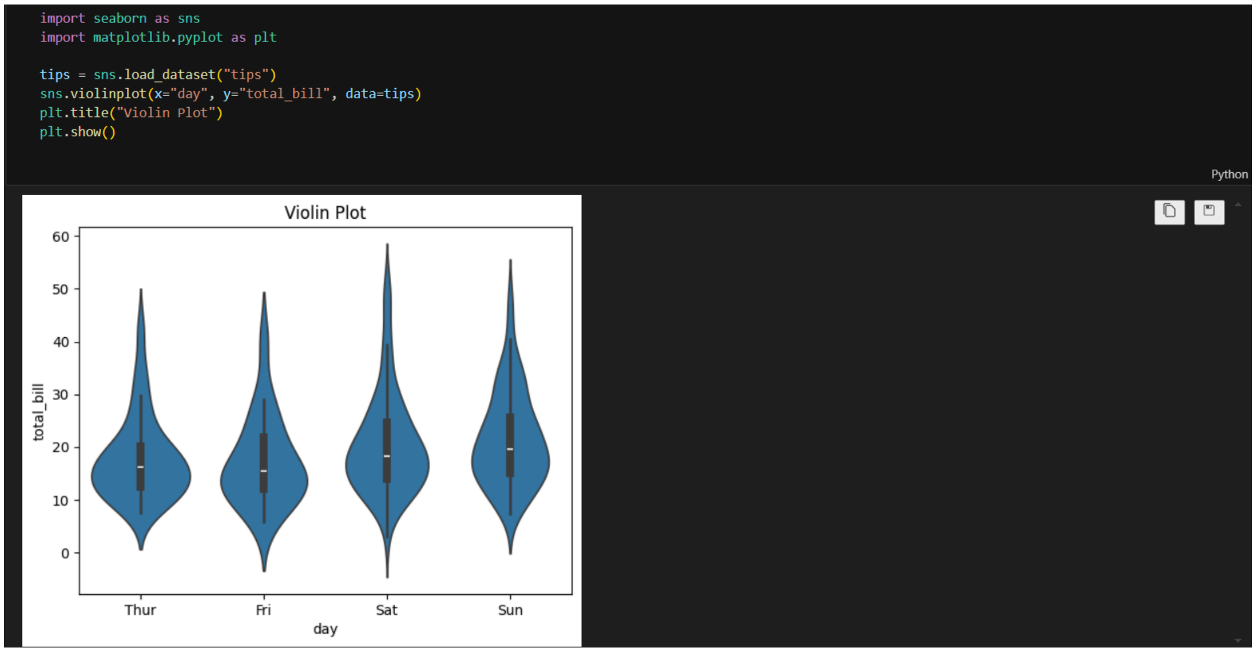
Key Features

1. **High-Level Interface: Simplifies the creation of complex visualizations with less code.**
2. **Statistical Visualization: Includes built-in themes and color palettes to make statistical plots more attractive.**
3. **Integration with Pandas: Works seamlessly with pandas DataFrames, making it easy to visualize data directly from pandas.**
4. **Rich Visualization Options: Supports a variety of plot types and customizations.**

<u>Graph Types</u>

i)viloin plot

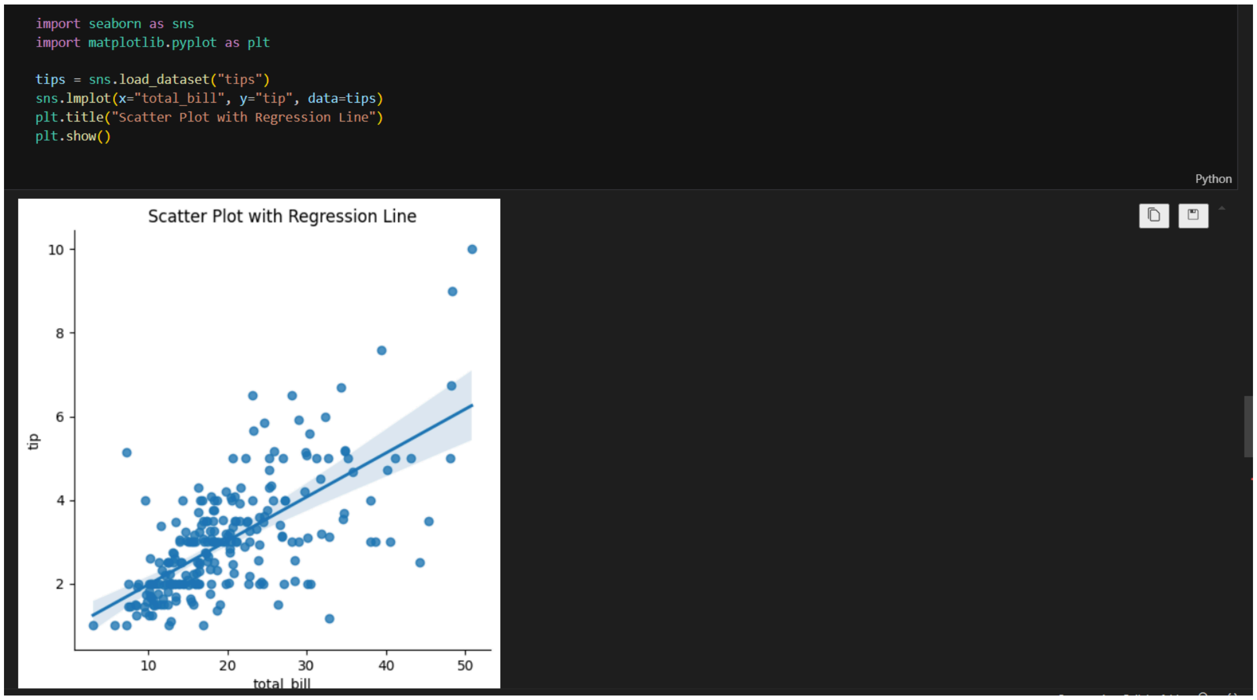
A violin plot is a method of plotting numeric data and can be seen as a combination of a box plot and a kernel density plot. It shows the distribution of the data across different categories. Violin plots are useful for comparing the distribution of multiple categories against each other.



```python
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset("tips")
sns.violinplot(x="day", y="total_bill", data=tips)
plt.title("Violin Plot")
plt.show()
```

ii) scatter plot with regression line

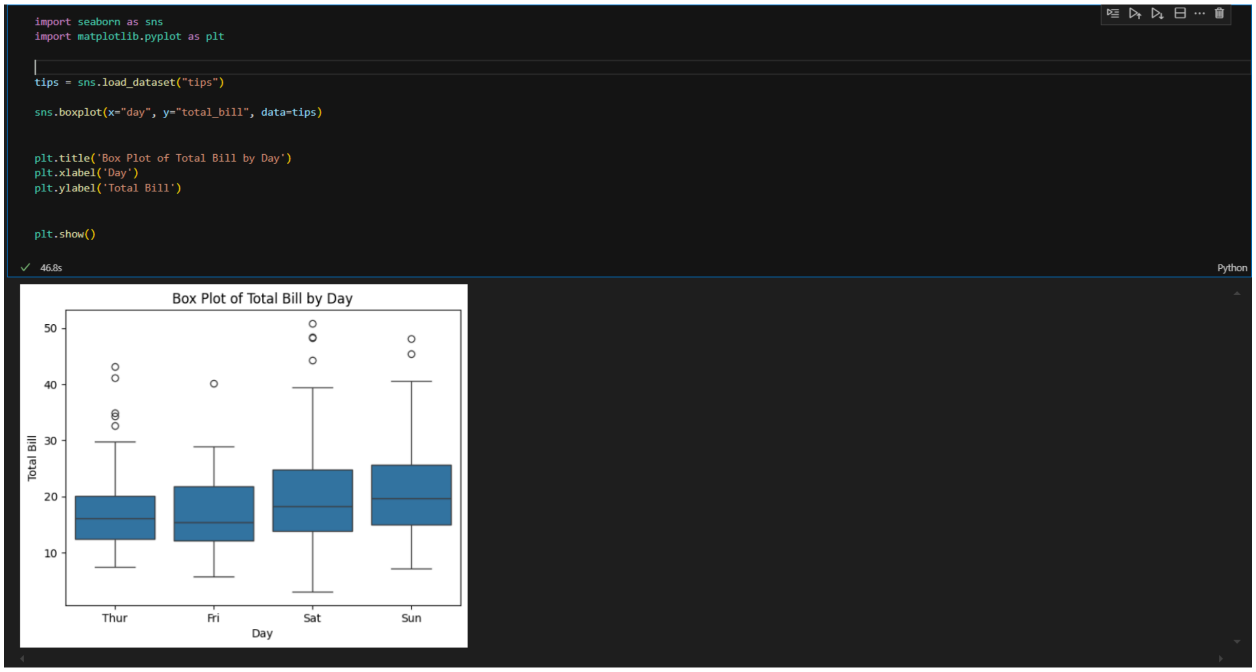
A scatter plot with a regression line is a useful way to visualize the relationship between two variables along with the trend indicated by a regression line. Seaborn makes it easy to create such plots using the `regplot()` and `lmplot()` functions, which automatically fit and plot the regression line along with the scatter plot.



```python
import seaborn as sns
import matplotlib.pyplot as plt

tips = sns.load_dataset("tips")
sns.lmplot(x="total_bill", y="tip", data=tips)
plt.title("Scatter Plot with Regression Line")
plt.show()
```

iii)Box Plot

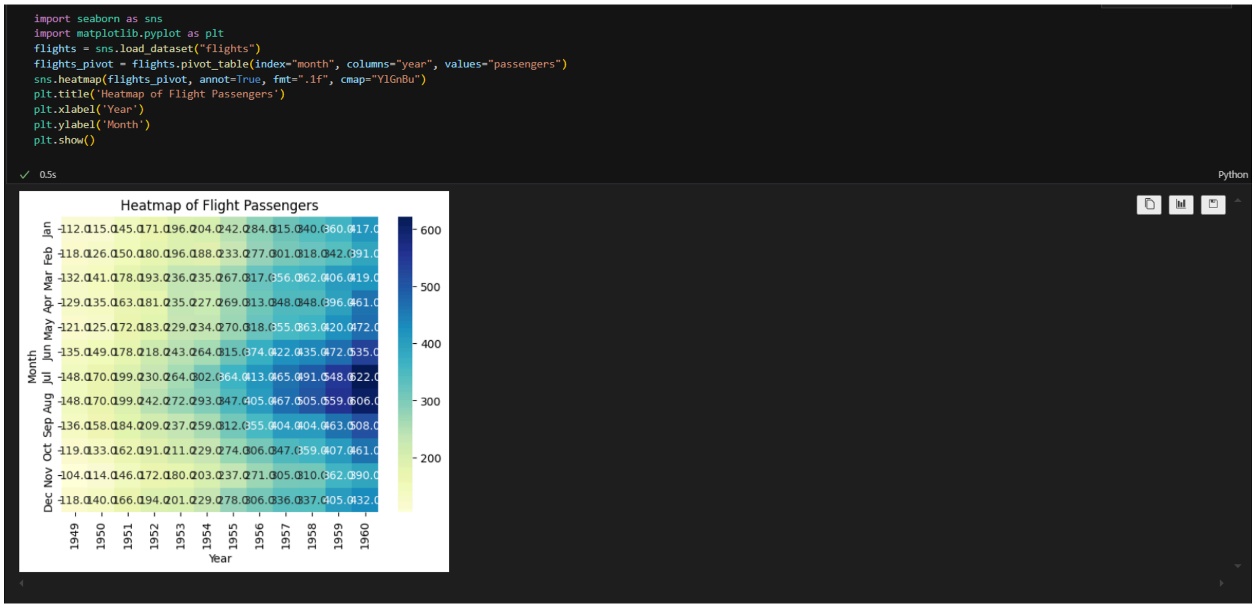
A box plot (or box-and-whisker plot) is a graphical representation of the distribution of numerical data through quartiles. It displays key statistical measures such as the median, quartiles, and potential outliers in a concise manner. Box plots are useful for comparing distributions across categories or groups.



```python
import seaborn as sns
import matplotlib.pyplot as plt


tips = sns.load_dataset("tips")

sns.boxplot(x="day", y="total_bill", data=tips)


plt.title('Box Plot of Total Bill by Day')
plt.xlabel('Day')
plt.ylabel('Total Bill')


plt.show()
```

iv)Heat map

A heatmap is a graphical representation of data where the individual values contained in a matrix are represented as colors. Heatmaps are particularly useful for visualizing the magnitude of data at a glance, making them ideal for exploring and understanding patterns in large datasets.

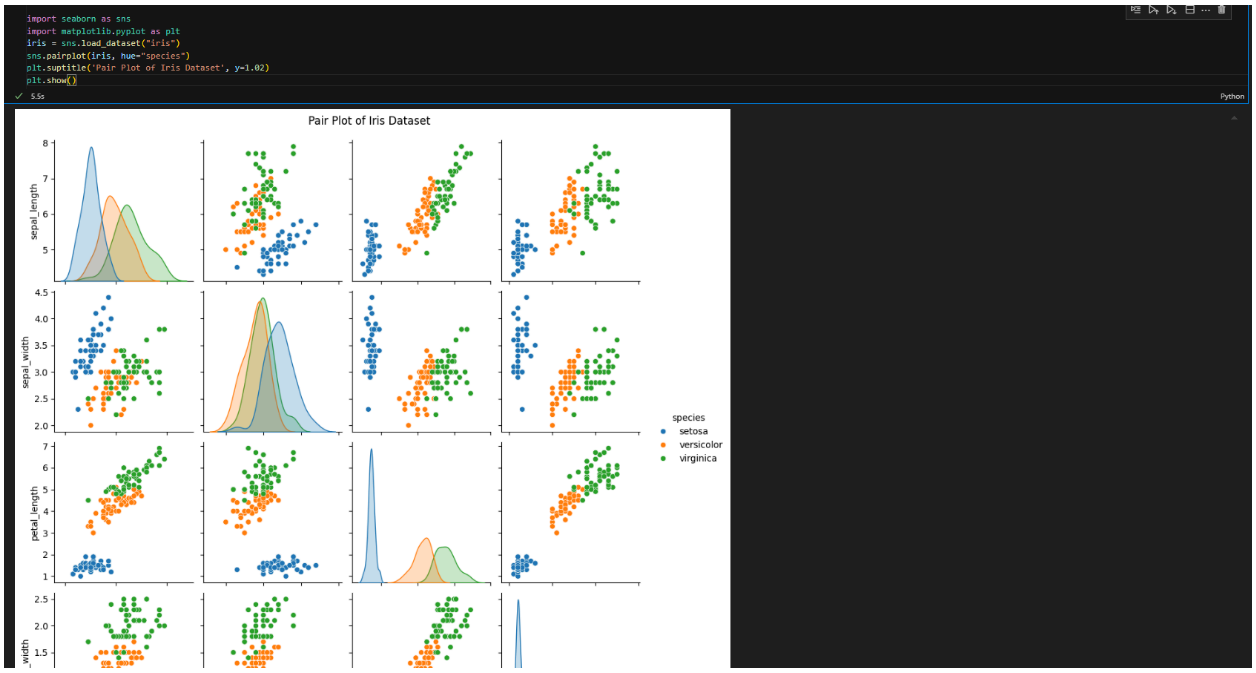


```python
import seaborn as sns
import matplotlib.pyplot as plt
flights = sns.load_dataset("flights")
flights_pivot = flights.pivot_table(index="month", columns="year", values="passengers")
sns.heatmap(flights_pivot, annot=True, fmt=".1f", cmap="YlGnBu")
plt.title('Heatmap of Flight Passengers')
plt.xlabel('Year')
plt.ylabel('Month')
plt.show()
```

✓ 0.5s

v)Pair Plot

A pair plot in Seaborn is a grid of pairwise plots that displays the relationships between pairs of variables in a dataset. It enables us to quickly visualize patterns, correlations, and distributions across multiple dimensions. Pair plots are especially useful when dealing with datasets containing multiple numerical variables.



```python
import seaborn as sns
import matplotlib.pyplot as plt
iris = sns.load_dataset("iris")
sns.pairplot(iris, hue="species")
plt.suptitle('Pair Plot of Iris Dataset', y=1.02)
plt.show()
```

COMPARISION BETWEEN MATPLOTLIB AND SEABORN

Ease of Use:

- **Matplotlib:**
  - **Strengths: Provides granular control over every aspect of a plot, allowing users to create highly customized visualizations.**
  - **Weaknesses: Requires more code to create complex plots compared to higher-level libraries like Seaborn. Beginners may find it challenging initially.**
- **Seaborn:**
  - **Strengths: Offers a higher-level interface with concise syntax for creating attractive statistical plots.**
  - **Weaknesses: While easier to use for common statistical plots, customization beyond Seaborn's built-in options may require falling back on Matplotlib.**

Customization Options:

- Matplotlib:
    - Strengths: Offers extensive customization options through its object-oriented API. Users have fine-grained control over plot elements.
    - Weaknesses: Customizing plots often involves writing more lines of code and understanding the Matplotlib object model.
- Seaborn:
    - Strengths: Provides a good balance between ease of use and customization. Offers built-in themes and color palettes that enhance plot aesthetics.
    - Weaknesses: Limited compared to Matplotlib in terms of low-level customization. Advanced customization may require using Matplotlib directly.

Interactivity:

- Matplotlib:
    - Strengths: Supports a variety of interactive features through tools like `matplotlib.widgets` and `mplcursors`.

    - **Weaknesses: Setting up interactivity can be more complex and requires additional code compared to more interactive-focused libraries.**
- **Seaborn:**
    - **Strengths: Primarily focused on static statistical plotting rather than interactivity.**
    - **Weaknesses: Limited built-in support for interactive plots. Users seeking interactive features may need to integrate with additional libraries or tools.**

Performance with Large Datasets:

- **Matplotlib:**
    - **Strengths: Efficient for rendering large datasets due to its lower-level approach and ability to optimize plot rendering.**
    - **Weaknesses: Performance can degrade with extremely large datasets or complex plots that involve many elements.**
- **Seaborn:**
    - **Strengths: Optimized for handling medium-sized datasets commonly encountered in statistical analysis.**

    - **Weaknesses: May face performance issues with very large datasets or plots requiring intricate statistical computations.**

Conclusion:

- **Matplotlib is powerful for creating customized, publication-quality plots with fine control over details. It's suitable for complex visualizations and scenarios where performance optimization is critical.**
- **Seaborn excels in producing attractive, informative statistical plots quickly and with minimal code. It's ideal for exploratory data analysis and visualizing relationships in datasets without the need for extensive customization.**

Choosing between Matplotlib and Seaborn often depends on the specific requirements of your visualization task, your familiarity with each library, and the trade-offs between customization, ease of use, and performance for your particular dataset size and complexity. Many users leverage both libraries, using

Matplotlib for detailed customization and Seaborn for rapid exploration and presentation of statistical insights.