VISUALIZATION

Visualization is a way of representing. Data visualization is representing of data in a graphical or pictorial format for better understanding of data. Visualization gives a good idea of data and the trends in it.

In python, data visualization has multiple libraries like matplotlib, seaborn, plotly etc.

Let us dive deeper into matplotlib and seaborn.

MATPLOTLIB

Matplotlib is a comprehensive library used to create static, animated, and interactive visualizations in Python. It is especially useful for 2D plotting and is the foundation for many other visualization libraries such as Seaborn and Pandas plot().

- It was created by John D. Hunter in 2003.
- It is inspired by the MATLAB plotting interface.
- Works well with NumPy, Pandas, and other scientific libraries.

A plot in matplotlib contains:

1. Figma

- The main container for all plot elements.
- A single figure can contain one or more subplots (Axes).

Created using:

```
fig = plt.figure()
```

2. Axes

- The actual plot (a region where data is plotted).
- A figure can have multiple Axes (for subplots).
- Each Axes contains:
    - X-axis
    - Y-axis
    - Plot elements (like lines, bars, etc.)

Set axis labels using:

```
ax.set_xlabel("X Label")
```

```
ax.set_ylabel("Y Label")
```

3.Axis

- Refers to XAxis and YAxis objects inside the Axes.
- Controls:
    - Tick marks
    - Tick labels
    - Axis limits

Example:

```
ax.set_xlim(0, 10)
ax.set_ylim(0, 100)
```

4. Artists

- Everything you see in a plot is an Artist.
- Two types:
    - Primitive Artists: lines, text, patches
    - Composite Artists: Axes, Figure, Legend
- Examples of artists:
    - ax.plot() → Line artist
    - ax.set_title() → Text artist
    - ax.grid() → Grid artist

5. Title

- Explains what the plot is about.

Example:

```
ax.set_title("Plot Title")
```

PYPLOT

pyplot is a module in the Matplotlib library, typically imported as:

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

We also import numpy as a support for the matplotlib :

```
import numpy as np
```

- It provides a simple interface for plotting, similar to MATLAB.

- It helps in creating and customizing plots quickly using functions like:
    - plt.plot()
    - plt.title()
    - plt.xlabel()
    - plt.ylabel()
    - plt.show()

Some of the plots in matplotlib are:

LINE CHART

A line plot displays information as a series of data points (markers) connected by straight line segments.

Use Case:
Used for visualizing trends over time (e.g., stock prices, temperature, performance metrics). Ideal for continuous datasets.

Example Libraries:

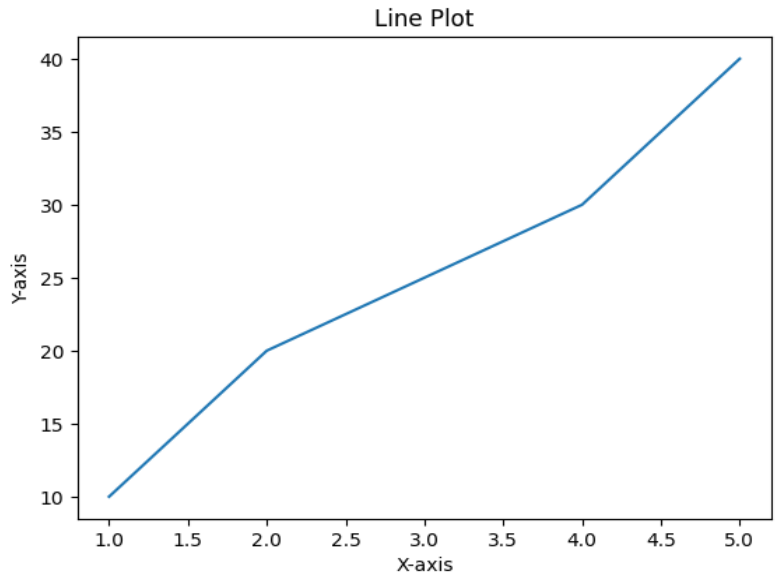
- matplotlib.pyplot.plot()
- seaborn.lineplot()

code snippet:

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

x = [1, 2, 3, 4, 5]
y = [10, 20, 25, 30, 40]
plt.plot(x, y)
plt.title("Line Plot")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Output:



Description:

- plot() is used for plotting a line chart where data points are connected by straight lines.
- title() sets the title of the graph.
- xlabel() and ylabel() label the x and y axes respectively.
- show() displays the plotted line chart.
  Use Case: Ideal for showing trends or changes over time like stock prices or temperature data.

BAR CHART

Bar charts represent categorical data with rectangular bars where the length of each bar is proportional to the value it represents.

Use Case:
Useful for comparing values across different categories (e.g., population by country, sales by product).

Example Libraries:

- matplotlib.pyplot.bar()
- seaborn.barplot()

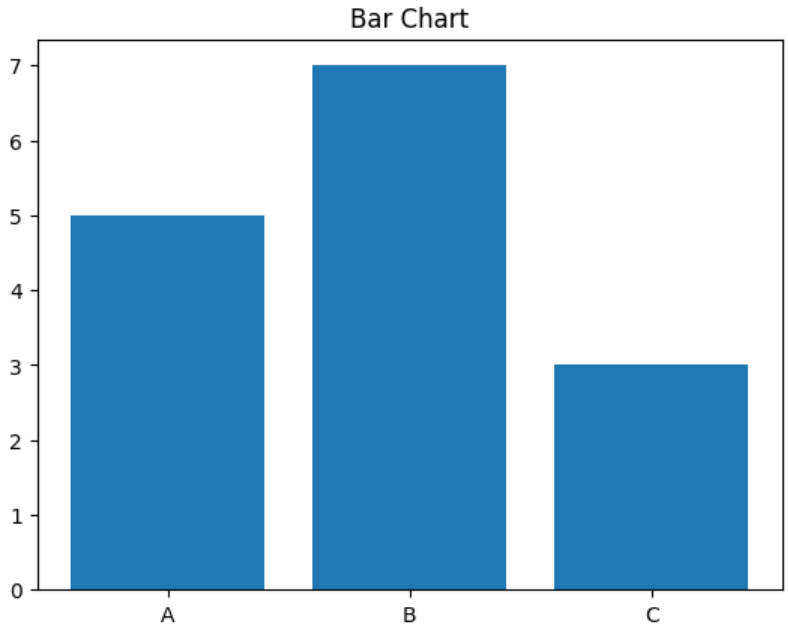
code snippet:

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

x = ['A', 'B', 'C']
y = [5, 7, 3]
plt.bar(x, y)
plt.title("Bar Chart")
```

```
plt.show()
```

Output:



Description:

- bar() is used to draw a bar chart. Each bar's height or length represents the value of a category.
- xlabel() and ylabel() help in labeling the axes.
- title() is used to set the chart's heading.
- show() renders the chart.
  Use Case: Helpful for comparing data between different categories, such as sales by region or products.

HISTOGRAM

A histogram displays the distribution of a dataset by dividing data into continuous intervals (bins) and counting the number of observations in each bin.

Use Case:
Used to show the frequency distribution of numerical data (e.g., exam scores, age groups).

Example Libraries:

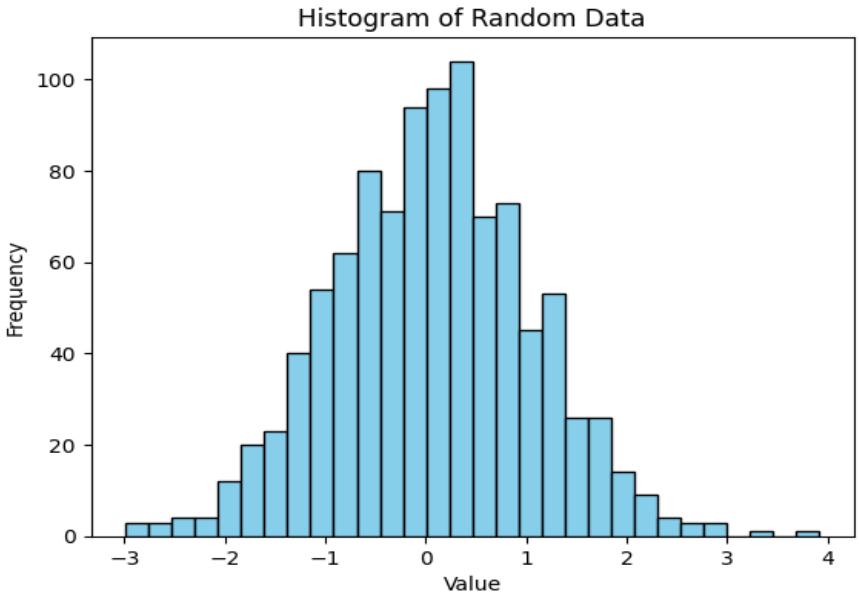
- matplotlib.pyplot.hist()
- seaborn.histplot()

code snippet:

```
import numpy as np
import matplotlib.pyplot as plt
data = np.random.randn(1000)
plt.hist(data, bins=30, color='skyblue', edgecolor='black')
plt.title("Histogram of Random Data")
plt.xlabel("Value")
```

```
plt.ylabel("Frequency")
plt.show()
```

Output:



Description:

- np.random.randn(1000): Generates 1000 random values from a standard normal distribution.
- plt.hist(): Plots the histogram with 30 bins, sky blue bars, and black edges.
- plt.title(), plt.xlabel(), plt.ylabel(): Add labels and title to the graph.
- plt.show(): Displays the final plot.

SCATTER PLOT

A scatter plot uses Cartesian coordinates to display values for typically two variables for a set of data. Each dot represents an observation.

Use Case:
Used to identify relationships or correlations between two continuous variables (e.g., height vs weight, advertising spend vs sales).

Example Libraries:

- matplotlib.pyplot.scatter()
- seaborn.scatterplot()

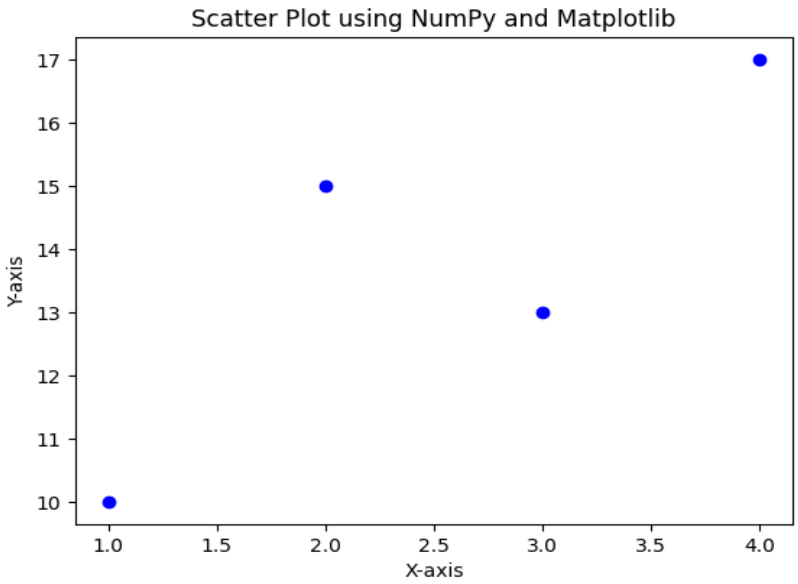
code snippet:

```
import numpy as np
```

```
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4])
y = np.array([10, 15, 13, 17])
plt.scatter(x, y, color='blue', marker='o')
plt.title("Scatter Plot using NumPy and Matplotlib")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Output:



Description:

- np.array(): Creates arrays for x and y values.
- plt.scatter(): Draws the scatter plot with circular blue markers.
- plt.title(), plt.xlabel(), plt.ylabel(): Add title and axis labels.
- plt.show(): Displays the graph.

PIECHART

A pie chart is a circular graph divided into slices to illustrate numerical proportions.

Use Case:
Used to show percentage or proportional data where the sum of all categories is 100% (e.g., market share, budget allocation).

Example Libraries:

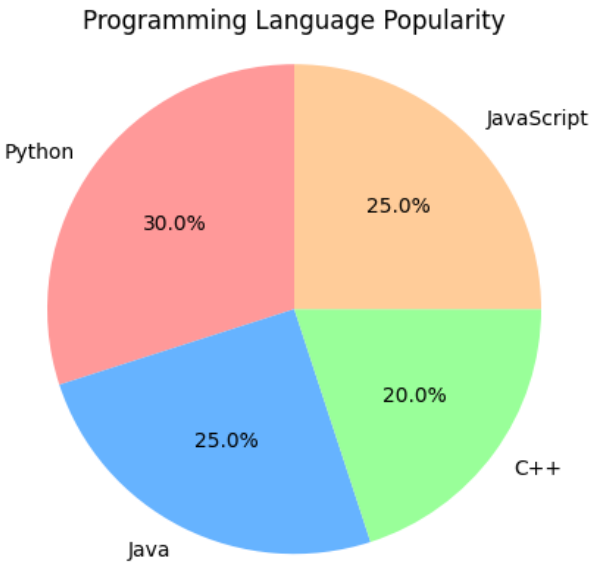
- matplotlib.pyplot.pie()

code snippet:

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

```
labels = ['Python', 'Java', 'C++', 'JavaScript']
data = np.array([30, 25, 20, 25])
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99']
plt.pie(data, labels=labels, colors=colors, autopct='%1.1f%%',
startangle=90)
plt.title("Programming Language Popularity")
plt.axis('equal')
```

Output:



Description:

- np.array() is used to define the data values.
- plt.pie() draws the pie chart.
- autopct='%1.1f%%' shows percent values on the chart.
- plt.axis('equal') makes the chart a proper circle.

AREA CHART

An area chart is a graph that displays quantitative data using filled areas under a line. It is similar to a line chart, but the area between the line and the axis is filled with color to emphasize the magnitude of values over time or across categories.

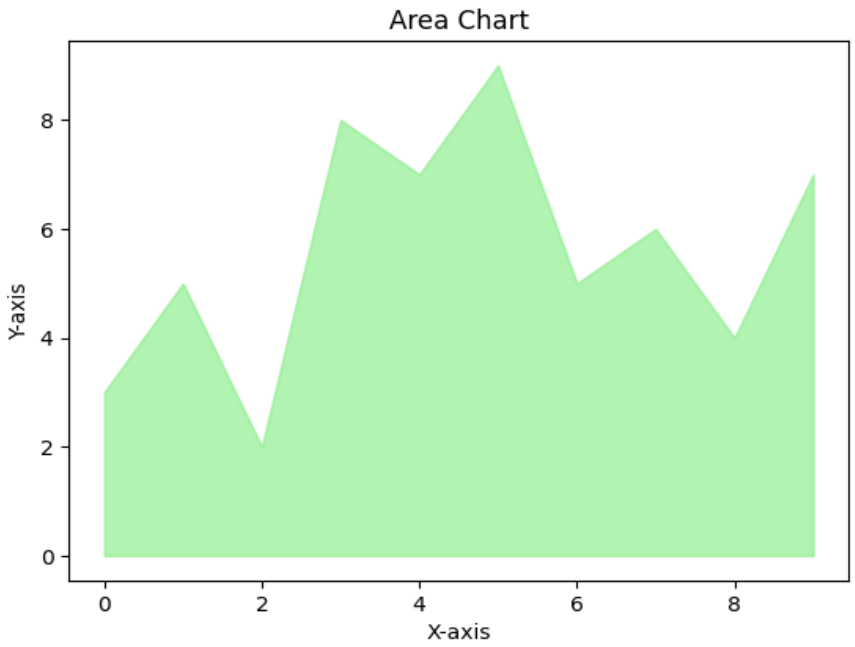
- Showing sales growth over time.
- Visualizing stock prices, website traffic, or rainfall amounts.
- Comparing part-to-whole relationships (in stacked area charts).

Code snippet:

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

```
x = np.arange(0, 10, 1)
y = np.array([3, 5, 2, 8, 7, 9, 5, 6, 4, 7])
plt.fill_between(x, y, color='lightgreen', alpha=0.7)
plt.title("Area Chart")
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```

Output:



Description:

- np.arange(0, 10, 1): Generates x values from 0 to 9.
- y: An array of sample y-values.
- plt.fill_between(x, y, ...): Fills the area between the x-axis and the y-values.
- color='lightgreen': Sets the fill color.
- alpha=0.7: Controls transparency.
- plt.title(), plt.xlabel(), plt.ylabel(): Adds title and labels.
- plt.show(): Displays the plot.

SEABORN

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive, informative, and statistical graphics. It is advanced than matplotlib and have different features which are default like style, color palette. Seaborn has different categories of plots like relational plot, categorical plot, distribution plot, regression plot, matrix plot.

The different plots in each category are:

1. Relational Plots:

It is used to visualize relationships between numerical variables.

- scatterplot()
- lineplot()
- relplot()

2. Categorical Plots:

It is used to compare values across categories (groups).

- barplot()
- countplot()
- boxplot()
- violinplot()
- swarmplot()
- pointplot()
- catplot()

3. Distribution Plots:

It is used to visualize the distribution (spread) of a dataset.

- histplot()
- kdeplot()
- rugplot()
- distplot()

4. Relational Plots:

It is used to show linear relationships and fit lines.

- regplot()
- lmplot()

5. Matrix Plots:

It is used to show relationships among multiple variables.

- heatmap()
- clustermap()

Here are some sample codes for some of the graphs.

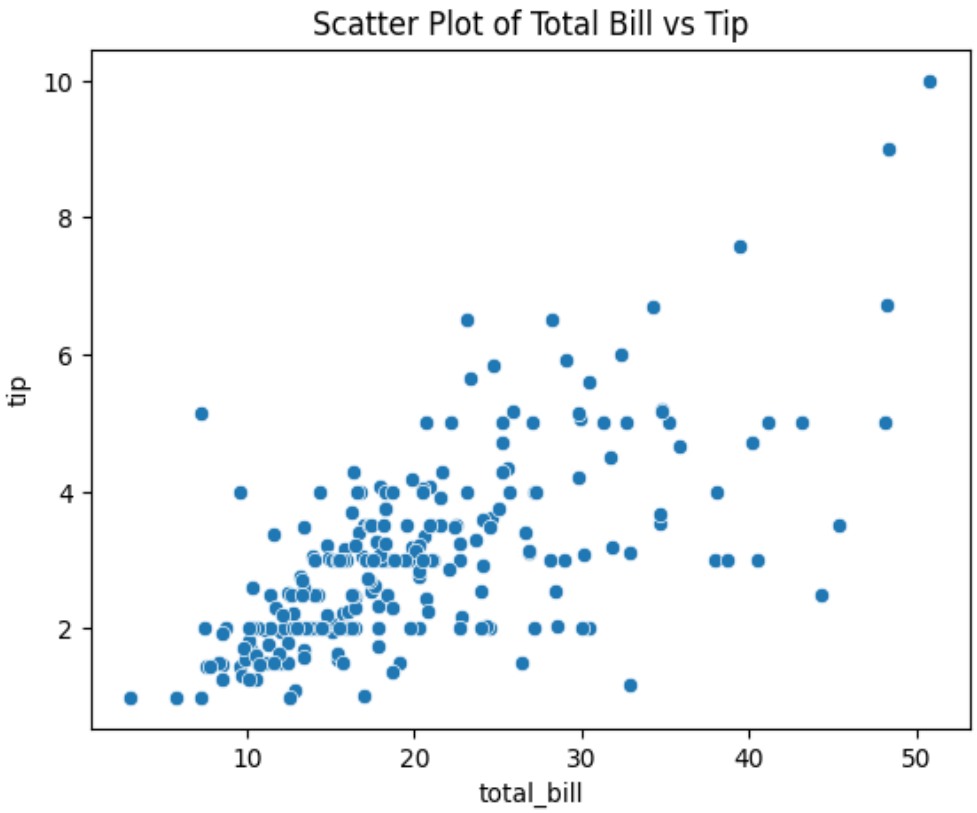
SCATTERPLOT

It creates a scatter plot to show the relationship between two numeric variables using dots.

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

tips = sns.load_dataset("tips")
sns.scatterplot(x='total_bill', y='tip', data=tips)
plt.title("Scatter Plot of Total Bill vs Tip")
plt.show()
```

Output:



Description:

- sns.load_dataset("tips") loads a sample dataset of restaurant bills and tips.
- sns.scatterplot() plots each row as a point showing the relationship between total_bill and tip.
- A scatter plot helps identify trends, patterns, or outliers between two numeric variables.

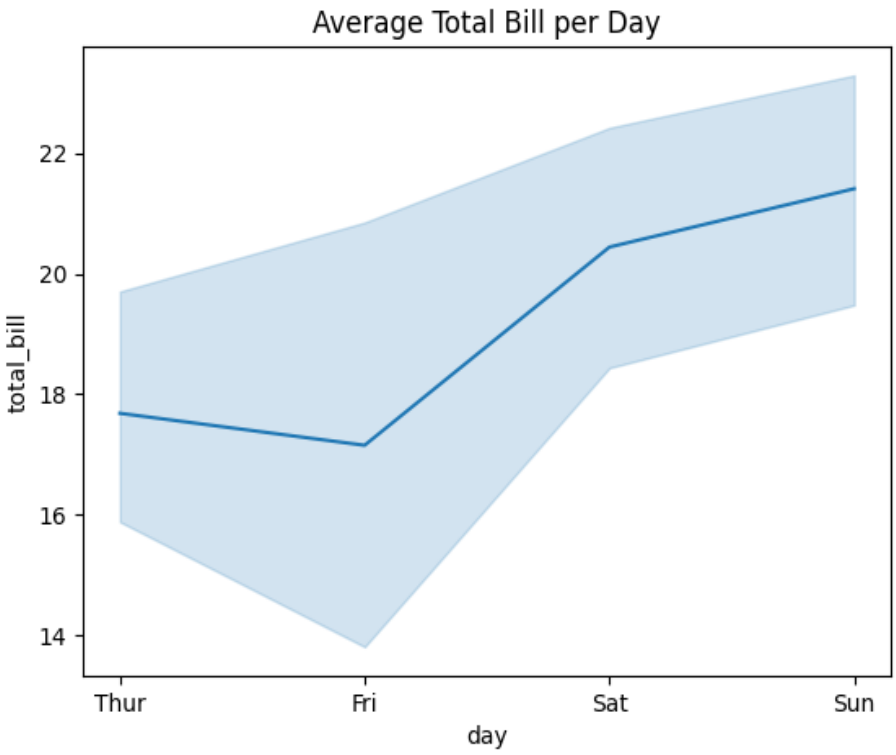
LINE PLOT

It creates a line plot to show a trend or pattern over time or sequential data.

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

sns.lineplot(x='day', y='total_bill', data=tips, estimator='mean')
plt.title("Average Total Bill per Day")
plt.show()
```

Output:



Description:

- sns.lineplot() creates a line graph where the x-axis is days (Thur, Fri, etc.) and the y-axis is the average total_bill.
- estimator='mean' computes and plots the average bill per day.
- Useful for showing trends over categories or time.

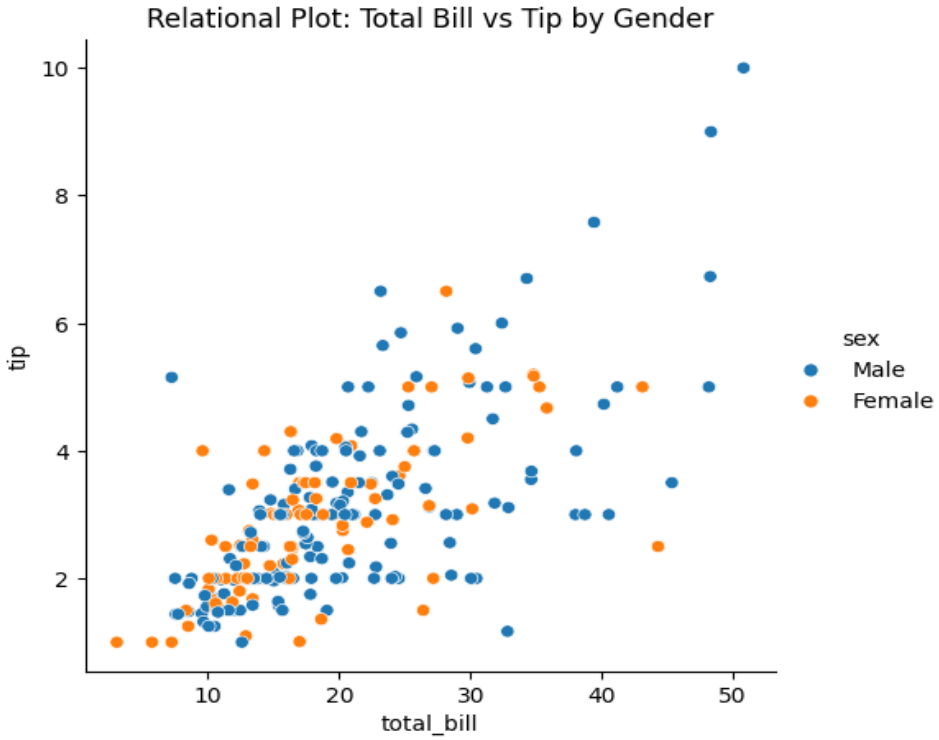
RELPLOT

It can create both scatter and line plots.

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

tips = sns.load_dataset("tips")
sns.relplot(x='total_bill', y='tip', hue='sex', data=tips)
plt.title("Relational Plot: Total Bill vs Tip by Gender")
plt.show()
```

Output:



- Description:
- sns.relplot() is a flexible function to make scatter or line plots.
- hue='sex' colors the points based on gender.
- This plot shows the relationship between bill and tip, grouped by gender, and is useful for multi-variable analysis.

BAR PLOT

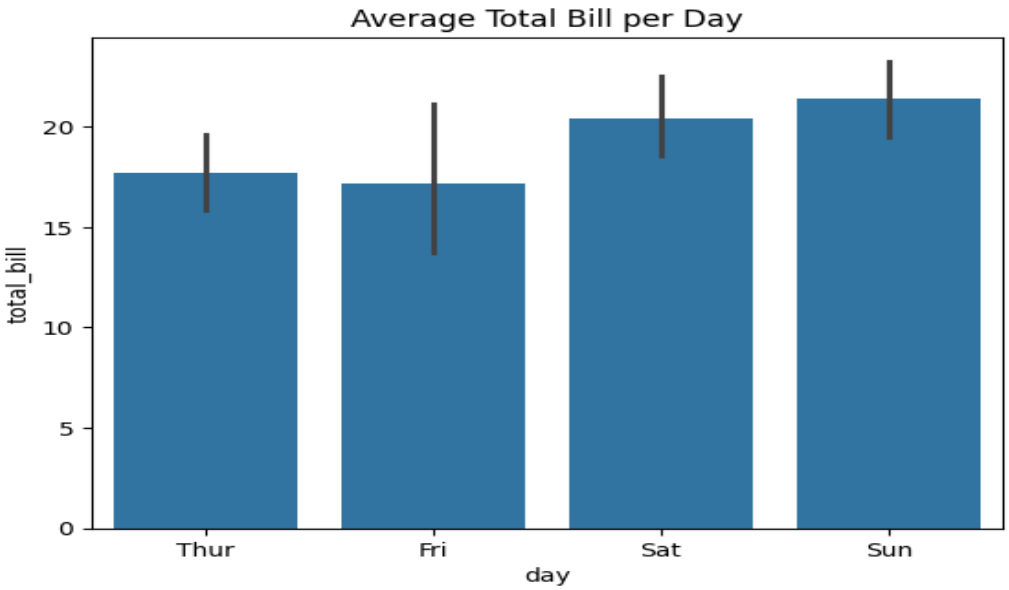
A bar plot shows the average value of a numerical variable grouped by categories.

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

# Load sample dataset
tips = sns.load_dataset("tips")

# Bar plot: Average total bill per day
sns.barplot(x="day", y="total_bill", data=tips)
plt.title("Average Total Bill per Day")
plt.show()
```

Output:



Description:

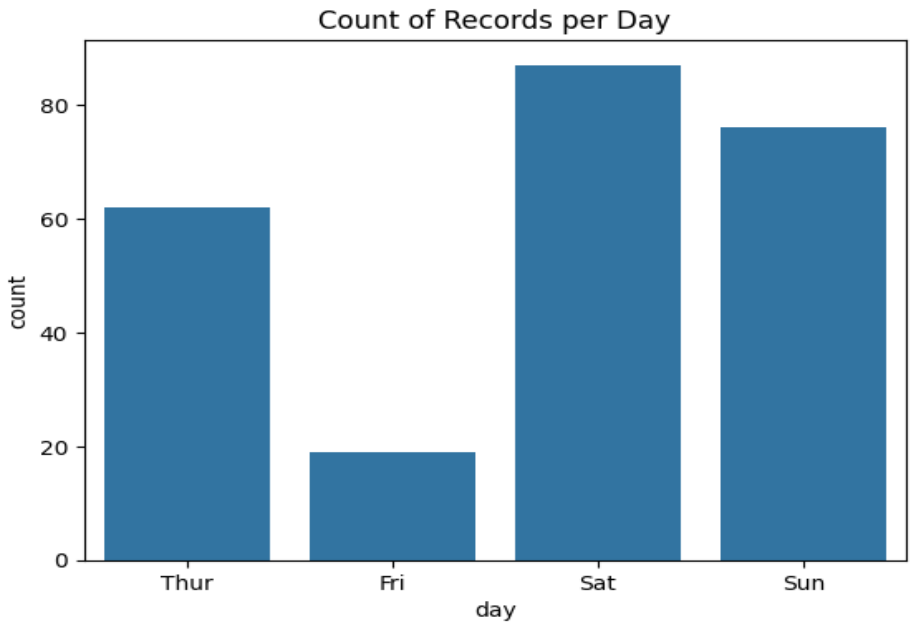
- sns.barplot() plots average values of a numerical variable (total_bill) across categories (day).
- It includes error bars (by default 95% CI).
- Useful for comparing mean values between groups.

COUNT PLOT

A count plot displays the number of observations for each category in a variable.

```
import seaborn as sns
tips = sns.load_dataset("tips")
sns.countplot(x="day", data=tips)
plt.title("Count of Records per Day")
plt.show()
```

Output:



Description:

- **sns.countplot()** shows the **count of occurrences** for each category in a single variable (day).
- Useful for visualizing **frequency** or **distribution** of categories.
- Does **not** require a numeric y-value.

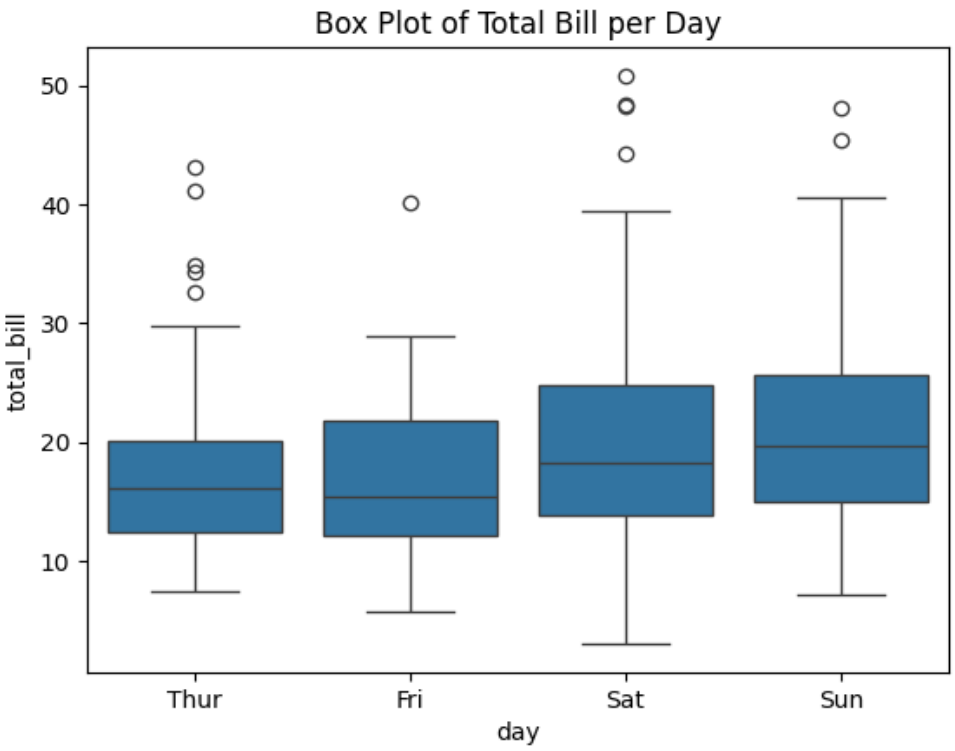
BOX PLOT

A box plot is a graphical summary of a dataset that shows:

- Central value (median)
- Spread (interquartile range - IQR)
- Skewness
- Outliers

```
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 per Day")
plt.show()
```

Output:



Description:

sns.boxplot() creates a box-and-whisker plot to summarize the distribution of a numeric variable grouped by categories.

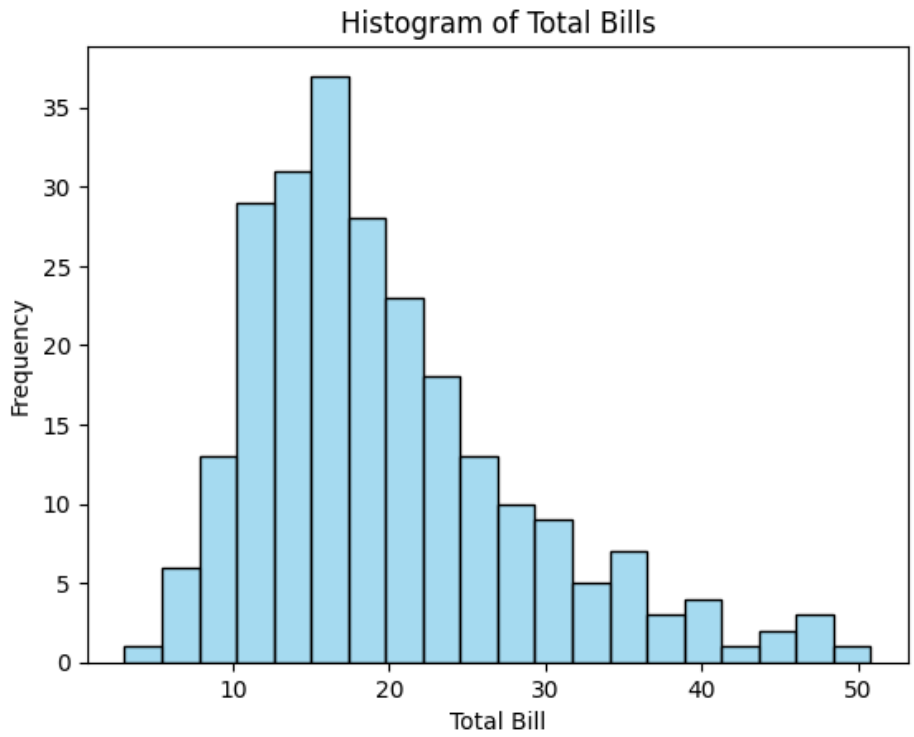
HISTOGRAM PLOT

Plots the frequency of observations within intervals and represents how often a value (or range of values) occurs in a dataset.

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

tips = sns.load_dataset("tips")
sns.histplot(tips['total_bill'], bins=20, kde=False, color='skyblue')
plt.title("Histogram of Total Bills")
plt.xlabel("Total Bill")
plt.ylabel("Frequency")
plt.show()
```

Output:



Description:

- histplot() shows the frequency of data in intervals (bins).
- The x-axis shows ranges of total_bill, and the y-axis shows how many bills fall into each range.
- Used to understand the distribution and shape of the data (e.g., skewed, normal, etc.).

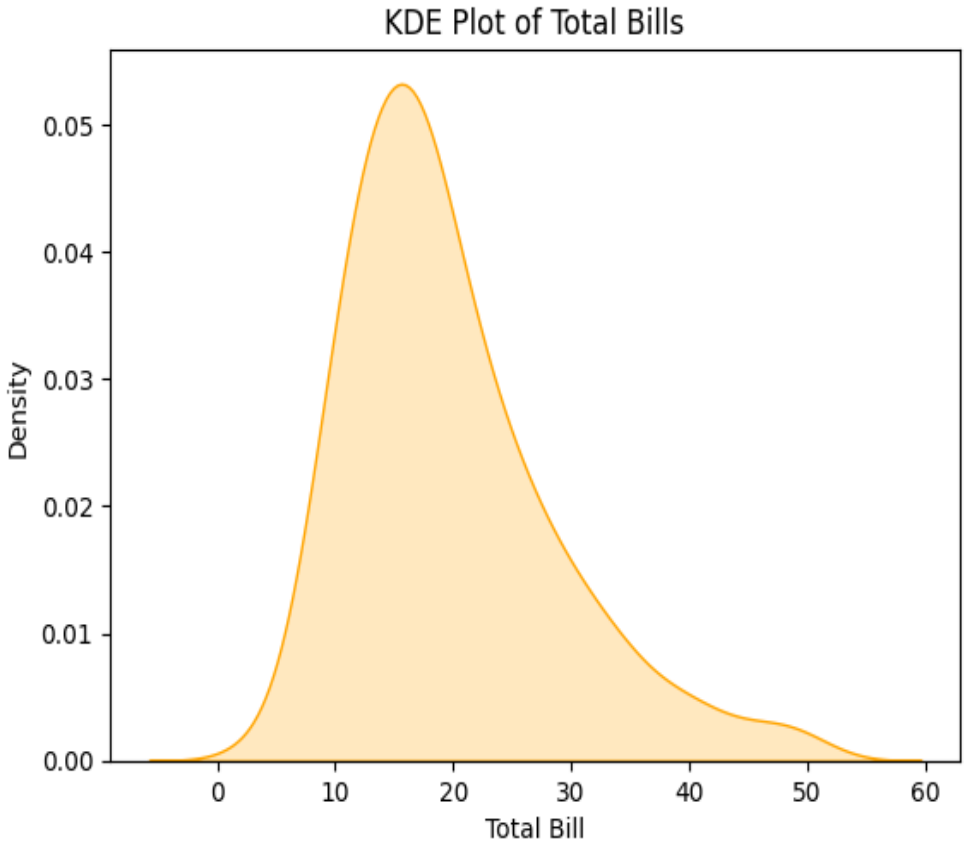
KDEPLOT() – KERNEL DENSITY ESTIMATE PLOT

Draws a smooth curve to estimate the probability density of a variable.

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

sns.kdeplot(data=tips['total_bill'], shade=True, color='orange')
plt.title("KDE Plot of Total Bills")
plt.xlabel("Total Bill")
plt.ylabel("Density")
plt.show()
```

Output:



Description:

- kdeplot() shows a smooth curve that estimates the probability density of the data.
- It's a continuous version of a histogram, good for understanding the underlying distribution.
- shade=True fills the area under the curve.

REGPLOT

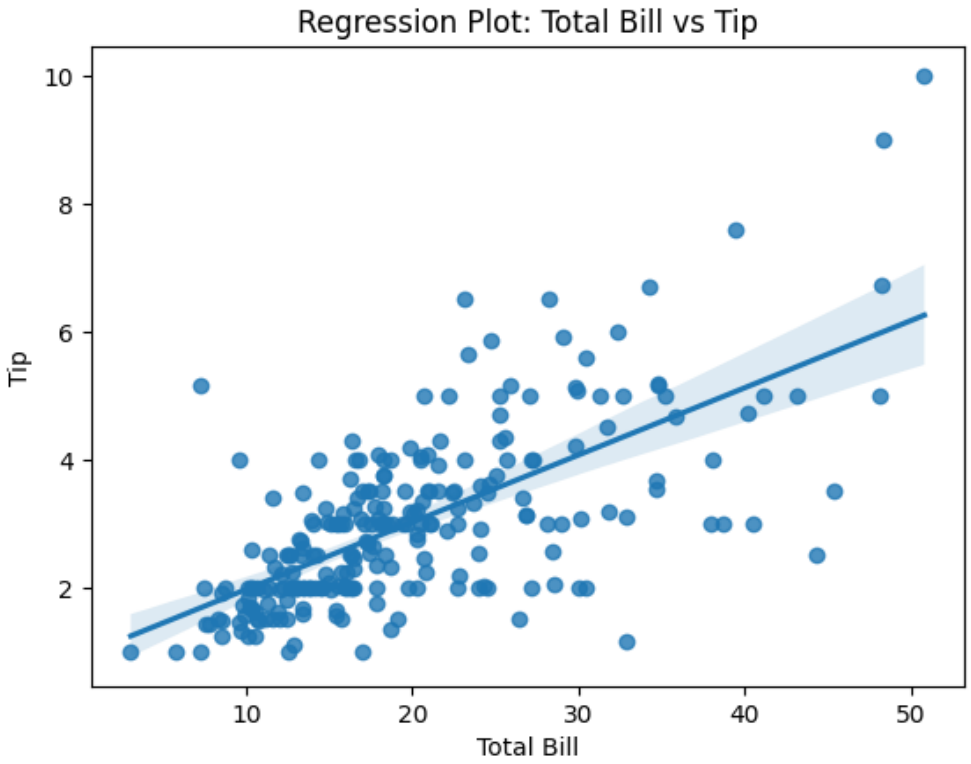
Plots a scatterplot with a linear regression line.

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

tips = sns.load_dataset("tips")
sns.regplot(x='total_bill', y='tip', data=tips)
```

```
plt.title("Regression Plot: Total Bill vs Tip")
plt.xlabel("Total Bill")
plt.ylabel("Tip")
plt.show()
```

Output:



Description:

- Dots represent individual observations of total_bill and tip.
- A straight line is drawn showing the linear regression trend.
- The shaded area represents the confidence interval for the regression.

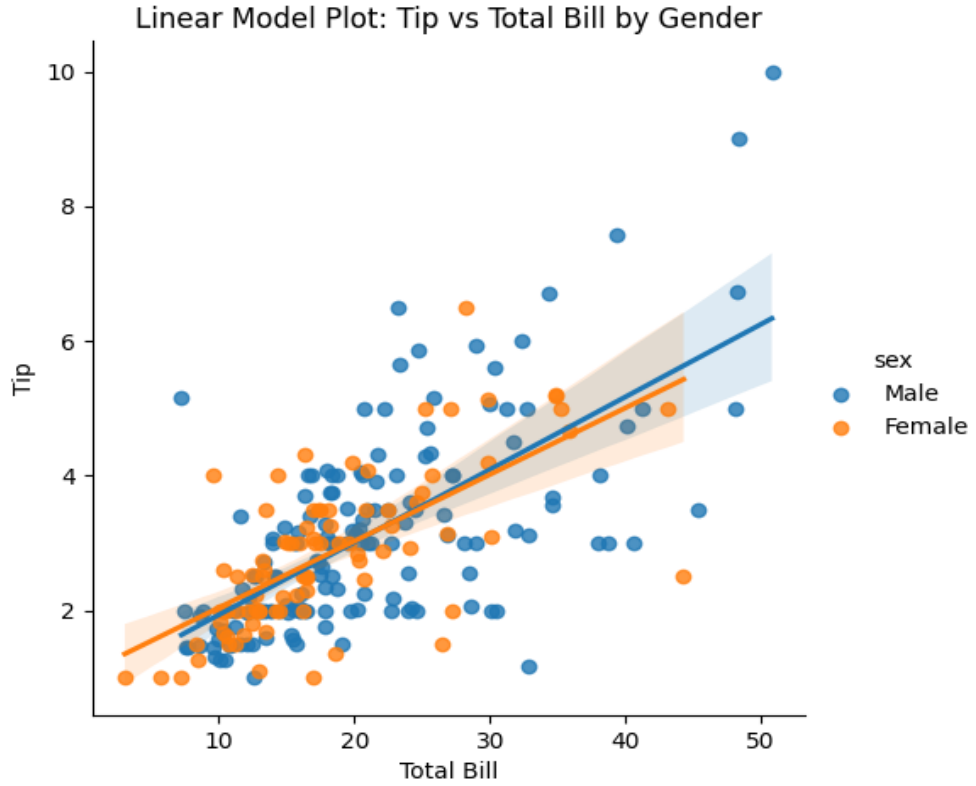
LMPLOT

Plots regression lines like regplot(), but with more flexibility and supports grouping and subplots using hue, col, or row.

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

tips = sns.load_dataset("tips")
sns.lmplot(x='total_bill', y='tip', hue='sex', data=tips)
plt.title("Linear Model Plot: Tip vs Total Bill by Gender")
plt.xlabel("Total Bill")
plt.ylabel("Tip")
plt.show()
```

Output:



Description:

- A separate regression line is drawn for each gender (Male/Female).
- Each line shows how tip relates to total_bill for that group.
- Points and trendlines are color-coded by the hue parameter.

HEATMAP

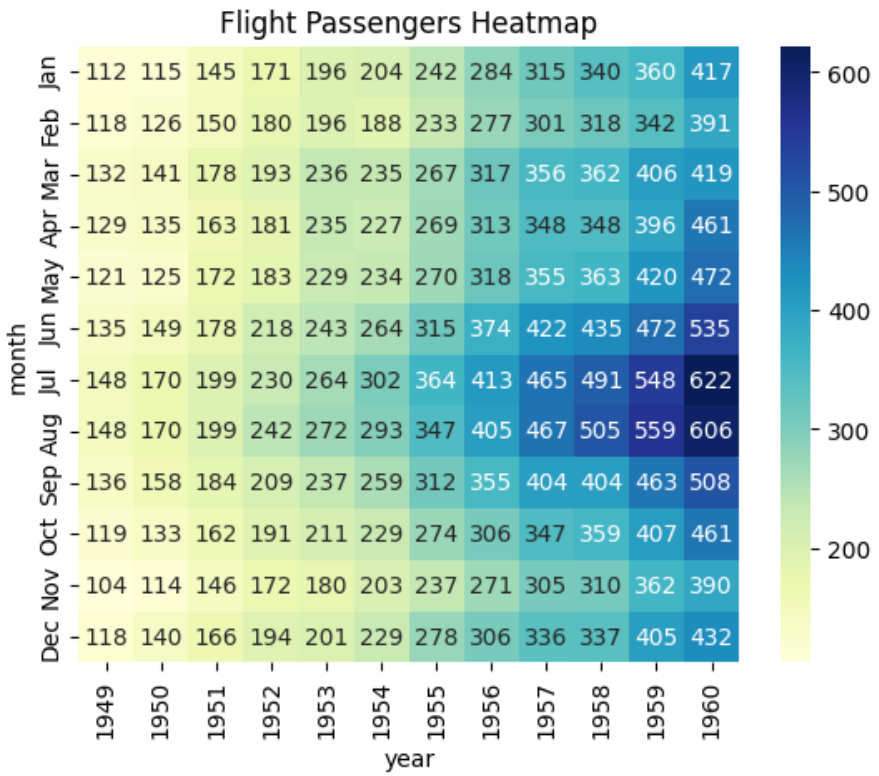
Displays a color-coded matrix to visualize numerical values like correlations or pivot tables.

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

flights = sns.load_dataset("flights")
pivot_table = flights.pivot(index="month", columns="year",
values="passengers")

sns.heatmap(pivot_table, annot=True, fmt='d', cmap='YlGnBu')
plt.title("Flight Passengers Heatmap")
plt.show()
```

Output:



Description:

- A color-coded matrix showing the number of passengers per month and year.
- Darker cells = more passengers.
- annot=True adds numbers inside each cell.
- Used to visualize density, correlation, or time-based changes in a matrix format.

CLUSTERMAP

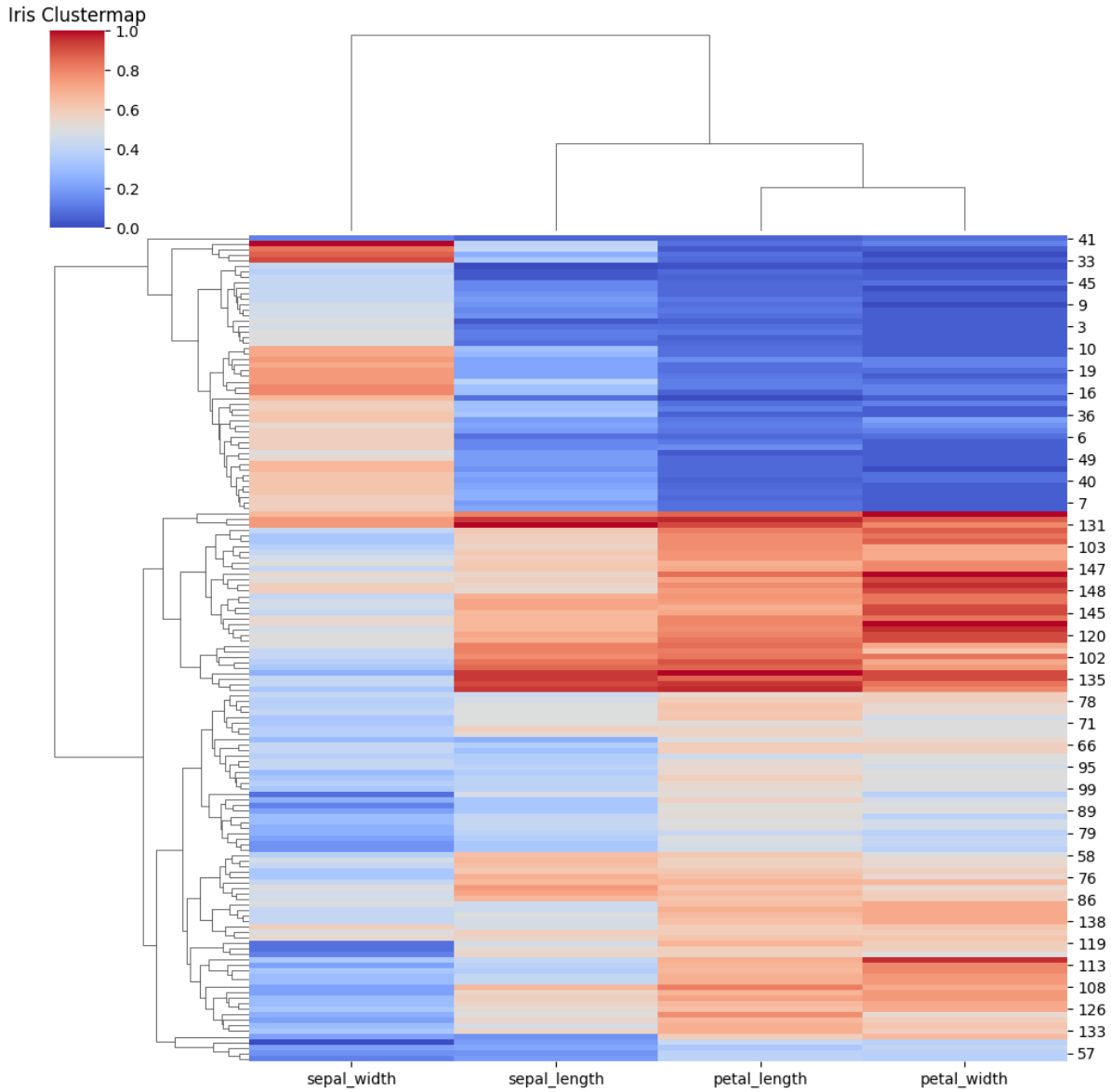
It extends heatmap by adding hierarchical clustering to group similar rows and columns automatically.

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

# Load dataset
iris = sns.load_dataset("iris")
iris_data = iris.drop("species", axis=1)

# Create clustermap
sns.clustermap(iris_data, cmap="coolwarm", standard_scale=1)
plt.title("Iris Clustermap")
plt.show()
```

Output:



Description:

- Similar to a heatmap but with hierarchical clustering.
- Automatically groups similar rows and columns together.
- Adds dendrograms (tree diagrams) to show how features are related.

COMPARISON OF MATPLOTLIB AND SEABORN

Matplotlib:

- Core Library : Matplotlib is the foundational data visualization library in Python, designed for creating static, animated, and interactive plots.
  It provides a low-level API for precise control over every element of a figure, making it highly customizable and powerful.
- Flexibility and Versatility : Matplotlib supports a wide variety of plots: from basic ones like line plots, bar charts, and scatter plots to advanced ones like 3D plots, polar plots, and geographical maps.
  You can build any custom plot type by combining elements like axes, figures, and shapes manually.
- Advanced Customization : With Matplotlib, you can fine-tune every detail of your plot—line style, marker type, colors, fonts, figure size, axis ticks, grids, legends, and more.
  This level of control is crucial when preparing publication-quality visuals or complex, domain-specific visualizations.
- Seamless Integration : Matplotlib integrates naturally with other Python libraries like NumPy, Pandas, SciPy, and Jupyter Notebooks.
  It forms the backend for many high-level libraries (e.g., Seaborn, Pandas .plot(), StatsModels), making it essential for data science workflows.

Advantages Of Matplotlib

- You can control every detail of the plot (colors, labels, sizes, etc.).
- Can create line plots, bar charts, scatter plots, 3D plots, pie charts, and more.
- Easily plots data from NumPy and other Python data structures.
- Great for scientific or publication-quality graphs.
- It's a mature library with strong community and support.

Seaborn

- Core Library : Seaborn is a high-level data visualization library built on top of Matplotlib.
  It provides a simplified interface for creating beautiful, informative, and statistical graphics with fewer lines of code.
- Statistical Power : Seaborn includes built-in support for many statistical plots such as box plots, violin plots, KDEs, heatmaps, and regression plots.
  These allow you to quickly explore patterns, distributions, and relationships in your dataset without manual calculation or setup.
- Beautiful by Default : Seaborn comes with aesthetically pleasing themes, color palettes, and style settings.
  Plots look professional and publication-ready out of the box, with minimal styling required.
- DataFrame-Friendly : It is designed to work directly with Pandas DataFrames, allowing you to reference columns by name.

This makes it especially efficient for exploratory data analysis (EDA) and simplifies plotting complex relationships in data.

- Integration and Extensibility : Seaborn integrates seamlessly with Matplotlib, NumPy, Pandas, and SciPy.
  It also allows advanced users to use Matplotlib functions for further customization, making it both powerful and flexible.

Advantages Of Seaborn

- Seaborn makes it easier to create complex plots with just one or two lines of code, especially when working with Pandas DataFrames.
- Automatically applies clean, professional-looking color schemes and themes, with minimal user input.
- Includes advanced plots like boxplots, violin plots, heatmaps, and regression lines, which are not native to Matplotlib.
- Automatically computes statistical summaries (like means, confidence intervals) in plots like barplot() and pointplot().
- Works directly with column names from DataFrames, allowing you to write intuitive, readable code during data analysis.