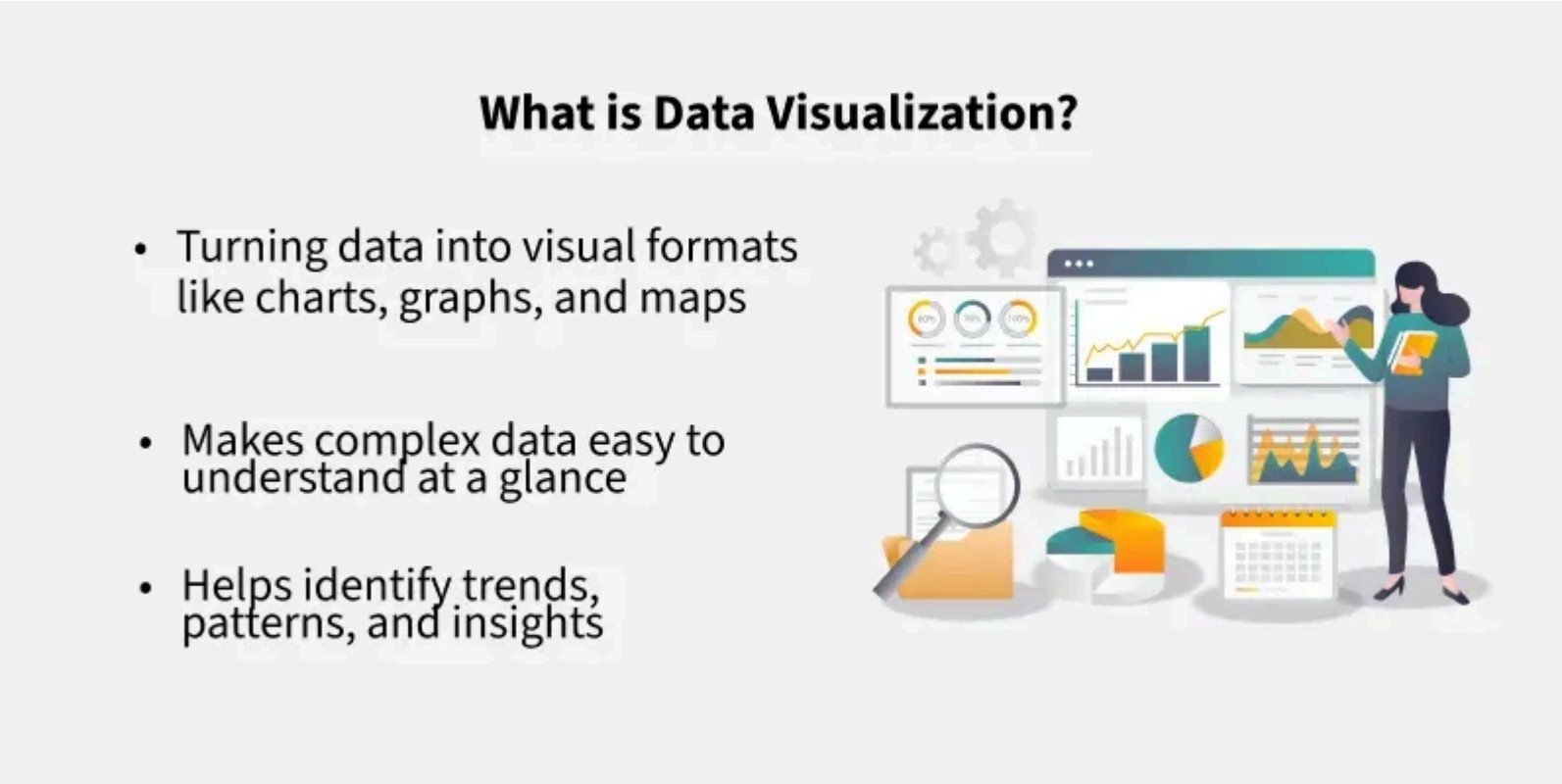
Deepika

AIML Trainee | AI/ML bootcamp

Data Visualizaton



Data Visualization

Data visualization uses charts, graphs and maps to present information clearly and simply. It turns complex data into visuals that are easy to understand.

With large amounts of data in every industry, visualization helps spot patterns and trends quickly, leading to faster and smarter decisions.

• Libraries for Data Visualization

- Matplotlib
- Seaborn
- Pandas
- Plotly
- Numpy

- **Matplotlib:-** Matplotlib is a popular 2D plotting library in Python, widely used for creating charts like line plots, bar charts, pie charts and more. It works across platforms and integrates with Jupyter, Python scripts and GUI apps. The pyplot module offers a MATLAB-like interface, making it easy to use and highly flexible

- **Seaborn:-** Seaborn is a Python visualization library built on top of Matplotlib, designed for creating attractive and informative statistical graphics. It works well with NumPy and pandas data and offers built-in themes, color palettes and functions to easily create plots like bar charts, histograms, scatterplots and more. It simplifies complex visualizations with less code.

- **Pandas:-** Pandas is a powerful open-source data analysis and manipulation library for Python. The library is particularly well-suited for handling labeled data such as tables with rows and columns. Pandas allows to create various graphs directly from your data using built-in functions.

- **Plotly:-** Plotly is a free, open-source Python library for creating interactive, web-based visualizations. Built on top of plotly.js, it supports over 40 chart types including 3D plots and contour plots. Plotly works in Jupyter notebooks, web apps and can save visuals as HTML files. It also works offline.

- **Numpy:-** NumPy provides several techniques for data visualization like line plots, scatter plots, bar graphs, and histograms.

-->Loading dataset from Kaggle (Tips Dataset)

[1]:
```
import seaborn  as sns
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

df=pd.read_csv("tips.csv")
df
```

[1]:

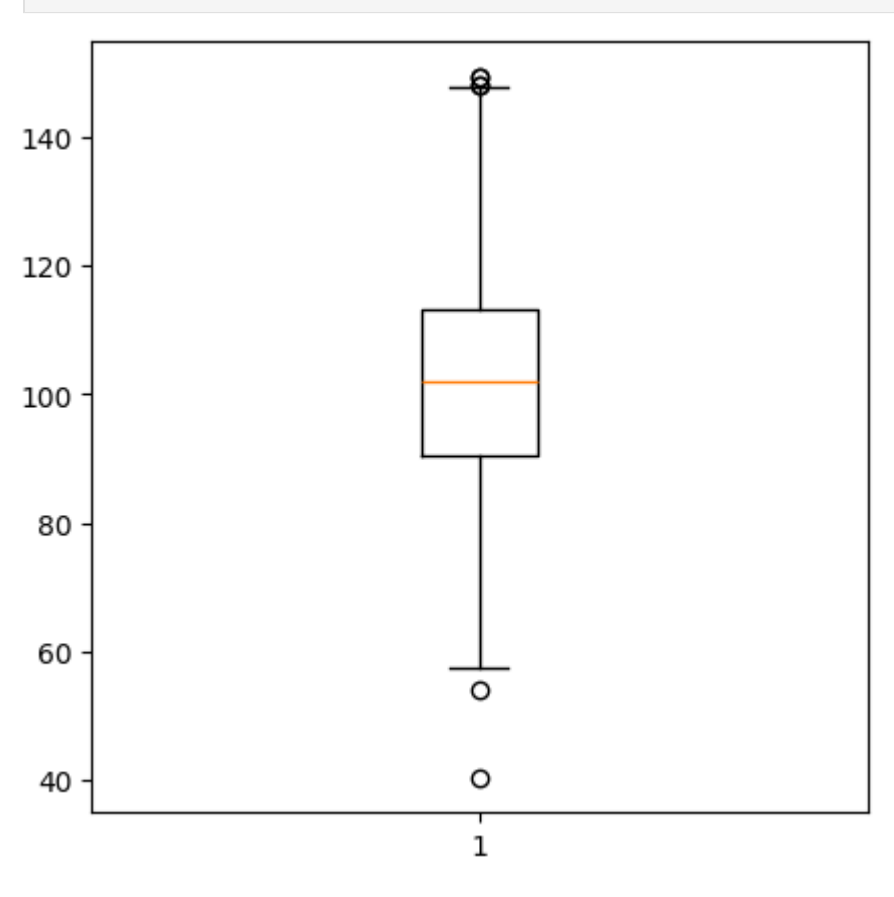
	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251
...	...	...	...	...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842	Sat2657
240	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404	Sat1766
241	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196	Sat3880
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950	Sat17
243	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139	Thur672

244 rows × 11 columns

• Types of Data Visualization:-

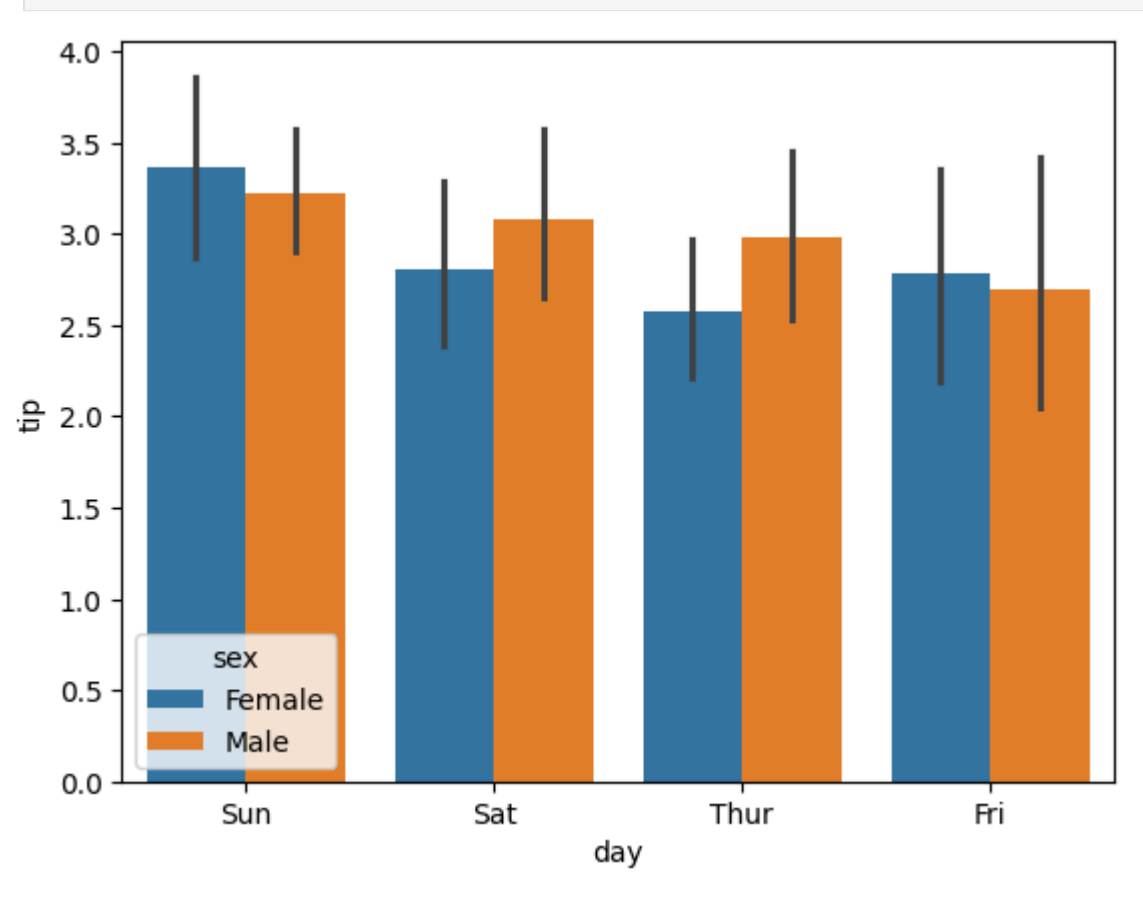
- **Box Plot:-** Box Plot is a graphical method to visualize data distribution for gaining insights and making informed decisions. Box plot is a type of chart that depicts a group of numerical data through their quartiles.

[2]:
```
#Box plot

np.random.seed(10)
d = np.random.normal(100, 20, 200)
fig = plt.figure(figsize =(5, 5))
plt.boxplot(d)
plt.show()
```



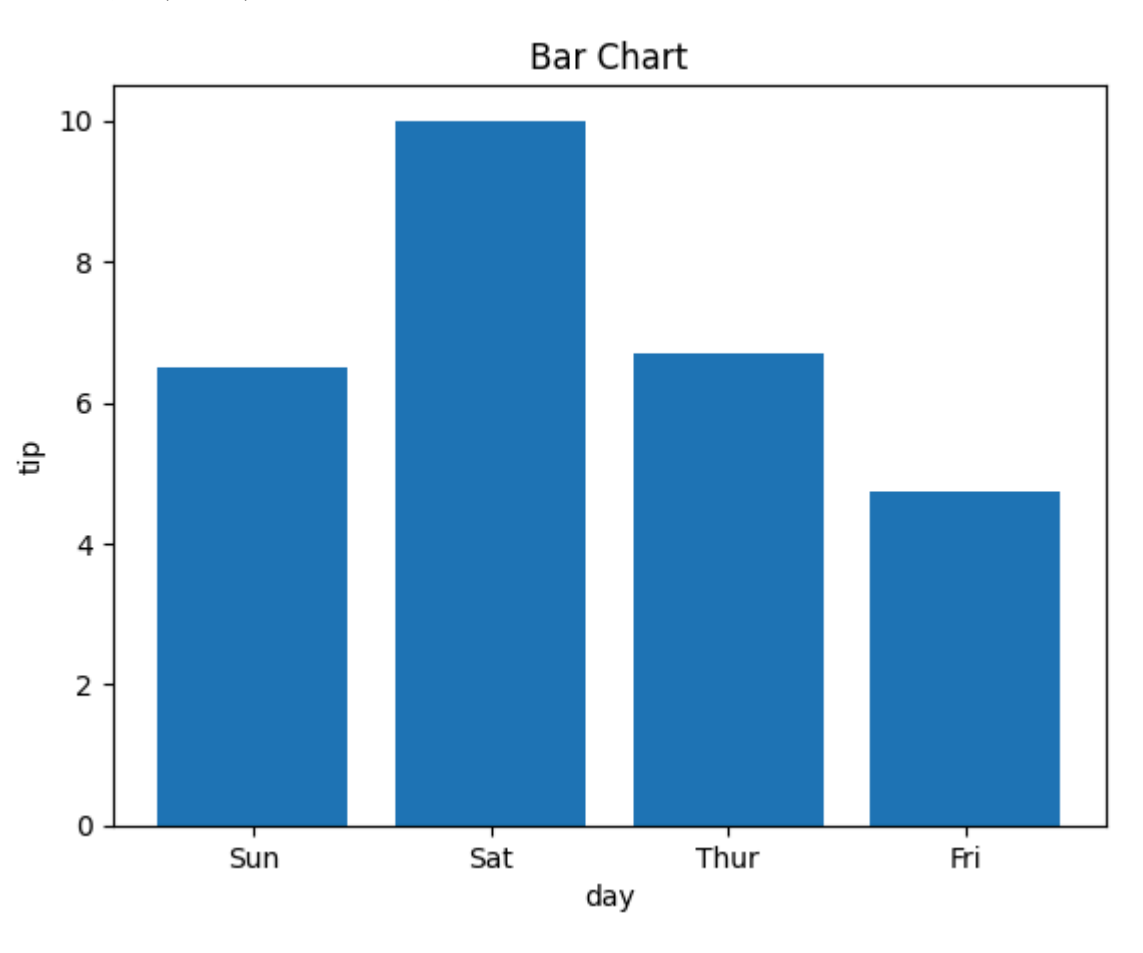
- **Bar Charts:-** Bar charts are used to compare values across different categories using rectangular bars. X-axis shows categories while Y-axis represents values. Common types include horizontal, stacked and grouped bar charts.

[3]:
```
#bar plot against day and tip

sns.barplot(x='day',y='tip', data=df,hue='sex')
plt.show()
```



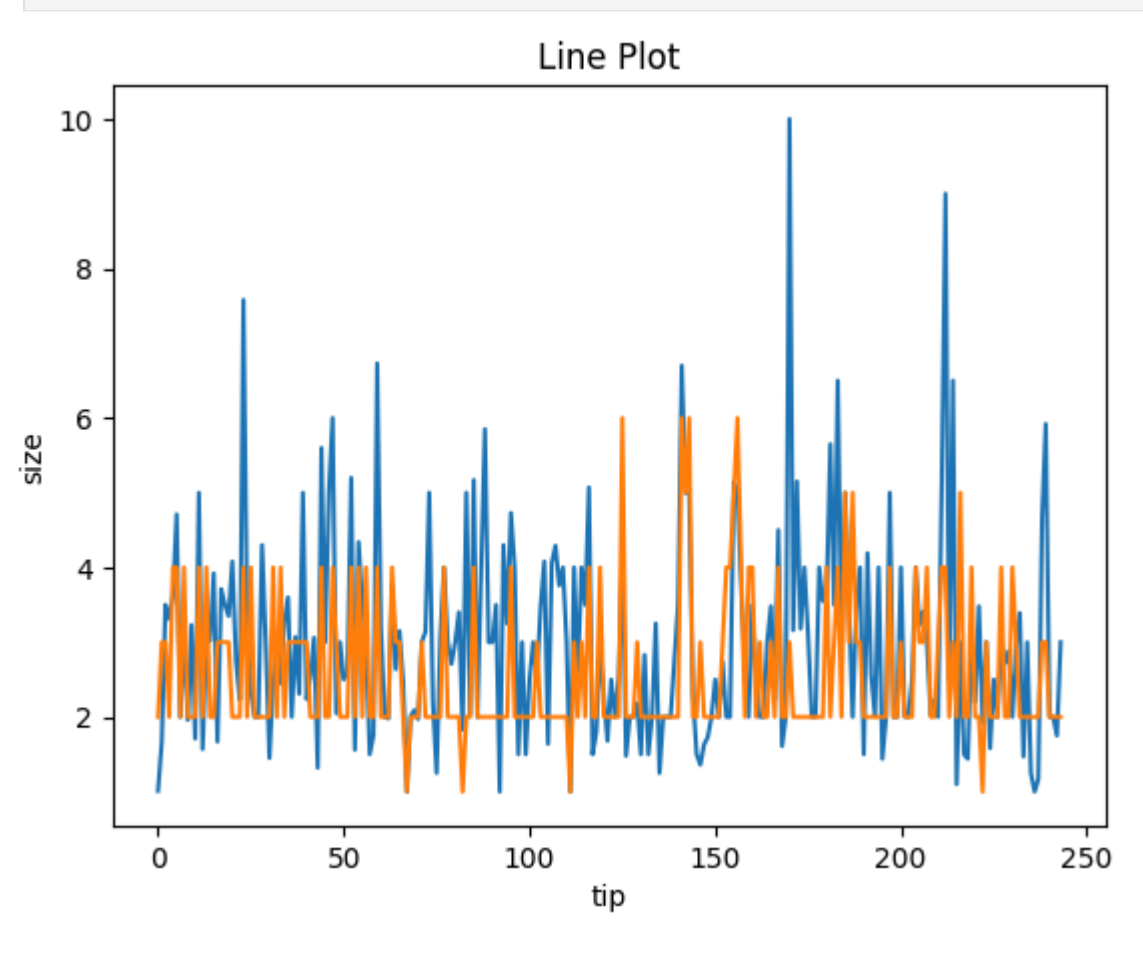
[4]:
```
#bar chart with day against tip


plt.bar(df["day"],df["tip"])
plt.xlabel("day")
plt.ylabel('tip')
plt.title('Bar Chart')
```

[4]: Text(0.5, 1.0, 'Bar Chart')



- **Line Charts:-**Line charts show how values change over time by connecting data points with lines. They help visualize trends like increases, decreases or stability.

[5]:
```
#Line plot with tip against size

plt.plot(df['tip'])
plt.plot(df['size'])
plt.title("Line Plot")
plt.xlabel("tip")
plt.ylabel('size')

plt.show()
```



- **Pie Charts:-**Pie charts are round charts divided into slices, where each slice shows a part of the whole. The size of each slice represents its percentage.

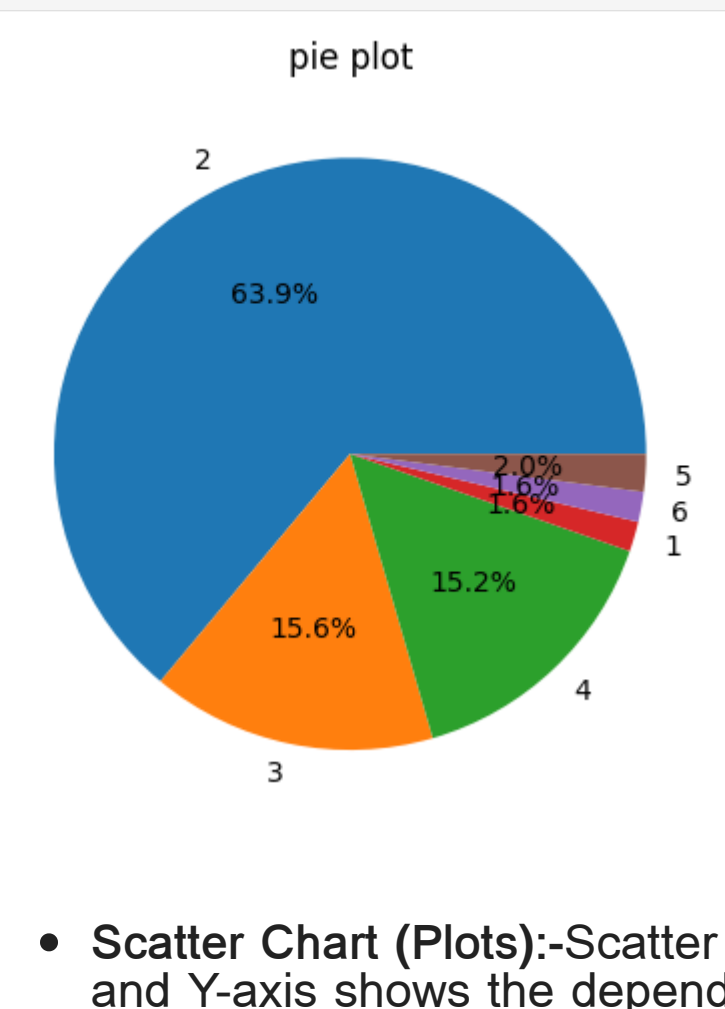
[10]:
```
df
```

[10]:

	total_bill	tip	sex	smoker	day	time	size	price_per_person	Payer Name	CC Number	Payment ID
0	16.99	1.01	Female	No	Sun	Dinner	2	8.49	Christy Cunningham	3560325168603410	Sun2959
1	10.34	1.66	Male	No	Sun	Dinner	3	3.45	Douglas Tucker	4478071379779230	Sun4608
2	21.01	3.50	Male	No	Sun	Dinner	3	7.00	Travis Walters	6011812112971322	Sun4458
3	23.68	3.31	Male	No	Sun	Dinner	2	11.84	Nathaniel Harris	4676137647685994	Sun5260
4	24.59	3.61	Female	No	Sun	Dinner	4	6.15	Tonya Carter	4832732618637221	Sun2251
...	...	...	...	...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3	9.68	Michael Avila	5296068606052842	Sat2657
240	27.18	2.00	Female	Yes	Sat	Dinner	2	13.59	Monica Sanders	3506806155565404	Sat1766
241	22.67	2.00	Male	Yes	Sat	Dinner	2	11.34	Keith Wong	6011891618747196	Sat3880
242	17.82	1.75	Male	No	Sat	Dinner	2	8.91	Dennis Dixon	4375220550950	Sat17
243	18.78	3.00	Female	No	Thur	Dinner	2	9.39	Michelle Hardin	3511451626698139	Thur672

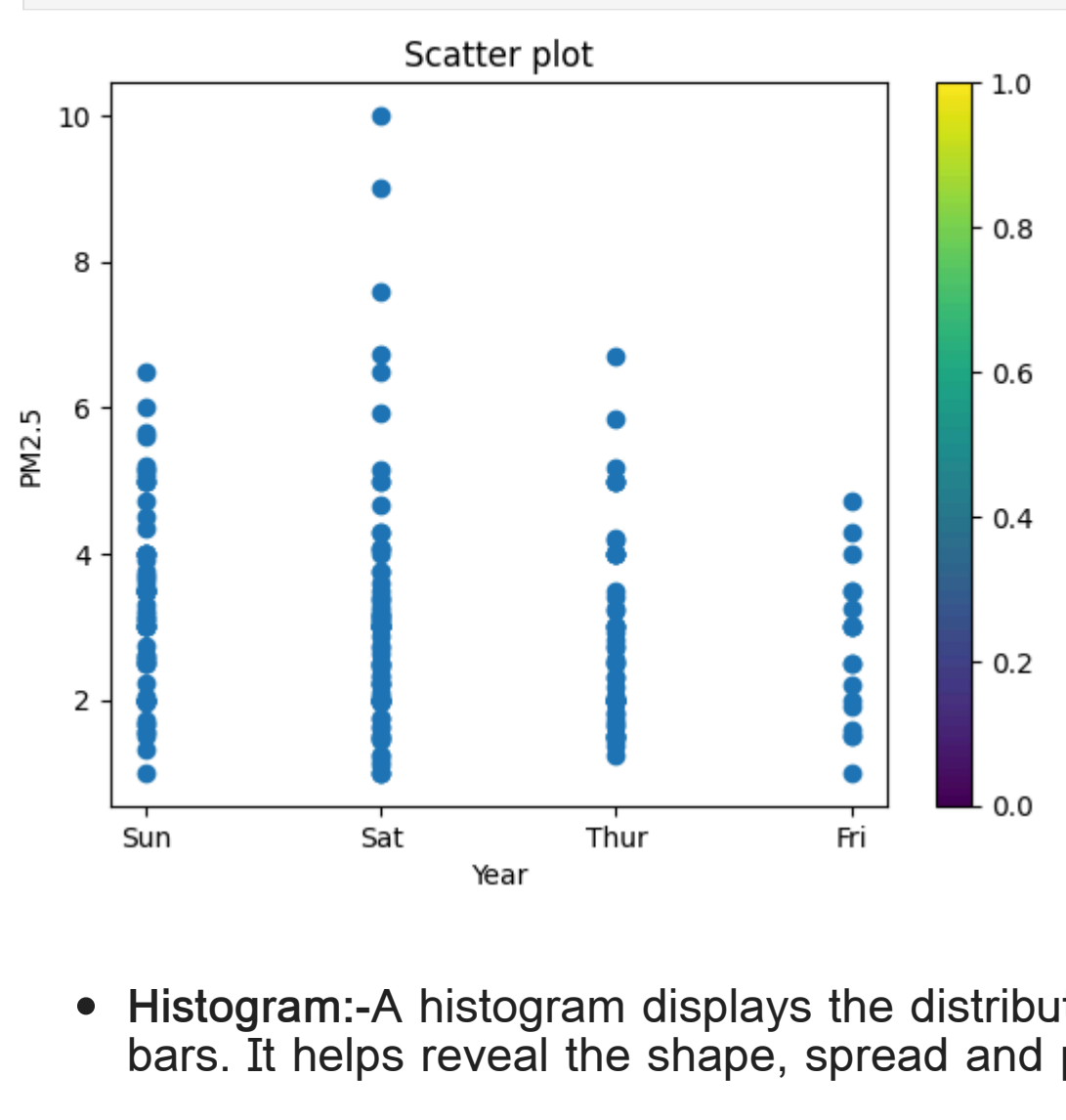
244 rows × 11 columns

[18]:
```
#pie chart

from collections import Counter
counts=Counter(df["size"])
label=list(counts.keys())
size=list(counts.values())

plt.pie(size,labels=label,autopct='%1.1f%%')
plt.title("pie plot")
plt.show()
```



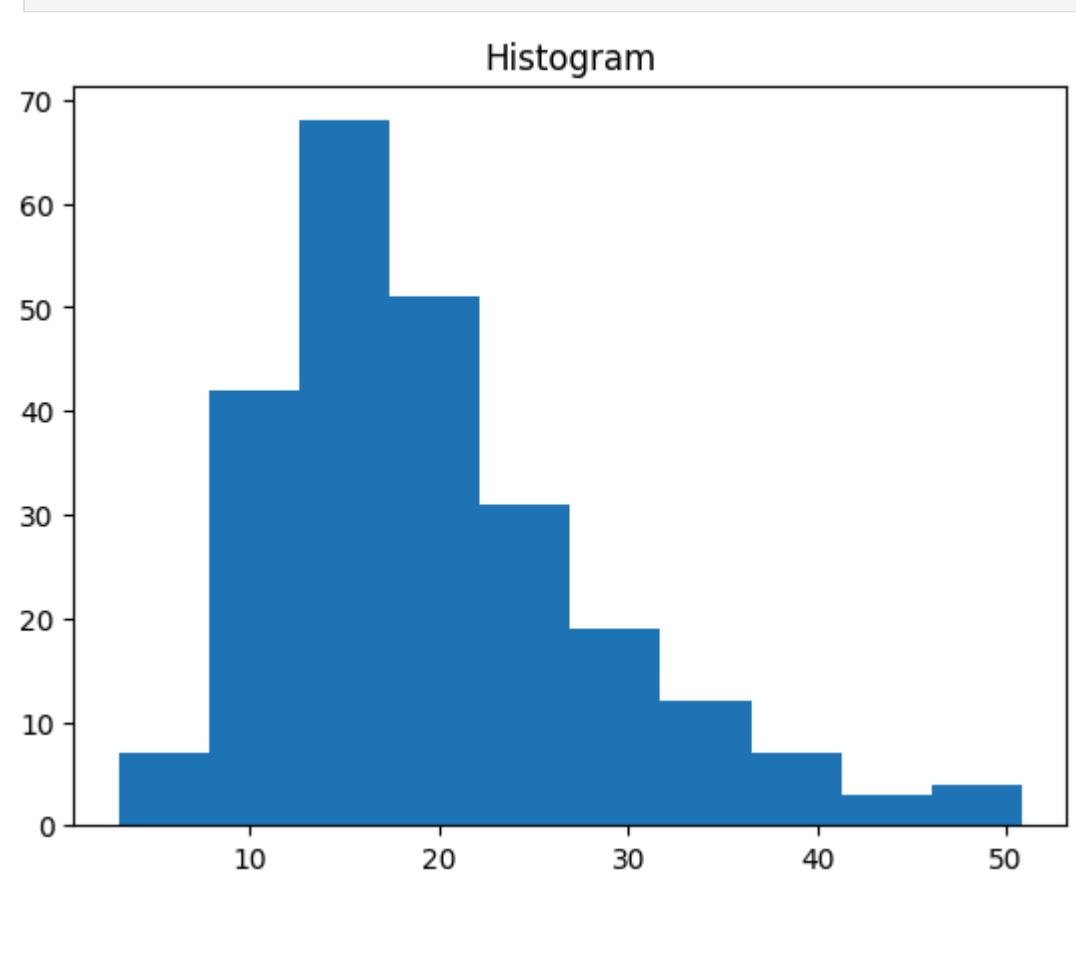
- **Scatter Chart (Plots):-**Scatter charts use dots to show relationship between two numerical variables. X-axis shows the independent variable and Y-axis shows the dependent variable.

[7]:
```
# Scatter plot with day against tip


plt.scatter(df["day"],df["tip"])
plt.title("Scatter plot")
plt.xlabel("Year")
plt.ylabel("PM2.5")
plt.colorbar()
plt.show()
```



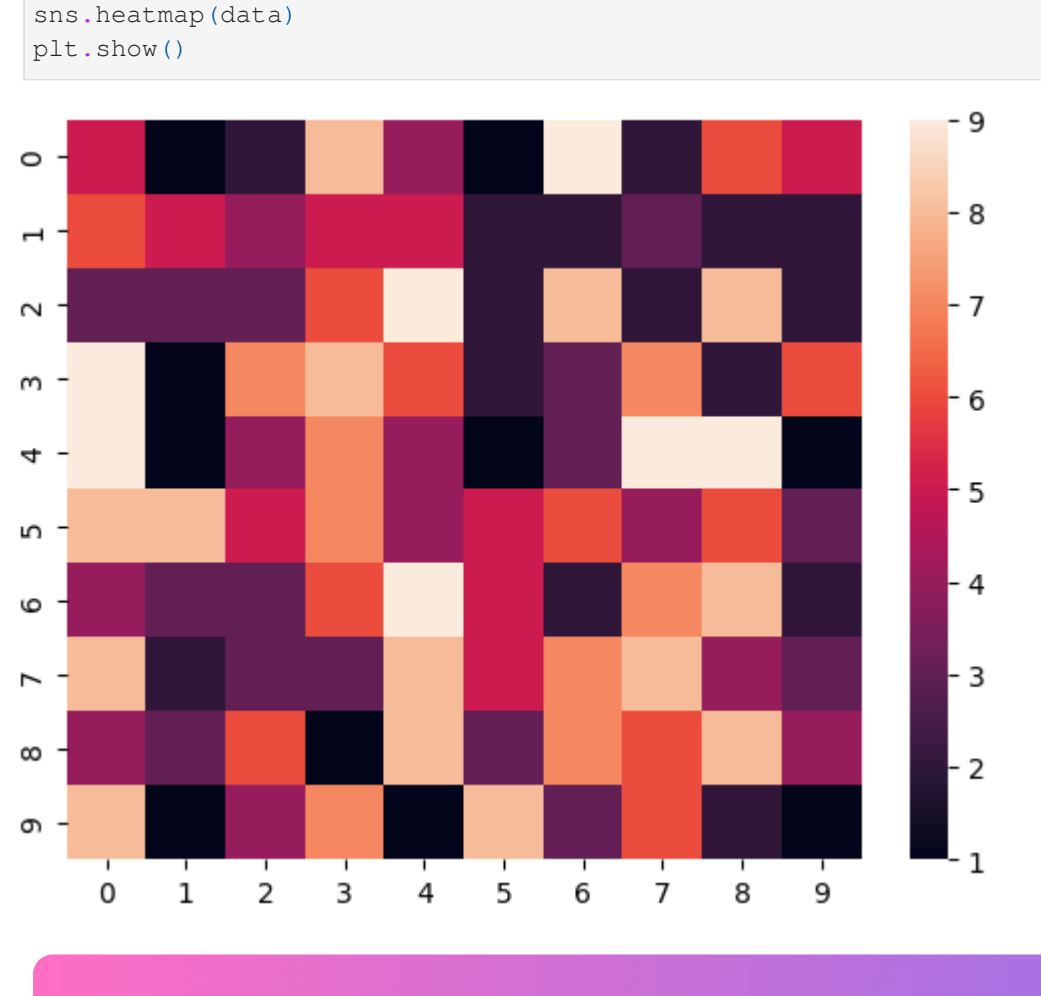
- **Histogram:-**A histogram displays the distribution of numerical data by grouping values into intervals (bins) and showing their frequency as bars. It helps reveal the shape, spread and patterns in the data.

[8]:
```
# histogram of total_bills

plt.hist(df['total_bill'])
plt.title("Histogram")
plt.show()
```



- **Heatmap:-** Heatmap data visualization is a technique that uses color to represent data values. The most common color schemes range from warm colors (such as red) to cool colors (such as blue), with warm colors typically representing higher values and cool colors representing lower values. This visual representation allows for quick and intuitive understanding of complex data sets.

[9]:
```
#heatmap

data = np.random.randint(1, 10, (10, 10))
sns.heatmap(data)
plt.show()
```



• Importance of Data Visualization:-

- **Simplifies Complex Data:**

It turns large and complicated data into visual formats like charts and graphs, making the information easier to understand.

- **Reveals Patterns and Trends:** It helps identify trends, relationships, and patterns that are not easily seen in raw data or tables.

- **Saves Time:** Visuals allow quicker interpretation of data, helping users spot key information at a glance instead of manually scanning through numbers.

- **Tells a Clear Story:** Data visuals guide the audience through the information step-by-step, making it easier to reach conclusions and make informed decisions.

• Real-World Use Cases for Data Visualization:-

- **Business Analytics:** Used to monitor company performance, track KPIs, and make data-driven decisions by visualizing trends, sales, and customer metrics.

- **Healthcare:** Helps in analyzing patient records, tracking disease outbreaks, and managing hospital operations through easy-to-read charts and dashboards.

- **Sports:** Used to visualize player statistics, team performance, and match outcomes, helping coaches and analysts improve strategies and training plans.

- **Retail and E-commerce:** Enables tracking of sales, customer preferences, and inventory levels, helping businesses adjust stock and marketing efforts effectively.

• Challenges in Data Visualization:-

- **Data Quality:** Accuracy of visualizations depends on the quality of the data. If the data is inaccurate or incomplete, the insights from the visualization will be misleading.

- **Choosing the Right Visualization:** Using the wrong type of visualization can distort the message. For example, a pie chart might not work well with many categories which leads to confusion.

- **Overload of Information:** Too much information in a visualization can overwhelm viewers. It's important to focus on key data points and avoid clutter.

____________ END ____________

THANKU YOU!