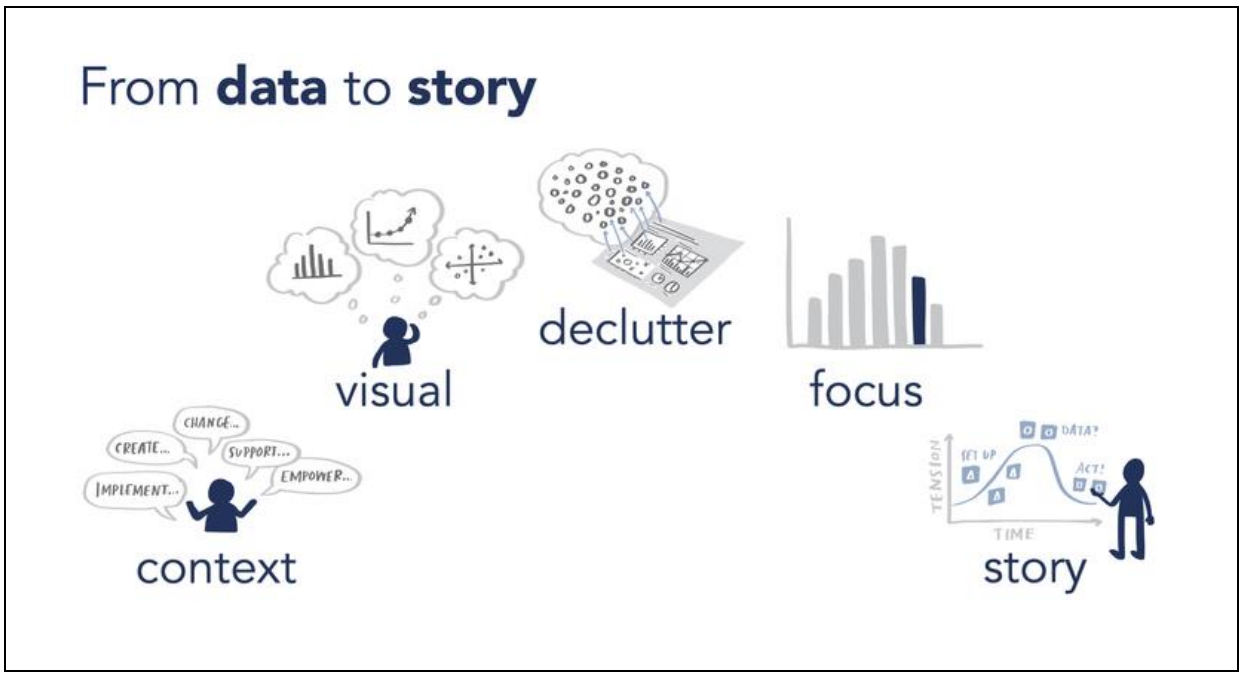
1. DATA VISUALIZATION PART – 1

Contents

1. DATA VISUALIZATION PART – 1



1. Data Visualization

- ✓ **Data Visualization** is the process of converting raw information (text, numbers, or symbols) into a graphical representation.
- ✓ If we visualize the data then it is very easy to understand.

Best quote

- ✓ A picture gives more meaningful information than thousand words

1.1. Example in words

- ✓ Reaching to target

2. Common data visualization techniques

- ✓ Bar charts
- ✓ Pie charts
- ✓ Line graphs
- ✓ Box plot
- ✓ Scatter plot & etc

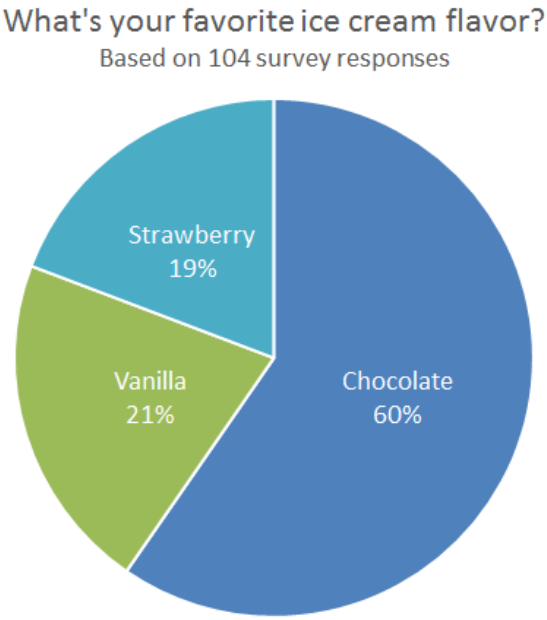
3. Advantages

- ✓ To identify trends, such as whether sales increasing or decreasing.
- ✓ To identify patterns, such as during weekend more sales.
- ✓ To identify relationships, such as if we study more hours then we will get good marks.
- ✓ To identify frequency, such as how often a product is purchased in a specific area & etc

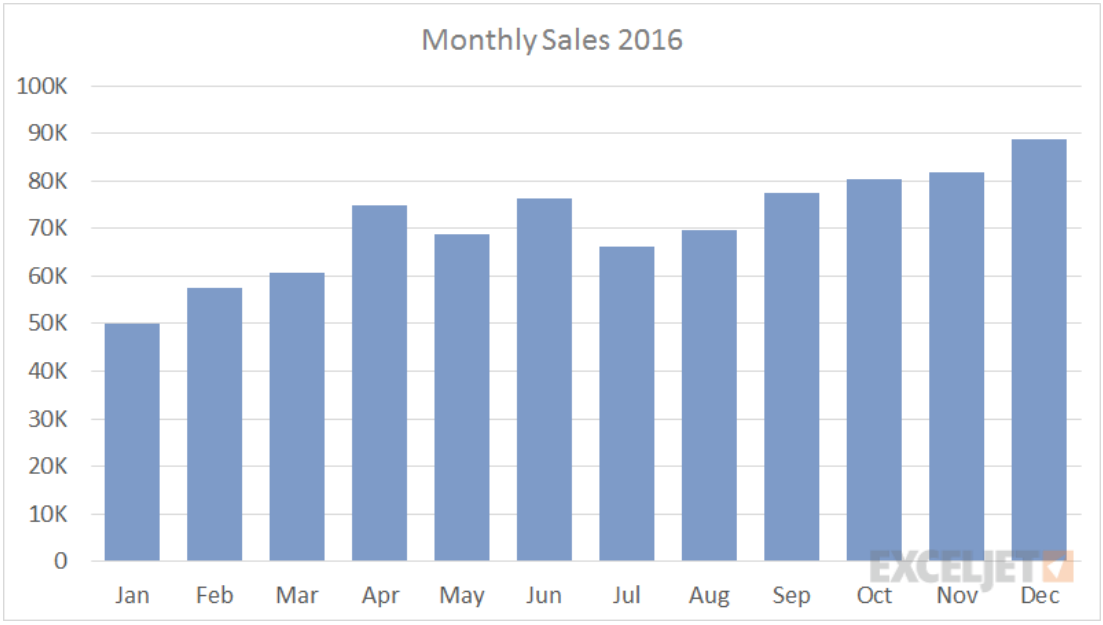
4. Few examples

What's your favorite ice cream flavor?

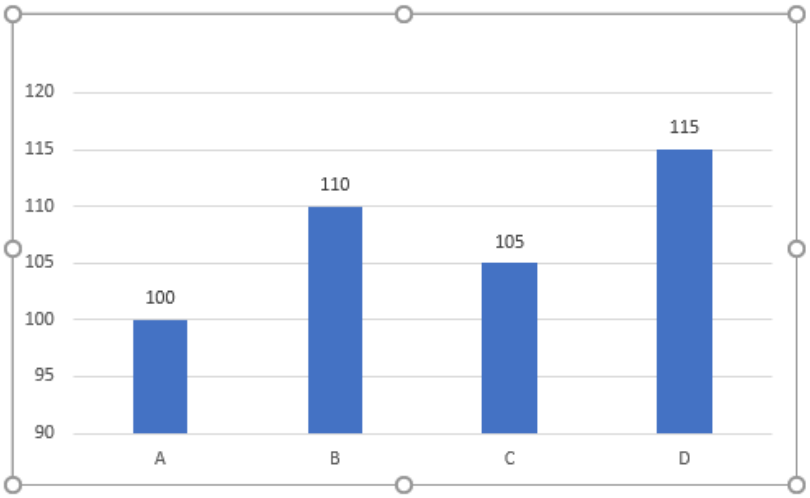
Flavor	Count
Chocolate	62
Vanilla	22
Strawberry	20



Bar chart

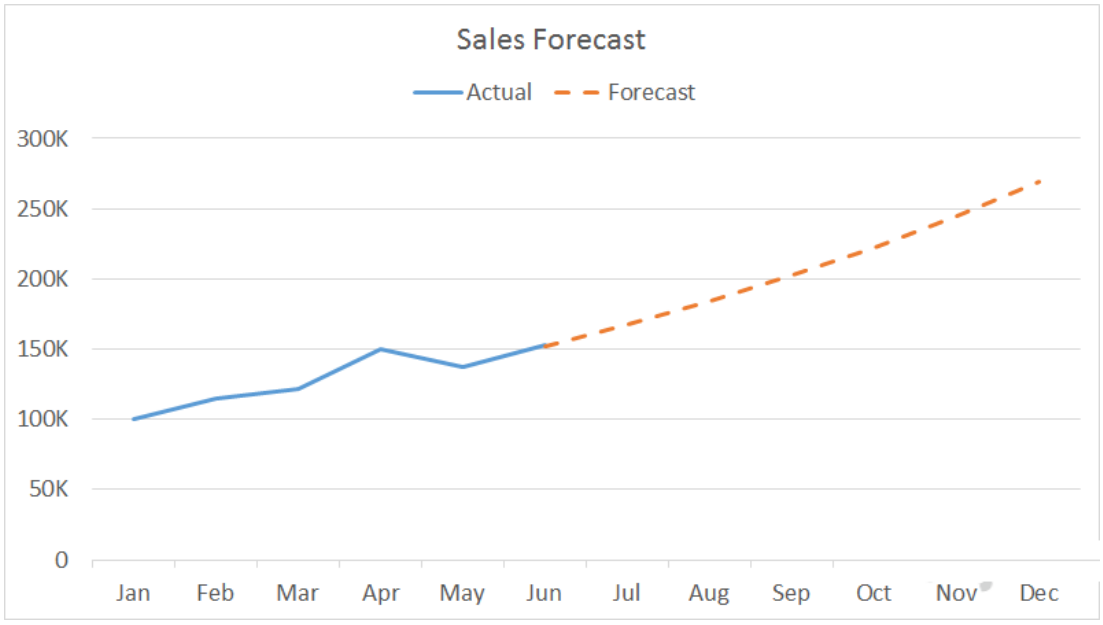


Group	Value
A	100
B	110
C	105
D	115



	Actual	Forecast
Jan	100K	
Feb	115K	
Mar	121K	
Apr	150K	
May	137K	
Jun	152K	152K
Jul		167K
Aug		184K
Sep		202K
Oct		223K
Nov		245K
Dec		269K

Line Chart



5. Matplotlib

- ✓ Matplotlib is the most popular plotting library in python.
- ✓ Using matplotlib we can plot the data.

Environment

- ✓ We can install this library by using pip command.

matplotlib installation

```
pip install matplotlib
```

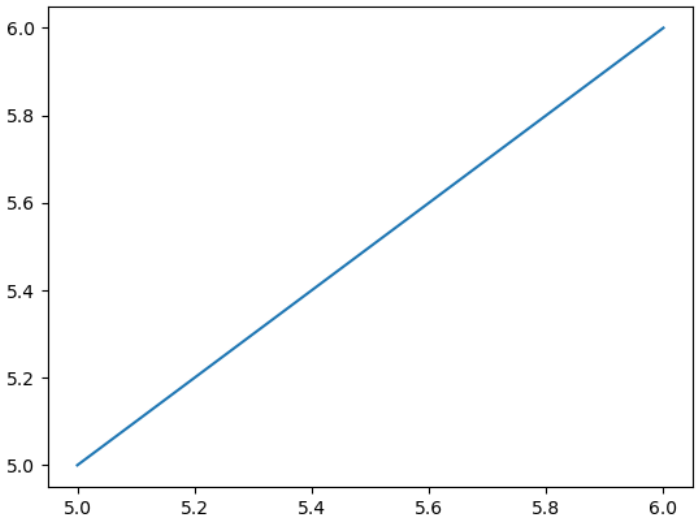
6. Line chart

- ✓ A line chart or line graph is a type of chart which displays information as a series of data points connected by straight line
- ✓ A line chart is often used to visualize a trend in data over intervals of time.

Program Name: Create a simple line chart
demo1.py

```
import matplotlib.pyplot as plt

x = [5, 6]
y = [5, 6]

plt.plot(x, y)

plt.show()
```

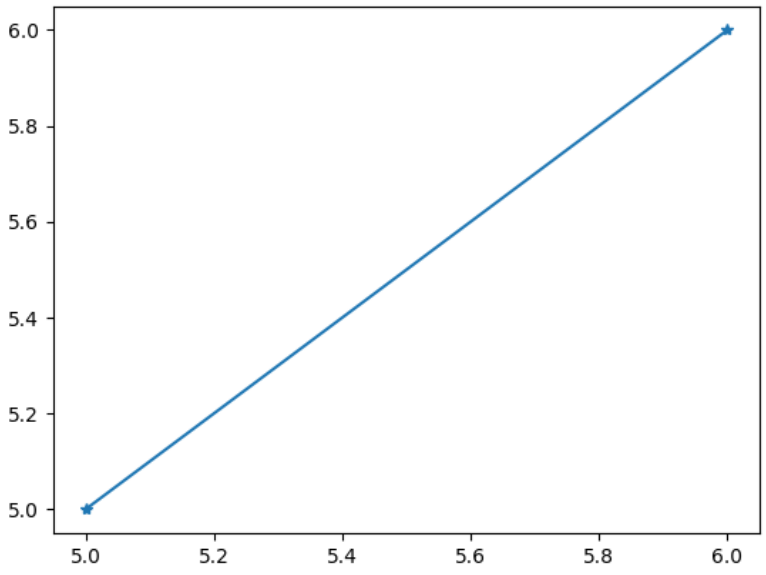
Output



Program Name: Create a simple line chart
demo2.py

```
import matplotlib.pyplot as plt

x = [5, 6]
y = [5, 6]

plt.plot(x, y, marker='*')

plt.show()
```

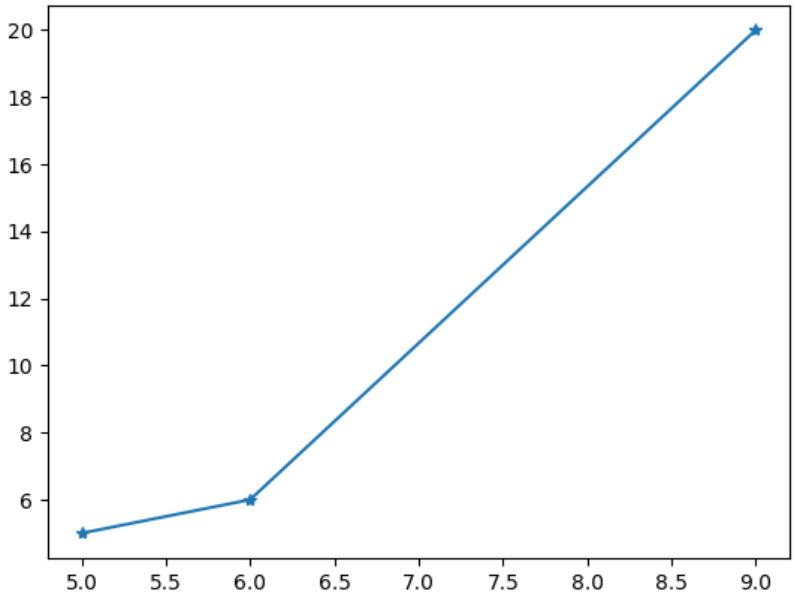
Output



Program Name: Create a simple line chart
demo3.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]

plt.plot(x, y, marker='*')

plt.show()
```

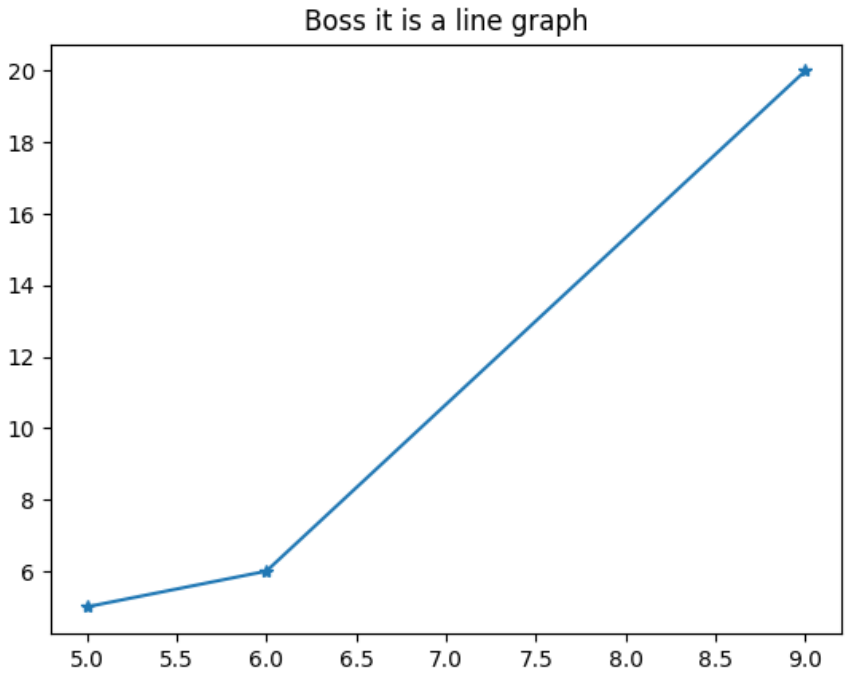
Output



Program Name: Create a simple line chart and title
demo4.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]

plt.title("Boss it is a line graph")

plt.plot(x, y, marker='*')

plt.show()
```

Output



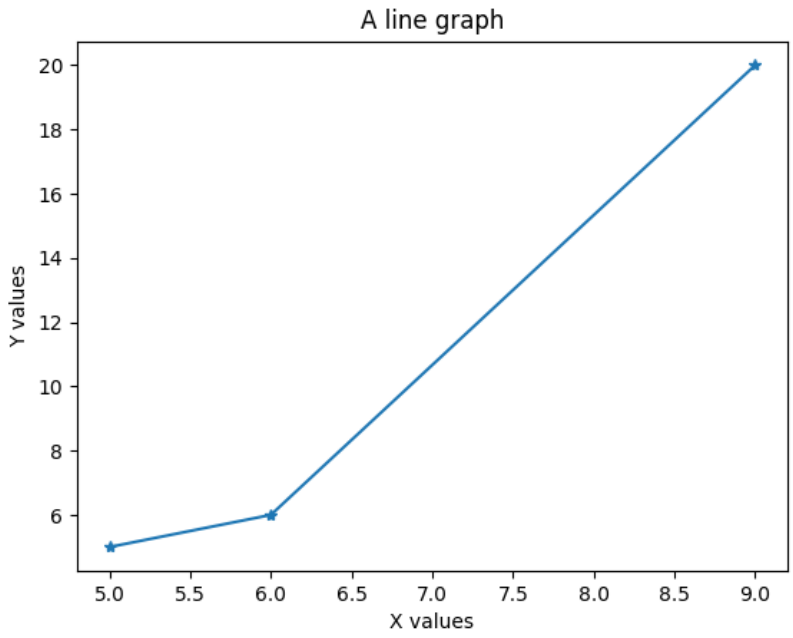
6.1. Labelling the axes

✓ We can label x axis and y axis by using xlabel and ylabel

Program Name: Create a simple line chart and giving title and labelling
demo5.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]

plt.title("A line graph")

plt.xlabel("X values")
plt.ylabel("Y values")

plt.plot(x, y, marker = '*')

plt.show()
```

Output

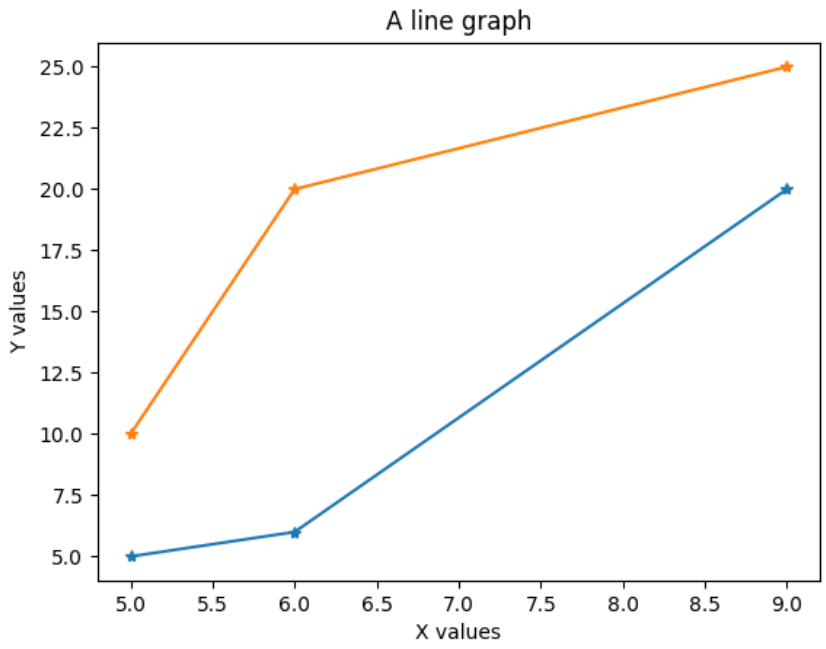


Program Name: Create two lines in single chart

demo6.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]
p = [10, 20, 25]

plt.title("A line graph")

plt.xlabel("X values")
plt.ylabel("Y values")

plt.plot(x, y, marker = '*')
plt.plot(x, p, marker = '*')

plt.show()
```

Output



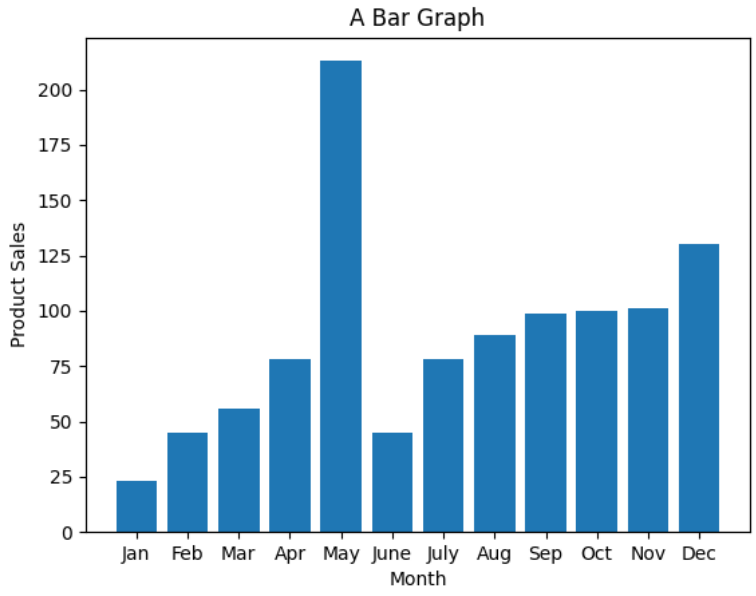
7. Bar Chart

✓ The bar graph is the graphical representation of categorical data.

Program Name: Creating bar chart
demo7.py

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"]
sales = [23, 45, 56, 78, 213, 45, 78, 89, 99, 100, 101, 130]

plt.title('A Bar Graph')

plt.xlabel('Month')
plt.ylabel('Product Sales')

plt.bar(months, sales)

plt.show()
```

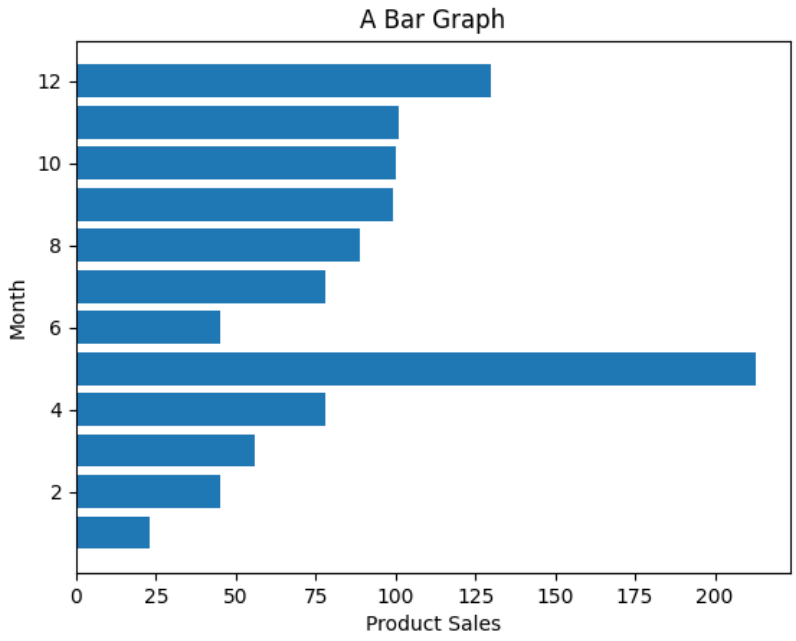
Output



Program Name: Creating horizontal bar chart
demo8.py

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"]
sales = [23, 45, 56, 78, 213, 45, 78, 89, 99, 100, 101, 130]

plt.title('A Bar Graph')

plt.xlabel('Product Sales')
plt.ylabel('Month')

plt.barh(months, sales)

plt.show()
```

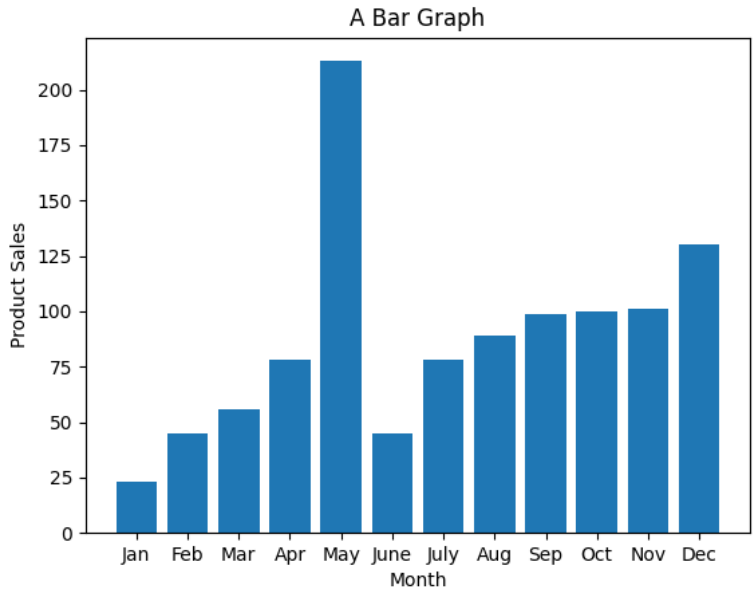
Output



Program Name: Creating horizontal bar chart
File name: demo9.py
File name: sales11.csv

```
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("sales11.csv")

plt.title('A Bar Graph')

plt.xlabel('Month')
plt.ylabel('Product Sales')

plt.bar(df.month, df.sales)

plt.show()
```

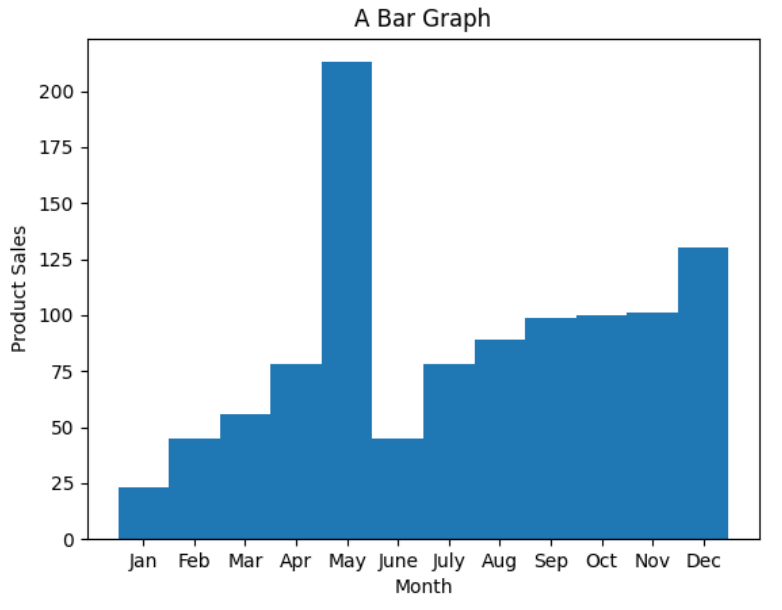
Output



Program Name: Creating bar chart
demo10.py

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"]
sales = [23, 45, 56, 78, 213, 45, 78, 89, 99, 100, 101, 130]

plt.title('A Bar Graph')

plt.xlabel('Month')
plt.ylabel('Product Sales')

plt.bar(months, sales, width = 1.0)

plt.show()
```

Output



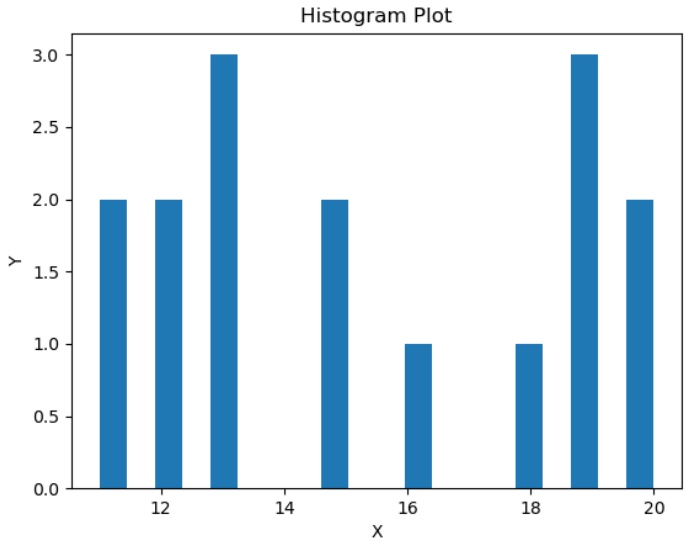
8. Histogram

- ✓ A histogram is the graphical representation of quantitative data.
- ✓ This displays the frequency/count of numerical data in bars.

Program Name: Creating histogram
demo11.py

```
import matplotlib.pyplot as plt

data = [12, 15, 13, 20, 19, 20, 11, 19, 11, 12, 19, 13, 15, 16, 18, 13]


plt.xlabel("X")
plt.ylabel("Y")
plt.title("Histogram Plot")

plt.hist(data, bins = 20)
plt.show()
```

Output



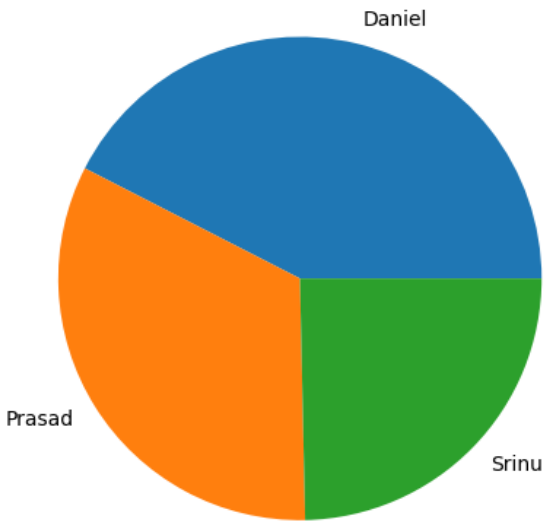
9. Pie Chart

- ✓ This is a circular plot that has been divided into slices displaying numerical proportions.
- ✓ Every slice in the pie chart shows the proportion of the element to the whole.
- ✓ A large category means that it will occupy a larger portion of the pie chart.

Program Name: Creating pie chart
demo12.py

```
import matplotlib.pyplot as plt

students = ['Daniel', 'Prasad', 'Srinu']
points = [62, 48, 36]

plt.pie(points, labels = students)

plt.axis('equal')
plt.show()
```

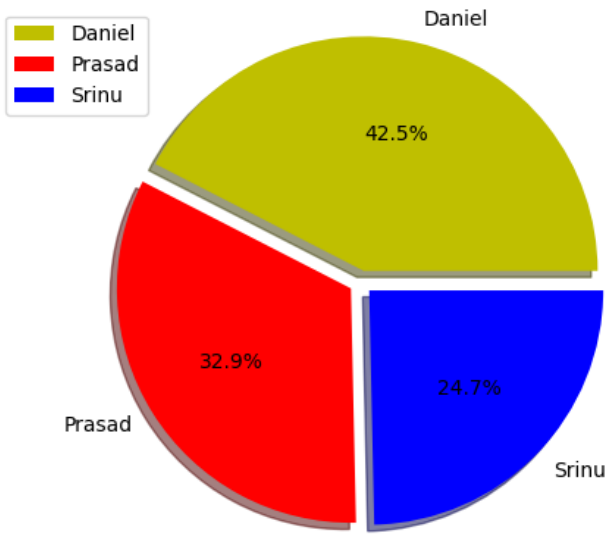
Output



Program Name: Creating pie chart
demo13.py

```
import matplotlib.pyplot as plt

students = ['Daniel', 'Prasad', 'Srinu']
points = [62, 48, 36]

c = ['y', 'r', 'b']

plt.pie(points, labels = students, colors = c , shadow = True,
explode = (0.05, 0.05, 0.05), autopct = '%1.1f%%')

plt.axis('equal')
plt.legend()
plt.show()
```

Output



9.1. Attributes

- ✓ The first parameter to the function is the list of numbers for every category.
  - labels attribute:
    - A list of categories separated by commas is then passed as the argument to labels attribute.
  - colors attribute:
    - To provide the color for every category.
  - To create shadows around the various categories in pie chart.
  - To split each slice of the pie chart into its own.

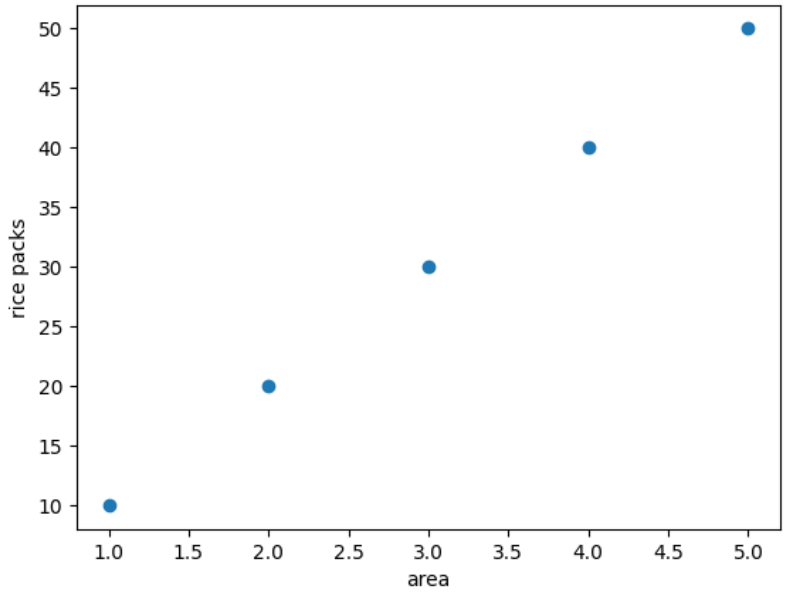
10. Scatter Plot

- ✓ In scatter plot each value in the data set is represented by a dot.
- ✓ By using this plot we can understand the relationship between two variables.

Program Name: Creating Scatter plot
demo14.py

```
import matplotlib.pyplot as plt

area = [1, 2, 3, 4, 5]
rice_packs = [10, 20, 30, 40, 50]

plt.xlabel('area')
plt.ylabel('rice packs')

plt.scatter(area, rice_packs)
plt.show()
```

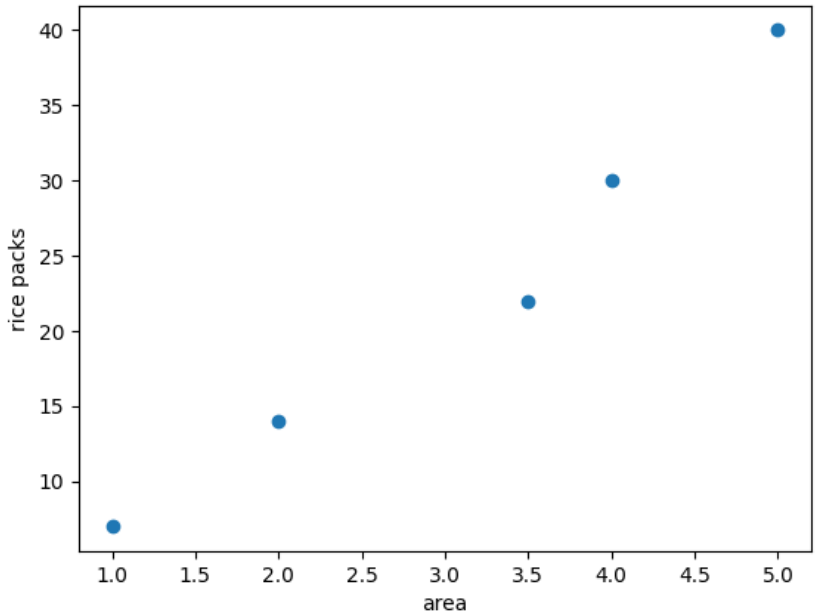
Output



Program Name: Creating Scatter plot
demo15.py

```
import matplotlib.pyplot as plt

area = [1, 2, 3.5, 4, 5]
rice_packs = [7, 14, 22, 30, 40]

plt.xlabel('area')
plt.ylabel('rice packs')

plt.scatter(area, rice_packs)
plt.show()
```

Output



11. Box Plots

- ✓ Box plots help us measure how well data in a dataset is distributed.
- ✓ The graph shows the maximum, minimum, median, first quartile and third quartiles of the dataset.

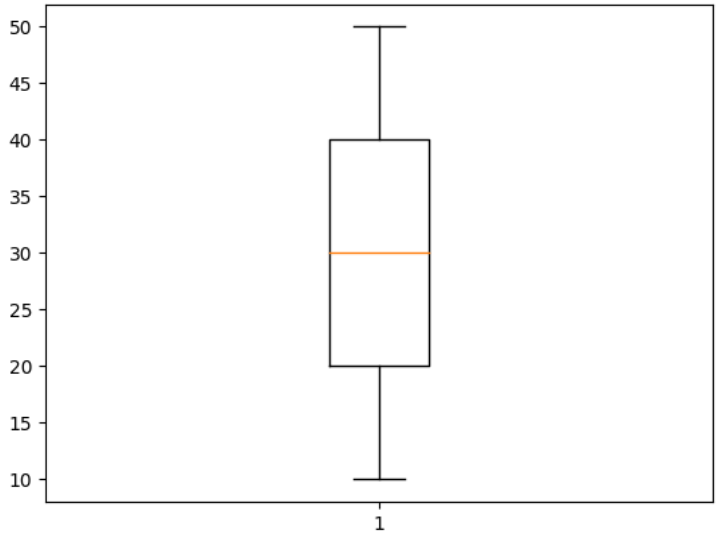
11.1. Use Box plots

- ✓ Use a boxplot when you need to get the overall statistical information about the data distribution.
- ✓ It is a good tool for detecting outliers in a dataset.

Program Name: Creating box plot
demo16.py

```
import matplotlib.pyplot as plt

data = [10, 20, 30, 40, 50]

plt.boxplot(data)
plt.show()
```

Output



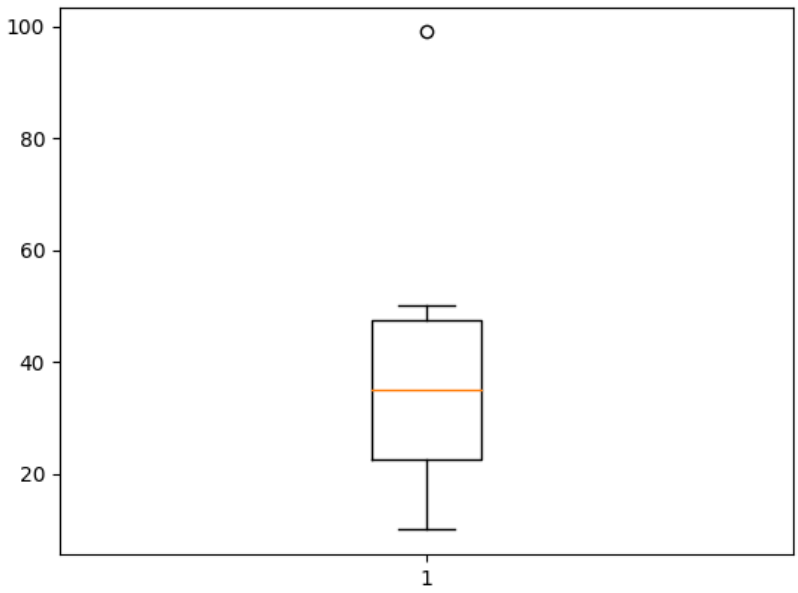
11.2. Box plot explanation

- ✓ The line dividing the box into two shows the median of the data.
- ✓ The end of the box represents the upper quartile (75%) while the start of the box represents the lower quartile (25%).
- ✓ The part between the upper quartile and the lower quartile is known as the Inter Quartile Range (IQR) and helps in approximating 50% of the middle data.

Program Name: Creating box plot with outlier
demo17.py

```
import matplotlib.pyplot as plt

data = [10, 20, 30, 40, 50, 90]

plt.boxplot(data)
plt.show()
```

Output



12. Heatmap

- ✓ A heatmap is a method of data visualization that plots data by replacing numbers with colours.
- ✓ If it is representing with color then it is very easy to understand patterns between different values in the dataset.
- ✓ It is used to visualize data in a two-dimensional format as a coloured map so that different colour variations represent different patterns between features.

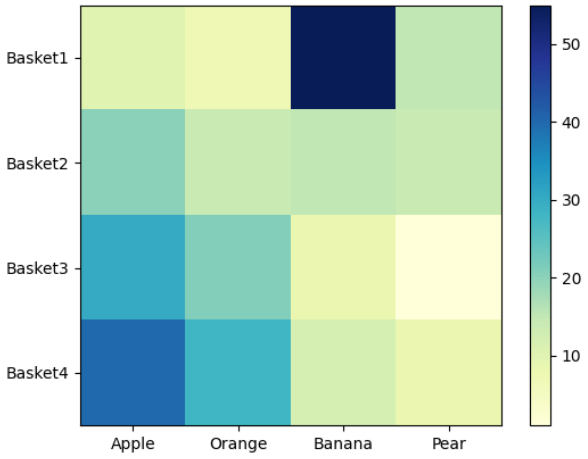
12.1. How to understand?

- ✓ A heatmap visualizes the relationship between features as a colour palette.
- ✓ While analysing a heatmap, always remember that **dark shades** represent a **high degree** of linear relationship between features and **light shades** represent a **low degree** of linear relationship between features.

Program Name: Creating box plot
demo18.py

```
import matplotlib.pyplot as plt
import pandas as pd

d = {
    "Apple": [10, 20, 30, 40],
    "Orange": [7, 14, 21, 28],
    "Banana": [55, 15, 8, 12],
    "Pear": [15, 14, 1, 8]
}

i = ['Basket1', 'Basket2', 'Basket3', 'Basket4']

df = pd.DataFrame(d, index = i)

plt.imshow(df, cmap = "YlGnBu")
plt.colorbar()

plt.xticks(range(len(df)), df.columns)
plt.yticks(range(len(df)), df.index)

plt.show()
```

Output

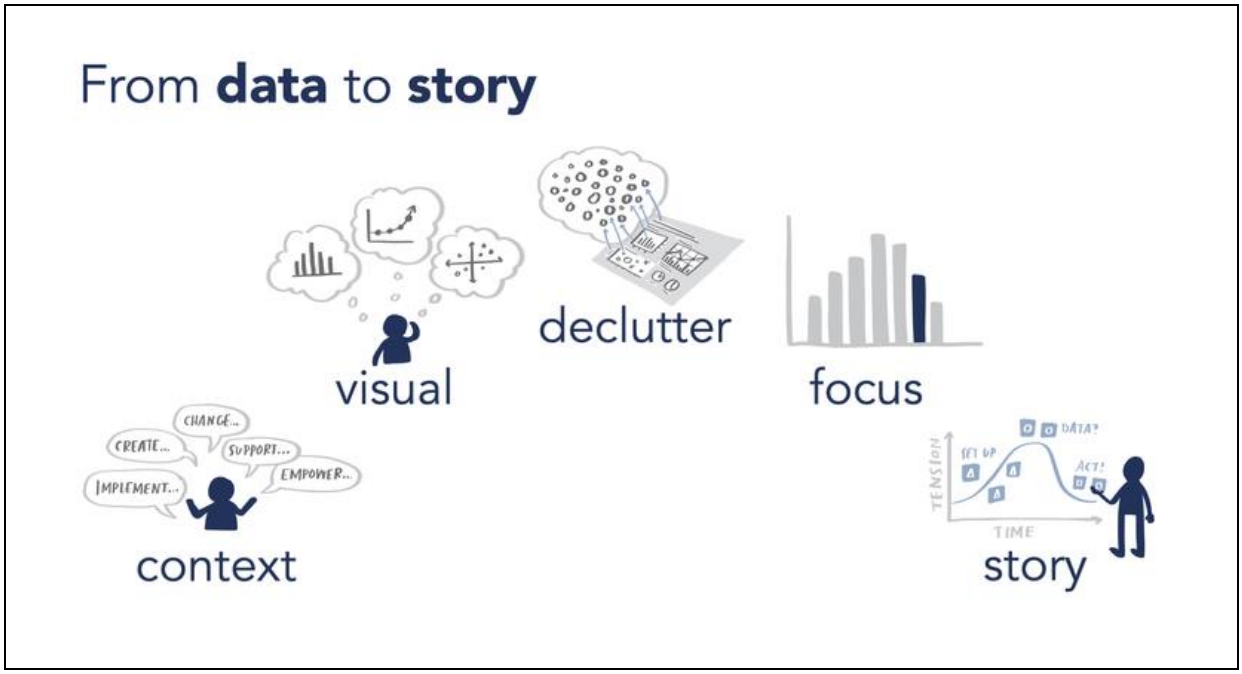


Data Visualization

Contents

Data Visualization

Data Visualization



1. Data Introduction

- ✓ Currently we are all living in the data world.
- ✓ Everyone is communicating by using devices and social networks, due to this huge amount of data is generating.
- ✓ All applications are generating data.
  - Ecommerce applications.
  - Banking applications.
  - Social network etc.

2. What is Data?

- ✓ Data is a collection of Facts.
- ✓ Facts can be,
  - Numbers
  - Alphabets
  - Alphanumeric
  - Symbols
  - Images
  - Audio
  - Video & etc

3. Data Visualization

- ✓ **Data Visualization** is the process of converting data into a graphical representation.
- ✓ If we visualize the data then it is very easy to understand.

Best quote

- ✓ A picture gives more meaningful information than thousand words

1.1. Example in words

- ✓ Reaching to target

4. Advantages

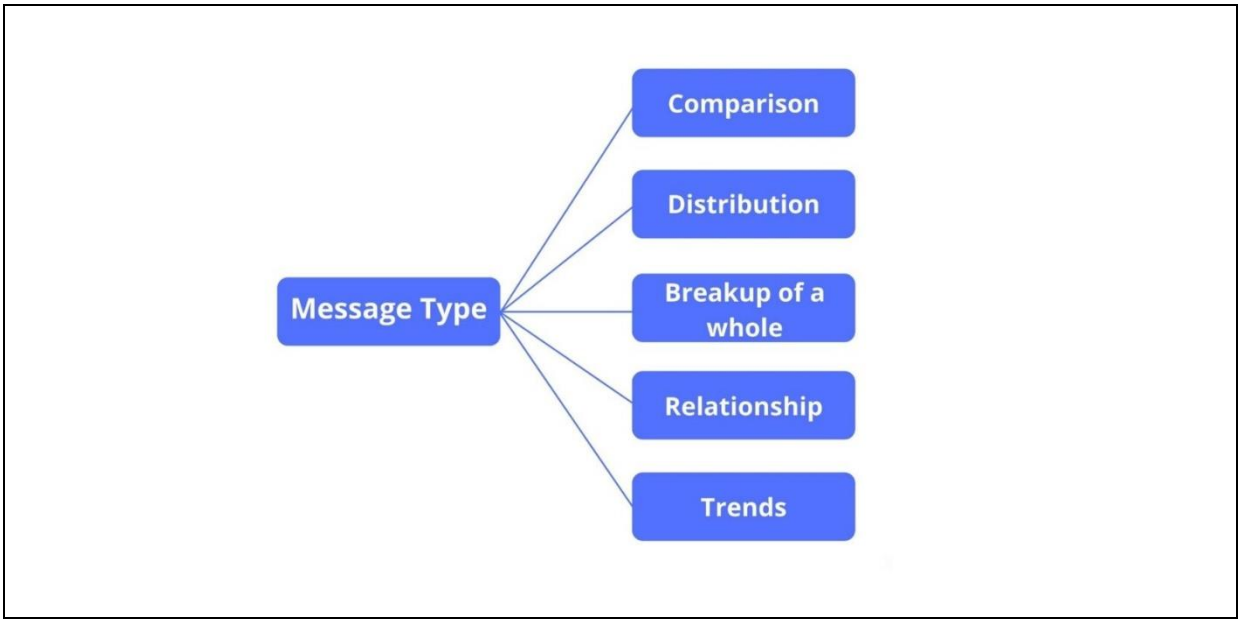
- ✓ To identify **trends**, such as whether sales increasing or decreasing.
- ✓ To identify **patterns**, such as during weekend more sales.
- ✓ To identify **relationships**, such as if we study more hours then we will get good marks.
- ✓ To identify **frequency**, such as how often a product is purchased in a specific area & etc

4.1. Common data visualization techniques

- ✓ Bar charts
- ✓ Pie charts
- ✓ Line graphs
- ✓ Box plot
- ✓ Scatter plot & etc

4.2. With data visualization

- ✓ We are sharing message/information to end users.



5. Real time Examples

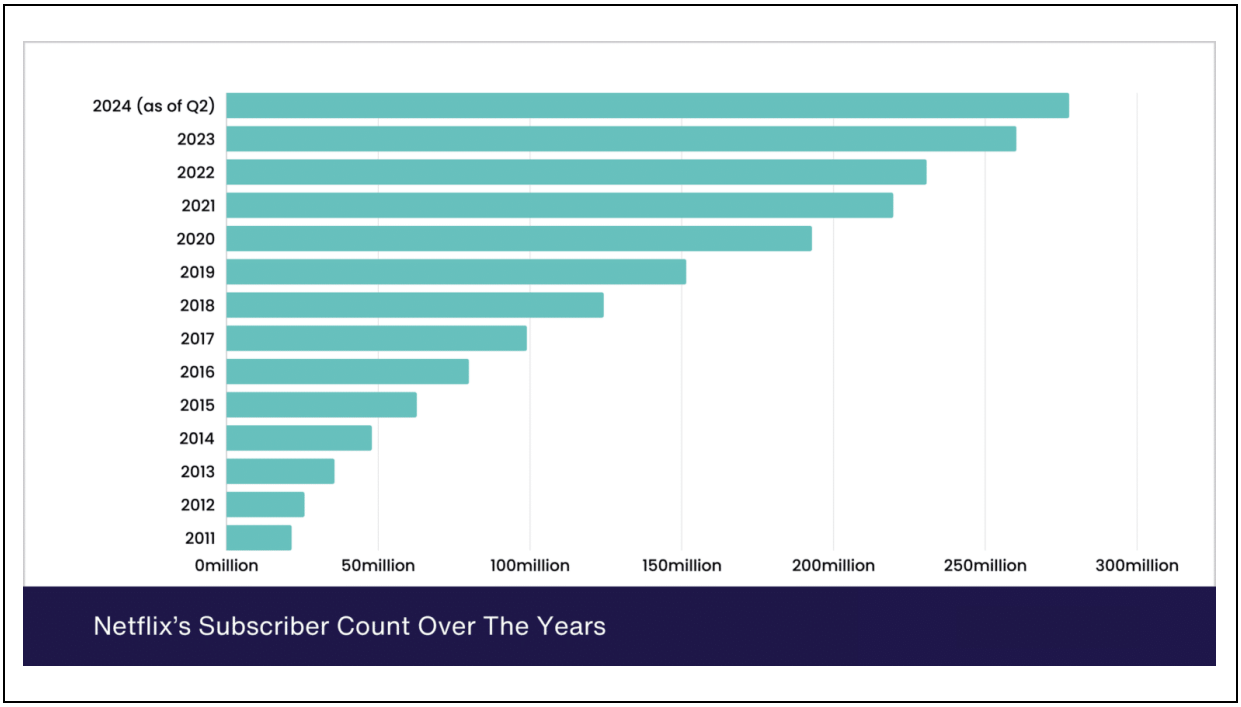
5.1. Data & Visual: Netflix Subscribers

Ref Link: https://www.demandsage.com/netflix-subscribers/

Top Netflix Statistics At A Glance

- Netflix has 277.65 million subscribers as of 2024.
- Netflix generated $18.93 billion in revenue in the first half of 2023.
- Women make up 51%, while Males make up 49% of all Netflix users.
- Netflix is preferred by 47% of Americans over other streaming platforms and is responsible for 8.4% of the screen time in the country.
- Around 65% of Netflix consumers are from outside of the United States of America & Canada.
- Netflix customers spend 62.1 minutes each day on average consuming content.

Year	Netflix Subscribers
2024 (as of Q2)	277.65 million
2023	260.28 million
2022	230.7 million
2021	219.7 million
2020	192.9 million
2019	151.5 million
2018	124.3 million
2017	99 million
2016	79.9 million
2015	62.7 million
2014	47.9 million
2013	35.6 million
2012	25.7 million
2011	21.5 million


2024 (as of Q2)
2023
2022
2021
2020
2019
2018
2017
2016
2015
2014
2013
2012
2011
0million
50million
100million
150million
200million
250million
300million
Netflix's Subscriber Count Over The Years

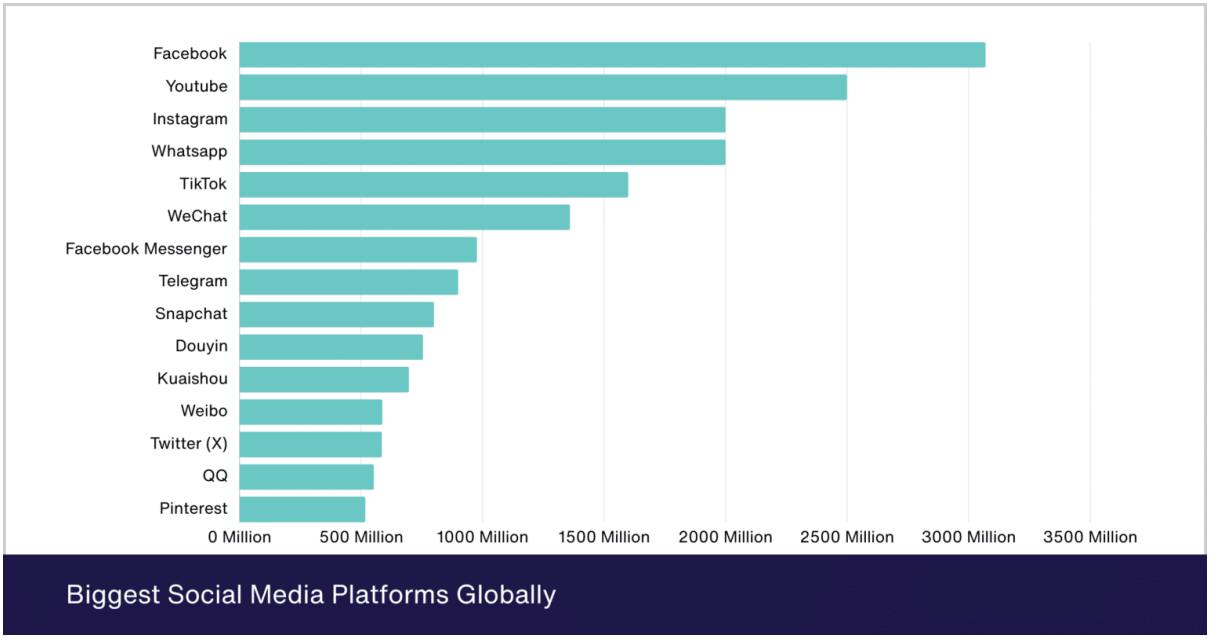
5.2. Data & Visual: Top Social Media Usage Statistics 2024

Ref Link: https://www.demandsage.com/social-media-users/

Top Social Media Usage Statistics 2024

- There are 5.17 billion social media users globally.
- 68% of the people in the United States use social media, approximately 308 million people.
- Facebook is the biggest social media platform, with over 3.07 billion users.
- A typical social media user interacts with 6.7 social media platforms.
- On average, users spend 2 hours and 20 minutes daily on Social media platforms.
- China has the highest number of social media users, with 1.07 billion users in the country.

Most Popular Social Media Platforms



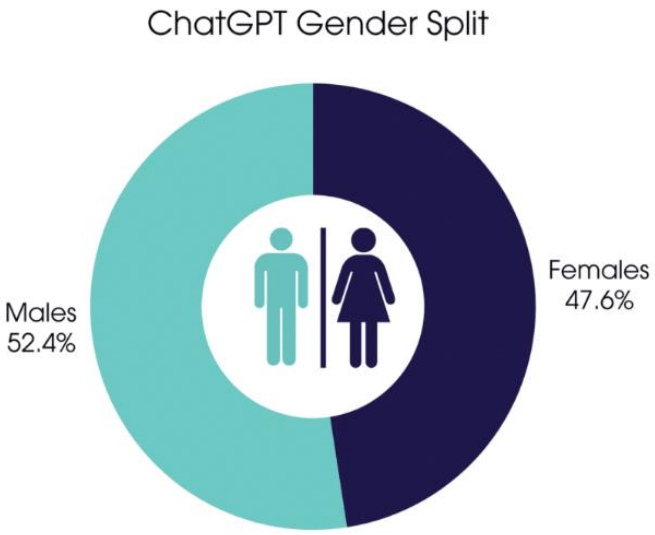
5.3. Data & Visual: Top ChatGPT Statistics (2024)

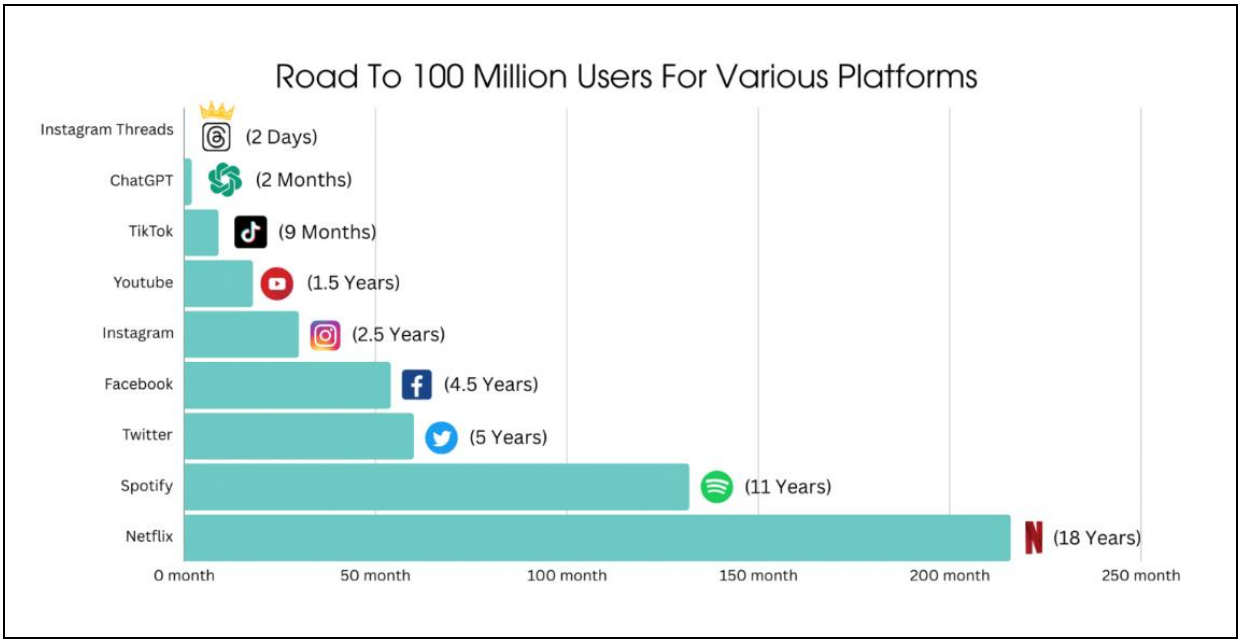
Ref Link: https://www.demandsage.com/chatgpt-statistics/

Top ChatGPT Statistics (2024)

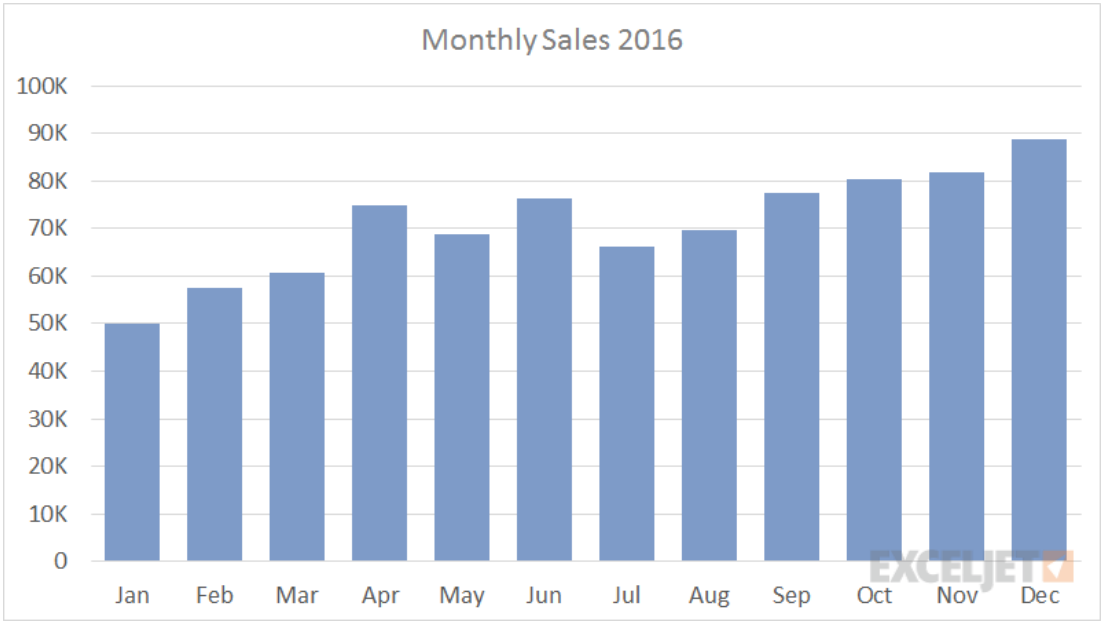
- ChatGPT has over **200 million weekly active users as of September 2024.**
- Around 77.2 million monthly active users in the US.
- ChatGPT Plus is used by 7.7 million people worldwide.
- ChatGPT reached 1 million users in just five days after its launch.
- More than 92% of Fortune 500 companies are using ChatGPT.
- ChatGPT is forecasted to generate a revenue of $1 billion in 2024.
- OpenAI spends approximately $700,000 every day to operate ChatGPT.
- ChatGPT gets over 1.54 billion page visits every month on average.

ChatGPT User Demographics

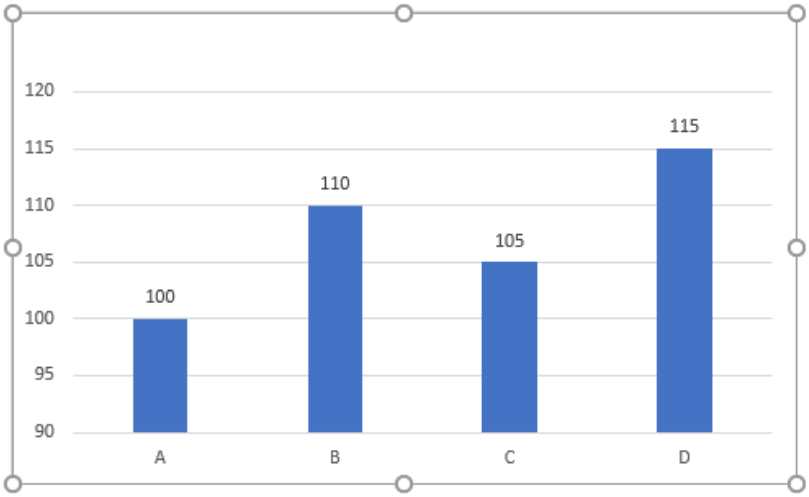



Road To 100 Million Users For Various Platforms
Instagram Threads
(2 Days)
ChatGPT
(2 Months)
TikTok
(9 Months)
Youtube
(1.5 Years)
Instagram
(2.5 Years)
Facebook
(4.5 Years)
Twitter
(5 Years)
Spotify
(11 Years)
Netflix
(18 Years)
0 month
50 month
100 month
150 month
200 month
250 month

Bar chart



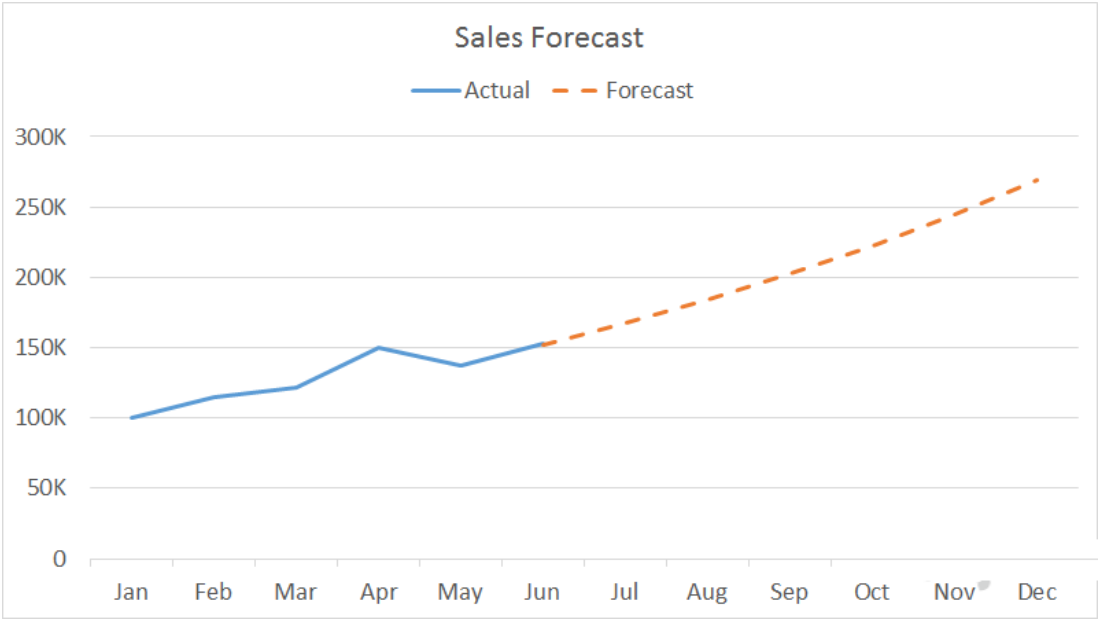
Group	Value
A	100
B	110
C	105
D	115



Sales Data

	Actual	Forecast
Jan	100K	
Feb	115K	
Mar	121K	
Apr	150K	
May	137K	
Jun	152K	152K
Jul		167K
Aug		184K
Sep		202K
Oct		223K
Nov		245K
Dec		269K

Line Chart



6. Process behind the Data Visualization

Steps

- ✓ Data collection
- ✓ Data cleaning
- ✓ Data analysis
- ✓ Chose the right visualization
- ✓ Creating visual representation
- ✓ Review and Iterative

6.1. Data collection

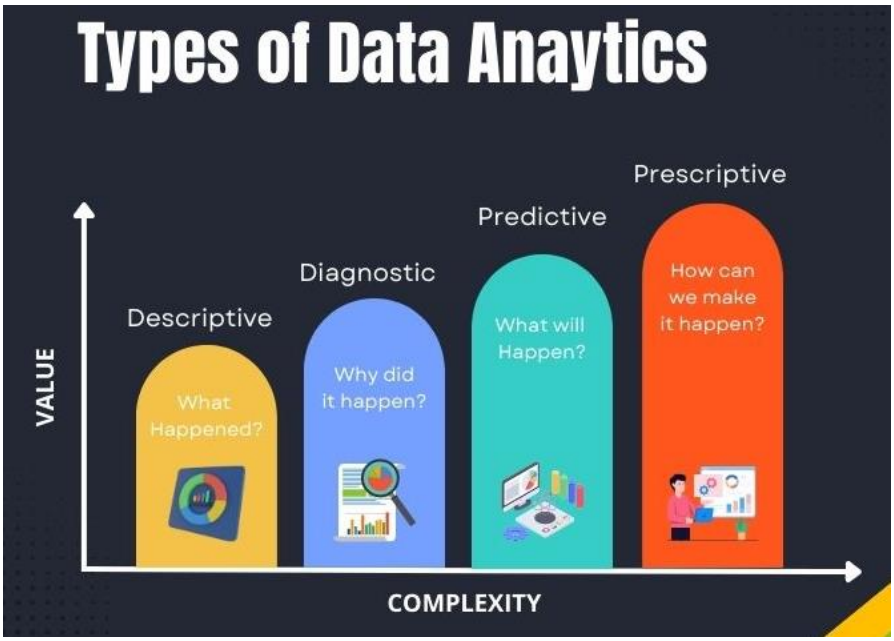
- ✓ Gather/collect the relevant data from difference sources.

6.2. Data cleaning

✓ Ensuring accuracy and consistency.

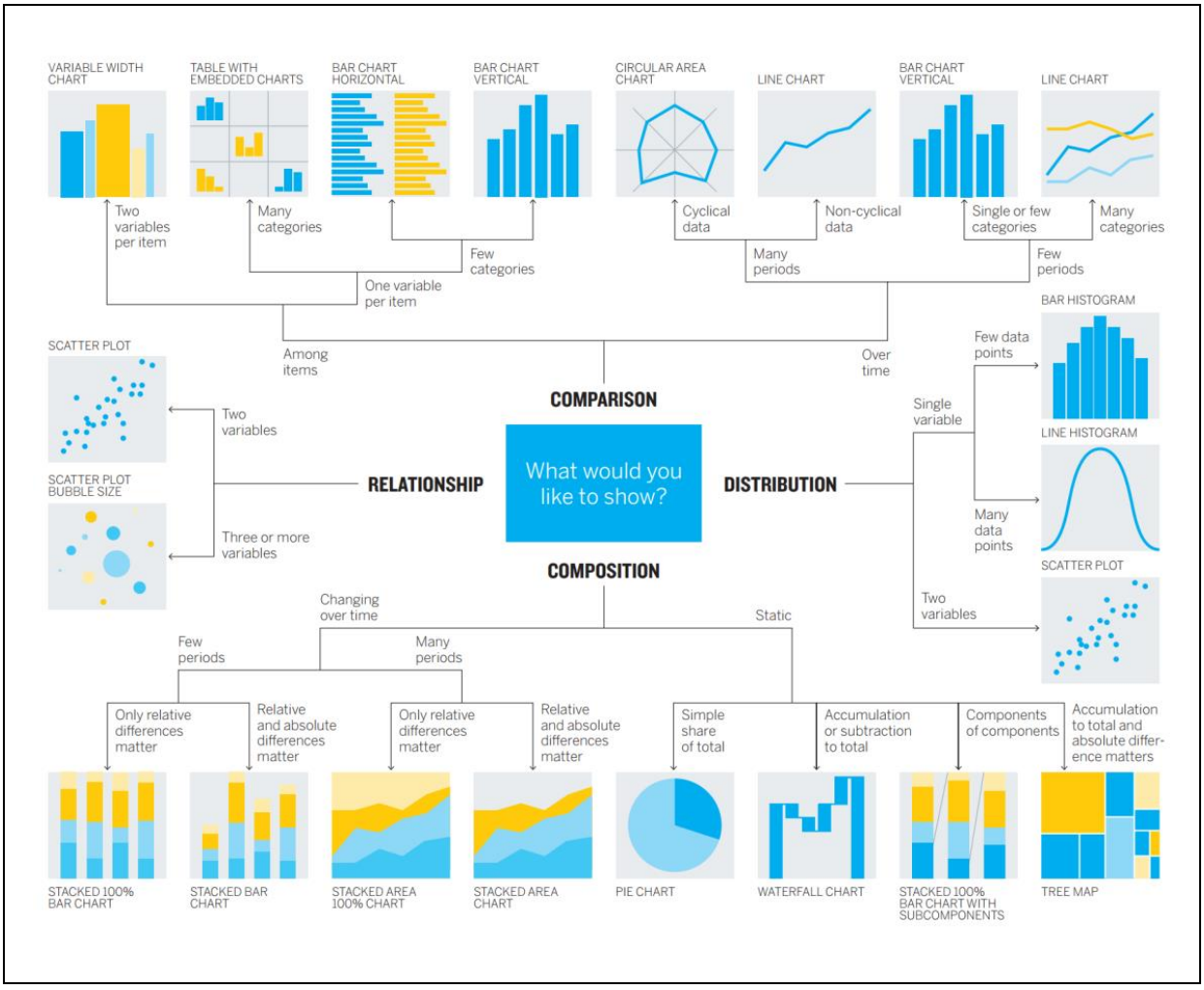
6.3. Data analysis

✓ Exploring data to identify trends and patterns.



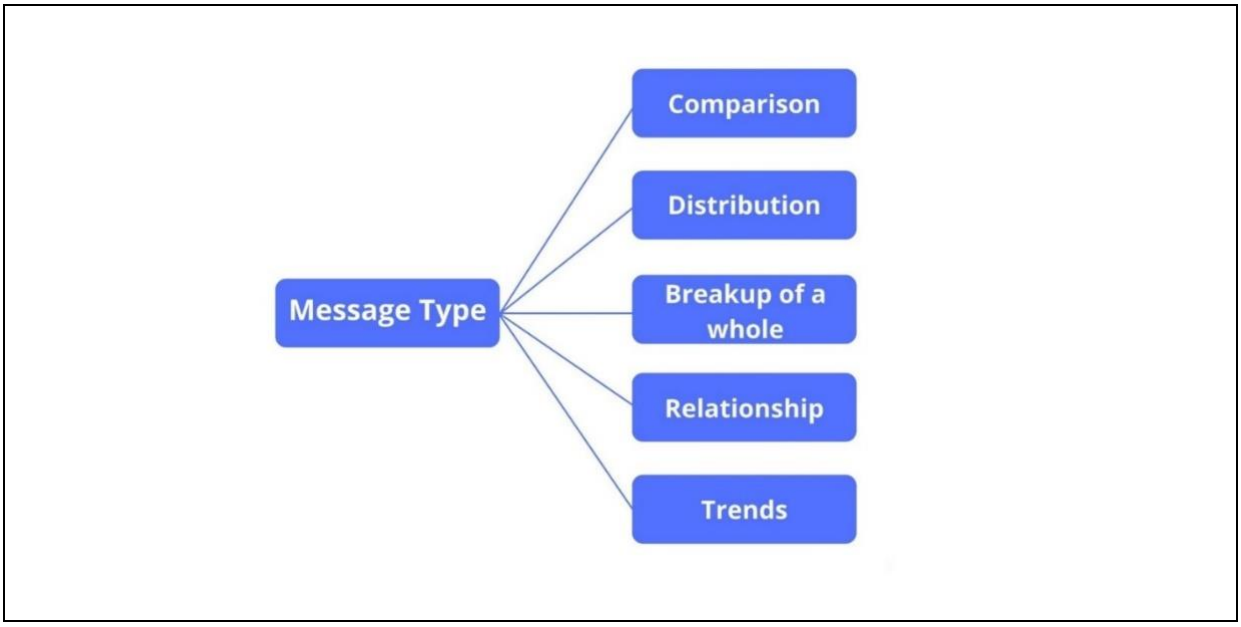
6.4. Chose the right visualization

✓ Selecting appropriate charts, graphs, or maps based on requirement.



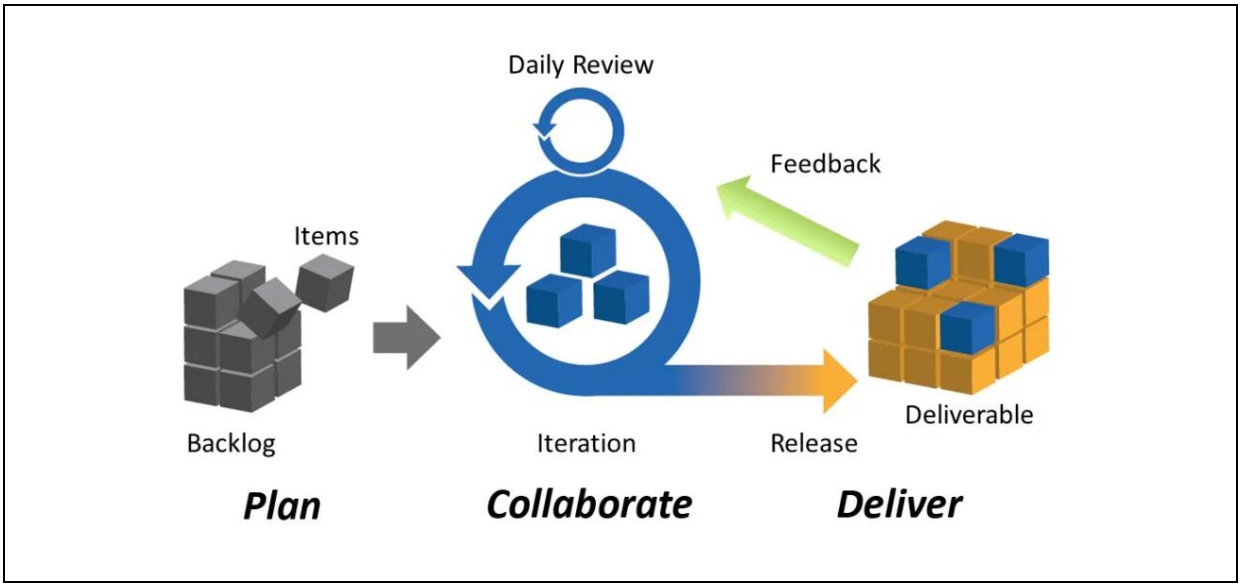
6.5. Creating visual representation

✓ Create data visualization to share information effectively



6.6. Review and Iterative

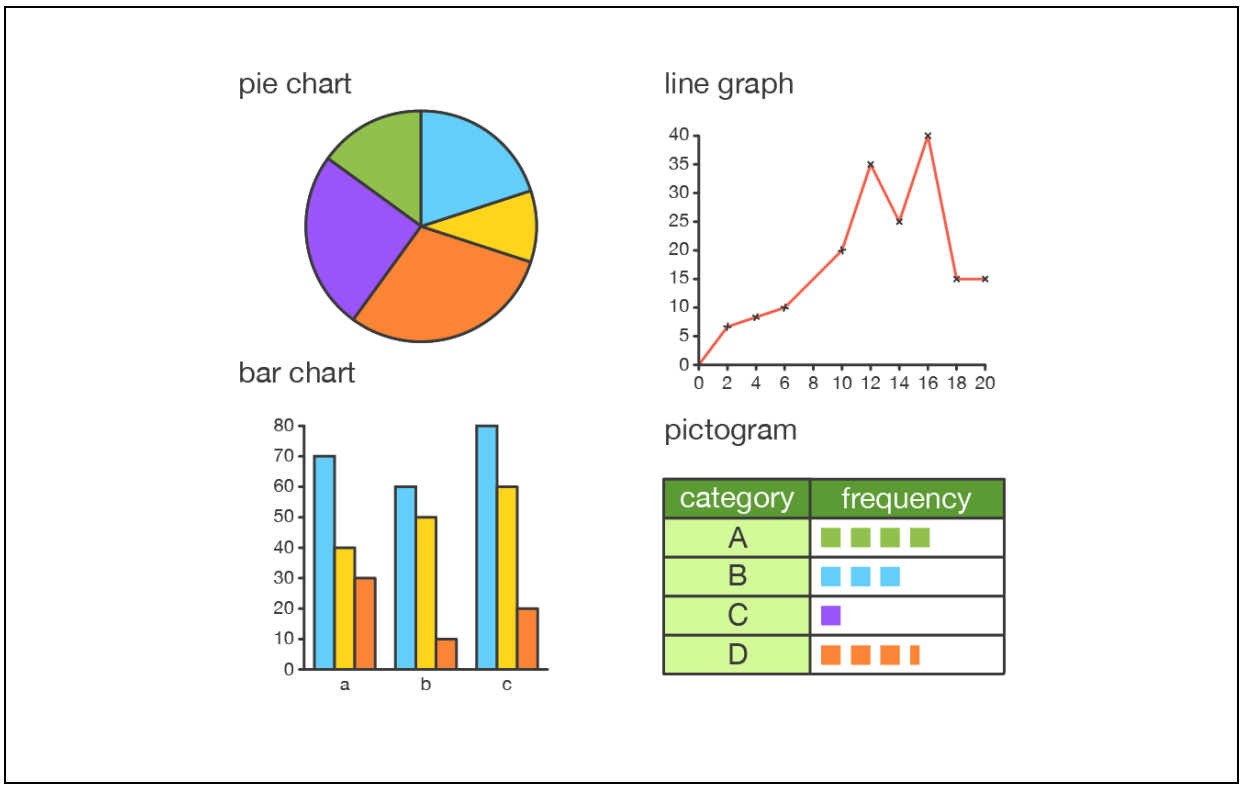
✓ Testing the visualization for clarity and effectiveness, making adjustments if needed.



7. Types of Data Visualization

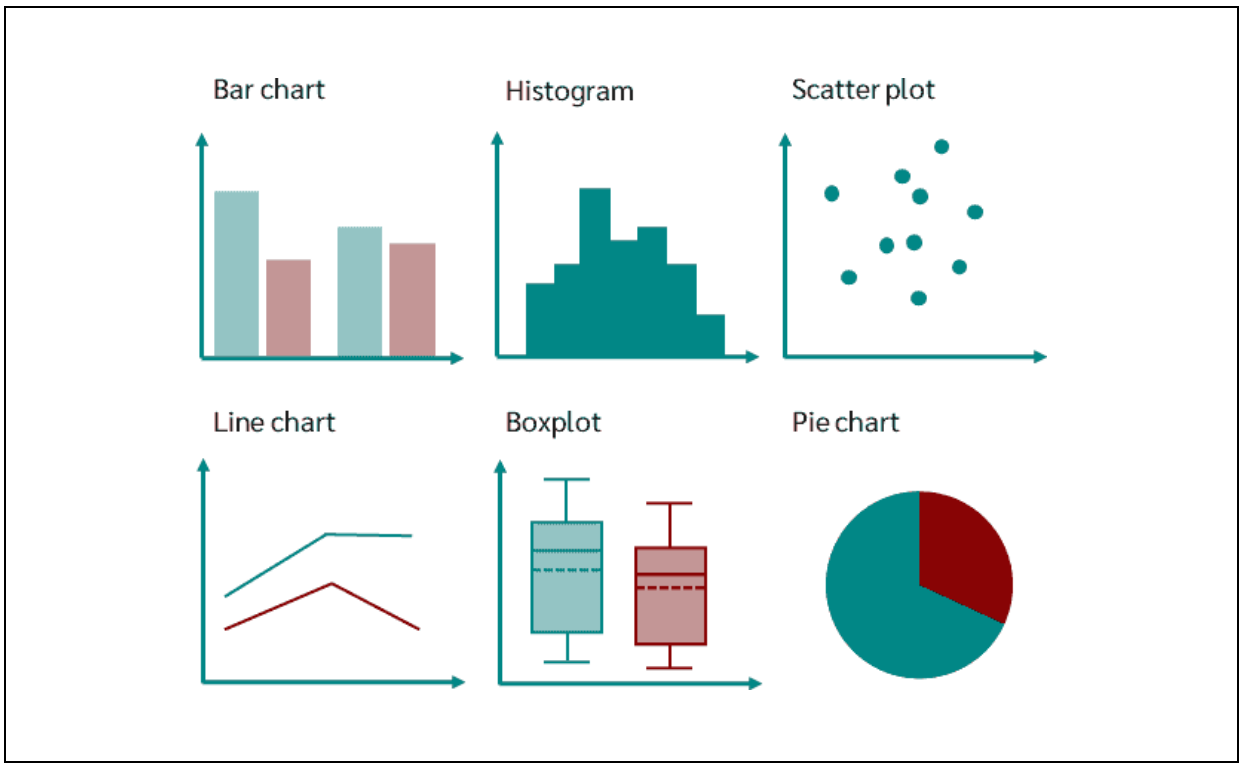
7.1. Basic Charts

- ✓ **Bar Chart**: Compares quantities across different categories.
  - **Column Chart**: Vertical version of the bar chart.
- ✓ **Line Chart**: Shows trends over time or continuous data.
- ✓ **Pie Chart**: Displays proportions of a whole; best for limited categories.



7.2. Statistical Visualizations

- ✓ **Histogram**: Represents the distribution of numerical data.
- ✓ **Box Plot**: Summarizes data distribution.
- ✓ **Scatter Plot**: Shows the relationship between two variables.



7.3. Advanced Charts

- ✓ **Heatmap**: Uses color to represent data values in a matrix format.
- ✓ **Bubble Chart**: A scatter plot with an added dimension represented by the size of the bubbles.

7.4. Interactive Visualizations

- ✓ **Dashboards**: Combines multiple visualizations to provide an overview of key metrics.
- ✓ **Data Explorer**: Allows users to interact with data, filtering and adjusting parameters.

7.5. Textual Visualizations

- ✓ **Word Cloud**: Displays text data, highlighting frequently used terms.
- ✓ **Tag Cloud**: Similar to a word cloud, often used for categorizing data.

8. Gestalt principles for Data Visualization

8.1. Proximity

- ✓ Grouping related items together.
  - Example: In a scatter plot, showing sales data for different products, grouping all electronics together.

8.2. Similarity

- ✓ Using similar shapes or colours to indicate relationships.
  - Example: Using same colors for regions in a bar chart (e.g., blue for East, green for West) helps viewers easily compare sales.

8.3. Closure

- ✓ Completing incomplete shapes to create a whole.
  - Example: A line graph, showing monthly temperature changes can use dotted lines for predictions, allowing viewers to intuitively connect the dots and grasp trends.

8.4. Continuity

- ✓ Following lines and patterns to guide the viewer's eye.
  - Example: A line graph, a stock price graph clearly shows trends, allowing viewers to easily spot increases and decreases over time.

9. Visualization reference model

- ✓ A visualization reference model works as a framework for understanding the components and process in Data Visualization.

Steps

- ✓ Data Collection
- ✓ Data Processing
- ✓ Visualization Design
- ✓ User Interaction
- ✓ Presentation
- ✓ Feedback
- ✓ Deployment

9.1. Data Layer

- ✓ **Data Sources**
  - Identify where the data is coming from (databases, APIs, spreadsheets).

- ✓ **Data Preparation**
  - Data Cleaning, transforming, and aggregating data to ensure it's ready for visualization.

9.2. Processing Layer

- ✓ **Data Analysis**:
  - Apply statistical methods or algorithms to extract insights from the data.

- ✓ **Data Reduction**:
  - Simplifying data by selecting key variables or filtering out noise to focus on important information.

9.3. Visualization Layer

- ✓ **Visual Encoding**:
  - Choosing how to represent data visually (e.g., using colors, shapes, sizes).

- ✓ **Chart Types**:
  - Selecting appropriate visualization types based on the data and insights (e.g., bar charts, line graphs, scatter plots).

- ✓ **Design Principles**:
  - Applying best practices for layout, color schemes, labelling, and accessibility.

9.4. Interaction Layer

- ✓ **Interactivity**:
  - Incorporating features like tooltips, filters, and zooming to allow users to explore data dynamically.

- ✓ **User Experience**:
  - Ensuring that the interaction is intuitive and enhances the understanding of the data.

9.5. Presentation Layer

- ✓ **Contextual Information**:
  - Providing background, legends, and annotations to help interpret the visualizations.

- ✓ **Storytelling**:
  - Structuring the visualizations to convey a narrative and guide the viewer through the insights.

9.6. Feedback Layer

- ✓ **User Testing**:
  - Gathering input from users to assess clarity, effectiveness, and engagement.

- ✓ **Iteration**:
  - Refining the visualizations based on feedback to improve understanding and impact.

9.7. Deployment Layer

- ✓ **Distribution**:
  - Sharing the visualizations through dashboards, reports, or web applications.

- ✓ **Maintenance**:
  - Regularly updating the visualizations to reflect new data and insights.

10. Data visualizations by the number of variables

- ✓ We can divide the data visualization based on the number of variables.

Types

- ✓ Univariate Visualizations
- ✓ Bivariate Visualizations
- ✓ Multivariate Visualizations

10.1. Univariate Visualizations

- ✓ These visualizations focus on a single variable, allowing you to explore its distribution and key statistics.
  - **Histograms**: Show the distribution of a continuous variable.
  - **Bar Charts**: Represent count of categories in a categorical variable.
  - **Box Plots**: Summarize the distribution, median, quartiles, and outliers of a single variable.
  - **Pie Charts**: Explains the proportions of categories in a categorical variable.
  - **Density Plots**: Display the distribution of a continuous variable in a smoothed format.

10.2. Bivariate Visualizations

- ✓ These visualizations explore the relationship between two variables, helping to identify correlations or patterns.
  - **Scatter Plots**: Show the relationship between two continuous variables.
  - **Line Graphs**: By using this we can display how one variable changes in relation to another variable
  - **Grouped Bar Charts**: Compare the values of a categorical variable across different groups.
  - **Heatmaps**: Represent the intensity of a variable across two dimensions (e.g., correlation matrices).
  - **Bubble Charts**: Extend scatter plots by adding a third variable represented by the size of the bubbles.

10.3. Multivariate Visualizations

- ✓ These involve three or more variables and can include
  - **3D Scatter Plots**: Visualize relationships among three continuous variables.
  - **Parallel Coordinates**: By using this we can understand how several variables relate to one another.
  - **Facet Grids**: Display multiple plots in a grid, each representing a subset of data based on one or more categorical variables.

11. Matplotlib

- ✓ Matplotlib is a powerful and widely-used plotting library for Python
- ✓ Using matplotlib we can plot the data.

Environment

- ✓ We can install this library by using pip command.

matplotlib installation

```
pip install matplotlib
```

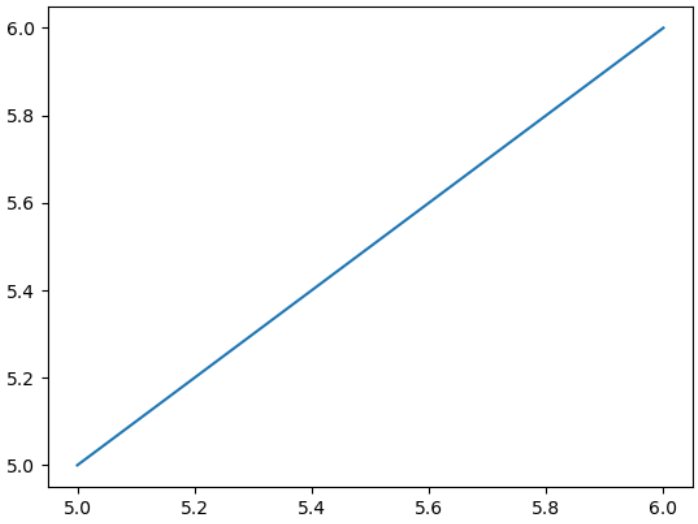
12. Line chart

- ✓ A line chart or line graph is a type of chart which displays information as a series of data points connected by straight line
- ✓ A line chart is often used to visualize a trend in data over intervals of time.

Program Name: Create a simple line chart
demo1.py

```
import matplotlib.pyplot as plt

x = [5, 6]
y = [5, 6]

plt.plot(x, y)

plt.show()
```

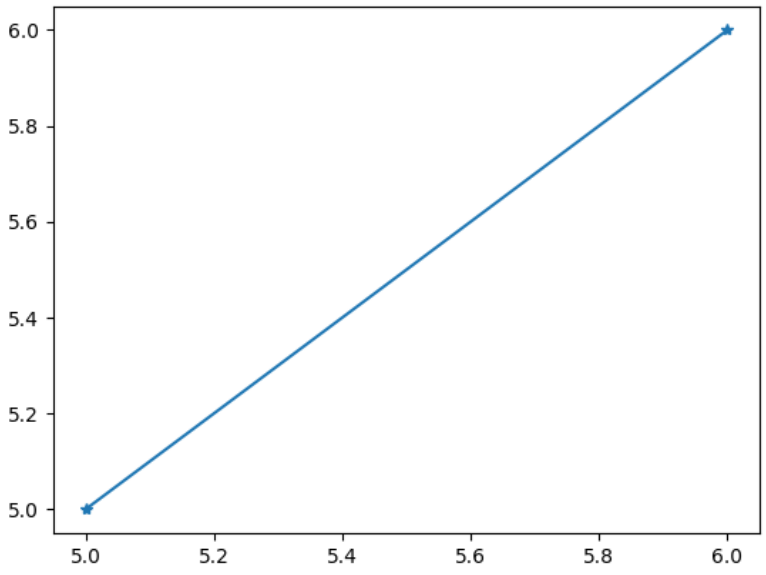
Output



Program Name: Create a simple line chart
demo2.py

```
import matplotlib.pyplot as plt

x = [5, 6]
y = [5, 6]

plt.plot(x, y, marker='*')

plt.show()
```

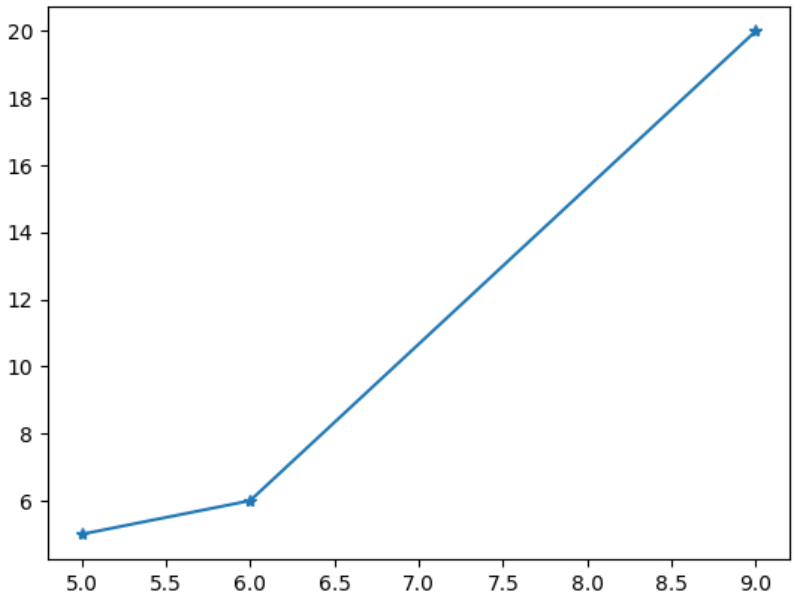
Output



Program Name: Create a simple line chart
demo3.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]

plt.plot(x, y, marker='*')

plt.show()
```

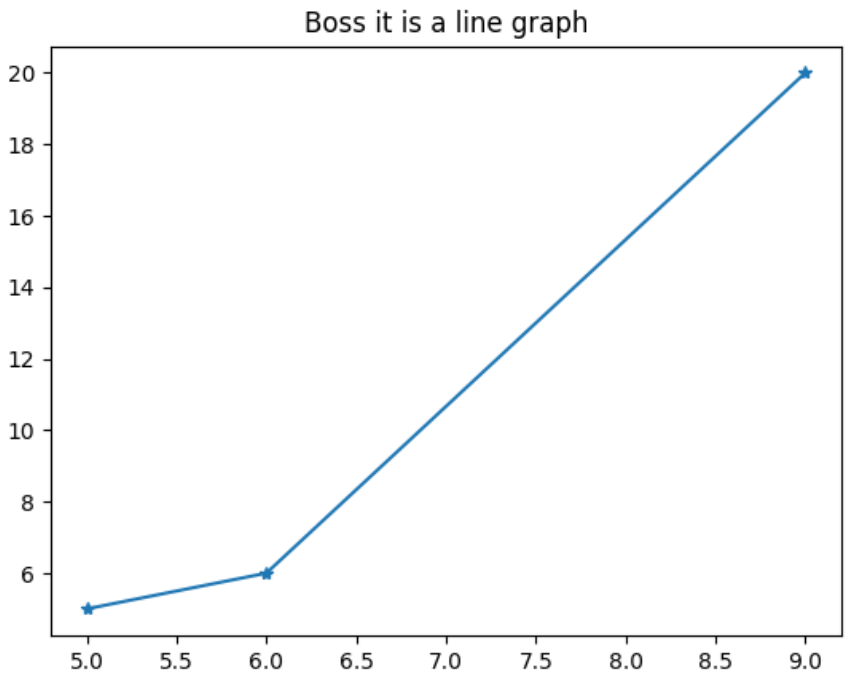
Output



Program Name: Create a simple line chart and title
demo4.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]

plt.title("Boss it is a line graph")

plt.plot(x, y, marker='*')

plt.show()
```

Output



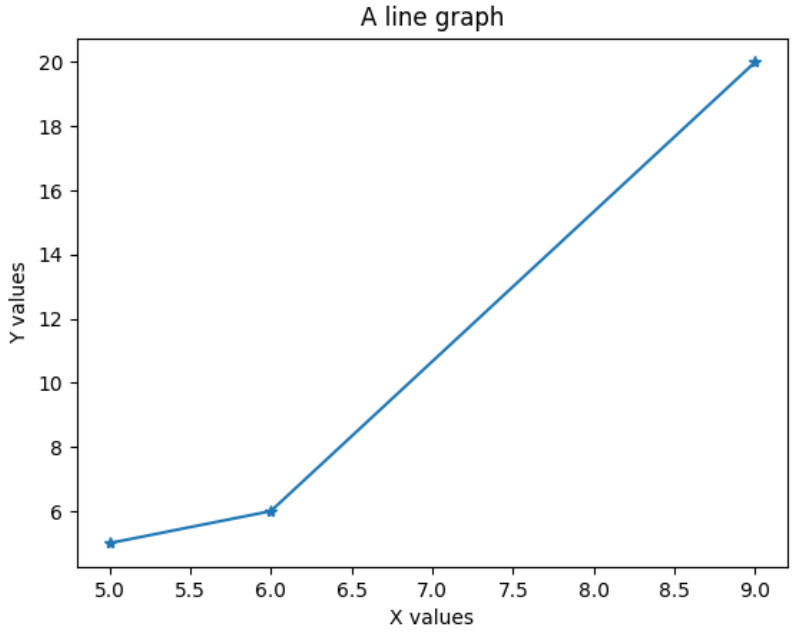
12.1. Labelling the axes

✓ We can label x axis and y axis by using xlabel and ylabel

Program Name: Create a simple line chart and giving title and labelling
demo5.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]

plt.title("A line graph")

plt.xlabel("X values")
plt.ylabel("Y values")

plt.plot(x, y, marker = '*')

plt.show()
```

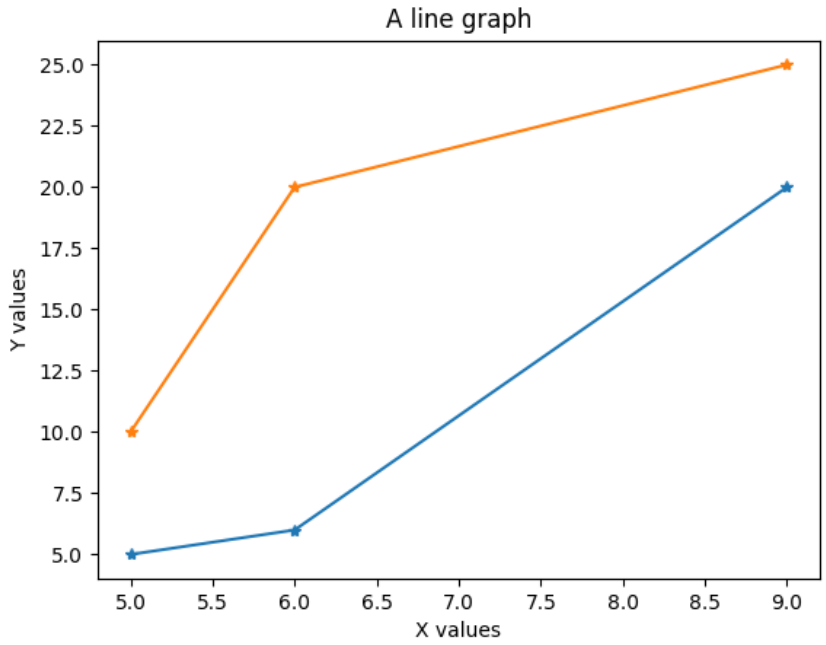
Output



Program Name: Create two lines in single chart
demo6.py

```
import matplotlib.pyplot as plt

x = [5, 6, 9]
y = [5, 6, 20]
p = [10, 20, 25]

plt.title("A line graph")

plt.xlabel("X values")
plt.ylabel("Y values")

plt.plot(x, y, marker = '*')
plt.plot(x, p, marker = '*')

plt.show()
```

Output



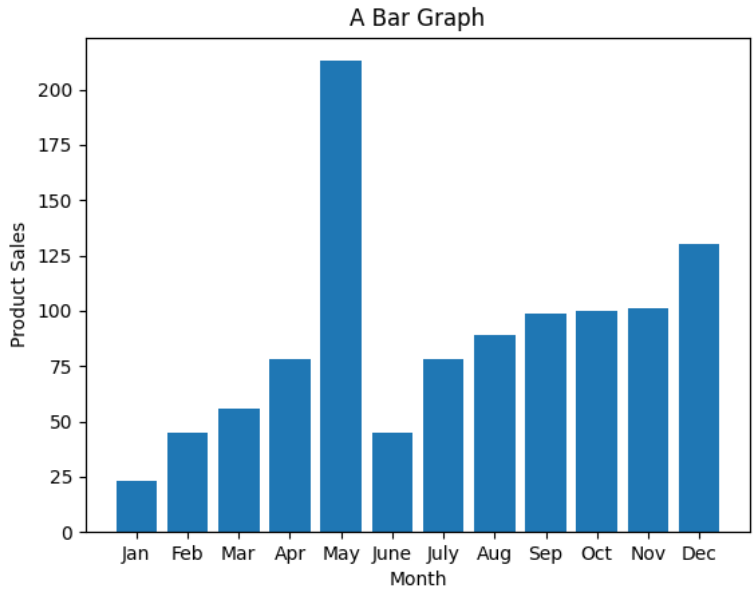
13. Bar Chart

✓ The bar graph is the graphical representation of categorical data.

Program Name: Creating bar chart
demo7.py

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"]
sales = [23, 45, 56, 78, 213, 45, 78, 89, 99, 100, 101, 130]

plt.title('A Bar Graph')

plt.xlabel('Month')
plt.ylabel('Product Sales')

plt.bar(months, sales)

plt.show()
```

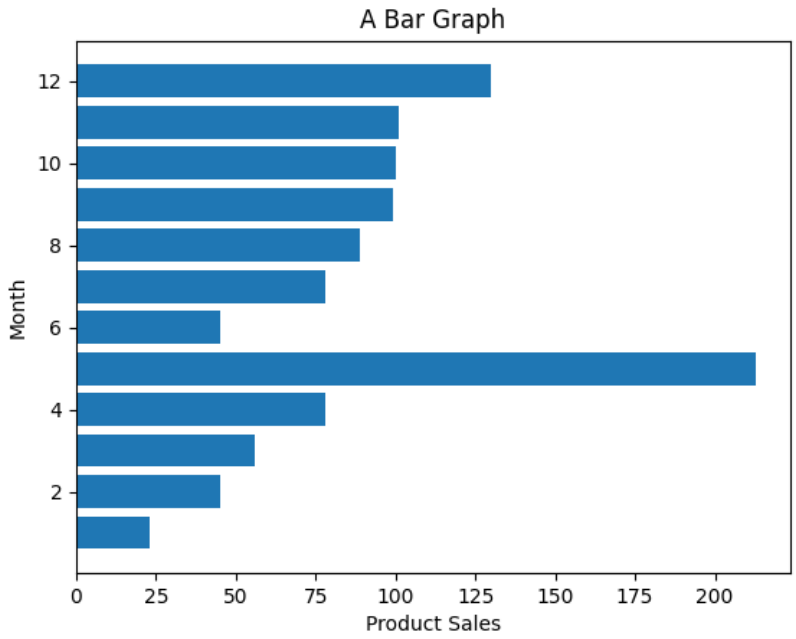
Output



Program Name: Creating horizontal bar chart
demo8.py

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"]
sales = [23, 45, 56, 78, 213, 45, 78, 89, 99, 100, 101, 130]

plt.title('A Bar Graph')

plt.xlabel('Product Sales')
plt.ylabel('Month')

plt.barh(months, sales)

plt.show()
```

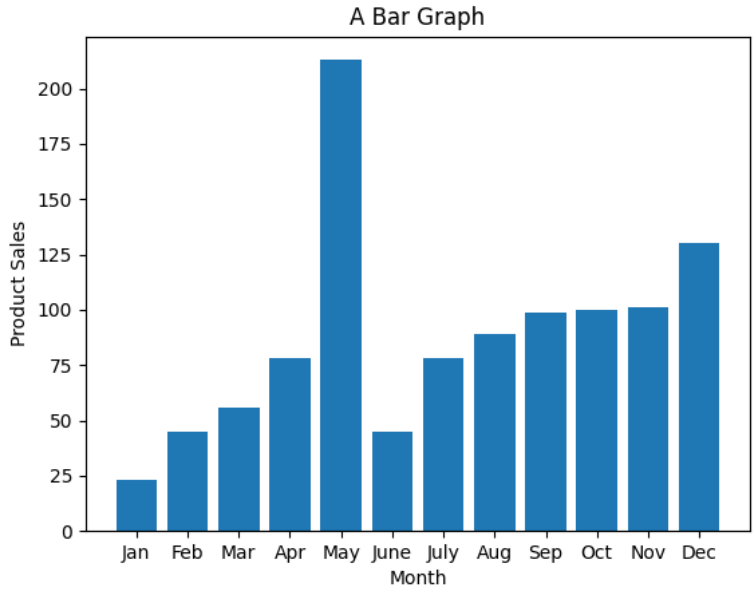
Output



Program Name: Creating horizontal bar chart
File name: demo9.py
File name: sales11.csv

```
import matplotlib.pyplot as plt
import pandas as pd

df = pd.read_csv("sales11.csv")

plt.title('A Bar Graph')

plt.xlabel('Month')
plt.ylabel('Product Sales')

plt.bar(df.month, df.sales)

plt.show()
```

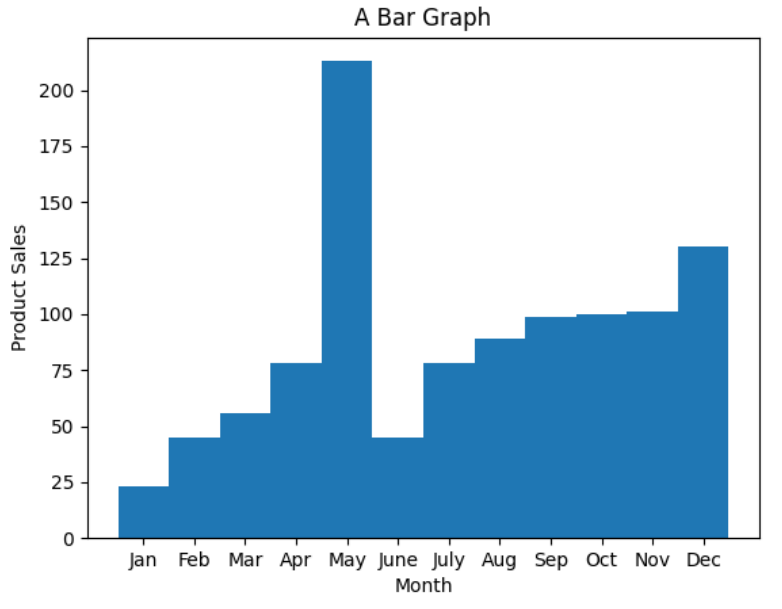
Output



Program Name: Creating bar chart
demo10.py

```
import matplotlib.pyplot as plt

months = ["Jan", "Feb", "Mar", "Apr", "May", "June", "July", "Aug",
"Sep", "Oct", "Nov", "Dec"]
sales = [23, 45, 56, 78, 213, 45, 78, 89, 99, 100, 101, 130]

plt.title('A Bar Graph')

plt.xlabel('Month')
plt.ylabel('Product Sales')

plt.bar(months, sales, width = 1.0)

plt.show()
```

Output



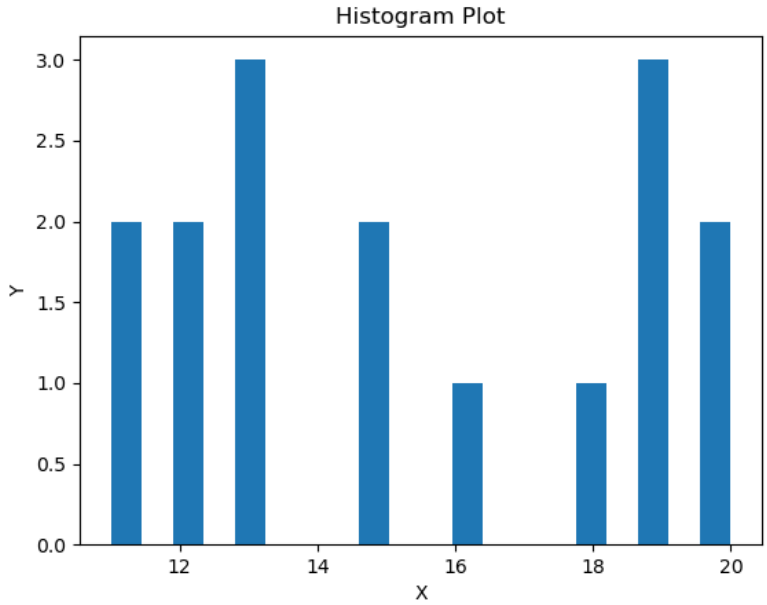
14. Histogram

- ✓ A histogram is the graphical representation of quantitative data.
- ✓ This displays the frequency/count of numerical data in bars.

Program Name: Creating histogram
demo11.py

```
import matplotlib.pyplot as plt

data = [12, 15, 13, 20, 19, 20, 11, 19, 11, 12, 19, 13, 15, 16, 18, 13]

plt.xlabel("X")
plt.ylabel("Y")
plt.title("Histogram Plot")

plt.hist(data, bins = 20)
plt.show()
```

Output



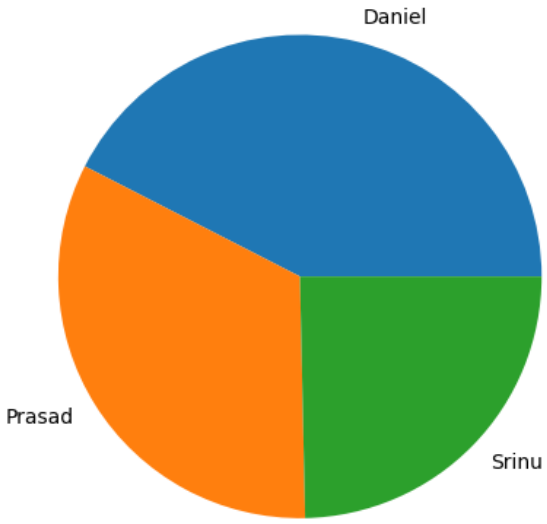
15. Pie Chart

- ✓ This is a circular plot that has been divided into slices displaying numerical proportions.
- ✓ Every slice in the pie chart shows the proportion of the element to the whole.
- ✓ A large category means that it will occupy a larger portion of the pie chart.

Program Name: Creating pie chart
demo12.py

```
import matplotlib.pyplot as plt

students = ['Daniel', 'Prasad', 'Srinu']
points = [62, 48, 36]

plt.pie(points, labels = students)

plt.axis('equal')
plt.show()
```

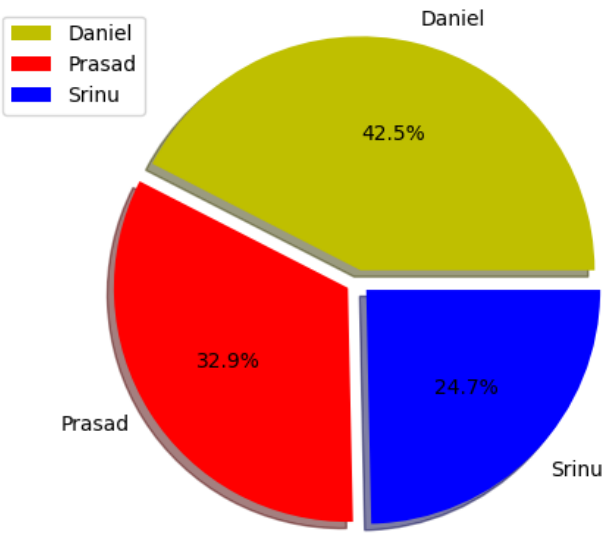
Output



Program Name: Creating pie chart
demo13.py

```
import matplotlib.pyplot as plt

students = ['Daniel', 'Prasad', 'Srinu']
points = [62, 48, 36]

c = ['y', 'r', 'b']

plt.pie(points, labels = students, colors = c , shadow = True,
explode = (0.05, 0.05, 0.05), autopct = '%1.1f%%')

plt.axis('equal')
plt.legend()
plt.show()
```

Output



15.1. Attributes

- ✓ The first parameter to the function is the list of numbers for every category.
  - labels attribute:
    - A list of categories separated by commas is then passed as the argument to labels attribute.
  - colors attribute:
    - To provide the color for every category.
  - To create shadows around the various categories in pie chart.
  - To split each slice of the pie chart into its own.

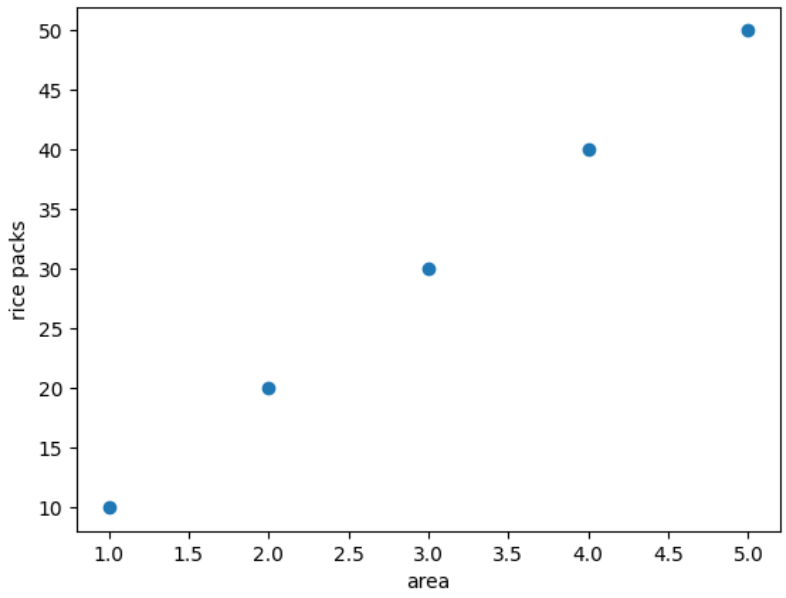
16. Scatter Plot

- ✓ In scatter plot each value in the data set is represented by a dot.
- ✓ By using this plot we can understand the relationship between two variables.

Program Name: Creating Scatter plot
demo14.py

```
import matplotlib.pyplot as plt

area = [1, 2, 3, 4, 5]
rice_packs = [10, 20, 30, 40, 50]

plt.xlabel('area')
plt.ylabel('rice packs')

plt.scatter(area, rice_packs)
plt.show()
```

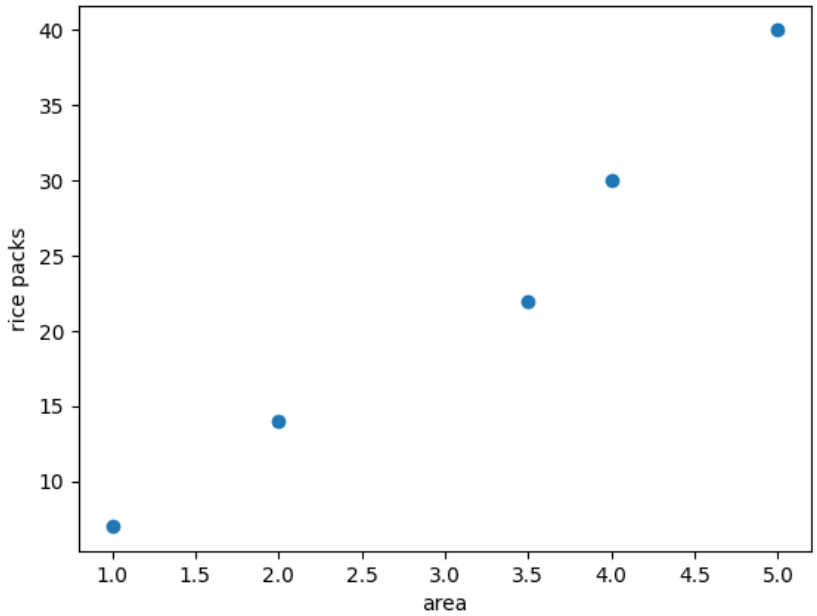
Output



Program Name: Creating Scatter plot
demo15.py

```
import matplotlib.pyplot as plt

area = [1, 2, 3.5, 4, 5]
rice_packs = [7, 14, 22, 30, 40]

plt.xlabel('area')
plt.ylabel('rice packs')

plt.scatter(area, rice_packs)
plt.show()
```

Output



17. Box Plots

- ✓ Box plots help us measure how well data in a dataset is distributed.
- ✓ The graph shows the maximum, minimum, median, first quartile and third quartiles of the dataset.

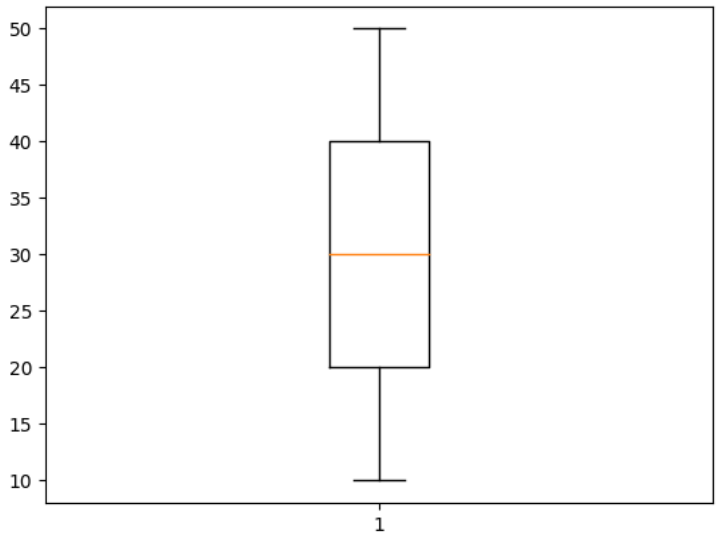
17.1. Use Box plots

- ✓ Use a boxplot when you need to get the overall statistical information about the data distribution.
- ✓ It is a good tool for detecting outliers in a dataset.

Program Name: Creating box plot
demo16.py

```
import matplotlib.pyplot as plt

data = [10, 20, 30, 40, 50]

plt.boxplot(data)
plt.show()
```

Output



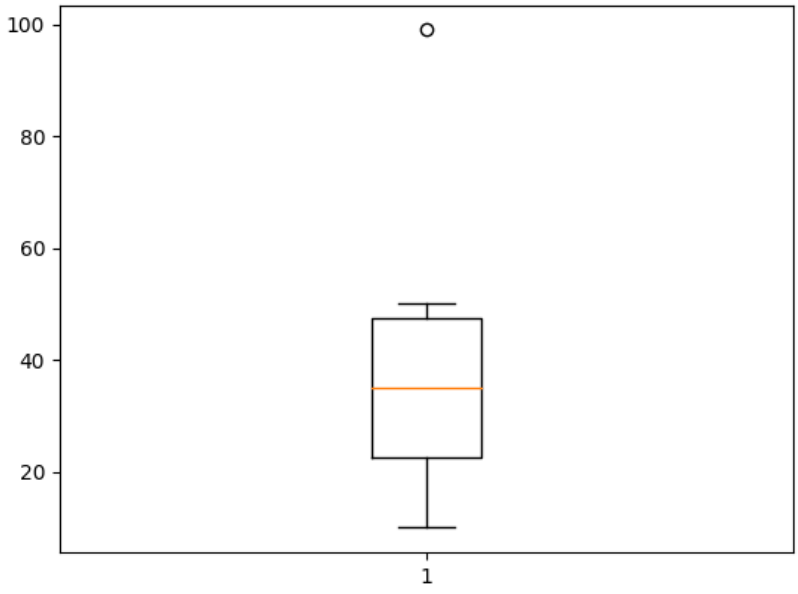
17.2. Box plot explanation

- ✓ The line dividing the box into two shows the median of the data.
- ✓ The end of the box represents the upper quartile (75%) while the start of the box represents the lower quartile (25%).
- ✓ The part between the upper quartile and the lower quartile is known as the Inter Quartile Range (IQR) and helps in approximating 50% of the middle data.

Program Name: Creating box plot with outlier
demo17.py

```
import matplotlib.pyplot as plt

data = [10, 20, 30, 40, 50, 90]

plt.boxplot(data)
plt.show()
```

Output



18. Heatmap

- ✓ A heatmap is a method of data visualization that plots data by replacing numbers with colours.
- ✓ If it is representing with color then it is very easy to understand patterns between different values in the dataset.
- ✓ It is used to visualize data in a two-dimensional format as a coloured map so that different colour variations represent different patterns between features.

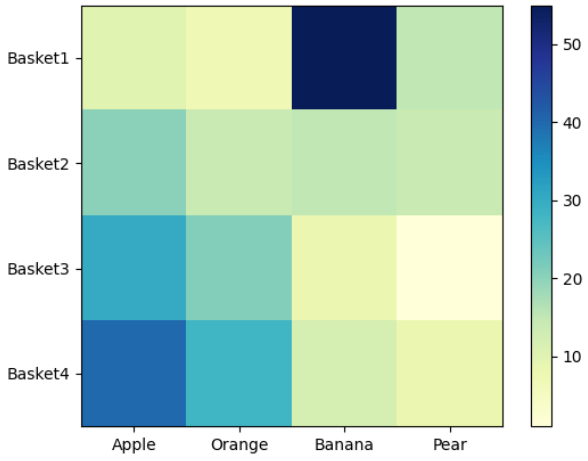
18.1. How to understand?

- ✓ A heatmap visualizes the relationship between features as a colour palette.
- ✓ While analysing a heatmap, always remember that **dark shades** represent a **high degree** of linear relationship between features and **light shades** represent a **low degree** of linear relationship between features.

Program Name: Creating box plot
demo18.py

```
import matplotlib.pyplot as plt
import pandas as pd

d = {
    "Apple": [10, 20, 30, 40],
    "Orange": [7, 14, 21, 28],
    "Banana": [55, 15, 8, 12],
    "Pear": [15, 14, 1, 8]
}

i = ['Basket1', 'Basket2', 'Basket3', 'Basket4']

df = pd.DataFrame(d, index = i)

plt.imshow(df, cmap = "YlGnBu")
plt.colorbar()

plt.xticks(range(len(df)), df.columns)
plt.yticks(range(len(df)), df.index)

plt.show()
```

Output



2. DATA VISUALIZATION – PART - 2

Contents

2. DATA VISUALIZATION – PART - 2

1. Titanic Introduction

- ✓ The Titanic was known as the unsinkable ship and was the largest, most luxurious passenger ship.
- ✓ Sadly, the British ocean liner sank on April 15, 1912, killing over many people while just few people got survived.
- ✓ Let's do analyse titanic dataset

2. Seaborn library

- ✓ Seaborn is advanced data visualization library.
- ✓ By using this we can visualize the data.

2.1. Environment

- ✓ We can install this library by using pip command.

Seaborn installation

```
pip install seaborn
```

Program Name: Loading titanic dataset
demo1.py

```
import seaborn as sns

df = sns.load_dataset('titanic')
print(df.head())
```

Output

```
   survived  pclass     sex   age  sibsp  parch  ...    who adult_male deck  embark_town  alive  alone
0         0       3    male  22.0      1      0  ...    man       True  NaN  Southampton     no  False
1         1       1  female  38.0      1      0  ...  woman      False    C    Cherbourg    yes  False
2         1       3  female  26.0      0      0  ...  woman      False  NaN  Southampton    yes   True
3         1       1  female  35.0      1      0  ...  woman      False    C  Southampton    yes  False
4         0       3    male  35.0      0      0  ...    man       True  NaN  Southampton     no   True

[5 rows x 15 columns]
```

3. Titanic data set understanding

- ✓ Let's understand the titanic dataset.
- ✓ Data Set Column Descriptions
  - pclass: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
  - survived: Survival (0 = No; 1 = Yes)
  - name: Name
  - sex: type of gender
  - age: Age
  - sibsp: Number of siblings/spouses aboard
  - parch: Number of parents/children aboard
  - fare: Passenger fare (British pound)
  - embarked: Port of embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)
  - adult_male: A male 18 or older (0 = No, 1=Yes)
  - deck: Deck of the ship
  - who: man (18+), woman (18+), child (<18)
  - alive: Yes, no
  - embarked_town: Port of embarkation (Cherbourg, Queenstown, Southampton)
  - class: Passenger class (1st; 2nd; 3rd)
  - alone: 1 = alone, 0 = not alone (you have at least 1 sibling, spouse, parent or child on board)

Program Name: Number of rows and columns
demo2.py

```
import seaborn as sns

df = sns.load_dataset('titanic')
print(df.shape)
```

Output

```
(891, 15)
```

Program Name: Display the columns
demo3.py

```
import seaborn as sns

df = sns.load_dataset('titanic')
print(df.columns)
```

Output

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
'alive', 'alone'], dtype='object')
```

Program Name: DataFrame information
demo4.py

```
import seaborn as sns

df = sns.load_dataset('titanic')
df.info()
```

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   survived     891 non-null    int64
 1   pclass       891 non-null    int64
 2   sex          891 non-null    object
 3   age          714 non-null    float64
 4   sibsp        891 non-null    int64
 5   parch        891 non-null    int64
 6   fare         891 non-null    float64
 7   embarked     889 non-null    object
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

Program Name: unique values for sex(gender) column
demo5.py

```
import seaborn as sns
import pandas as pd

df = sns.load_dataset('titanic')
result = df['sex'].unique()
print(result)
```

Output

```
['male' 'female']
```

4. Data Analysis

- ✓ From the data **max price/fare** a passenger paid for a ticket in this data set was 512.3292 British pounds, and the **minimum price/fare** was 0 British pounds.
- ✓ There is missing data for age column.
- ✓ The **mean** age is 29.699 and the oldest passenger in this data set was 80 years old, while the youngest was only .42 years old (about 5 months).

Program Name: describe() method
demo6.py

```
import seaborn as sns

df = sns.load_dataset('titanic')
print(df.describe())
```

Output

```
          survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

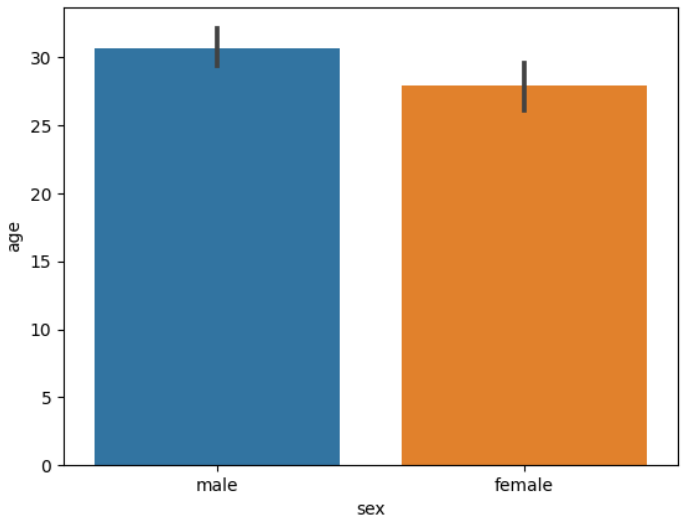
5. Bar Plot

✓ A bar plot shows the mean value of every value in a categorical column.

Program Name: Creating bar plot
demo7.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')
sns.barplot(x = 'sex', y = 'age', data = df)

plt.show()
```

Output



✓ The plot clearly shows that the average age for all male passengers is above 30 while the average age of the female passengers is between 25 and 30.

6. Get a count of the number of survivors

- ✓ **0** represents not survived
- ✓ **1** means survived.

Program Name: Get a count of the number of survivors
demo8.py

```
import seaborn as sns

df = sns.load_dataset('titanic')
print(df['survived'].value_counts())
```

Output

```
0    549
1    342
Name: survived, dtype: int64
```

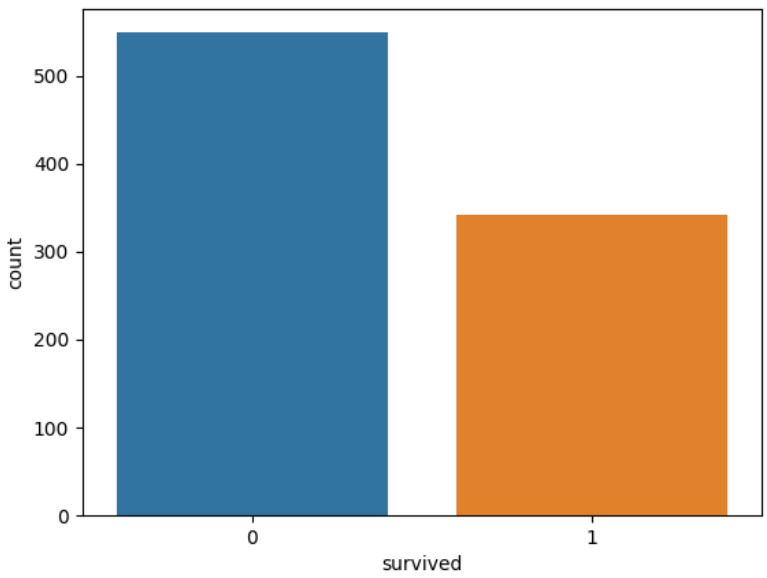
7. Count Plot

- ✓ This type of plot is similar to the bar plot, it displays the count of categories in a specific column.
- ✓ By using we can calculate the total number or count of survived and not survived.

Program Name: Get a count of the number of survivors
demo9.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')
sns.countplot(x = "survived", data = df)

plt.show()
```

Output



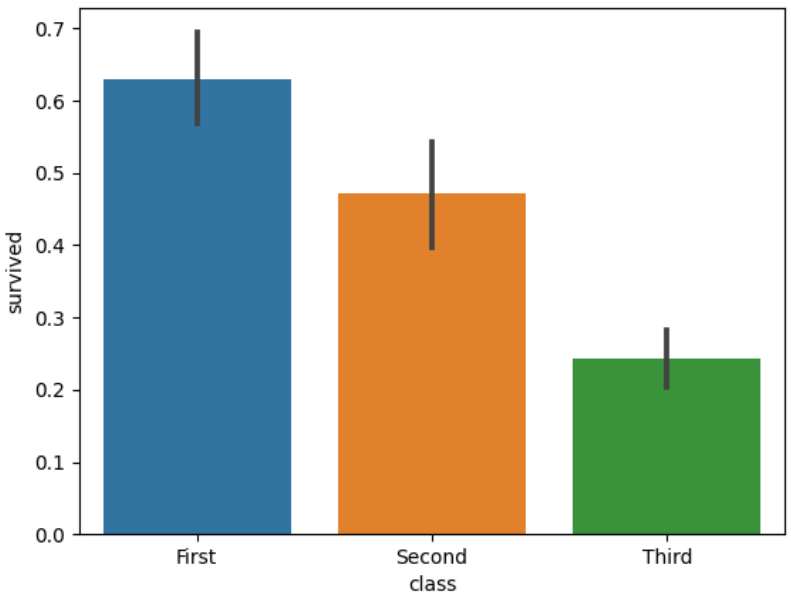
9. Plot the survival rate of each class

- ✓ A little over 60% of the passengers in first class survived. Less than 30% of passengers in third class survived.
- ✓ That means less than half of the passengers in third class survived, compared to the passengers in first class.

Program Name: Plot the survival rate of each class
demo10.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')
sns.barplot(x = 'class', y = 'survived', data = df)

plt.show()
```

Output



10. Let's understand the survival rate by gender and class.

- ✓ From the pivot table below, we see that females in first class had a survival rate of about 96.8%, meaning the majority of them survived.
- ✓ Males in third class had the lowest survival rate at about 13.54%, meaning the majority of them did not survive.

Program Name: Plot the survival rate of each class
demo11.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')

result = df.pivot_table('survived', index = 'sex', columns = 'class')

print(result)
```

Output

```
class       First    Second     Third
sex
female  0.968085  0.921053  0.500000
male    0.368852  0.157407  0.135447
```

11. Let's understand the survival rate by gender, age and class.

Program Name: Plot the survival rate by gender, age and class
demo12.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')

diff_ages = pd.cut(df['age'], [0, 18, 80])

result = df.pivot_table('survived', ['sex', diff_ages], 'class')

print(result)
```

Output

```
class                First    Second     Third
sex    age
female (0, 18]    0.909091  1.000000  0.511628
       (18, 80]   0.972973  0.900000  0.423729
male   (0, 18]    0.800000  0.600000  0.215686
       (18, 80]   0.375000  0.071429  0.133663
```

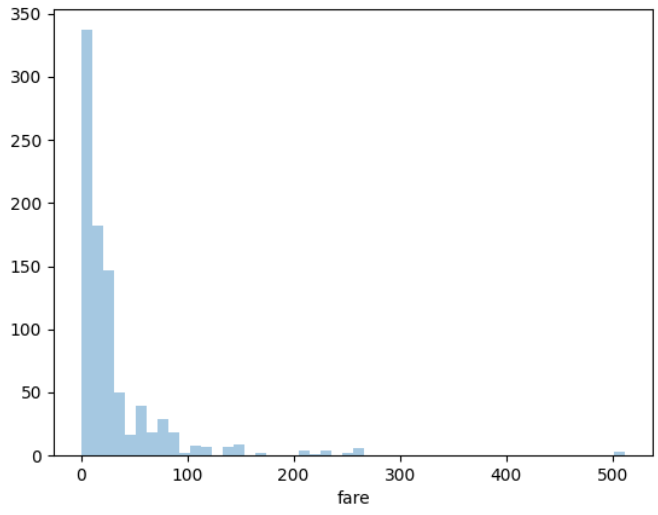
12. Dist Plot

- ✓ To create distribution plot we need to call distplot(p) function.
- ✓ This will create histogram distribution of a dataset for a column.
- ✓ We can plot the price of the ticket for every passenger

Program Name: Finding Most of the tickets, using dist plot
demo13.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')
sns.distplot(df['fare'], kde = False)

plt.show()
```

Output



- ✓ The above plot shows that most of the tickets have been sold between 0 and 50 dollars.

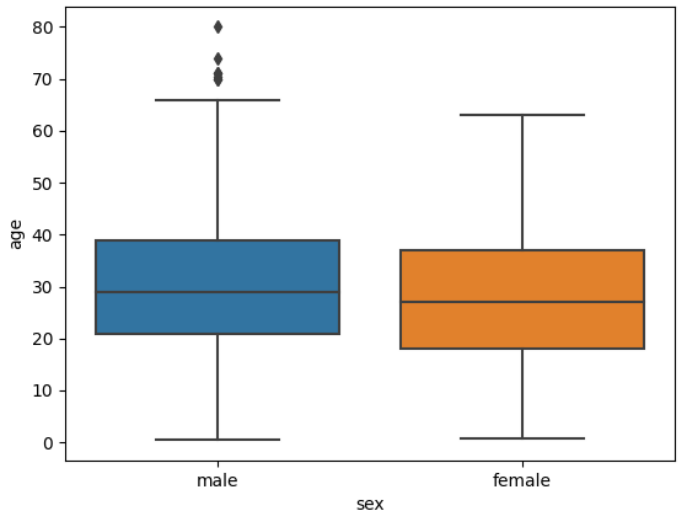
13. Box Plot

- ✓ The box plot is used to display the distribution of the categorical data in the form of quartiles like Q1, Q2, Q3 and Q3.
- ✓ The center of the box shows the median value.
- ✓ Now let's plot a box plot that displays the distribution for the age with respect to each gender.

Program Name: Creating a boxplot with sex column(gender) survived
demo14.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')
sns.boxplot(x = 'sex', y = 'age', data = df)

plt.show()
```

Output



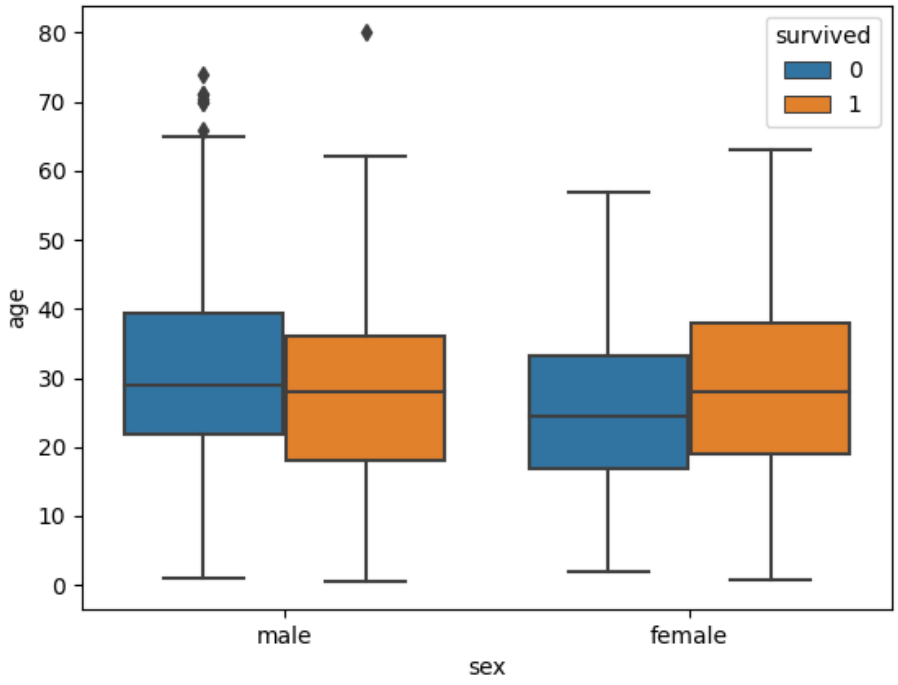
- ✓ The first quartile-Q1 starts at around 3 and ends at 22 which mean that 25% of the passengers are aged between 5 and 22.
- ✓ The second quartile-Q2 starts at around 23 and ends at around 28 which mean that 25% of the passengers are aged between 23 and 28.
- ✓ Similarly, the third quartile-Q3 starts and ends between 29 and 38, hence 25% passengers are aged within this range and finally the fourth or last quartile—Q4 starts at 39 and ends around 76.
- ✓ The part between the upper quartile and the lower quartile is known as the **Inter Quartile Range (IQR)** and helps in approximating 50% of the middle data.

Program Name: Creating a boxplot with survived

demo15.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')
sns.boxplot(x = 'sex', y = 'age', data = df, hue = "survived")

plt.show()
```

Output



- ✓ Other than the information about the age of the passengers, the above plot also shows the distribution of passengers who survived.
- ✓ The plot shows that most young males survived compared to females.

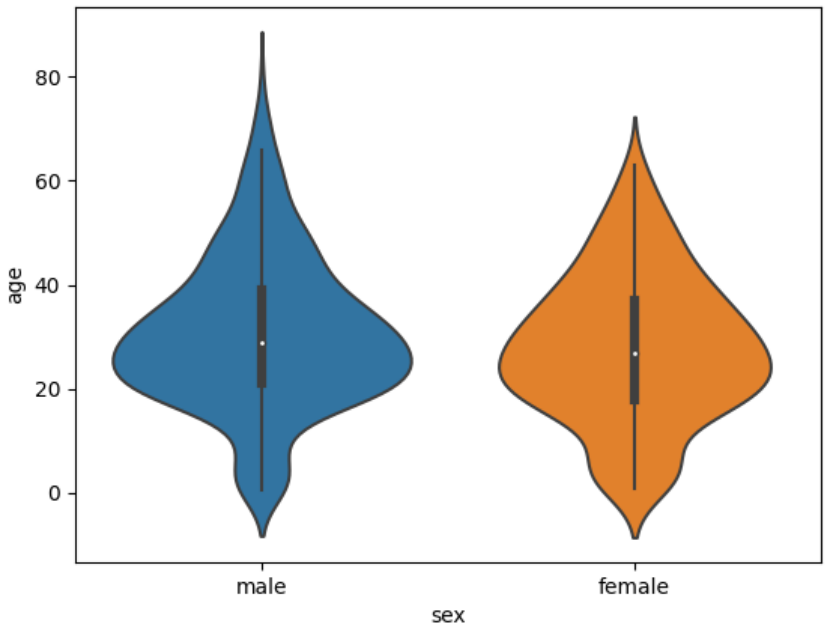
14. Violin Plot

✓ This type of plot is the same as the box plot, but with a violin plot, we can display all components corresponding to a data point.

Program Name: Creating violin plot
demo16.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')
sns.violinplot(x = 'sex', y = 'age', data = df)

plt.show()
```

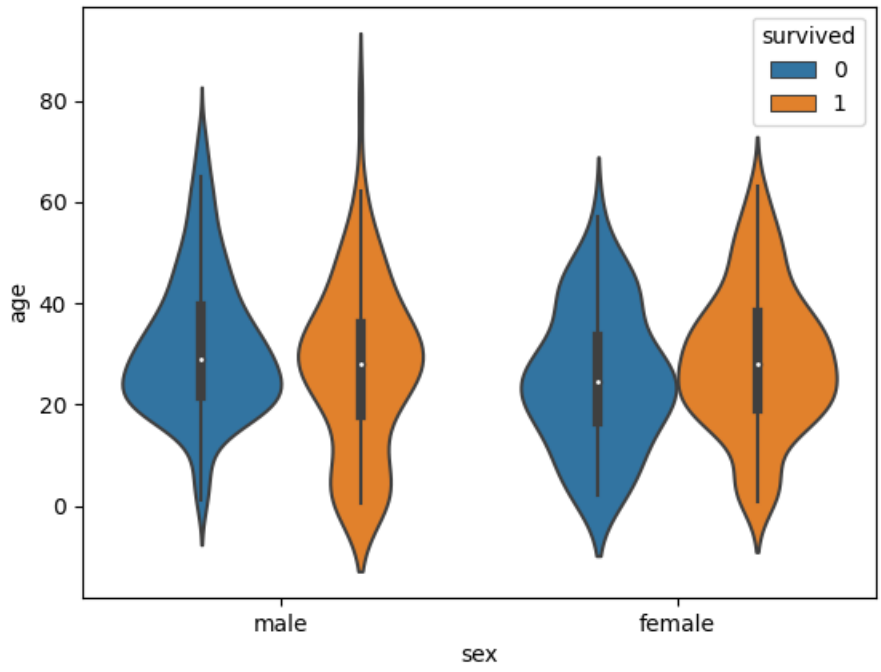
Output



Program Name: Creating violin plot with survived
demo17.py

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

import warnings
warnings.filterwarnings('ignore')

df = sns.load_dataset('titanic')
sns.violinplot(x = 'sex', y = 'age', data = df, hue = 'survived')

plt.show()
```

Output



15. Word Cloud

- ✓ A word cloud is a data visualization technique.
- ✓ This technique displays most used words in large font and the least used words in small font.
- ✓ It helps to get an idea about your text data,

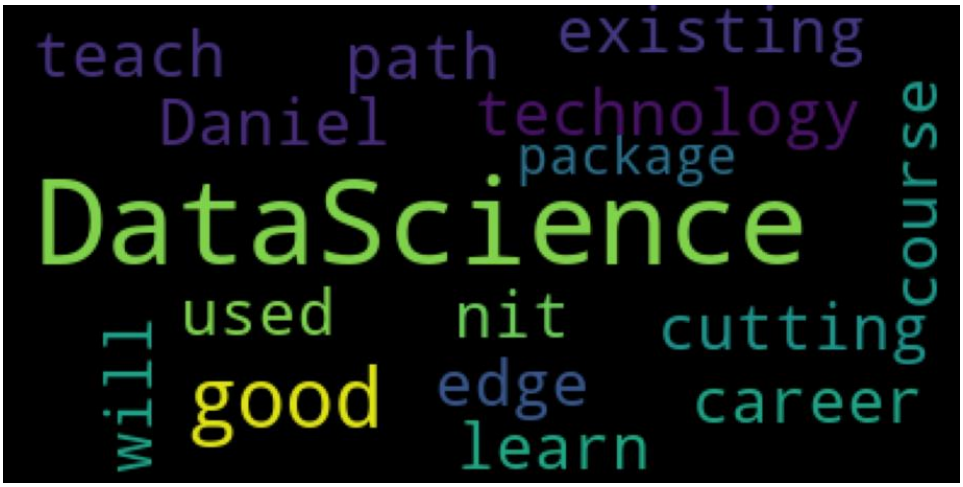
We need to install

```
pip install wordcloud
```

Program Name: Wordcloud example
demo18.py

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

text = "DataScience having good career path, DataScience is
cutting edge technology, DataScience course is existing in nit,
Daniel used to teach DataScience, if we learn DataScience then we
will get good package"

wc = WordCloud()
wc.generate(text)

plt.figure(figsize = (12, 12))
plt.imshow(wc)

plt.axis('off')
plt.show()
```

Output



16. Sunburst plot

- ✓ A sunburst plot is a very popular data visualization technique used to visualize hierarchical data.
- ✓ In every level of the hierarchy is represented by a ring or circle.
- ✓ Whereas the innermost circle or ring is the highest level of the hierarchy.

We need to install

```
pip install plotly
```

Program Name: Sunburst plot example
demo19.py

```
import plotly.express as px

data = px.data.tips()
figure = px.sunburst(data, path = ["day", "sex"], values =
"total_bill")

figure.show()
```

Output