Department of Computer Engineering

**Academic Year:** 2022-2023 **Semester:** VIII

**Subject:** Applied Data Science **Class / Division:** BE/CMPN

**Name:** **Roll Number:**

**Experiment No.: 4**

**Explore data visualization techniques.**

**Aim :** Explore data visualization techniques.**.**

**I-OBJECTIVE**

* To understand basic concepts of data visualization and techniques
* To Explore data visualization techniques

**II-THEORY**

## What are data visualizations?

Simply put, [data visualizations](https://www.sisense.com/glossary/data-visualization/) allow humans to explore data in many different ways and see patterns and insights that would not be possible when looking at the raw form. Humans crave narrative and visualizations allow us to pull a story out of our stores of data.

The phrase “A picture is worth a thousand words” is expressly true when turning huge piles of data into images a viewer can actually understand and derive meaning from. Children’s storybooks contain lots of images, but very few words. As kids, we don’t know many words, but the visuals allow us to easily understand the story.

In our modern digital world, we have [huge amounts of data](https://www.sisense.com/blog/delivering-more-impactful-insights-from-your-cloud-data/) all around us. Data scientists and ML engineers get most of the data they deal with data in a structured or unstructured data format, however, it’s difficult for humans to understand and analyze this. Data visualizations (or graphical representations of data) are vital for understanding the data. They help users explore data through visual elements like charts, graphs, plots, maps, and other visualizations.

## Different types of exploratory data analysis

In every dataset, we have many **variables**(also called features, input-variables, or independent-variables) and **target/output variables**(also known as labels, dependent-variables, classes, or class-labels). The data scientist’s job is to completely understand each feature individually and the relationship between different features. The goal is to get ready the dataset for ML algorithms implementation.

We have three methods for exploratory data analysis:

### Univariate analysis

In the univariate analysis, each variable is analyzed individually. It will get us to the complete statistical data for each feature. There are a variety of data visualization techniques for univariate analysis, including **Box Plot, Histogram, PDF, CDF**

### Bivariate analysis

Bivariate analysis is performed to find the relationship between each feature with the target variable.  Data visualization techniques for bivariate analysis are **Scatter Plot**and **Heatmap**

### Multivariate Analysis

As the name signifies, multivariate analysis is performed to understand the relationship between different features of the dataset. One of the main multivariate analysis data visualization techniques is the **Pair Plot**.

## Data Visualization in Python

There are a wide array of libraries you can use to create Python data visualizations, including **Matplotlib, seaborn, Plotly**, and others. A Python data visualization helps a user understand data in a variety of ways: Distribution, mean, median, outlier, skewness, correlation, and spread measurements.

### Bar Plot

A bar plot is a plot that presents categorical data with rectangular bars. The length or height of bars is proportional to the frequency of the category. We can count the values of various categories using bar plots.

### Pie Chart

Pie Chart is a circular chart that uses pie slices to show the relative size of data. The arc length of each pie slice is proportional to the quantity it represents. It works beautifully on categorical values. There are different variants of pie charts available.

### Box-plot

Box-plot gives us a five-number summary of any variable: the minimum, maximum, the sample median, the first and third quartile. Box-plot helps in measuring two observations:

1. Skewness of distribution  
2. Outliers (Outliers comes outside the box-plot)

### Histogram and PDF

A histogram is a graphical representation of the distribution of numerical data. It is an estimate of the probability distribution of a continuous variable. Histogram basically represents the number of points that exist for each bin(range of values). PDF is a **Probability Density Function** which is basically smoothening of the histogram.

### Scatter Plots

A scatter plot is a plot that shows the relationship between two variables of a data set.

### Heat Map

A heatmap is a graphical representation of data in which data values are represented as colors. It uses color in order to communicate the correlation between two variables. Values are between -1 to 1. 1 denotes perfect positive correlation. 0 means no correlation and -1 means the highest negative correlation.

### Line chart

The line chart represents a series of data points connected by a straight line. It is generally used to visualize data that changes over time.

**III Implementation-**

**Dataset :** Tipping data

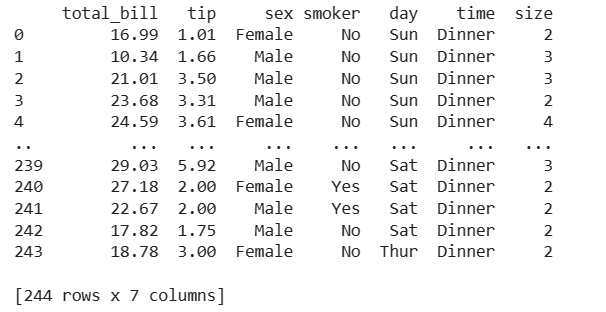
**Description :** It is a dataset of tips received by a waiter over the period of few months The dataset contains 244 records. There are 7 variables in this dataset such as tip, bill, day, time etc

**Code:**

import plotly.express as px

df = px.data.tips()

print(df)



1. **Bubble Chart**

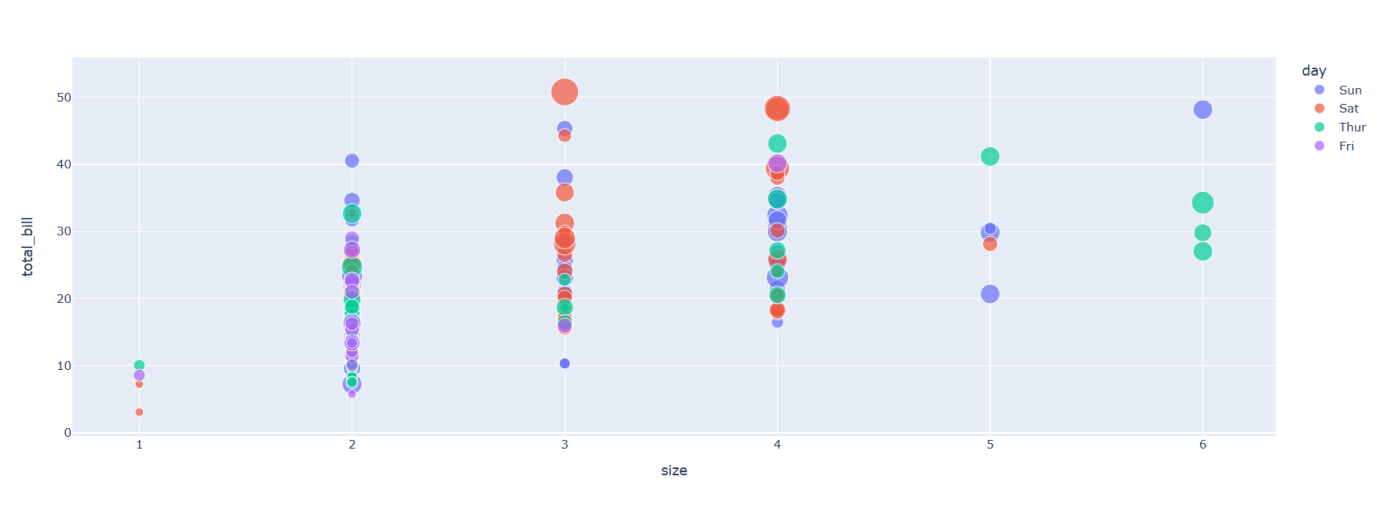
fig = px.scatter(df, x="size", y="total\_bill",

color="day",

size='tip',

hover\_data=['sex'])

fig.show()

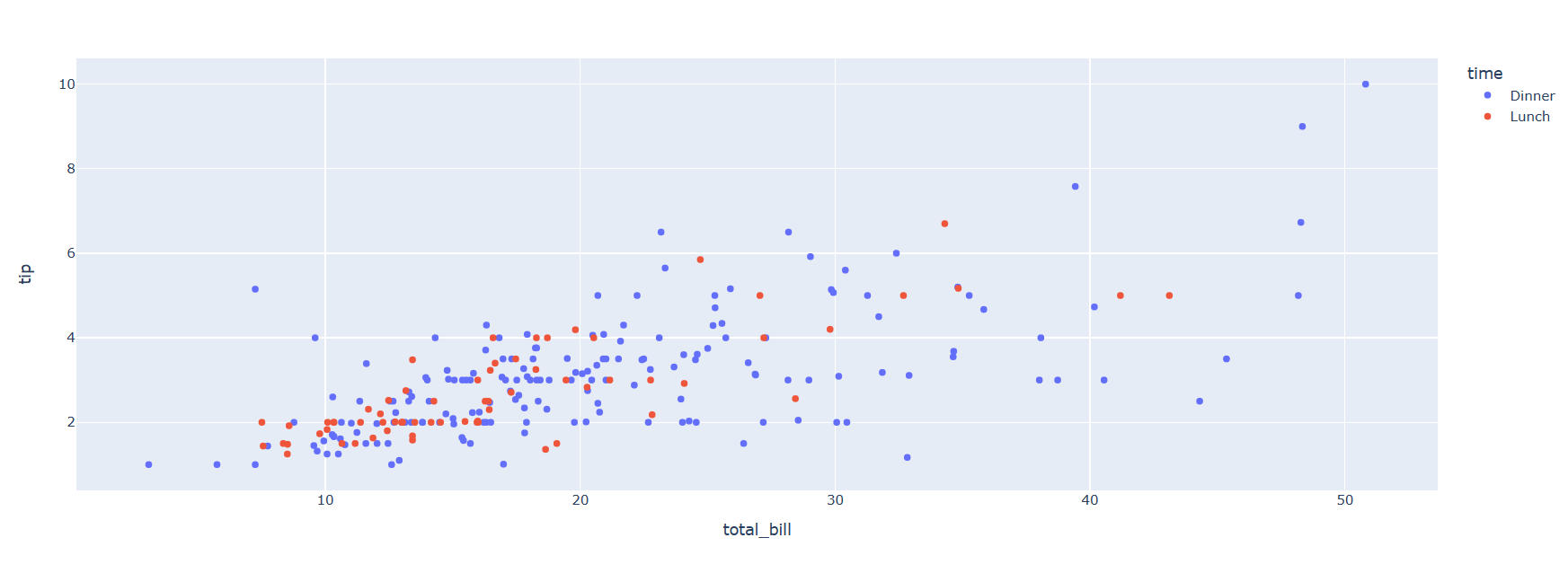


**Figure 1 :- Bubble Chart**

1. **Scatter Plot**

fig1 = px.scatter(df, x="total\_bill", y="tip", color="time")

fig1.show()



**Figure 2 :- Scatter Plot**

1. **Heatmaps**

hd = []

ld = []

dd = []

tld = []

tdd =[]

fld = []

fdd =[]

sald = []

sadd =[]

suld = []

sudd =[]

for i in range(244):

if df["time"][i] == 'Lunch':

if df["day"][i]== 'Thur':

tld.append(df["size"][i])

elif df["day"][i]== 'Fri':

fld.append(df["size"][i])

elif df["day"][i]== 'Sat':

sald.append(df["size"][i])

else:

suld.append(df["size"][i])

else:

if df["day"][i]== 'Thur':

tdd.append(df["size"][i])

elif df["day"][i]== 'Fri':

fdd.append(df["size"][i])

elif df["day"][i]== 'Sat':

sadd.append(df["size"][i])

else:

sudd.append(df["size"][i])

ld.append(sum(tld))

ld.append(sum(fld))

ld.append(sum(sald))

ld.append(sum(suld))

dd.append(sum(tdd))

dd.append(sum(fdd))

dd.append(sum(sadd))

dd.append(sum(sudd))

hd.append(ld)

hd.append(dd)

fig = px.imshow(hd,

labels=dict(x="day", y="time", color="size"),

x=['Thur', 'Fri','Sat','Sun'],

y=['Lunch','Dinner']

)

fig.update\_xaxes(side="top")

fig.show()

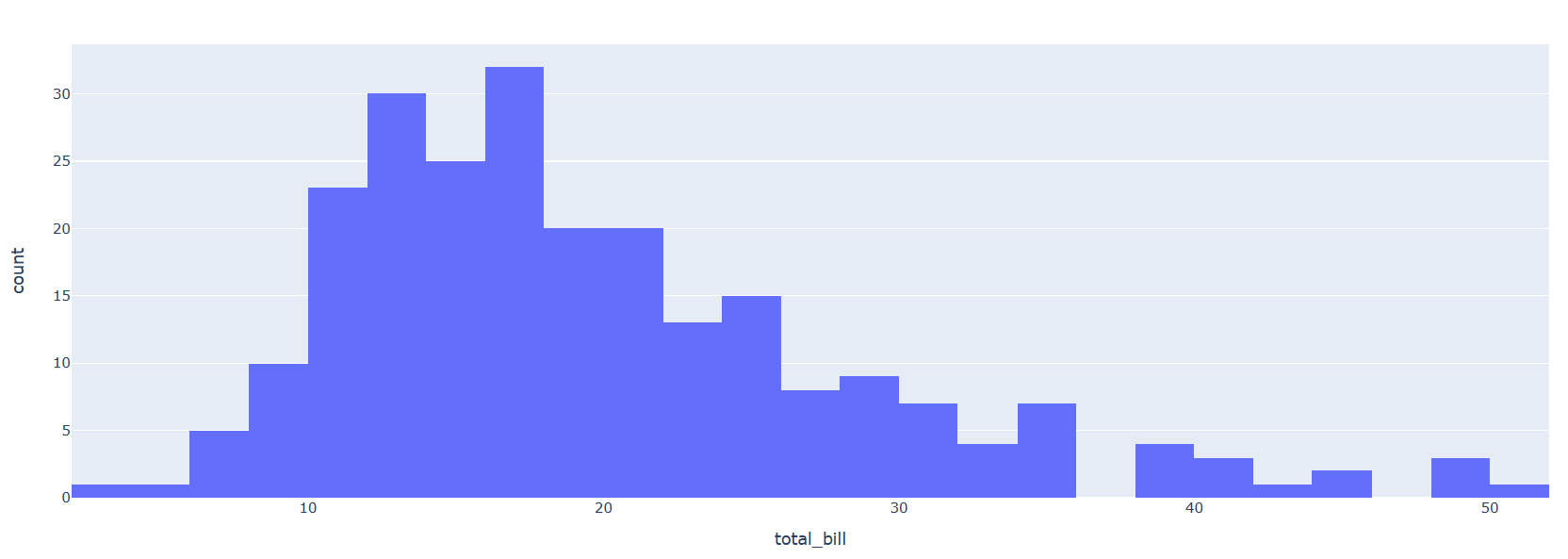


**Figure 3 :- Heatmaps**

1. **Histogram**

fig = px.histogram(df, x="total\_bill")

fig.show()

**Figure 4 :- Histogram**

1. **Boxplot**

df["total\_bill"].describe()]

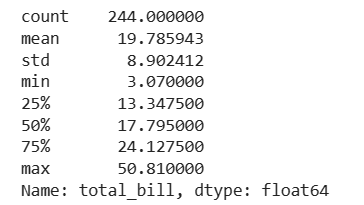
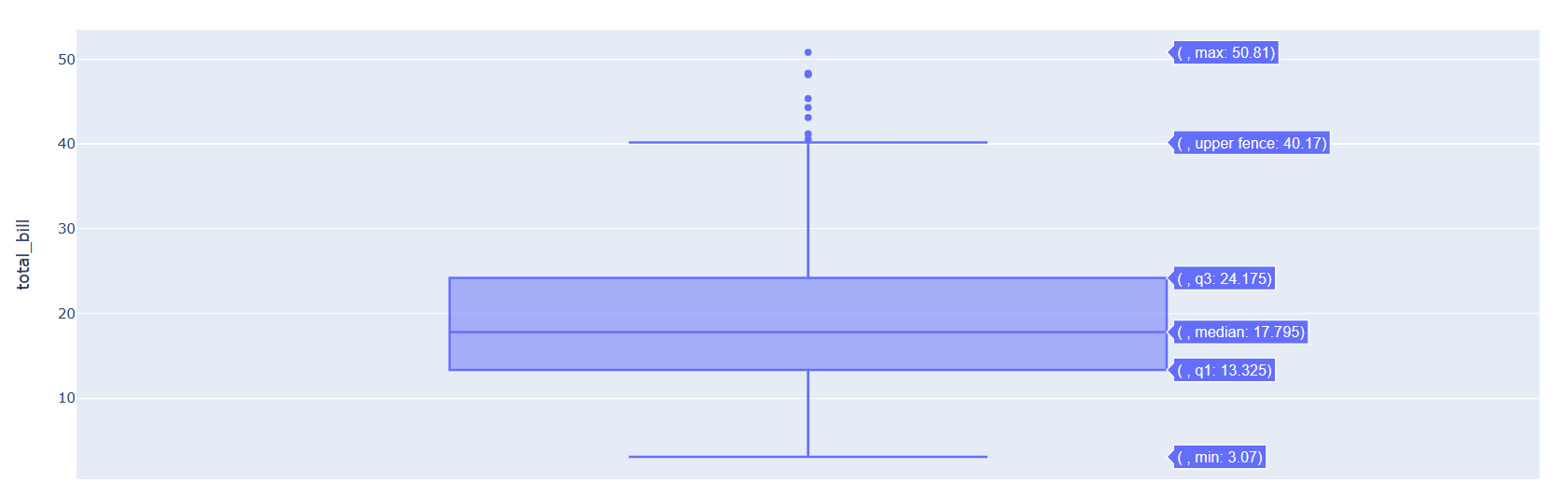


fig = px.box(df, y="total\_bill")

fig.show()



**Figure 5 :- Boxplot**

1. **Density Plot**

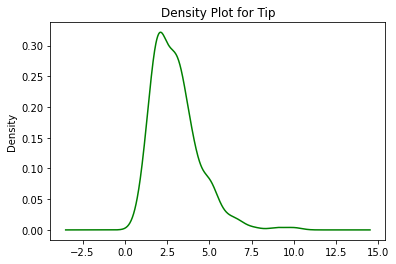
import matplotlib.pyplot as plt

# density plot for 'tip'

df['tip'].plot.density(color='green')

plt.title('Density Plot for Tip')

plt.show()



**Figure 6 :- Density Plot**

1. **Ogives less than**

import numpy as np

data = df["tip"]

print(max(data))

print(min(data))



# creating class interval

classInterval = [0,1,2,3,4,5,6,7,8,9,10]

# calculating frequency and class interval

values, base = np.histogram(data, bins=classInterval)

# calculating cumulative sum

cumsum = np.cumsum(values)

# plotting the ogive graph

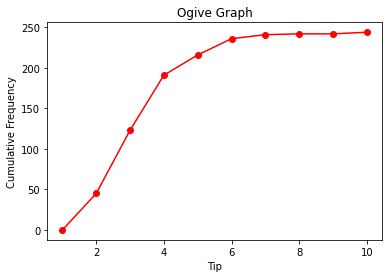
plt.plot(base[1:], cumsum, color='red', marker='o', linestyle='-')

# formatting

plt.title('Ogive Graph')

plt.xlabel('Tip')

plt.ylabel('Cumulative Frequency')



**Figure 7 :- Ogives less than**

1. **Ogives more than**

# reversing cumulative frequency

res = np.flipud(cumsum)

# plotting ogive

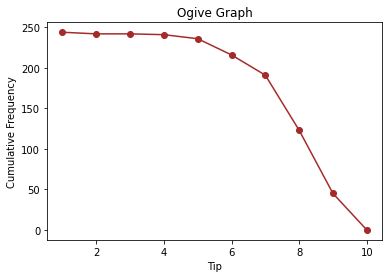
plt.plot(base[1:], res, color='brown', marker='o', linestyle='-')

# formatting the graph

plt.title('Ogive Graph')

plt.xlabel('Tip')

plt.ylabel('Cumulative Frequency')



**Figure 8 :- Ogives more than**

1. **Pie Chart**

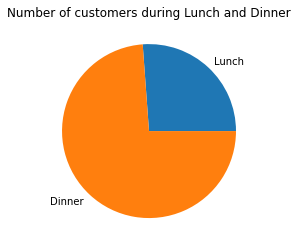
cd = [sum(ld),sum(dd)]

time = ['Lunch','Dinner']

plt.pie(cd, labels = time)

plt.title("Number of customers during Lunch and Dinner")

plt.show()

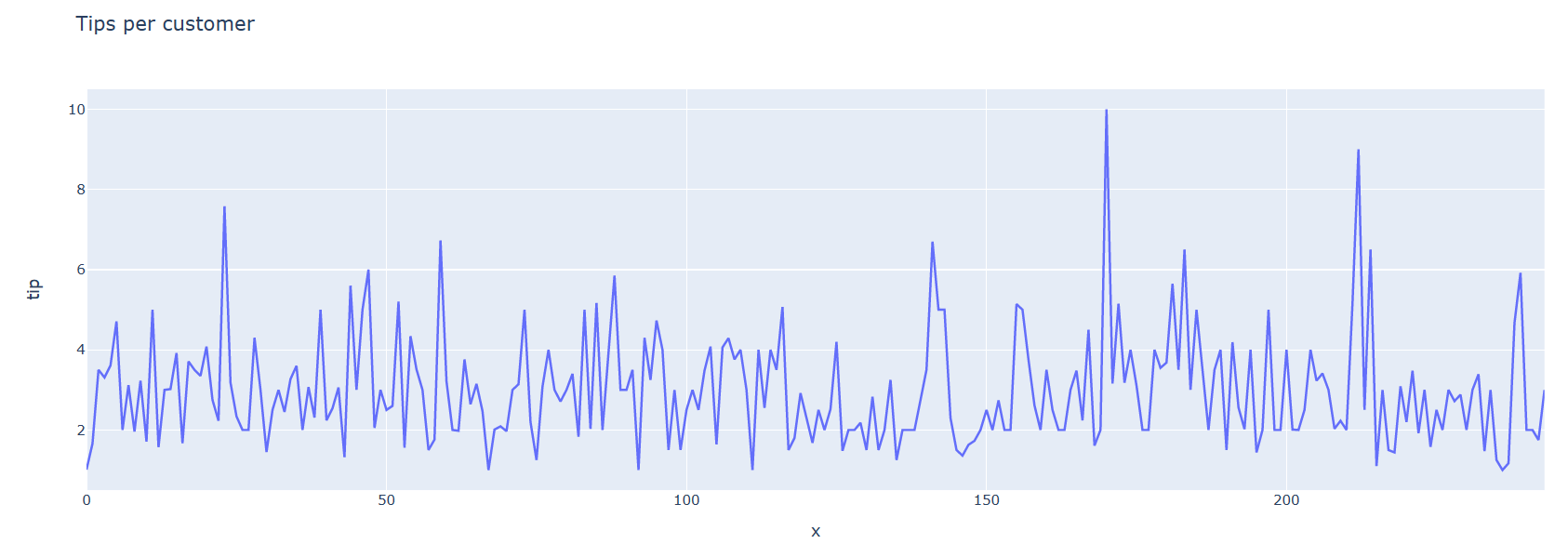


**Figure 9 :- Pie Diagram**

1. **Line Chart**

fig = px.line(df, x=range(244), y="tip", title='Tips per customer')

fig.show()



**Figure 10 :- Line Chart**

**IV- CONCLUSION**

We have understood thebasic concepts of data visualization and techniques. We also explored the various types of data visualization techniques that exist.

**V-REFERENCES**

**https://www.sisense.com/blog/data-visualizations-in-python-and-r/**

**https://gilberttanner.com/blog/introduction-to-data-visualization-inpython/**

**https://www.geeksforgeeks.org/data-visualization-with-python/**

**https://plotly.com/python/plotly-express/**

**https://plotly.com/python-api-reference/generated/plotly.data.html**

**https://www.geeksforgeeks.org/density-plots-with-pandas-in-python/**

**https://www.geeksforgeeks.org/how-to-create-an-ogive-graph-in-python/**

**VI- POST LAB QUESTION/ANSWER**

## Q1 List popular plotting libraries:

* [**Matplotlib:**](https://matplotlib.org/)low level, provides lots of freedom
* [**Pandas Visualization:**](https://pandas.pydata.org/pandas-docs/stable/visualization.html)easy to use interface, built on Matplotlib
* [**Seaborn:**](https://seaborn.pydata.org/)high-level interface, great default styles
* [**plotnine:**](https://plotnine.readthedocs.io/en/stable/)based on R’s ggplot2, uses [Grammar of Graphics](https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448)
* [**Plotly:**](https://plot.ly/python/)can create interactive plots