Name: Allan Rodrigues
Class: TE IT A
Roll no: 59
Pid:191104

St. Francis Institute of Technology, Mumbai-400 103
Department of Information Technology

A.Y. 2021-2022
Class: TE-ITA/B, Semester: VI

Subject: **Data Science Lab**

Experiment – 2: To implement Data Visualization/Data Exploratory Analysis using Matplolib and Seaborn.

1. **Aim:** To implement Data Visualization using Matplotlib and Seaborn.
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand Matplotlib Functions
   - Understand Seaborn Functions
3. **Outcomes:** After study of this experiment, the student will be able to
   - Understand data visualization /Exploratory Data Analysis using Matplolib and Seaborn.
4. **Prerequisite:** Fundamentals of Python Programming and Database Management System.
5. **Requirements:** Python Installation, Personal Computer, Windows operating system, Internet Connection, Microsoft Word.
6. **Pre-Experiment Exercise:**
   Brief Theory:
   Basic Concepts of Matplolib and Seaborn.
7. **Laboratory Exercise**
   A. Procedure:

```
import numpy as np
a=np.array([[1,2,3],[4,5,6],[7,8,9]])
a

"""**matplotlib**

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations
in Python. Matplotlib makes easy things easy and hard things possible.

1. Create publication quality plots.
2. Make interactive figures that can zoom, pan, update.
3. Customize visual style and layout.
4. Export to many file formats .
5. Embed in JupyterLab and Graphical User Interfaces.
6. Use a rich array of third-party packages built on Matplotlib.

**matplotlib.pyplot**

matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot
function makes some change to a figure: e.g., creates a figure, creates a plotting area in a
figure, plots some lines in a plotting area, decorates the plot with labels, etc.

**numpy.pi**

While both numpy.pi and math.pi provides us with the same value, the advantage of numpy.pi is that
it helps us by avoiding the dependency of obtaining the constant value of pi from a different
library.

**np.sin**

This mathematical function helps user to calculate trigonometric sine for all x(being the array
elements).
"""

# Sinusoidal plot
```

```
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(0.0,2.0,0.01)
y=np.sin(2*np.pi*x)
plt.plot(x,y)

#Barplot
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
x=[1,2,3,4,5,6,7]
y=[23,43,65,32,87,45,90]
plt.bar(x,y,color='orange')
plt.title('Average Score of Students')
plt.xlabel("Students")
plt.ylabel("Avg. Score")
fig.savefig("testing.jpg")

"""**Error Bars**

An error bar is a line through a point on a graph, parallel to one of the axes, which represents
the uncertainty or variation of the corresponding coordinate of the point.

Error bars can communicate the following information about your data:

1. How spread the data are around the mean value (small SD bar = low spread, data are clumped
around the mean; larger SD bar = larger spread, data are more variable from the mean).

2. The reliability of the mean value as a representative number for the data set.  In other words,
how accurately the mean value represents the data (small SD bar = more reliable, larger SD bar =
less reliable).  It's important to note that just because you have a larger SD, it does not
indicate your data is not valid.
3. The likelihood of there being a significant difference between data sets.

**Significant Difference**

A "significant difference" means that the results that are seen are most likely not due to chance
or sampling error.  In any experiment or observation that involves sampling from a population,
there is always the possibility that an observed effect would have occurred due to sampling error
alone.  But if result is "significant,"  then the investigator may conclude that the observed
effect actually reflects the characteristics of the population rather than just sampling error or
chance.

#single barplots with error bars
N=5
men=(20,35,30,36,27)
women=(25,32,34,20,25)
mstd=(2,2,2,2,2)
i=np.arange(N)
p1=plt.bar(i,men,width=0.35,yerr=mstd)
p2=plt.bar(i,women,width=0.35,bottom=men,yerr=mstd)
plt.xticks(i,('G1','G2','G3','G4','G5'))
plt.yticks(np.arange(0,90,10))
plt.legend((p1[0],p2[0]),('Men','Women'))

"""**Histogram**

A histogram is a graphical representation of a grouped frequency distribution with continuous
classes.

A histogram is a diagram involving rectangles whose area is proportional to the frequency of a
variable and width is equal to the class interval.

The histogram graph is used under certain conditions. They are:

1. The data should be numerical.
2. A histogram is used to check the shape of the data distribution.
3. Used to check whether the process changes from one period to another.
4. Used to determine whether the output is different when it involves two or more processes.
5. Used to analyse whether the given process meets the customer requirements.

Histograms and bar charts are different. In the bar chart, each column represents the group which
is defined by a categorical variable, whereas in the histogram each column is defined by the
continuous and quantitative variable.

**Difference Between Histogram and Bar Graph**

1. It is a two-dimensional figure
2. The frequency is shown by the area of each rectangle
3. It shows rectangles touching each other

**Bar Graph**

1. It is a one-dimensional figure
2. The height shows the frequency and the width has no significance.
3. It consists of rectangles separated from each other with equal spaces.

"""
```

```
#Histogram
m=100
std=15
x=m+std*np.random.randn(480)
y=50
plt.hist(x,y,color='blue',density=False)
plt.grid(color='gray',linestyle='--',linewidth=2,axis='x')
plt.grid(color='gray',linestyle='--',linewidth=2,axis='y')

#Pie Chart
labels='cricket','Hockey','Tennis','Football'
v=[40,20,10,30]
e=(0,0,0.1,0)
plt.pie(v,explode=e,labels=labels,autopct='%1.3f%%',shadow=True,startangle=0)

"""numpy.linspace(start, stop, num)
Return evenly spaced numbers over a specified interval.

Returns num evenly spaced samples, calculated over the interval [start, stop].

The endpoint of the interval can optionally be excluded.
"""

x=np.linspace(0,10,1000)
plt.plot(x,np.sin(x))
plt.plot(x,np.sin(x-1),color='g')
plt.plot(x,np.sin(x-2),color='0.25')
plt.plot(x,np.sin(x-3),color='red')

x=np.linspace(0,10,1000)
plt.plot(x,np.sin(x))
plt.plot(x,np.sin(x-1),linestyle='solid')
plt.plot(x,np.sin(x-2),linestyle='dashed')
plt.plot(x,np.sin(x-3),linestyle='dashdot')
plt.plot(x,np.sin(x-4),linestyle='dotted')

#plt.plot(x,np.sin(x))
plt.plot(x,np.sin(x-1),linestyle='-')
plt.plot(x,np.sin(x-2),linestyle='--')
plt.plot(x,np.sin(x-3),linestyle='-.')
plt.plot(x,np.sin(x-4),linestyle=':')

#Scatterplot
x=np.linspace(0,10,50)
y=np.sin(x)
plt.scatter(x,y)

x=np.linspace(0,10,50)
plt.plot(x,np.sin(x),'-o')


#Seaborn
import seaborn as sns
data=sns.load_dataset("iris")
sns.lineplot(x="sepal_length", y="sepal_width",data=data)

sns.scatterplot(x="sepal_length", y="sepal_width",data=data)

sns.stripplot(x="species", y="sepal_length",data=data)



sns.violinplot(x="species", y="sepal_length",data=data)

sns.swarmplot(x="species", y="sepal_length",data=data)
```

Paste Screenshots of above commands.

exp2.ipynb

File Edit View Insert Runtime Tools Help Saving...

Comment Share

+ Code + Text

RAM Disk Editing

```
[ ] import matplotlib.pyplot as plt
    import numpy as np
    a = np.array([[1,2,3],[4,5,6],[7,8,9]])
    a
```

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

```
[ ] a=np.arange(0.1,2.0,0.01)
    a
```

```
array([0.1 , 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ,
       0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31,
       0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42,
       0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 , 0.51, 0.52, 0.53,
       0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 , 0.61, 0.62, 0.63, 0.64,
       0.65, 0.66, 0.67, 0.68, 0.69, 0.7 , 0.71, 0.72, 0.73, 0.74, 0.75,
       0.76, 0.77, 0.78, 0.79, 0.8 , 0.81, 0.82, 0.83, 0.84, 0.85, 0.86,
       0.87, 0.88, 0.89, 0.9 , 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97,
       0.98, 0.99, 1.  , 1.01, 1.02, 1.03, 1.04, 1.05, 1.06, 1.07, 1.08,
       1.09, 1.1 , 1.11, 1.12, 1.13, 1.14, 1.15, 1.16, 1.17, 1.18, 1.19,
       1.2 , 1.21, 1.22, 1.23, 1.24, 1.25, 1.26, 1.27, 1.28, 1.29, 1.3 ,
       1.31, 1.32, 1.33, 1.34, 1.35, 1.36, 1.37, 1.38, 1.39, 1.4 , 1.41,
       1.42, 1.43, 1.44, 1.45, 1.46, 1.47, 1.48, 1.49, 1.5 , 1.51, 1.52,
       1.53, 1.54, 1.55, 1.56, 1.57, 1.58, 1.59, 1.6 , 1.61, 1.62, 1.63,
       1.64, 1.65, 1.66, 1.67, 1.68, 1.69, 1.7 , 1.71, 1.72, 1.73, 1.74,
       1.75, 1.76, 1.77, 1.78, 1.79, 1.8 , 1.81, 1.82, 1.83, 1.84, 1.85,
```

exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

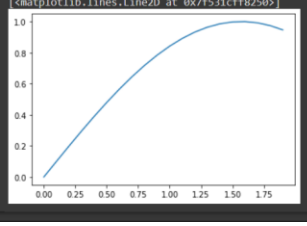
Comment Share

+ Code + Text

RAM Disk Editing

```
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(0.0,2.0,0.1)
y=np.sin(x)
plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7f531cff8250>]
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

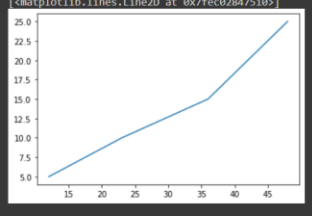
Comment Share

+ Code + Text

RAM Disk Editing

```
[ ] x=[12,23,36,48]
    y=[5,10,15,25]
    plt.plot(x,y)
```

```
[<matplotlib.lines.Line2D at 0x7fec02847510>]
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

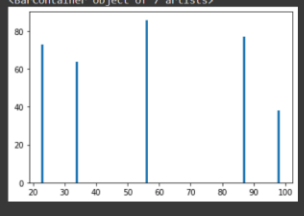
+ Code + Text

RAM Disk Editing

```
[ ] x=[34,56,87,98,56,34,23]
    y=[39,46,77,38,86,64,73]

    plt.bar(x,y)
```

```
<BarContainer object of 7 artists>
```



exp2.ipynb

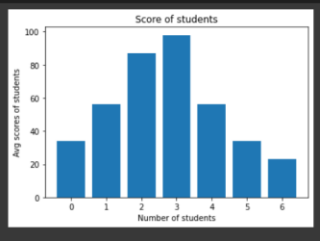
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
import matplotlib.pyplot as plt
fig= plt.figure()
x=[34,56,87,98,56,34,23]
plt.bar(range(len(x)),x)
plt.title("Score of students")
plt.xlabel("Number of students")
plt.ylabel("Avg scores of students")
fig.savefig("test.jpg")
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

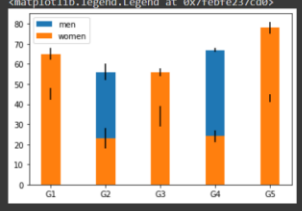
Comment Share

+ Code + Text

RAM Disk Editing

```
men=(45,56,34,67,43)
women=(65,23,56,24,78)
mstd=(3,4,5,1,2)
fstd=(3,5,2,3,3)
i = np.arange(len(men))
p1 = plt.bar(i,men,width=0.35,yerr=mstd)
p2= plt.bar(i,women,width=0.35,yerr=fstd)
plt.xticks(i,('G1','G2','G3','G4','G5'))
plt.yticks(np.arange(0,90,10))
plt.legend((p1[0],p2[0]),('men','women'))
```

```
<matplotlib.legend.Legend at 0x7febfe237cd0>
```




```
np.random.seed(123456)
```

exp2.ipynb

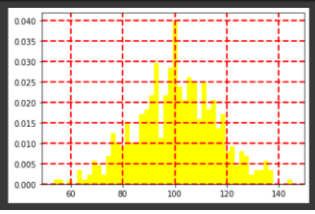
File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Editing

```
m=100;
std=15;
x=m+std*np.random.randn(480)
y=50;
plt.hist(x,y,color='yellow',density=True)
plt.grid(color='red',linestyle='--',axis='y',linewidth=2,alpha=2)
plt.grid(color='red',linestyle='--',axis='x',linewidth=2,alpha=3)
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

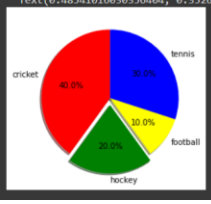
Comment Share

+ Code + Text

RAM Disk Editing

```
labels='cricket','hockey','football','tennis'
v=[40,20,10,30]
e=[0,0.1,0,0]
colors=['red','green','yellow','blue']
plt.pie(v,labels=labels,shadow=True,startangle=90,explode=e,autopct='%1.1f%%',colors=colors)
```

```
([<matplotlib.patches.Wedge at 0x7febfce74990>,
  <matplotlib.patches.Wedge at 0x7febfce81310>,
  <matplotlib.patches.Wedge at 0x7febfce81e50>,
  <matplotlib.patches.Wedge at 0x7febfce8a910>],
 [Text(-1.046162174289765, 0.3399186742226879, 'cricket'),
  Text(5.617605452084956e-08, -1.1999999999999986, 'hockey'),
  Text(0.8899187331606258, -0.6465637233635886, 'football'),
  Text(0.8899186272232008, 0.6465638691739385, 'tennis')],
 [Text(-0.5706339132489631, 0.18541018593964795, '40.0%'),
  Text(3.2769365137162226e-08, -0.6999999999999992, '20.0%'),
  Text(0.48541021808761 4, -0.3526711218346847, '10.0%'),
  Text(0.48541016030356404, 0.3526712013676027 7, '30.0%')])
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

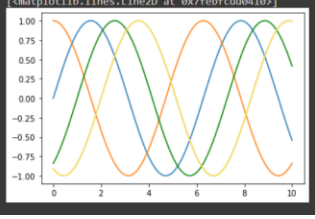
Comment Share

+ Code + Text

RAM Disk Editing

```
x=np.linspace(0,10,100)
plt.plot(x,np.sin(x))
plt.plot(x,np.cos(x))
plt.plot(x,np.sin(x-1),color='g')
plt.plot(x,np.sin(x-2),color='0.75')
plt.plot(x,np.sin(x-2),color='#FFDD44')
```

```
[<matplotlib.lines.Line2D at 0x7febfcdd0410>]
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

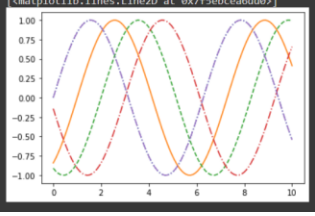
Comment Share

+ Code + Text

RAM Disk Editing

```
x=np.linspace(0,10,100)
plt.plot(x,np.sin(x),linestyle='dotted')
plt.plot(x,np.sin(x-1),linestyle='solid')
plt.plot(x,np.sin(x-2),linestyle='dashed')
plt.plot(x,np.sin(x-3),linestyle='dashdot')
plt.plot(x,np.sin(x),linestyle='-.')
```

```
[<matplotlib.lines.Line2D at 0x7f5ebcea6dd0>]
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

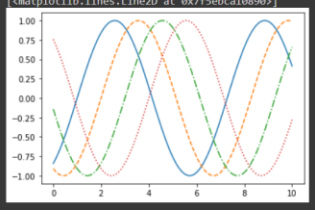
Comment Share

+ Code + Text

RAM Disk Editing

```
[ ] plt.plot(x,np.sin(x-1),linestyle='-')
    plt.plot(x,np.sin(x-2),linestyle='--')
    plt.plot(x,np.sin(x-3),linestyle='-.')
    plt.plot(x,np.sin(x-4),linestyle=':')
```

```
[<matplotlib.lines.Line2D at 0x7f5ebca10890>]
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

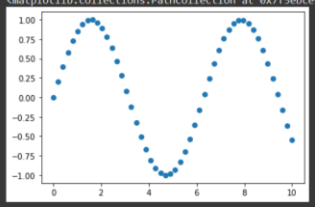
Comment Share

+ Code + Text

RAM Disk Editing

```
[ ] x=np.linspace(0,10,50)
    y=np.sin(x)
    plt.scatter(x,y)
```

```
<matplotlib.collections.PathCollection at 0x7f5ebce977d0>
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

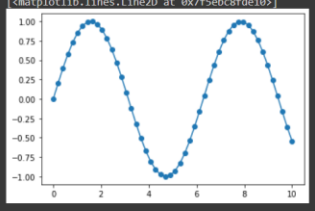
Comment Share

+ Code + Text

RAM Disk Editing

```
[ ] x=np.linspace(0,10,50)
    plt.plot(x,np.sin(x),'-o')
```

```
[<matplotlib.lines.Line2D at 0x7f5ebc8fde10>]
```



```
[ ] import seaborn as sns
    data=sns.load_dataset("iris")
    data
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa

exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

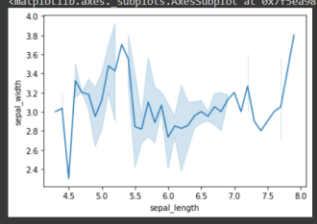
Comment Share

+ Code + Text

RAM Disk Editing

```
[ ] sns.lineplot(x="sepal_length", y="sepal_width",data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ea9873950>
```




```
[ ] sns.scatterplot(x="sepal_length", y="sepal_width",data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ea877de10>
```

exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

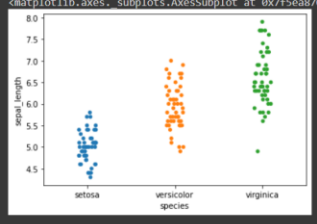
Comment Share

+ Code + Text

RAM Disk Editing

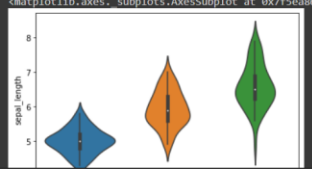
```
[ ] sns.stripplot(x="species", y="sepal_length",data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ea8702750>
```




```
[ ] sns.violinplot(x="species", y="sepal_length",data=data)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ea8665510>
```



exp2.ipynb

File Edit View Insert Runtime Tools Help All changes saved

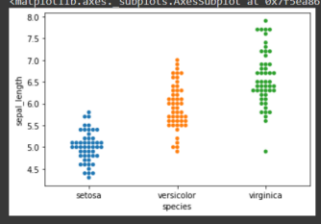
Comment Share

+ Code + Text

RAM Disk Editing

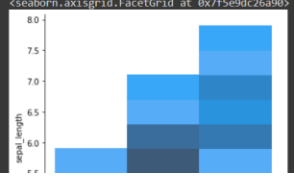
```
[ ] sns.swarmplot(x="species", y="sepal_length",data=data)
```

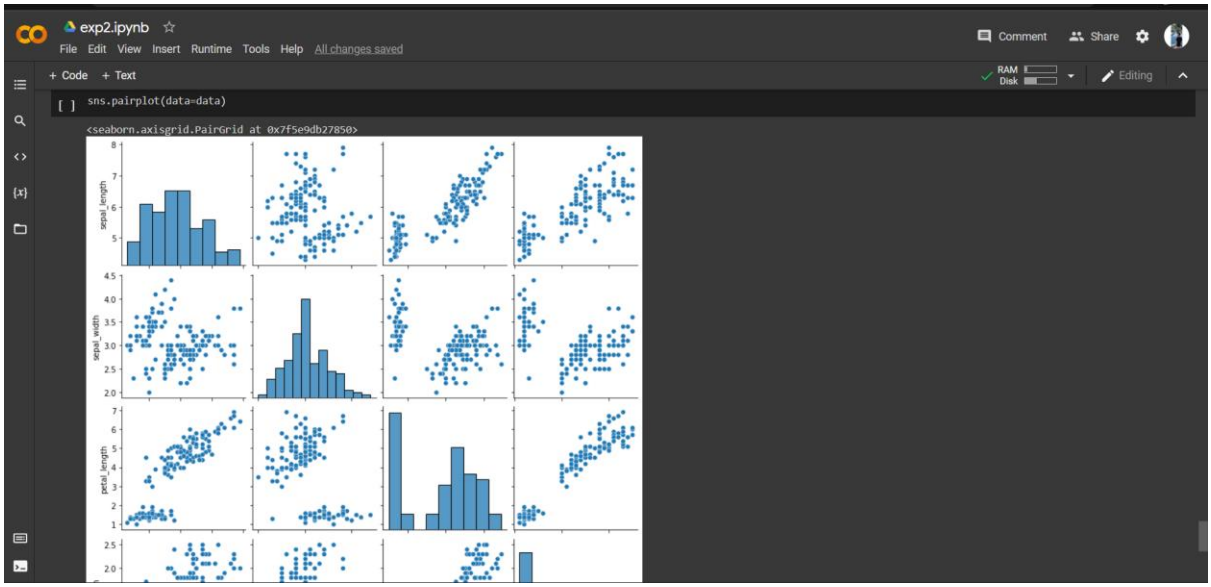
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5ea86705d0>
```




```
[ ] sns.displot(x="species", y="sepal_length",data=data)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f5e9dc26a90>
```





8. Post-Experiments Exercise

A. Extended Theory: (Soft Copy)

What is difference between table and graph? (Ref Link 1)

Table	Graph
A table is a method of representing data in the form of row and columns to get a quick overview of data.	A Graph is a method of representing data in varied forms to a comprehensible understanding of concepts.
A table doesn't make use of symbols because it depicts the details of the data in a formal structure.	A graph makes use of different symbols such as slices, lines, bars, etc to depict the concepts in a better way.
The content is represented in the form of rows and columns. It is predominantly used to present data in the form of numbers, quantities, names, etc.	The content is presented in the form of pie charts, flowcharts, line charts, bars, etc to explain the larger concepts of the data that is presented. It is used to make data understandable.
The table is used to make data brief and quick to overview.	Graphs make the given data or the concept comprehensible.

B. Questions:

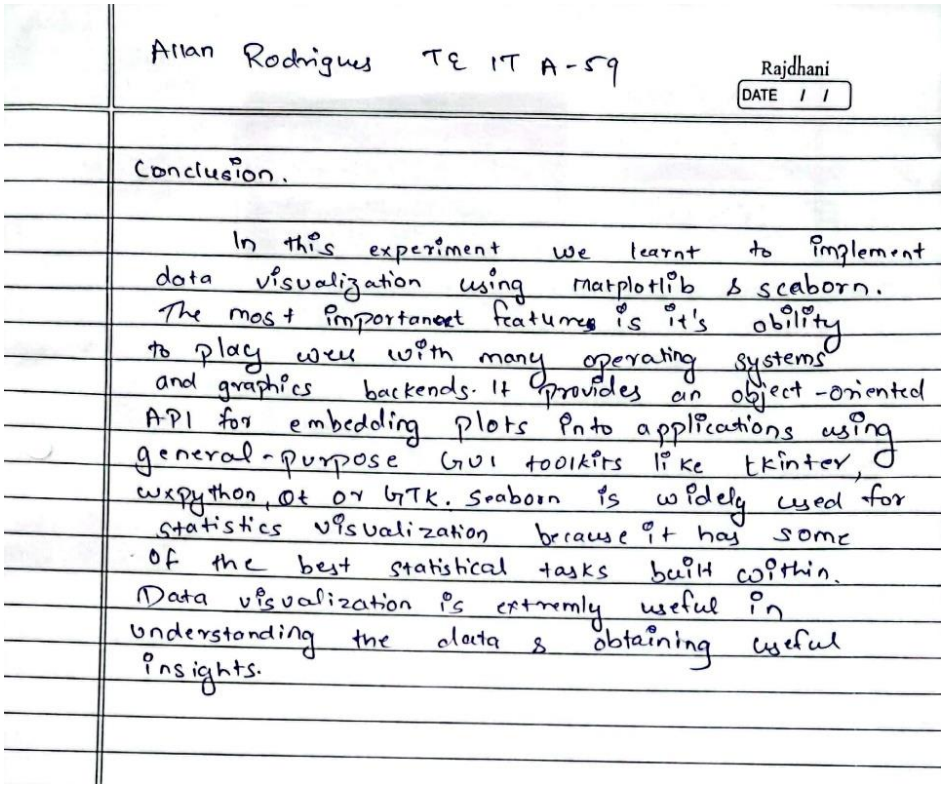
What is the difference between Matplotlib and Seaborn? (Refer Link 2)

Matplotlib	Seaborn
It is utilized for making basic graphs. Datasets are visualised with the help of bargraphs, histograms, piecharts, scatter plots, lines and so on.	Seaborn contains a number of patterns and plots for data visualization. It uses fascinating themes. It helps in compiling whole data into a single plot. It also provides distribution of data.
It uses comparatively complex and lengthy syntax. Example: Syntax for bargraph- matplotlib.pyplot.bar(x_axis, y_axis).	It uses comparatively simple syntax which is easier to learn and understand. Example: Syntax for bargraph- seaborn.barplot(x_axis, y_axis).

We can open and use multiple figures simultaneously. However they are closed distinctly. Syntax to close one figure at a time: matplotlib.pyplot.close(). Syntax to close all the figures: matplotlib.pyplot.close("all")	Seaborn sets time for the creation of each figure. However, it may lead to (OOM) out of memory issues
Matplotlib is well connected with Numpy and Pandas and acts as a graphics package for data visualization in python. Pyplot provides similar features and syntax as in MATLAB. Therefore, MATLAB users can easily study it.	Seaborn is more comfortable in handling Pandas data frames. It uses basic sets of methods to provide beautiful graphics in python.
Matplotlib is a highly customized and robust	Seaborn avoids overlapping of plots with the help of its default themes
Matplotlib plots various graphs using Pandas and Numpy	Seaborn is the extended version of Matplotlib which uses Matplotlib along with Numpy and Pandas for plotting graphs

C. Conclusion:

Write the significance of the topic studied in the experiment.



Allan Rodrigues TE IT A-59 Rajdhani DATE / /

Conclusion.

In this experiment we learnt to implement data visualization using matplotlib & seaborn. The most importantant features is it's ability to play well with many operating systems and graphics backends. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like tkinter, wxpython, Qt or GTK. Seaborn is widely used for statistics visualization because it has some of the best statistical tasks built within. Data visualization is extremly useful in understanding the data & obtaining useful insights.

D. References:

1. https://www.wallstreetmojo.com/graphs-vs-charts/
2. https://www.geeksforgeeks.org/difference-between-matplotlib-vs-seaborn/
3. https://matplotlib.org/stable/tutorials/index
4. https://seaborn.pydata.org/