Department of Computer Science and Engineering

Lab Manual

for

Exploratory Data Analytics

CCS346

Regulation 2021

CCS346 – V Semester



ADHIPARASAKTHI ENGINEERING COLLEGE

Melmaruvathur, Chengalpet District- 603319.

INDEX

Expt. No.	Name of the Experiments	Page No.
1.	Install the Data Analysis and Visualization Tool R/ Python /Tableau Public/Power BI.	1
2.	Perform Exploratory Data Analysis (EDA) on with datasets like email dataset. Export all your emails as a dataset, import them inside a pandas Data Frame, visualize them and get different insights from the data.	24
3.	Working with Numpy Arrays, Pandas Data frame, Basic Plots using Matplotlib	29
4.	Explore various variable and row filters in R for cleaning data. Apply various plot features in R on sample data sets and Visualize	39
5.	Query the database tables and explore natural, equi and outer joins.	44
6.	Perform Data Analysis and representation on a Map using various Map data sets with Mouse Rollover effect, user interaction	50
7.	Build cartographic visualization for multiple datasets involving various countries of the world; states and districts in India etc.	53
8.	Perform EDA on Wine Quality Data Set. Use a case study on a data set and apply the various EDA and visualization techniques and present an analysis report.	62

Exp No : 1 **Install the Data Analysis and Visualization Tool**
Date: **R/ Python /Tableau Public/Power BI.**

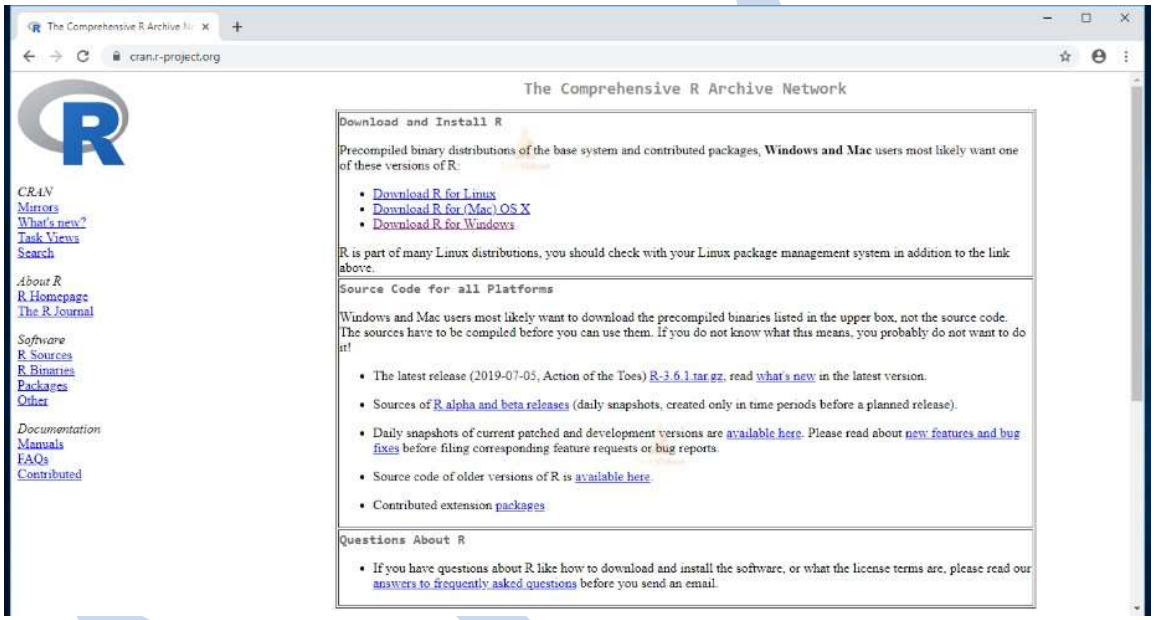
Aim:

To Install the data Analysis and Visualization tool: R/ Python /Tableau Public/ Power BI.

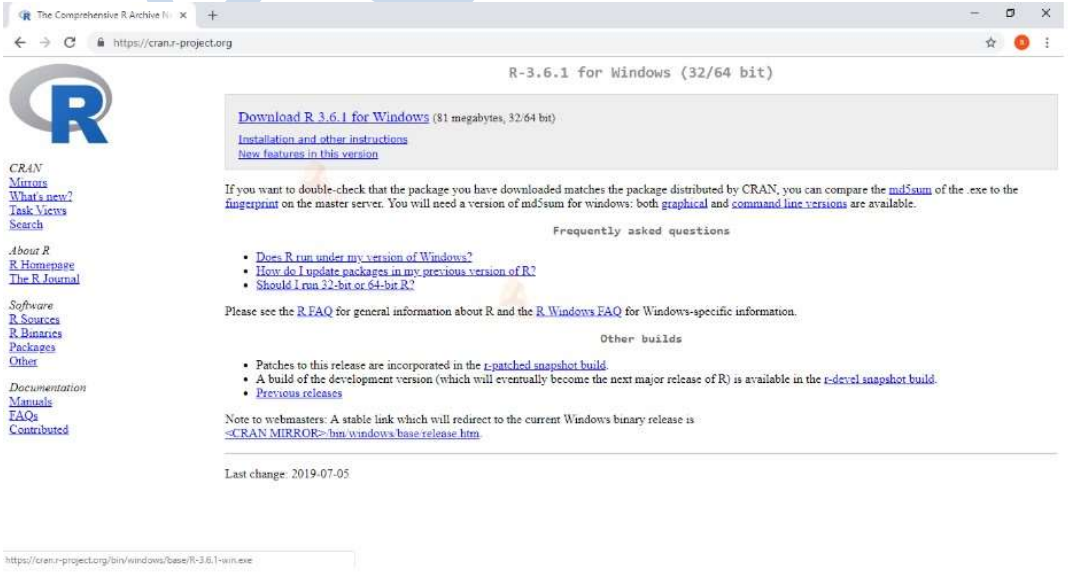
Procedure:

1. Install the data Analysis and Visualization tool: R

Step 1: Go to CRAN R project website.

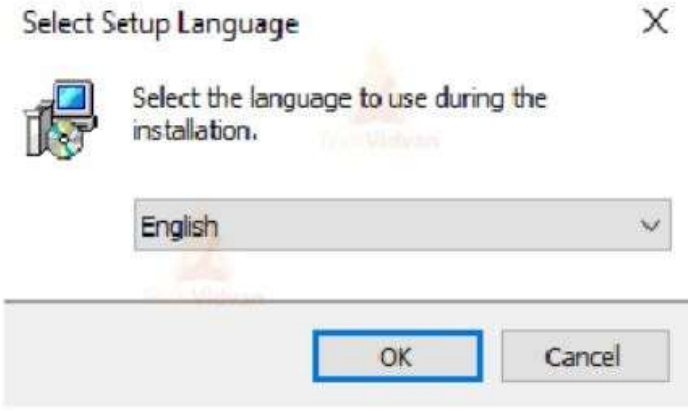


Step 2: Click on the **Download R for Windows** link.
Step 3: Click on the **base** subdirectory link or **install R for the first time** link.
Step 4: Click **Download R X.X.X for Windows** (X.X.X stand for the latest version of R. eg: 3.6.1) and save the executable .exe file.

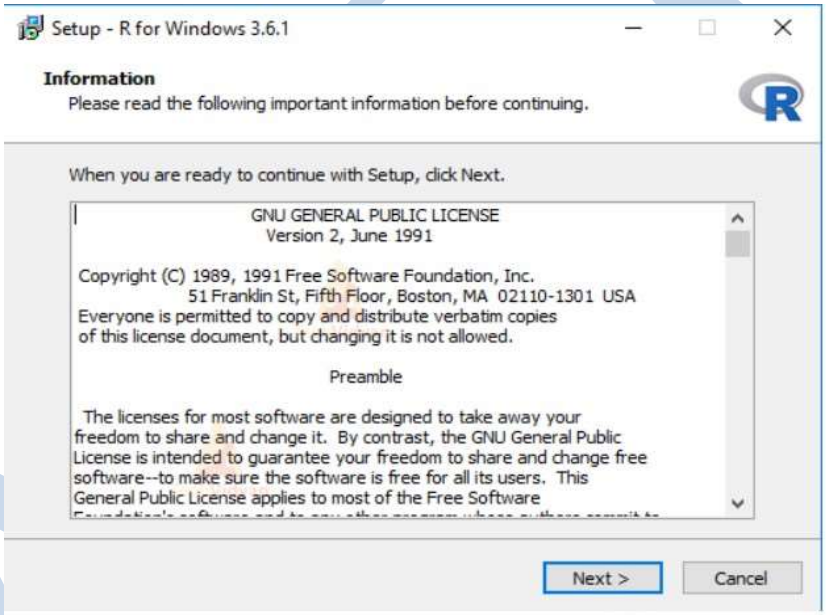


Step 5: Run the **.exe** file and follow the installation instructions.

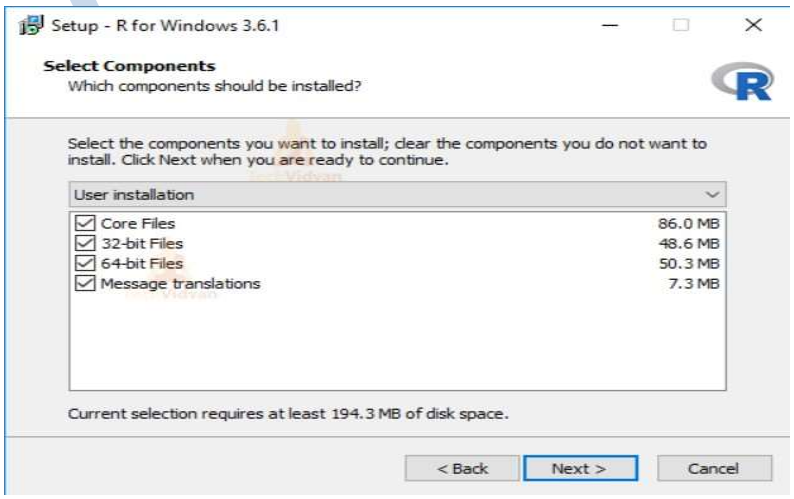
Step 5.a: Select the desired language and then click **Next**.



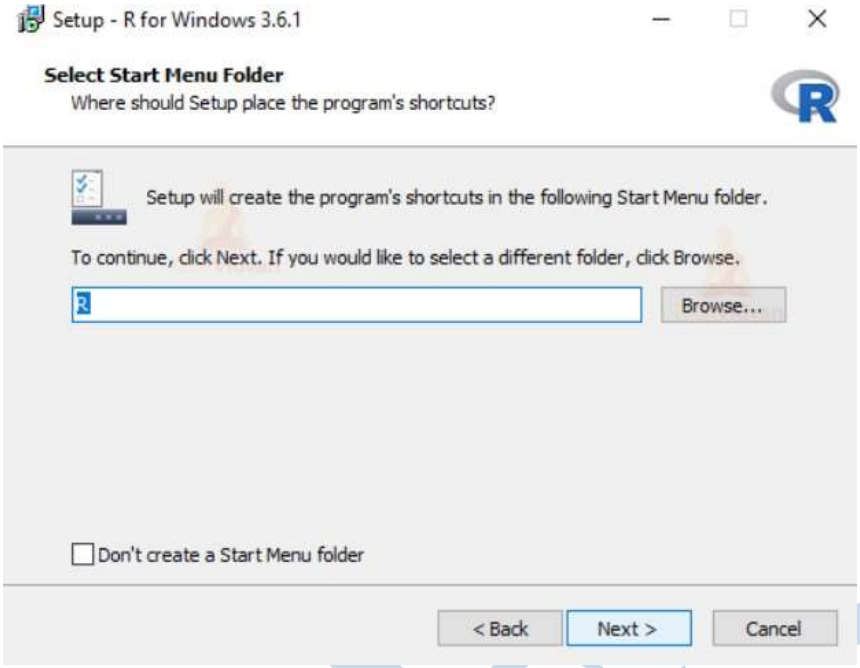
Step 5b: Read the license agreement and click **Next**.



Step 5c: Select the components you wish to install (it is recommended to install all the components). Click **Next**.



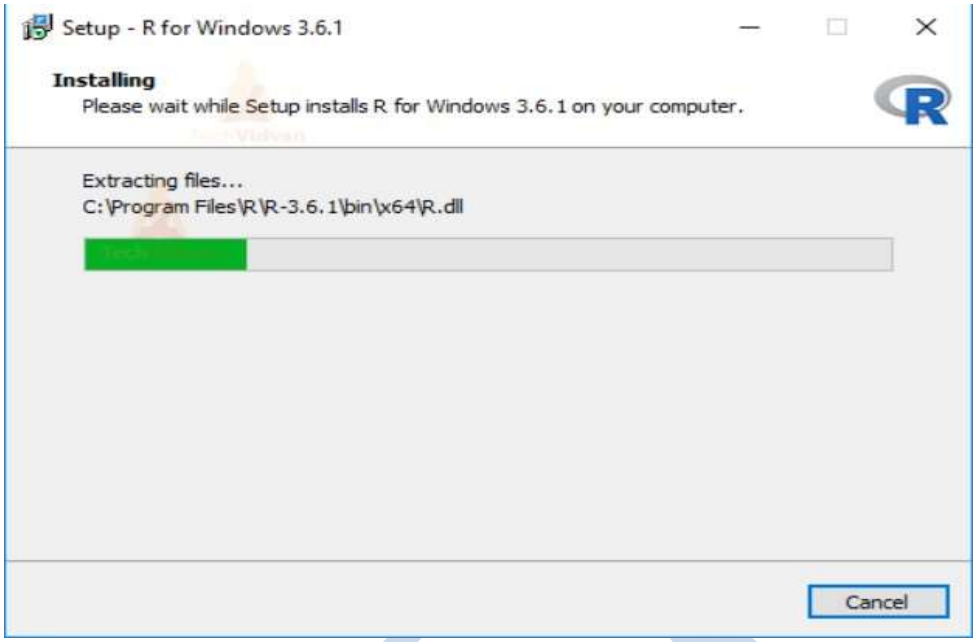
Step 5d: Select the components you wish to install (it is recommended to install all the components). Click **Next**.



Step 5e: Select additional tasks like creating desktop shortcuts etc. then click **Next**.



Step 5f: Wait for the installation process to complete.

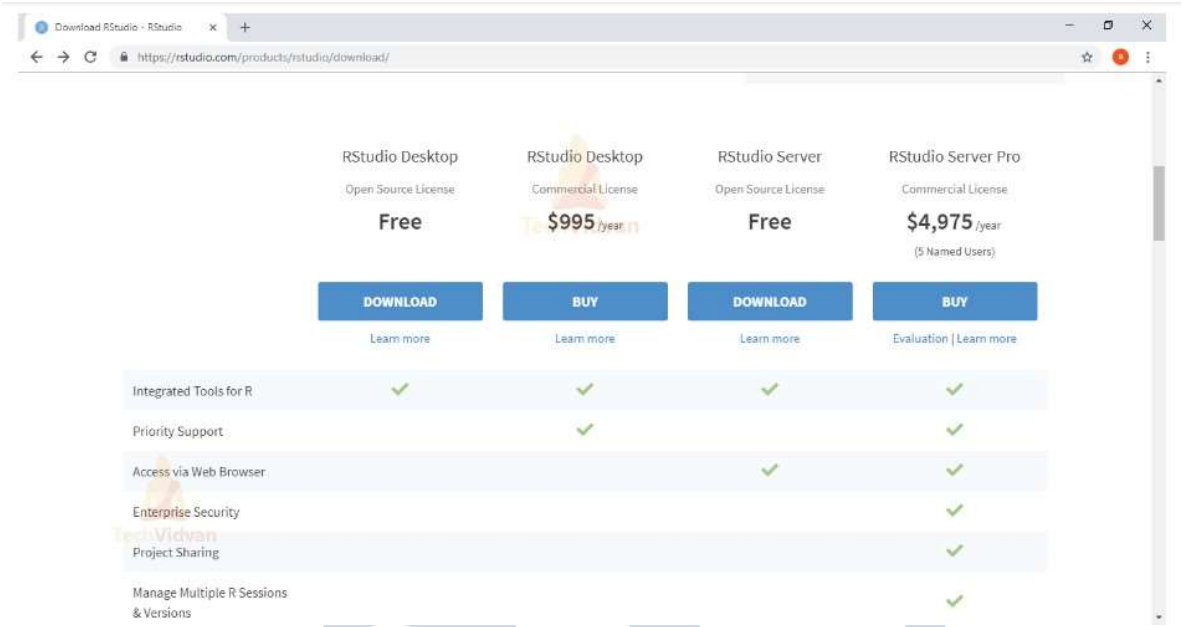


Step 5g: Click on **Finish** to complete the installation.



Install RStudio on Windows

Step 1: With R-base installed, let's move on to installing RStudio. To begin, go to download RStudio and click on the download button for **RStudio desktop**



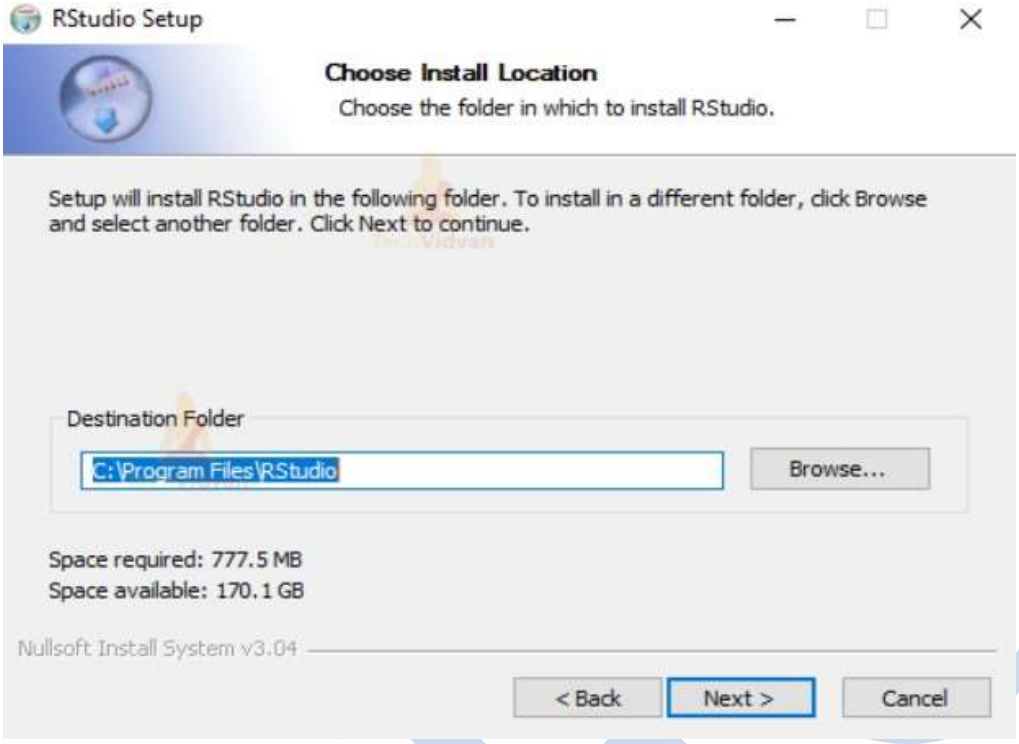
Step 2: Click on the link for the windows version of RStudio and save the .exe file.

Step 3: Run the .exe and follow the installation instructions.

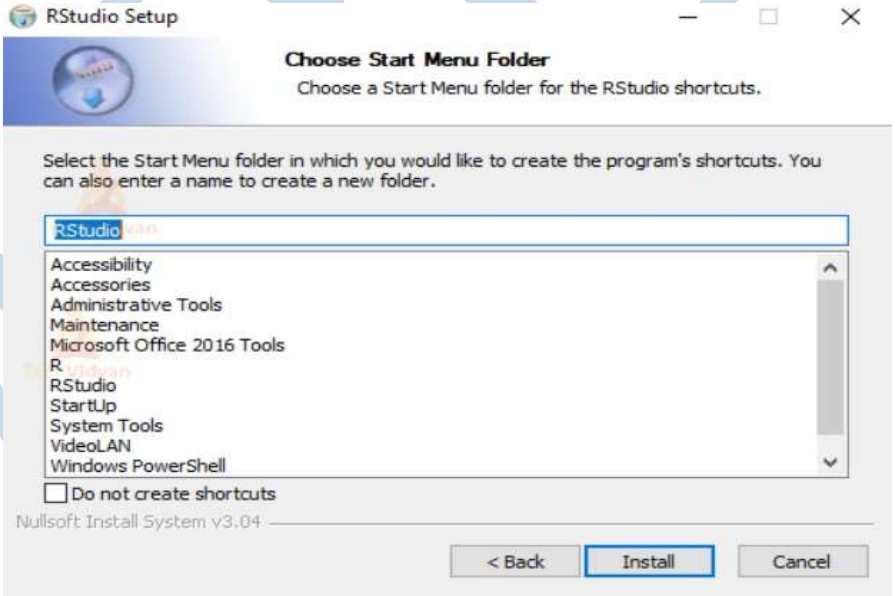
Step 3a: Click **Next** on the welcome window.



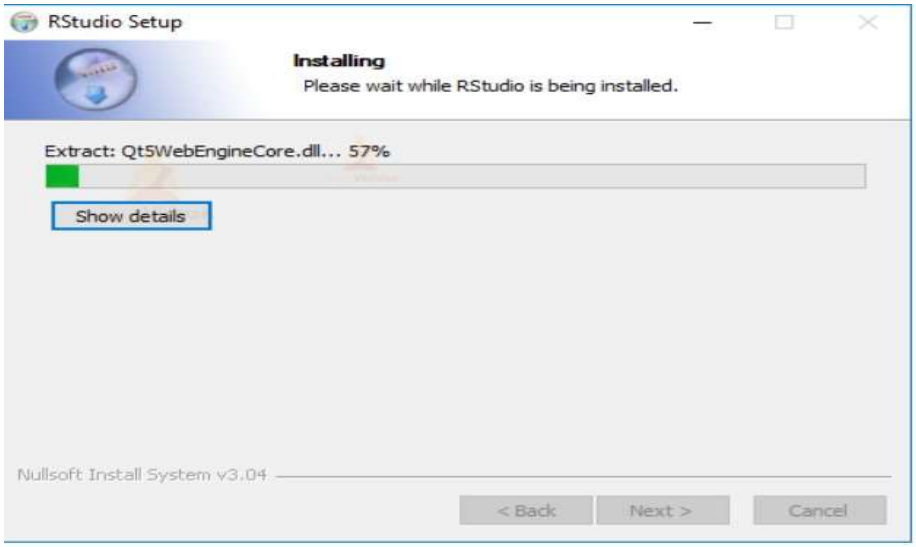
Step 3b: Enter/browse the path to the installation folder and click **Next** to proceed.



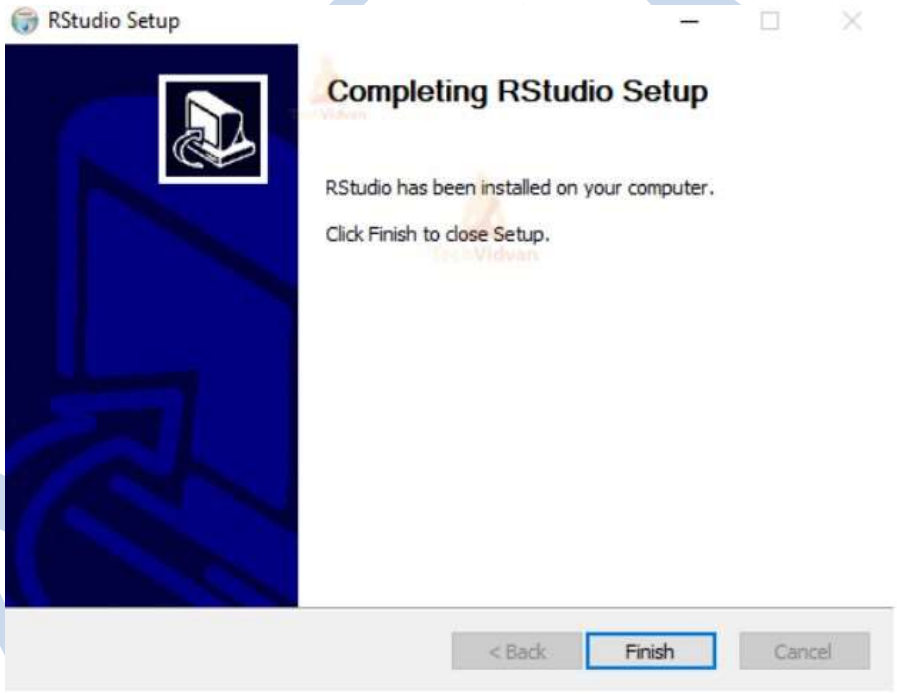
Step 3c: Select the folder for the start menu shortcut or click on do not create shortcuts and then click **Next**



Step 3d: Wait for the installation process to complete.



Step 3e: Click **Finish** to end the installation.



2. Install the data Analysis and Visualization tool: Python

Step 1: Downloading the Python Installer

1. Go to the official Python download page for Windows.
2. Find a stable Python 3 release. This tutorial was tested with Python version 3.10.10.
3. Click the appropriate link for your system to download the executable file: **Windows installer (64-bit)** or **Windows installer (32-bit)**



Step 2: Running the Executable Installer

1. After the installer is downloaded, double-click the .exe file, for example python-3.10.10-amd64.exe, to run the Python installer.
2. Select the **Install launcher for all users** checkbox, which enables all users of the computer to access the Python launcher application.
3. Select the **Add python.exe to PATH** checkbox, which enables users to launch Python from the command line.

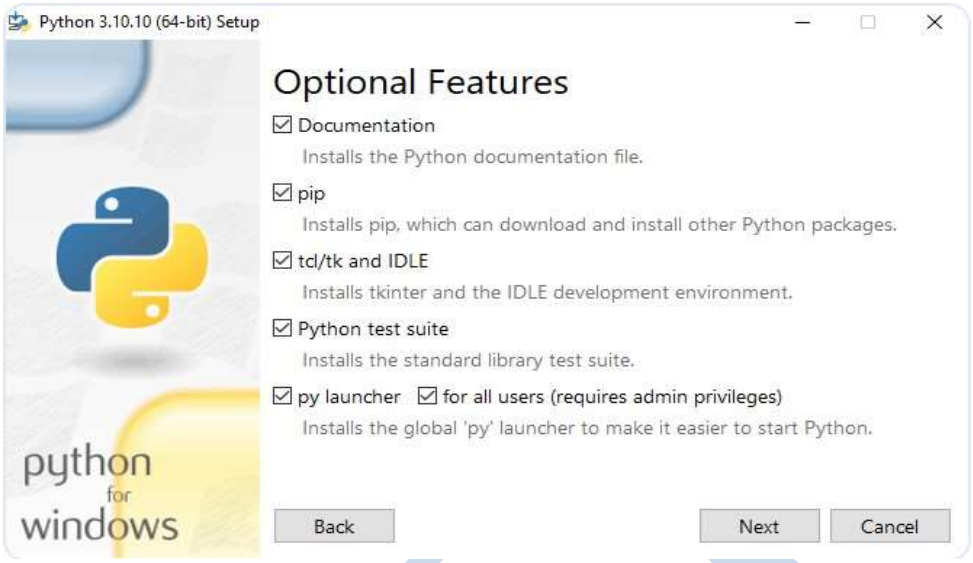


4. If you're just getting started with Python and you want to install it with default features as described in the dialog, then click **Install Now** and go to Step 4 - Verify the Python Installation. To install

other optional and advanced features, click **Customize installation** and continue.

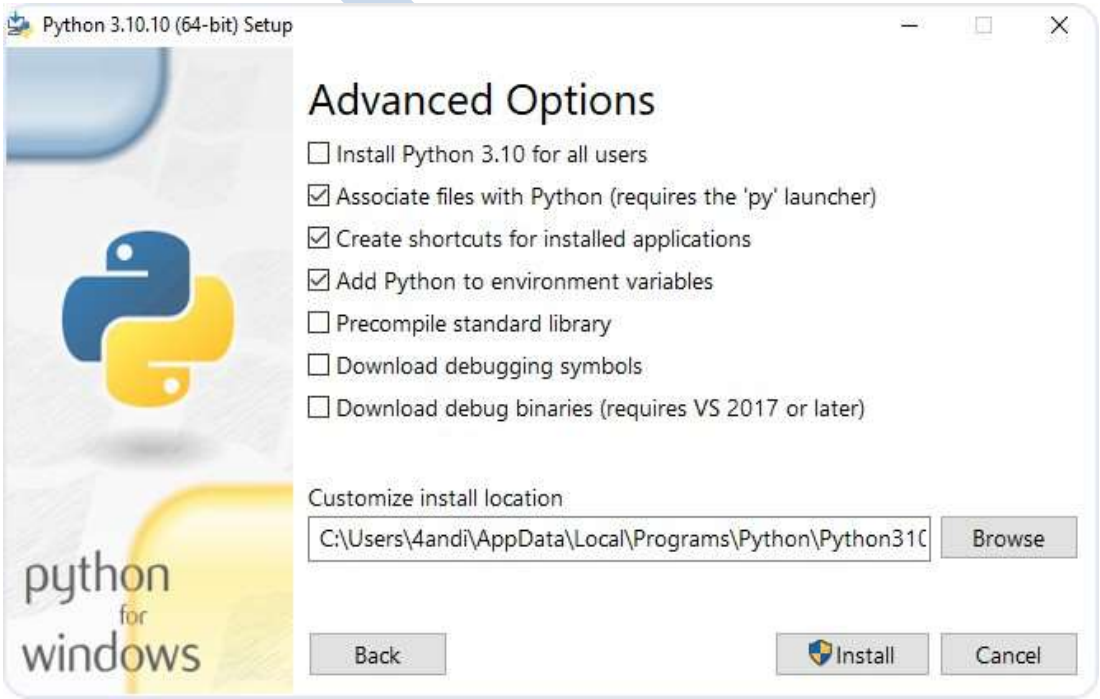
APEC

5. The **Optional Features** include common tools and resources for Python and you can install all of them, even if you don't plan to use them.



6. Select some or all of the following options:

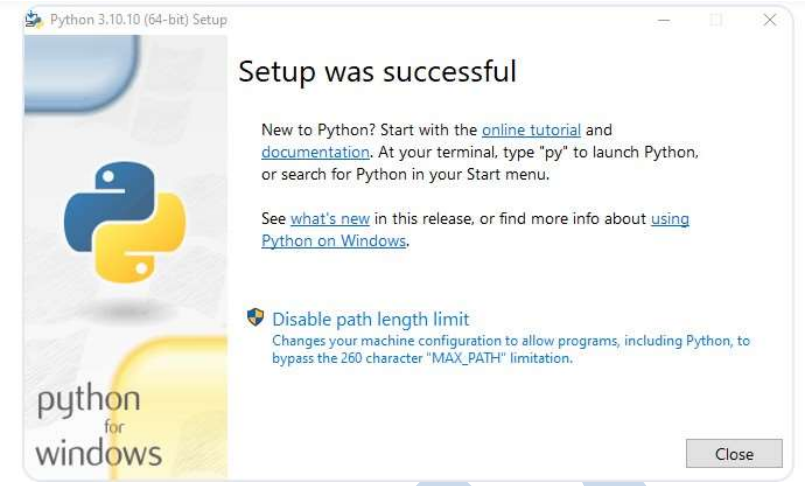
    1. **Documentation**: recommended
    2. **pip**: recommended if you want to install other Python packages, such as NumPy or pandas
    3. **tcl/tk and IDLE**: recommended if you plan to use IDLE or follow tutorials that use it
    4. **Python test suite**: recommended for testing and learning
    5. **py launcher** and **for all users**: recommended to enable users to launch Python from the command line
7. Click **Next**.
8. The **Advanced Options** dialog displays.



9. Click **Install** to start the installation.

APEC

10. After the installation is complete, a **Setup was successful** message displays.



Step 3: Adding Python to the Environment Variables (optional)

1. Go to **Start** and enter advanced system settings in the search bar.
2. Click **View advanced system settings**.
3. In the **System Properties** dialog, click the **Advanced** tab and then click **Environment Variables**.
4. Depending on your installation:
    - If you selected **Install for all users** during installation, select **Path** from the list of **System Variables** and click **Edit**.
    - If you didn't select **Install for all users** during installation, select **Path** from the list of **User Variables** and click **Edit**.
5. Click **New** and enter the Python directory path, then click **OK** until all the dialogs are closed.

Step 4: Verify the Python Installation

Go to **Start** and enter cmd in the search bar. Click **Command Prompt**.

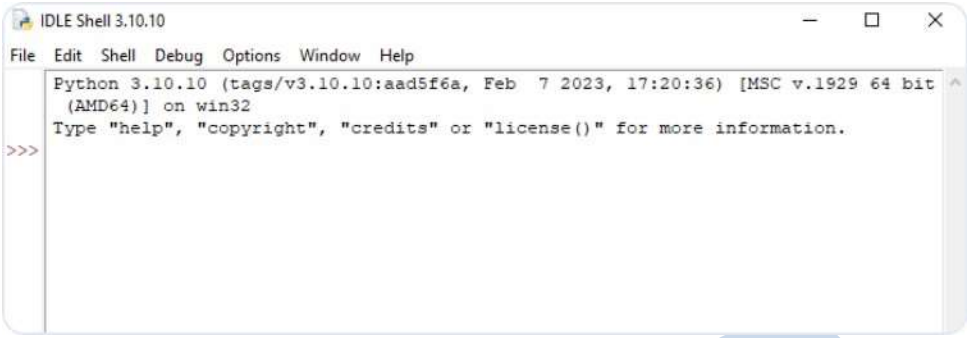
Enter the following command in the command prompt:

```
python --version
```

An example of the output is:

```
Output
Python 3.10.10
```

You can also check the version of Python by opening the IDLE application. Go to **Start** and enter python in the search bar and then click the IDLE app, for example **IDLE (Python 3.10 64-bit)**.



Step 5: You can start coding in Python using IDLE or your preferred code editor.

3.Install the data Analysis and Visualization tool: Tableau Public

About Tableau

Tableau can also refer to an analytics platform thatmakes it easier for people to explore and manage data. Tableau can extract data from:

- Excel
- Complex oracle databases
- PDF
- Google Cloud SQL
- Amazon Web Services

USES OF SOFTWARE TOOLS:

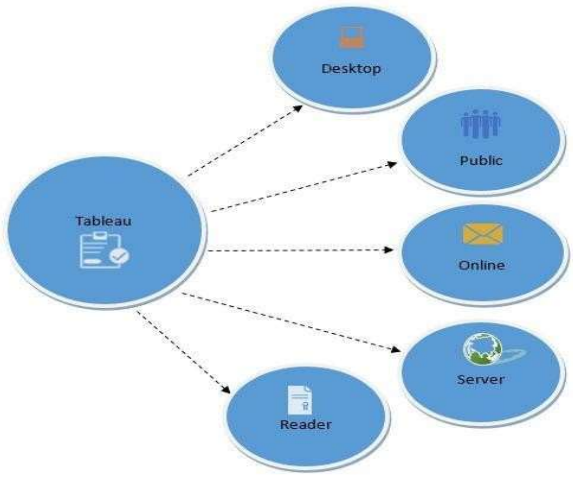
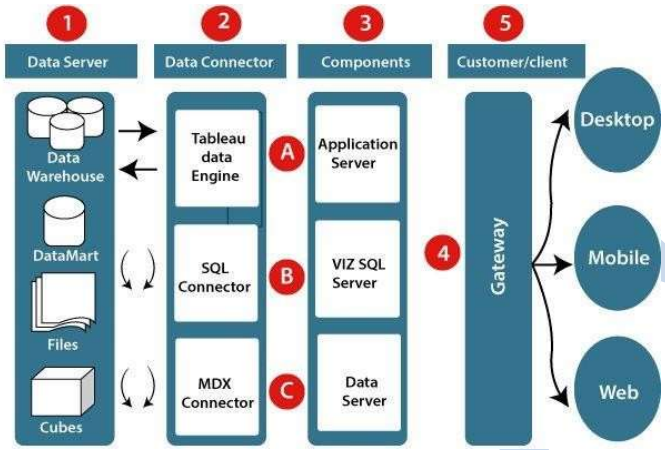


Tableau Architecture :



The different component of the Tableau architecture:

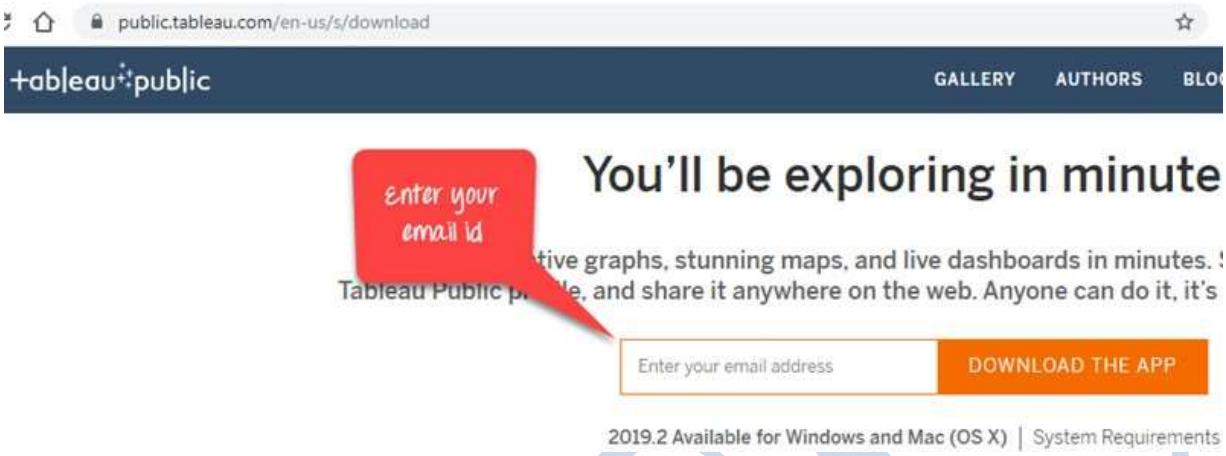
1. **Data server:-** The primary component of Tableau Architecture is the Data sources which can connect to it.

2. **Data connector:-** The Data Connectors provide an interface to connect external data sources with the Tableau Data Server.

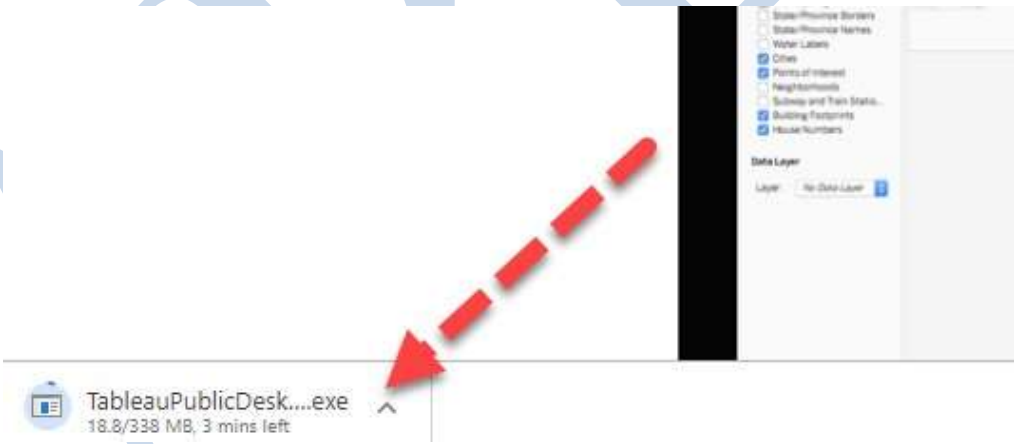
3. **Components of Tableau server:** Different types of component of the Tableau server are:
   - Application server
   - VizQL server
   - Data server
4. **Gateway:** The gateway directed the requests from users to Tableau components. When the client sends a request, it is forwarded to the external load balancer for processing. The gateway works as a distributor of processes to different components. In case of absence of external load balancer, the gateway also works as a load balancer. For single server configuration, one gateway or primary server manages all the processes. For multiple server configurations, one physical system works as a primary server, and others are used as worker servers. Only one machine is used as a primary server in Tableau Server environment.
5. **Clients:** The visualizations and dashboards in Tableau server can be edited and viewed using different clients. Clients are a web browser, mobile applications, and Tableau Desktop.
   - **Web Browser:** Web browsers like Google Chrome, Safari, and Firefox support the Tableau server. The visualization and contents in the dashboard can be edited by using these web browser.
   - **Mobile Application:** The dashboard from the server can be interactively visualized using mobile application and browser. It is used to edit and view the contents in the workbook.
   - **Tableau Desktop:** Tableau desktop is a business analytics tool. It is used

   to view, create, and publish the dashboard in Tableau server. Users can access the various data source and build visualization in Tableau desktop.

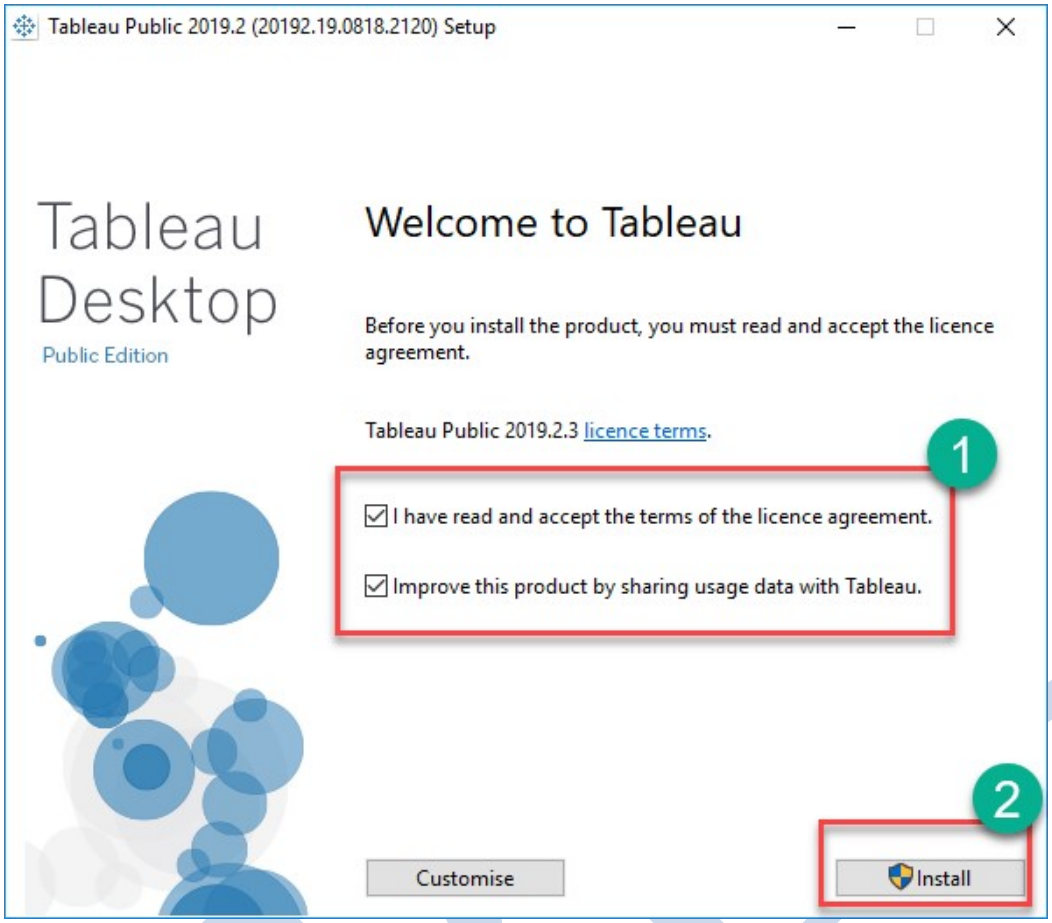
Step 1: Go to https://public.tableau.com/en-us/s/download on your web browser. Now you need to enter your email id and click on "**DOWNLOAD THE APP**" button.



Step 2: This will start downloading the .exe file for Windows by default, and you can see the downloading process in the bottom left corner of the website.



Step 3: Open the downloaded file. Accept the terms and conditions and click on "Install" button.



Step 4: After installation Start Screen of Tableau is shown

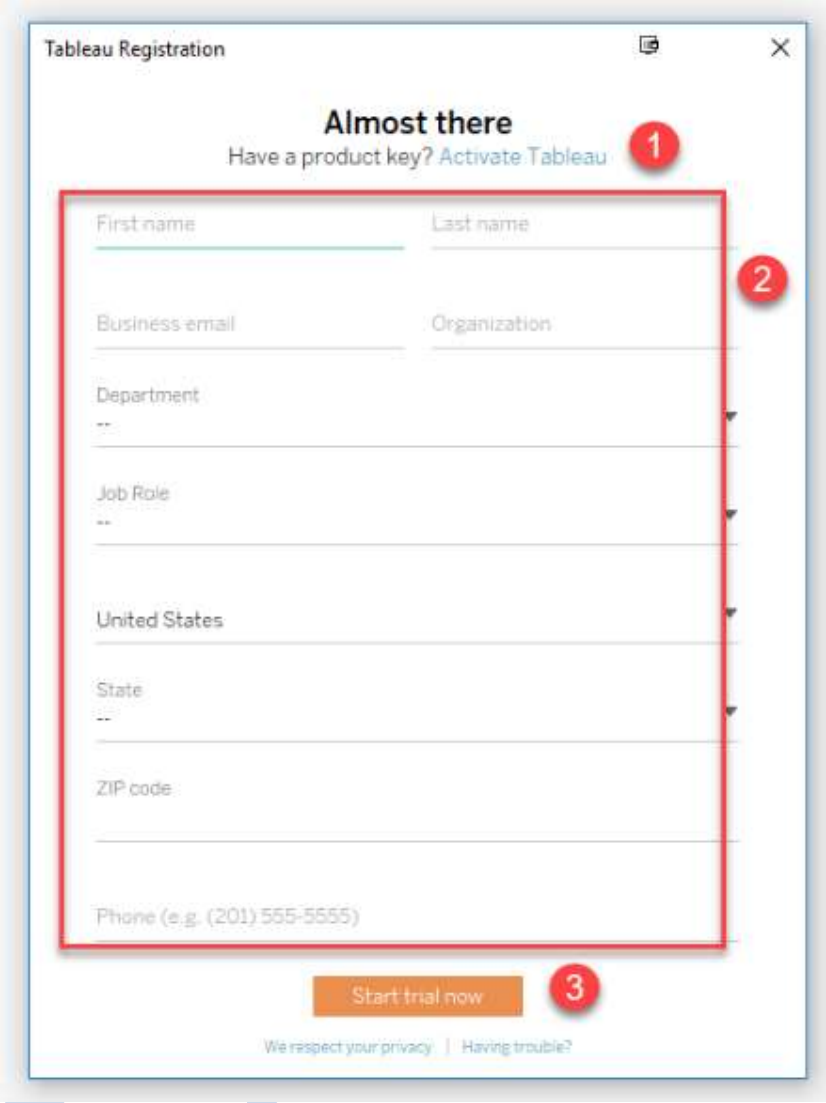


Step 5: Once the installation is completed, open the Tableau Desktop software

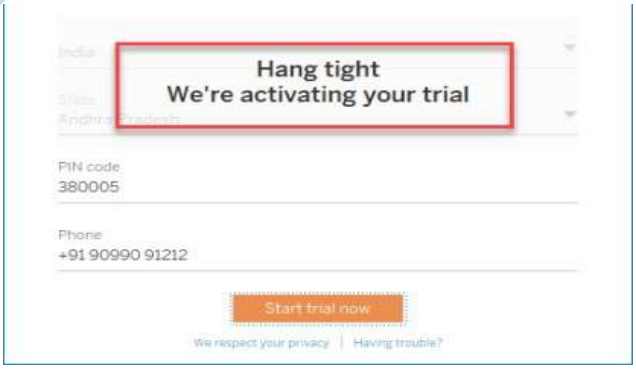
APEC

Step 6: In the Registration window

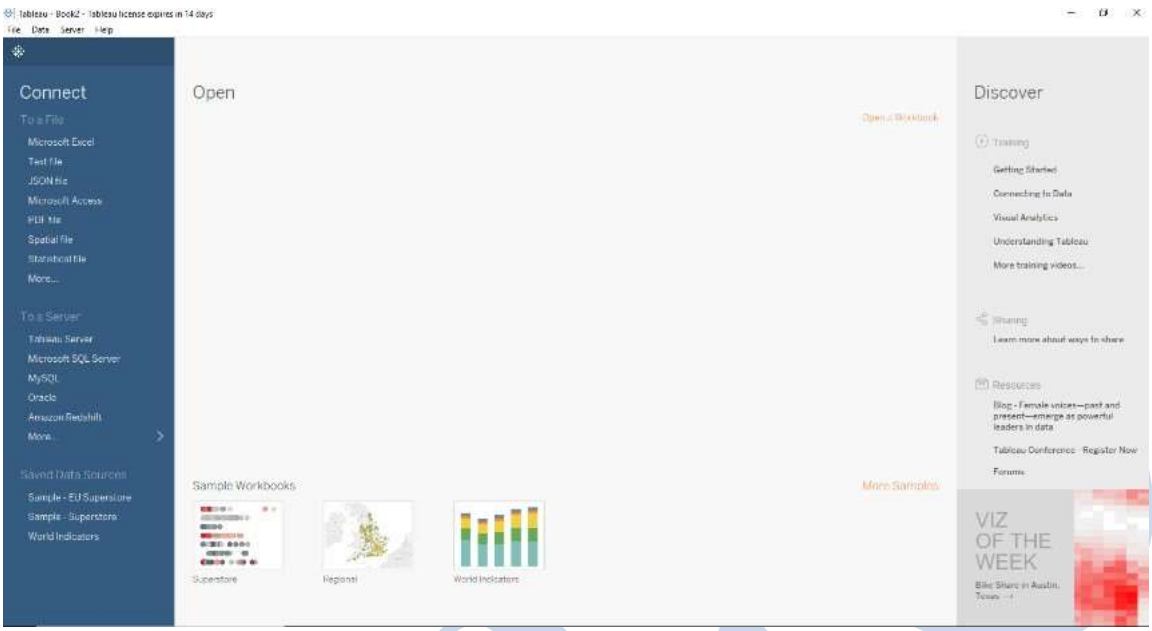
1. Click on Activate Tableau and entire your license details
2. If you do not have a license, enter your credentials
3. Click on Start Trial now



Step 7: Wait for registration to complete



Step 8: Start Screen of Tableau is shown



4. **Install the data Analysis and Visualization tool: Power BI**

About Power BI

1. Power BI is one of the most popular Data Visualization and BusinessIntelligence tool
2. The Power BI tool is the collection of apps, data connectors,and software services which helps to convert data from different data sources into interactive dashboards and BI reports

Power BI Components

1.P ower Query: Power Query is a business intelligence tool available in Excel that allows youto import data from many different sources and then clean, transform and reshape your data as needed.

2. **Power Pivot:** used to model your data and perform more complex calculations than Excelcan handle.

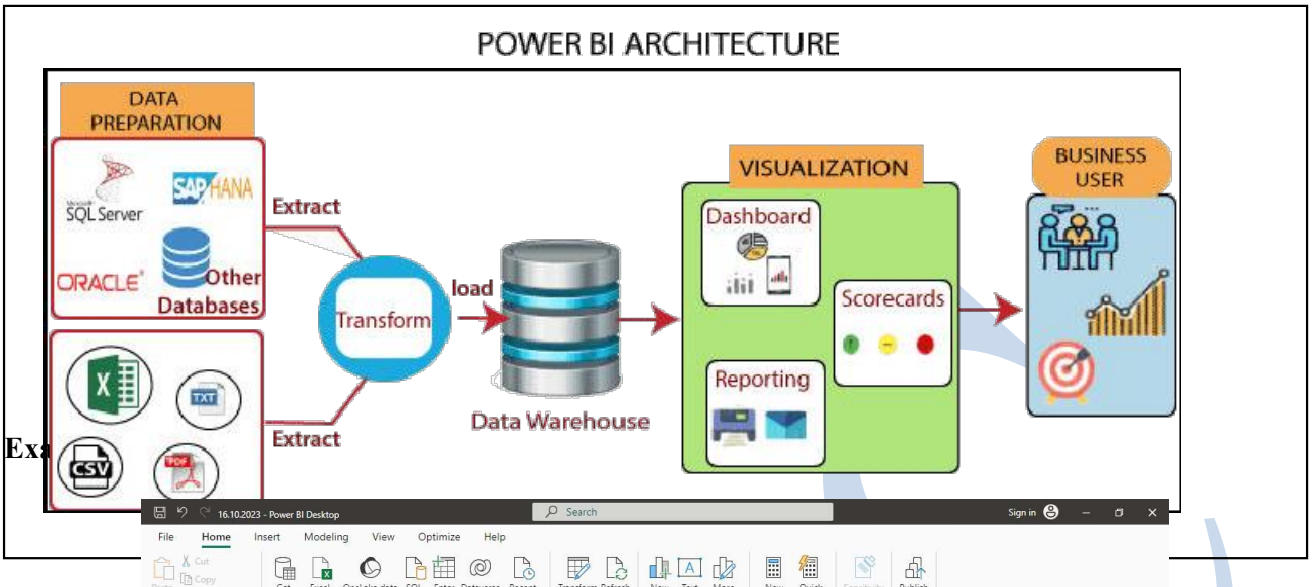
3. **Power View:** data visualization technology that lets you create interactive charts, graphs,maps, and other visuals that bring your data to life

4. **Power Map:** It brings the data to life with interactive geographical visualization.
5. **Power BI Service:** Power BI service is a cloud-based business analytics and data visualization service that enables anyone to visualize and analyze data with greater speed, efficiency, and understanding.
6. **Power BI Q&A:** You can ask any questions and get an immediate response with the natural language query.

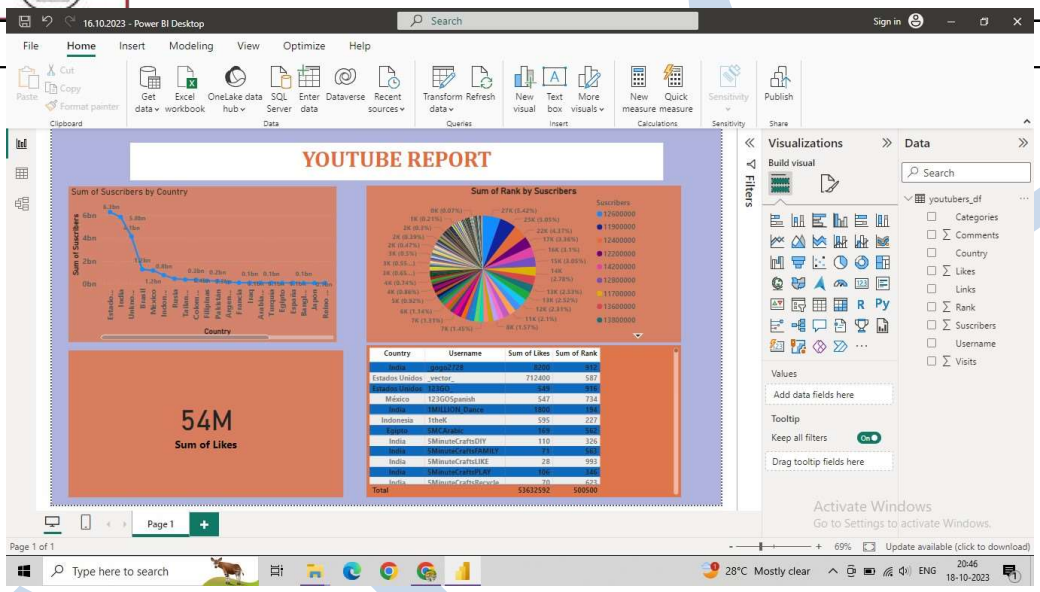
7. **Data Management Gateway:** The gateway is responsible for creating the connection and passing data through.

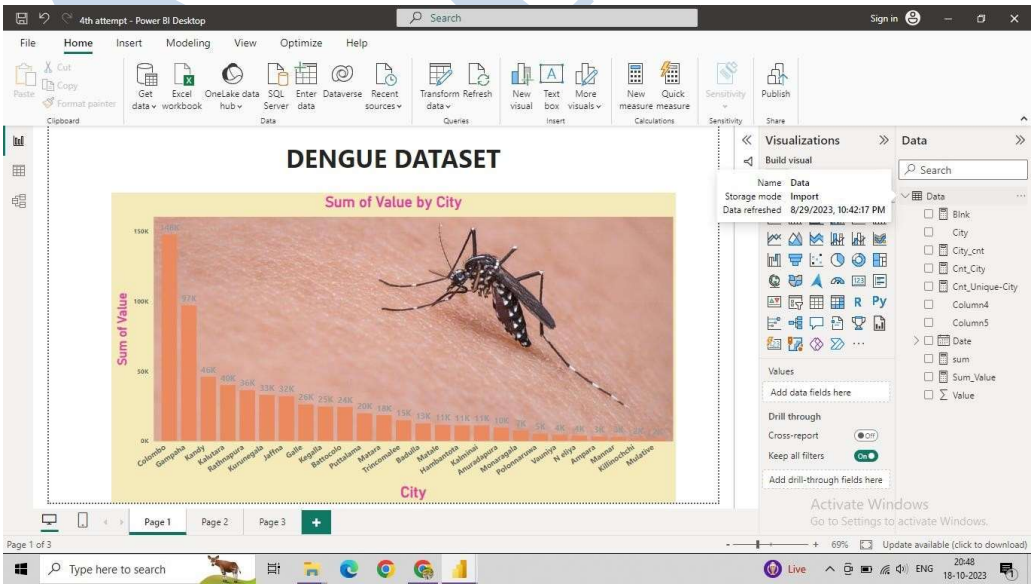
8. **Data Catalog:** By using the data catalog, you can quickly discover and reuse the queries.

APEC


POWER BI ARCHITECTURE
DATA PREPARATION
SQL Server
SAP HANA
ORACLE
Other Databases
Extract
Transform
load
Data Warehouse
Extract
VISUALIZATION
Dashboard
Scorecards
Reporting
BUSINESS USER

Ex

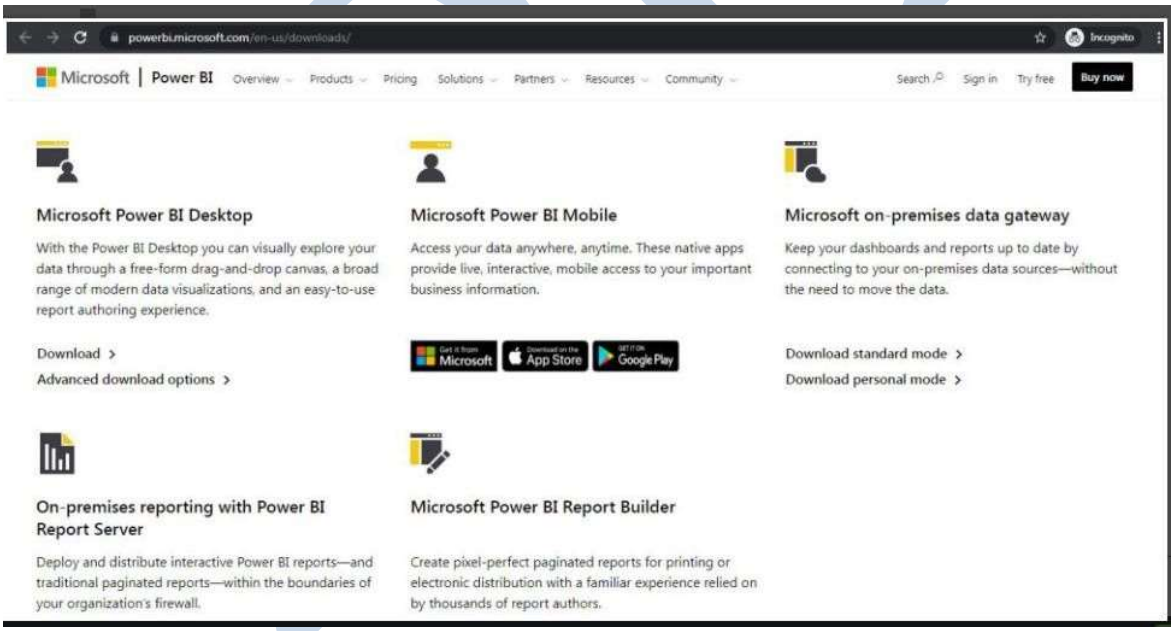

YOUTUBE REPORT
Sum of Suscribers by Country
Sum of Rank by Suscribers
54M
Sum of Likes
Visualizations
Data


DENGUE DATASET
Sum of Value by City
Sum of Value
City
Visualizations
Data

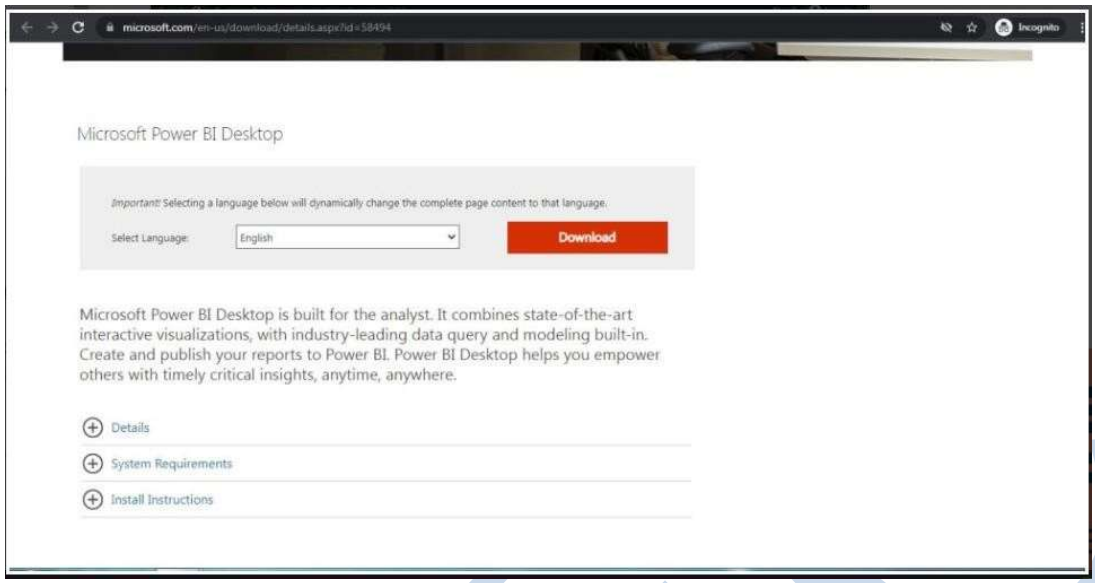
Step1: Visit the official website on any web browser.



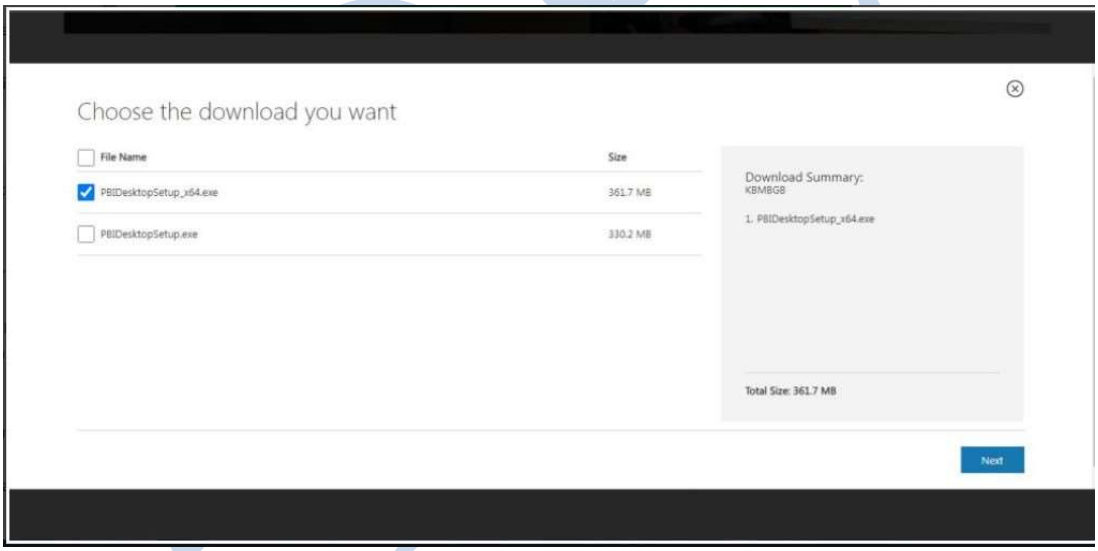
Step 2: Click on Microsoft Power BI desktop application.



Step 3: Next webpage will open, click on Download Button.



Step 4: On the next webpage choose the setup option according to your system configuration, let's take the first setup, click on the Next button. Downloading of the executable file will start shortly. It is a big 361.7 MB file that will take some minutes.



Step 5: Now check for the executable file in downloads in your system and run it.



Step 6: It will prompt confirmation to run the file on your system. Click on Run.



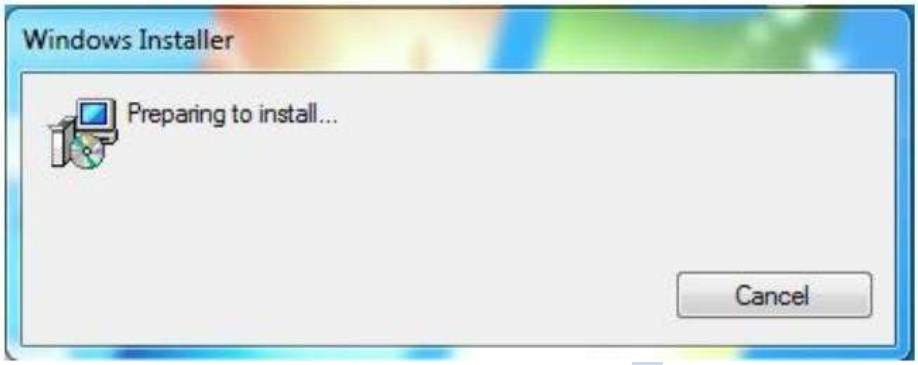
Step 7: Next screen will appear, select language, and click on Next.



Step 8: It will prompt confirmation to make changes to your system. Click on yes.



Step 9: The system will prepare for installation



Step 10: Setup screen will appear, click on Next.



Step 11: The next screen will be of License Agreement, check the box, I accept the terms in License Agreement and click on Next.



Step 12: Choose the destination folder and click on Next.



Step 13: The last screen will appear to click on the Install button.



Step 14: After this installation process will start and will hardly take a minute to complete the installation.



Step 15: Click on Finish to finish the installation process.



Step 16: Power BI is successfully installed on the system and an icon is created on the desktop.



Step 17: Run the software and see the interface



Result :

Thus the install the data Analysis and Visualization tool: R/ Python /Tableau Public/ Power BI was verified and installed Successfully

APEC

Exp No : 2 Perform Exploratory Data Analysis (EDA) on with datasets like email dataset.

Date: **Export all your emails as a dataset, import them inside a pandas DataFrame, visualize them and get different insights from the data.**

Aim :

Perform exploratory data analysis (EDA) on with datasets like email data set. Export all your emails as a dataset, import them inside a pandas data frame, visualize them and get different insights from the data.

Algorithm :

1. Export Email Data: Export your emails from your email client (e.g., Outlook, Gmail) and save the exported data as a file (e.g., emails.csv, emails.xlsx).
2. Import Required Libraries: In Python, import the necessary libraries, including pandas, matplotlib, and seaborn, for data manipulation, visualization, and insights.
3. Load Email Data: Use pandas to read the exported email dataset file into a pandas DataFrame.
4. Perform EDA: Analyze and explore the email dataset to gain insights. Here are some examples of EDA techniques you can apply.

Program:

```
import pandas as pd
import numpy as np

# for visualization
import matplotlib as mpl
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import plotly.express as px
import plotly.figure_factory as ff
from plotly.subplots import make_subplots
import plotly.graph_objs as go
from wordcloud import WordCloud
```

```
df = pd.read_csv('../input/spam-email/spam.csv')
msno.matrix(df).set_title('Distribution of missing values',fontsize=20)
```

Output :



```
print(df.shape)
df.head(3)
```

Output :

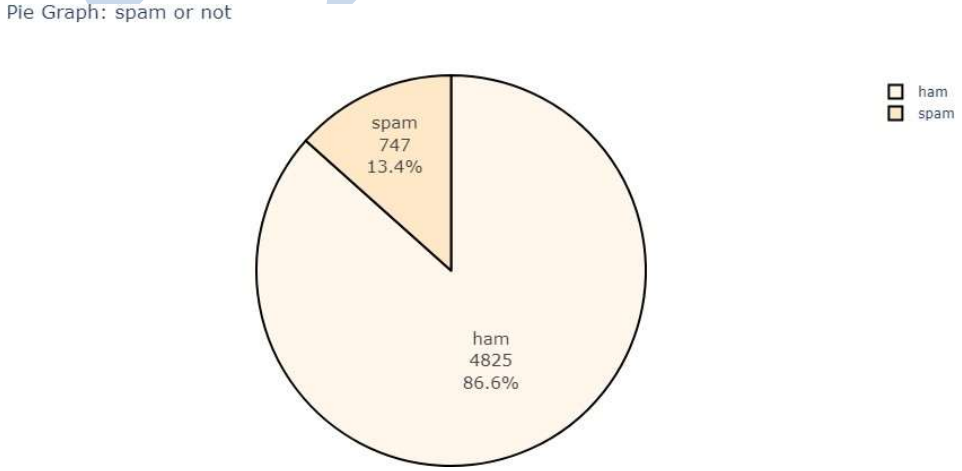
(5572, 2)

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...

```
category_ct = df['Category'].value_counts()

fig = px.pie(values=category_ct.values,
             names=category_ct.index,
             color_discrete_sequence=px.colors.sequential.OrRd,
             title= 'Pie Graph: spam or not')
fig.update_traces(hoverinfo='label+percent', textinfo='label+value+percent', textfont_size=15,
                  marker=dict(line=dict(color='#000000', width=2)))
fig.show()
```

Output :



```
categories = pd.get_dummies(df["Category"])
spam_or_not = pd.concat([df, categories], axis=1)
spam_or_not.drop('Category',axis=1,inplace=True)

df["length"] = df["Message"].apply(len)

ham = df.loc[np.where(spam_or_not['ham'] == 1)].reset_index()
spam = df.loc[np.where(spam_or_not['ham'] == 0)].reset_index()

ham.drop('index',axis=1,inplace=True)
spam.drop('index',axis=1,inplace=True)


hist_data = [ham['length'],spam['length']]

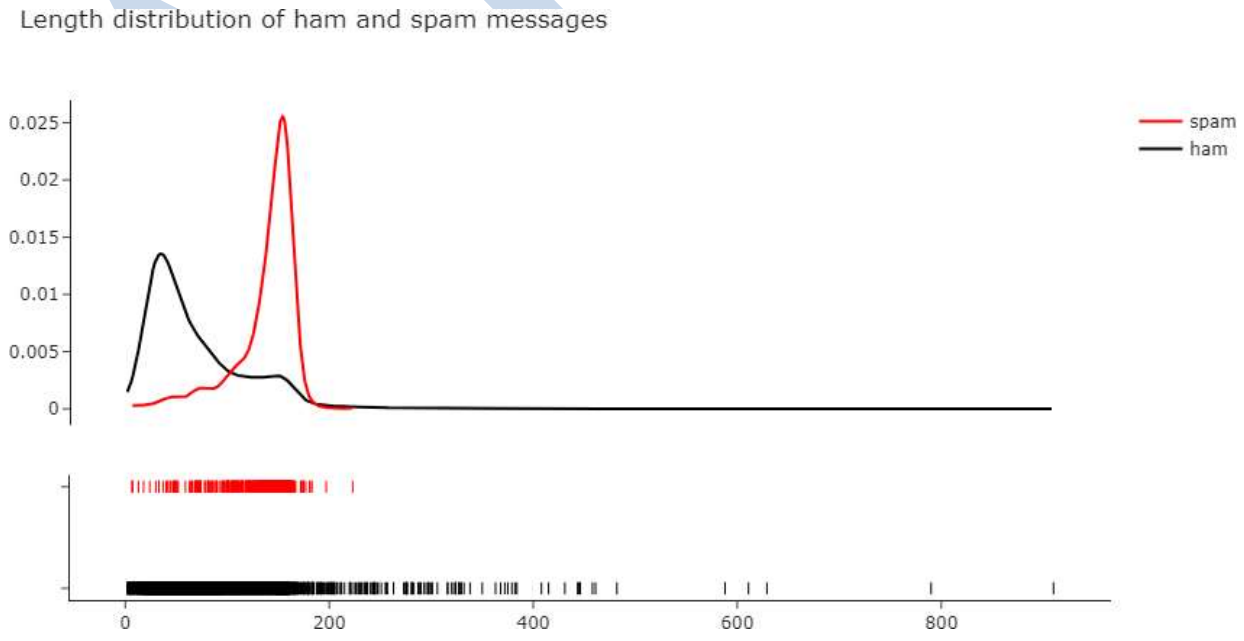
group_labels = ['ham','spam']

colors = ['black', 'red']

# Create distplot with curve_type set to 'normal'
fig = ff.create_distplot(hist_data, group_labels, show_hist=False, colors=colors)

# Add title
fig.update_layout(title_text='Length distribution of ham and spam messages',
                  template = 'simple_white')
fig.show()
```

Output :



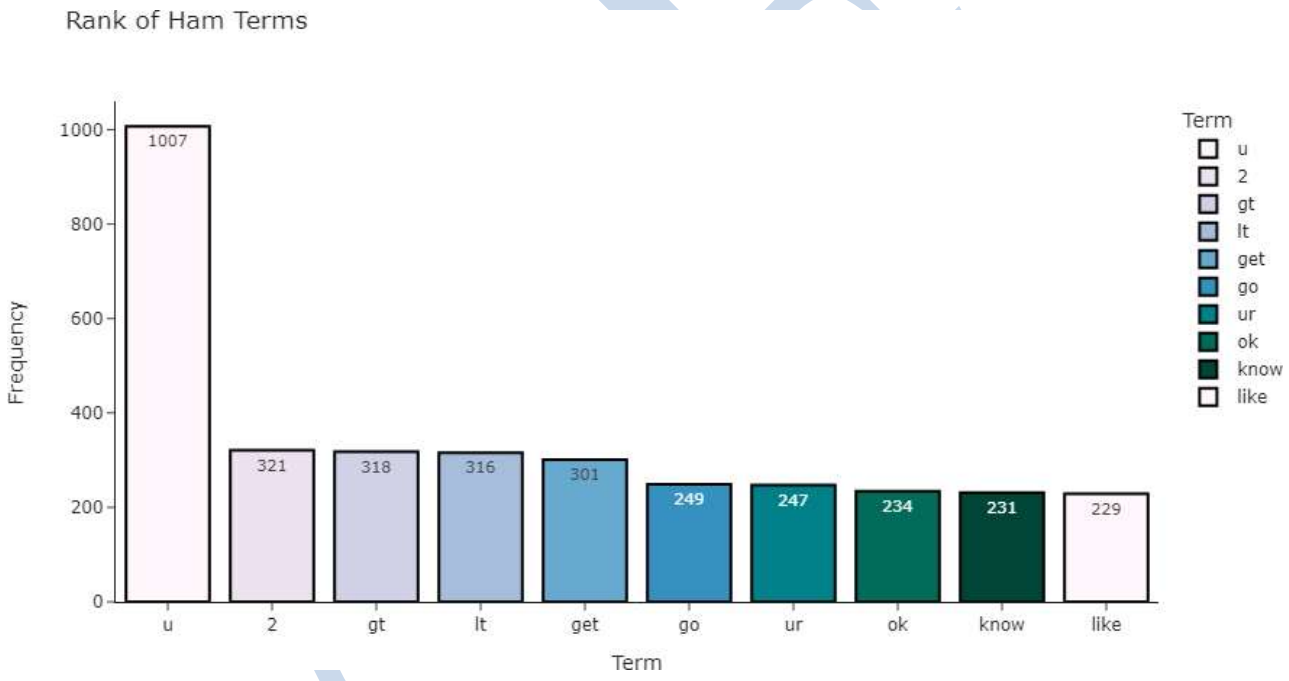
```
freq_df = Freq_df(removed)
top_10 = freq_df[:10]

fig = px.bar(top_10, x = 'Term', y = 'Frequency',text = 'Frequency', color='Term',
             color_discrete_sequence=px.colors.sequential.PuBuGn,
             title = 'Rank of Ham Terms',
             template = "simple_white"
              )

for idx in range(len(top_10)):
    fig.data[idx].marker.line.width = 2
    fig.data[idx].marker.line.color = "black"

fig.update_traces(textposition='inside',
                  textfont_size=11)
fig.show()
```

Output :



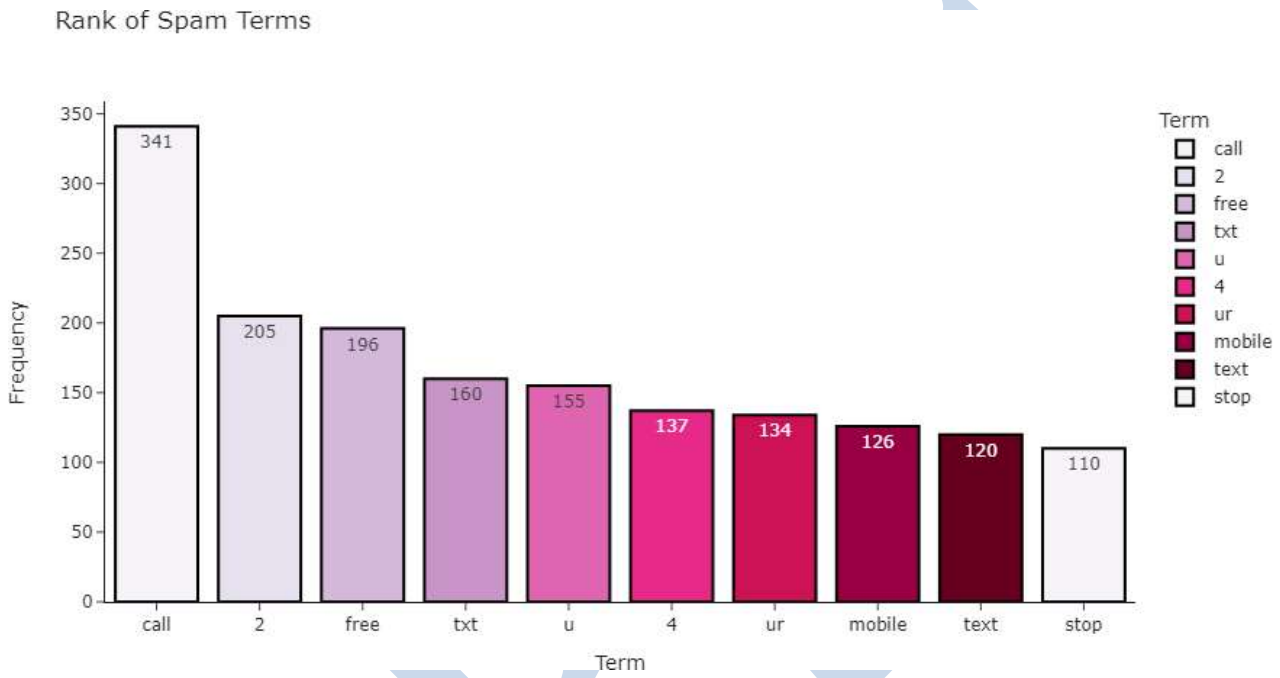

```
string = get_all_str(spam)
words = get_word(string)
removed = remove_stopword('1',words)
freq_df = Freq_df(removed)
top_10 = freq_df[:10]

fig = px.bar(top_10, x = 'Term', y = 'Frequency',text = 'Frequency',
             color_discrete_sequence=px.colors.sequential.PuRd,
             title = 'Rank of Spam Terms',
             template = "simple_white",
             color='Term')
```

```
for idx in range(len(top_10)):
    fig.data[idx].marker.line.width = 2
    fig.data[idx].marker.line.color = "black"

fig.show()
```

Output :



Result :

Thus the exploratory data analysis (EDA) on with datasets like email data set. Export all your emails as a dataset, import them inside a pandas data frame, visualize them and get different insights from the data

CCS346 - EDA
was executed and verified successfully.

APEC

Exp No : 3 **Working with Numpy Arrays, Pandas Dataframe,**

Data: **Basic Plots using Matplotlib**

Aim :

To write a python program for numpy arrays,pandas dataframe and basic plots using matplotlib

Algorithm :

Step 1 : Import the python libraries.

Step 2 : Create a numpy arrays and pandas dataframe.

Step 3 : Perform some basic operations on that arrays like indexing,slicing.

Step 4 : Plot the graph for the created arrays and dataframe using matplotlib.

Program :

```
CREATING NUMPY ARRAYS
```

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
```

```
[1 2 3 4 5]
```

```
import numpy as np
arr = np.array((1, 2, 3, 4, 5))
print(arr)
```

```
[1 2 3 4 5]
```

```
import numpy as np
arr = np.array(42)
print(arr)
```

```
42
```

```
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)
```

```
[[1 2 3]
 [4 5 6]]
```

```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

```
0
1
2
3
```

NUMPY ARRAY INDEXING

```
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr[0])
```

```
1
```

```
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr[1])
```

```
2
```

```
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr[2] + arr[3])
```

```
7
```

```
import numpy as np
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('2nd element on 1st row: ', arr[0, 1])
```

```
2nd element on 1st row:  2
```

CREATING PANDAS SERIES

```
import pandas as pd
a = [1, 3, 5]
D = pd.Series(a)
print(D)
```

```
0    1
1    3
2    5
dtype: int64
```

```
import pandas as pd
a = [1, 3, 5]
D = pd.Series(a, index = ["x", "y", "z"])
print(D)
```

```
x    1
y    3
z    5
dtype: int64
```

```
print(D["y"])
```

```
3
```

```
import pandas as pd
calories = {"day1": 420, "day2": 380, "day3": 390}
P = pd.Series(calories)
print(P)
```

```
day1    420
day2    380
day3    390
dtype: int64
```

```
import pandas as pd
calories = {"day1": 420, "day2": 380, "day3": 390}
var = pd.Series(calories, index = ["day1", "day2"])
print(var)
```

```
day1    420
day2    380
dtype: int64
```

```
CREATING PANDAS DATAFRAME
```

```
import pandas as pd
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
df = pd.DataFrame(data)
print(df)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

```
PANDAS DATAFRAME INDEXING
```

```
print(df.loc[0])
```

```
calories    420
duration     50
Name: 0, dtype: int64
```

```
print(df.loc[[0, 1]])
```

```
   calories  duration
0       420        50
1       380        40
```

MATPLOTLIP

```
import matplotlib.pyplot as plt

import numpy as np

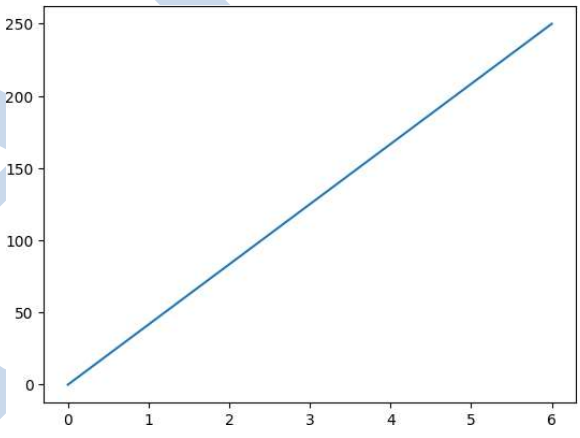
xpoints = np.array([0, 6])

ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)

plt.show()
```

Output :



Program:

```
import matplotlib.pyplot as plt

import numpy as np

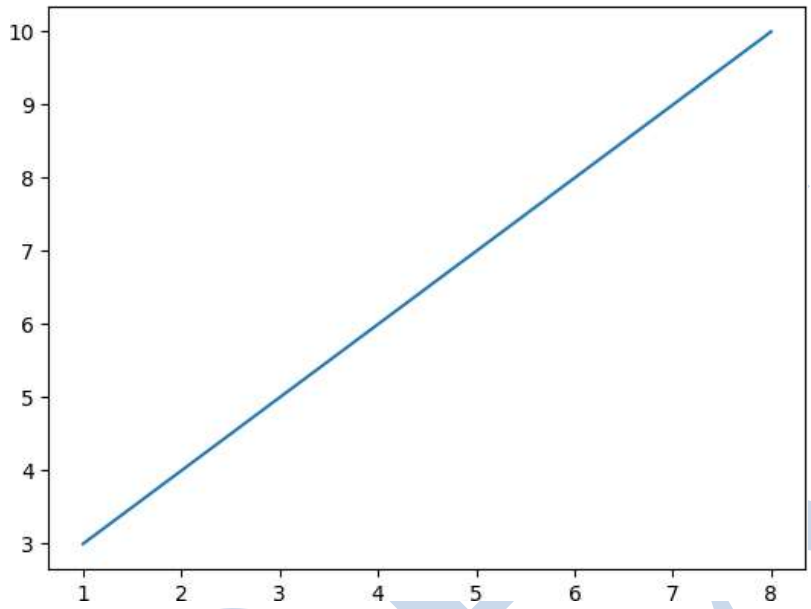
xpoints = np.array([1, 8])
```

```
ypoints = np.array([3, 10])
```

APEC

```
plt.plot(xpoints, ypoints)
plt.show()
```

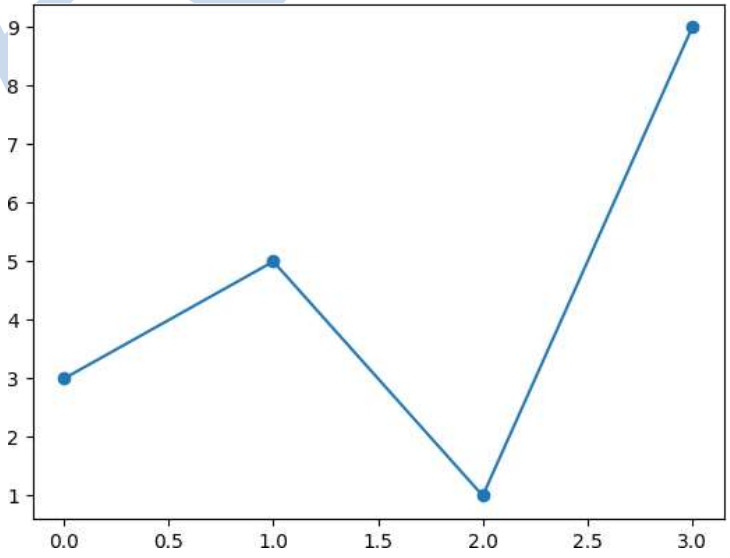
Output :



Program:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 5, 1, 9])
plt.plot(ypoints, marker = 'o')
plt.show()
```

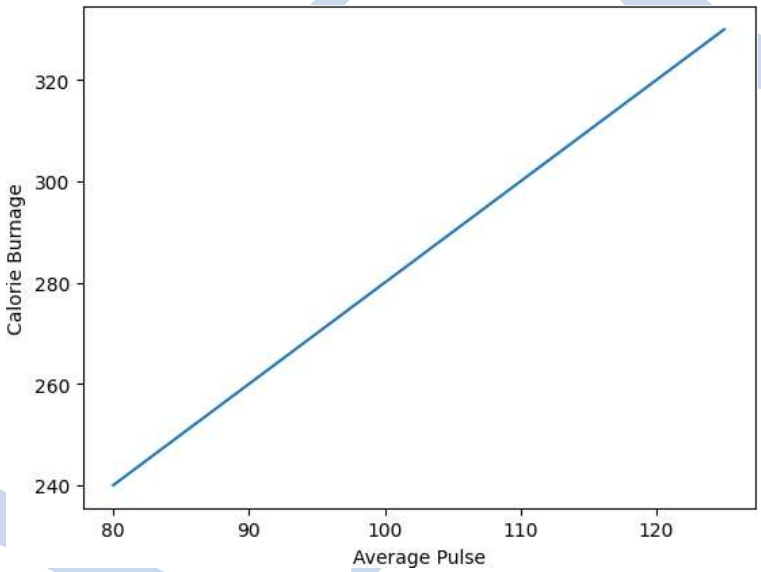
Output :



Program:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.plot(x, y)
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.show()
```

Output:

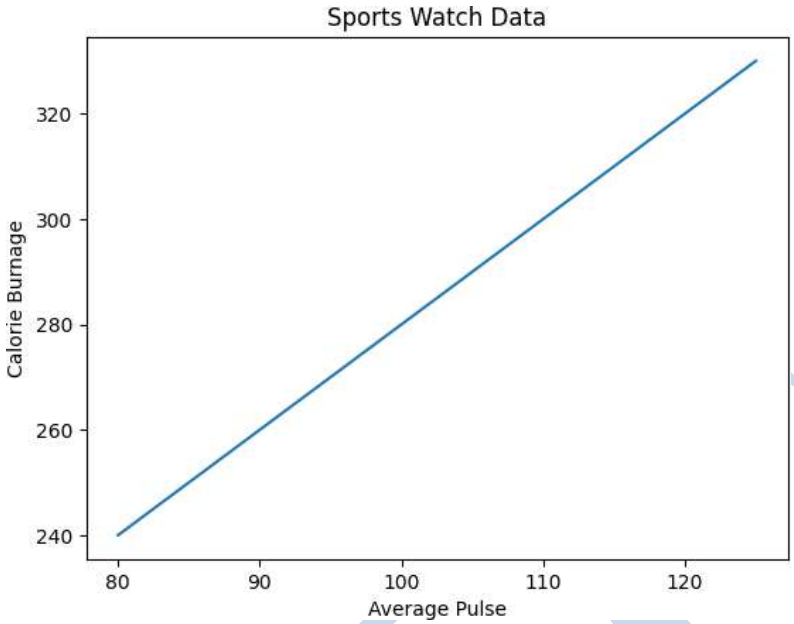


Program:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
y = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.plot(x, y)
plt.title("Sports Watch Data")
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.show()
```

Output :

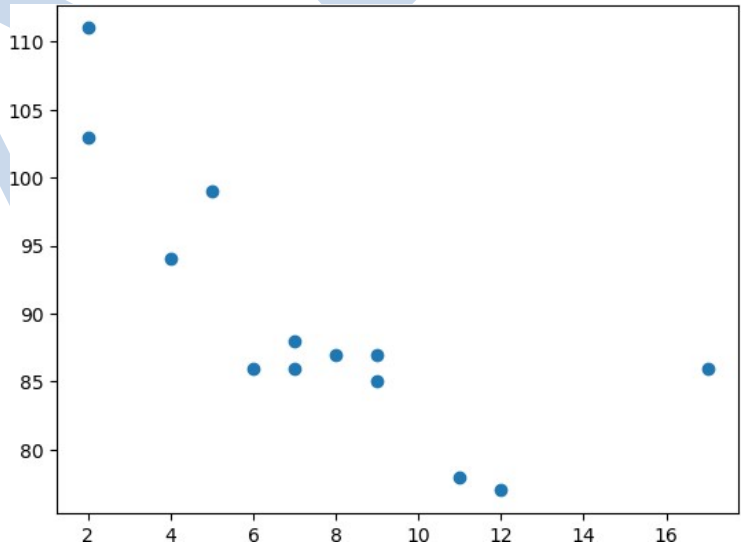
APEC



Program:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])
plt.scatter(x, y)
plt.show()
```

Output :



Program:

```
import matplotlib.pyplot as plt

import numpy as np
```

APEC

```
x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])

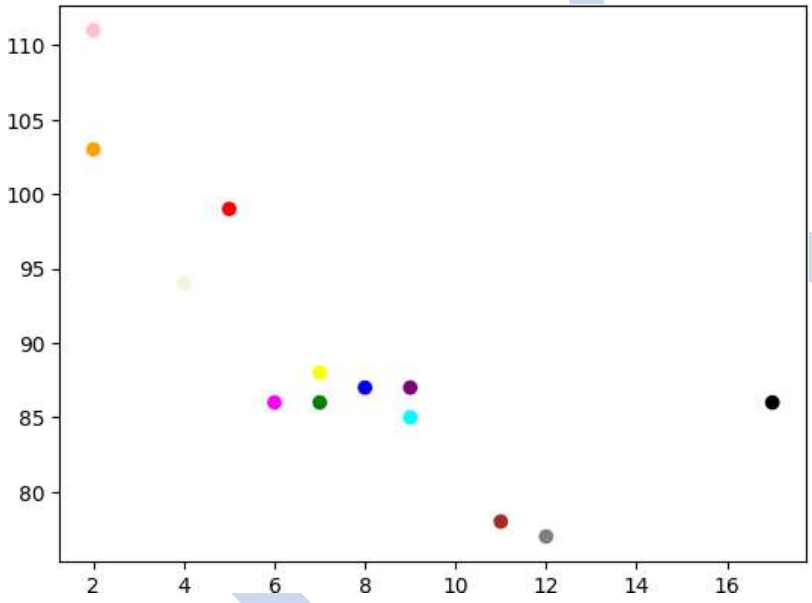
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

colors=np.array(["red","green","blue","yellow","pink","black","orange","purple","beige","brown","gray","cyan","magenta"])

plt.scatter(x, y, c=colors)

plt.show()
```

Output :



Program:

```
import matplotlib.pyplot as plt

import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])

y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

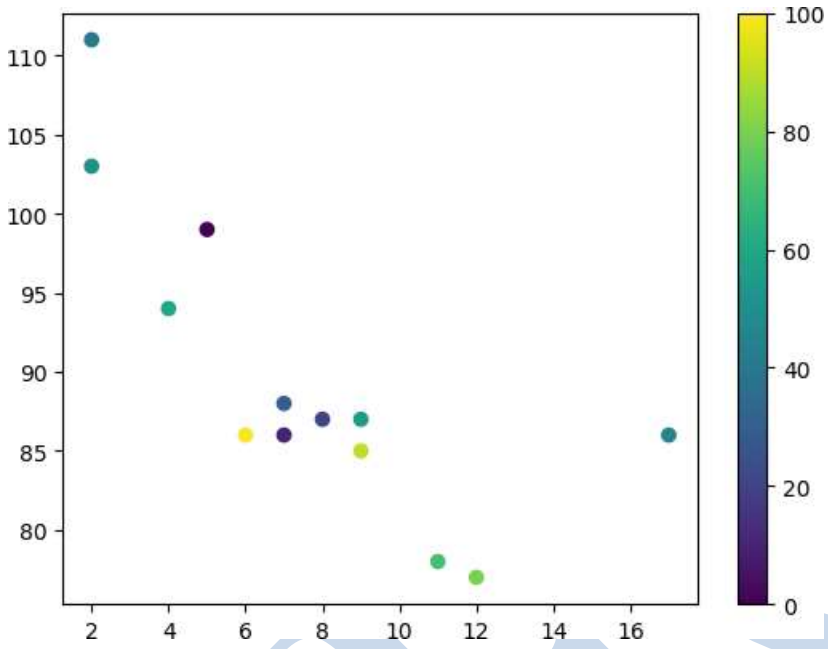
colors = np.array([0, 10, 20, 30, 40, 45, 50, 55, 60, 70, 80, 90, 100])

plt.scatter(x, y, c=colors, cmap='viridis')

plt.colorbar()

plt.show()
```

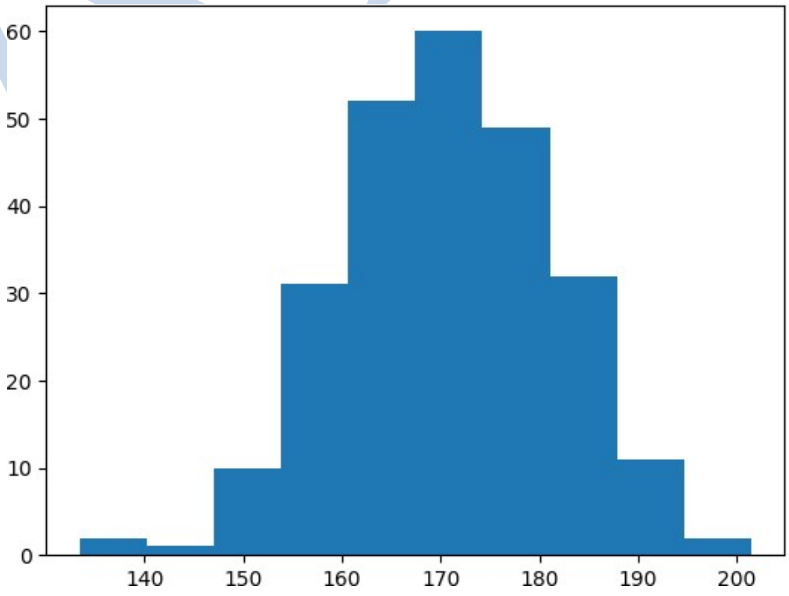
Output :



Program:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.normal(170, 10, 250)
plt.hist(x)
plt.show()
```

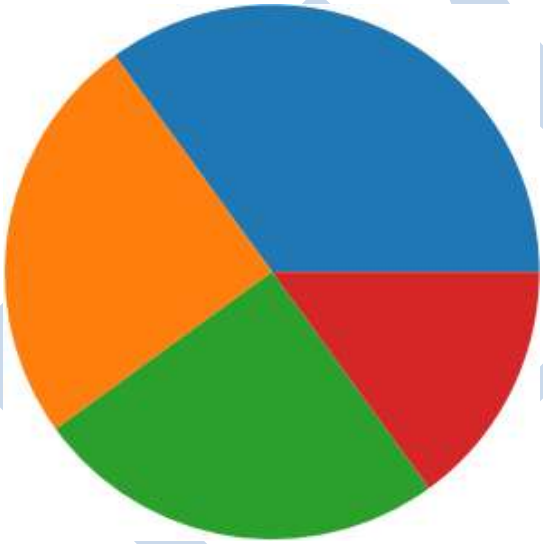
Output :



Program:

```
import matplotlib.pyplot as plt
import numpy as np
y = np.array([35, 25, 25, 15])
plt.pie(y)
plt.show()
```

Output :



Result :

AUCC

Thus the working with Numpy arrays, Pandas data frames ,Basic plots using Matplotlib was executed and verified successfully.

APEC

Exp No : 4 **Explore various variable and row filters in R for cleaning data.**
Date: **Apply various plot features in R on sample data sets and Visualize**

Aim :

To explore various variable and row filters in R for cleaning data. Apply various plot features in R on sample data sets and visualize.

Algorithm :

Step1: Data Cleaning and Filtering:

a. Variable Filters:

- Selecting Variables: Use the select() function from the dplyr package to choose specific variables from a dataset.
- Dropping Variables: Use the select() function with a minus sign (-) to drop variables from a dataset.
- Renaming Variables: Use the rename() function from the dplyr package to rename variables.

b. Row Filters:

- Filtering Rows based on Conditions: Use the filter() function from the dplyr package to filter rows based on specific conditions.
- Filtering Rows based on Multiple Conditions: Combine multiple conditions using logical operators such as & (AND) and | (OR).

Step2: Plot Features in R:

R provides various libraries for creating plots and visualizations. Here are a few commonly used packages and plot types:

a. Base R:

- Scatter Plots: Use the plot() function to create a scatter plot.
- Bar Plots: Use the barplot() function to generate a bar plot.
- Histograms: Use the hist() function to create a histogram.

b. ggplot2:

- Scatter Plots: Use the geom_point() function in combination with ggplot() to create a scatter plot.
- Bar Plots: Use the geom_bar() function to generate a bar plot.
- Histograms: Use the geom_histogram() function to create a histogram.

c. Plotly:

- Interactive Plots: Use the plot_ly() function to create interactive plots, such as scatter plots, bar plots, and histograms.

.

Program:

```
dim(Iris.csv)

names(Iris.csv)

rownames(Iris.csv)

library(plotly)

library(ggplot2)
```

CCS346 - EDA

```
library(dplyr)
```

APEC

```
data(iris)
names(iris)
select_vars<-select(iris,Sepal.Length,Sepal.Width)
print(select_vars)
dropped_vars<-select(iris,Species)
print(dropped_vars)
renamed_vars<-rename(iris,Length=Sepal.Length,Width=Sepal.Width)
print(renamed_vars)
filtered_data<-filter(iris,Petal.Width>0.2)
print(filtered_data)
```

Output:

```
>dim(Iris.csv)
[1] 150 6
```

Program:

```
names(Iris.csv)
```

Output:

```
[1] "Id"            "SepalLengthCm" "SepalWidthCm"  "PetalLengthCm" "PetalWidthCm"
[6] "Species"
```

Program:

```
library(plotly)
library(ggplot2)
library(dplyr)
data(iris)
names(iris)
```

Output:

```
"Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

APEC

Program:

```
select_vars<-select(iris,Sepal.Length,Sepal.Width)
```

APEC

```
print(select_vars)
```

Output:

	Sepal.Length	Sepal.Width
1	5.1	3.5
2	4.9	3.0
3	4.7	3.2
4	4.6	3.1
5	5.0	3.6
6	5.4	3.9
7	4.6	3.4
8	5.0	3.4
9	4.4	2.9
10	4.9	3.1
11	5.4	3.7
12	4.8	3.4
13	4.8	3.0
14	4.3	3.0
15	5.8	4.0
16	5.7	4.4
17	5.4	3.9
18	5.1	3.5
19	5.7	3.8
20	5.1	3.8
21	5.4	3.4
22	5.1	3.7
23	4.6	3.6
24	5.1	3.3
25	4.8	3.4
26	5.0	3.0
27	5.0	3.4

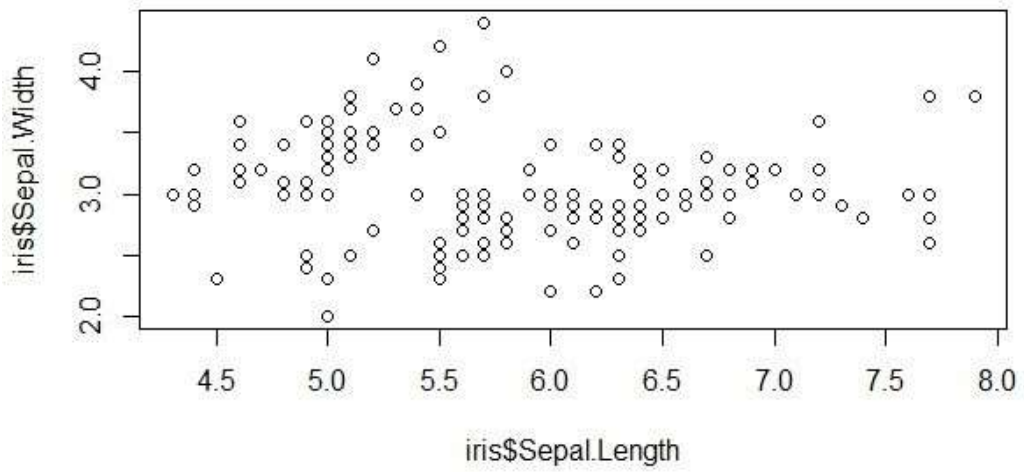
Program:

```
plot(iris$Sepal.Length,iris$Sepal.Width)
```

APEC

Output:

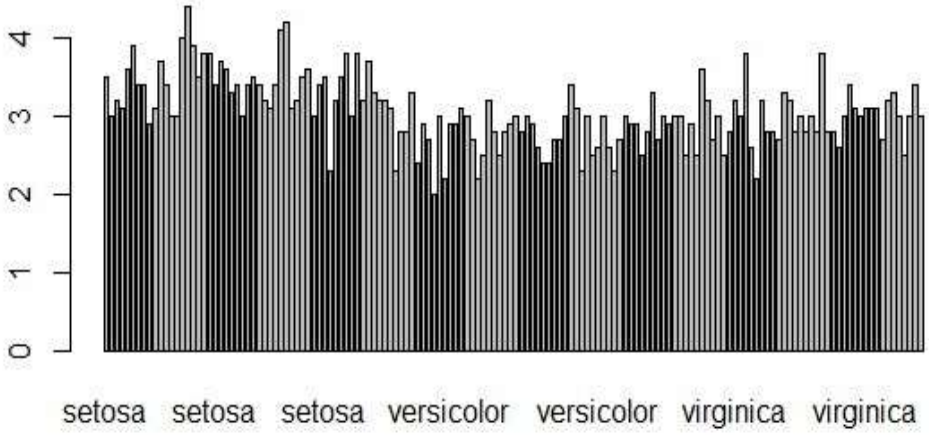
APEC



Program:

barplot(iris$Sepal.Width,names.arg = iris$Species)

Output:

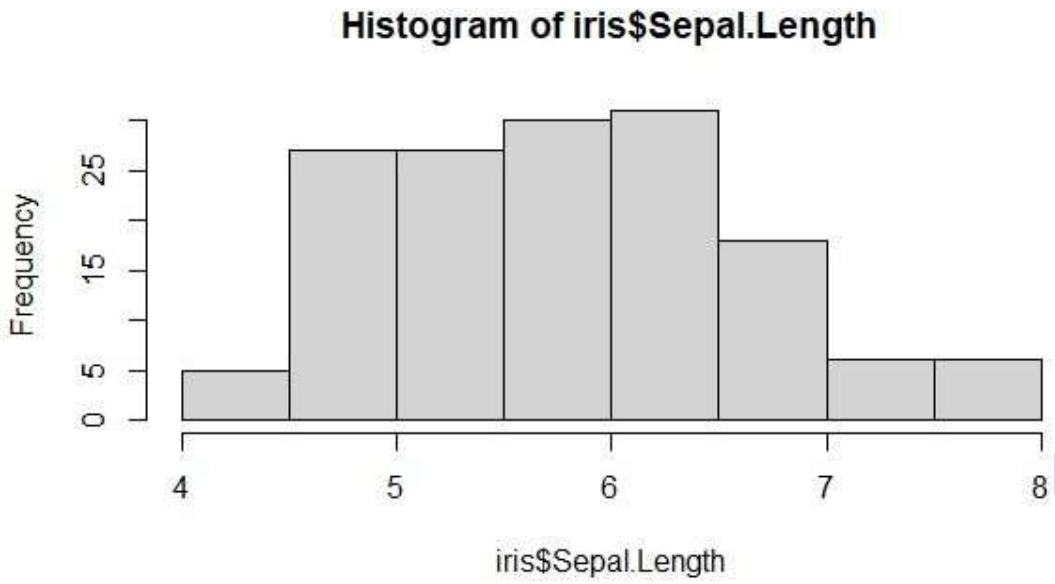


CCS346 - EDA

Program:

```
hist(iris$Sepal.Length)
```

Output:



Result :

Thus the exploring of various variable and row filters in R for cleaning data. Apply various plot features in R on sample data sets and visualization was executed and verified successfully.

APEC

Exp No : 5 Perform Time Series Analysis and apply the various visualization techniques.
Date:

Aim:

To Perform Time Series Analysis and apply the various visualization techniques in using for jupyter notebook.

Algorithm:

Step 1: Import Required Libraries:

Python provides several libraries for time series analysis, including pandas, numpy, matplotlib, and seaborn. Import these libraries to get started.

Step 2: Load and Explore the Data:

Load your time series data into a pandas DataFrame and examine its structure and properties.

Step 3: Preprocess the Data:

Time series data often requires preprocessing steps such as handling missing values, converting data types, or resampling to a consistent frequency. Perform these preprocessing steps as necessary.

Step 4: Visualize the Time Series:

Plotting the time series provides an initial understanding of the data patterns and helps identify trends, seasonality, and outliers. Use the matplotlib library for basic time series plots.

Step 5: Decompose the Time Series:

Decomposition separates a time series into its underlying components: trend, seasonality, and residual (or error). This helps in understanding the individual contributions of each component.

Step 6: Check for Stationarity:

Stationarity is an important assumption for many time series models. Test the stationarity of the time series by examining the rolling mean, rolling standard deviation, and conducting statistical tests such as the Augmented Dickey-Fuller (ADF) test.

Step 7: Apply Smoothing Techniques:

Smoothing techniques such as moving averages or exponential smoothing can help remove noise and highlight underlying patterns. Apply these techniques to obtain smoothed versions of the time series.

Step 8: Perform Seasonal Decomposition of Time Series with Trend and Residual (STL):

The Seasonal and Trend decomposition using Loess (STL) is a powerful technique to decompose a time series into trend, seasonal, and residual components while handling non-linearities. Apply STL decomposition to analyze the time series.

Program:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

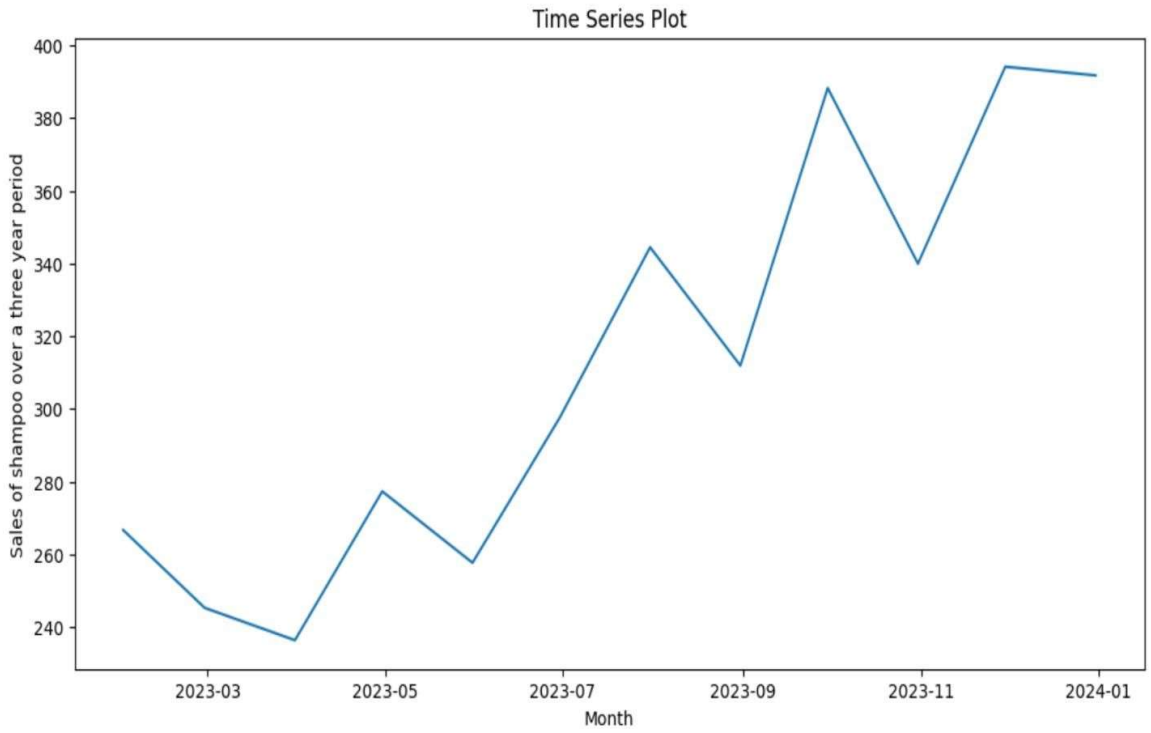
```
# Load the data into a DataFrame
df = pd.read_csv('sales-of-shampoo-over-a-three-ye.csv')
print(df.head())
print(df.info())
print(df.describe())
```

```
         Date  Sales of shampoo over a three year period
0  01-01-2023                                      266.0
1  01-02-2023                                      145.9
2  01-03-2023                                      183.1
3  01-04-2023                                      119.3
4  01-05-2023                                      180.3
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 2 columns):
 #   Column                                      Non-Null Count  Dtype
---  ------                                      --------------  -----
 0   Date                                        36 non-null     object
 1   Sales of shampoo over a three year period  36 non-null     float64
dtypes: float64(1), object(1)
memory usage: 708.0+ bytes
None
       Sales of shampoo over a three year period
count                                  36.000000
mean                                  312.600000
std                                   148.937164
min                                   119.300000
25%                                   192.450000
50%                                   280.150000
75%                                   411.100000
max                                   682.000000
```

```
print(df.head())
```

```
            Sales of shampoo over a three year period
Date
2023-01-01                                      266.0
2023-01-02                                      145.9
2023-01-03                                      183.1
2023-01-04                                      119.3
2023-01-05                                      180.3
```

```
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Sales of shampoo over a three year period'])
plt.title('Time Series Plot')
plt.xlabel('Month')
plt.ylabel('Sales of shampoo over a three year period')
plt.show()
```

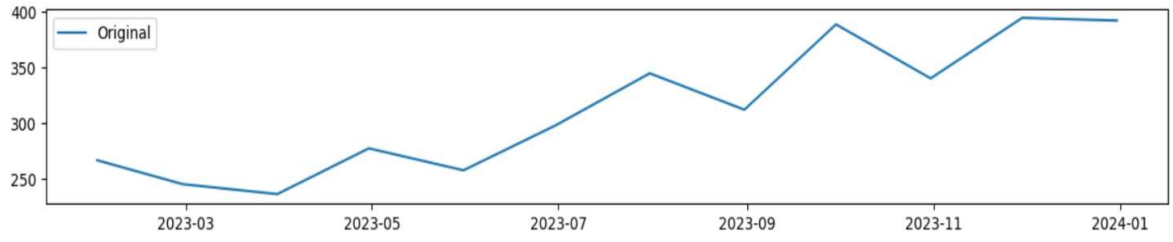


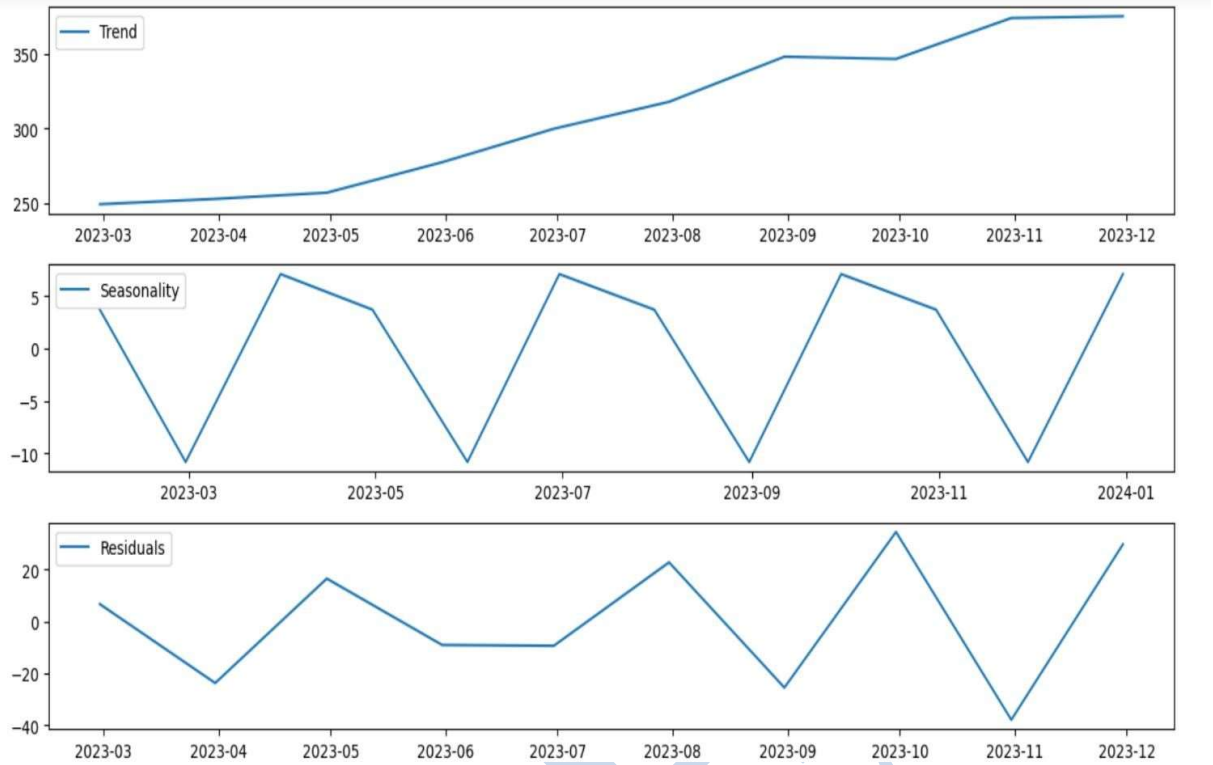

```
from statsmodels.tsa.seasonal import seasonal_decompose

# Decompose the time series

x=df['Sales of shampoo over a three year period']
decomposition = seasonal_decompose(x, model='additive',period=3)

# Plot the decomposed components
plt.figure(figsize=(12, 8))
plt.subplot(4, 1, 1)
plt.plot(df.index, df['Sales of shampoo over a three year period'], label='Original')
plt.legend(loc='best')
plt.subplot(4, 1, 2)
plt.plot(df.index, decomposition.trend, label='Trend')
plt.legend(loc='best')
plt.subplot(4, 1, 3)
plt.plot(df.index, decomposition.seasonal, label='Seasonality')
plt.legend(loc='best')
plt.subplot(4, 1, 4)
plt.plot(df.index, decomposition.resid, label='Residuals')
plt.legend(loc='best')
plt.tight_layout()
plt.show()
```



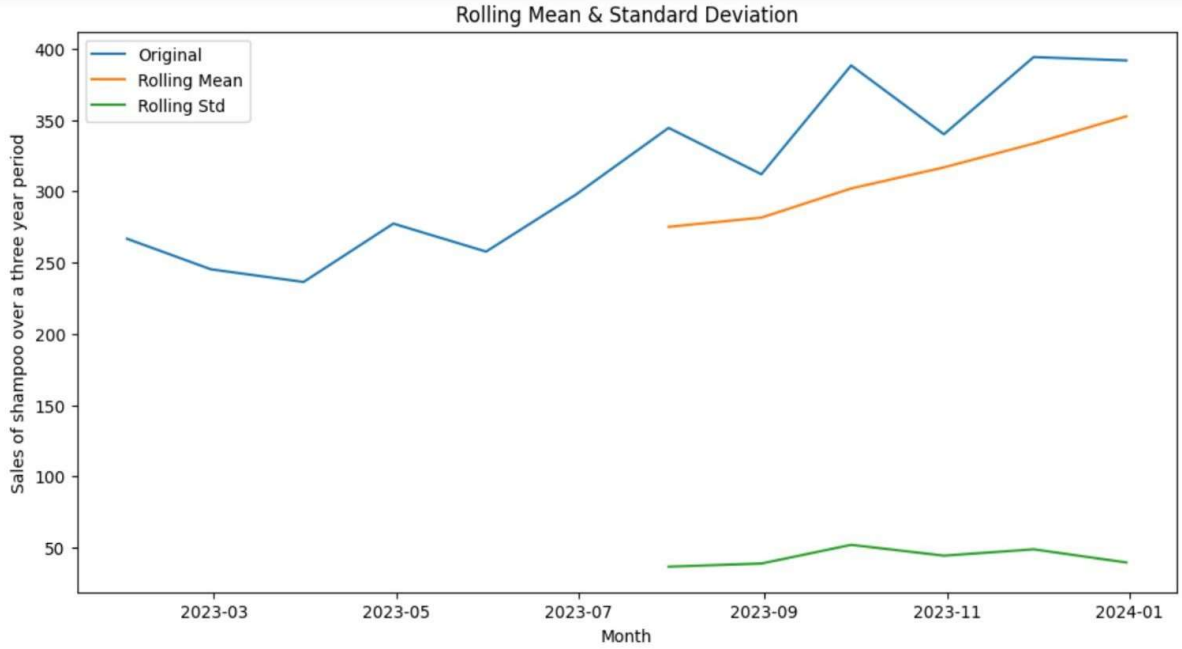



```python
from statsmodels.tsa.stattools import adfuller

# Calculate rolling statistics
rolling_mean = df['Sales of shampoo over a three year period'].rolling(window=7).mean()
rolling_std = df['Sales of shampoo over a three year period'].rolling(window=7).std()

# Plot rolling statistics
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Sales of shampoo over a three year period'], label='Original')
plt.plot(df.index, rolling_mean, label='Rolling Mean')
plt.plot(df.index, rolling_std, label='Rolling Std')
plt.title('Rolling Mean & Standard Deviation')
plt.xlabel('Month')
plt.ylabel('Sales of shampoo over a three year period')
plt.legend(loc='best')
plt.show()

# Perform the ADF test
adf_result = adfuller(df['Sales of shampoo over a three year period'])
print('ADF Statistic:', adf_result[0])
print('p-value:', adf_result[1])
print('Critical Values:')
for key, value in adf_result[4].items():
    print(f'{key}: {value}')
```



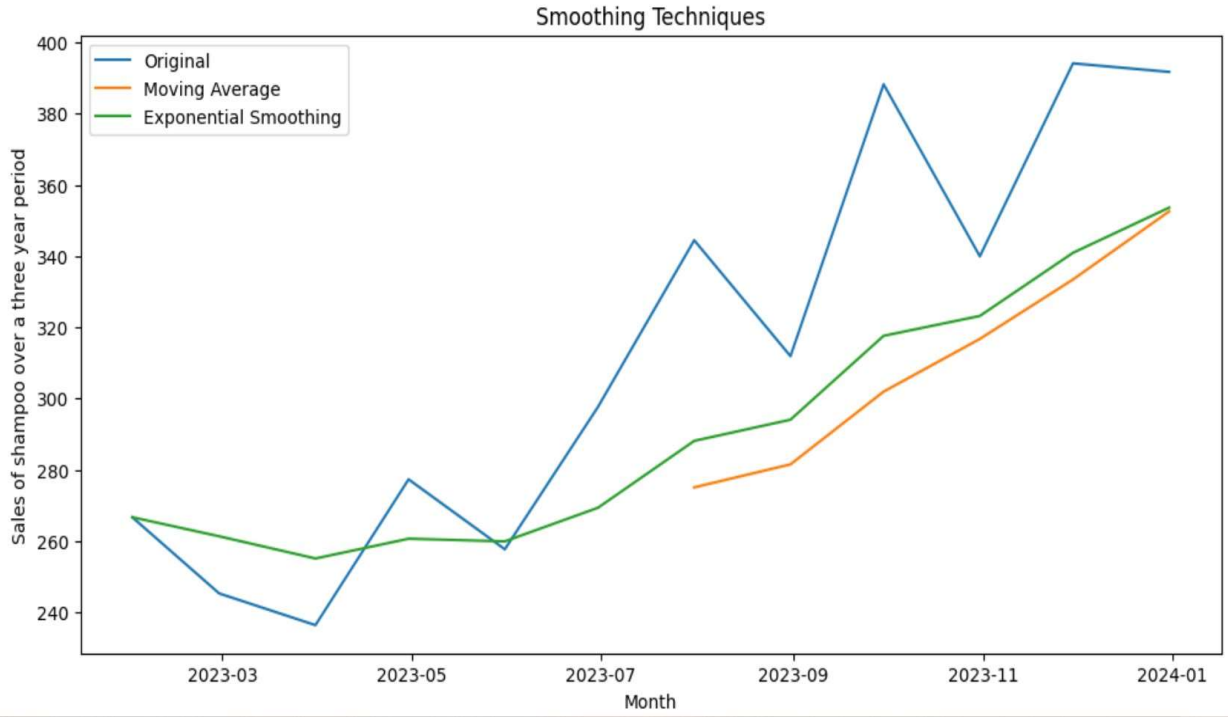

```
ADF Statistic: 0.026263177622043745
p-value: 0.9606508861612252
Critical Values:
1%: -4.473135048010974
5%: -3.28988060356653
10%: -2.7723823456790124
```

CCS346 - EDA

```
# Moving Average Smoothing
df['ma_smooth'] = df['Sales of shampoo over a three year period'].rolling(window=7).mean()

# Exponential Smoothing
df['exp_smooth'] = df['Sales of shampoo over a three year period'].ewm(span=7, adjust=False).mean()

# Plot the original and smoothed time series
plt.figure(figsize=(12, 6))
plt.plot(df.index, df['Sales of shampoo over a three year period'], label='Original')
plt.plot(df.index, df['ma_smooth'], label='Moving Average')
plt.plot(df.index, df['exp_smooth'], label='Exponential Smoothing')
plt.title('Smoothing Techniques')
plt.xlabel('Month')
plt.ylabel('Sales of shampoo over a three year period')
plt.legend(loc='best')
plt.show()
```



Result:

Thus the Python Program to Perform Time Series Analysis and apply the various visualization techniques was executed and verified successfully.

APEC

Exp No : 6 **Perform Data Analysis and representation on a Map using various**

Data: **Map data sets with Mouse Rollover effect, user interaction**

Aim :

To Perform Data Analysis and representation on a Map using various Map data sets with Mouse Rollover effect, user interaction.

Algorithm :

Step1: Set up your development environment:

- Install necessary libraries and tools, such as Leaflet.js and D3.js.
- Create an HTML file to host your map visualization.

Step2: Obtain map data:

- Find relevant map datasets that contain the geographical information you want to analyze.
- Common sources include public APIs, open data portals, or your own datasets.

Step3: Prepare your data:

- Ensure your data is in a format that can be easily consumed by the visualization libraries.
- Convert your data into GeoJSON format, which is a common format for representing geographic features.

Step4: Create an interactive map:

- Use Leaflet.js to create a base map and display it on your HTML page.
- Customize the map's appearance, such as zoom level, initial location, and map layers.

Step5: Add data layers:

- Use D3.js to overlay your GeoJSON data onto the Leaflet map.
- Apply different styles and colors to represent the data based on your analysis requirements.

Step6:Implement mouse rollover effects:

- Use JavaScript event listeners to track mouse movements over map elements.
- Trigger actions when a mouse hovers over specific areas or features, such as displaying tooltips or highlighting regions.

Step7: Enable user interaction:

- Implement additional interactive elements, such as buttons, sliders, or checkboxes, to allow users to control the data visualization.
- Connect these elements to update the map or change the displayed data based on user selections.

Step8: Publish and deploy:

APEC

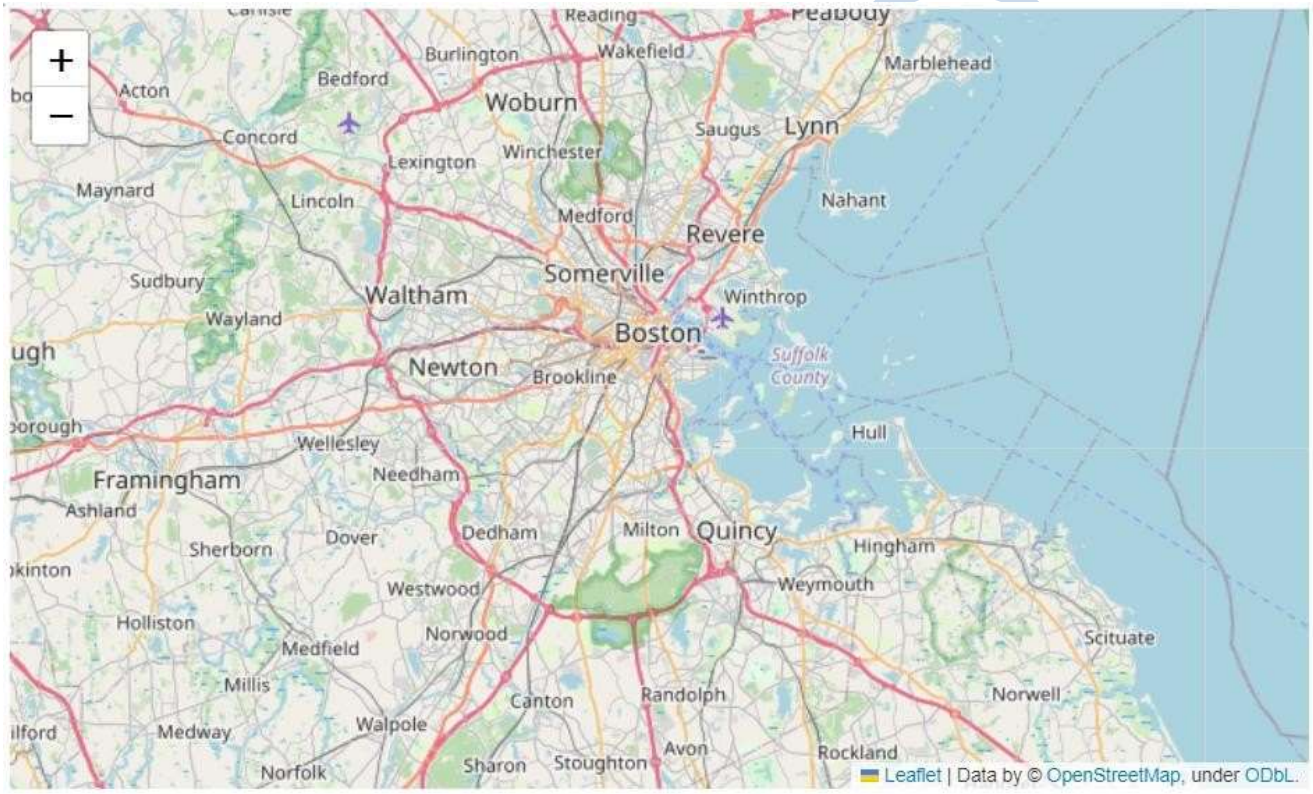
- Test your map visualization to ensure it works as expected.

APEC

Program :

```
import folium
from folium import Choropleth, Circle, Marker
from folium.plugins import HeatMap, MarkerCluster
m_1 = folium.Map(location=[42.32,-71.0589], tiles='openstreetmap', zoom_start=10)
m_1
```

Output :



Program:

```
crimes = pd.read_csv("../input/geospatial-learn-course-data/crimes-in-boston/crimes-in-boston/crime.csv", encoding='latin-1')

crimes.dropna(subset=['Lat', 'Long', 'DISTRICT'], inplace=True)

crimes = crimes[crimes.OFFENSE_CODE_GROUP.isin([
    'Larceny', 'Auto Theft', 'Robbery', 'Larceny From Motor Vehicle', 'Residential Burglary',
    'Simple Assault', 'Harassment', 'Ballistics', 'Aggravated Assault', 'Other Burglary',
    'Arson', 'Commercial Burglary', 'HOME INVASION', 'Homicide', 'Criminal Harassment',
    'Manslaughter'])]
crimes = crimes[crimes.YEAR>=2018]


crimes.head()
```

Output :

APEC

	INCIDENT_NUMBER	OFFENSE_CODE	OFFENSE_CODE_GROUP	OFFENSE_DESCRIP
0	I182070945	619	Larceny	LARCENY ALL OTH
6	I182070933	724	Auto Theft	AUTO THEFT
8	I182070931	301	Robbery	ROBBERY - STREET
19	I182070915	614	Larceny From Motor Vehicle	LARCENY THEFT F MV - NON-ACCESS
24	I182070908	522	Residential Burglary	BURGLARY - RESIDENTIAL - NO FORCE

Program :

```
daytime_robberies = crimes[((crimes.OFFENSE_CODE_GROUP == 'Robbery') & \
                    (crimes.HOUR.isin(range(9,18))))]
```

```
m_2 = folium.Map(location=[42.32,-71.0589], tiles='cartodbpositron', zoom_start=13)

for idx, row in daytime_robberies.iterrows():
    Marker([row['Lat'], row['Long']]).add_to(m_2)

m_2
```

Output :



Result :

Thus the Data Analysis and representation on a Map using various Map data sets with Mouse Rollover

effect, user interaction was executed and verified successfully.

APEC

Exp No : 7 **Build cartographic visualization for multiple datasets involving various countries of the world; states and districts in India etc.**

Date:

Aim:

To Build cartographic visualization for multiple datasets involving various countries of the world; states and districts in India etc.

Algorithm:

Step 1: Obtain the data:

Collect the datasets for the countries of the world, states in India, and districts within those states. Ensure the datasets contain relevant information for your visualization, such as numerical values or categorical data that can be mapped.

Step 2: Preprocess the data:

Clean and preprocess the datasets to ensure they are in a usable format. Handle missing values, standardize data formats, and perform any necessary transformations.

Step 3: Choose a mapping library:

Select a mapping library or tool that supports the features and functionalities you require for your visualization. Popular options include D3.js, Leaflet, Mapbox, or Python libraries like Geopandas or Folium.

Step 4: Obtain geographic boundaries:

Acquire shapefiles or GeoJSON files containing the geographic boundaries for the countries, states, and districts you want to visualize. These files define the polygons that represent the respective boundaries.

Step 5: Merge data with boundaries:

Join the datasets with the geographic boundaries based on a common identifier (such as country or region codes) to associate the data with the corresponding polygons.

Step 6: Define a color scale:

Determine the color scale or classification scheme you want to use to represent the data values. For numerical data, you can use a gradient or diverging color scheme. For categorical data, assign distinct colors to each category.

Step 7: Create the base map:

Use the mapping library to create the base map by rendering the country and state boundaries. Customize the map's style, such as the background color, stroke width, and fill color.

Step 8: Visualize the data:

Overlay the data onto the map by coloring the polygons based on the associated values. Apply the chosen color scale to differentiate the data across the map.

Step 9: Test and refine:

Validate the visualization by examining its accuracy and usability. Ensure that the map accurately represents the data and that users can interpret it effectively. Refine the visualization as needed based on feedback or usability testing.

Step 10: Deploy and share:

Once satisfied with the visualization, deploy it on a suitable platform or embed it in a website. Share it with the intended audience to communicate the insights derived from the data.

Program:

Landslide Analysis

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

import seaborn as sns
import geopandas as gpd
from shapely.geometry import Point

sns.set_style('whitegrid')
```

Importing the dataset

```
df = pd.read_csv('globallandslides.csv')
df_copy = pd.read_csv('globallandslides.csv')  # making a copy of dataframe
```

```
pd.set_option('display.max_columns', None)
```

```
df.head()
```

Out[4]:

	source_name	source_link	event_id	event_date	event_time	event_title	event_description	location_description	location_a
0	AGU	https://blogs.agu.org/landslideblog/2008/10/14...	684	08/01/2008 12:00:00 AM	NaN	Sigou Village, Loufan County, Shanxi Province	occurred early in morning, 11 villagers buried...	Sigou Village, Loufan County, Shanxi Province	
1	Oregonian	http://www.oregonlive.com/news/index.ssf/2009/...	956	01/02/2009 02:00:00 AM	NaN	Lake Oswego, Oregon	Hours of heavy rain are to blame for an overni...	Lake Oswego, Oregon	
2	CBS News	https://www.cbsnews.com/news/dozens-missing-af...	973	01/19/2007 12:00:00 AM	NaN	San Ramon district, 195 miles northeast of the...	(CBS/AP) At least 10 people died and as many a...	San Ramon district, 195 miles northeast of the...	
3	Reuters	https://in.reuters.com/article/idINIndia-41450...	1067	07/31/2009 12:00:00 AM	NaN	Dailekh district	One person was killed in Dailekh district, pol...	Dailekh district	

```
df = df[df.country_name=="India"]  # Selecting data for India
```

```
df["Year"] = pd.to_datetime(df["event_date"]).dt.year    # This extracts the year in which the event to
```

```
df = df[df.landslide_category=="landslide"]  # Selecting only landslide category
```

Task 1 : A map of only landslides that have happened within India

```
ls_df = df   # making a dataframe of landslides
ls_df = ls_df.dropna(axis=0, subset=['admin_division_name'])
```

Performing data cleaning ¶

```
# Data cleaning
ls_df["admin_division_name"].replace("Nāgāland", "Nagaland",inplace = True)
ls_df["admin_division_name"].replace("Meghālaya", "Meghalaya",inplace = True)
ls_df["admin_division_name"].replace("Tamil Nādu", "Tamil Nadu",inplace = True)
ls_df["admin_division_name"].replace("Karnātaka", "Karnataka",inplace = True)
ls_df["admin_division_name"].replace("Gujarāt", "Gujarat",inplace = True)
ls_df["admin_division_name"].replace("Arunāchal Pradesh", "Arunachal Pradesh",inplace = True)
```

```
state_df = ls_df["admin_division_name"].value_counts()   # This gives the number of landslides that toc
state_df = state_df.to_frame()
state_df.reset_index(level=0, inplace=True)
state_df.columns = ['State', 'Count']                    # Renaming the columns
```

```
state_df.head()
```

Out[12]:

	State	Count
0	Kashmir	189
1	Uttarakhand	167
2	Himachal Pradesh	137
3	Assam	74
4	Nagaland	65

In [13]:
```
state_df.at[15,"Count"] = 69   # Since the entries "Bengal" and "West Bengal" are
state_df.drop(7)
```

Out[13]:

	State	Count
0	Kashmir	189
1	Uttarakhand	167
2	Himachal Pradesh	137
3	Assam	74
4	Nagaland	65
5	Maharashtra	54
6	Manipur	47
8	Kerala	43
9	Arunachal Pradesh	40
10	Tamil Nadu	32
11	Sikkim	28
12	Karnataka	28
13	Meghalaya	25
14	Mizoram	25
15	West Bengal	69

16	Goa	19
17	Andhra Pradesh	15
18	Rajasthan	5
19	Odisha	5
20	NCT	3
21	Orissa	3
22	Uttar Pradesh	2
23	Gujarat	2
24	Tripura	2
25	Haryana	2
26	State of Odisha	1
27	Madhya Pradesh	1
28	Bihar	1
29	Jharkhand	1

In [14]:
```
fp = r'Maps_with_python\india-polygon.shp'
map_df = gpd.read_file(fp)                    # Reads the shape file of India for plotting purpos
map_df_copy = gpd.read_file(fp)
map_df.head()
```

Out[14]:

	id	st_nm	geometry
0	None	Andaman and Nicobar Islands	MULTIPOLYGON (((93.84831 7.24028, 93.92705 7.0...
1	None	Arunachal Pradesh	POLYGON ((95.23643 26.68105, 95.19594 27.03612...
2	None	Assam	POLYGON ((95.19594 27.03612, 95.08795 26.94578...
3	None	Bihar	POLYGON ((88.11357 26.54028, 88.28006 26.37640...
4	None	Chandigarh	POLYGON ((76.84208 30.76124, 76.83758 30.72552...

In [15]:
```
state_df.at[0,"State"] = "Jammu and Kashmir"       # Renaming certain entries to avoid conflicts while
state_df.at[20,"State"] = "Delhi"
```

In [16]:
```
#Merging the data
merged = map_df.set_index('st_nm').join(state_df.set_index('State'))
merged['Count'] = merged['Count'].replace(np.nan, 0)
merged.head()
```

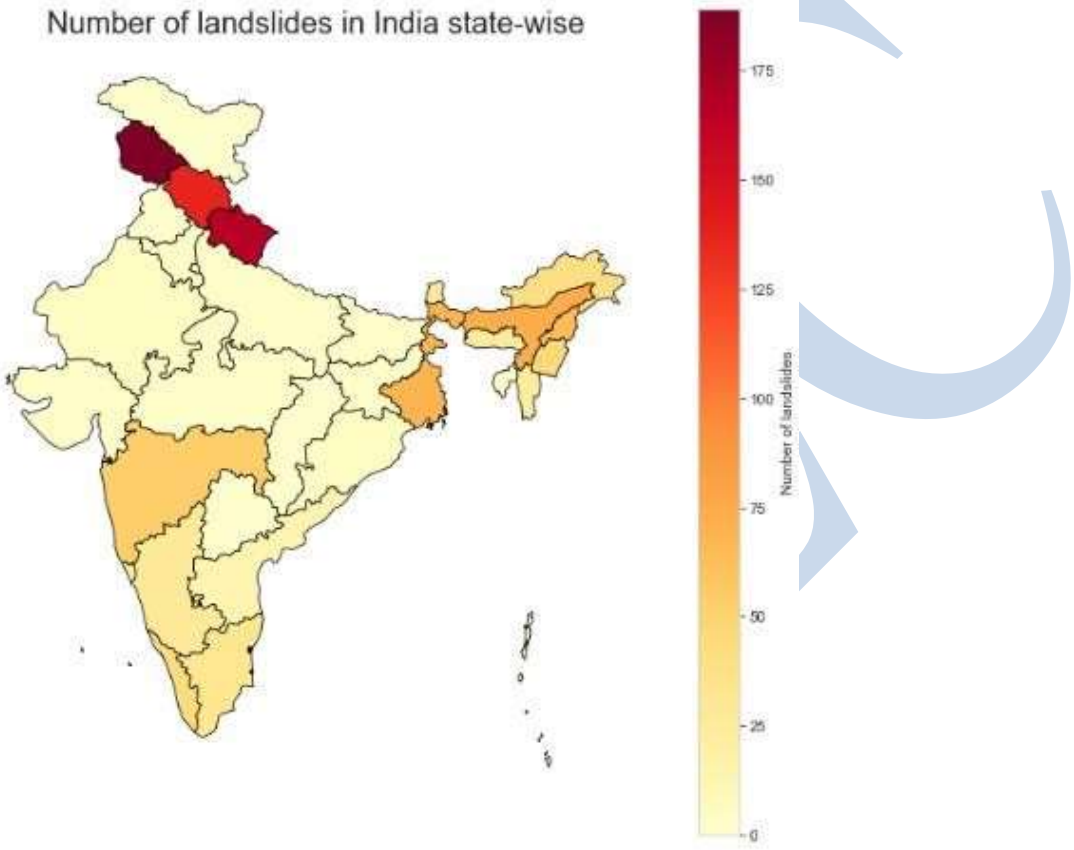
Out[16]:

st_nm	id	geometry	Count
Andaman and Nicobar Islands	None	MULTIPOLYGON (((93.84831 7.24028, 93.92705 7.0...	0.0
Arunachal Pradesh	None	POLYGON ((95.23643 26.68105, 95.19594 27.03612...	40.0
Assam	None	POLYGON ((95.19594 27.03612, 95.08795 26.94578...	74.0
Bihar	None	POLYGON ((88.11357 26.54028, 88.28006 26.37640...	1.0
Chandigarh	None	POLYGON ((76.84208 30.76124, 76.83758 30.72552...	0.0

In [17]:
```
#Create figure and axes for Matplotlib and set the title
fig, ax = plt.subplots(1, figsize=(10, 10))
ax.axis('off')
ax.set_title('Number of landslides in India state-wise', fontdict={'fontsize': '20', 'fontweight' : '10'})

# Plot the figure
merged.plot(column='Count',cmap='YlOrRd', linewidth=0.8, ax=ax, edgecolor='0', \
            legend=True,markersize=[39.739192, -104.990337], legend_kwds={'label': "Number of landslides"})
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2230d29d908>
```



In [18]:
```
# Finding the latitudes and longitudes of landslides that took place in India
lat = df['latitude'].tolist()
long = df['longitude'].tolist()
```

In [19]:
```
india_state_df = pd.read_csv('state wise lat and long.csv')     # Reading the coordinates of Indian states for plotting purpose
india_state_df.head()
```

Out[19]:

	State	Longitude	Latitude
0	Andaman and Nicobar	92.904257	11.845455
1	Andhra Pradesh	79.973851	16.557796
2	Arunachal Pradesh	94.662314	27.729050
3	Assam	92.685686	26.336086
4	Bihar	85.625341	25.766303

In [20]:
```
india_state_df.at[0,"State"] = "Andaman and Nicobar Islands"    # Renaming certaing entries to avoid co
india_state_df.at[25,"State"] = "Odisha"
india_state_df.at[33,"State"] = "Uttarakhand"
```

In [21]:
```
# Merge the data
state_merged_df = map_df.set_index('st_nm').join(india_state_df.set_index('State'))
```

In [22]:
```
state_merged_df.reset_index(level=0, inplace=True)
state_merged_df.head()
```

Out[22]:

	st_nm	id	geometry	Longitude	Latitude
0	Andaman and Nicobar Islands	None	MULTIPOLYGON (((93.84831 7.24028, 93.92705 7.0...	92.904257	11.845455
1	Arunachal Pradesh	None	POLYGON ((95.23643 26.68105, 95.19594 27.03612...	94.662314	27.729050
2	Assam	None	POLYGON ((95.19594 27.03612, 95.08795 26.94578...	92.685686	26.336086
3	Bihar	None	POLYGON ((88.11357 26.54028, 88.28006 26.37640...	85.625341	25.766303
4	Chandigarh	None	POLYGON ((76.84208 30.76124, 76.83758 30.72552...	76.758725	30.742600

In [23]:
```
# Handling missng values
state_merged_df.at[36,"Latitude"] = 34.2268
state_merged_df.at[36,"Longitude"] = 77.5619
state_merged_df.at[28,"Latitude"] = 18.1124
state_merged_df.at[28,"Longitude"] = 79.0193
```

In [24]:
```
geometry = [Point(xy) for xy in zip(long,lat)]
fig,ax = plt.subplots(figsize = (13,13))
ax.axis('off')
ax.set_title('Map of India showing where landslides have occured over the years', fontdict={'fontsize':

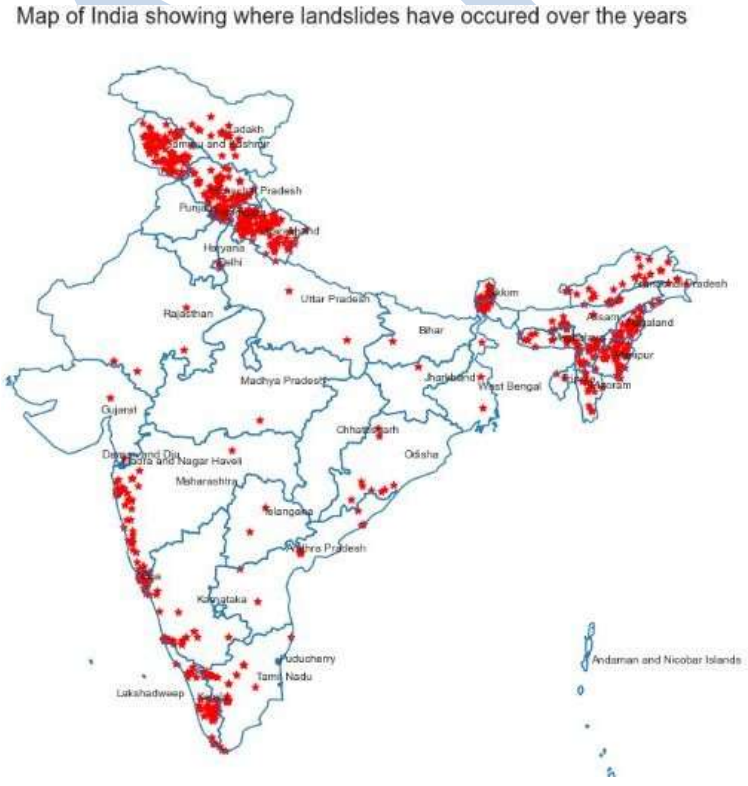
map_df.boundary.plot(ax = ax)

geo_df = gpd.GeoDataFrame(geometry = geometry)

g = geo_df.plot(ax = ax, markersize = 50, color = 'red',marker = '*',label = 'Landslide')

for i in range(len(state_merged_df)):
    plt.text(state_merged_df.Longitude[i],state_merged_df.Latitude[i],"{}".format(state_merged_df.st_nm

plt.show()
```



In [25]:
```
# Finding the latitudes and longitudes of landlides that took place in Assam
lat = df[df['admin_division_name'] == "Assam"]['latitude'].tolist()
long = df[df['admin_division_name'] == "Assam"]['longitude'].tolist()
```

In [26]:
```
len(lat)
```

Out[26]: 74

In [27]:
```
len(long)
```

Out[27]: 74

In [28]:
```
fp = r'Districts shape file\Census_2011\2011_Dist.shp'
assam_df = gpd.read_file(fp)                          # Reads the shape file of Assam for plotting pur
assam_df .head()
```

Out[28]:

	DISTRICT	ST_NM	ST_CEN_CD	DT_CEN_CD	censuscode	geometry
0	Adilabad	Andhra Pradesh	28	1	532	POLYGON ((78.84972 19.76010, 78.85102 19.75945...
1	Agra	Uttar Pradesh	9	15	146	POLYGON ((78.19803 27.40280, 78.19804 27.40278...
2	Ahmadabad	Gujarat	24	7	474	MULTIPOLYGON (((72.03456 23.50527, 72.03337 23...
3	Ahmadnagar	Maharashtra	27	26	522	POLYGON ((74.67333 19.94670, 74.67393 19.93509...
4	Aizawl	Mizoram	15	3	283	POLYGON ((92.98749 24.40453, 92.99107 24.40236...

In [29]:
```
assam_df  = assam_df[assam_df.ST_NM == "Assam"]
```

In [30]:
```
# Reads the data of latitudes and longitudes for the districts of Assam
district_df = pd.read_csv('district wise lat and long.csv')
```

In [31]:
```
district_df = district_df[district_df.State == "Assam"]
```

In [32]:
```
district_df.drop(['Population in 2001', 'Population in 2011'], axis='columns', inplace=True)  # removir
```

In [33]:
```
district_df.head()
```

Out[33]:

	State	District	Latitude	Longitude
24	Assam	Barpeta	26.427872	90.979120
25	Assam	Bongaigaon	26.450125	90.673430
26	Assam	Cachar	24.815571	92.868497
27	Assam	Darrang	26.584632	92.041413
28	Assam	Dhemaji	27.586533	94.713369

In [34]:
```
# Merge the data
merged_df = assam_df.set_index('DISTRICT').join(district_df.set_index('District'))
merged_df.head()
merged_df = merged_df[merged_df.Latitude.notna()]
merged_df.head()
```

Out[34]:

DISTRICT	ST_NM	ST_CEN_CD	DT_CEN_CD	censuscode	geometry	State	Latitude	Longitude
Barpeta	Assam	18	4	303	POLYGON ((91.16927 26.65240, 91.16787 26.64642...	Assam	26.427872	90.979120
Bongaigaon	Assam	18	20	319	POLYGON ((90.85418 26.52336, 90.85336 26.52153...	Assam	26.450125	90.673430
Cachar	Assam	18	17	316	POLYGON ((93.25235 24.97158, 93.25237 24.96905...	Assam	24.815571	92.868497
Darrang	Assam	18	26	325	POLYGON ((92.32865 26.67036, 92.33490 26.66867...	Assam	26.584632	92.041413
Dhemaji	Assam	18	9	308	POLYGON ((94.58063 27.21371, 94.57922 27.21355...	Assam	27.586533	94.713369

In [35]:
```
merged_df.reset_index(level=0, inplace=True)
merged_df.head()
```

Out[35]:

	DISTRICT	ST_NM	ST_CEN_CD	DT_CEN_CD	censuscode	geometry	State	Latitude	Longitude
0	Barpeta	Assam	18	4	303	POLYGON ((91.16927 26.65240, 91.16787 26.64642...	Assam	26.427872	90.979120
1	Bongaigaon	Assam	18	20	319	POLYGON ((90.85418 26.52336, 90.85336 26.52153...	Assam	26.450125	90.673430
2	Cachar	Assam	18	17	316	POLYGON ((93.25235 24.97158, 93.25237 24.96905...	Assam	24.815571	92.868497
3	Darrang	Assam	18	26	325	POLYGON ((92.32865 26.67036, 92.33490 26.66867...	Assam	26.584632	92.041413
4	Dhemaji	Assam	18	9	308	POLYGON ((94.58063 27.21371, 94.57922 27.21355...	Assam	27.586533	94.713369

In [36]:
```
geometry = [Point(xy) for xy in zip(long,lat)]
fig,ax = plt.subplots(figsize = (15,15))
ax.axis('off')
ax.set_title('Map of Assam showing where landslides have occured over the years', fontdict={'fontsize': '20', 'fontweight' : '10'

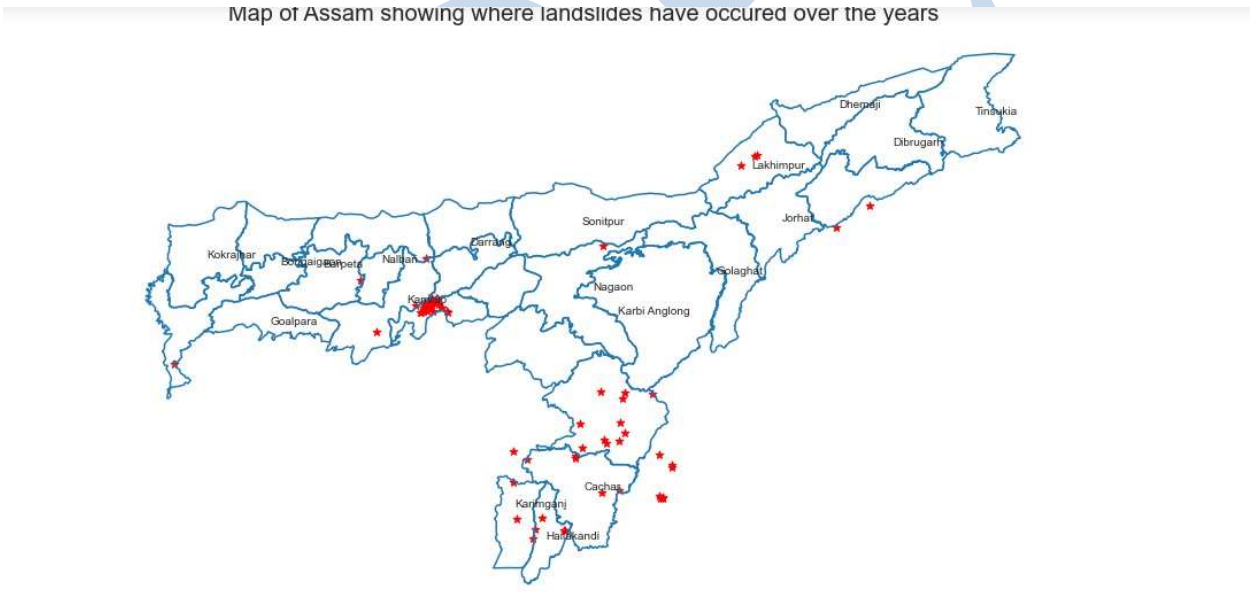
assam_df.boundary.plot(ax = ax)

geo_df = gpd.GeoDataFrame(geometry = geometry)

g = geo_df.plot(ax = ax, markersize = 50, color = 'red',marker = '*',label = 'Landslide')

for i in range(len(district_df)):
    plt.text(merged_df.Longitude[i],merged_df.Latitude[i],"{}".format(merged_df.DISTRICT[i]),size=10)

plt.show()
```



Task 2 : A plot showing the total number of landslides that have happened in India every year

In [37]:
```
y_values = df.groupby(['Year']).country_name.count().values
```

In [38]:
```
y_values.tolist()
```

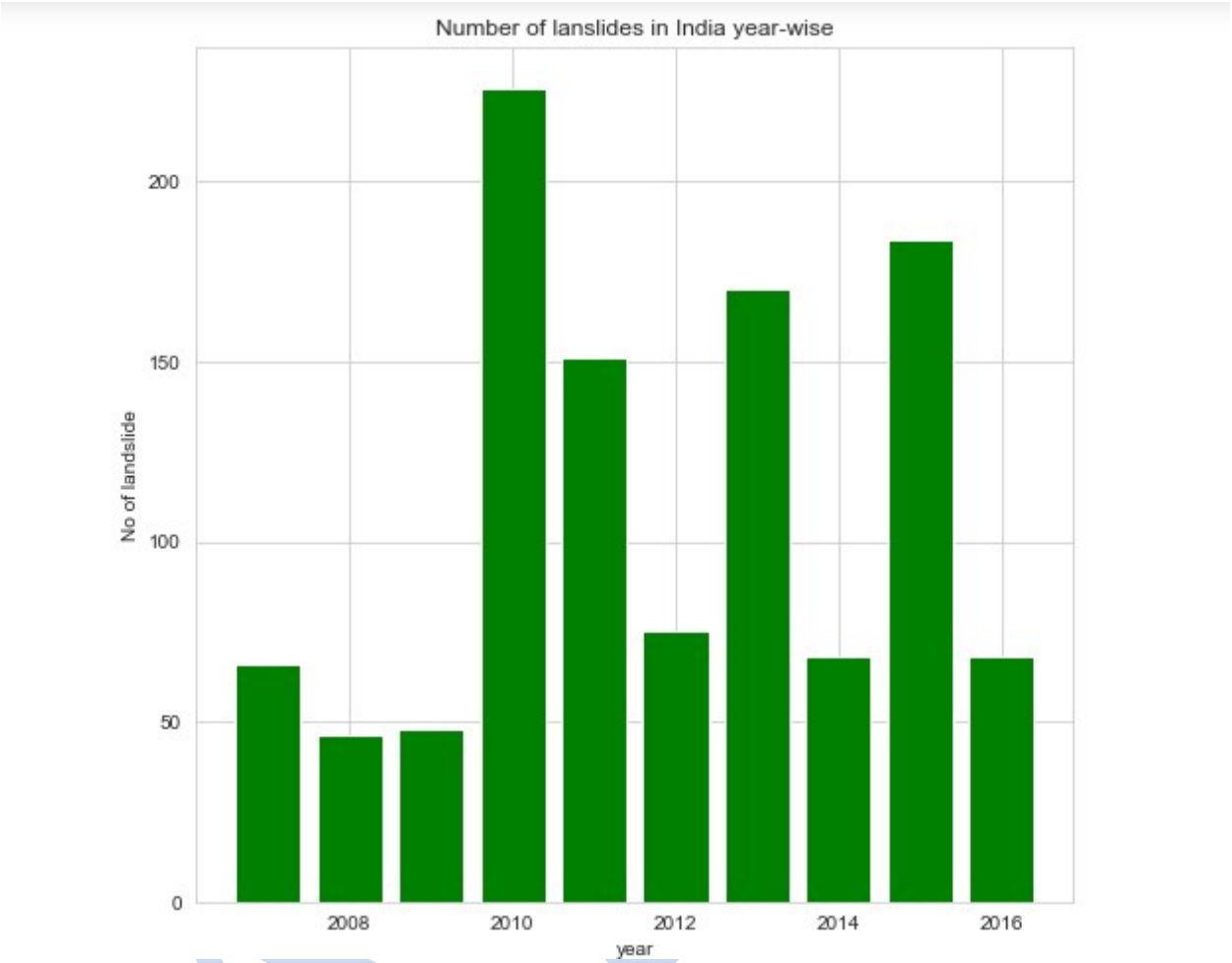
Out[38]: [66, 46, 48, 226, 151, 75, 170, 68, 184, 68]

In [39]:
```
x_values = df.groupby(['Year']).country_name.count().index.values
```

In [40]:
```
x_values.tolist()
```

Out[40]: [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016]

In [41]:
```
plt.figure(figsize = (8,8))
plt.bar(x_values,y_values, color = "green")
plt.xlabel("year")
plt.ylabel("No of landslide")
plt.title("Number of lanslides in India year-wise")
plt.show()
```



Result:

Thus the Build cartographic visualization for multiple datasets involving various countries of the world; states and districts in India was verified and Executed Successfully.

APEC

Exp No : 8 **Perform EDA on Wine Quality Data Set. Use a case study on a data set and apply the various EDA and visualization techniques and present an analysis report.**

Date:

Aim:

To Perform EDA on Wine Quality Data Set. Use a case study on a data set and apply the various EDA and visualization techniques and present an analysis report.

Algorithm:

Step 1: Load the Dataset:

Start by loading the Wine Quality Data Set into your preferred data analysis environment, such as Python or R.

Step 2: Explore the Dataset:

Take a look at the dataset's structure, features, and their data types. Check for missing values, duplicates, or any other data quality issues.

Step 3: Summarize the Dataset:

Calculate basic statistics of the dataset, such as mean, median, standard deviation, minimum, maximum, etc. This will provide an initial understanding of the data distribution.

Step 4: Visualize the Target Variable:

Since the dataset includes a quality rating for each wine sample, it's important to understand its distribution. Create a histogram or bar plot to visualize the distribution of wine quality ratings.

Step 5: Explore Correlations:

Calculate the correlation matrix to understand the relationships between different features. Visualize the correlation matrix using a heatmap to identify potential correlations between features.

Step 6: Analyze Feature Distributions:

Examine the distribution of each feature using histograms, box plots, or violin plots. This will help identify outliers, skewed distributions, or any patterns within the data.

Step 7: Feature Interactions:

Investigate how different features interact with each other. Use scatter plots or pair plots to visualize the relationships between pairs of features and observe any trends or patterns.

Step 8: Feature Engineering:

Based on the insights gained from the EDA, consider creating new features or transforming existing ones to enhance the predictive power of your models.

Step 9: Additional Analysis:

Perform any additional analysis that may be relevant to your specific case study. This could include comparing distributions across different wine types, analyzing the impact of specific features on wine quality, or any other exploration based on your domain knowledge.

Step 10: Draw Conclusions:

Summarize your findings and insights from the EDA. Identify any interesting patterns, correlations, or outliers that could be further investigated or used in predictive modeling.

Program:

```
import pandas as pd

# Load the dataset
wine_data = pd.read_csv('WineQT.csv')
```

```
# Check the structure of the dataset
print(wine_data.head())

# Check the data types of the features
print(wine_data.dtypes)

# Check for missing values
print(wine_data.isnull().sum())

# Check for duplicates
print(wine_data.duplicated().sum())
```

```
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         0.04             2.3      0.092
3           11.2              0.28         0.56             1.9      0.075
4            7.4              0.70         0.00             1.9      0.076
```

```
   free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                 11.0                  34.0   0.9978  3.51       0.56
1                 25.0                  67.0   0.9968  3.20       0.68
2                 15.0                  54.0   0.9970  3.26       0.65
3                 17.0                  60.0   0.9980  3.16       0.58
4                 11.0                  34.0   0.9978  3.51       0.56

   alcohol  quality  Id
0      9.4        5   0
1      9.8        5   1
2      9.8        5   2
3      9.8        6   3
4      9.4        5   4
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
alcohol                 float64
quality                   int64
Id                        int64
dtype: object
```

```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
Id                      0
dtype: int64
0
```

```
# Calculate basic statistics
print(wine_data.describe())
```

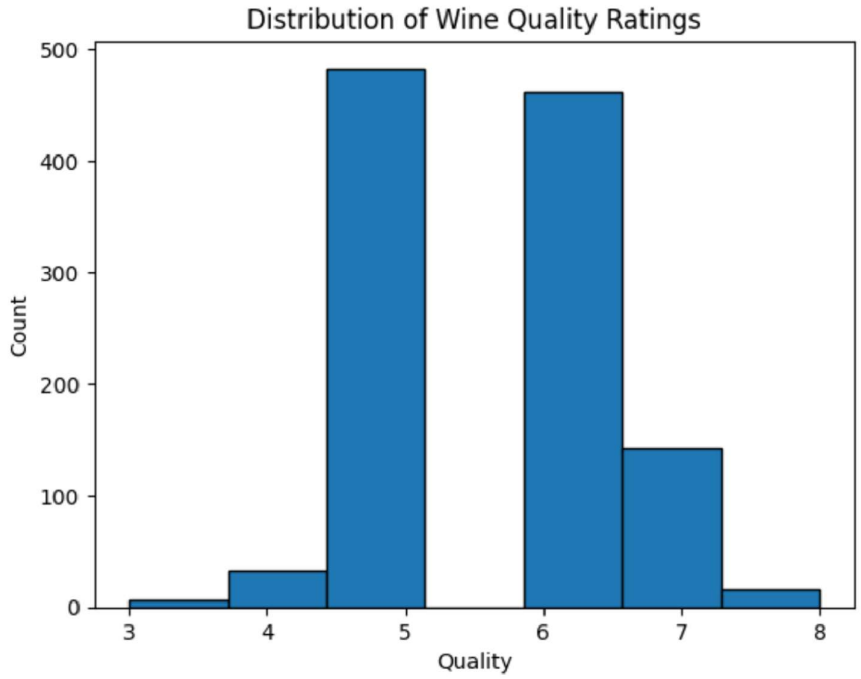
```
       fixed acidity  volatile acidity  citric acid  residual sugar  \
count    1143.000000       1143.000000  1143.000000     1143.000000
mean        8.311111          0.531339     0.268364        2.532152
std         1.747595          0.179633     0.196686        1.355917
min         4.600000          0.120000     0.000000        0.900000
25%         7.100000          0.392500     0.090000        1.900000
50%         7.900000          0.520000     0.250000        2.200000
75%         9.100000          0.640000     0.420000        2.600000
max        15.900000          1.580000     1.000000       15.500000

         chlorides  free sulfur dioxide  total sulfur dioxide      density  \
count  1143.000000          1143.000000           1143.000000  1143.000000
mean      0.086933            15.615486             45.914698     0.996730
std       0.047267            10.250486             32.782130     0.001925
min       0.012000             1.000000              6.000000     0.990070
25%       0.070000             7.000000             21.000000     0.995570
50%       0.079000            13.000000             37.000000     0.996680
75%       0.090000            21.000000             61.000000     0.997845
max       0.611000            68.000000            289.000000     1.003690
```

	pH	sulphates	alcohol	quality	Id
count	1143.000000	1143.000000	1143.000000	1143.000000	1143.000000
mean	3.311015	0.657708	10.442111	5.657043	804.969379
std	0.156664	0.170399	1.082196	0.805824	463.997116
min	2.740000	0.330000	8.400000	3.000000	0.000000
25%	3.205000	0.550000	9.500000	5.000000	411.000000
50%	3.310000	0.620000	10.200000	6.000000	794.000000
75%	3.400000	0.730000	11.100000	6.000000	1209.500000
max	4.010000	2.000000	14.900000	8.000000	1597.000000

```
import matplotlib.pyplot as plt

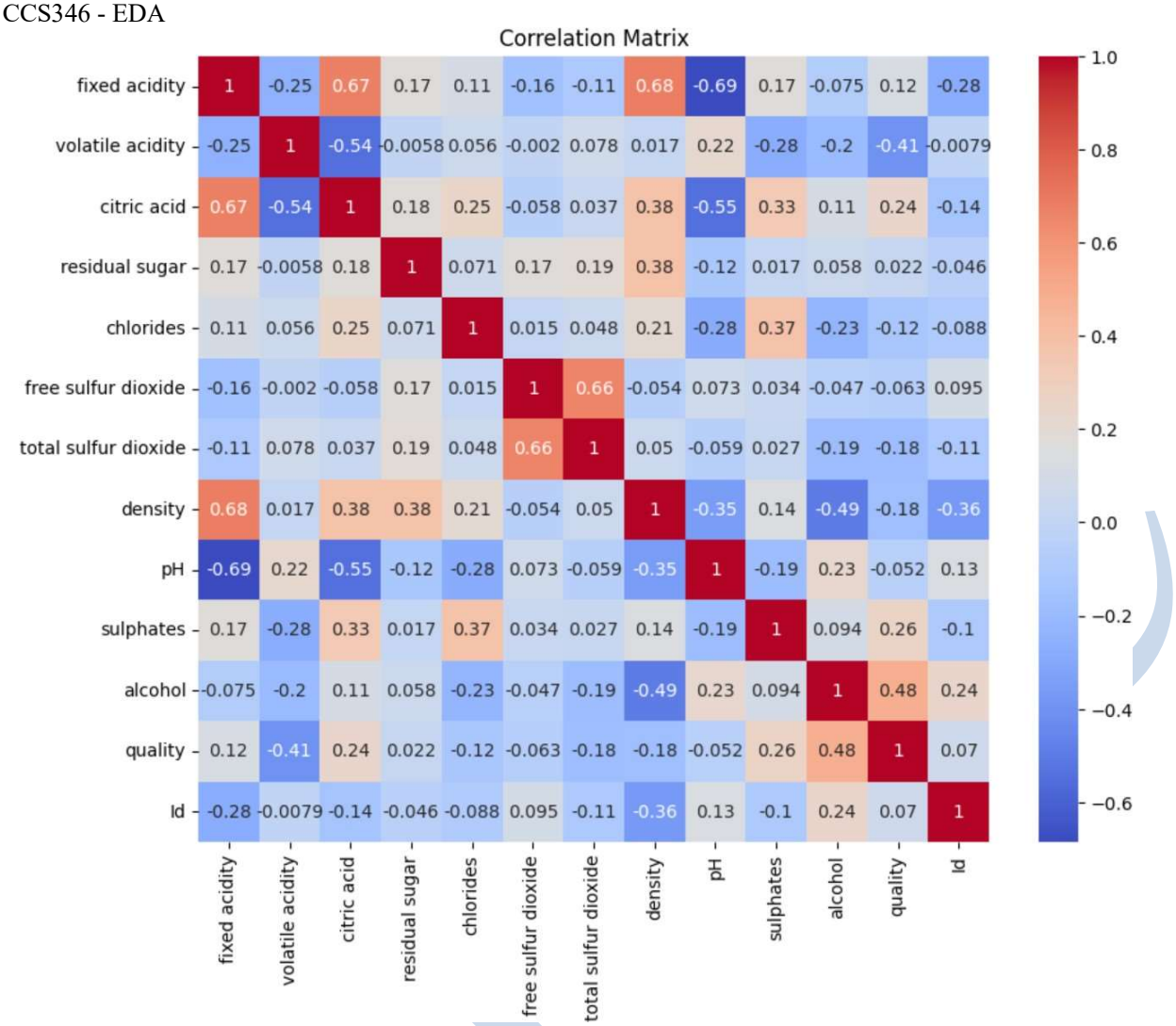
# Create a histogram of wine quality ratings
plt.hist(wine_data['quality'], bins=7, edgecolor='black')
plt.xlabel('Quality')
plt.ylabel('Count')
plt.title('Distribution of Wine Quality Ratings')
plt.show()
```



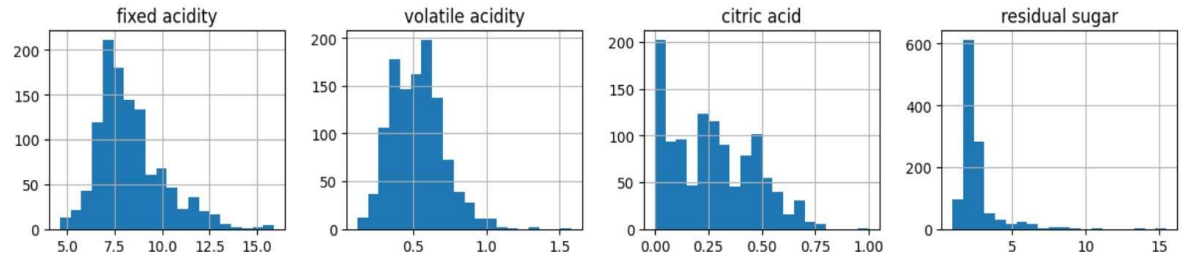

```
import seaborn as sns

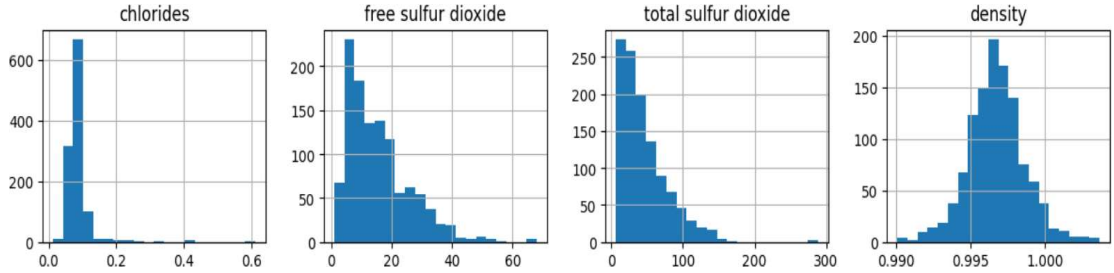
# Calculate the correlation matrix
correlation_matrix = wine_data.corr()

# Visualize the correlation matrix using a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



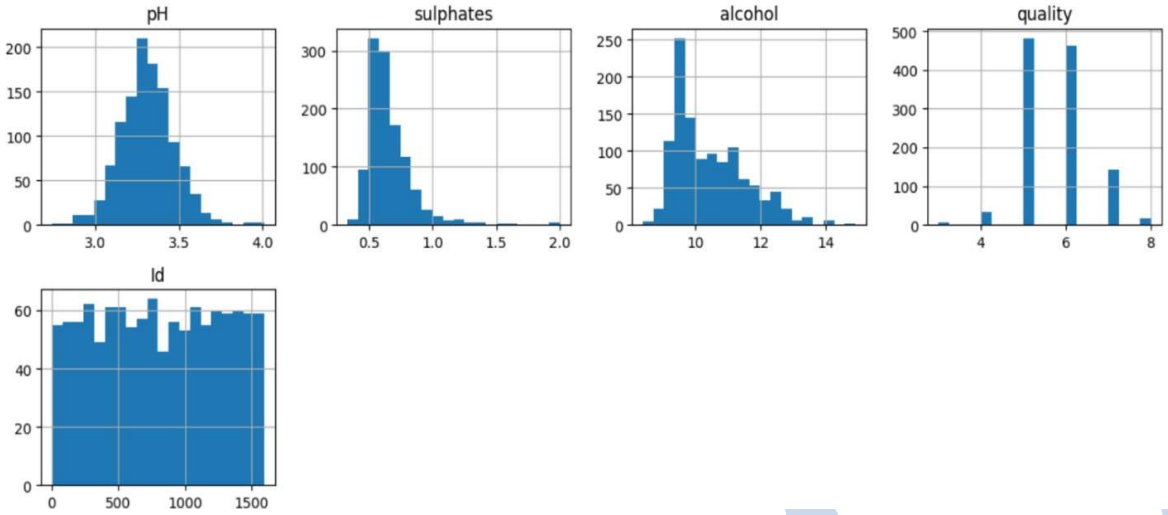

```
# Create histograms for each feature
wine_data.hist(bins=20, figsize=(12, 10))
plt.tight_layout()
plt.show()
```



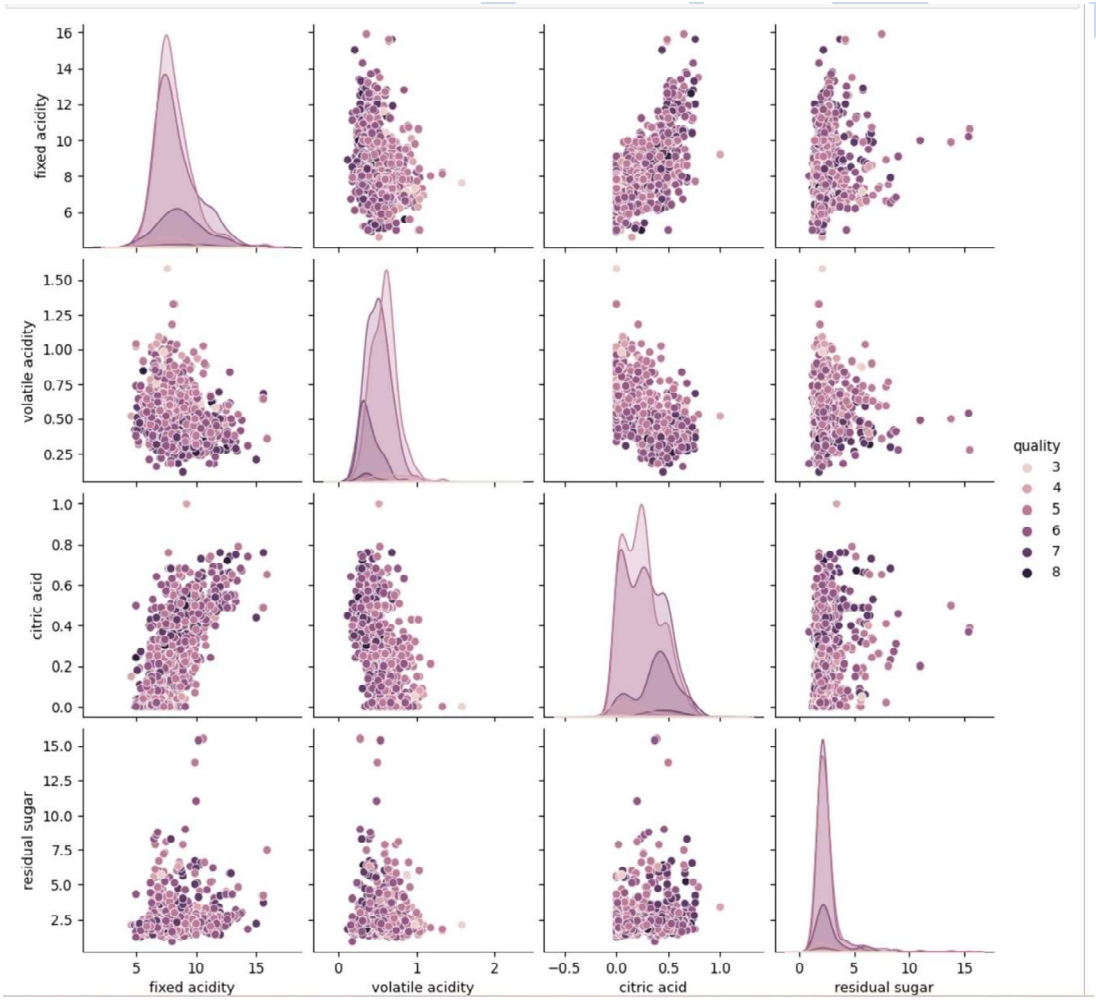

chlorides
free sulfur dioxide
total sulfur dioxide
density

APEC

CCS346 - EDA




```
# Create a pair plot to visualize feature interactions
sns.pairplot(wine_data, vars=['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar'], hue='quality')
plt.show()
```



Result:

Thus the data exploration and visualization to Perform EDA on Wine Quality Data Set. Use a case study

on a data set and apply the various EDA and visualization techniques and present an analysis report was executed and verified successfully.

APEC