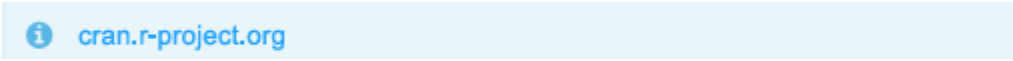
EX.NO: 1 DATE:	Install the data Analysis and Visualization tool: R/python/ Power BI.

Aim:

Procedure:

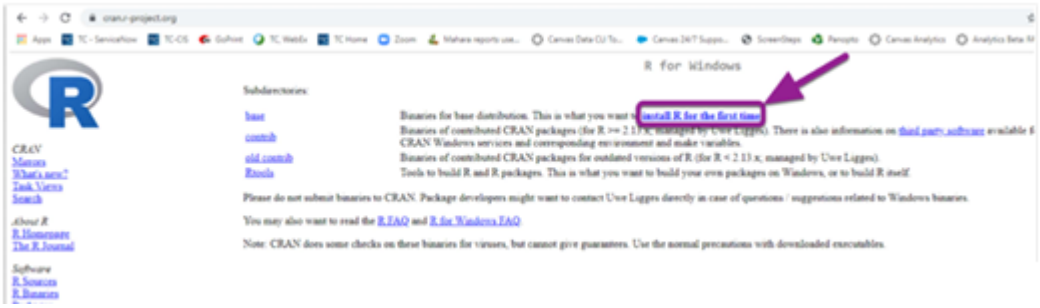
Install R and R-Studio for windows

1.To install R, go to cran.r-project.org



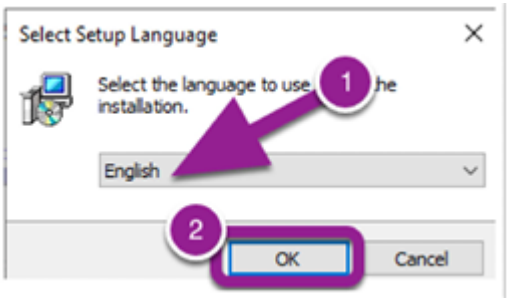
2.Depending on your operating system, click Download R for (your operating system).

3.Click on install R for the first time.

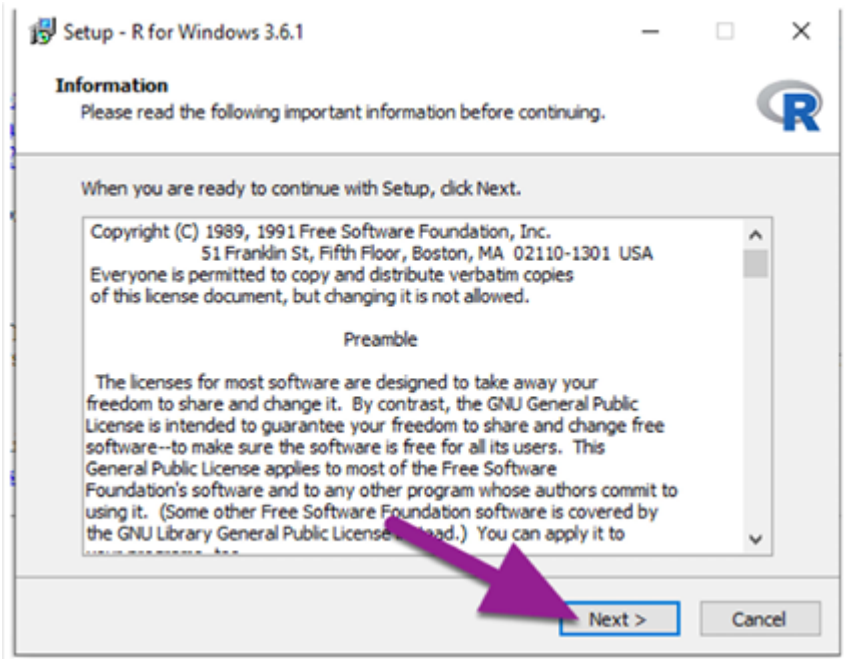


4.Click Download R for Windows. Open the downloaded file.

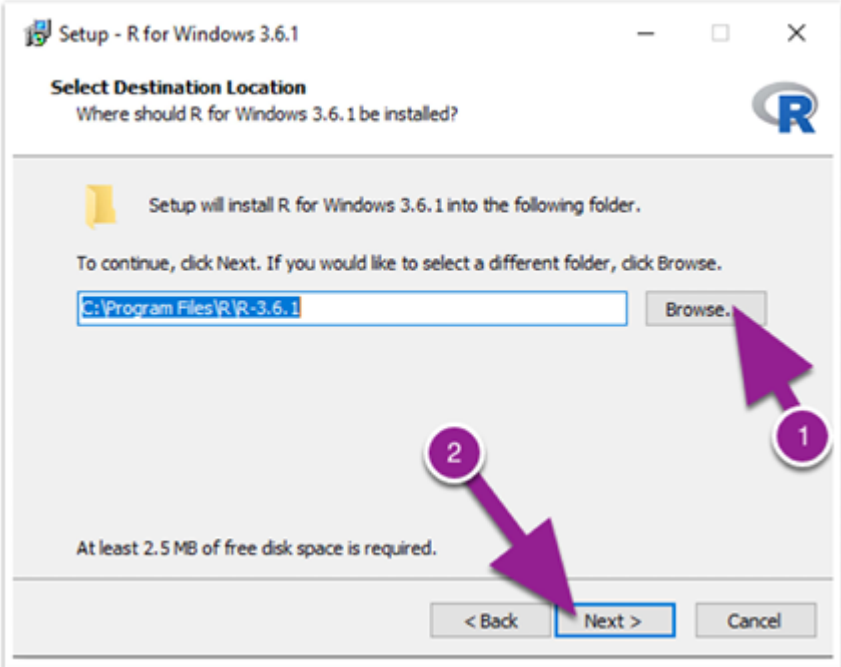
5.Select the language you would like to use during the installation. Then Click Ok.



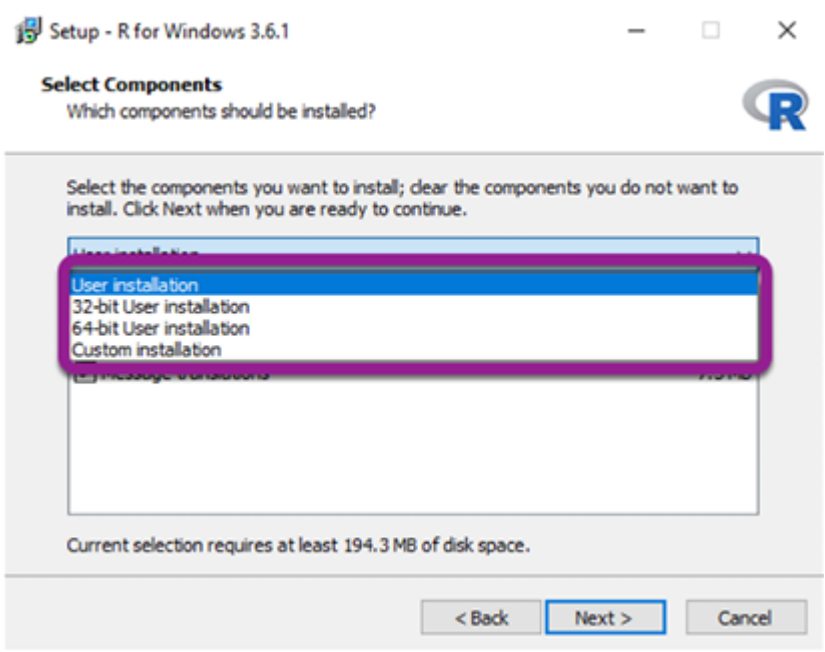
6.Click Next.



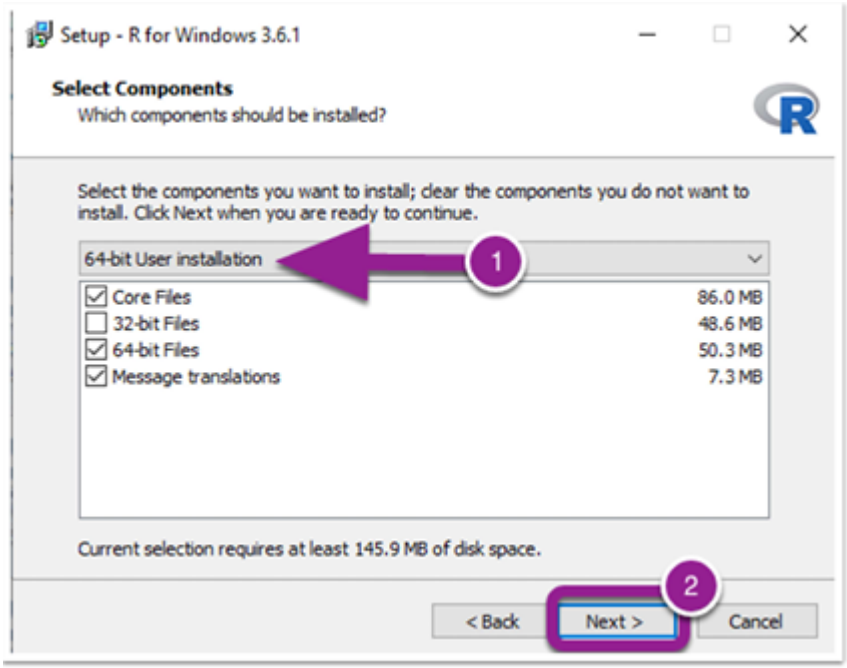
7.Select where you would like R to be installed. It will default to your program Files on your C Drive. Click Next.



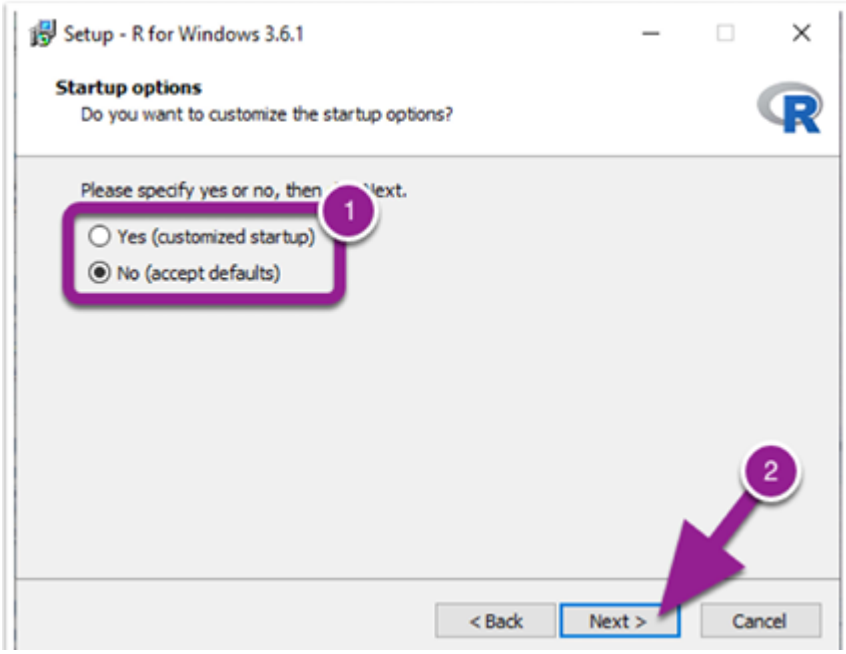
8. You can then choose whuch installation you would like.



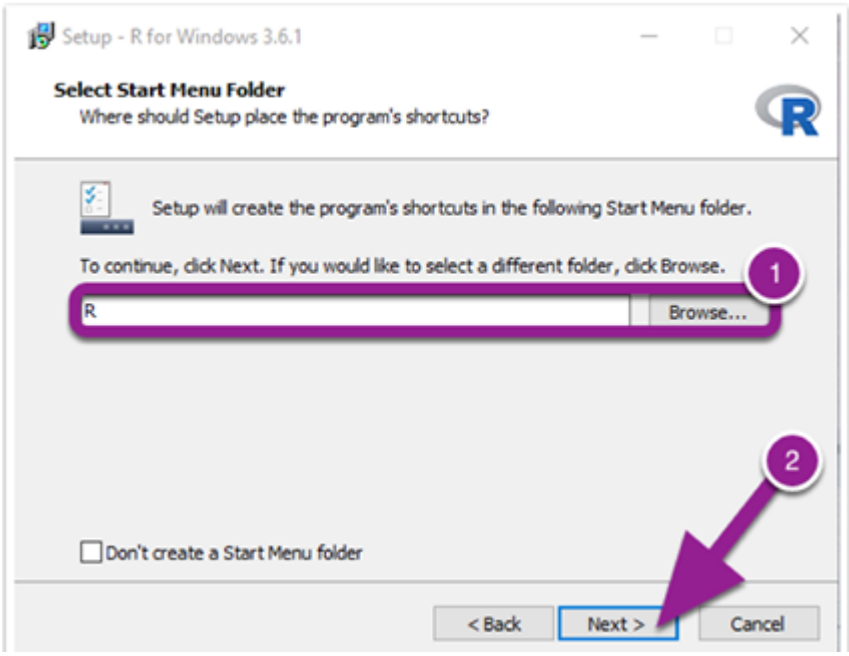
9.(Optional) If your computer is a 64-bit user Installation. Then Click Next.



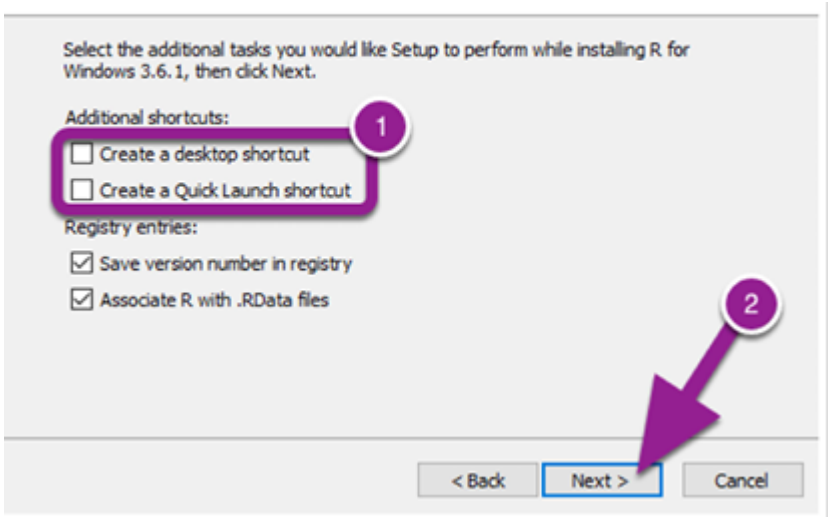
10.Then specify if you want to customized your startup or just use the defaults. Then click Next.



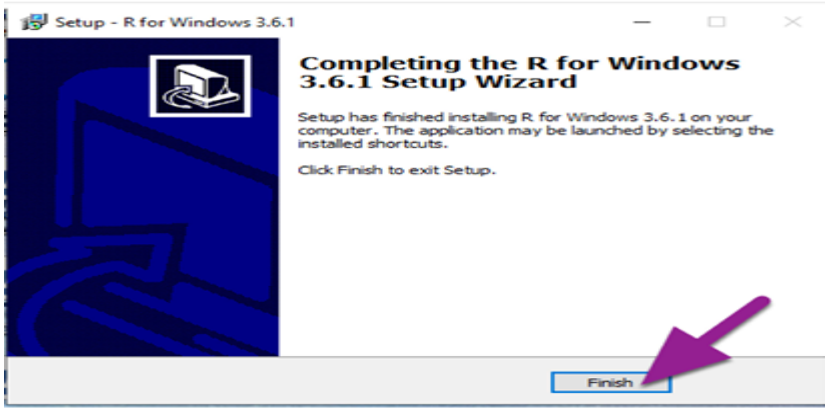
11.Then you can choose the folder that you want R to be saved within or the default if the R folder that was created. Once you have finished, Click Next.



12.You can then select additional shortcuts if you would like. Click Next.



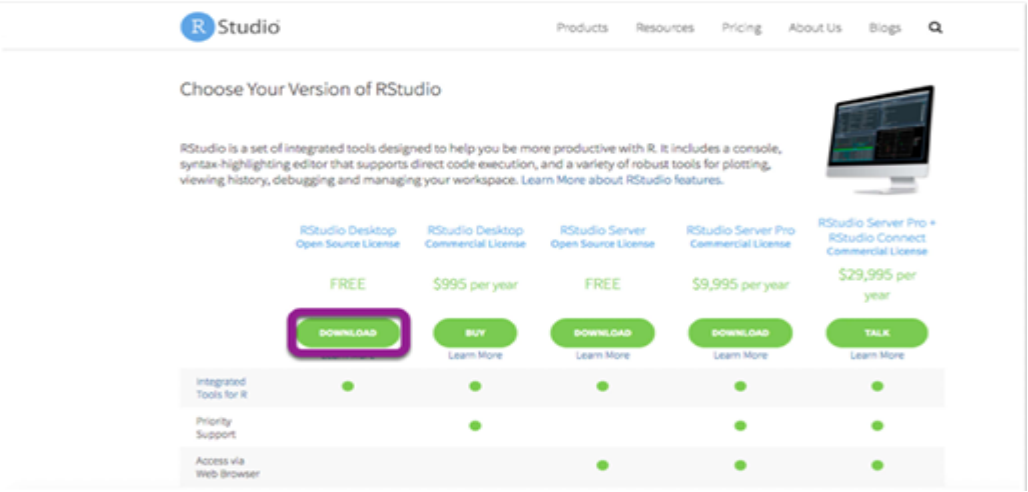
13.Click Finish.



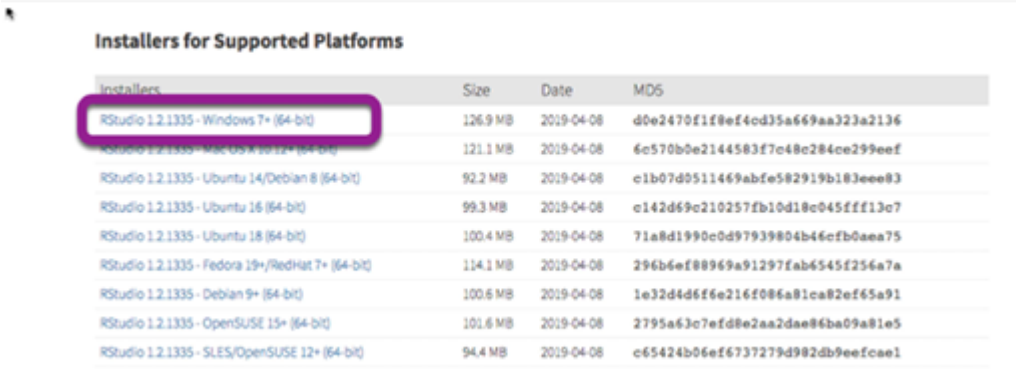
14.Next, download Rstudio. Go to www.rstudio.com

15. Click Download RStudio.

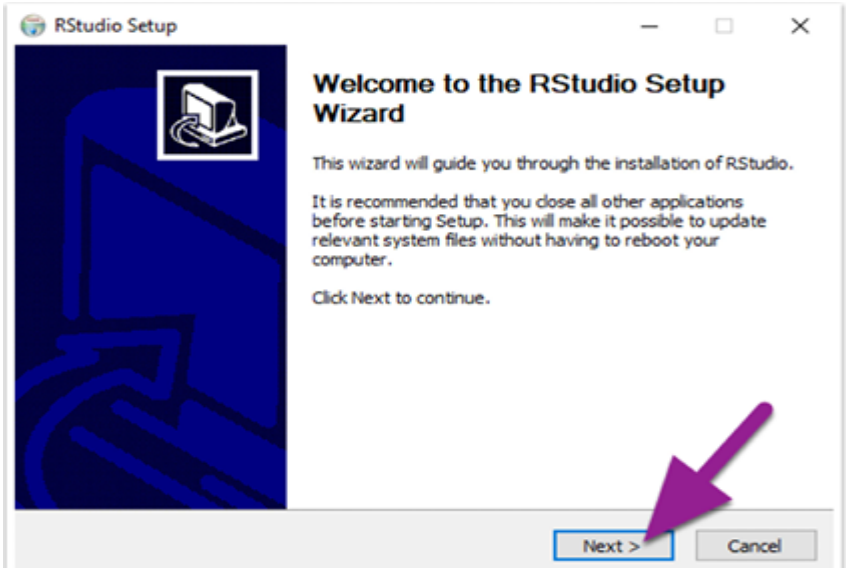
16.Click Download under RstudioDesktop-open source Liscense



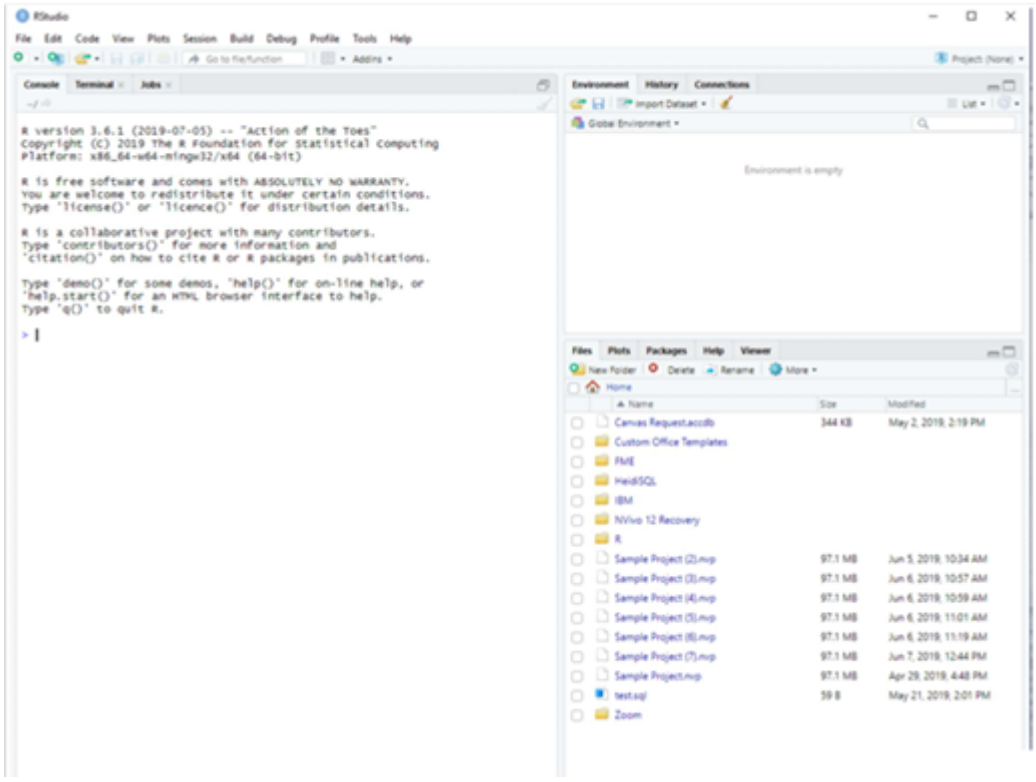
17.Click on the Operating system that you are working with.



18.The RStudio installation wizard will pop-up.
Click Next and go through the installation steps.



19. R and RStudio successfully Installed



R-Studio

Default chapter 12

Installing R and R-Studio 2

Install R and R-Studio for Windows

Install R and R-Studio for Mac

Install From the User Interface

Applies to: Tableau Desktop, Tableau Prep, Tableau Public
The articles in this section describe how to install Tableau Desktop or Tableau Prep Builder from the user interface. Also included are instructions for installing Tableau Desktop Public Edition, which doesn't require activation.

For information about how to activate your product:

- From the command line, see Install From the Command Line
- From the user interface, see Register and Activate from the User Interface (desktop_deploy_activate_license.htm)

Note: If you are upgrading, see Upgrade Tableau Desktop and Tableau Prep Builder (desktop deploy_upgrade.htm) for information about preparing for an upgrade.

During installation, Tableau configures default settings for your display language and repository location. If you want to change those settings you can do this after install is complete. Tableau also enables certain features for you by default such as usage reporting or automated product updates (Tableau Desktop only). For information about how to turn off these features and more, see Change Installation Settings after Installation

Before you begin

- **Installers:** For information about where to find and download the installers for

Tableau Desktop, Tableau Desktop Public Edition, Tableau Prep Builder, and Tableau Reader, see Where's the installer? (desktop_deploy_download.htm)

- **Compatibility with Server:** Tableau products are not always released at the same

time. When installing a new version Tableau Desktop, make sure it is compatible with Tableau Server. See Finding and Resolving Compatibility Issues (desktop_deploy_compatibility.htm).

- **Install Desktop and Prep on the same computer:** Tableau Prep Builder is designed

to work with Tableau Desktop. It is recommended that you install Tableau Prep Builder on the same computer as Tableau Desktop.

- **Do not install Tableau Prep Builder on the same computer running Tableau**

Server: Tableau Server Resource Manager (SRM) can't distinguish between Tableau Server protocol server process and Tableau Prep Builder protocol server process. If the computer resources are exhausted, SRM may terminate the protocol server process belonging to Tableau Prep Builder, which has no recovery mechanism.

- **Windows installer requirement:** If you're using a deployment tool that requires the

Windows installer (.msi file) to install Tableau Desktop or Tableau Prep Builder, follow the instructions in Extract and run the Windows (MSI) installer (desktop_deploy_automate.htm#tmsi) to extract the .msi file from the Tableau installer .exe file.

Install Tableau Desktop

1. As an Administrator, log in to the computer where you are installing Tableau Desktop.
2. Depending on your operating system, do one of the following:
   - For Windows: Run the installer and follow the prompts.
   - For Mac: Open the Disk image file (.DMG) and double-click the installer

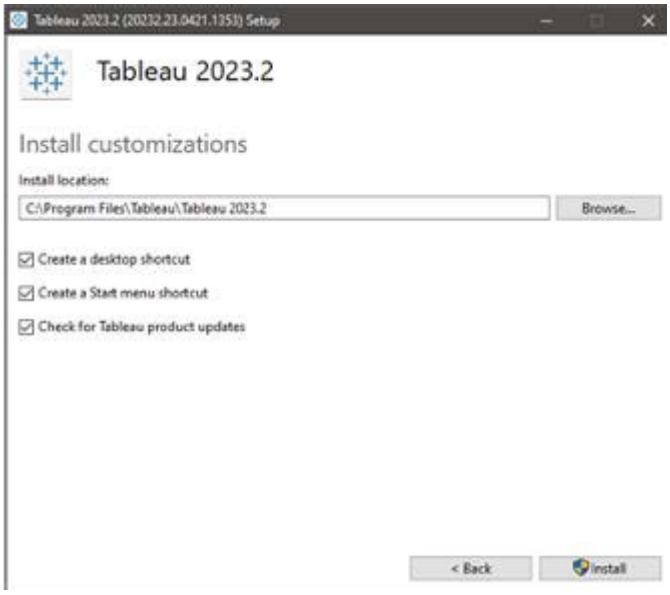
package (.PKG) to start the installation.
Drivers for some data sources are installed automatically when you install Tableau Desktop. See **the Database drivers installed with Tableau Desktop and Tableau Prep Builder** section in the Before You Install (desktop_deploy_intro.htm) topic for specifics.
3. To enable or disable usage reporting complete the following steps for your operating

This option allows us to gather usage pattern data to improve the product. For more information about this option and how to turn it off after installation, see Turn off usage reporting (desktop deploy setting changes.htm#usage). For more information about the type of data we collect, see Tableau Product Usage Data
Windows
· To opt out of providing usage data, select the Don't send product usage data check box.
Mac

- On the Installation Type step, in the bottom-left of the install wizard, click

Customize. To opt out of sending product usage data, select the Don't send product usage data check box.
4. (Optional) On Windows, to customize the install, on the Install welcome screen, click Customize and change any of the following options:

Windows



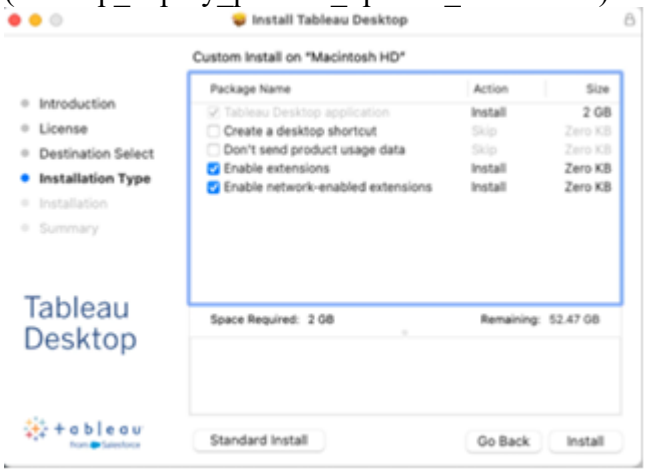
- **Install location:** Specify a different location to install Tableau Desktop.

Important: If you specify a custom directory for the install location and plan to install future releases to this same location, you need to specify a version specific sub-folder to install to. Otherwise you will need to uninstall the previous version first. Side-by-side installs of multiple versions in the same sub-directory is not supported.

o Create a desktop shortcut: Clear the check box if you don't want to automatically create a desktop shortcut for Tableau.

o Create a Start menu shortcut: Clear the check box if you don't want to automatically add a shortcut for Tableau to the Start menu.

o Check for Tableau product updates: Clear the check box if you want to disable the product update feature. This feature checks for maintenance updates and installs them automatically. If you disable this option at install it also disables the menu option for users. For more information about the product update feature, see Control Product Updates for Tableau Desktop (desktop_deploy_product_updates_admin.htm).



o Create a desktop shortcut: Select the check box to automatically create a desktop shortcut for Tableau Prep Builder.
o Don't send product usage data Select the check box to opt out of sending product usage data.
o Enable extensions Clear the check box to opt out of dashboard extensions to expand dashboard functionality with the help of web applications created by Tableau and third-party developers.
o Enable network-enabled extensions Clear the check box to opt out of network-enabled dashboard extensions. Network-enabled dashboard extensions are extensions that run on web servers that can be located inside or outside of your local network.
Note: Starting in version 2019.4.1, only the PostgreSQL driver is installed automatically on the Mac. If you need other database drivers, you can install them from the Driver Download page.
5. **Click Install** to begin installation. If you run into any difficulties, see Troubleshoot

Your Tableau Desktop or Tableau Prep Builder Installation (desktop_deploy_troubleshoot.htm).
6. For Windows environments, if you selected to add a start menu or desktop shortcut, you can start the product from the Start menu or Desktop. To start the product from the executable, go to the install directory (the default is: C:Program Files\Tableau\Tableau <version>\bin) and select tableau.exe.

Install Tableau Prep Builder

1. As an Administrator, log in to the computer where you are installing Tableau Prep Builder.
2. Depending on your operating system, do one of the following:
o For Windows: Run the installer and follow the prompts.
o For Mac: Open the Disk image file (.DMG), and then double-click the installer package (.PKG) to start the installation.
3. When prompted, accept the licensing agreement to continue the installation.
4. To enable or disable usage reporting complete the following steps for your operating system.

This option allows us to gather usage pattern data to improve the product. For more information about this option and how to turn it off after installation, see Turn off usage reporting (desktop deploy_setting_changes.htm#usage). For more information about the type of data we collect, see Tableau Product Usage Data
Windows

- To opt out of providing usage data, select the Don't send product usage data

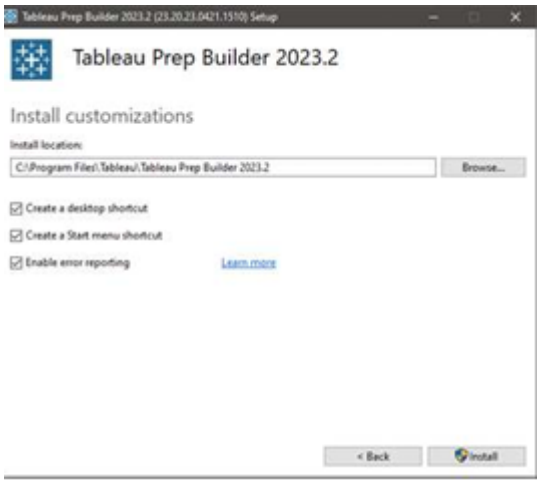
check box.
Mac

- On the Installation Type step, in the bottom-left of the install wizard, click

Customize. To opt out of sending product usage data, select the Don't send product usage data check box.

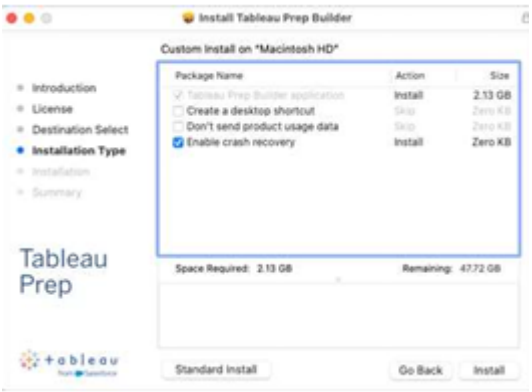
5. (Optional) To customize the **install,** on the Install welcome screen for Windows or on the **Installation Type** step for the Mac, click **Customize** and change any of the following options:

Windows



Install location: Specify a different location to install Tableau Prep Builder.
o Create a desktop shortcut: Clear the check box if you don't want to automatically create a desktop shortcut for Tableau Prep Builder.
o Create a Start menu shortcut: Clear the check box if you don't want to automatically add a shortcut for Tableau Prep Builder to the Start menu
o Enable error reporting : If Tableau Prep Builder has a problem and shuts down unexpectedly, crash dump files and logs are generated and placed in **your My Tableau Prep Builder Repository > Logs and My Tableau Prep Builder Repository > Logs > crashdumps** files.

To turn off this option during install, clear this check box during install. To turn this option off after installation see Turn off error reporting .
Important:Tableau Prep Builder is only available in 64-bit and If you already have 32-bit drivers installed, you'll need to install the 64-bit version of those drivers to connect to your data with Tableau Prep Builder.



o Create a desktop shortcut: Select the check box to automatically create a desktop shortcut for Tableau Prep Builder.
o Don't send product usage data Select the check box to opt out of sending product usage data.

Enable crash recovery (version 2020.3.3 and later): Clear the check box to turn off file recovery. In the event of a crash, flow files won't automatically be saved. For more information about managing this option post-install, see Turn off file recovery (desktop_deploy_setting_changes.htm#autosave_off).
6. Click **Install** to begin the product installation.

7. For Windows environments, if you selected to add a start menu or desktop shortcut, you can start the product from the Start menu or Desktop. To start the product from the executable, go to the install directory (the default is: C:\Program Files\Tableau\Tableau Prep Builder<version> and select Tableau Prep Builder.exe.

Install Tableau Desktop Public Edition

1. As an Administrator, log in to the computer where you are installing Tableau Desktop Public Edition.
2. Depending on your operating system, do one of the following:
o For Windows: Run the installer and follow the prompts.
o For Mac: Open the Disk image file (.DMG), and then double-click the installer

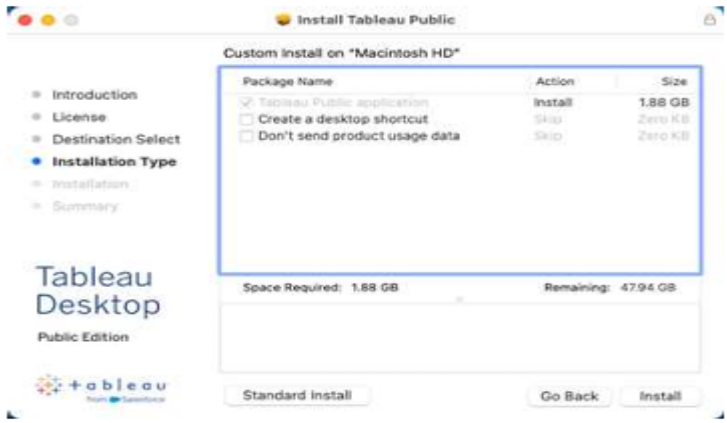
package (.PKG) to start the installation.
3. On the Welcome screen, ensure that the product is the Tableau Desktop Public Edition and accept the licensing agreement to continue the installation.

4. To enable or disable usage reporting complete the following steps for your operating system.
This option allows us to gather usage pattern data to improve the product. For more information about this option and how to turn it off after installation, see Turn off usage reporting (desktop_deploy_setting_changes htm#usage). For more information about the type of data we collect, see Tableau Product Usage Data
Windows
To opt out of providing usage data, select the Don't send product usage data check box.
Mac
On the **Installation Type** step, in the bottom-left of the install wizard, click **Customize**. To opt out of sending product usage data, select the Don't send product usage data check box.
5. (Optional) To customize the install, on the **Install** welcome screen for Windows or on the **Installation Type** step for the Mac, click **Customize** and change any of the following options:

Windows

Install location: Specify a different location to install Tableau Prep Builder.
o Create desktop shortcut: Clear the check box if you don't want to automatically create a desktop shortcut for Tableau Desktop Public Edition.

o Create Start menu shortcut: Clear the check box if you don't want to automatically create a Start menu shorteut for Tableau Desktop Public Edition.
o Check for Tableau product updates: Clear the check box if you want to disable the product update feature. This feature checks for maintenance updates and installs them automatically. If you disable this option at install it also disables the menu option for users.
Mac



o Create a desktop shortcut: Select the check box to automatically create a desktop shortcut for Tableau Prep Builder.
o Don't send product usage data Select the check box to opt out of sending product usage data.
6. Click **Install** to begin the product installation.

Other articles in this section

- Register and Activate from the User Interface

(desktop_deploy_activate_Jicense.htm).

- Using the User Interface Uninstaller (desktop_deploy_uninstall_ui.htm)

In this article, we will guide you on how to download and install Microsoft Power BI Desktop in Windows OS in simple & easy steps.

Before we are proceeding to download the Power BI desktop, you must be aware of the following system requirements.

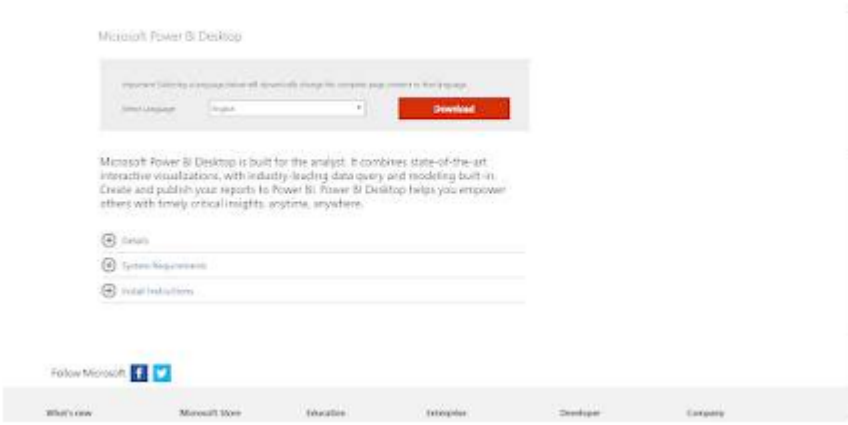
The following are the minimum requirements for installing Power BI Desktop:

- Microsoft Power BI Desktop supports - Windows 7, Windows 8, Windows 8.1, Windows Server 2008 R2, Windows 10, Windows Server 2012, Windows Server 2012 R2.
- Supports both 64-bit(x64) and 32-bit(x86) platforms.
- Requires Internet Explorer10 or a new version of IE(Internet Explorer).
- Requires .Net Framework 4.5.
- Our CPU must have at least 1 gigahertz(GHz) speed.

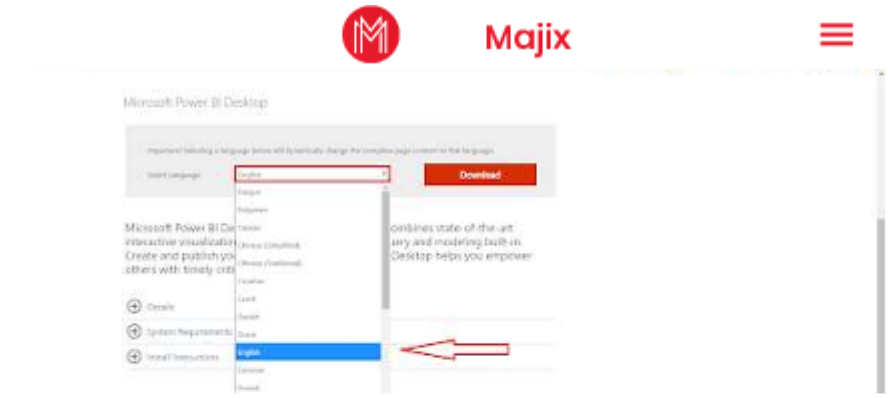
How to Download Power BI Desktop - Step-by-Step Guide

Step 1. To download the Power BI Desktop application, first, go to the Official Power BI Website. The following image shows you a download page from the official website and also the Download or Language Options.

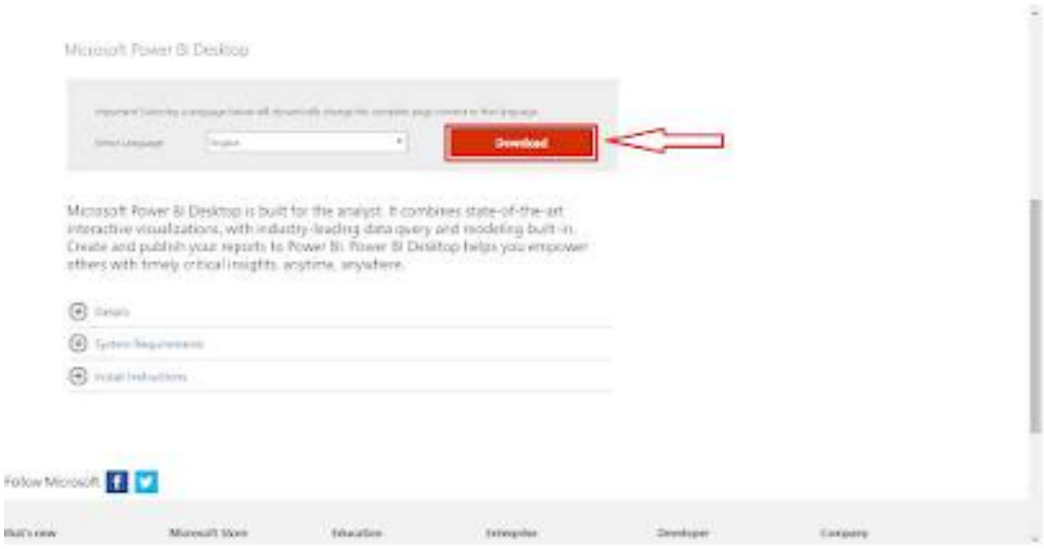
Step 2. Once you dick the Download or Language Options, you will notice the following dropdown box.



Step 3. Next, you can select any one of the languages as per your need in the dropdown box.



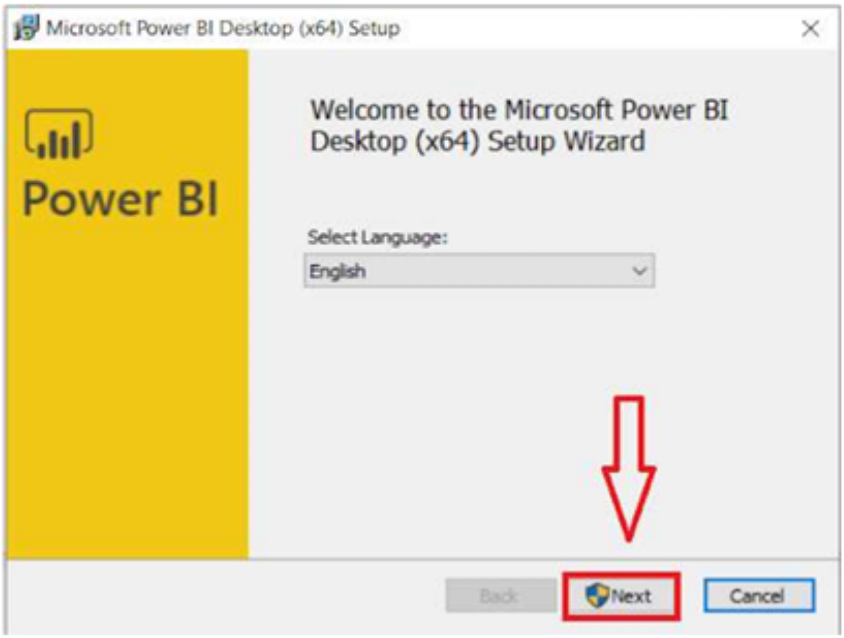
Step 4. Once a language is selected, it will dynamically change the content to that language. Please click the Download button.



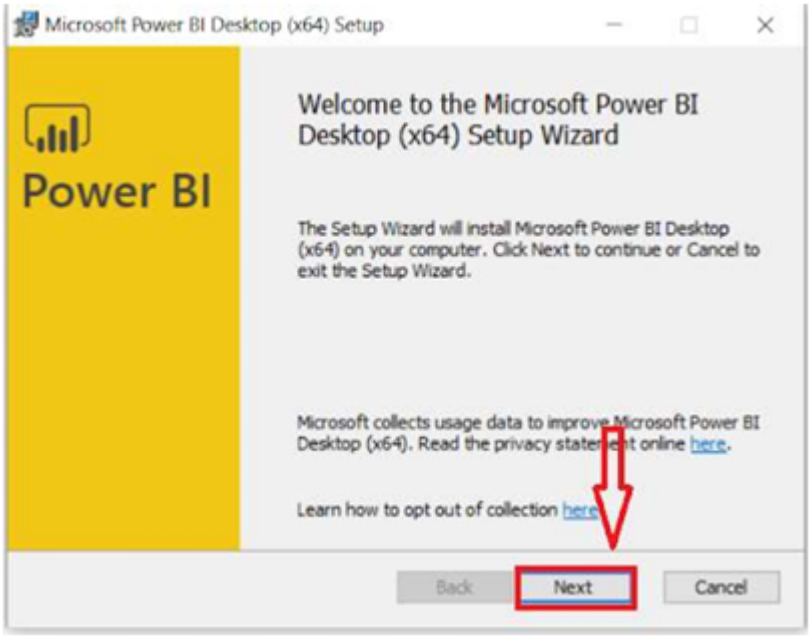
Visit here to know about Power BI Architecture

Step 6. The **Next button** will be enabled once you select either of the **FileNames**. Now, Click the **Next button** to download it on your device.

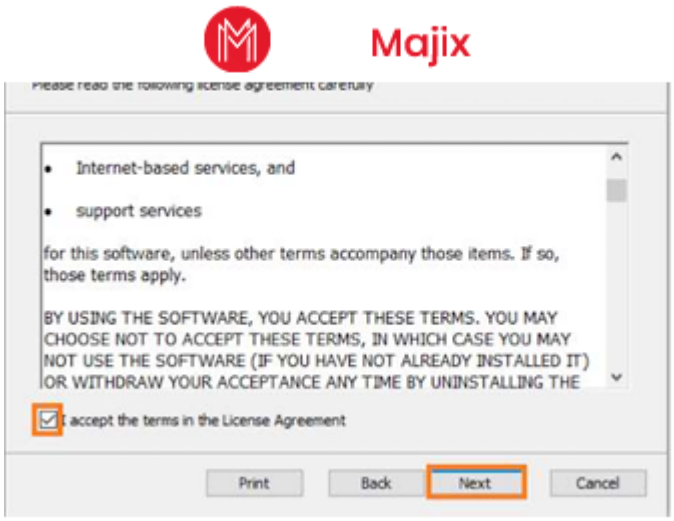
Do you Know- What is Power BI Slicer is?



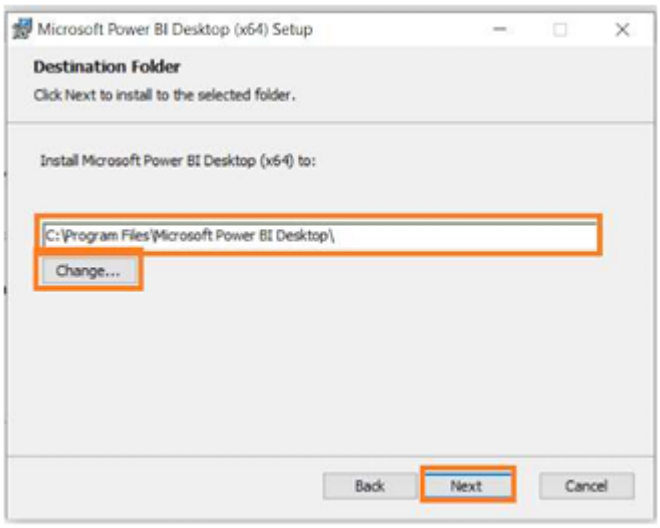
Step 8. Once you click the Next button, you will be asked to click the Next button to continue or the Cancel button to exit in the dialog box.



Step 9. The license agreement dialog box is displayed once you click the Next button. Now the Next button will be enabled after you click the checkbox.



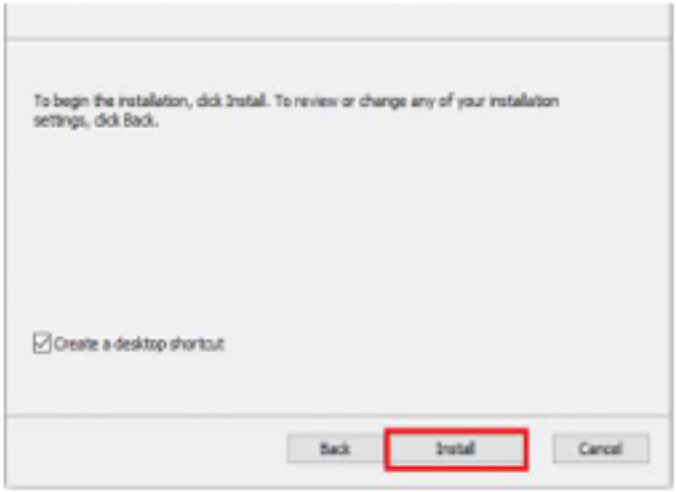
Step 10. Click the Next button to open the Destination folder. The folder allows you to either leave the default C location or use the Change button to alter your desired location for installing Power BI Desktop Application in your device.



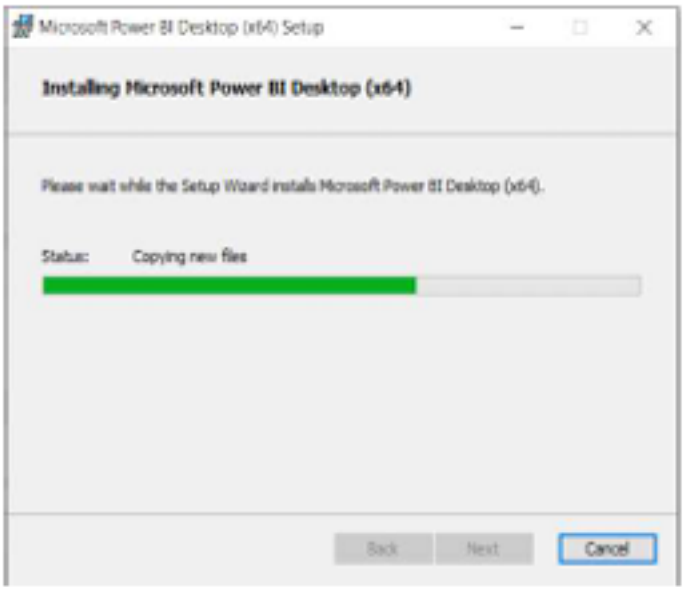
Step 11. Please, click the Next button to give you the following alternatives.

Also Read Power BI Gateway

Step 12. Are you ready to Install? If, yes click the Install button (or) Do you want to review or change any of the installation settings? If yes, click Back Button. Next, click the Install button for installation.

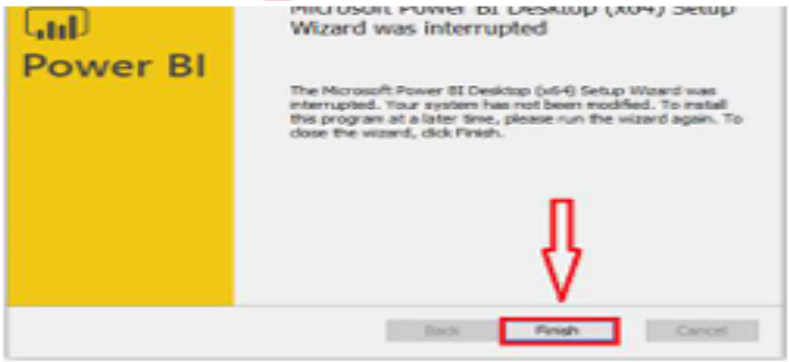


Step 13. Please, wait until the installation is finished.

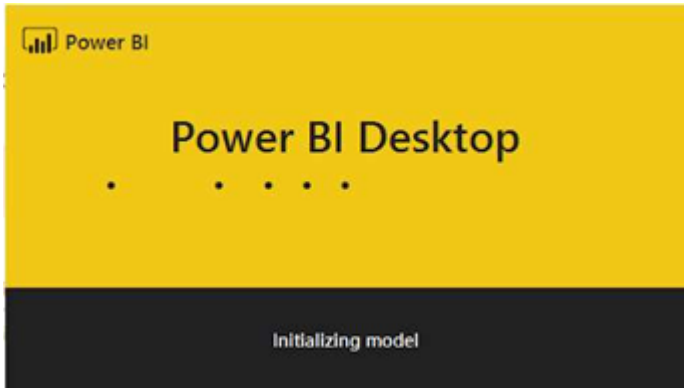


Step 14. Click the **Finish button** to initialize the process.

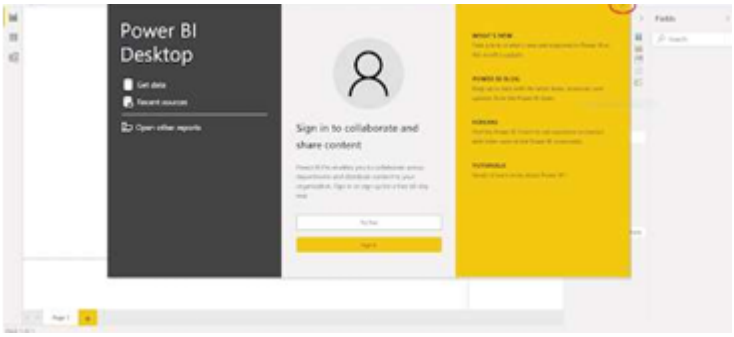
Majix



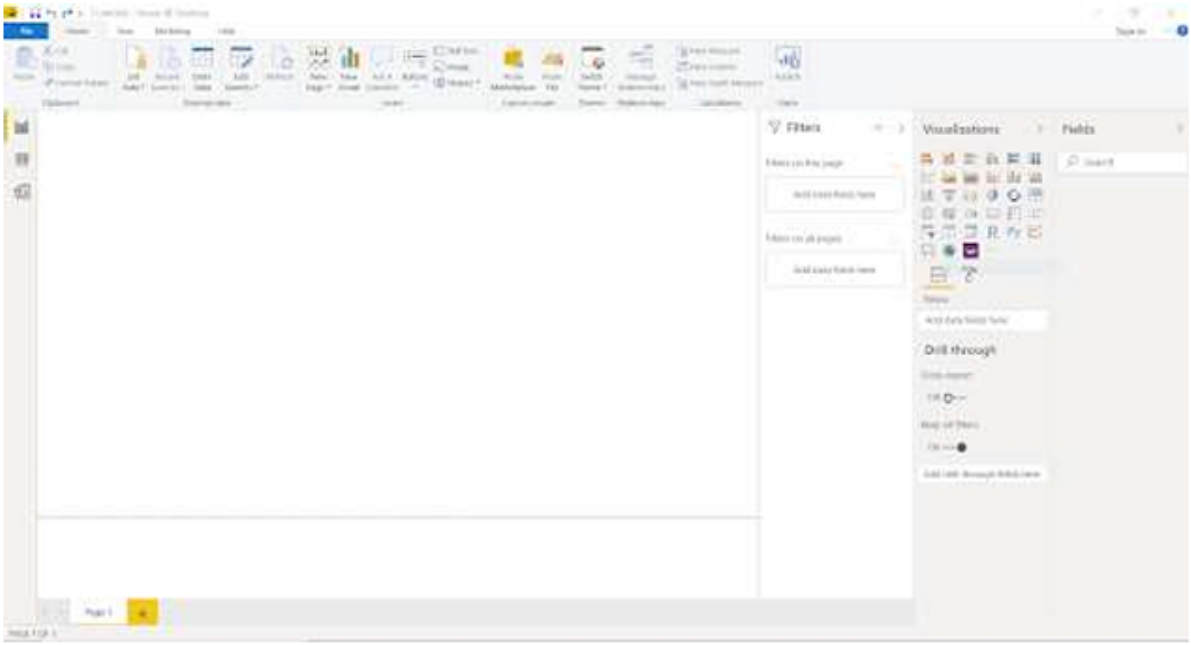
Step 15:Please, wait for a few seconds to start power BI Desktop.



Step 16: Here you can see the installed Power BI Desktop page.



- That's it. Power BI Desktop is now ready for analytics or report development! From there, you can begin creating data models or interactive reports.
- When Power Bl is installed, it generates a welcome screen.
- This screen is used to launch different options related to getting data, enriching the existing data models, creating reports as well as publishing and sharing reports.



So, this was all about installing a Microsoft product Power BI Desktop. Hope you like our explanation.

Final words

Hence, in this guide, we saw how easy it is to install the Power BI Desktop on your omputer. It does not take more than 5 minutes in the best case.

Result:

EX.NO:2 DATE:	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.

Aim:

Algorithm:

Program:

```
import mailbox
import csv
import pandas as pd
import numpy as np
import re
import datetime
import pytz
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
from matplotlib import gridspec
import matplotlib.patches as mpatches
from email.header import decode_header
from scipy import ndimage
from scipy.interpolate import interp1d

# ----------- Step 1: Extract Data from MBOX File ------------
mbox = mailbox.mbox('gmail.mbox')

with open('gmail_cleaned.csv', 'w', newline='', encoding='utf-8') as outputfile:
    writer = csv.writer(outputfile)
    writer.writerow(['subject', 'from', 'date', 'to', 'label', 'thread'])
    for message in mbox:
        writer.writerow([
            message['subject'], message['from'], message['date'], message['to'],
            message['X-Gmail-Labels'], message['X-GM-THRID']
        ])

# ----------- Step 2: Load and Clean Data ------------
dfs = pd.read_csv('gmail_cleaned.csv', skiprows=1,
                  names=['subject', 'from', 'date', 'to', 'label', 'thread'])

# Decode email subjects
def decode_subject(subject):
    try:
        decoded_parts = decode_header(subject)
        return ''.join([
            part.decode(enc or 'utf-8') if isinstance(part, bytes) else part
            for part, enc in decoded_parts
        ])
    except:
        return subject

dfs['subject'] = dfs['subject'].astype(str).apply(decode_subject)

# Convert dates to datetime
dfs['date'] = pd.to_datetime(dfs['date'], errors='coerce', utc=True)
dfs = dfs[dfs['date'].notna()]

# ----------- Step 3: Email Parsing and Labeling ------------
```

```
def extract_email_ID(string):
    email = re.findall(r'<(.+?)>', str(string))
    if not email:
        email = list(filter(lambda y: '@' in y, str(string).split()))
    return email[0] if email else np.nan

dfs['from'] = dfs['from'].apply(extract_email_ID)

myemail = 'itsmeskm99@gmail.com'
dfs['label'] = dfs['from'].apply(lambda x: 'sent' if x == myemail else 'inbox')
dfs.drop(columns='to', inplace=True)

# ----------- Step 4: Timezone Conversion and Features ------------
est = pytz.timezone('US/Eastern')
dfs['date'] = dfs['date'].apply(lambda x: x.astimezone(est))

dfs['dayofweek'] = dfs['date'].dt.day_name()
dfs['dayofweek'] = pd.Categorical(dfs['dayofweek'], categories=[
    'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'
], ordered=True)

dfs['timeofday'] = dfs['date'].dt.hour + dfs['date'].dt.minute / 60 + dfs['date'].dt.second / 3600
dfs['hour'] = dfs['date'].dt.hour
dfs['year_int'] = dfs['date'].dt.year
dfs['year'] = dfs['date'].dt.year + dfs['date'].dt.dayofyear / 365.25
dfs.index = dfs['date']
dfs.drop(columns='date', inplace=True)
print(dfs.tail())

# ----------- Step 5: Plotting Utilities ------------
def plot_todo_vs_year(df, ax, color='C0', s=0.5, title=''):
    df.plot.scatter('year', 'timeofday', s=s, alpha=0.6, ax=ax, color=color)
    ax.set_ylim(0, 24)
    ax.yaxis.set_major_locator(MaxNLocator(8))
    ax.set_yticklabels([
        datetime.datetime.strptime(str(int(np.mod(ts, 24))), "%H").strftime("%I %p")
        for ts in ax.get_yticks()
    ])
    ax.set_title(title)
    ax.grid(ls=':', color='k')
    return ax

def plot_number_perday_per_year(df, ax, label=None, dt=0.3, **plot_kwargs):
    if df.empty or df['year'].dropna().empty:
        return ax
    year = df['year'].dropna().values
    T = year.max() - year.min()
    bins = max(int(T / dt), 1)
    weights = 1 / (np.ones_like(year) * dt * 365.25)
    ax.hist(year, bins=bins, weights=weights, label=label, **plot_kwargs)
```

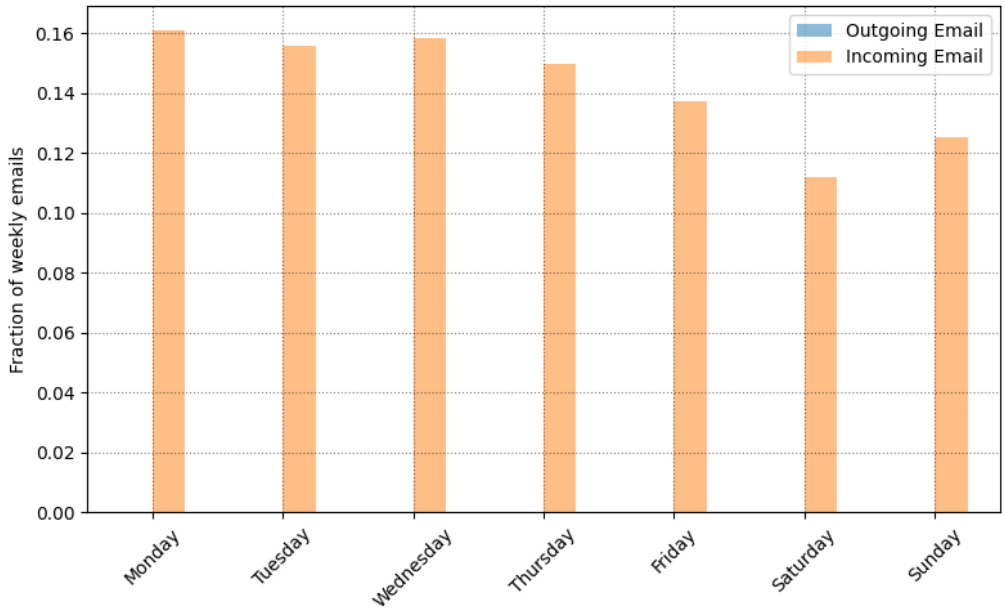
```
    ax.grid(ls=':', color='k')
    return ax


def plot_number_perdhour_per_year(df, ax, label=None, dt=1, smooth=False,
weight_fun=None, **plot_kwargs):
    if df.empty or df['timeofday'].dropna().empty or df['year'].dropna().empty:
        return ax

    tod = df['timeofday'].dropna().values
    year = df['year'].dropna().values

    Ty = year.max() - year.min()
    T = tod.max() - tod.min()
    bins = max(int(T / dt), 1)

    weights = weight_fun(df) if weight_fun else 1 / (np.ones_like(tod) * Ty * 365.25 / dt)

    if smooth:
        if len(tod) < bins:
            return ax  # Not enough data to smooth
        hst, xedges = np.histogram(tod, bins=bins, weights=weights)
        x = np.delete(xedges, -1) + 0.5 * (xedges[1] - xedges[0])
        hst = ndimage.gaussian_filter(hst, sigma=0.75)
        f = interp1d(x, hst, kind='cubic')
        x = np.linspace(x.min(), x.max(), 10000)
        hst = f(x)
        ax.plot(x, hst, label=label, **plot_kwargs)
    else:
        ax.hist(tod, bins=bins, weights=weights, label=label, **plot_kwargs)

    ax.grid(ls=':', color='k')
    orientation = plot_kwargs.get('orientation', 'vertical')
    if orientation == 'vertical':
        ax.set_xlim(0, 24)
        ax.xaxis.set_major_locator(MaxNLocator(8))
        ax.set_xticklabels([
            datetime.datetime.strptime(str(int(np.mod(ts, 24))), "%H").strftime("%I %p")
            for ts in ax.get_xticks()
        ])
    else:
        ax.set_ylim(0, 24)
        ax.yaxis.set_major_locator(MaxNLocator(8))
        ax.set_yticklabels([
            datetime.datetime.strptime(str(int(np.mod(ts, 24))), "%H").strftime("%I %p")
            for ts in ax.get_yticks()
        ])
    return ax
```

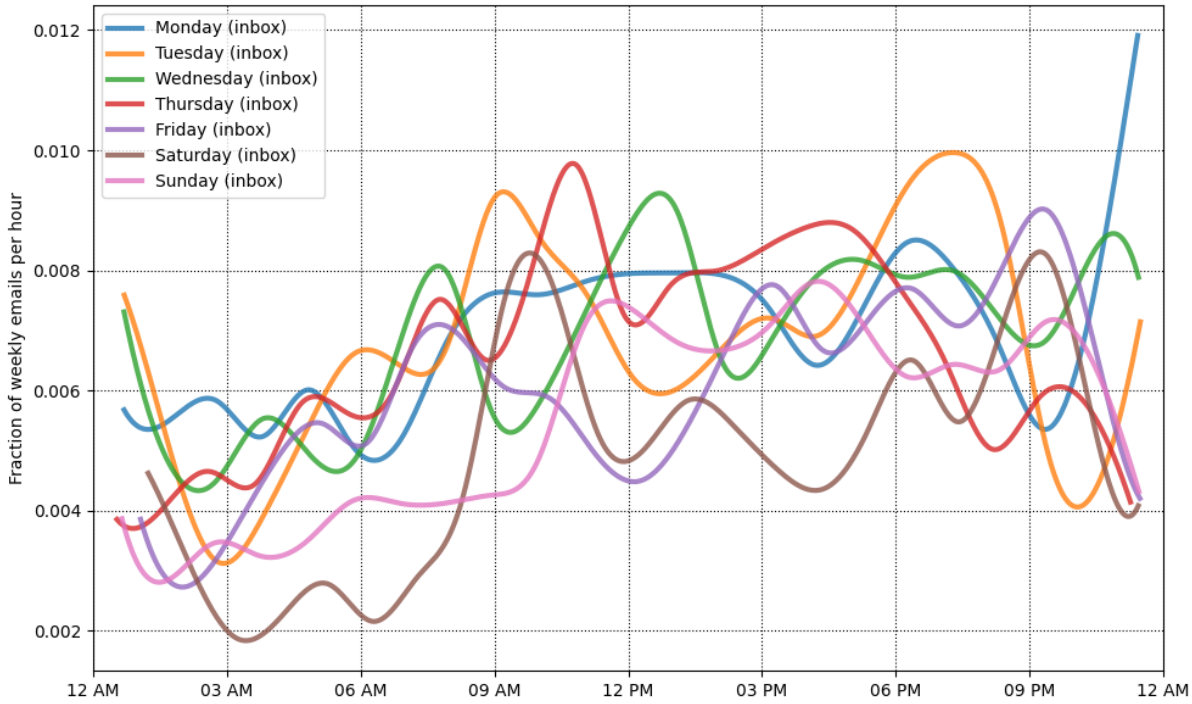
```
class TriplePlot:
    def __init__(self):
        gs = gridspec.GridSpec(6, 6)
        self.ax1 = plt.subplot(gs[2:6, :4])
        self.ax2 = plt.subplot(gs[2:6, 4:6], sharey=self.ax1)
        plt.setp(self.ax2.get_yticklabels(), visible=False)
        self.ax3 = plt.subplot(gs[:2, :4])
        plt.setp(self.ax3.get_xticklabels(), visible=False)

    def plot(self, df, color='darkblue', alpha=0.8, markersize=0.5, yr_bin=0.1, hr_bin=0.5):
        plot_todo_vs_year(df, self.ax1, color=color, s=markersize)
        plot_number_perdhour_per_year(df, self.ax2, dt=hr_bin, color=color, alpha=alpha,
orientation='horizontal')
        self.ax2.set_xlabel('Average emails per hour')
        plot_number_perday_per_year(df, self.ax3, dt=yr_bin, color=color, alpha=alpha)
        self.ax3.set_ylabel('Average emails per day')

# ----------- Step 6: Visualizations ------------
sent = dfs[dfs['label'] == 'sent']
received = dfs[dfs['label'] == 'inbox']

# Scatter Plot for Sent vs Received
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(15, 4))
plot_todo_vs_year(sent, ax[0], title='Sent')
plot_todo_vs_year(received, ax[1], title='Received')
plt.tight_layout()
plt.show()

# Weekly Distribution
sdw = sent.groupby('dayofweek').size() / len(sent)
rdw = received.groupby('dayofweek').size() / len(received)
df_tmp = pd.DataFrame(data={'Outgoing Email': sdw, 'Incoming Email': rdw})
df_tmp.plot(kind='bar', rot=45, figsize=(8, 5), alpha=0.5)
plt.xlabel('')
plt.ylabel('Fraction of weekly emails')
plt.grid(ls=':', color='k', alpha=0.5)
plt.tight_layout()
plt.show()

# Smoothed per-hour email fraction by weekday
plt.figure(figsize=(10, 6))
ax = plt.subplot(111)
for ct, dow in enumerate(dfs.dayofweek.cat.categories):
    df_r = received[received['dayofweek'] == dow]
    df_s = sent[sent['dayofweek'] == dow]

    r_weights = lambda df: np.ones(len(df)) / len(received)
    s_weights = lambda df: np.ones(len(df)) / len(sent)
```

```
    plot_number_perdhour_per_year(df_r, ax, dt=1, smooth=True, color=f'C{ct}', alpha=0.8,
lw=3, label=f"{dow} (inbox)", weight_fun=r_weights)
    plot_number_perdhour_per_year(df_s, ax, dt=1, smooth=True, color=f'C{ct}', alpha=0.8,
lw=2, label=f"{dow} (sent)", ls='--', weight_fun=s_weights)

ax.set_ylabel('Fraction of weekly emails per hour')
plt.legend(loc='upper left')
plt.tight_layout()
plt.show()
```

Output:

```
                                                      subject  \
date
2007-08-04 13:53:21-04:00                          Tax Refund!
2007-08-04 15:23:44-04:00  your paypal account need to be updated!
2007-08-04 17:08:37-04:00                Message from eBay member
2007-08-05 13:13:25-04:00  your paypal account need to be updated!
2007-08-06 15:21:51-04:00              Update your PayPal records

                                                        from  label  \
date
2007-08-04 13:53:21-04:00                       refund@irs.gov  inbox
2007-08-04 15:23:44-04:00            online-service@paypalini.com  inbox
2007-08-04 17:08:37-04:00  eBay_sellers_messages@eBay_memmber.com  inbox
2007-08-05 13:13:25-04:00             serviceconfirms@pay-pal.com  inbox
2007-08-06 15:21:51-04:00                  service@paypal.com  inbox

                           thread dayofweek  timeofday  hour  year_int  \
date
2007-08-04 13:53:21-04:00     NaN  Saturday  13.889167    13       2007
2007-08-04 15:23:44-04:00     NaN  Saturday  15.395556    15       2007
2007-08-04 17:08:37-04:00     NaN  Saturday  17.143611    17       2007
2007-08-05 13:13:25-04:00     NaN    Sunday  13.223611    13       2007
2007-08-06 15:21:51-04:00     NaN    Monday  15.364167    15       2007
```




Monday (inbox)
Tuesday (inbox)
Wednesday (inbox)
Thursday (inbox)
Friday (inbox)
Saturday (inbox)
Sunday (inbox)
Fraction of weekly emails per hour
0.012
0.010
0.008
0.006
0.004
0.002
12 AM
03 AM
06 AM
09 AM
12 PM
03 PM
06 PM
09 PM
12 AM

Result:

EX.NO:3 DATE:	Working with Nupy arrays, Pandas data frames, Basic plots using Matplotlib.Numpy arrays using matplotlib

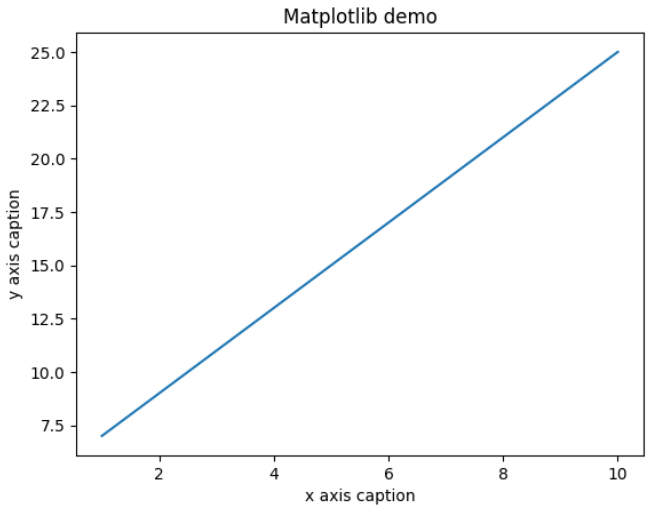
Aim:

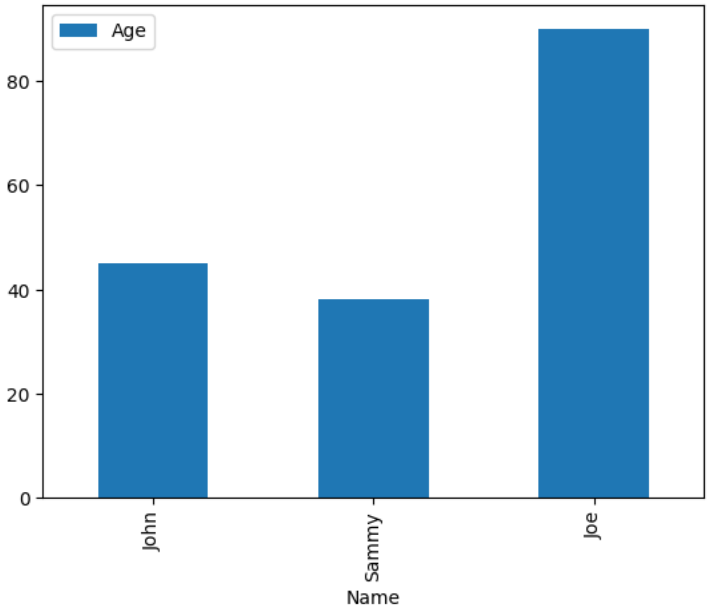
Algorithm:

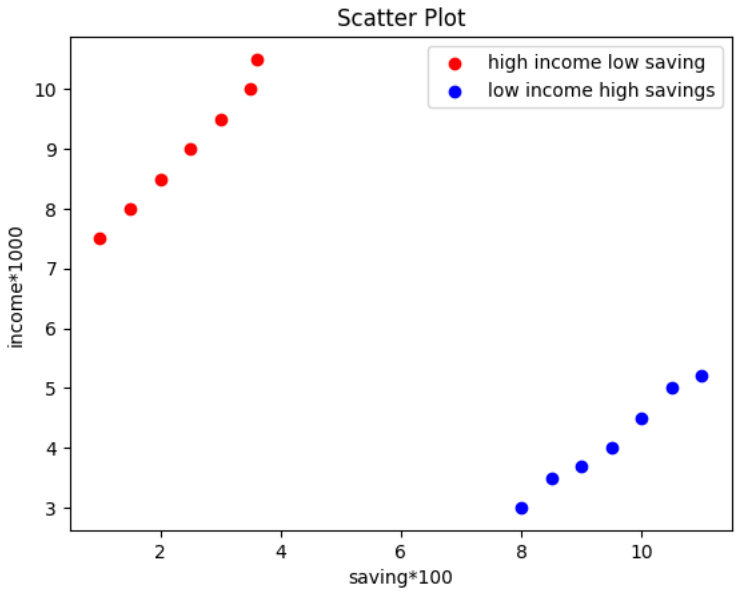
Program:

```
#Working with Numpy arrays
import numpy as np
from matplotlib import pyplot as plt
x = np.arange(1,11)
y = 2 * x + 5
plt.title("Matplotlib demo")
plt.xlabel("x axis caption")
plt.ylabel("y axis caption")
plt.plot(x,y)
plt.show()
#Pandas dataframe using matplotlib
import pandas as pd
import matplotlib.pyplot as plt
df = pd.DataFrame({'Name': ['John', 'Sammy', 'Joe'],'Age': [45, 38, 90]})
df.plot(x="Name", y="Age", kind="bar")
#Basic plot
import matplotlib.pyplot as plt
x = [1,1.5,2,2.5,3,3.5,3.6]
y = [7.5,8,8.5,9,9.5,10,10.5]
x1=[8,8.5,9,9.5,10,10.5,11]
y1=[3,3.5,3.7,4,4.5,5,5.2]
plt.scatter(x,y, label='high income low saving',color='r')
plt.scatter(x1,y1,label='low income high savings',color='b')
plt.xlabel('saving*100')
plt.ylabel('income*1000')
plt.title('Scatter Plot')
plt.legend()
plt.show()
```

Output:




Age
80
60
40
20
0
John
Sammy
Joe
Name


Scatter Plot
high income low saving
low income high savings
income*1000
saving*100
10
9
8
7
6
5
4
3
2
4
6
8
10

Result:

EX.NO:4 **DATE:**	**Explore various variable and row filters in python for cleaning data** **Apply various plot features in python on sample data sets and visualize**

Aim:

Algorithm:

Program:

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print (df)
```

Output:

```
        one       two     three
a -1.596587  0.762620 -0.453494
b       NaN       NaN       NaN
c  0.536723 -0.241202 -0.455903
d       NaN       NaN       NaN
e  0.937639  0.648610 -1.148315
f  0.189710  0.305573  0.643276
g       NaN       NaN       NaN
h -0.214411  0.625403 -1.118437
```

Program:

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print (df['one'].isnull())
```

Output:

```
a    False
b     True
c    False
d     True
e    False
f    False
g     True
h    False
Name: one, dtype: bool
```

Program:

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(3, 3), index=['a', 'c', 'e'],columns=['one',
```

```
'two', 'three'])
df = df.reindex(['a', 'b', 'c'])
print (df)
print ("NaN replaced with '0':")
print (df.fillna(0))
```

Output:

```
         one       two     three
a -0.515503 -1.897424  2.257559
b       NaN       NaN       NaN
c -1.452117 -1.088794 -0.236839
NaN replaced with '0':
         one       two     three
a -0.515503 -1.897424  2.257559
b  0.000000  0.000000  0.000000
c -1.452117 -1.088794 -0.236839
```

Program:

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',
'h'],columns=['one', 'two', 'three'])
df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])
print (df.dropna())
```

Output:

```
        one       two     three
a  1.968301 -0.993507  0.462340
c -0.284187 -1.232066  1.500748
e  0.879697 -0.827373  0.548273
f -0.596764 -2.482547  1.312961
h -0.113099  1.777143 -0.505040
```

Program:

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'one':[10,20,30,40,50,2000],
'two':[1000,0,30,40,50,60]})
print (df.replace({1000:10,2000:60}))
```

Output:

```
   one  two
0   10   10
1   20    0
2   30   30
3   40   40
4   50   50
5   60   60
```

Result:

EX.NO: 5 DATE:	Perform Time Series Analysis and apply the various visualization techniques.

Aim:

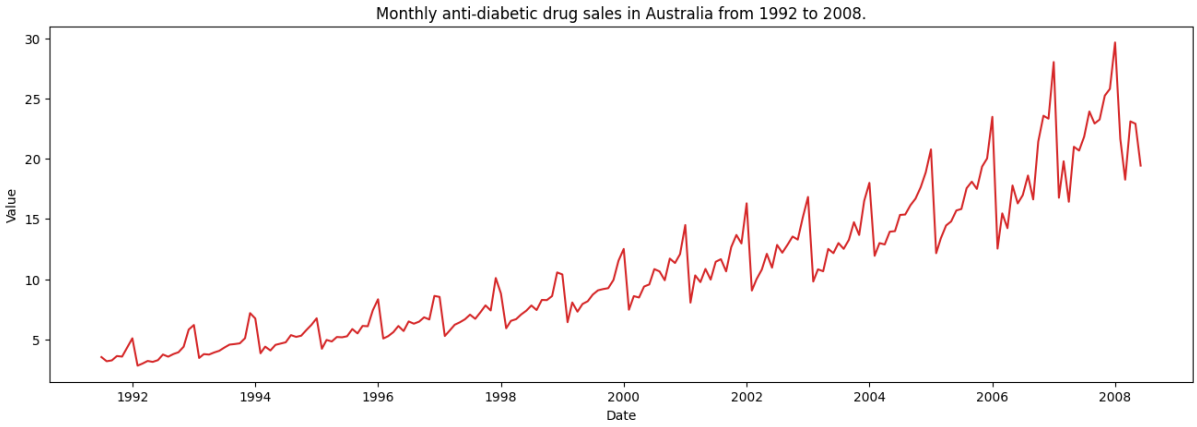
Algorithm:

Program:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
plt.rcParams.update({'figure.figsize': (10, 7), 'figure.dpi': 120}) #
df=pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv',
parse_dates=['date'])
df.head()

import matplotlib.pyplot as plt
df=pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv',
parse_dates=['date'], index_col='date')
# Draw Plot
def plot_df(df, x, y, title="", xlabel='Date', ylabel='Value', dpi=100):
    plt.figure(figsize=(16,5), dpi=dpi)
    plt.plot(x, y, color='tab:red')
    plt.gca().set(title=title, xlabel=xlabel, ylabel=ylabel)
    plt.show()
plot_df(df, x=df.index, y=df.value, title='Monthly anti-diabetic drug sales in Australia from
1992 to 2008.')
```

Output:

	date	value
0	1991-07-01	3.526591
1	1991-08-01	3.180891
2	1991-09-01	3.252221
3	1991-10-01	3.611003
4	1991-11-01	3.565869


Monthly anti-diabetic drug sales in Australia from 1992 to 2008.
Value
Date
30
25
20
15
10
5
1992
1994
1996
1998
2000
2002
2004
2006
2008

Result:

EX.NO: 6 DATE:	Perform Data Analysis and representation on a Map using various Map data sets with Mouse Rollover effect, user interaction, etc..

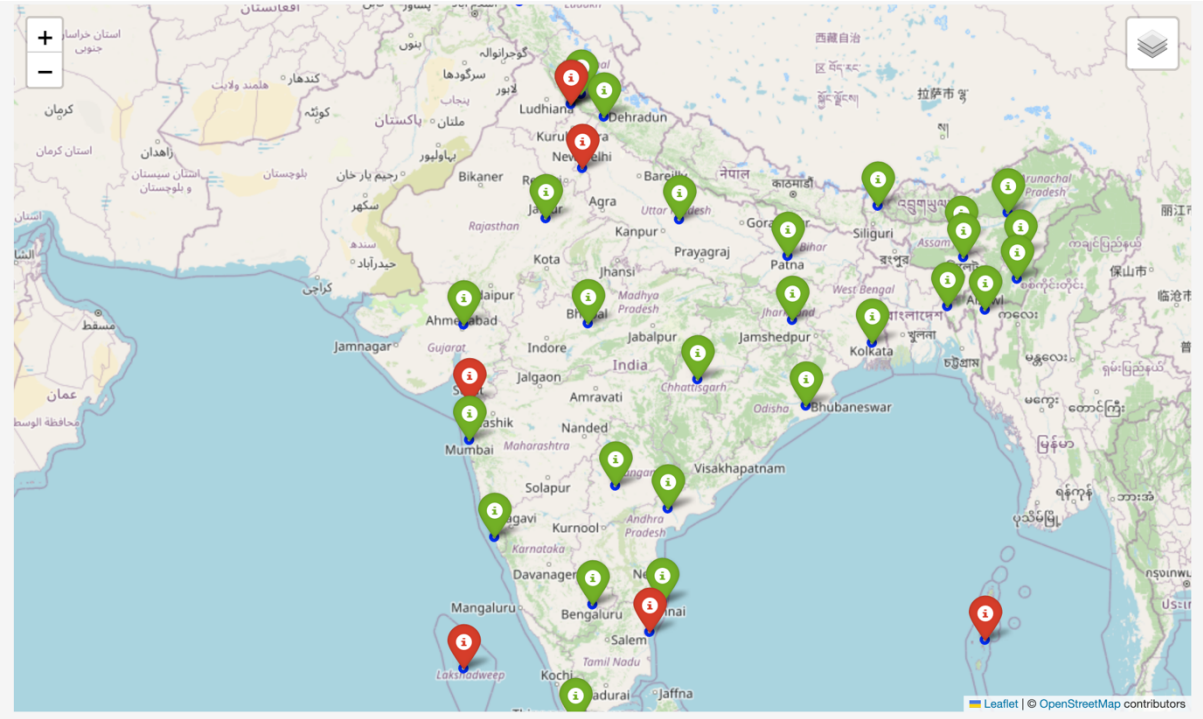
Aim:

Algorithm:

Program:

```
import folium
import pandas as pd

# Load dataset
data = pd.read_csv("LatLong.csv")

# Center map over India
m = folium.Map(location=[22.0, 78.0], zoom_start=5)

# Add markers
for _, row in data.iterrows():
    popup = f"<b>{row['capital']}</b><br>{row['type']}: {row['region']}"
    folium.Marker(
        location=[row['lat'], row['lon']],
        popup=folium.Popup(popup, max_width=200),
        icon=folium.Icon(color='green' if row['type']=='State' else 'red', icon='info-sign')
    ).add_to(m)

# Add hover tooltip
for _, row in data.iterrows():
    folium.CircleMarker(
        location=[row['lat'], row['lon']],
        radius=3,
        color='blue',
        fill=True,
        fill_opacity=0,
        tooltip=f"{row['capital']} ({row['region']})"
    ).add_to(m)

# Layer toggle
folium.LayerControl().add_to(m)

# Save and show
m
```

Output:



Result:

EX.NO: 7 DATE:	Build cartographic visualization for multiple datasets involving various countries of the world states and districts in India etc.

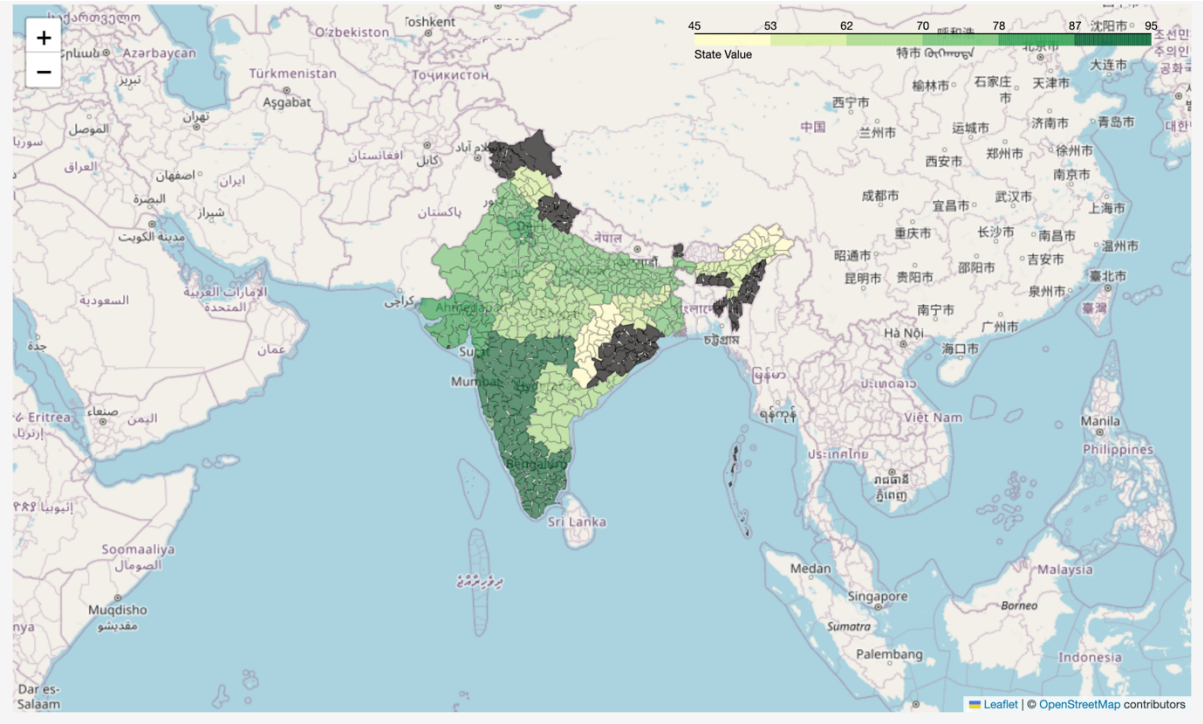
Aim:

Algorithm:

Program:

```
import folium
import pandas as pd
data=pd.read_csv('ex7_dataset.csv')
m=folium.Map(location=[20.5937, 78.9629], zoom_start=5)
folium.Choropleth(
    geo_data='india_district.geojson',
    name='choropleth',
    data=data,
    columns=['State', 'Value'],
    key_on ='feature.properties.NAME_1',
    fill_color='YlGn',
    fill_opacity=0.7,
    line_opacity=0.2,
    legend_name='State Value'
).add_to(m)
m.save('india_map.html')
```

Output:



Result:

EX.NO: 8 DATE:	Perform EDA on Wine Quality Data Set

Aim:

Algorithm:

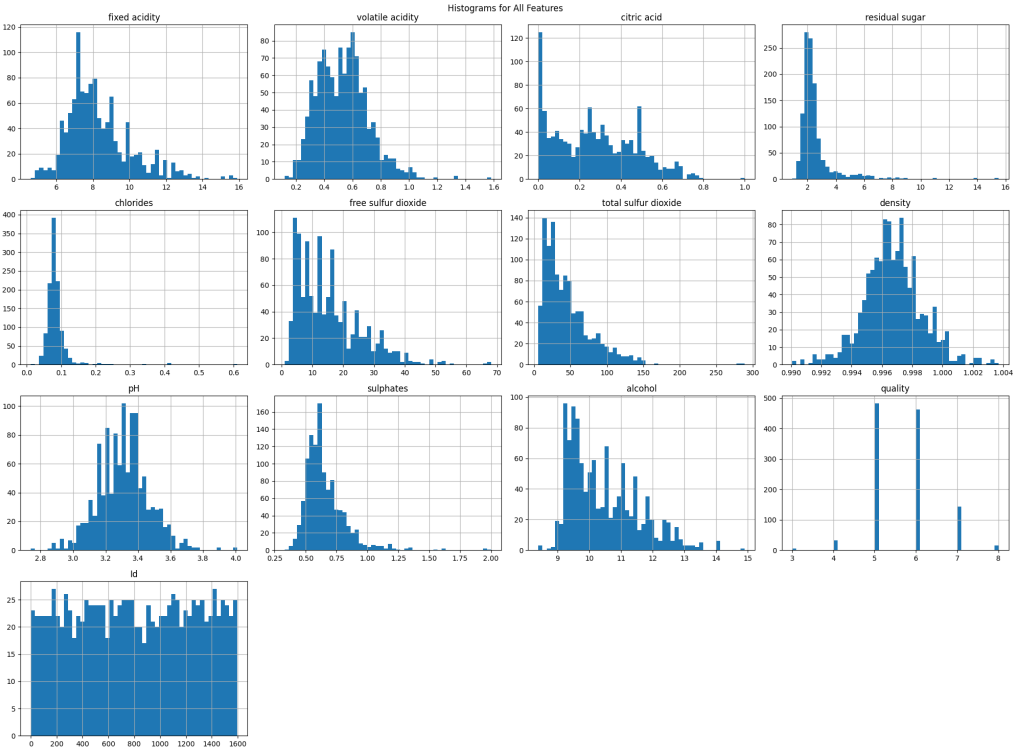
Program:

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

# Load dataset
df = pd.read_csv('ex8_dataset.csv')

# Show basic info
print("First 5 rows of the dataset:")
print(df.head())

print("\nSummary statistics:")
print(df.describe())

print("\nMissing values per column:")
print(df.isnull().sum())

# Histograms
print("\nGenerating histograms...")
df.hist(bins=50, figsize=(20, 15))
plt.suptitle("Histograms for All Features")
plt.tight_layout()
plt.savefig("histograms.png")
plt.show()

# Scatter matrix
print("\nGenerating scatter matrix...")
pd.plotting.scatter_matrix(df, figsize=(20, 20))
plt.suptitle("Scatter Matrix of Features")
plt.savefig("scatter_matrix.png")
plt.show()

# Correlation matrix and heatmap
print("\nGenerating correlation heatmap...")
correlation_matrix = df.corr(numeric_only=True)
plt.figure(figsize=(12, 10))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.savefig("correlation_heatmap.png")
plt.show()

# Box plots for numerical columns
print("\nGenerating box plots for outlier detection...")
numeric_cols = df.select_dtypes(include=['int64', 'float64']).columns
for col in numeric_cols:
    plt.figure(figsize=(8, 4))
    sns.boxplot(data=df, x=col)
    plt.title(f"Box Plot of {col}")
```

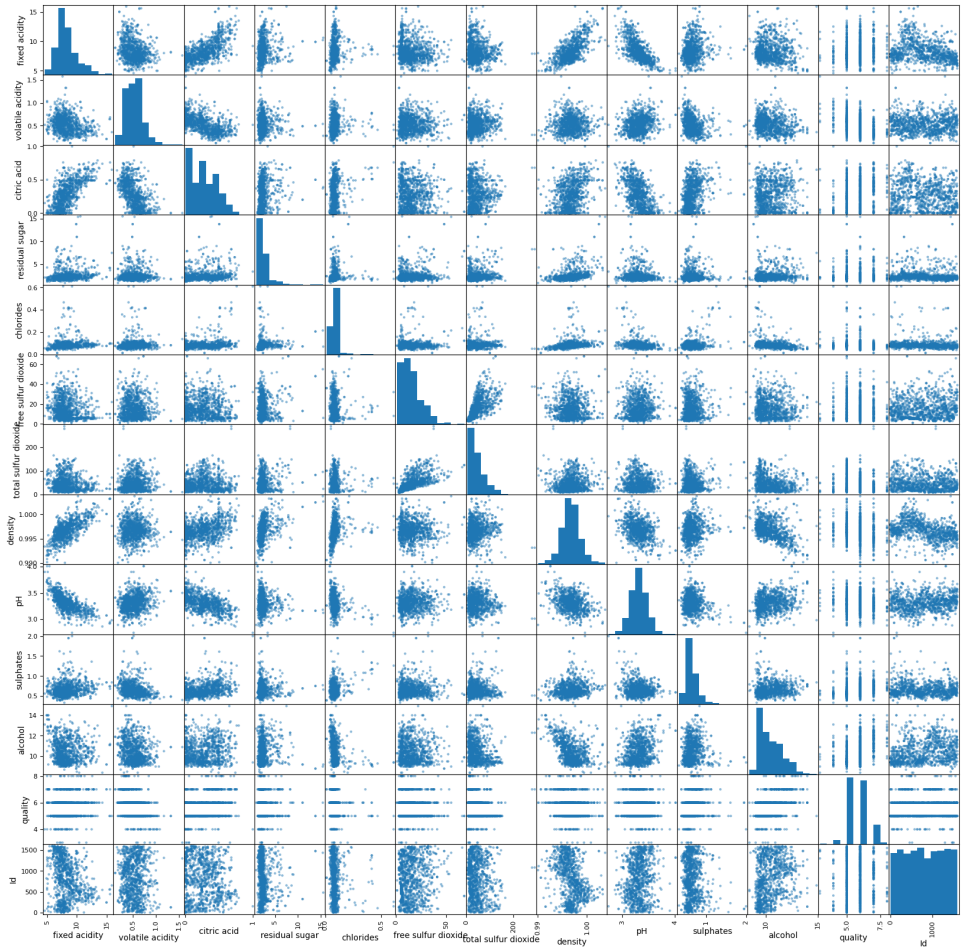
```
    plt.savefig(f"boxplot_{col}.png")
    plt.show()

# Value counts for categorical columns
print("\nValue counts for categorical columns:")
categorical_cols = df.select_dtypes(include=['object', 'category']).columns
for col in categorical_cols:
    print(f"\nColumn: {col}")
    print(df[col].value_counts())

print("\nAnalysis complete. Plots saved as PNG files.")
```

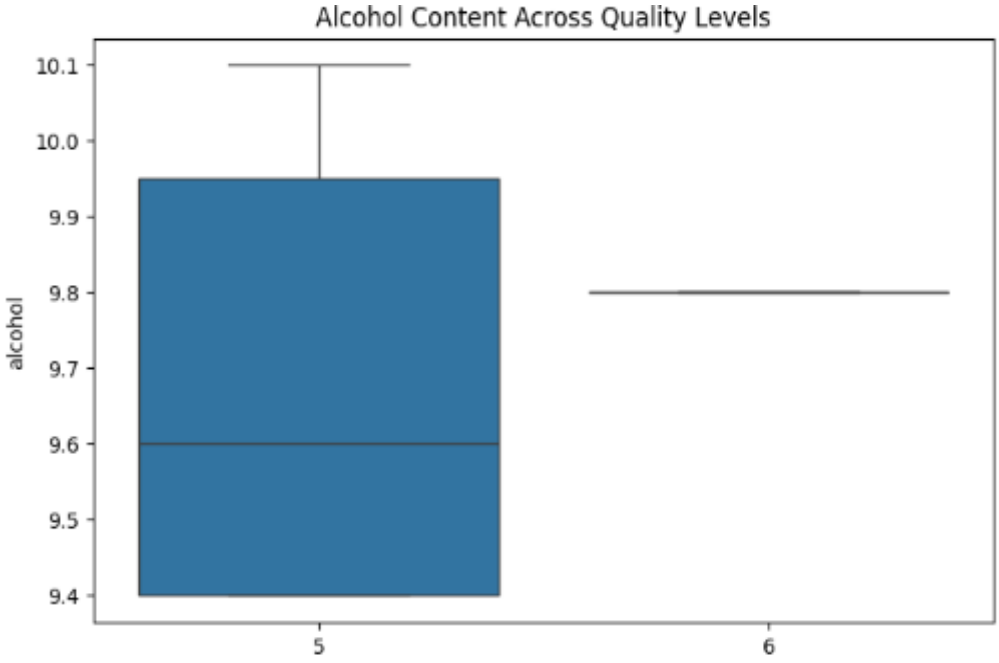
Output:

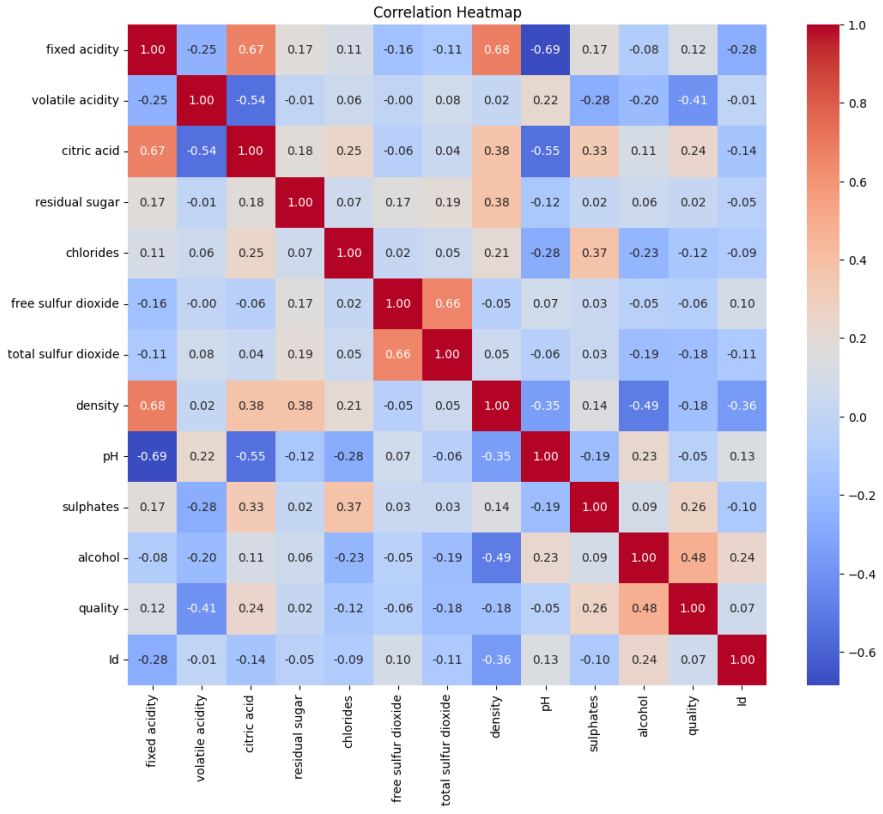
```
First 5 rows of the dataset:
   fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0            7.4              0.70         0.00             1.9      0.076
1            7.8              0.88         0.00             2.6      0.098
2            7.8              0.76         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

Summary statistics:
       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
```

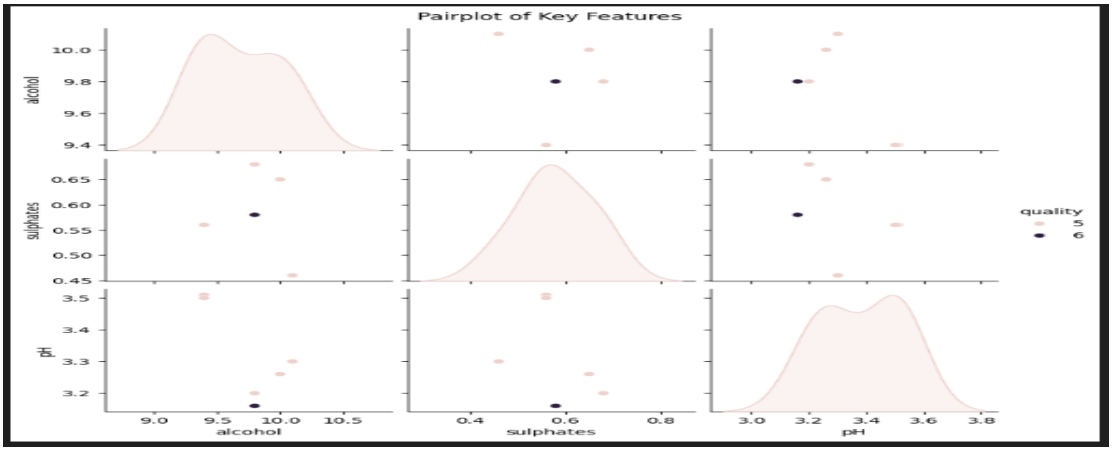


Scatter Matrix of Features




Alcohol Content Across Quality Levels
alcohol
10.1
10.0
9.9
9.8
9.7
9.6
9.5
9.4
5
6
quality


Correlation Heatmap
fixed acidity 1.00 -0.25 0.67 0.17 0.11 -0.16 -0.11 0.68 -0.69 0.17 -0.08 0.12 -0.28
volatile acidity -0.25 1.00 -0.54 -0.01 0.06 -0.00 0.08 0.02 0.22 -0.28 -0.20 -0.41 -0.01
citric acid 0.67 -0.54 1.00 0.18 0.25 -0.06 0.04 0.38 -0.55 0.33 0.11 0.24 -0.14
residual sugar 0.17 -0.01 0.18 1.00 0.07 0.17 0.19 0.38 -0.12 0.02 0.06 0.02 -0.05
chlorides 0.11 0.06 0.25 0.07 1.00 0.02 0.05 0.21 -0.28 0.37 -0.23 -0.12 -0.09
free sulfur dioxide -0.16 -0.00 -0.06 0.17 0.02 1.00 0.66 -0.05 0.07 0.03 -0.05 -0.06 0.10
total sulfur dioxide -0.11 0.08 0.04 0.19 0.05 0.66 1.00 0.05 -0.06 0.03 -0.19 -0.18 -0.11
density 0.68 0.02 0.38 0.38 0.21 -0.05 0.05 1.00 -0.35 0.14 -0.49 -0.18 -0.36
pH -0.69 0.22 -0.55 -0.12 -0.28 0.07 -0.06 -0.35 1.00 -0.19 0.23 -0.05 0.13
sulphates 0.17 -0.28 0.33 0.02 0.37 0.03 0.03 0.14 -0.19 1.00 0.09 0.26 -0.10
alcohol -0.08 -0.20 0.11 0.06 -0.23 -0.05 -0.19 -0.49 0.23 0.09 1.00 0.48 0.24
quality 0.12 -0.41 0.24 0.02 -0.12 -0.06 -0.18 -0.18 -0.05 0.26 0.48 1.00 0.07
Id -0.28 -0.01 -0.14 -0.05 -0.09 0.10 -0.11 -0.36 0.13 -0.10 0.24 0.07 1.00
fixed acidity
volatile acidity
citric acid
residual sugar
chlorides
free sulfur dioxide
total sulfur dioxide
density
pH
sulphates
alcohol
quality
Id
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


Pairplot of Key Features
alcohol
sulphates
pH
quality
5
6
alcohol
sulphates
pH

Result:

EX.NO: 9 DATE:	Use a case study on a data set and apply the various EDA and visualization techniques and present an analysis report

Aim:

Algorithm:

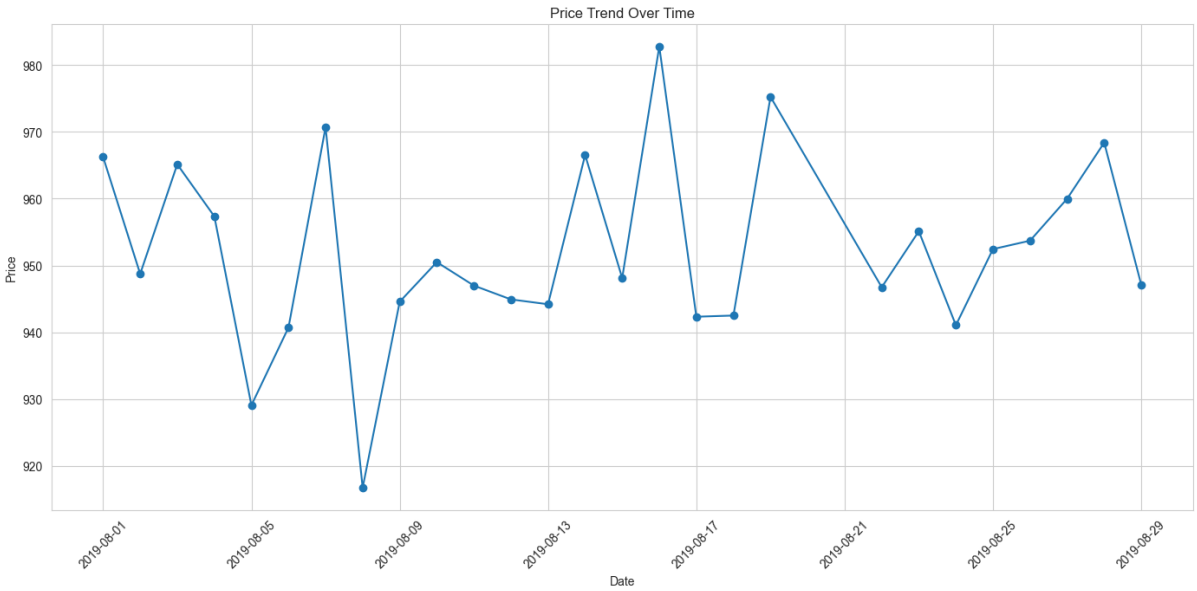
Program:

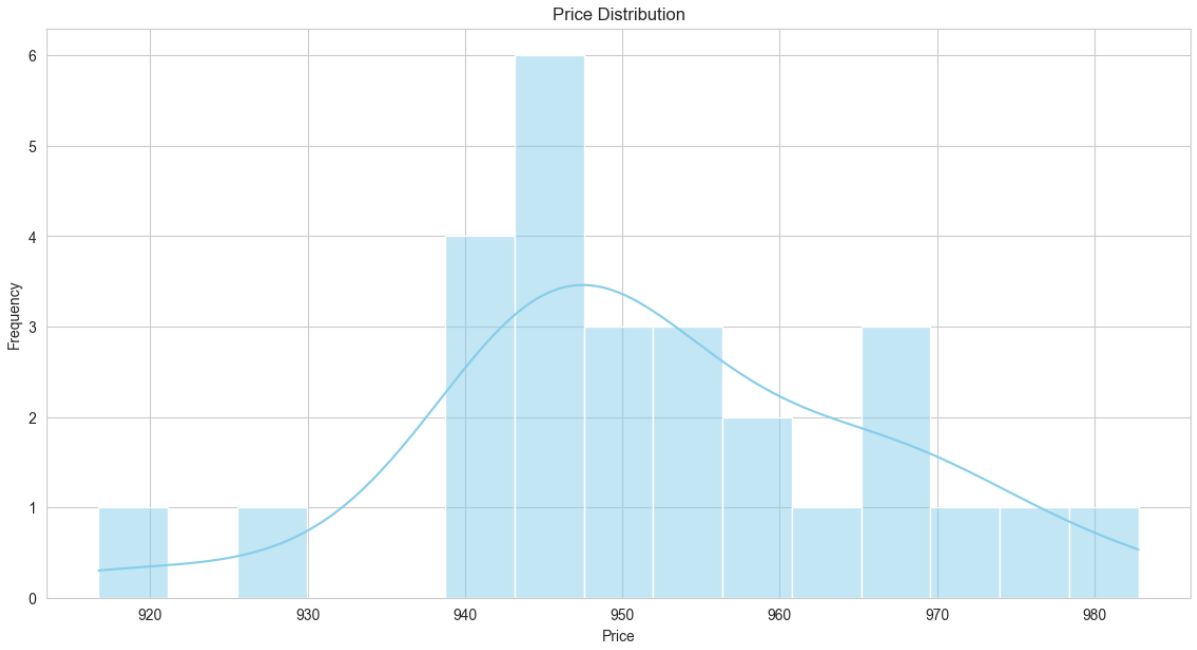
```
import datetime
import pandas as pd
import random
import radar
from faker import Faker
import matplotlib.pyplot as plt
import seaborn as sns

# Setup
fake = Faker()
sns.set_style('whitegrid')  # Correctly apply whitegrid style via seaborn
plt.rcParams['figure.figsize'] = (14, 7)

# Generate Synthetic Data
def generate_data(n):
    data = []
    start = datetime.datetime(2019, 8, 1)
    end = datetime.datetime(2019, 8, 30)

    for _ in range(n):
        date = radar.random_datetime(start='2019-08-01', stop='2019-08-30').strftime("%Y-
%m-%d")
        price = round(random.uniform(900, 1000), 4)
        data.append([date, price])

    return data

# Generate and Prepare DataFrame
n = 100  # Number of records
listdata = generate_data(n)
df = pd.DataFrame(listdata, columns=['Date', 'Price'])
df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')

# Group by Date (in case of duplicate dates)
df_grouped = df.groupby('Date').mean().reset_index()

# Display Data
print("\nFirst 10 rows of dataset:\n", df_grouped.head(10))
print("\nData Summary:\n", df_grouped.describe())

# Plotting Price Trend over Time
plt.figure()
plt.plot(df_grouped['Date'], df_grouped['Price'], marker='o', linestyle='-')
plt.title('Price Trend Over Time')
plt.xlabel('Date')
plt.ylabel('Price')
plt.xticks(rotation=45)
plt.tight_layout()
```

```
plt.show()

# Distribution Plot
plt.figure()
sns.histplot(df_grouped['Price'], bins=15, kde=True, color='skyblue')
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()

# Box Plot
plt.figure()
sns.boxplot(y=df_grouped['Price'], color='orange')
plt.title('Boxplot of Prices')
plt.ylabel('Price')
plt.show()
```

Output:

```
First 10 rows of dataset:
         Date       Price
0 2019-08-01  966.243400
1 2019-08-02  948.737975
2 2019-08-03  965.151775
3 2019-08-04  957.356200
4 2019-08-05  929.060600
5 2019-08-06  940.712800
6 2019-08-07  970.718325
7 2019-08-08  916.708300
8 2019-08-09  944.574050
9 2019-08-10  950.505467

Data Summary:
                      Date       Price
count                   27   27.000000
mean   2019-08-14 14:13:20  952.148321
min    2019-08-01 00:00:00  916.708300
25%    2019-08-07 12:00:00  944.383112
50%    2019-08-14 00:00:00  948.737975
75%    2019-08-22 12:00:00  962.573763
max    2019-08-29 00:00:00  982.802400
std                    NaN   14.244269
```


Price Trend Over Time
Price
Date
980
970
960
950
940
930
920
2019-08-01
2019-08-05
2019-08-09
2019-08-13
2019-08-17
2019-08-21
2019-08-25
2019-08-29


Price Distribution
Frequency
Price
6
5
4
3
2
1
0
920
930
940
950
960
970
980


Boxplot of Prices
Price
980
970
960
950
940
930
920

Result: