

EX.NO: 1

DATE:

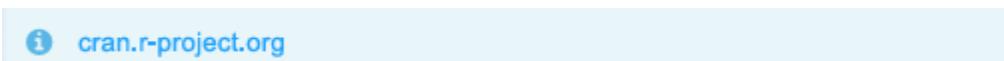
Install The Data Analysis And Visualization Tool: R/Python/ Power BI.

Aim:

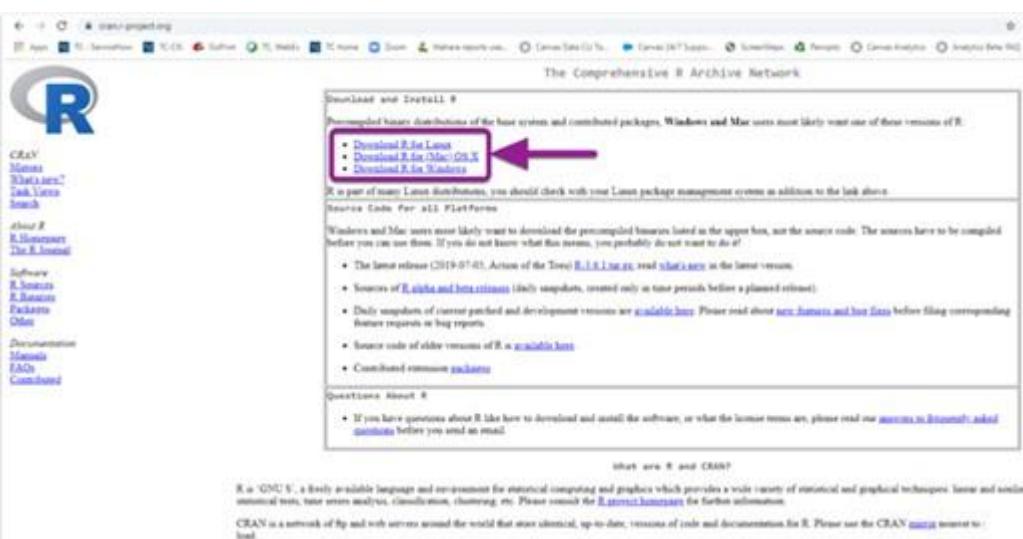
Procedure:

Install R and R-Studio for windows

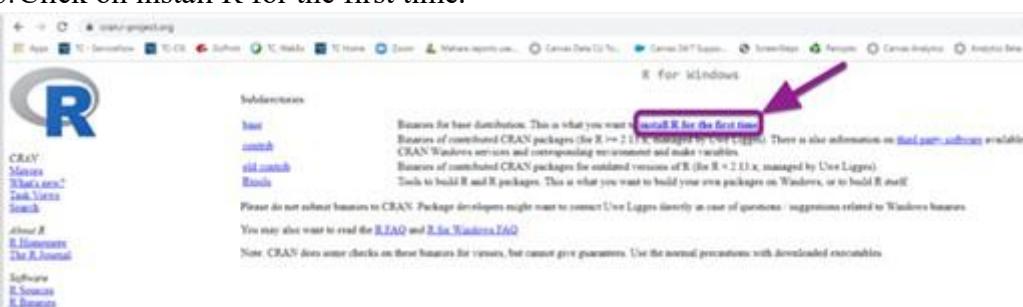
1. To install R, go to cran.r-project.org



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



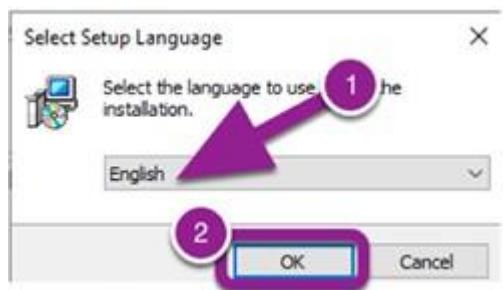
3. Click on install R for the first time.



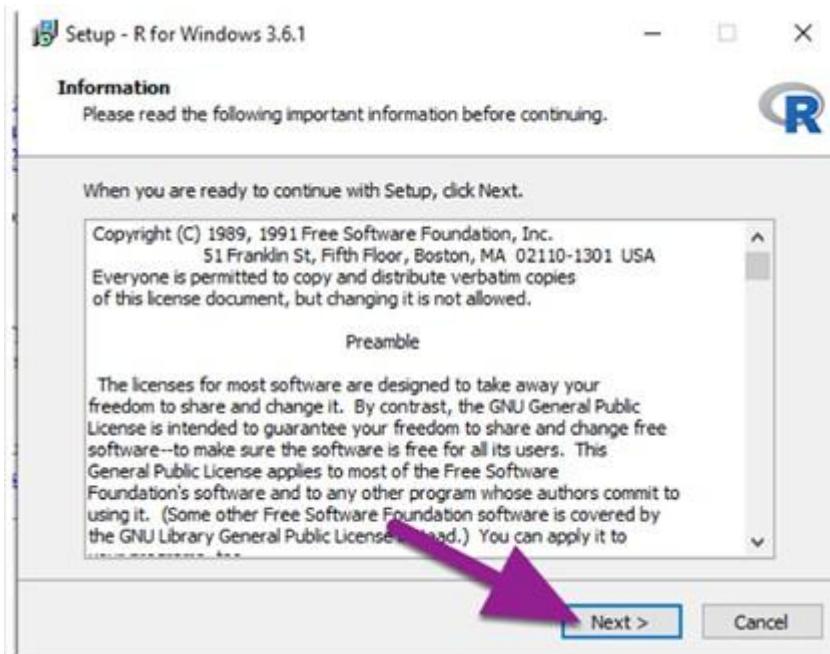
4. Click Download R for Windows. Open the downloaded file.



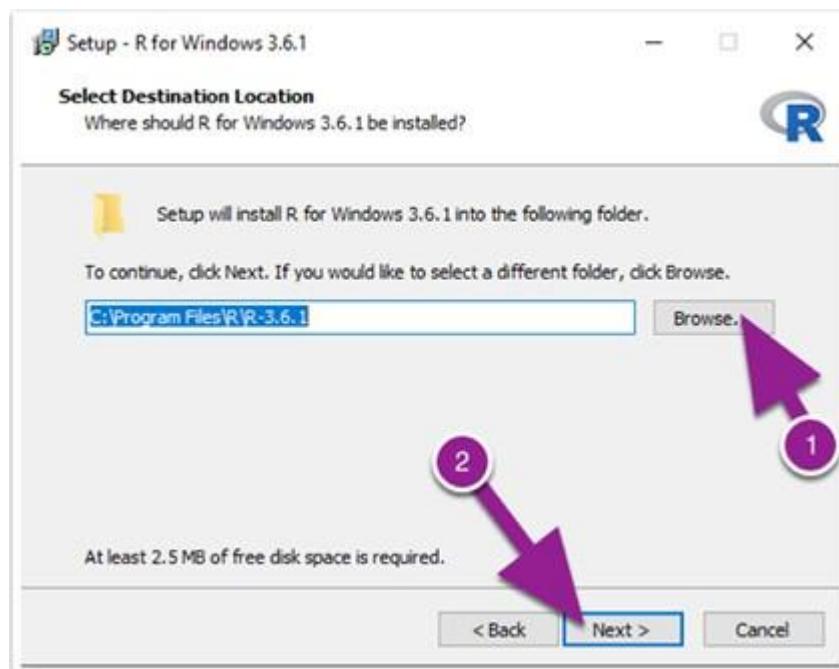
5. Select the language you would like to use during the installation. Then Click Ok.



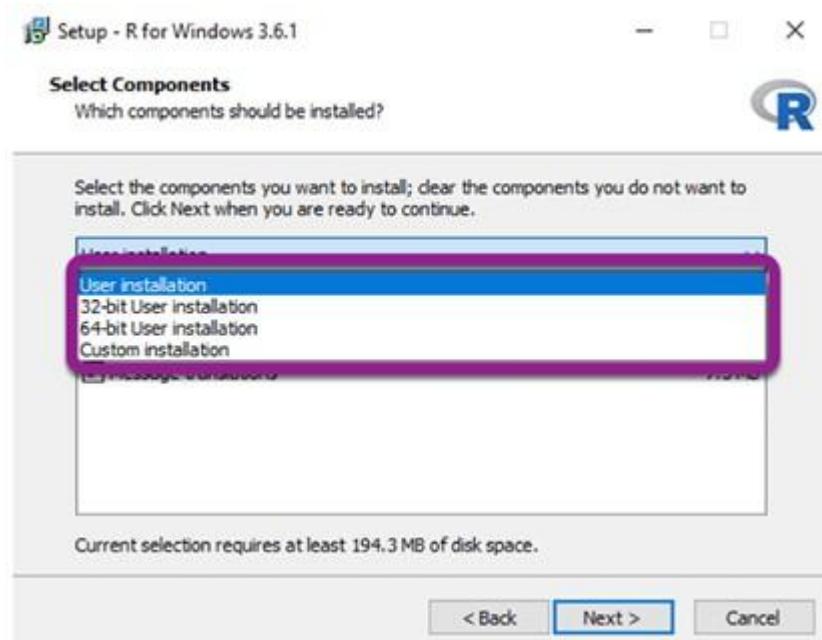
6. Click Next.



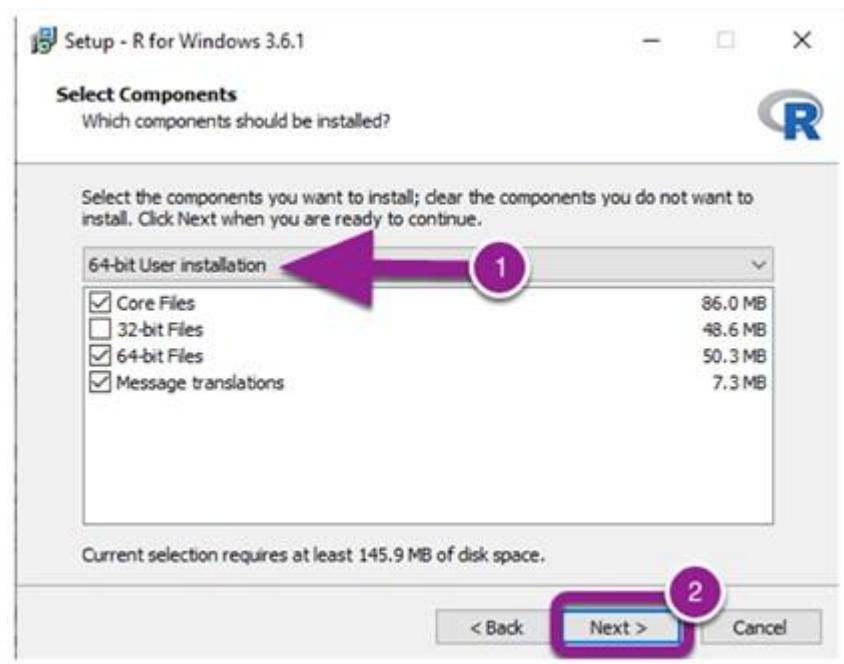
7. Select where you would like R to be installed. It will default to your program Files on your C Drive. Click Next.



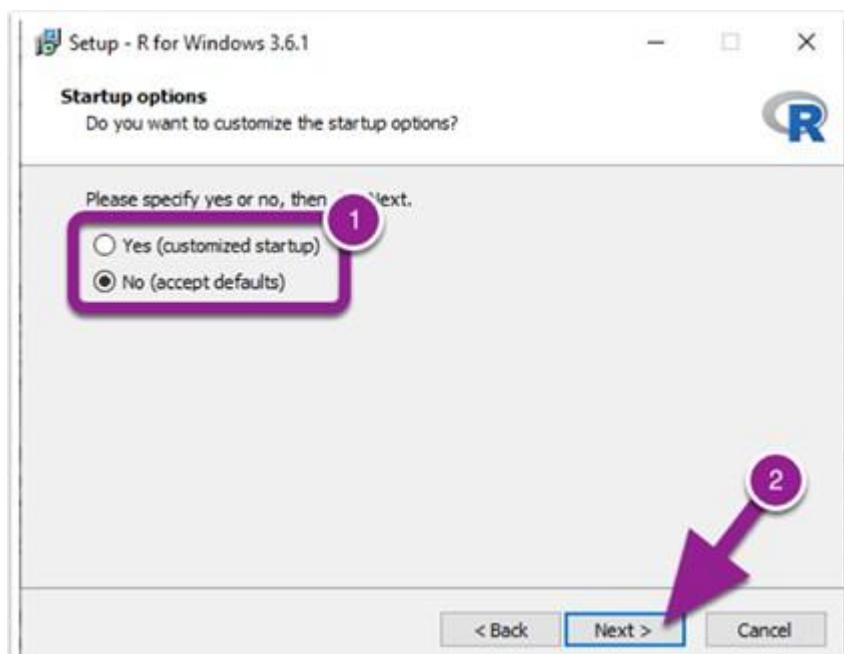
8. You can then choose which installation you would like.



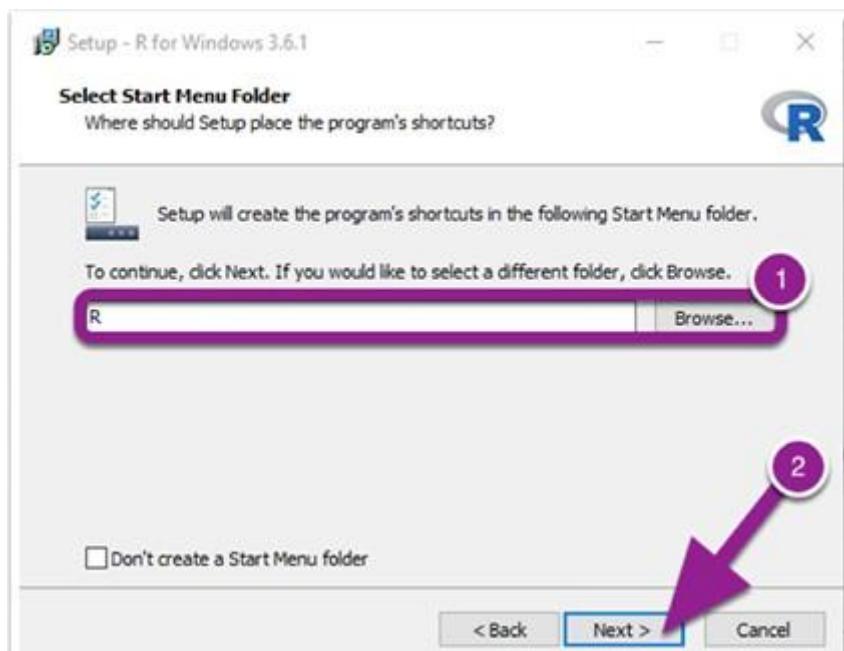
- 9.(Optional) If your computer is a 64-bit user Installation. Then Click Next.



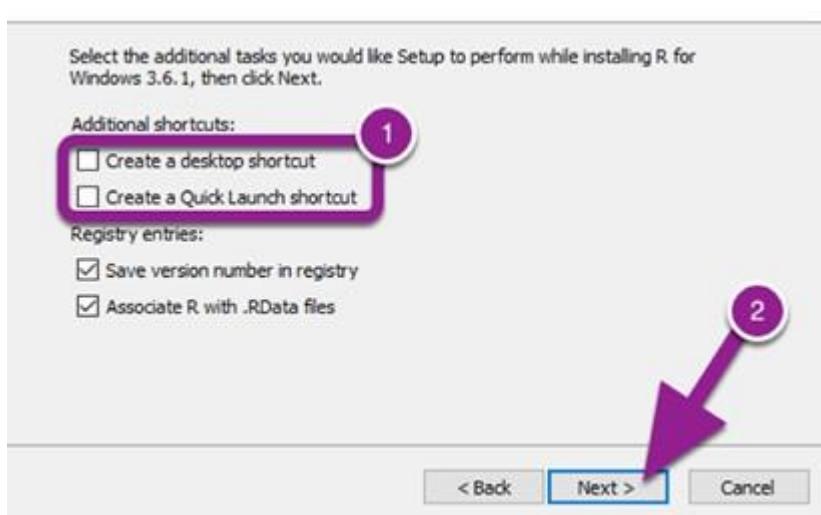
10. Then specify if you want to customized your startup or just use the defaults. Then click Next.



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.

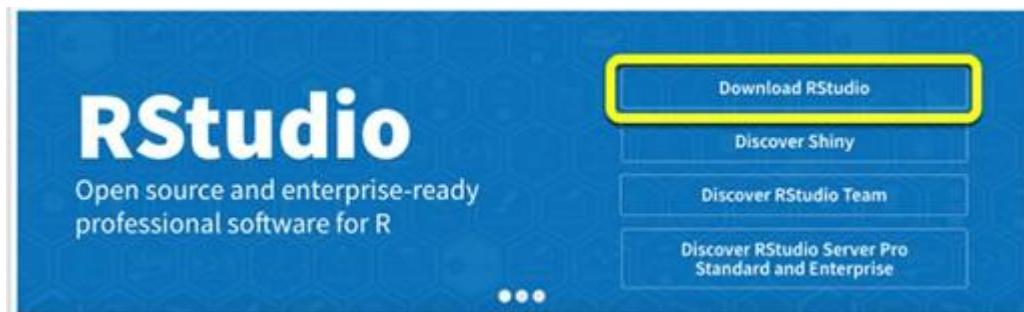


13. Click Finish.



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

15. Click Download RStudio.



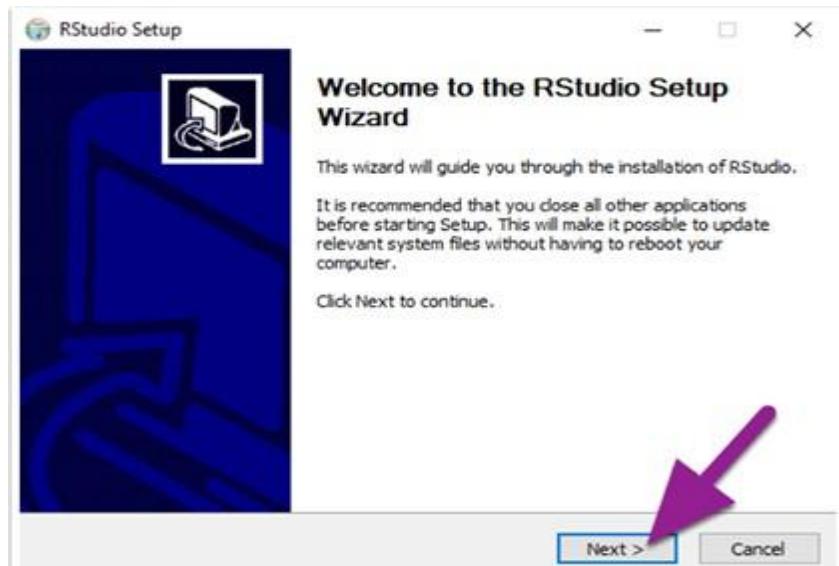
16. Click Download under RstudioDesktop-open source License

A screenshot of the RStudio Desktop download page. The top navigation bar includes links for Products, Resources, Pricing, About Us, and Blogs. Below the navigation, there is a section titled "Choose Your Version of RStudio" with a brief description of RStudio's features. The main content shows five license options: RStudio Desktop Open Source License (FREE, DOWNLOAD), RStudio Desktop Commercial License (\$995 per year, BUY), RStudio Server Open Source License (FREE, DOWNLOAD), RStudio Server Pro Commercial License (\$9,995 per year, DOWNLOAD), and RStudio Server Pro + RStudio Connect Commercial License (\$29,995 per year, TALK). A large image of a computer monitor displaying RStudio is on the right. Below the options, there are three comparison tables: "Integrated Tools for R", "Priority Support", and "Access via Web Browser". The "DOWNLOAD" button for the open-source desktop version is highlighted with a purple box.

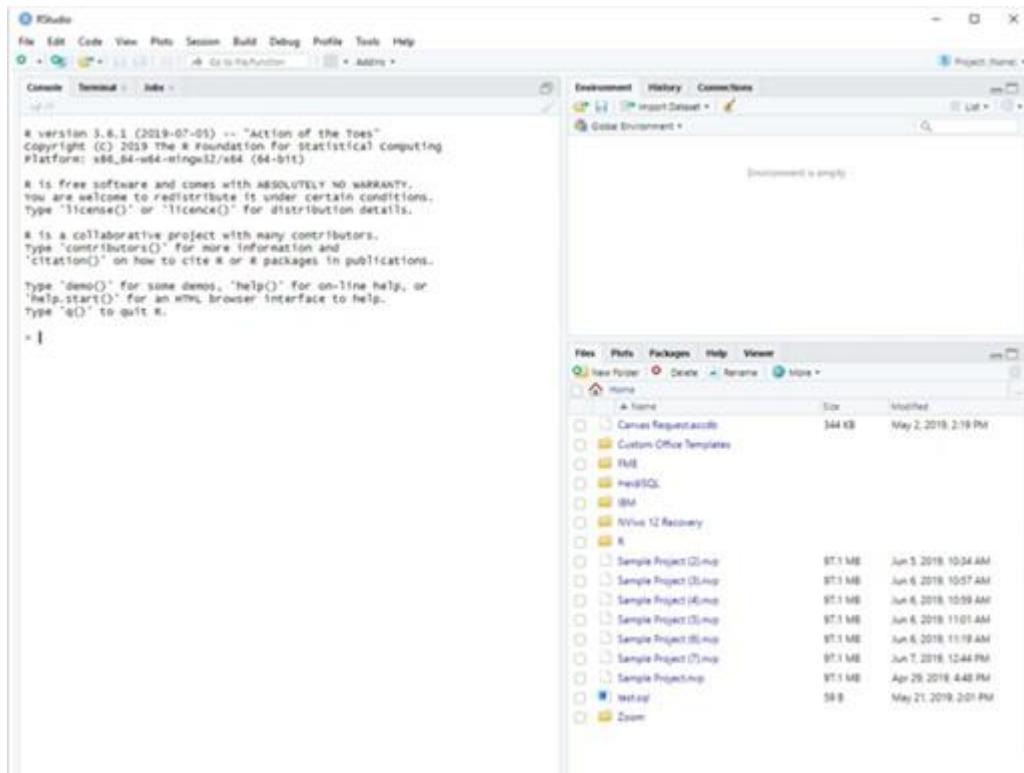
17. Click on the Operating system that you are working with.

Installers for Supported Platforms				
Installer	Size	Date	Md5	
RStudio 1.2.1335 - Windows 7+ (64-bit)	126.9 MB	2019-04-08	d0e2470f1f8effcd35a669aa323a2136	
RStudio 1.2.1335 - Mac OS X 10.8+ (64-bit)	121.1 MB	2019-04-08	6c570b0e2144583f7e48e284ce299eeef	
RStudio 1.2.1335 - Ubuntu 14/Debian 8 (64-bit)	92.2 MB	2019-04-08	c1b07d0511469abfe382919b183eee03	
RStudio 1.2.1335 - Ubuntu 16 (64-bit)	99.3 MB	2019-04-08	c142dd9e210257fb10d1bc045ffff13c7	
RStudio 1.2.1335 - Ubuntu 18 (64-bit)	100.4 MB	2019-04-08	71a8d1990c0d97939804b46cfb0ama75	
RStudio 1.2.1335 - Fedora 19+/RedHat 7+ (64-bit)	114.1 MB	2019-04-08	296b6ef88969a91297fab6545f256a7a	
RStudio 1.2.1335 - Debian 9+ (64-bit)	100.6 MB	2019-04-08	1e32d4d6f6e216f086a81ca82ef65a91	
RStudio 1.2.1335 - OpenSUSE 15+ (64-bit)	101.8 MB	2019-04-08	2795a63c7efde2aa2daa86ba09a81e5	
RStudio 1.2.1335 - SLES/OpenSUSE 12+ (64-bit)	94.4 MB	2019-04-08	c65424b06ef6737279d982db9eefcae1	

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.
3. 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.

4. (Optional) On Windows, to customize the install, on the Install welcome screen, click Customize and change any of the following options:

Windows



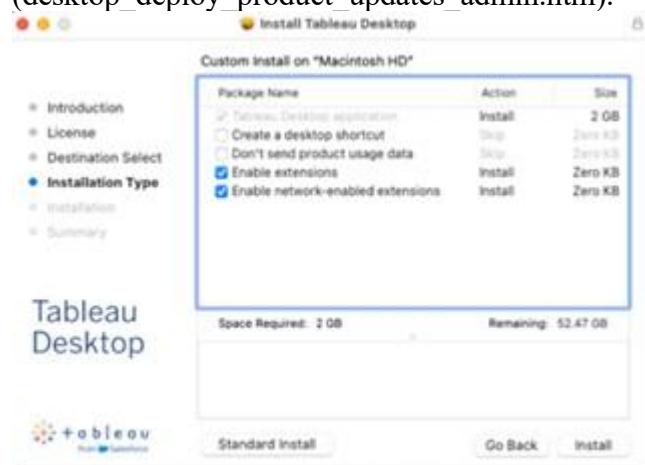
- **Install location:** Specify a different location to install Tableau Desktop.

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

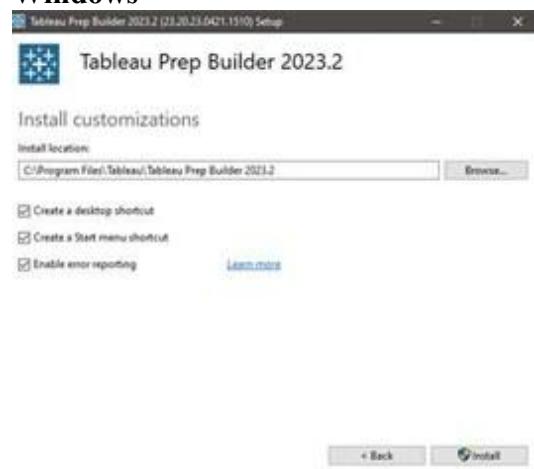
- To opt out of providing usage data, select the Don't send product usage data check box.

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:

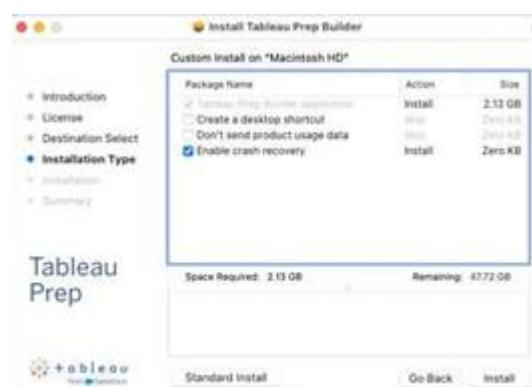
Windows



- 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

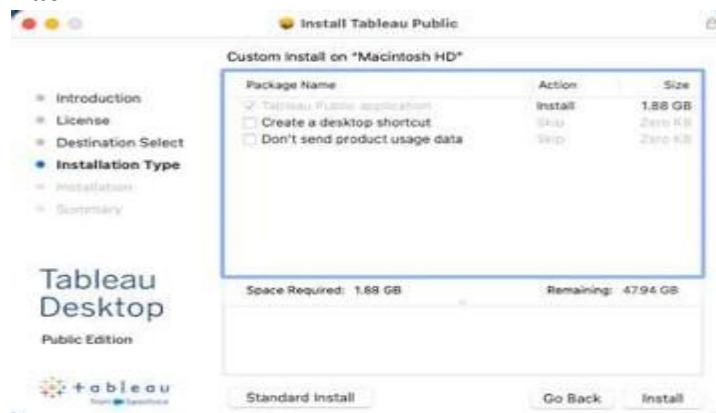


Install location: Specify a different location to install Tableau Prep Builder.

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_License.htm).

- Using the User Interface Uninstaller (desktop_deploy_uninstall_ui.htm)

[Home](#) > [Blog](#) > [Power BI](#) >

How to Download and Install Power BI Desktop

Start using PowerBI Desktop. This post will walk you through the detailed steps of downloading and installing the Microsoft PowerBI Desktop on windows.

Rating: 4.6 ★★★★☆

4100

[GET TRAINED AND CERTIFIED](#)

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.

Want to build your career as a **Power BI Developer**? then Enrol in to our [Power BI Online Certification Training](#)



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 click 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.



Preparing for Microsoft Power BI Interview? Here's Frequently Asked [Microsoft Power BI Interview Questions](#)



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



Follow Microsoft



[Visit here to know about Power BI Architecture](#)

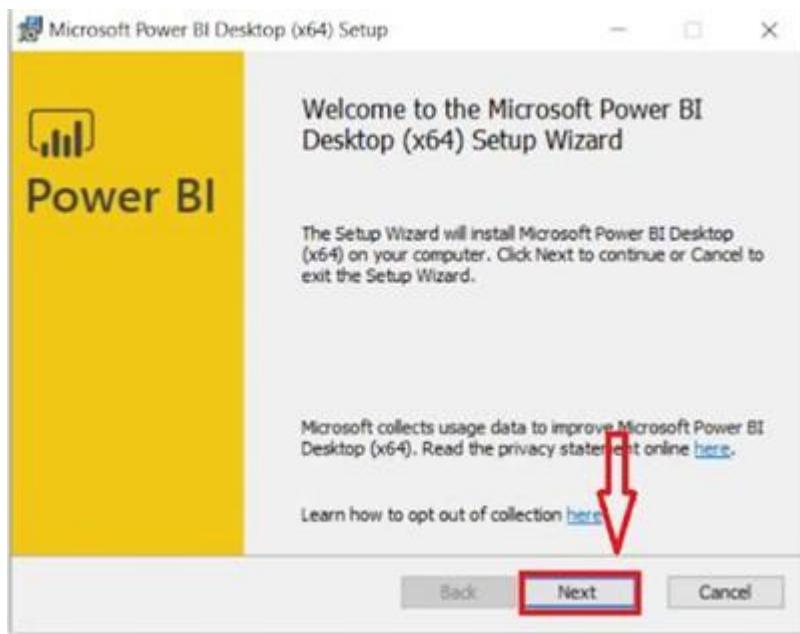
Step 6. The Next button will be enabled once you select either of the FileNames. Now, Click the Next button to download it on your device.



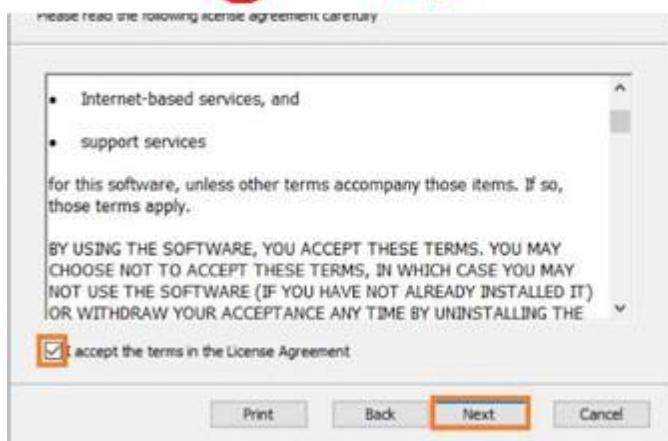
Do you Know- What is Power BI Slicer Is?



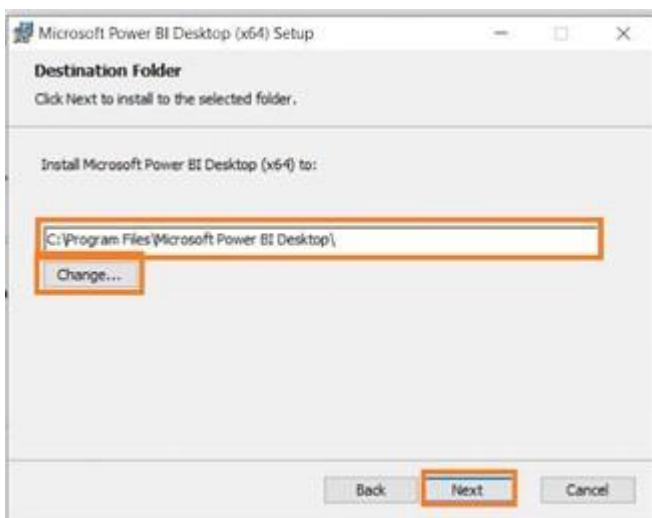
Step 8. Once you click the Next button, you will be asked to click the Next button to continue or the Cancel button to exit in the dialog box.



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.

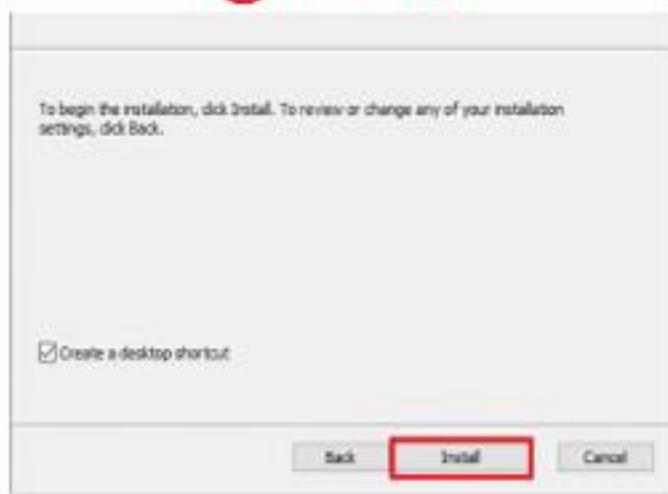
Also Read [Power BI Gateway](#)



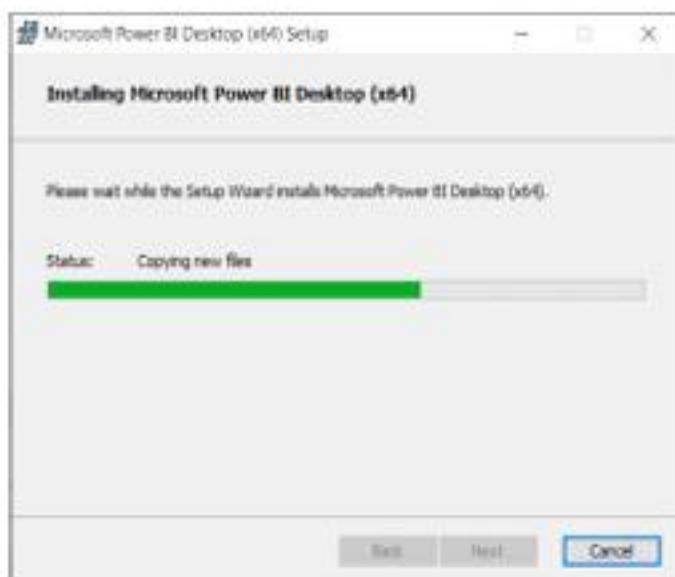
Step 12. Are you ready to Install? If, yes click the Install button (or) Do you want to review or change any of the installation settings? If yes, click Back Button. Next, click the Install button for installation.



Majix



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



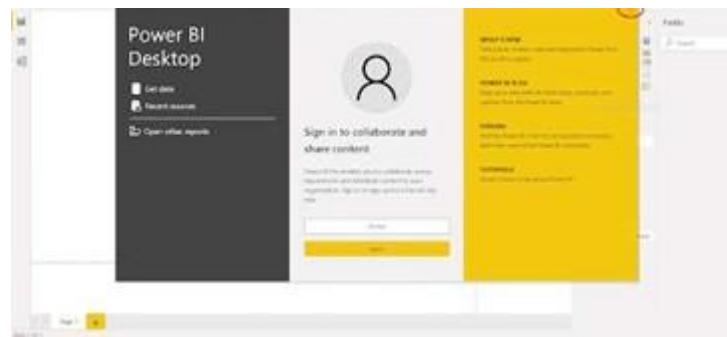
Step 14. Click the **Finish** button to initialize the process.



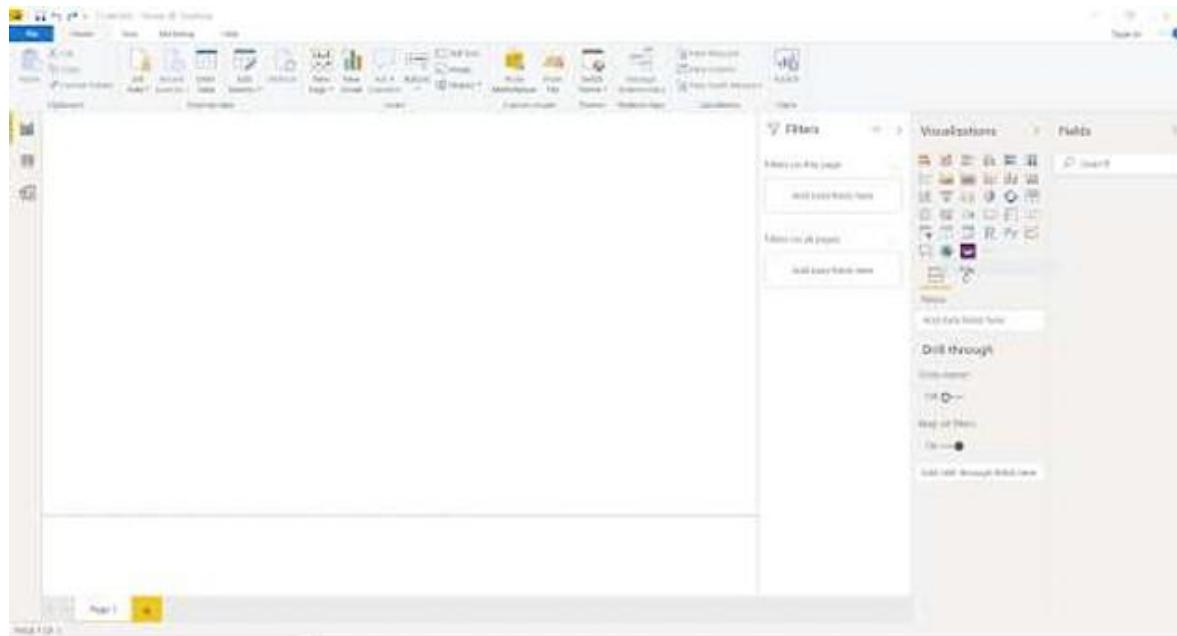
Step 15: Please, wait for a few seconds to start Power BI Desktop.



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 BI 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 computer. It does not take more than 5 minutes in the best case.

Result:

Ex 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 numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import mailbox

import re

import pytz

import datetime

from scipy import ndimage

from scipy.interpolate import interp1d

import matplotlib.gridspec as gridspec

import matplotlib.patches as mpatches

from matplotlib.ticker import MaxNLocator

# Mount Google Drive (for Google Colab)

from google.colab import drive

drive.mount('/content/gdrive')

# Load the mbox file

mboxfile = "/content/gdrive/MyDrive/Takeout/All mail Including Spam and Trash-001.mbox"

mbox = mailbox.mbox(mboxfile)

# Write email metadata to CSV

import csv

with open('mailbox.csv', 'w') 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']
    ])

# Read the CSV with headers

dfs = pd.read_csv('mailbox.csv')

dfs['date'] = pd.to_datetime(dfs['date'], errors='coerce', utc=True)

dfs = dfs[dfs['date'].notna()]

# Extract email address

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(lambda x: extract_email_ID(x))

# Classify as sent or inbox

myemail = 'itsmeskm99@gmail.com'

dfs['label'] = dfs['from'].apply(lambda x: 'sent' if x == myemail else 'inbox')

dfs.drop(columns='to', inplace=True)

# Convert timezone to US/Eastern

def refactor_timezone(x):

    est = pytz.timezone('US/Eastern')

    return x.astimezone(est)

dfs['date'] = dfs['date'].apply(lambda x: refactor_timezone(x))

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'].apply(lambda x: x.hour + x.minute/60 + x.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']

del dfs['date']

print(dfs.index.min().strftime('%a, %d %b %Y %I:%M %p'))

print(dfs.index.max().strftime('%a, %d %b %Y %I:%M %p'))

print(dfs['label'].value_counts())

# Scatter Plot Function
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_xlim(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_xlabel("")

    ax.set_ylabel("")

    ax.set_title(title)

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

    return ax

sent = dfs[dfs['label'] == 'sent']

received = dfs[dfs['label'] == 'inbox']

```

```

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')

# Plot histogram per day per year

def plot_number_perday_per_year(df, ax, label=None, dt=0.3, **plot_kwargs):
    year = df[df['year'].notna()]['year'].values

    T = year.max() - year.min()
    bins = int(T / dt)

    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')

# Hourly histogram

def plot_number_perdhour_per_year(df, ax, label=None, dt=1, smooth=False, weight_fun=None,
                                    **plot_kwargs):
    tod = df[df['timeofday'].notna()]['timeofday'].values

    year = df[df['year'].notna()]['year'].values

    Ty = year.max() - year.min()
    T = tod.max() - tod.min()

    bins = int(T / dt)

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

    if 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)

    orientation = plot_kwargs.get('orientation')

    if orientation == 'horizontal':

        ax.set_ylimits(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()])

    else:

        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()])

# Day of week histogram

sdw = sent.groupby('dayofweek').size() / len(sent)

rdw = received.groupby('dayofweek').size() / len(received)

df_tmp = pd.DataFrame({'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)

```

```

# Day-wise smoothed time-of-day pattern

plt.figure(figsize=(8, 5))

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]

    # Check if the dataframes for this day of the week are empty

    if not df_r.empty:

        wfun_r = lambda x: np.ones(len(df_r)) / len(received)

        plot_number_perdhour_per_year(df_r, ax, dt=1, smooth=True, color=fC{ct}', alpha=0.8, lw=3,
label=dow, weight_fun=wfun_r)

    if not df_s.empty:

        wfun_s = lambda x: np.ones(len(df_s)) / len(sent)

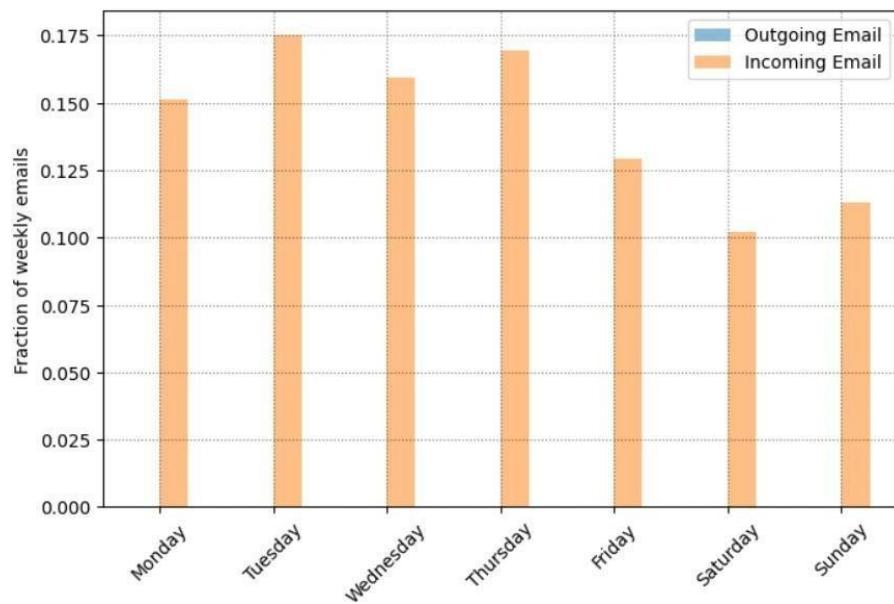
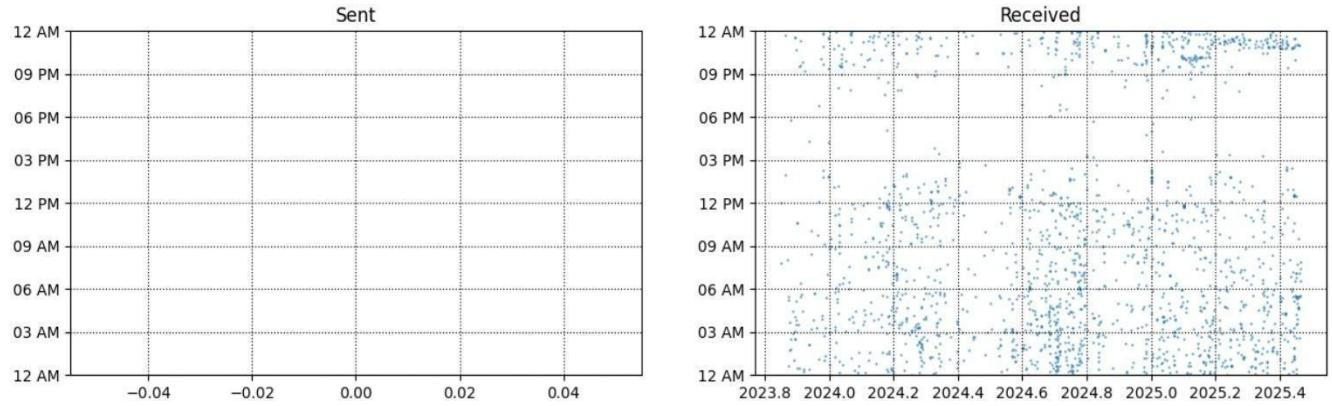
        plot_number_perdhour_per_year(df_s, ax, dt=1, smooth=True, color=fC{ct}', alpha=0.8, lw=2,
label=dow + ' (sent)', ls='--', weight_fun=wfun_s)

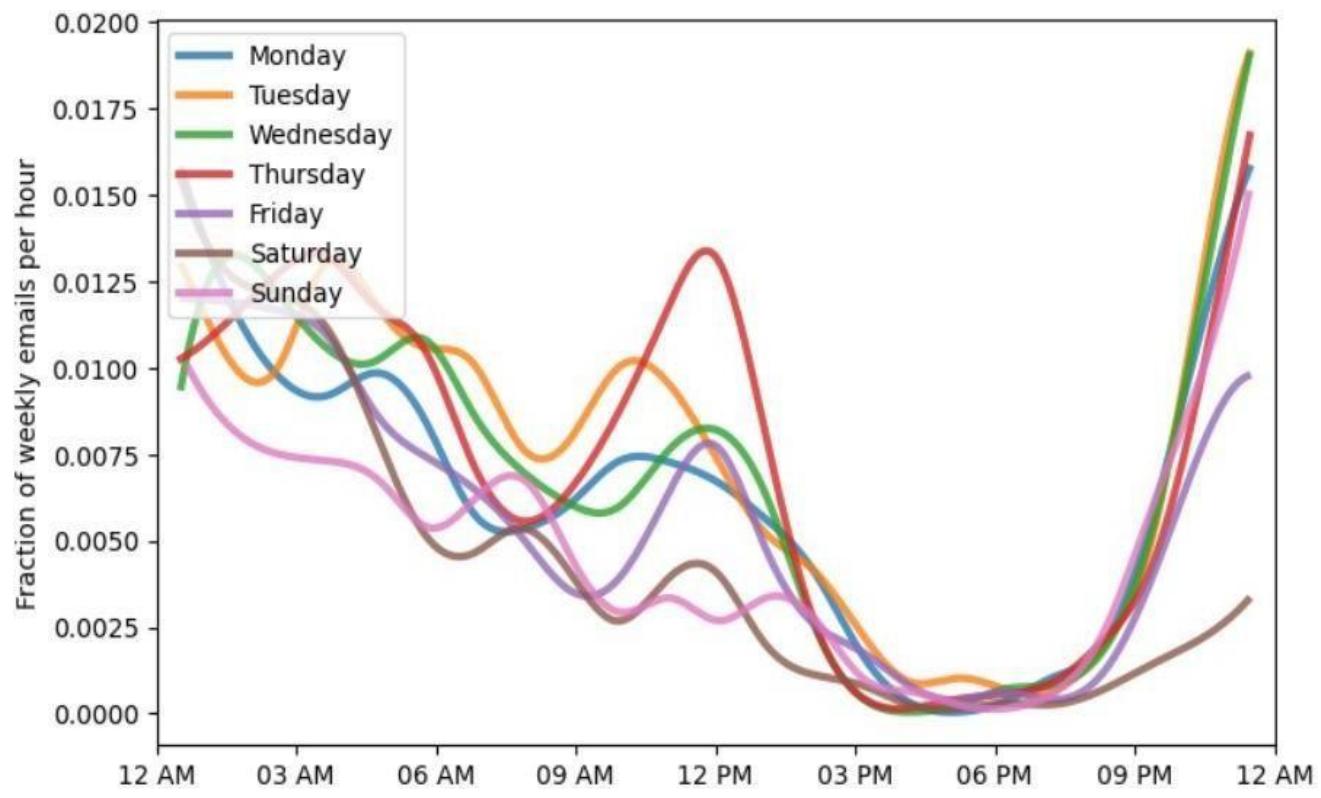
    ax.set_ylabel('Fraction of weekly emails per hour')

    plt.legend(loc='upper left')

```

Output:





Result:

Ex No.3

DATE:

**Working with Numpy arrays, Pandas data frames, Basic plots using
Matplotlib.Numpy arrays using matplotlib**

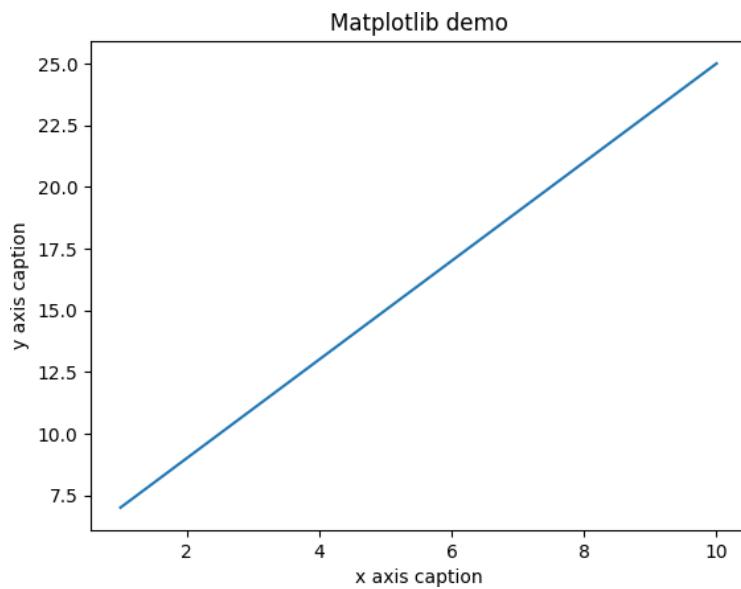
Aim:

Algorithm:

Program:

```
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()
```

Output:

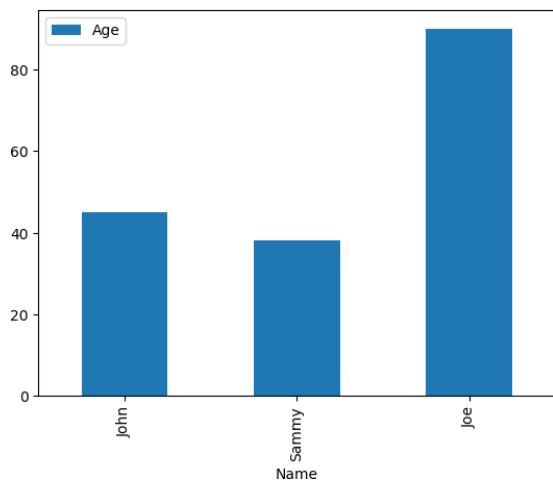


Result:

3 (b) 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")
```

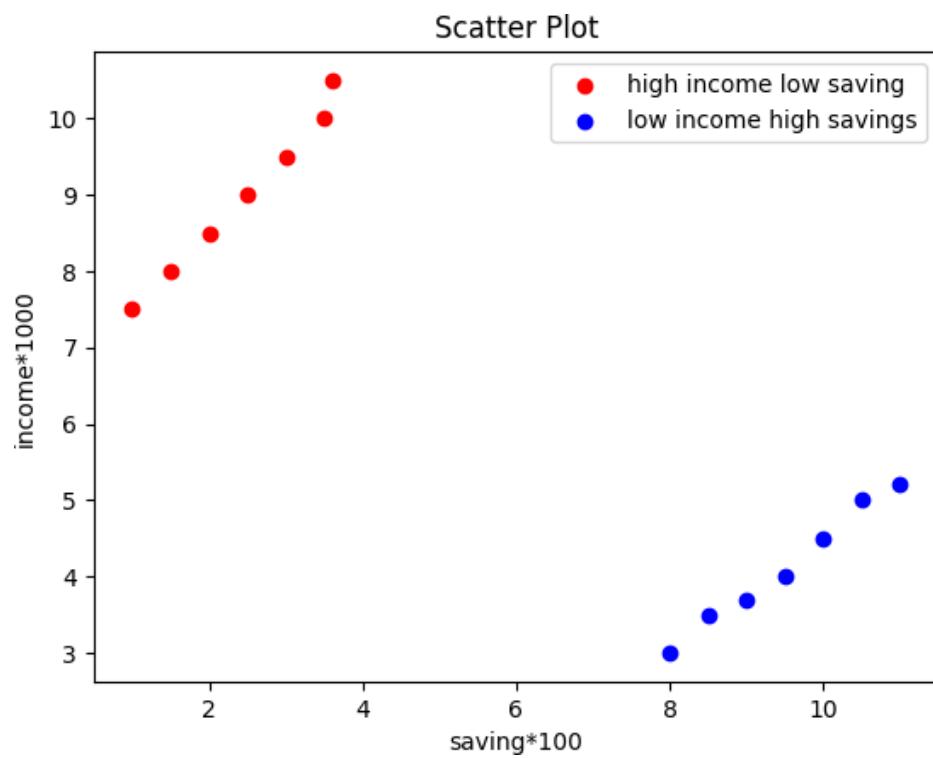
Output:



Basic scatter plots

```
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:



Result:

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

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	0.741315	-1.078900	0.070286
b	NaN	NaN	NaN
c	-1.060693	2.380625	0.811428
d	NaN	NaN	NaN
e	-1.192621	-0.307155	-0.901638
f	-0.384553	-0.533400	1.390651
g	NaN	NaN	NaN
h	-1.503602	-1.213194	0.532317

Program :(Checking duplicate)

```
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:(filling missing data)

```
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  1.083383  2.459676 -0.48089
b      NaN      NaN      NaN
c  0.121610  0.570540  0.41011
NaN replaced with '0':
          one      two      three
a  1.083383  2.459676 -0.48089
b  0.000000  0.000000  0.00000
c  0.121610  0.570540  0.41011
```

Program:(Drop missing values)

```
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	0.349134	-1.099138	-2.930355
c	0.213396	-2.039898	1.102388
e	0.311756	0.240544	-0.229244
f	1.699095	-0.290680	0.376914
h	1.658983	-0.470066	0.289034

Program:(Replace missing or generic values)

```
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})

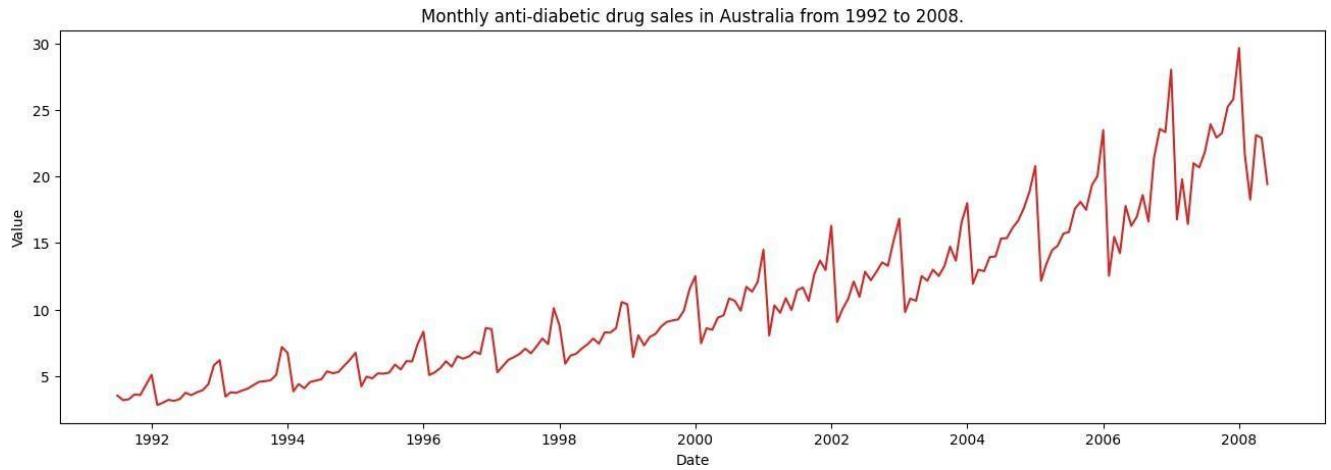
df=pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv',
parse_dates=['date'])
df.head()
```

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

```
import matplotlib.pyplot as plt
df=pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master/a10.csv',
parse_dates=['date'], index_col='date')
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:



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

data=pd.read_csv("C:\\Users\\sundara pandi\\Downloads\\earthquake_1995-2023.csv")

m=folium.Map(location=[13.0827, 80.2707], zoom_start=13)

location_data = data.groupby(['latitude', 'longitude']).size().reset_index(name='count')

for i in range(len(location_data)):

    latitude = location_data.iloc[i]['latitude']

    longitude = location_data.iloc[i]['longitude']

    count = location_data.iloc[i]['count']

    popup_text = f'Count: {count}'

    folium.Marker(

        location=[latitude, longitude],

        popup=folium.Popup(popup_text, parse_html=True),

        icon=folium.Icon(color='red')).add_to(m)

folium.LayerControl().add_to(m)

m.save("map_output.html")
```

Output:



Result:

Ex No.7

DATE:

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

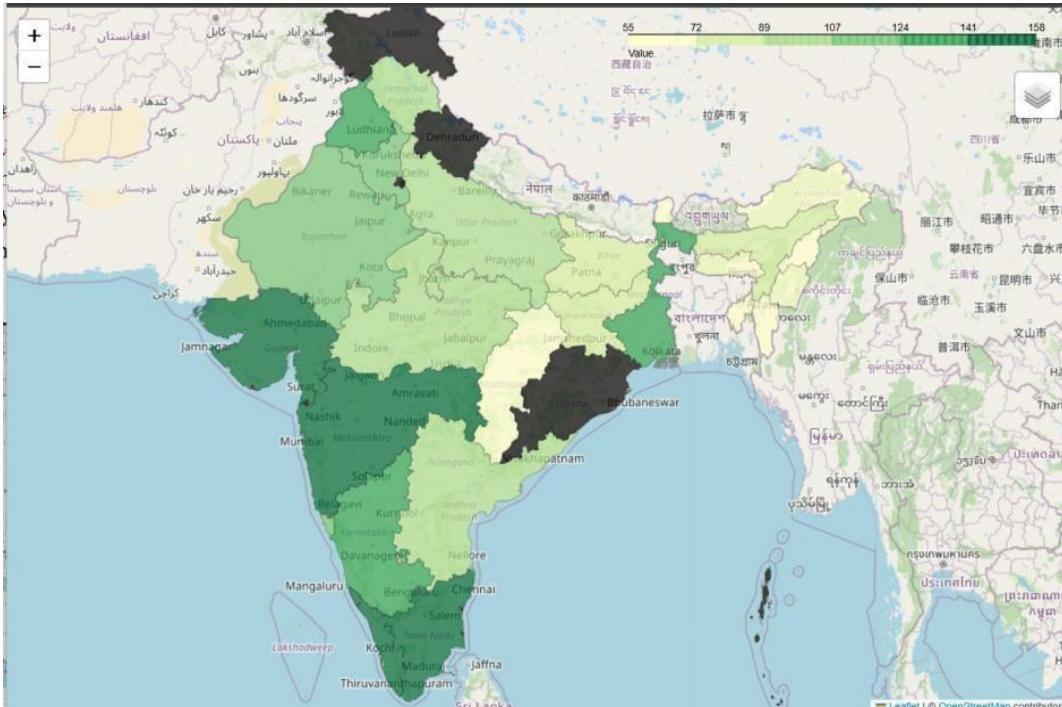
Aim:

Algorithm:

Program:

```
import folium
import pandas as pd
data = pd.read_csv("C:\\\\Users\\\\sundara pandi\\\\Desktop\\\\Dev\\\\india dataset.csv")
m = folium.Map(location=[20.5937, 78.9629], zoom_start=5)
folium.Choropleth(
    geo_data="india_state_geo.json",
    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="Value"
).add_to(m)
folium.LayerControl().add_to(m)
m.save("india_map.html")
```

Output:



Result:

Ex 8.
Date:

Perform EDA on Wine Quality Data Set

Aim:

Algorithm:

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv('winequality-white.csv',delimiter=';')
print(data.head())
print(data.describe())
plt.show()
pd.plotting.scatter_matrix(data,figsize=(20,20))
plt.show()
```

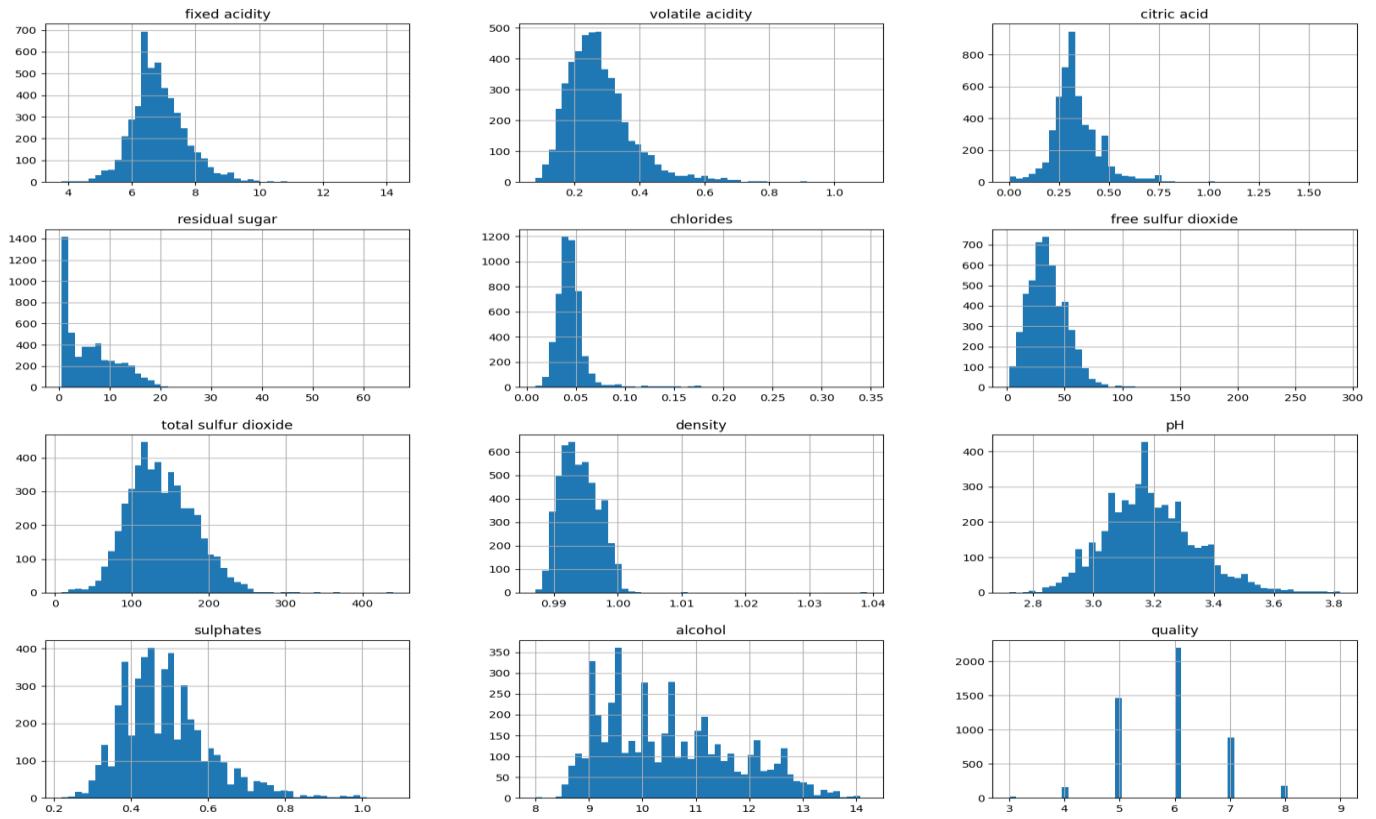
Output:

```
...    fixed acidity  volatile acidity  citric acid  residual sugar  chlorides \
0           7.0          0.27         0.36        20.7       0.045
1           6.3          0.30         0.34         1.6       0.049
2           8.1          0.28         0.40         6.9       0.050
3           7.2          0.23         0.32         8.5       0.058
4           7.2          0.23         0.32         8.5       0.058

      free sulfur dioxide  total sulfur dioxide  density  pH  sulphates \
0                  45.0            170.0     1.0010  3.00      0.45
1                  14.0            132.0     0.9940  3.30      0.49
2                  30.0             97.0     0.9951  3.26      0.44
3                  47.0            186.0     0.9956  3.19      0.40
4                  47.0            186.0     0.9956  3.19      0.40

      alcohol  quality
0      8.8      6
1      9.5      6
2     10.1      6
3      9.9      6
4      9.9      6

      fixed acidity  volatile acidity  citric acid  residual sugar \
count   4898.000000    4898.000000   4898.000000   4898.000000
mean     6.854788     0.278241     0.334192     6.391415
std      0.843868     0.100795     0.121020     5.072058
min      3.800000     0.080000     0.000000     0.600000
...
25%     3.090000     0.410000     9.500000     5.000000
50%     3.180000     0.470000    10.400000     6.000000
75%     3.280000     0.550000    11.400000     6.000000
max     3.820000     1.080000    14.200000     9.000000
```



Result:

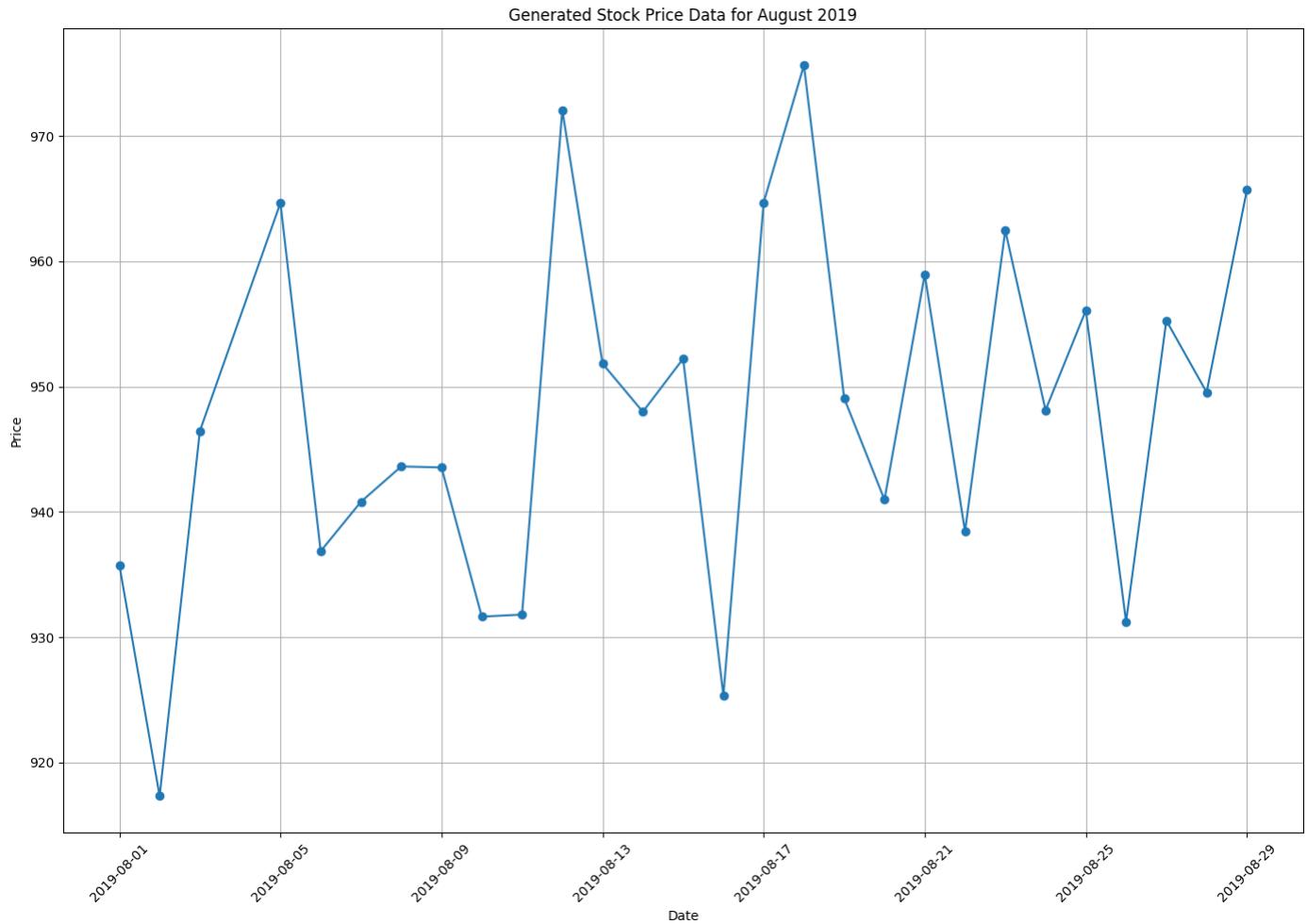
Ex 9.	Use a case study on a data set and apply the various EDA and visualization techniques and present an analysis report
Date:	

Program:

```
import datetime  
  
import random  
  
import pandas as pd  
  
import matplotlib.pyplot as plt  
  
from radar import random_datetime  
  
def generateData(n):  
  
    listdata = []  
  
    start = datetime.datetime(2019, 8, 1)  
  
    end = datetime.datetime(2019, 8, 30)  
  
    for _ in range(n):  
  
        date = random_datetime(start=start, stop=end).strftime("%Y-%m-%d")  
  
        price = round(random.uniform(900, 1000), 4)  
  
        listdata.append([date, price])  
  
    df = pd.DataFrame(listdata, columns=['Date', 'Price'])  
  
    df['Date'] = pd.to_datetime(df['Date'], format='%Y-%m-%d')  
  
    df = df.groupby(by='Date').mean()  
  
    return df  
  
df = generateData(100)  
  
plt.rcParams['figure.figsize'] = (14, 10)  
  
plt.plot(df.index, df['Price'], marker='o')  
  
plt.title("Generated Stock Price Data for August 2019")  
  
plt.xlabel("Date")  
  
plt.ylabel("Price")
```

```
plt.grid(True)  
plt.xticks(rotation=45)  
plt.tight_layout()  
plt.show()
```

Output:



Result:

Ex No: 10

Date:

Statistical Hypothesis Testing

Z-Test, T-Test, F-Test

Aim:

Algorithm:

Program:

```
class_A_scores = [78, 85, 90, 87, 76, 82, 89, 84, 91, 77] # Sample A  
class_B_scores = [72, 70, 75, 78, 80, 74, 77, 79, 73, 76] # Sample B
```

```
import numpy as np  
  
from scipy.stats import ttest_1samp, ttest_ind, ttest_rel, f  
  
from statsmodels.stats.weightstats import ztest
```

```
# Sample dataset  
  
class_A_scores = [78, 85, 90, 87, 76, 82, 89, 84, 91, 77]  
class_B_scores = [72, 70, 75, 78, 80, 74, 77, 79, 73, 76]  
population_mean = 80
```

```
print("==== Z-Test ===")  
  
# One-sample Z-test  
  
z_stat, p_val = ztest(class_A_scores, value=population_mean)  
print("One-sample Z-test (Class A vs Population Mean):")  
print("Z-statistic:", z_stat, "| P-value:", p_val)
```

```
# Two-sample Z-test  
  
z_stat, p_val = ztest(class_A_scores, class_B_scores)  
print("\nTwo-sample Z-test (Class A vs Class B):")
```

```
print("Z-statistic:", z_stat, "| P-value:", p_val)  
print("\n==== T-Test ===")  
  
# One-sample T-test  
  
t_stat, p_val = ttest_1samp(class_A_scores, population_mean)  
print("One-sample T-test (Class A vs Population Mean):")  
print("T-statistic:", t_stat, "| P-value:", p_val)
```

```
# Independent two-sample T-test
t_stat, p_val = ttest_ind(class_A_scores, class_B_scores)

print("\nIndependent Two-sample T-test (Class A vs Class B):")

print("T-statistic:", t_stat, "| P-value:", p_val)

# Paired T-test (assume scores before and after training)
before_training = [60, 65, 68, 70, 75]
after_training = [70, 75, 78, 80, 85]

t_stat, p_val = ttest_rel(before_training, after_training)

print("\nPaired T-test (Before vs After Training):")

print("T-statistic:", t_stat, "| P-value:", p_val)

print("\n==== F-Test ====")

# F-test to compare variance between Class A and Class
B
var1 = np.var(class_A_scores, ddof=1)
var2 = np.var(class_B_scores, ddof=1)
f_stat = var1 / var2
df1 = len(class_A_scores) - 1
df2 = len(class_B_scores) - 1
p_val = 1 - f.cdf(f_stat, df1, df2)
print("F-statistic (Class A vs Class B variances):", f_stat)
print("P-value (one-tailed):", p_val)
```

Output:

```
==== Z-Test ====
One-sample Z-test (Class A vs Population Mean):
Z-statistic: 2.2396703154756357 | P-value: 0.02511233407102513

Two-sample Z-test (Class A vs Class B):
Z-statistic: 4.219056529150508 | P-value: 2.453267629329014e-05

==== T-Test ====
One-sample T-test (Class A vs Population Mean):
T-statistic: 2.2396703154756357 | P-value: 0.05187118134045702

Independent Two-sample T-test (Class A vs Class B):
T-statistic: 4.219056529150508 | P-value: 0.0005159136080406046

Paired T-test (Before vs After Training):
T-statistic: -inf | P-value: 0.0

==== F-Test ====
F-statistic (Class A vs Class B variances): 2.953463203463203
P-value (one-tailed): 0.06119106487728909
```

Result:

Ex No: 11

Date:

Longitudinal Development

Aim:

Algorithm:

Program:

```
import pandas as pd
import numpy as np
import statsmodels.formula.api as smf

# Simulate longitudinal data
np.random.seed(1)
n_participants = 100
time_points = [0, 3, 6, 9, 12]

data = []

for pid in range(1, n_participants + 1):
    base_sleep = np.random.normal(8, 1) # baseline sleep duration (in
    hours) for t in time_points:
        usage = np.random.normal(3 + 0.2 * t, 1) # social media use slightly increases over time
        sleep_quality = base_sleep - 0.3 * usage + np.random.normal(0, 0.5)
        # effect of usage on sleep
        data.append([pid, t, usage, sleep_quality])

df = pd.DataFrame(data, columns=['ParticipantID', 'Month', 'SocialMediaUsage', 'SleepHours'])

# Fit linear mixed model
model = smf.mixedlm("SleepHours ~ SocialMediaUsage + Month", df,
groups=df["ParticipantID"])
result = model.fit()

print(result.summary())
```

Output:

```
Mixed Linear Model Regression Results
=====
Model:           MixedLM  Dependent Variable:  SleepHours
No. Observations:  500      Method:            REML
No. Groups:       100      Scale:             0.2572
Min. group size:  5       Log-Likelihood:     -522.1425
Max. group size:  5       Converged:         Yes
Mean group size:  5.0

-----
          Coef.  Std.Err.      z   P>|z| [0.025 0.975]
-----
Intercept      8.230    0.132  62.253 0.000  7.971  8.489
SocialMediaUsage -0.312   0.027 -11.617 0.000 -0.365 -0.259
Month          -0.016   0.007  -2.229 0.026 -0.030 -0.002
Group Var       0.887    0.294

=====
```

Result:

ExNo:12

Date:

Dimensionality Reduction & Feature Insights

Aim:

Algorithm:

Program:

```
# Import necessary libraries

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

# Load the dataset

df = sns.load_dataset("iris")

print("Original Dataset Shape:", df.shape)

# Drop the categorical target column for

PCA_X = df.drop("species", axis=1)

# Step 1: Standardize the features

scaler = StandardScaler()

X_scaled = scaler.fit_transform(PCA_X)

# Step 2: Apply PCA

pca = PCA(n_components=2)

# Reduce to 2 dimensions for visualization

X_pca = pca.fit_transform(X_scaled)

# Step 3: Create a new DataFrame with principal components

pca_df = pd.DataFrame(data=X_pca, columns=["PC1", "PC2"])

pca_df["species"] = df["species"]
```

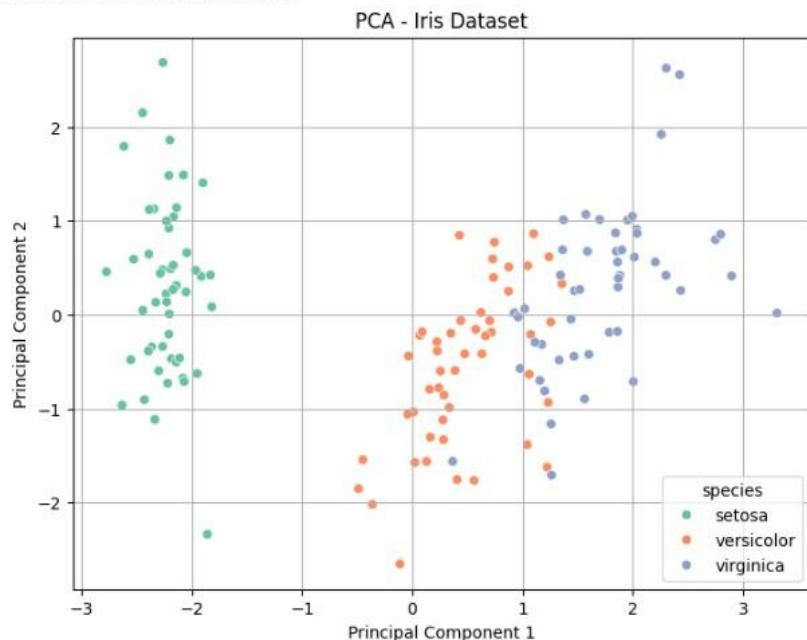
```
# Step 4: Visualize the reduced dimensions  
plt.figure(figsize=(8,6))  
sns.scatterplot(x="PC1", y="PC2", hue="species", data=pca_df, palette="Set2")  
plt.title("PCA - Iris Dataset")  
plt.xlabel("Principal Component 1")  
plt.ylabel("Principal Component 2")  
plt.grid(True)  
plt.show()
```

```
# Step 5: Feature Contribution (Loadings)  
loadings = pd.DataFrame(pca.components_.T, columns=["PC1", "PC2"], index=X.columns)  
print("\n Feature Contribution to Principal Components:\n")  
print(loadings)
```

```
# Step 6: Plot feature contributions  
plt.figure(figsize=(8,5))  
loadings.plot(kind='bar')  
plt.title("Feature Contribution to PC1 and PC2")  
plt.ylabel("Loading Score")  
plt.grid(True)  
plt.xticks(rotation=0)  
plt.tight_layout()  
plt.show()
```

Output:

Original Dataset Shape: (150, 5)

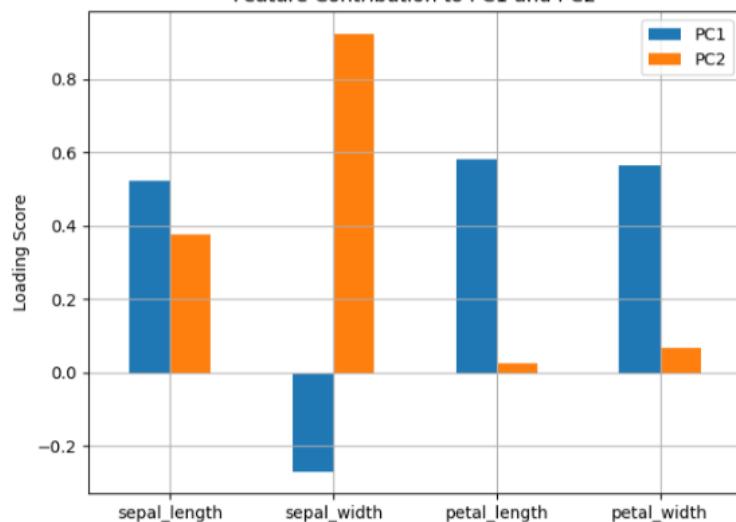


Feature Contribution to Principal Components:

	PC1	PC2
sepal_length	0.521066	0.377418
sepal_width	-0.269347	0.923296
petal_length	0.580413	0.024492
petal_width	0.564857	0.066942

<Figure size 800x500 with 0 Axes>

Feature Contribution to PC1 and PC2



Result:

Ex. No: 13

Date:

Miniproject – Credit Card Fraud Data Analysis And Visualization

Introduction:

With the rapid growth of digital transactions, financial fraud has become a major concern for banks, businesses, and consumers. Traditional methods often fail to detect fraudulent activity effectively, especially when such transactions make up a very small fraction of the overall data.

This project applies data analytics to identify patterns in transaction behavior that are associated with fraud. Using a real-world dataset, the project explores demographic and merchant-level trends, applies statistical analysis to detect significant associations, and uses visual tools to present insights clearly. The goal is to demonstrate how AI techniques can support early fraud detection and informed decision-making.

Problem Statement:

The presence of highly imbalanced fraud data and complex behavioral patterns makes manual fraud detection ineffective. The key challenges addressed include:

- Extremely low fraud occurrence (0.57%)
- Difficulty in correlating user attributes (e.g., gender, merchant) with fraud likelihood
- Need for automated detection and pattern discovery

Objectives:

1. Analyze and preprocess financial transaction data
2. Extract temporal and behavioral patterns associated with fraud
3. Perform statistical correlation tests to identify fraud-related factors
4. Visualize the entire transaction landscape using Power BI
5. Quantify fraud risk across attributes like merchant, category, gender

Data Collection:

The dataset was sourced from Kaggle's "Credit Card Fraud Detection" repository by Kartik2112. It is a simulated credit card transaction dataset containing legitimate and fraud transactions from the duration 1st Jan 2019 - 31st Dec 2020. It covers credit cards of 1000 customers doing transactions with a pool of 800 merchants. This simulation was run for the duration - 1 Jan 2019 to 31 Dec 2020.

No of rows: 10,48,575

Dataset Structure

The dataset consists of multiple weather attributes, including:

1. **Transaction date and time** – Timestamp for each observation.
2. **Merchant** – The merchant the payment has been made to.
3. **Category** – Category of the purchase made.
4. **Amount** – Amount of money spent (Dollars).
5. **Names of the card holders** – First and last names of the card holders.
6. **Gender** – Gender of the card holder.
7. **Addresses** – Addresses of the card holders, including the street, city , latitudes and longitudes.
8. **Job** – The job of the registered card holder.
9. **State** – The state the card holder is residing in.
10. **DoB** – Date of Birth of the card holder.
11. **Is Fraud** – Is the transaction fraudulent (Yes- 1, No- 0).

Data Preprocessing & Cleaning:

1. Handling Missing Values

Although the dataset was complete as it was a simulation, null values have been checked for to ensure data quality. It has been handled using techniques such as:

- Forward Fill (FFill): Propagates the last valid value forward.
- Mean/Median Imputation: Replaces missing numeric entries with the average or median.
- Zero Fill: Used when missing values imply absence.

```
df.fillna(method='ffill', inplace=True)
```

2. Removing Duplicates

Duplicate rows can distort analysis, especially in transaction-level data. All duplicates were removed to ensure one-to-one record mapping per transaction.

```
df.drop_duplicates(inplace=True)
```

3. Date- Time Feature Extraction

The original transaction time was converted to a datetime object to extract meaningful features. Year, Month, Day, Hour were derived for time-based trend analysis.

```
df['trans_date_trans_time'] = pd.to_datetime(df['trans_date_trans_time'], format='%d-%m-%Y %H:%M')
```

```
df['year'] = df['trans_date_trans_time'].dt.year
```

```
df['month'] = df['trans_date_trans_time'].dt.month
```

```
df['day'] = df['trans_date_trans_time'].dt.day
```

```
df['hour'] = df['trans_date_trans_time'].dt.hour
```

4. Encoding Categorical Data

Merchant names contained a "fraud_" prefix which was removed for clarity. Other categorical fields (e.g., gender, category) can be label-encoded if used for modelling.

```
df['merchant'] = df['merchant'].str.replace('fraud_', "", regex=True)
```

5. Dropping irrelevant Columns

Columns such as credit card number, transaction number, and UNIX timestamp were removed as they do not contribute to fraud detection or analysis.

```
df.drop(columns=['cc_num', 'trans_num', 'unix_time', 'trans_date_trans_time'], inplace=True)
```

Exploratory Data Analysis:

EDA helps identify patterns, trends, and relationships among features.

1. Summary Statistics

Summary statistics are numerical values that describe and summarize the main characteristics of a dataset. They provide a quick overview of the distribution, central tendency, and variability of the data.

```
df=pd.read_csv(r'C:\Users\vigne\Downloads\archive(11)\fraudtrain.csv')
print(df.shape)
print(df.head())
print(df.info())
print(df.describe())
```

(1048575, 23)

S.no	trans_date_trans_time	cc_num	merchant	category	amt	first	last	gender	...	long	city_pop
0	01-01-2019 00:00	2.703190e+15	fraud_Rippin, Kub and Mann	misc_net	4.97	Jennifer	Banks	F	...	-81.1781	3495
1	1	6.304230e+11	fraud_Heller, Gutmann and Zieme	grocery_pos	107.23	Stephanie	Gill	F	...	-118.2105	149
2	2	3.885950e+13	fraud_Lind-Buckridge	entertainment	220.11	Edward	Sanchez	M	...	-112.2620	4154
3	3	3.534090e+15	fraud_Kutch, Hermiston and Farrell	gas_transport	45.00	Jeremy	White	M	...	-112.1138	1939
4	4	3.755340e+14	fraud_Keeling-Crist	misc_pos	41.96	Tyler	Garcia	M	...	-79.4629	99

[5 rows x 23 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 1048575 entries, 0 to 1048574

Data columns (total 23 columns):

#	Column	Non-Null Count	Dtype
0	S.no	1048575	int64
1	trans_date_trans_time	1048575	object
2	cc_num	1048575	float64
3	merchant	1048575	object
4	category	1048575	object
5	amt	1048575	float64
6	first	1048575	object
7	last	1048575	object
8	gender	1048575	object
9	street	1048575	object
10	city	1048575	object
11	state	1048575	object
12	zip	1048575	int64
13	lat	1048575	float64
14	long	1048575	float64
15	city_pop	1048575	int64
16	job	1048575	object
17	dob	1048575	object
18	trans_num	1048575	object
19	unix_time	1048575	int64
20	merch_lat	1048575	float64
21	merch_long	1048575	float64
22	is_fraud	1048575	int64

dtypes: float64(6), int64(5), object(12)

memory usage: 184.0+ MB

2. Data Visualization

Visualizing data provides insights into seasonal variations, correlations, and anomalies.

We have used PowerBI for data visualization purposes with various charts like Pie chart, Line chart, Stacked column chart, Cards, Ribbon charts and Clustered Column charts for visualizing the data in different formats . The dashboard presents insights into the data and how each of the features affects the results.

PowerBI Dashboard:



No of fraudulent transactions:

The Pie chart Displays the percentage of fraudulent transactions out of the total number of transactions.

Number of fraudulent transactions: 6006

Fraud count time series:

The line chart time series displays the Number of fraudulent transactions over the dates from 1st January 2019 to 31st December 2020.

No of Fraud transactions to merchants:

The stacked column chart displays the number of fraud transactions made to the merchants. The Highest number of fraud transactions has been made to Padberg-Welch

Fraud transaction in Spending category:

The ribbon chart displays the number of fraudulent transactions in the each of the category the transaction is classified under (groceries, fuel, online shopping, etc). The highest number of fraudulent transactions is under grocery shopping.

Fraud count over the years:

The bar chart displays the fraud counts in the years 2019 and 2020. The highest number of frauds occurred in 2019 with a total count of 5220 and 786 in 2020.

Fraud count by gender:

The bar chart displays the amount of frauds committed by Male and Female individuals. Males have committed 3301 counts of fraud and Females 2975 counts of fraud.

Cards:

The cards display the aggregation of numerical data such as the total number of fraudulent transactions (6006), Total number of transactions (1.05 million) and the total amount of fraudulent currency (3.1 million).

Feature Engineering Techniques Used

1. Extracted year, month, day, and hour from trans_date_trans_time to enable temporal pattern analysis.

```
df['trans_date_trans_time']=pd.to_datetime(df['trans_date_trans_time'],format='%d-%m-%Y %H:%M')
df['year']=df['trans_date_trans_time'].dt.year
df['month']=df['trans_date_trans_time'].dt.month
df['day']=df['trans_date_trans_time'].dt.day
df['hour']=df['trans_date_trans_time'].dt.hour
```

2. Text Cleaning:

Removed the "fraud_" prefix from the merchant field to simplify grouping and interpretation.

```
df['merchant']=df['merchant'].str.replace('fraud_', '', regex=True)
```

3. Irrelevant feature removal:

Dropped fields like cc_num, unix_time, trans_num that add noise or leak sensitive info without analytical value.

```
df.drop(columns=['cc_num','trans_num','unix_time','trans_date_trans_time'],axis=1,inplace=True)
```

Correlation Findings:

The correlation between gender and fraudulent transactions has been calculated by using the Chi-square test as they are both categorical data.

```
contingency_table = pd.crosstab(df['gender'], df['is_fraud']==1)
chi2, p, dof, expected = chi2_contingency(contingency_table)
print("Chi-square Statistic:{0:.2f}".format(chi2))
print(f"Degrees of Freedom: {dof}")
print("p-value:{0:.8f}".format(p))
```

Degrees of Freedom: 1
p-value: 4.940014157810641e-16
There is a significant association between gender and being a fraudster.

Key Findings:

- Fraudulent transactions occurred across multiple merchants, with certain vendors showing >30 fraud cases.
- Fraud was more prevalent in grocery_pos and shopping_net categories.
- Statistically significant gender disparity in fraud involvement.
- Temporal peaks were observed around specific dates and hours.

Future Enhancements:

- Integrate a machine learning model for real-time fraud prediction
- Use geolocation clustering to detect location-based fraud rings
- Create a web dashboard for financial institutions

Conclusion:

The project "Credit Card Fraud Analysis and Visualization" aimed to explore transaction data to identify patterns associated with fraudulent activity. Through data preprocessing, feature extraction, and statistical analysis, key insights were uncovered regarding the distribution of fraud across categories, merchants, and user demographics.

Power BI visualizations further highlighted trends in fraud frequency over time, high-risk merchant profiles, and fraud concentration across transaction categories. Despite a low overall fraud rate, the project demonstrated that meaningful anomalies can be detected through careful data examinations.