COVID 19 USING COGNOS

Data Analytics with cognos – Phase 3

DOCUMENTATION

Team Members:

1.SAGEETHA S(au613021205041)

2.DEEPIKA M (au61302120505)

3.ELAKKIYA V(au6130212050)

4.HARSHINI M(au6130212050)

5.SRI VARTHINI K(au613021205054)

Phase 3: Development Part 1

Problem Statement:

Start bulding the covid 19 cases analysis using IBM Cognos for visualization. Load the dataset using python and data manipulation libraries (e.g., pandas).

Dataset Link:

<https://www.kaggle.com/datasets/chakradharmattapalli/covid-19-cases>

Overview the process

1.Import Libraries:

Begin by importing the necessary libraries, such as pandas for data manipulation.

2.Load the Dataset:

Use pd.read\_csv() or other appropriate methods to load your dataset into a pandas DataFrame.

3.Explore the Dataset:

Display the initial rows, check for missing values, and explore basic statistics to understand the structure and content of the data.

4.Handle Missing Values:

Decide on an appropriate strategy for dealing with missing values, such as dropping rows or filling values based on a specific strategy.

5.Additional Preprocessing Steps:

Depending on the nature of your data, consider additional preprocessing steps such as feature scaling, handling outliers, processing date-time features, dealing with text data, feature engineering, or discretization.

Loading the dataset:

1.Importing libraries

Here, for preprocessing the dataset and manipulate the data,

pandas is the library used to frame the data.

Code:

import pandas as pd

2.Loading the dataset

In this step, we are framing the data into the table using

DataFrame in pandas, and display the head or 5 rows of the

dataset.

Code:

# Replace with the actual filename

file\_path=data=pd.read\_csv("C:/Users/sagee/Downloads/Covid\_19\_cases4.csv")

data

Preprocessing the dataset

3.Explore the dataset:

After framing data, the first few or five rows of the data in

displayed using the head() function.

Code:

data

Output:

dateRep day month year cases deaths countriesAndTerritories

0 31-05-2021 31 5 2021 366 5 Austria

1 30-05-2021 30 5 2021 570 6 Austria

2 29-05-2021 29 5 2021 538 11 Austria

3 28-05-2021 28 5 2021 639 4 Austria

4 27-05-2021 27 5 2021 405 19 Austria

... ... ... ... ... ... ... ...

2725 06-03-2021 6 3 2021 3455 17 Sweden

2726 05-03-2021 5 3 2021 4069 12 Sweden

2727 04-03-2021 4 3 2021 4884 14 Sweden

2728 03-03-2021 3 3 2021 4876 19 Sweden

2729 02-03-2021 2 3 2021 6191 19 Sweden

2730 rows × 7 columns

Code:

print(data.head())

OUTPUT

dateRep day month year cases deaths countriesAndTerritories

0 31-05-2021 31 5 2021 366 5 Austria

1 30-05-2021 30 5 2021 570 6 Austria

2 29-05-2021 29 5 2021 538 11 Austria

3 28-05-2021 28 5 2021 639 4 Austria

4 27-05-2021 27 5 2021 405 19 Austria

4.Check for missing values:

In this step, the missing values or null values, if it present in the

data are separated and number of null values are shown

through this code.

Code:

print("Missing values:\n", data.isnull().sum())

OUTPUT

Missing values:

dateRep 0

day 0

month 0

year 0

cases 0

deaths 0

countriesAndTerritories 0

dtype: int64

5.Check datatype:

In this step, the data type of the columns are discussed

Code: print("Data Types:\n", data.dtypes)

Output:

Data Types:

dateRep object

day int64

month int64

year int64

cases int64

deaths int64

countriesAndTerritories object

dtype: object

6.Check basic statistics:

the statistics of the columns such as count, mean, std, min,

max, 25%, 50%, 75% are shown through the describe()

function command.

Code:

print("Summary Statistics:\n", data.describe())

Output:

Summary Statistics:

day month year cases deaths

count 2730.000000 2730.000000 2730.0 2730.000000 2730.000000

mean 16.000000 4.010989 2021.0 3661.010989 65.291941

std 8.765919 0.818813 0.0 6490.510073 113.956634

min 1.000000 3.000000 2021.0 -2001.000000 -3.000000

25% 8.000000 3.000000 2021.0 361.250000 2.000000

50% 16.000000 4.000000 2021.0 926.500000 14.500000

75% 24.000000 5.000000 2021.0 3916.250000 72.000000

max 31.000000 5.000000 2021.0 53843.000000 956.000000

7.Additional Preprocessing steps:

Perform any other preprocessing steps that are specific to

your dataset and analysis goals. This may include scaling

numeric features, handling outliers, or creating new features.

8.Saving Preprocessed dataset:

In this step, if we made substantial changes to the

dataset and want to save the preprocessed version, you can

use the following Code.

Code:

# Save the preprocessed dataset to a new CSV file

data.to\_csv('preprocessed\_dataset.csv', index=False)

**9.Vizualization:**

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv("C:/Users/sagee/Downloads/Covid\_19\_cases4.csv")

Y = data.iloc[61:,1].values

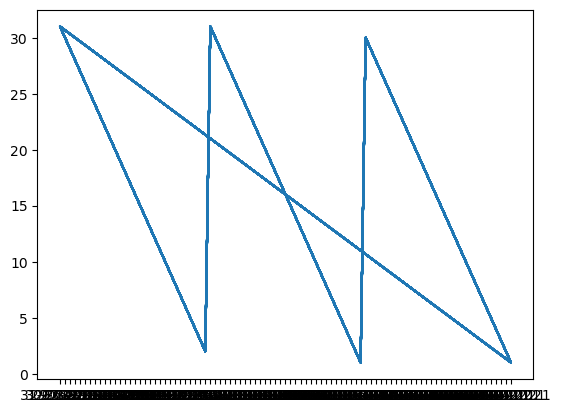
R = data.iloc[61:,3].values

D = data.iloc[61:,5].values

X = data.iloc[61:,0]

plt.plot(X,Y)

Output:



import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv("C:/Users/sagee/Downloads/Covid\_19\_cases4.csv")

Y = data.iloc[61:,1].values

R = data.iloc[61:,3].values

D = data.iloc[61:,5].values

X = data.iloc[61:,0]

plt.figure(figsize=(25,8))

ax = plt.axes()

ax.grid(linewidth=0.4, color='#8f8f8f')

ax.set\_facecolor("black")

ax.set\_xlabel('\nDate',size=25,color='#4bb4f2')

ax.set\_ylabel('Number of Confirmed Cases\n',

size=25,color='#4bb4f2')

ax.plot(X,Y,

color='#1F77B4',

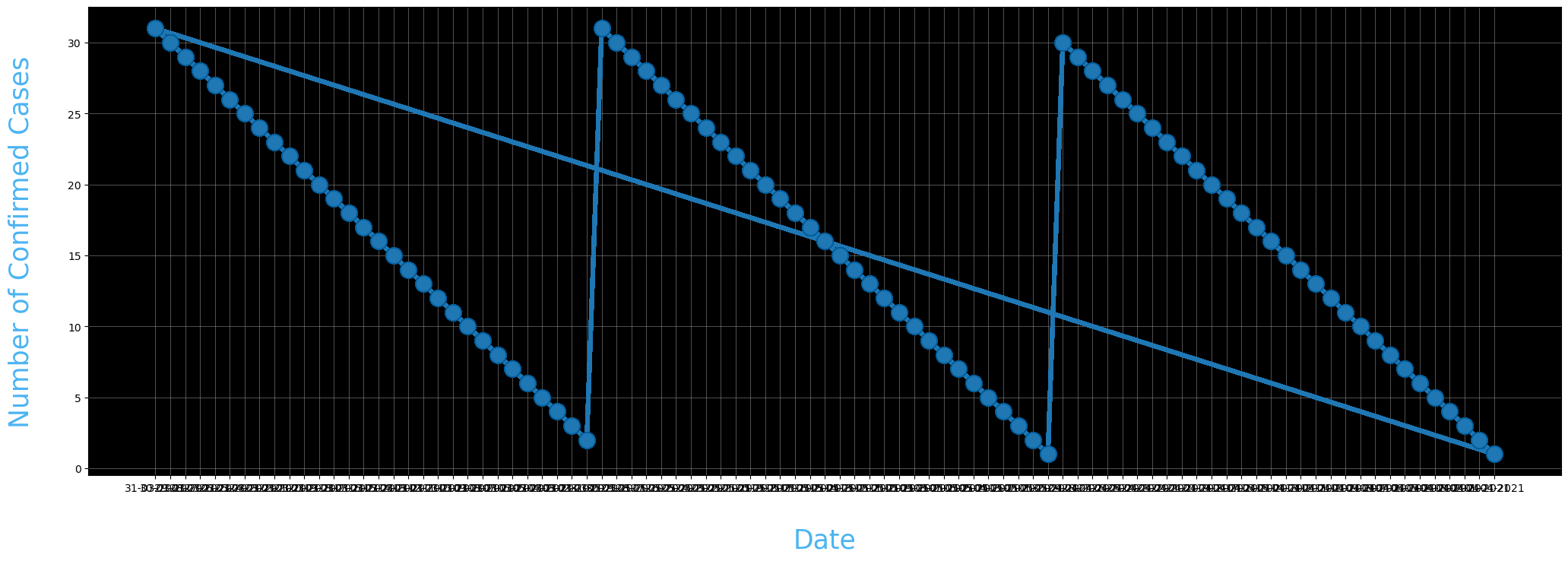
marker='o',

linewidth=4,

markersize=15,

markeredgecolor='#035E9B')

Output:



import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

data = pd.read\_csv("C:/Users/sagee/Downloads/Covid\_19\_cases4.csv")

Y = data.iloc[61:,1].values

R = data.iloc[61:,3].values

D = data.iloc[61:,5].values

X = data.iloc[61:,0]

plt.figure(figsize=(25,8))

ax = plt.axes()

ax.grid(linewidth=0.4, color='#8f8f8f')

ax.set\_facecolor("black")

ax.set\_xlabel('\nDate',size=25,color='#4bb4f2')

ax.set\_ylabel('Number of Confirmed Cases\n',

size=25,color='#4bb4f2')

plt.xticks(rotation='vertical',size='20',color='white')

plt.yticks(size=20,color='white')

plt.tick\_params(size=20,color='white')

for i,j in zip(X,Y):

ax.annotate(str(j),xy=(i,j+100),color='white',size='13')

ax.annotate('Second Lockdown 15th April',

xy=(15.2, 860),

xytext=(19.9,500),

color='white',

size='25',

arrowprops=dict(color='white',

linewidth=0.025))

plt.title("COVID-19 IN : Daily Confirmed\n",

size=50,color='#28a9ff')

ax.plot(X,Y,

color='#1F77B4',

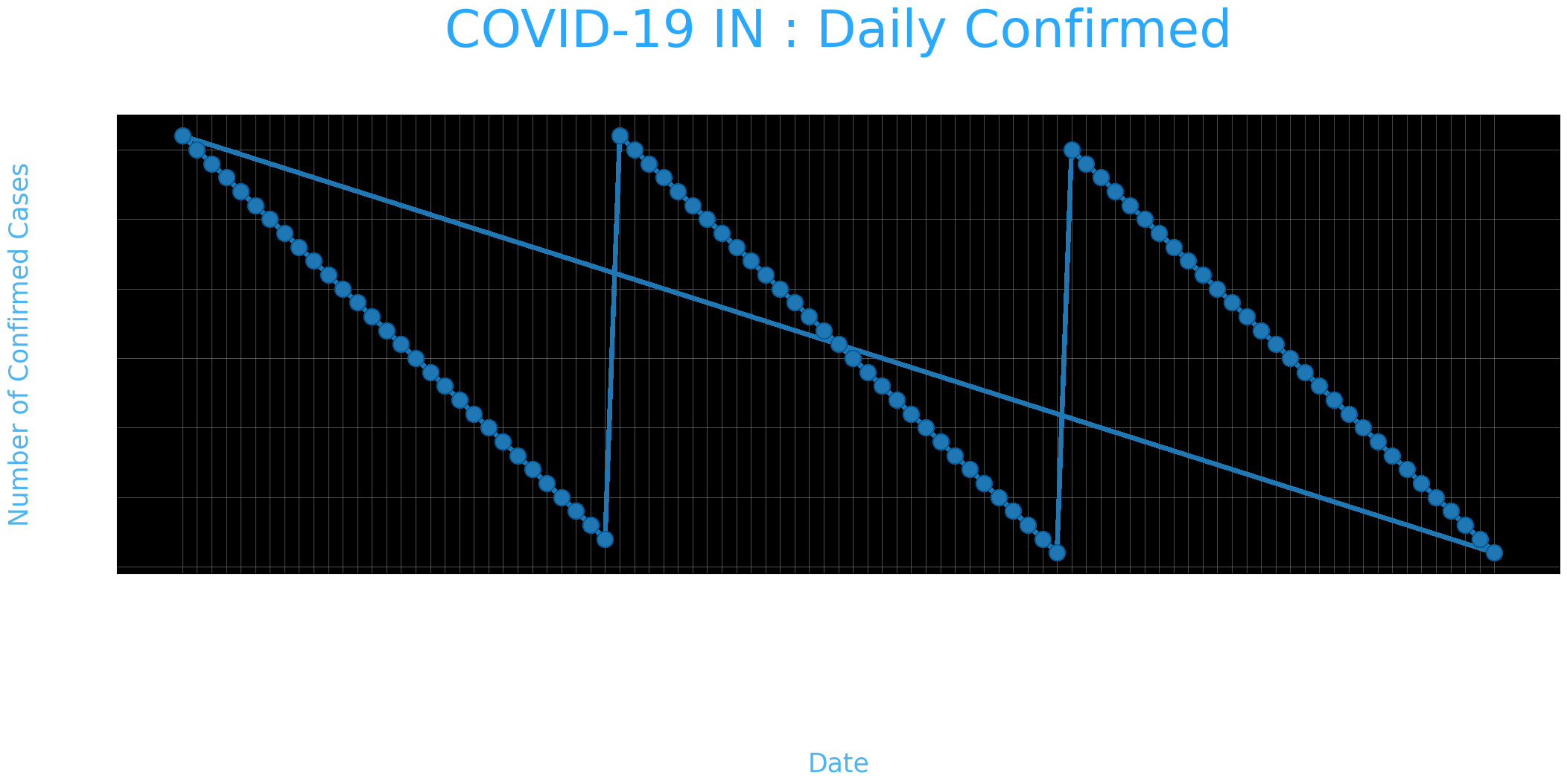
marker='o',

linewidth=4,

markersize=15,

markeredgecolor='#035E9B')

Output:



data = pd.read\_csv("C:/Users/sagee/Downloads/Covid\_19\_cases4.csv")

data.head()

re=data.iloc[:30,5].values

de=data.iloc[:30,4].values

co=data.iloc[:30,3].values

x=list(data.iloc[:30,0])

plt.figure(figsize=(25,10))

ax=plt.axes()

ax.set\_facecolor('black')

ax.grid(linewidth=0.4, color='#8f8f8f')

plt.xticks(rotation='vertical',

size='20',

color='white')#ticks of X

plt.yticks(size='20',color='white')

ax.set\_xlabel('\nDistrict',size=25,

color='#4bb4f2')

ax.set\_ylabel('No. of cases\n',size=25,

color='#4bb4f2')

plt.tick\_params(size=20,color='white')

ax.set\_title('Maharashtra District wise breakdown\n',

size=50,color='#28a9ff')

plt.bar(x,co,label='re')

plt.bar(x,re,label='re',color='green')

plt.bar(x,de,label='re',color='red')

for i,j in zip(x,co):

ax.annotate(str(int(j)),

xy=(i,j+3),

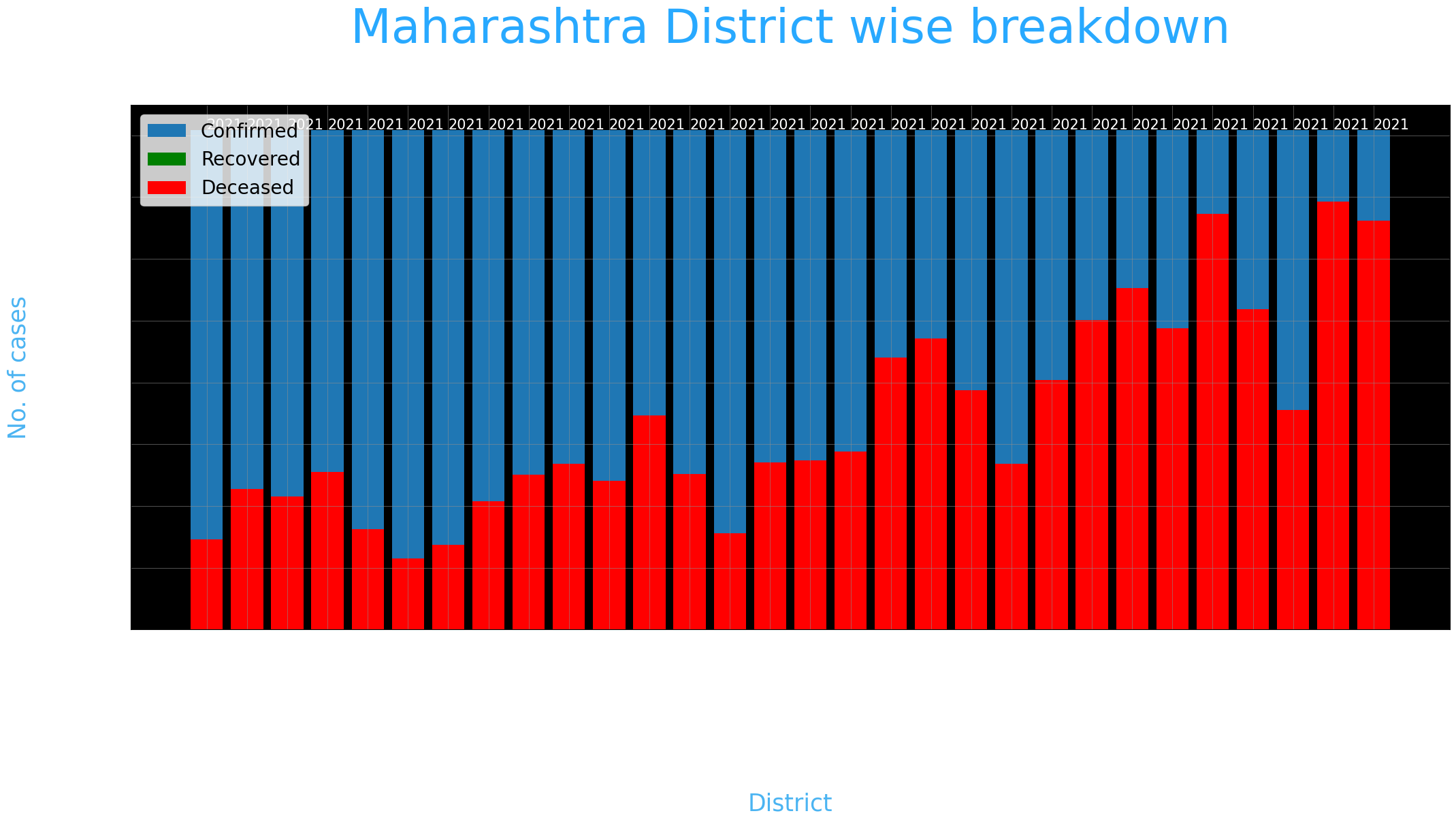
color='white',

size='15')

plt.legend(['Confirmed','Recovered','Deceased'],

fontsize=20)

Output:



Conclusion:

In conclusion, the outlined data loading and preprocessing

steps provide a foundational framework for preparing a dataset

for analysis in Python using the pandas library. By following

these steps, you can ensure that your data is in a suitable

format and quality for further exploration and visualization

tasks.