**DATASET - https://www.kaggle.com/duttadebadri/covid19-india-complete-data**

- Importing the required libraries

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

- Importing the excel file and its sheets as individual DataFrames.

```
df_main = pd.ExcelFile('COVID19.xlsx')
df1 = pd.read_excel(df_main, 'Raw Data')
df2 = pd.read_excel(df_main, 'Death & Recovered')
df3 = pd.read_excel(df_main, 'State-Wise Data')
df4 = pd.read_excel(df_main, 'Daily Cases Time-Series')
df5 = pd.read_excel(df_main, 'State-Wise Testing Data')
```

---

- Displaying data using sub-setting by specific column

```
recovered = df1[df1['Current Status'] == 'Recovered']
print(recovered.head(6))
```

This will only display the recovered patient's data. Here head (6) displays the first 6 rows in the dataframe.

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

We can use this syntax before print () to display all the columns in the dataframe.

- In the second dataframe we have the patient status column. We are writing a function here which will give us the total patients who are Deceased, Recovered and the patients who are Not Counted by Any State in the form of a dictionary.

```
def Patient_Status_Count(df, col_name):
    status_count = {}
    col = df[col_name]
    for status in col:
        if status in status_count.keys():
            status_count[status] = status_count[status] +
1
        else:
            status_count[status] = 1
    return status_count


result = Patient_Status_Count(df2, 'Patient_Status')
print(result)
```

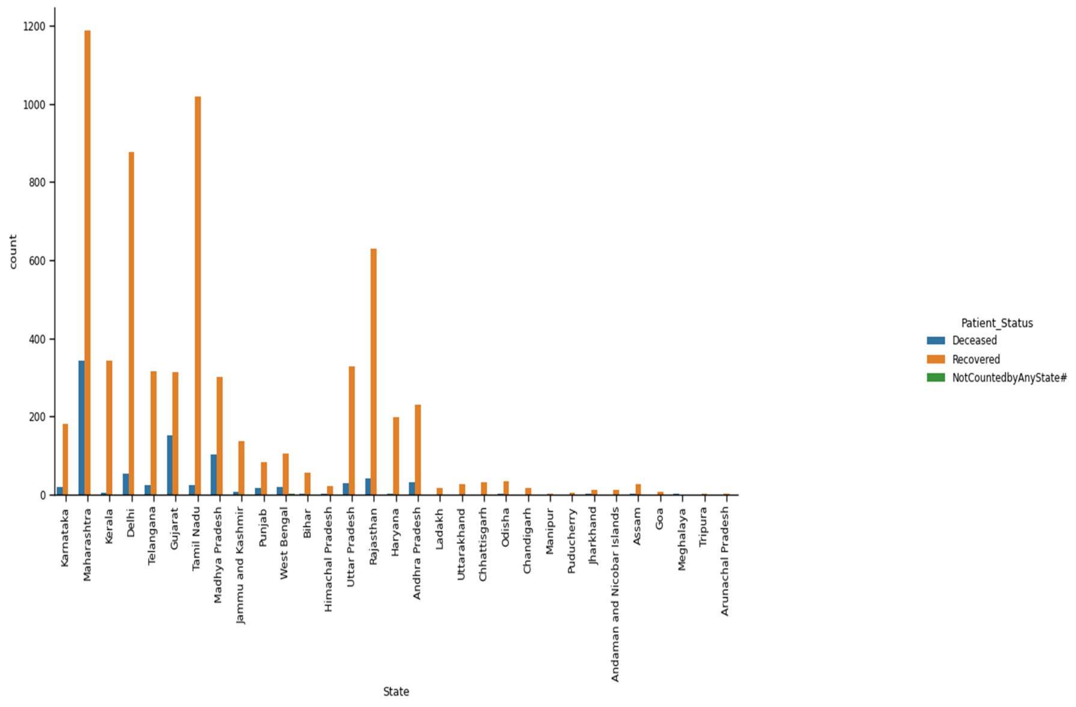
**OUTPUT**

```
{‘Deceased’:881, ‘Recovered’:6523,
‘NotCountedbyAnyState#’:3}
```

- Plotting the patient status using seaborn

```
print(df2.columns)
sns.catplot(x='State', kind='count',
hue='Patient_Status', data=df2)
plt.xticks(rotation=90)
plt.show()
```

Here we have used count plot of category plots.
We have taken the name of states on X-Axis and we use the patient status column as a hue, so that the plot will be visualized and coloured based on the patient status.
df2.columns displays all the columns in the given dataframe.



- The third data frame has state-wise data. A new dataframe called cnf_data was created from df3 using loc () function.

```
cnf_data = df3.loc[:, ['State', 'Confirmed', 'Recovered',
'Deaths']]
cnf_data = cnf_data.dropna()
```

Here dropna() was used to remove the rows with null or zero values and percentages for confirmed, recovered cases and fatalities were calculated.

```
print(cnf_data[cnf_data.Percentage_Recovered ==
cnf_data.Percentage_Recovered.max()])
```

This syntax will give us the rows with max patients recovered.

```
andhra = cnf_data.query("State == 'Andhra Pradesh'")
print(andhra)
```

We can use the sql’s query to get values for a specific row based on name or index.

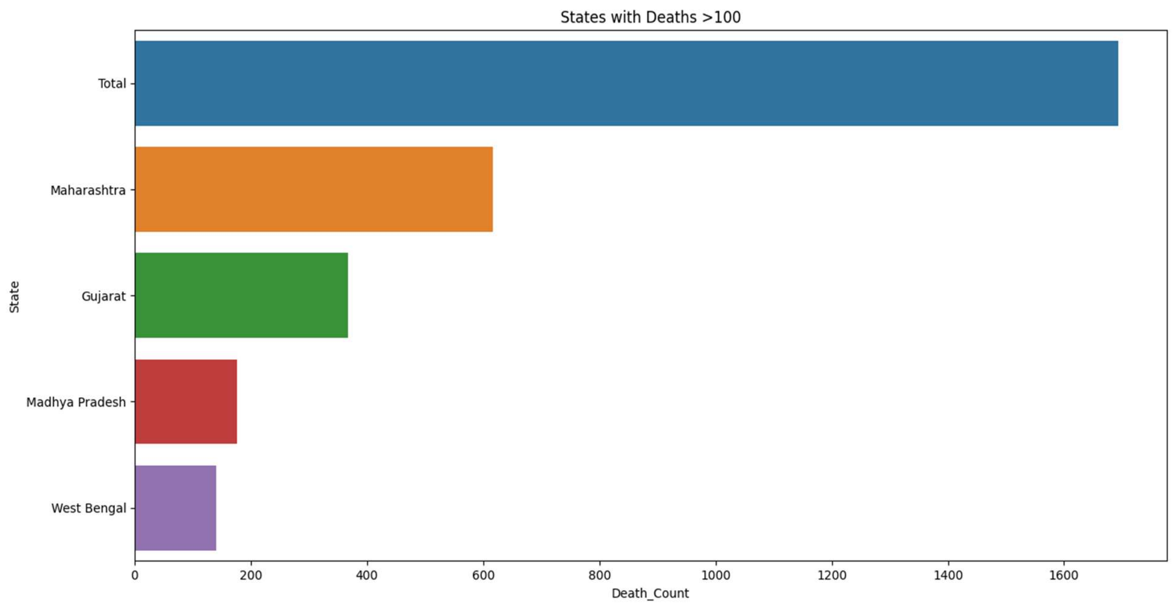
```
cnf_data["Percentage_Recovered"] = cnf_data['Recovered']
/ cnf_data['Confirmed'] * 100

cnf_data['Deaths_Percentage'] = cnf_data['Deaths'] /
cnf_data['Confirmed'] * 100

cnf_data['ActiveCases_Percentage'] =
(cnf_data['Confirmed'] - cnf_data['Recovered'] -
cnf_data['Deaths']) / cnf_data['Confirmed'] * 100

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

The above syntaxes will create new columns for the percentages and include the respective values under them.

```
deaths_greater100 = df3.query("Deaths > 100")
print(deaths_greater100)
deaths_greater100_plot = sns.barplot(x='Deaths',
y='State', data=deaths_greater100)
deaths_greater100_plot.set(xlabel='Death_Count',
ylabel='State')
deaths_greater100_plot.set_title('States with
Deaths>100')
```

The bar plot for the states which has more than 100 deaths is plotted in the next page.

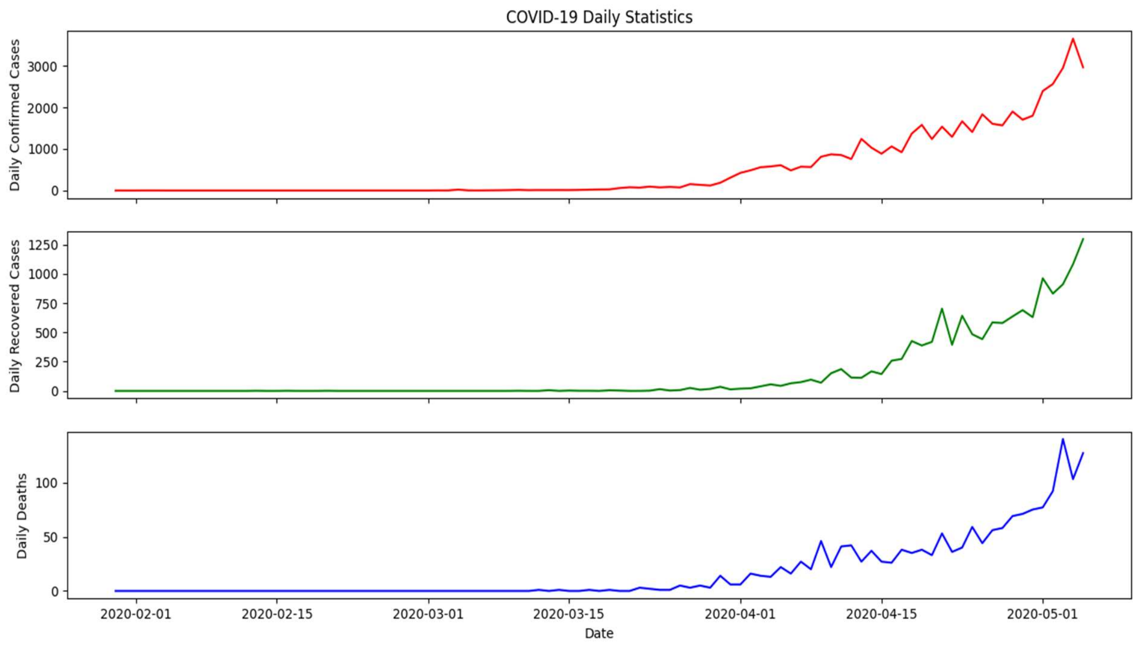
**OUTPUT**



- A subplot was created for the daily registered positive cases, recoveries and fatalities from the dataframe-4 using matplotlib.

```
fig, ax = plt.subplots(3, 1, sharex=True)
ax[0].plot(df4['Date'], df4['Daily Confirmed'], c='Red')
ax[1].plot(df4['Date'], df4['Daily Recovered'],c='Green')
ax[2].plot(df4['Date'], df4['Daily Deceased'], c='Blue')
ax[0].set_title('COVID-19 Daily Statistics')
ax[2].set_xlabel("Date")
ax[2].set_ylabel('Daily Deaths')
ax[1].set_ylabel('Daily Recovered Cases')
ax[0].set_ylabel('Daily Confirmed Cases')
plt.show()
```

The sharex will give the same X - scale for all the plots and title for the whole visualization and labels of Y-axis for all the three subplots are given using set_xlabel() and set_ylabel().



---

- The fifth dataframe contains the daily tests data which are carried out by each state.

```
tests_by_state = df5.loc[:, ['State', 'Total Tested']]
tests_by_state = tests_by_state.dropna()
```

Initially a new data frame was created with state and tests as columns and the null value columns are dropped.

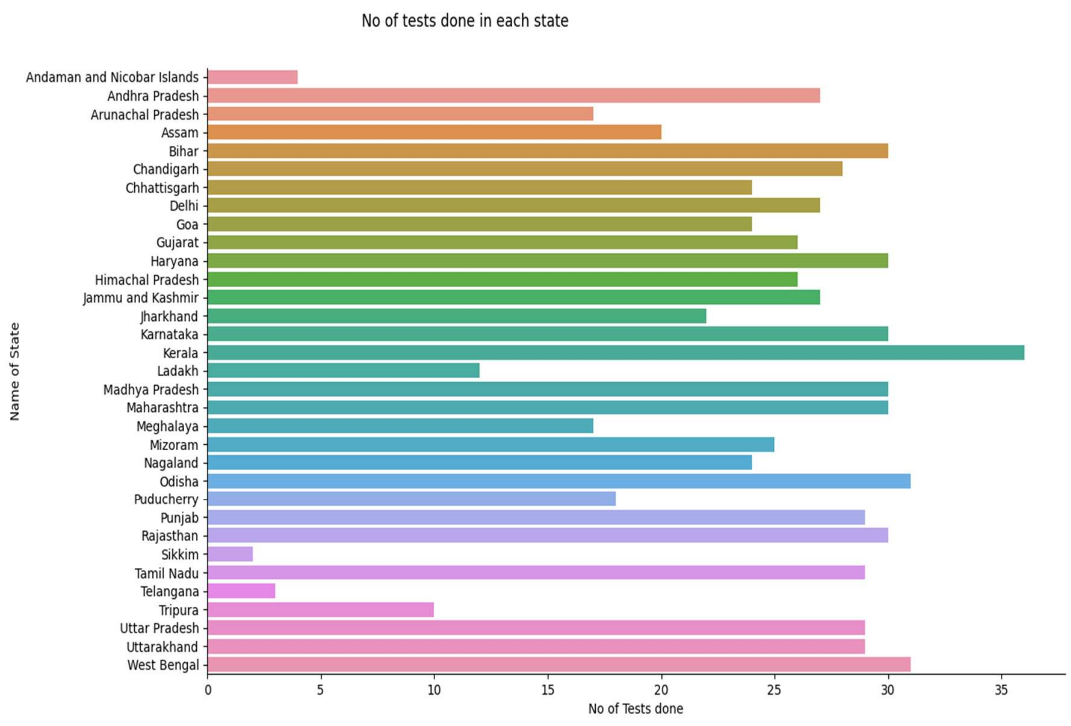
The output had 777 rows displaying the test done in different intervals for each state.

```
total_tests_by_state
=tests_by_state.groupby('State').sum()
print(total_tests_by_state)
```

Now the output dataframe had one row for each state. Now the dataframe has 33 rows.

```
plot = sns.catplot(y='State', kind='count',
data=tests_by_state)
plot.set(xlabel='No of Tests done', ylabel="Name of
State")
plot.fig.suptitle('No of tests done in each state')
plt.show()
```

**OUTPUT**



**PSUEDO CODE**

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

df_main = pd.ExcelFile('COVID19.xlsx')
df1 = pd.read_excel(df_main, 'Raw Data')
df2 = pd.read_excel(df_main, 'Death & Recovered')
df3 = pd.read_excel(df_main, 'State-Wise Data')
df4 = pd.read_excel(df_main, 'Daily Cases Time-Series')
df5 = pd.read_excel(df_main, 'State-Wise Testing Data')

# ---------------------------DF1------------------------------
-----------------#

recovered = df1[df1['Current Status'] == 'Recovered']
pd.set_option('display.max_columns', None)
print(recovered.head(6))


# ---------------------------DF2------------------------------
-----------------#

def Patient_Status_Count(df, col_name):
    status_count = {}
    col = df[col_name]
    for status in col:
        if status in status_count.keys():
            status_count[status] = status_count[status] + 1
        else:
            status_count[status] = 1
    return status_count


result = Patient_Status_Count(df2, 'Patient_Status')
print(result)
sns.set_context('talk')
print(df2.columns)
sns.catplot(x='State', kind='count', hue='Patient_Status',
data=df2)
plt.xticks(rotation=90)

# ---------------------------DF3------------------------------
-----------------#
```

```
andhra_deaths = df3.loc[df3['State'] == 'Andhra Pradesh',
['Deaths']]
print(andhra_deaths)

andhra = df3.query("State == 'Andhra Pradesh'")
print(andhra)

deaths_greater100 = df3.query("Deaths > 100")
print(deaths_greater100)

deaths_greater100_plot = sns.barplot(x='Deaths', y='State',
data=deaths_greater100)
deaths_greater100_plot.set(xlabel='Death_Count',
ylabel='State')
deaths_greater100_plot.set_title('States with Deaths >100')

cnf_data = df3.loc[:, ['State', 'Confirmed', 'Recovered',
'Deaths']]
cnf_data = cnf_data.dropna()
cnf_data["Percentage_Recovered"] = cnf_data['Recovered'] /
cnf_data['Confirmed'] * 100
cnf_data['Deaths_Percentage'] = cnf_data['Deaths'] /
cnf_data['Confirmed'] * 100
cnf_data['ActiveCases_Percentage'] = (cnf_data['Confirmed'] -
cnf_data['Recovered'] - cnf_data['Deaths']) / cnf_data[
    'Confirmed'] * 100
pd.set_option('display.max_columns', None)
print(cnf_data)
print(cnf_data[cnf_data.Percentage_Recovered ==
cnf_data.Percentage_Recovered.max()])

# ---------------------------DF4-------------------------------
-----------------#

fig, ax = plt.subplots(3, 1, sharex=True)
ax[0].plot(df4['Date'], df4['Daily Confirmed'], c='Red')
ax[1].plot(df4['Date'], df4['Daily Recovered'], c='Green')
ax[2].plot(df4['Date'], df4['Daily Deceased'], c='Blue')
ax[0].set_title('COVID-19 Daily Statistics')
ax[2].set_xlabel("Date")
ax[2].set_ylabel('Daily Deaths')
ax[1].set_ylabel('Daily Recovered Cases')
ax[0].set_ylabel('Daily Confirmed Cases')

# ---------------------------DF5-------------------------------
-----------------#

tests_by_state = df5.loc[:, ['State', 'Total Tested']]
tests_by_state = tests_by_state.dropna()
print(tests_by_state)
total_tests_by_state = tests_by_state.groupby('State').sum()
```

```
print(total_tests_by_state)
print(total_tests_by_state.shape)
print(total_tests_by_state.index)
plot = sns.catplot(y='State', kind='count',
data=tests_by_state)
plot.set(xlabel='No of Tests done', ylabel="Name of State")
plot.fig.suptitle('No of tests done in each state')

# -----------------------------------------------------------
-------------#

plt.show()
```

❖ ***Mahesh Srivinay Rayavarapu***