Covid-19 Data Analysis using Python

Title of the project:

Covid-19 Data Analysis using Python.

Description:

Hello everyone!

In this tutorial, we are going to analyze Covid-19 data using Python. We mainly use the plotly and matplotlib libraries for this work.

It works on information related to the confirmed cases, active cases, recovered cases, serious/critical cases and death cases. In particular, we analyze data of top 20 countries' cases and plot information using treemap, pie chart, bar graph and line graph.

Prerequisites:

1) Dataset files of covid cases with a .csv extension.
2) Install Jupyter Notebook or any similar working environment with the latest version of Python installed.
3) Python language.
4) Knowledge of Python libraries like numpy, pandas, matplotlib.

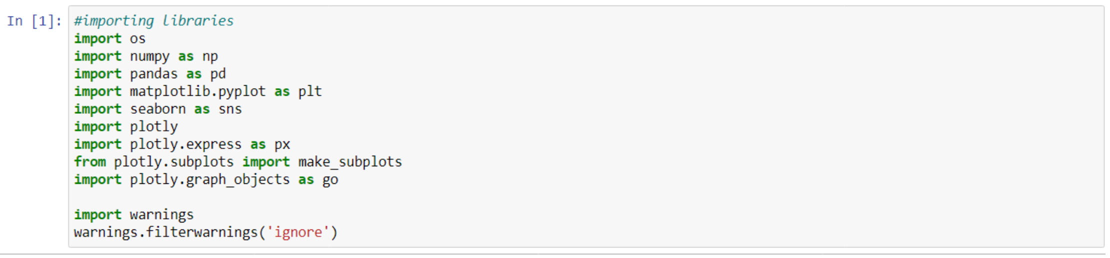
Datasets:

It contains the datasets of-

i. worldometer data, (209, 16)
ii.country wise data, (187, 15)
iii.day wise data, (188, 12)
iv. combined data, (35156, 10)

Implementation:

1) Import the required Python libraries.



```
In [1]: #importing libraries
        import os
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import plotly
        import plotly.express as px
        from plotly.subplots import make_subplots
        import plotly.graph_objects as go

        import warnings
        warnings.filterwarnings('ignore')
```

2) Reading the datasets. It contains datasets of worldometer data, Country wise data, day wise data and combined data. All these datasets are present in .csv extension files.

```
In [2]: # Enter the datasets
        path = 'D:\INTERNSHIP_PROJECTS\Covid-19 Data Analysis\Covid-19_dataset'

        file = os.listdir(path)
        file

Out[2]: ['combined.csv', 'country_wise.csv', 'day_wise.csv', 'worldometer.csv']
```

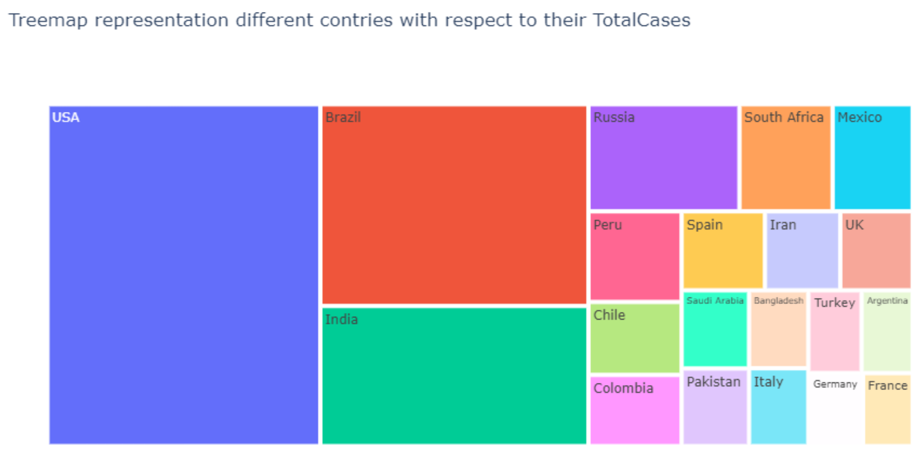
```
In [3]: # Reading the datasets

        def read(path, file):
            return pd.read_csv(path+'/'+file)
```

```
In [4]: # combined dataset

        combined_data = read(path, file[0])
        print(combined_data.shape)
        combined_data.head()

        (35156, 10)

Out[4]:
```

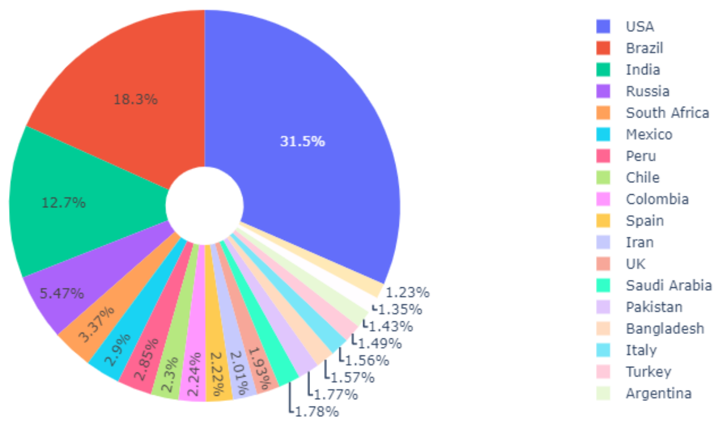
	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	WHO Region
0	2020-01-22	Afghanistan	0	0	0	0	0	0	0	Eastern Mediterranean
1	2020-01-22	Albania	0	0	0	0	0	0	0	Europe
2	2020-01-22	Algeria	0	0	0	0	0	0	0	Africa
3	2020-01-22	Andorra	0	0	0	0	0	0	0	Europe
4	2020-01-22	Angola	0	0	0	0	0	0	0	Africa

3) First, we analyze country-wise information. We obtain information of the countries in terms of total cases, active-cases, recovered cases and death cases . We plot this information using treemap and pie charts.

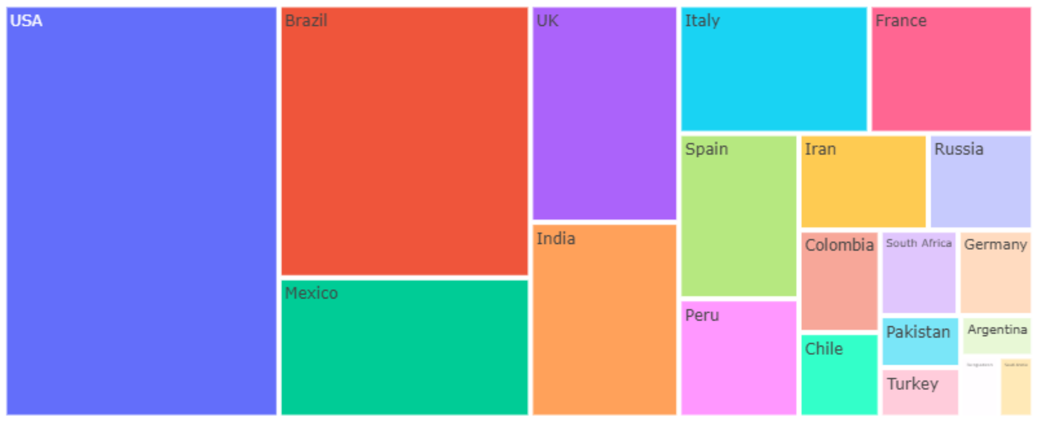
```
In [10]: # representation of data of top 20 countries
         cases = ['TotalCases','TotalDeaths','TotalRecovered', 'ActiveCases']
         labels = world_dataset[0:20]['Country/Region'].values
         top_20_total_cases = world_dataset.iloc[0:20]

         #treemap and pie chart
         for i in range(len(cases)):
             fig1 = px.treemap(top_20_total_cases, values = cases[i], path = ['Country/Region'],
                               title = 'Treemap representation different contries with respect to their {}'.format(cases[i]))
             fig1.show()
             fig2 = px.pie(top_20_total_cases, values = cases[i], names=labels, hole = 0.2,
                          title = "Pie chart representation top 20 different contries with respect to their {}".format(cases[i]))

             fig2.show()
```



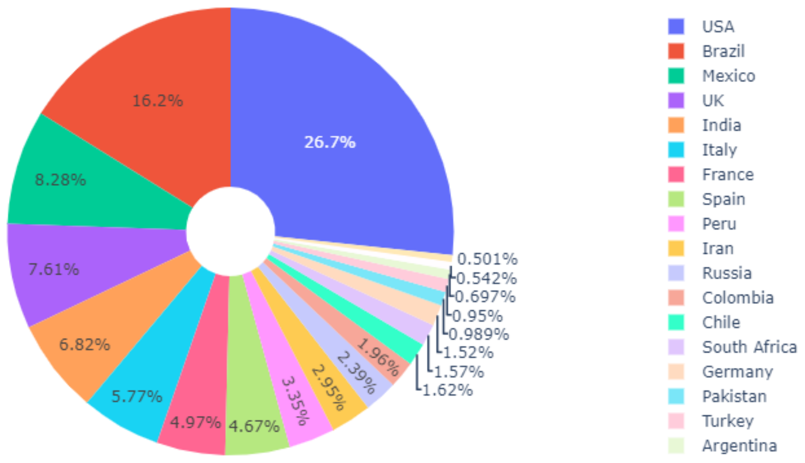
Pie chart representation top 20 different contries with respect to their TotalCases



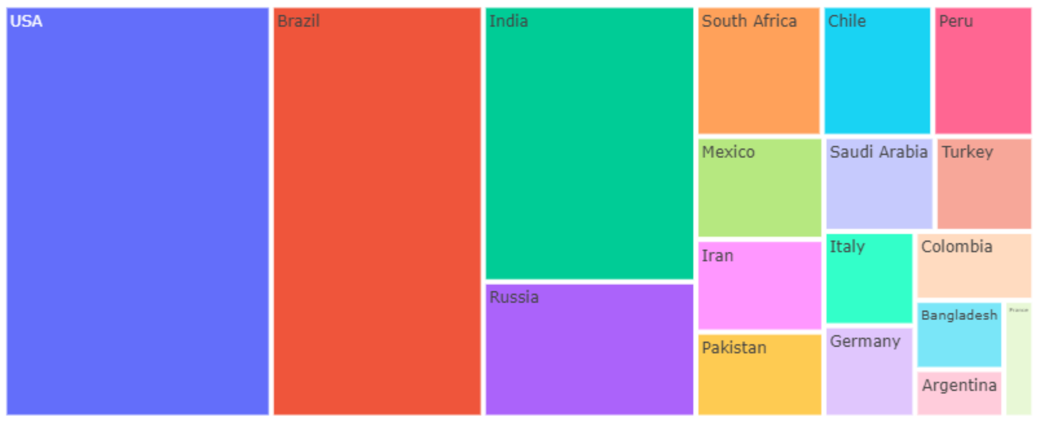
Treemap representation different contries with respect to their TotalDeaths



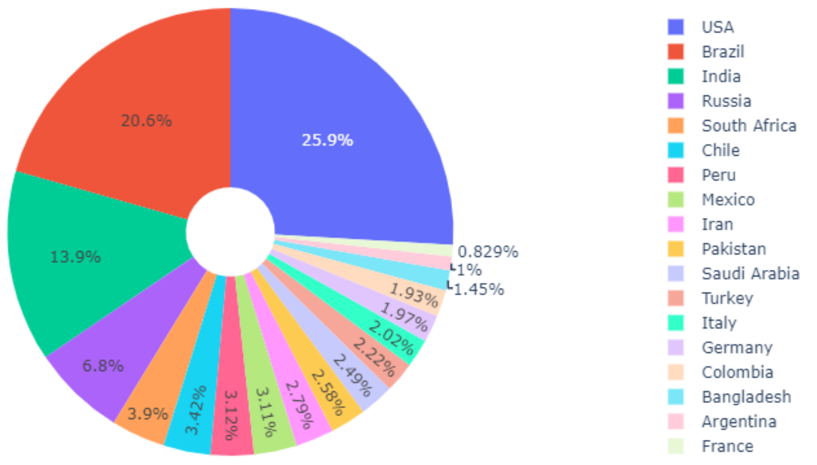
Pie chart representation top 20 different contries with respect to their TotalDeaths



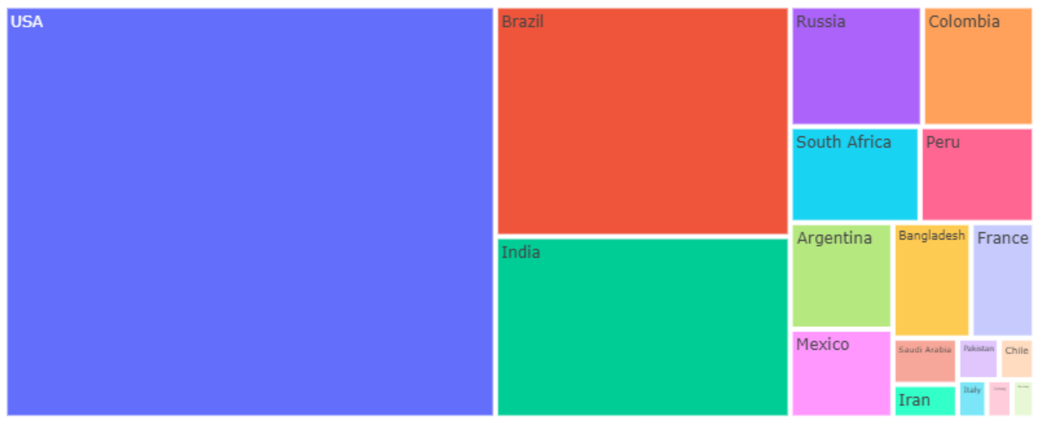
Treemap representation different contries with respect to their TotalRecovered

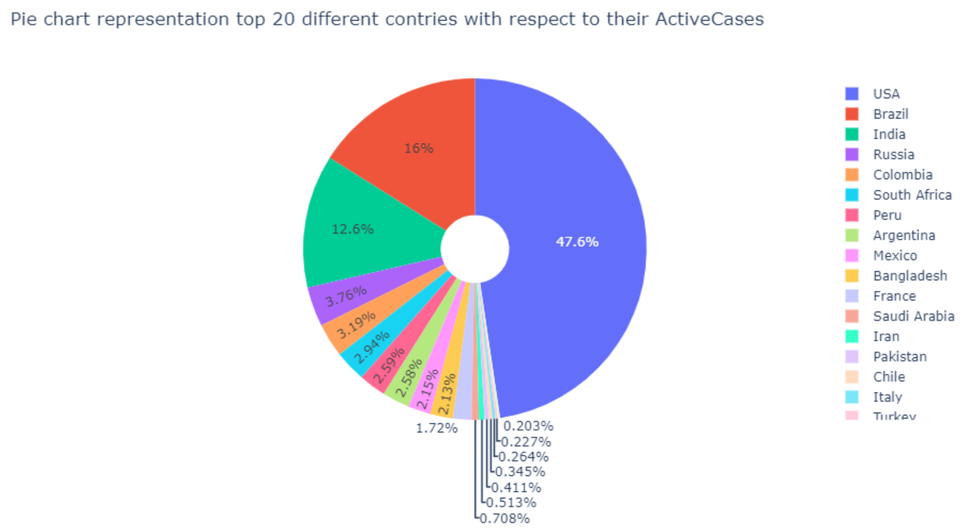


Pie chart representation top 20 different contries with respect to their TotalRecovered



Treemap representation different contries with respect to their ActiveCases





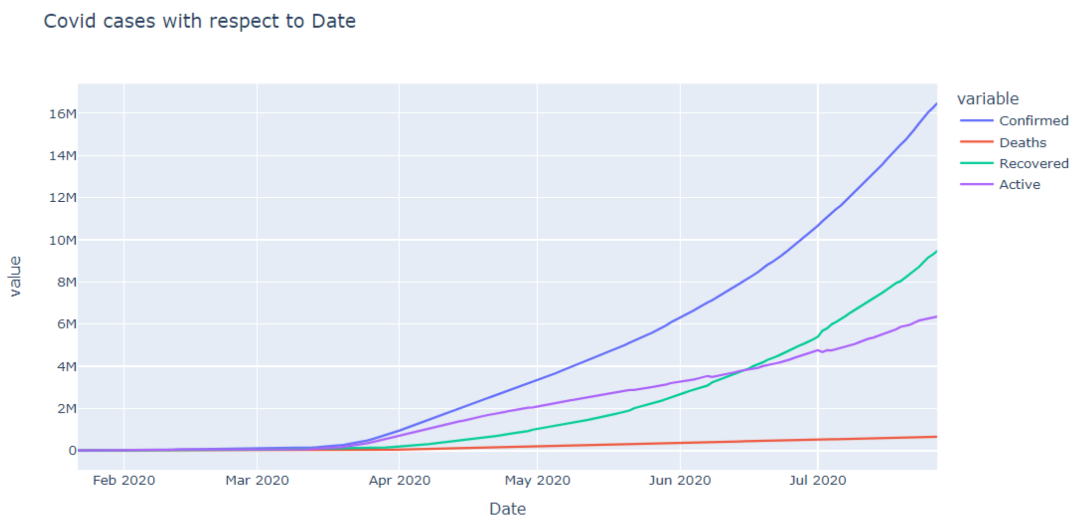
4) After that, we analyze day-wise information. It includes information of confirmed cases, active cases, recovered cases and death cases . We plot this information using a line plot.

```
In [11]: day_wise_data.head()
```

Out[11]:

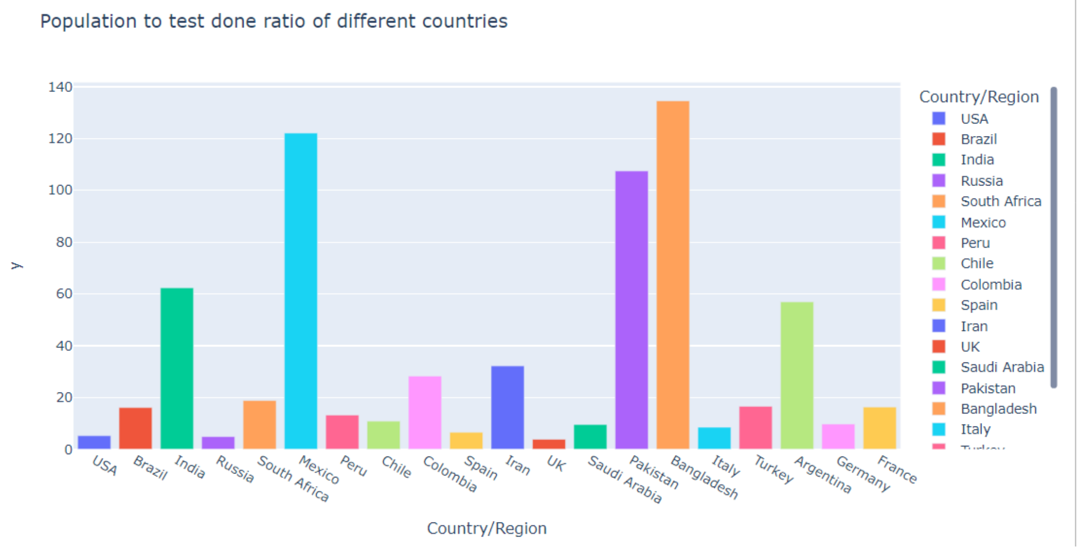
	Date	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	Deaths / 100 Cases	Recovered / 100 Cases	Deaths / 100 Recovered	No. of countries
0	2020-01-22	555	17	28	510	0	0	0	3.06	5.05	60.71	6
1	2020-01-23	654	18	30	606	99	1	2	2.75	4.59	60.00	8
2	2020-01-24	941	26	36	879	287	8	6	2.76	3.83	72.22	9
3	2020-01-25	1434	42	39	1353	493	16	3	2.93	2.72	107.69	11
4	2020-01-26	2118	56	52	2010	684	14	13	2.64	2.46	107.69	13

```
In [13]: cases = ['Confirmed', 'Deaths', 'Recovered', 'Active']

         fig3 = px.line(day_wise_data, x = 'Date', y = cases, title='Covid cases with respect to Date')
         fig3.show()
```



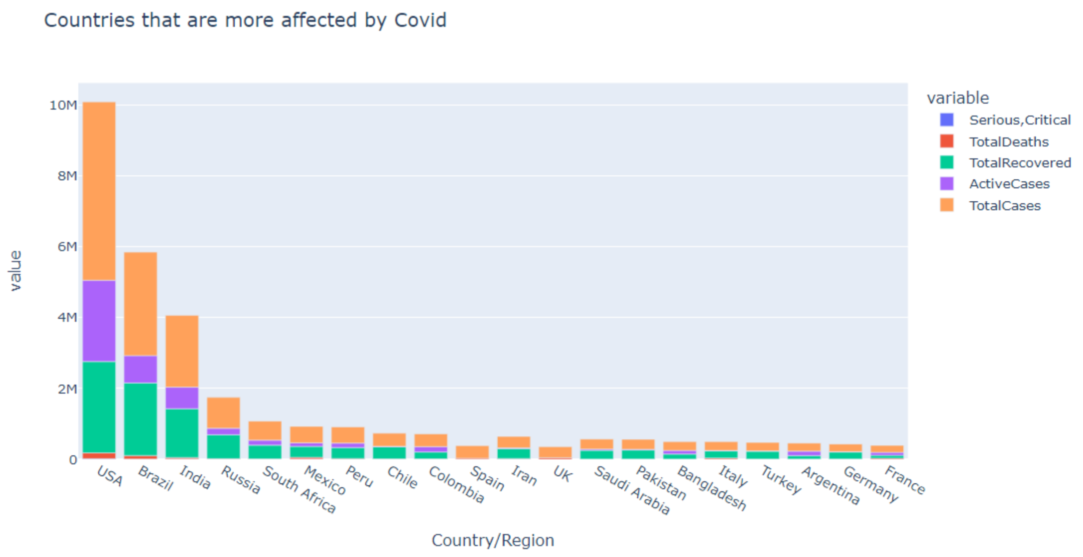
5) Next, we calculate the ratio between population and test done.

```
In [15]: population_to_test_ratio = world_dataset['Population']/world_dataset['TotalTests'].iloc[0:20]

         fig4 = px.bar(world_dataset.iloc[0:20], x = 'Country/Region', y = population_to_test_ratio[0:20],
                      color = 'Country/Region', title = 'Population to test done ratio of different countries')
         fig4.show()
```



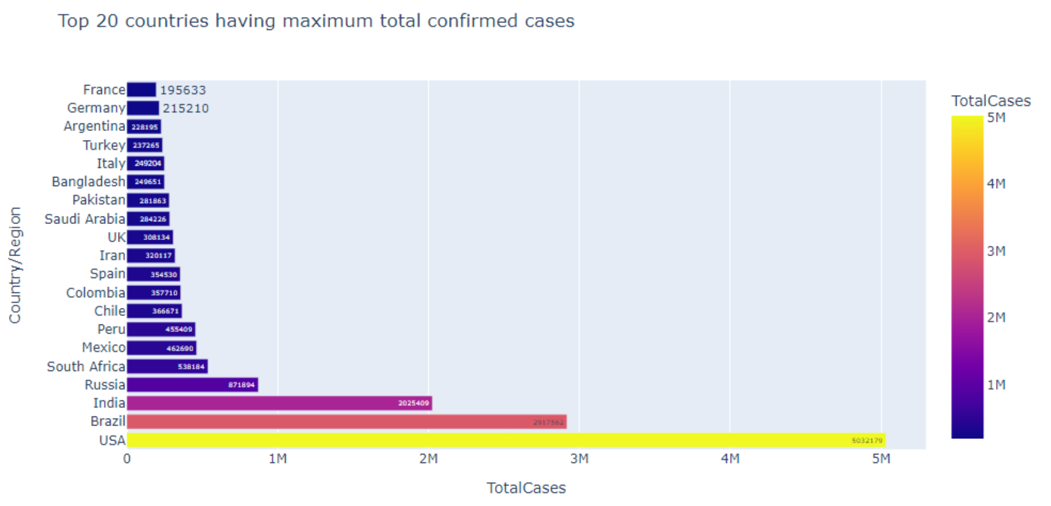
6) Now, we check for the top 20 countries in terms of max total confirmed cases, max total active cases, max total recovered cases, max total deaths and serious critical condition cases.

```
In [17]: cases = ['Serious,Critical','TotalDeaths', 'TotalRecovered','ActiveCases','TotalCases']

         max_20_total_cases = world_dataset.iloc[0:20]
         fig5 = px.bar(max_20_total_cases, x = 'Country/Region', y = cases, title='Countries that are more affected by Covid')
         fig5.show()
```



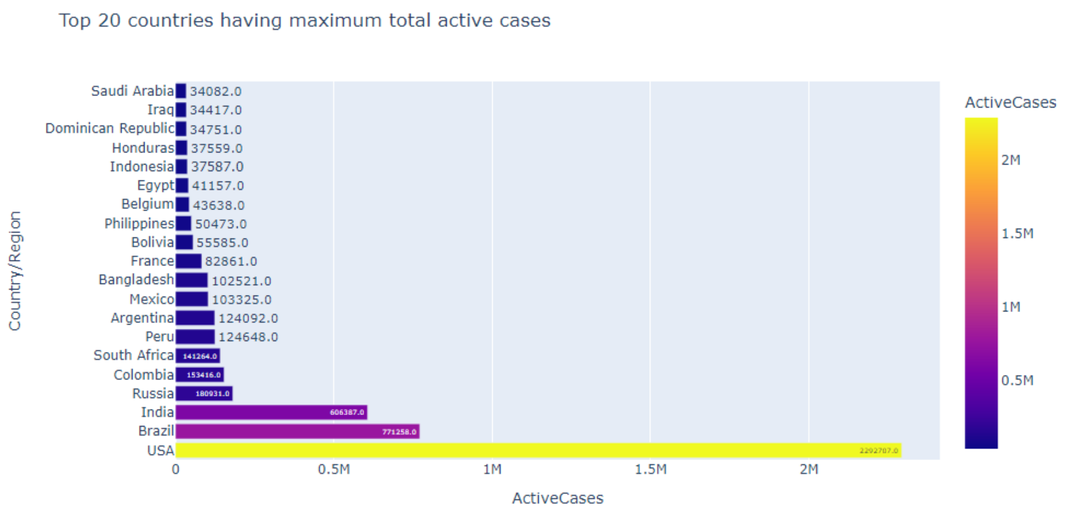
i) Top 20 countries having maximun total confirmed cases

```
In [18]: max_20_confirmed_cases = world_dataset.iloc[0:20]

         fig6 = px.bar(max_20_confirmed_cases, y ='Country/Region', x = 'TotalCases', color='TotalCases',
                       text = 'TotalCases', title = 'Top 20 countries having maximum total confirmed cases')
         fig6.show()
```



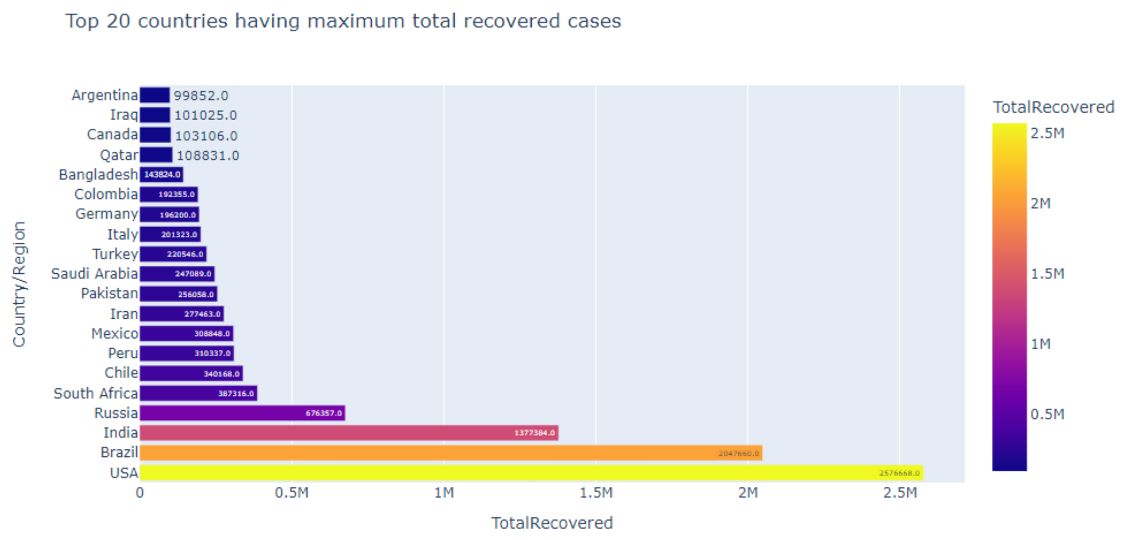
ii) Top 20 countries having maximum total active cases

```
In [19]: max_20_active_cases = world_dataset.sort_values(by='ActiveCases', ascending=False).iloc[0:20]

         fig7 = px.bar(max_20_active_cases, y ='Country/Region', x = 'ActiveCases', color='ActiveCases',
                       text = 'ActiveCases', title = 'Top 20 countries having maximum total active cases')
         fig7.show()
```



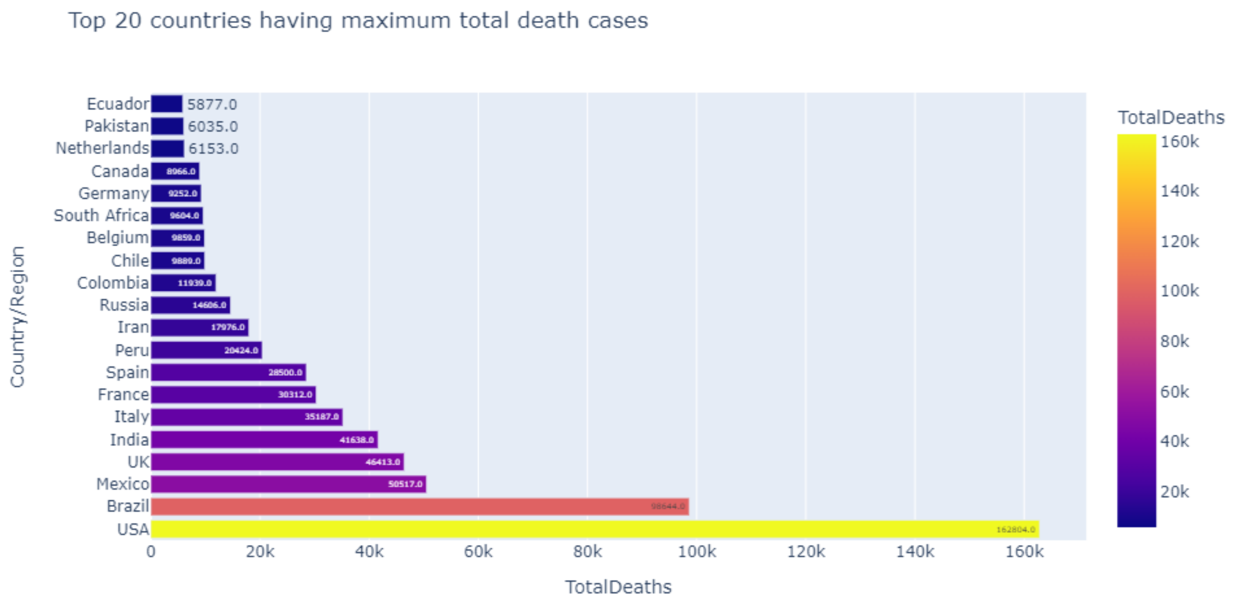
iii) Top 20 countries having maximum total recovered cases

```
In [20]: max_20_recovered_data = world_dataset.sort_values(by='TotalRecovered', ascending=False).iloc[0:20]

         fig8 = px.bar(max_20_recovered_data, y ='Country/Region', x = 'TotalRecovered', color='TotalRecovered',
                       text = 'TotalRecovered', title = 'Top 20 countries having maximum total recovered cases')
         fig8.show()
```



iv) Top 20 countries having maximum total death cases

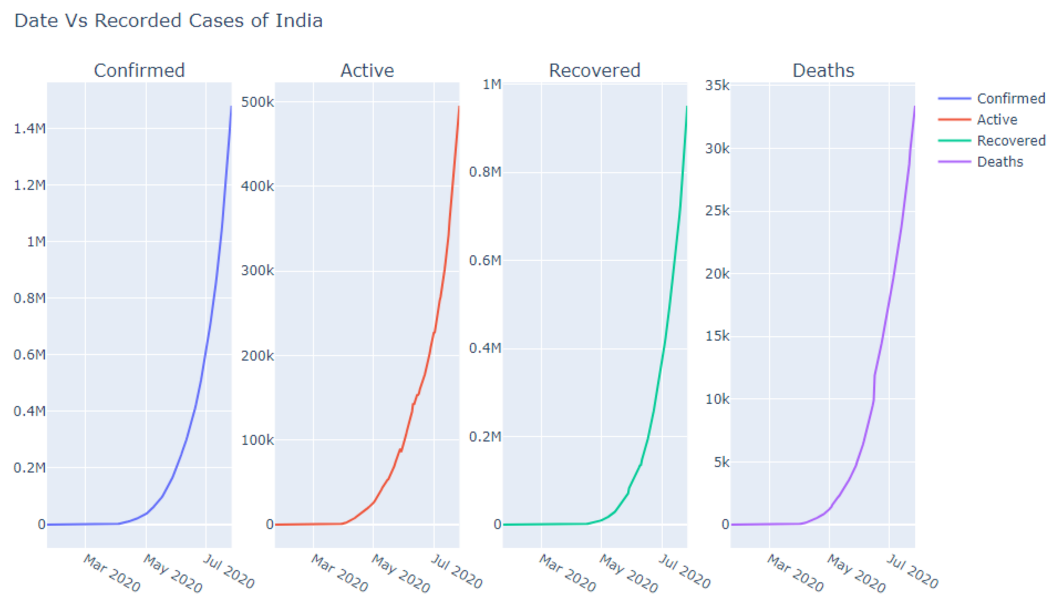
```
In [21]: max_20_total_deaths = world_dataset.sort_values(by='TotalDeaths', ascending=False).iloc[0:20]

         fig9 = px.bar(max_20_total_deaths, y ='Country/Region', x = 'TotalDeaths', color='TotalDeaths',
                       text = 'TotalDeaths', title = 'Top 20 countries having maximum total death cases')
         fig9.show()
```



7) At last, we analyze information for a particular country. We can choose any country in the world. It provides information about confirmed, active, recovered and death cases along with dates. We plot this information using a line graph.

```
In [23]: def country_information(combined_data,country):

             data=combined_data[combined_data['Country/Region']==country]
             df=data.loc[:,['Date','Confirmed','Deaths','Recovered','Active']]

             fig10 = make_subplots(rows=1, cols=4,subplot_titles=("Confirmed", "Active", "Recovered",'Deaths'))

             fig10.add_trace(go.Scatter(name="Confirmed",x=df['Date'],y=df['Confirmed']), row=1, col=1)
             fig10.add_trace(go.Scatter(name="Active",x=df['Date'],y=df['Active']), row=1, col=2)
             fig10.add_trace(go.Scatter(name="Recovered",x=df['Date'],y=df['Recovered']), row=1, col=3)
             fig10.add_trace(go.Scatter(name="Deaths",x=df['Date'],y=df['Deaths']), row=1, col=4)

             fig10.update_layout(height=600, width=1000, title_text="Date Vs Recorded Cases of {}".format(country))
             fig10.show()
```

```
In [24]: country_information(combined_data,'India')
```


Date Vs Recorded Cases of India
Confirmed
Active
Recovered
Deaths
0.2M
0.4M
0.6M
0.8M
1M
1.2M
1.4M
100k
200k
300k
400k
500k
0.2M
0.4M
0.6M
0.8M
1M
5k
10k
15k
20k
25k
30k
35k
Mar 2020
May 2020
Jul 2020
Confirmed
Active
Recovered
Deaths