

Covid-19 Data Analysis using Python

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')
```

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:\INTERSHIP_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]:
```

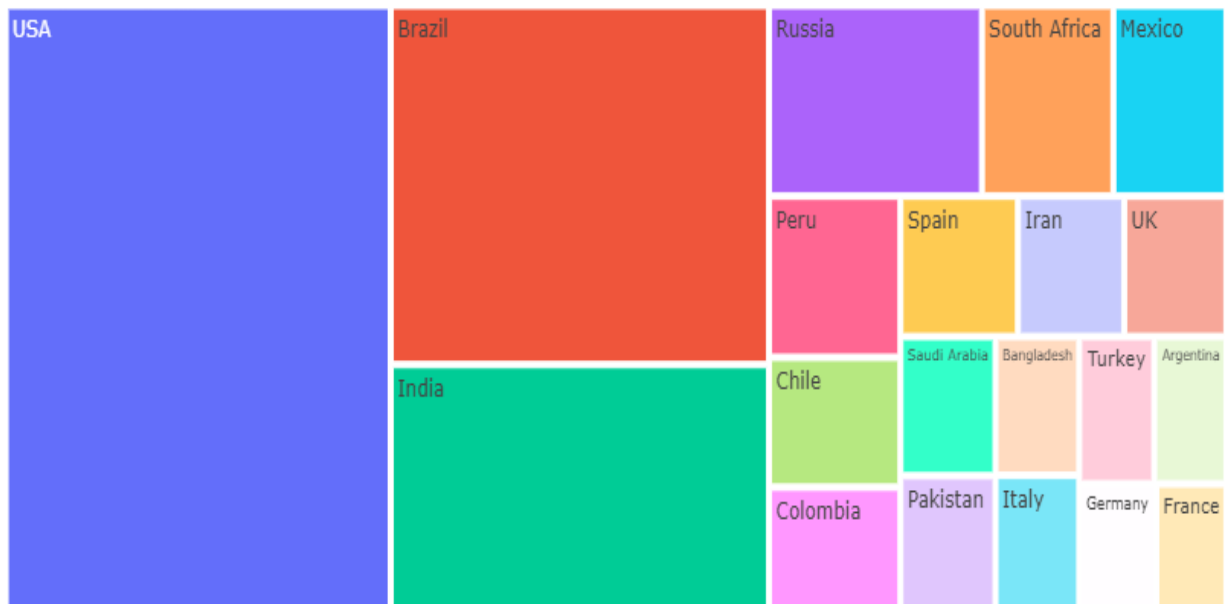
	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	WHO Region
0	2020-01-22	Afghanistan	0	0	0	0	0	0	0	Eastern Mediterranean
1	2020-01-22	Albania	0	0	0	0	0	0	0	Europe
2	2020-01-22	Algeria	0	0	0	0	0	0	0	Africa
3	2020-01-22	Andorra	0	0	0	0	0	0	0	Europe
4	2020-01-22	Angola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

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.

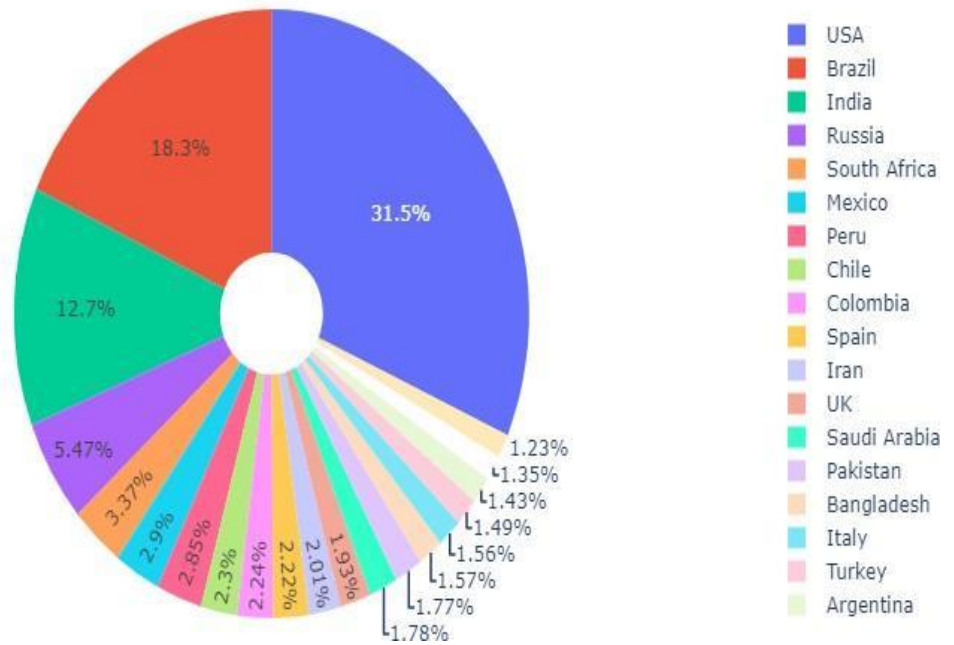
```
In [10]: # representation of data of top 20 countries
cases = ['TotalCases', 'TotalDeaths', 'TotalRecovered', 'ActiveCases']
labels = world_dataset[0:20]['Country/Region'].values
top_20_total_cases = world_dataset.iloc[0:20]

#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()
```

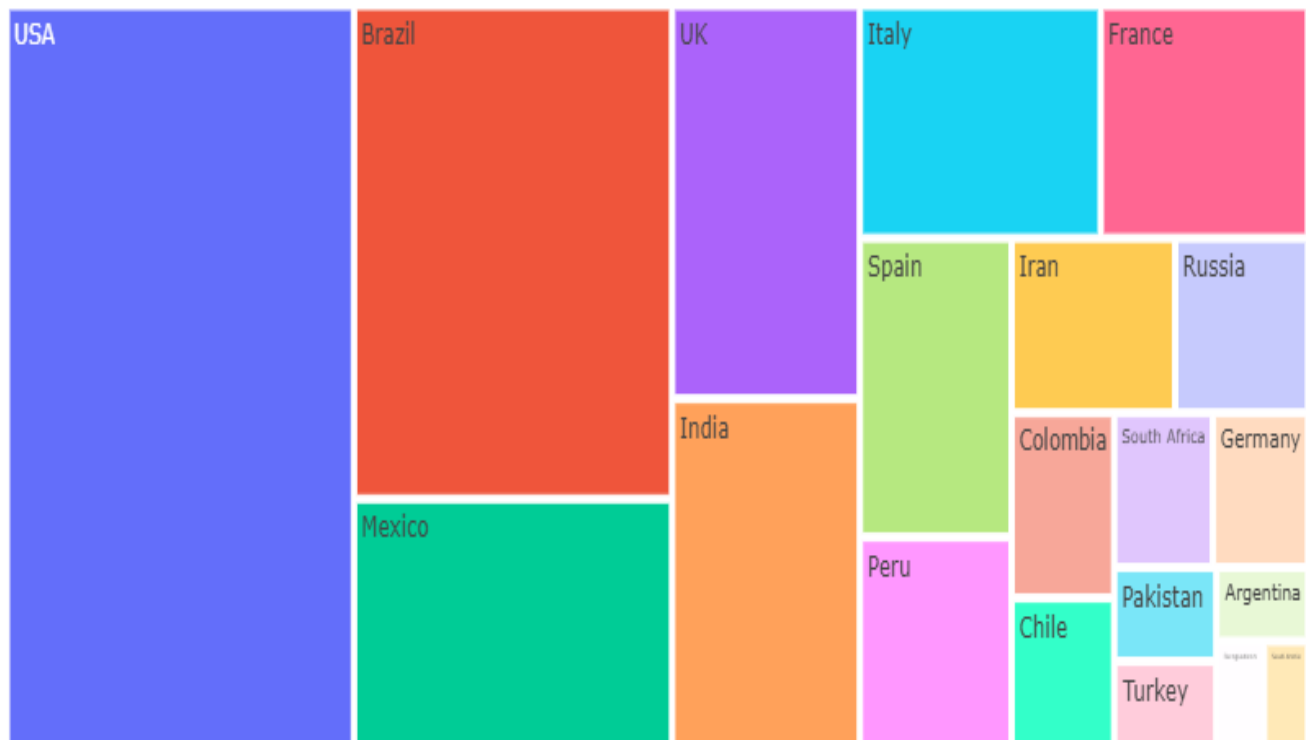
Treemap representation different contries with respect to their TotalCases



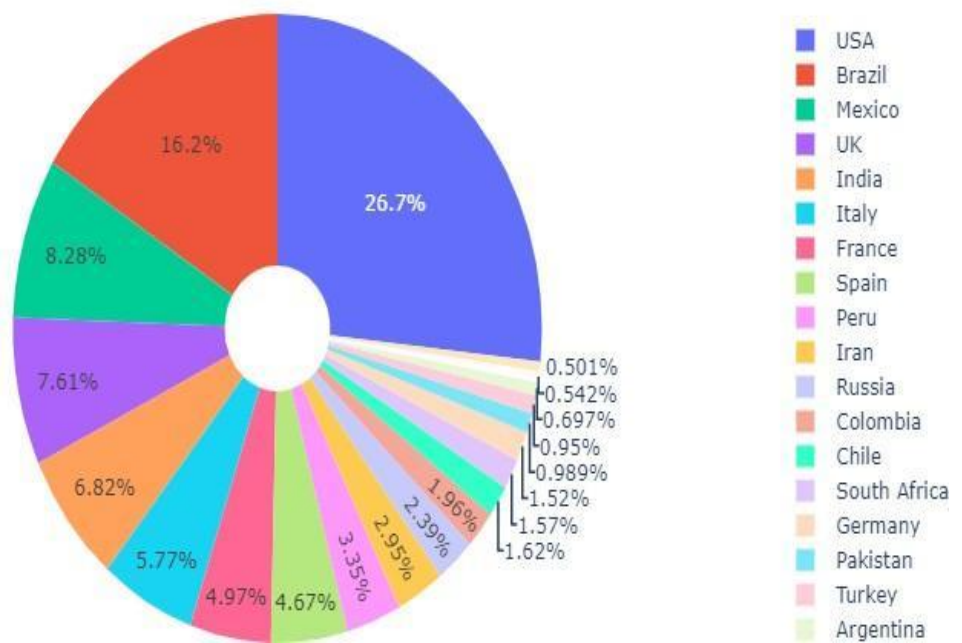
Pie chart representation top 20 different contries with respect to their TotalCases



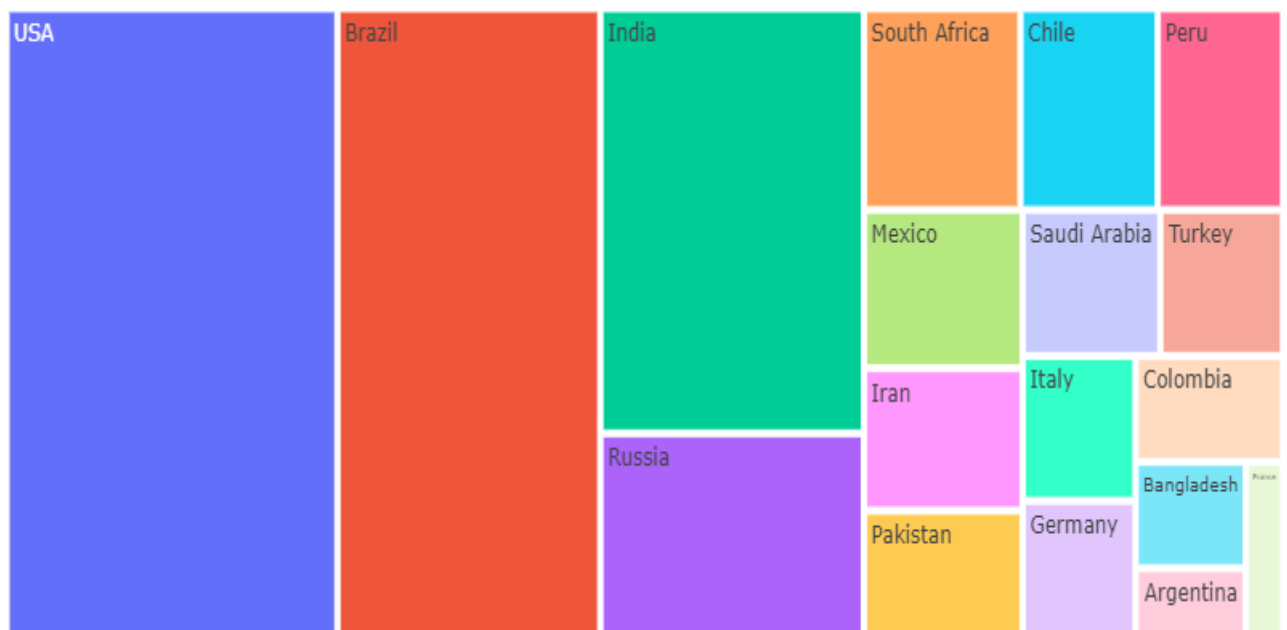
Treemap representation different contries with respect to their TotalDeaths



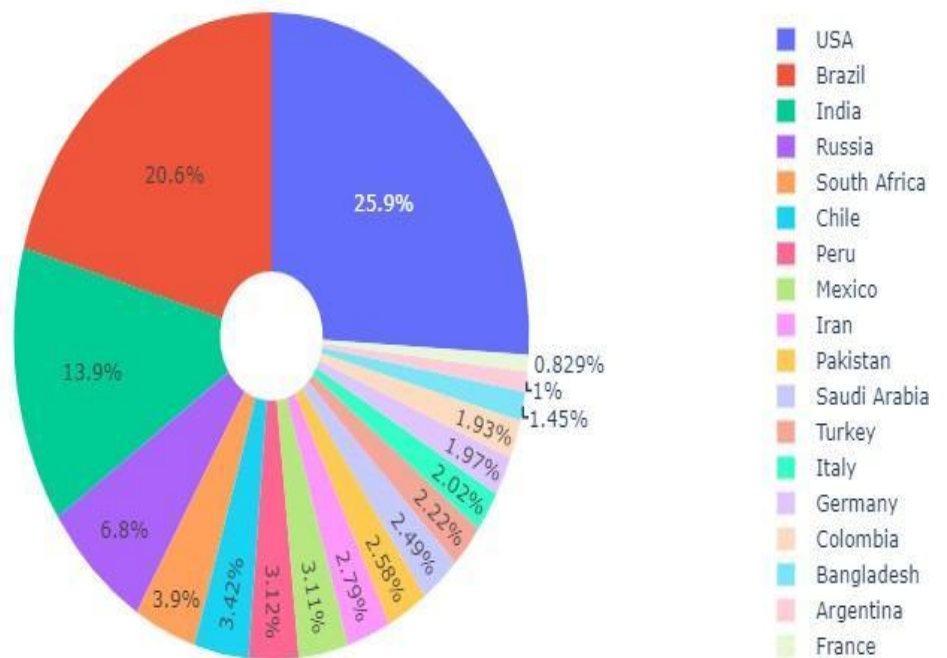
Pie chart representation top 20 different contries with respect to their TotalDeaths



Treemap representation different contries with respect to their TotalRecovered



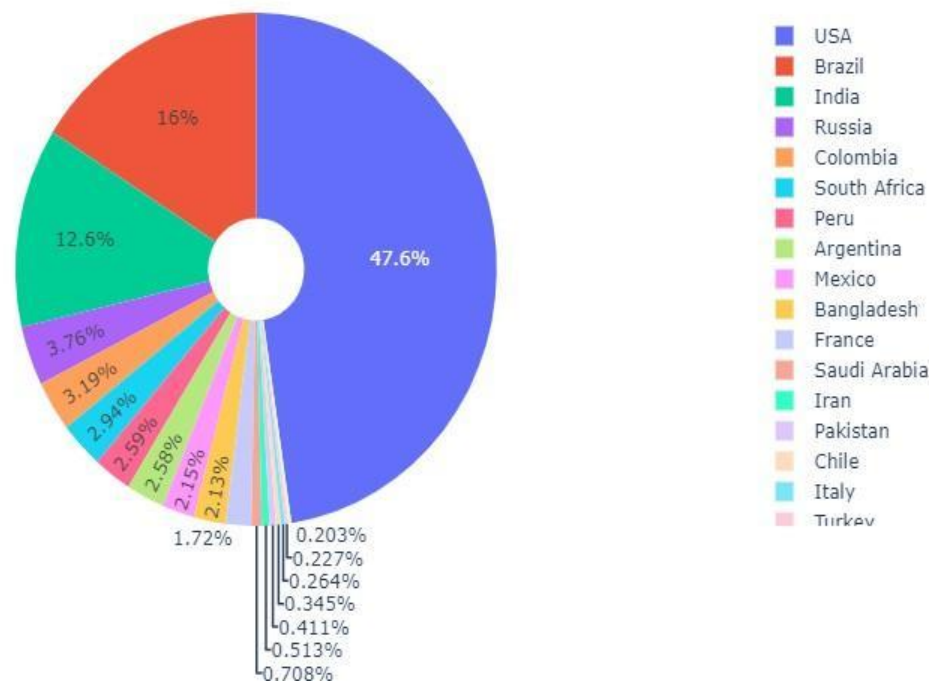
Pie chart representation top 20 different contries with respect to their TotalRecovered



Treemap representation different contries with respect to their ActiveCases



Pie chart representation top 20 different countries with respect to their ActiveCases

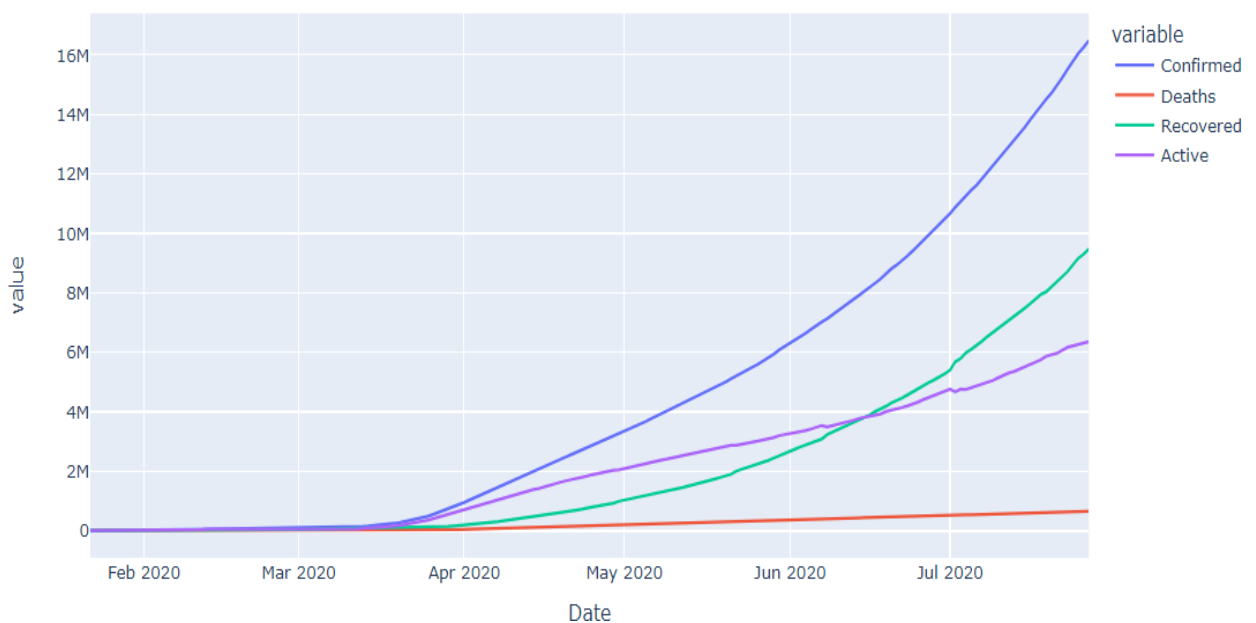


After that, we analyze day-wise information. It includes information of confirmed cases, active cases, recovered cases and death cases . We plot this information

```
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()
```

Covid cases with respect to Date



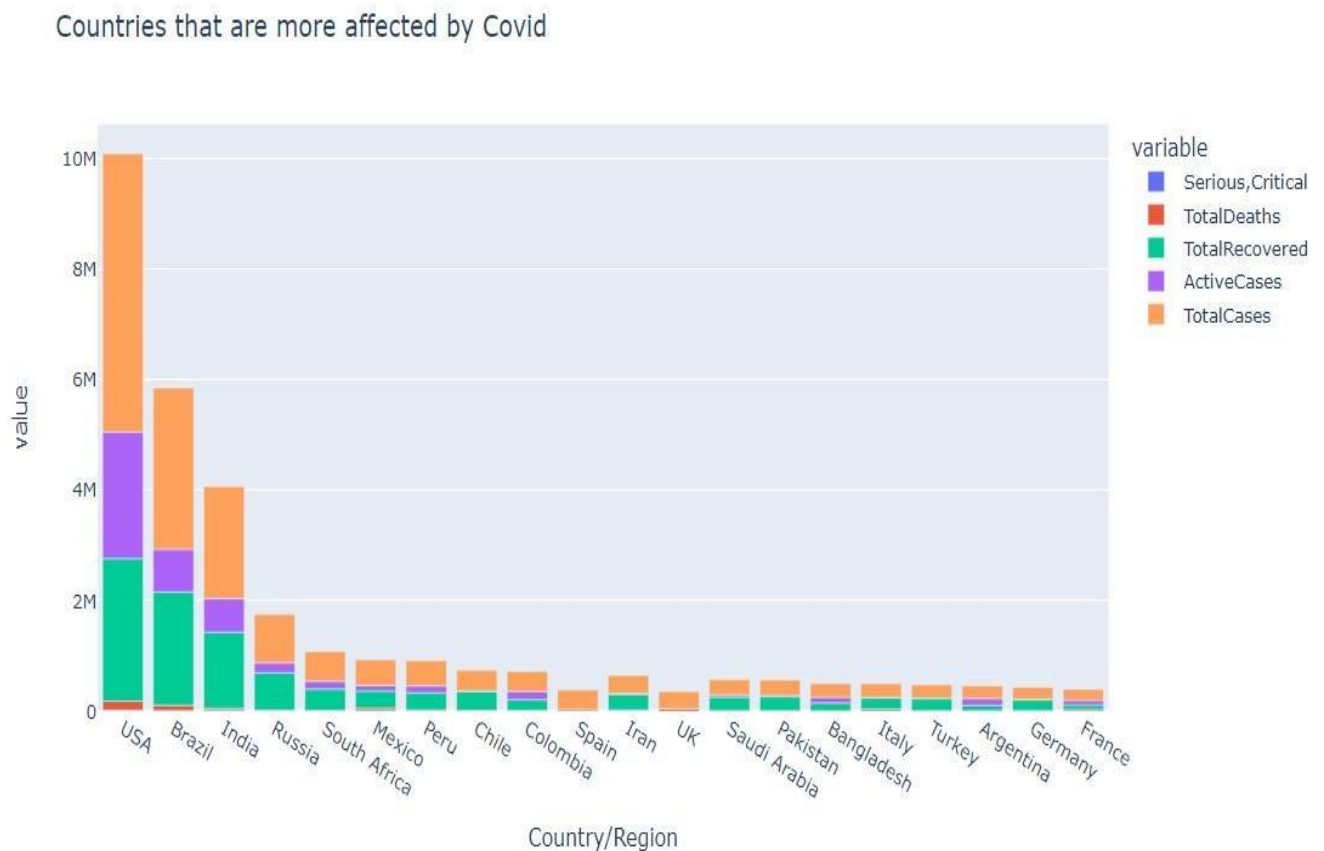
Next, we calculate the ratio between population and test done.

```
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()
```

```
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()
```



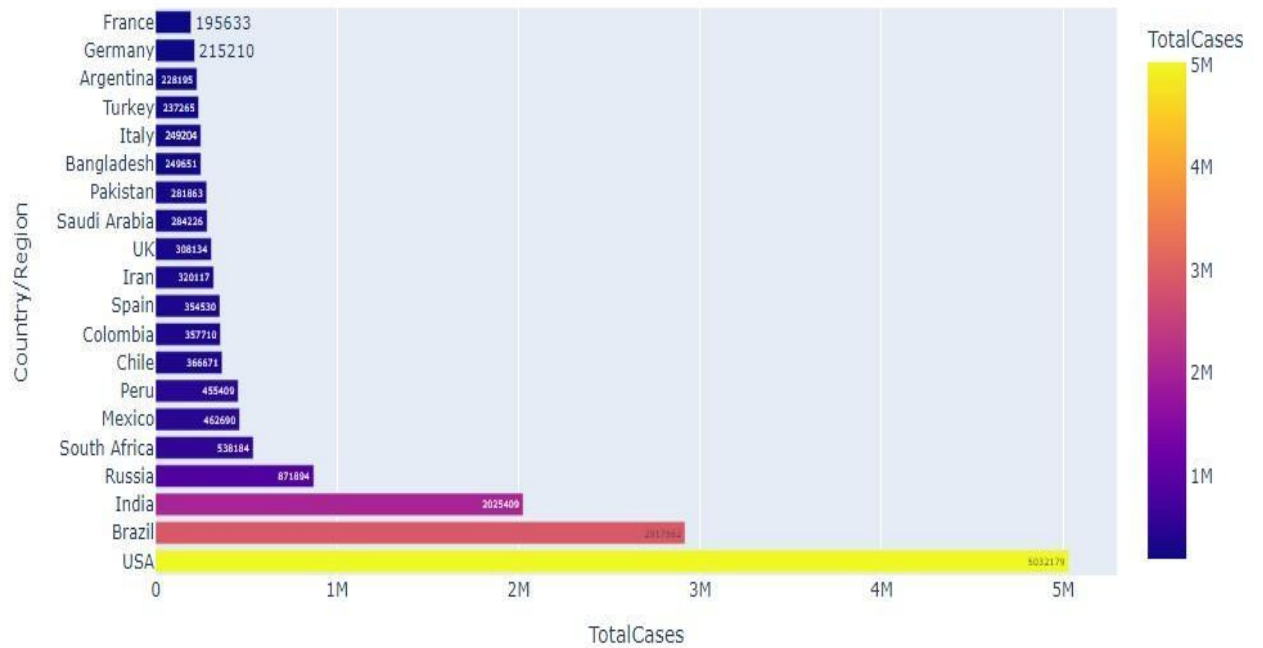
Now, we check for the top 20 countries in terms of max total confirmed cases, max total active cases, max total recovered cases, max total deaths and serious critical condition cases.

i) Top 20 countries having maximum 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()
```

Top 20 countries having maximum total confirmed cases

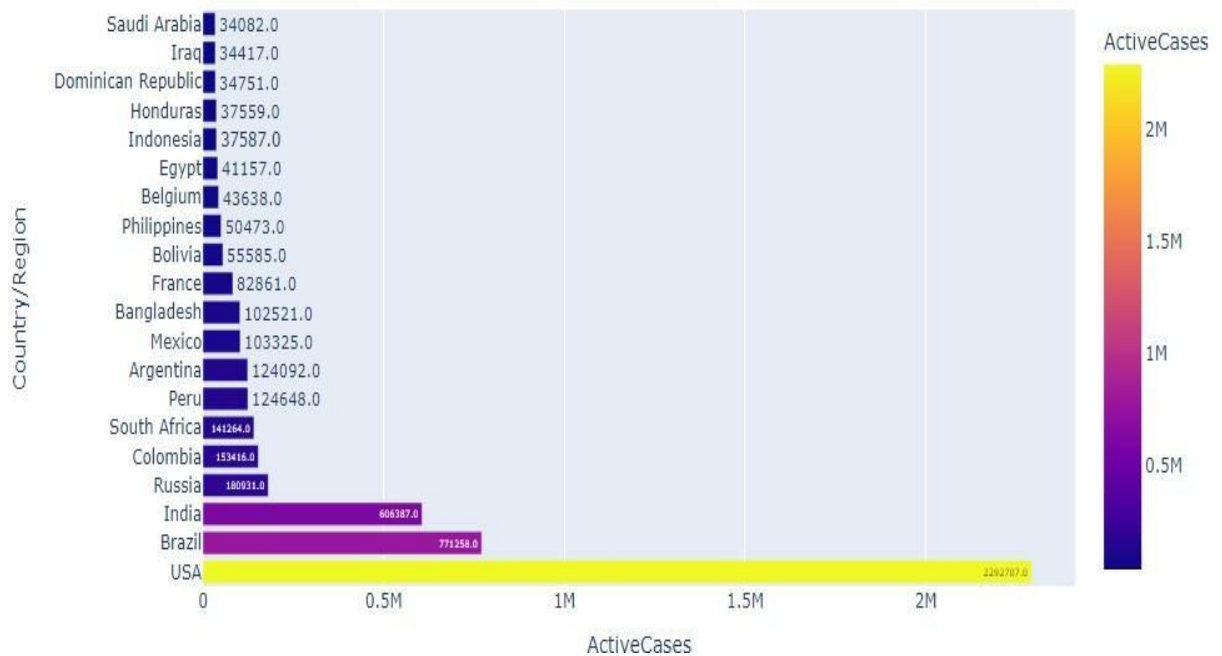


ii) Top 20 countries having maximum total active cases

```
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()
```

Top 20 countries having maximum total active cases

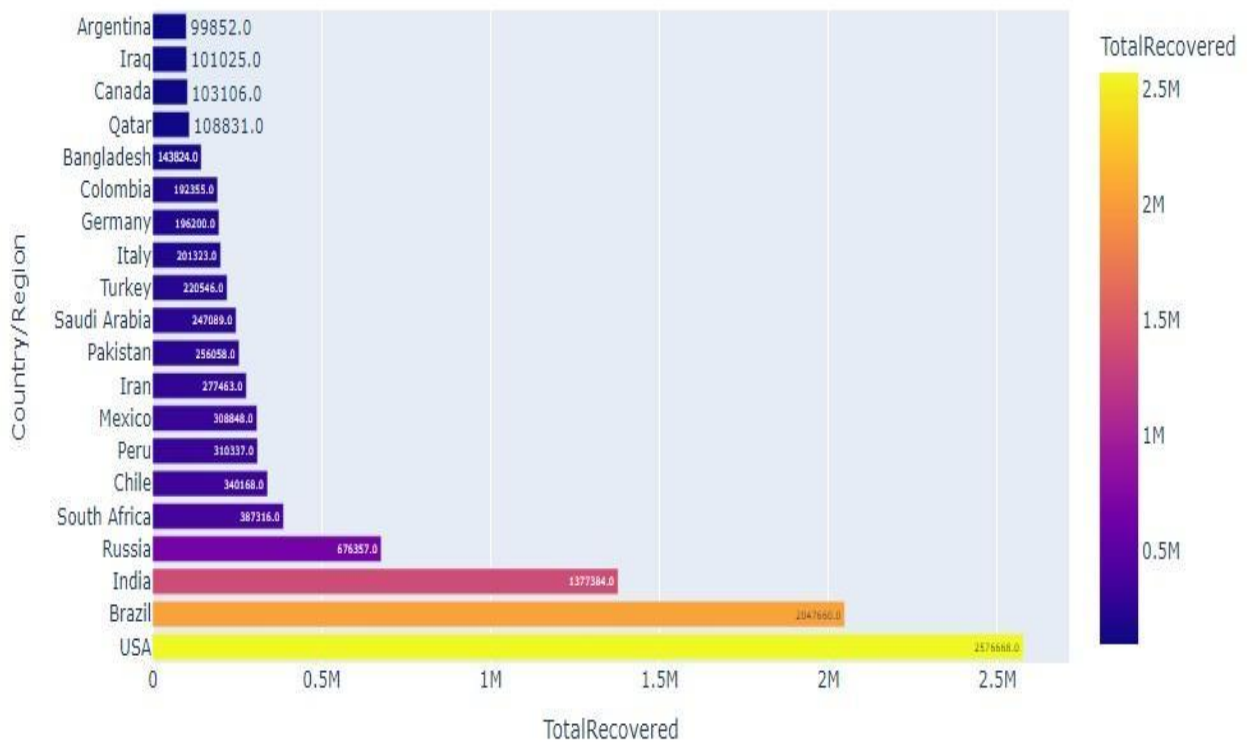


iii) Top 20 countries having maximum total recovered cases

```
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()
```

Top 20 countries having maximum total recovered cases

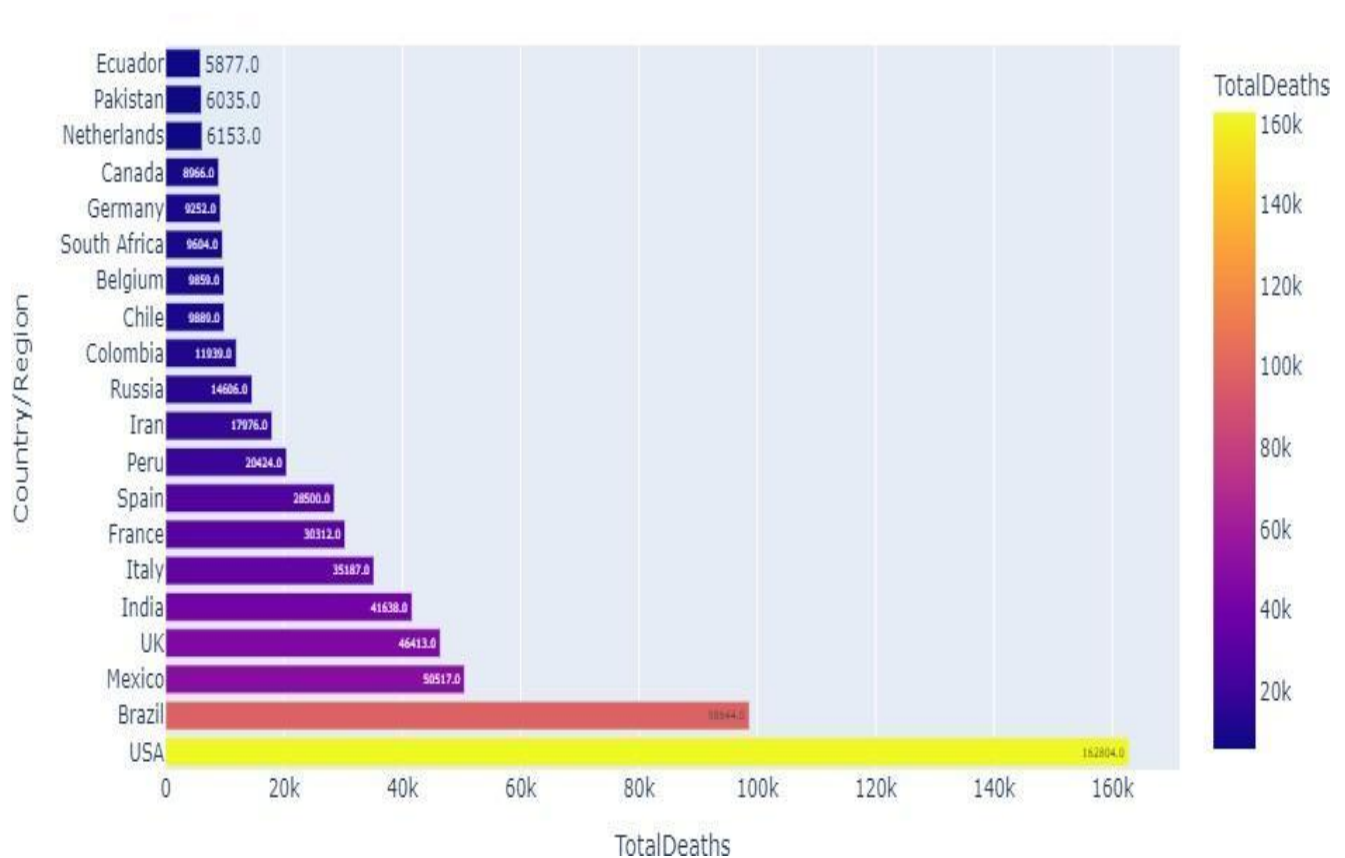


iv) Top 20 countries having maximum total death cases

```
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()
```

Top 20 countries having maximum total death cases



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

