SCHOOL OF COMPUTER ENGINEERING

Winter Semester -2021

PROJECT REPORT

CSE3020 – DATA VISUALIZATION

COVID-19 DATA ANALYSIS AND VISUALIZATION	
TEAM NO : 13	
18BEC0349	V.VINITH REDDY
18BEC0370	G.SAI NARENDER RAO

Submitted to

Dr. Archana Tamizharasan

SCOPE

Project at a glance

No of objectives considered: Data Understanding, Data Preparation, Exploratory Analysis, Validation, Visualization & Presentation.	
Language used: Python	
Statistical Measured used: Pearson's correlation, Box plot.	
Library(s) used: Numpy, pandas, seaborn, pycountry, plotly, bar_chart_race, matplotlib.	
Total no of Visuals created:	< **29** >
No of Individual Chart types used:	< **12** >
Bar Chart	**< 2 >**
Scatter Plot	**< 2 >**
Pie Chart	**< 2 >**
Tree Map	**< 1 >**
Stacked Bar	**< 1 >**
Box Plot	**< 1 >**
Pearson's correlation	**< 1 >**
Area chart	**< 6 >**
Line Chart	**< 8 >**
Word Cloud	**< 2 >**
Geo Map	**< 2 >**
Bar Race Chart	**< 1 >**
Total Charts in project	**< 29 >**

1.1 Project Statement

Since its first identification in December 2019 in Wuhan, China, this virus has taken the world by storm. And spread globally, causing thousands of deaths and having an enormous impact on our health systems and economies. To analyze the cases received daily in a country and their total cases, daily new cases, active cases, total deaths, new deaths and going to summarize the current knowledge about the epidemiology by utilizing different plots with our parameters and visualizing the data progress of this pandemic from various views and perspectives.

1.2 Project Objective

By using the dataset from kaggle website, plotting the various plots understanding the overview of the dataset parameters and visualizing the dataset by required conditions and understanding the plots. And making a simple way of analyzing the dataset.

1.3 Modules

Dataset collection:

In this step the dataset is collected from the kaggle website. In this dataset we have 218 countries which are represented in this data. All of countries have records dating from 2020-2-15 until 2021-05-23 (464 days per country). That's with the exception of China, which has records dating from 2020-1-22 until 2021-05-23 (488 days per country).

Summary Data Columns Description:

- **country**: designates the Country in which the row's data was observed.
- **continent**: designates the Continent of the observed country.
- **total_confirmed**: designates the total number of confirmed cases in the observed country.
- **total_deaths**: designates the total number of confirmed deaths in the observed country.
- **total_recovered**: designates the total number of confirmed recoveries in the observed country.
- **active_cases**: designates the number of active cases in the observed country.
- **serious_or_critical**: designates the estimated number of cases in serious or critical conditions in the observed country.
- **total_cases_per_1m_population**: designates the number of total cases per 1 million populations in the observed country.
- **total_deaths_per_1m_population**: designates the number of total deaths per 1 million populations in the observed country.
- **total_tests**: designates the number of total tests done in the observed country.
- **total_tests_per_1m_population**: designates the number of total test done per 1 million populations in the observed country.
- **population**: designates the population count in the observed country.

Daily Data Columns Description:

- **date**: designates the date of observation of the row's data in YYYY-MM-DD format.
- **country**: designates the Country in which the row's data was observed.
- **cumulative_total_cases**: designates the cumulative number of confirmed cases as of the row's date, for the row's country.
- **daily_new_cases**: designates the daily new number of confirmed cases on the row's date, for the row's country.
- **active_cases**: designates the number of active cases (i.e., confirmed cases that still didn't recover nor die) on the row's date, for the row's country.
- **cumulative_total_deaths**: designates the cumulative number of confirmed deaths as of the row's date, for the row's country.
- **daily_new_deaths**: designates the daily new number of confirmed deaths on the row's date, for the row's country.

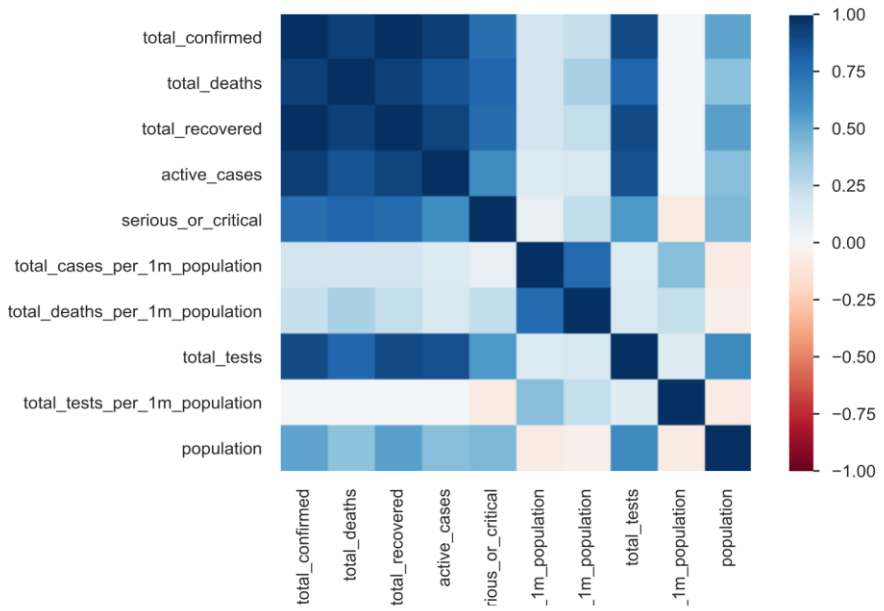
Dataset:https://www.kaggle.com/josephassaker/covid19-global-dataset

Proposed approach:

Here we have 2 data sets describing the situation of the pandemic worldwide. So we are using these data sets and modify them according to the needs of prebuilt functions and libraries in python to visualize data and to see if there are any relation ships between the parameters involved and where the covid-19 is highest and how other parameters are effecting and finding the recovery and deaths of countries and continents and making it easy to grasp the situation using different plots like pie chart , correlation matrix , tree maps and bar charts etc. we tried to find patterns such as the surge of cases in India now which has been said as a second wave of corona virus and in other countries . we tried to find any major outliers for covid which can be used to find the cause of outliers and try to minimize the damage by taking precautions.

1.4 Code with Visuals

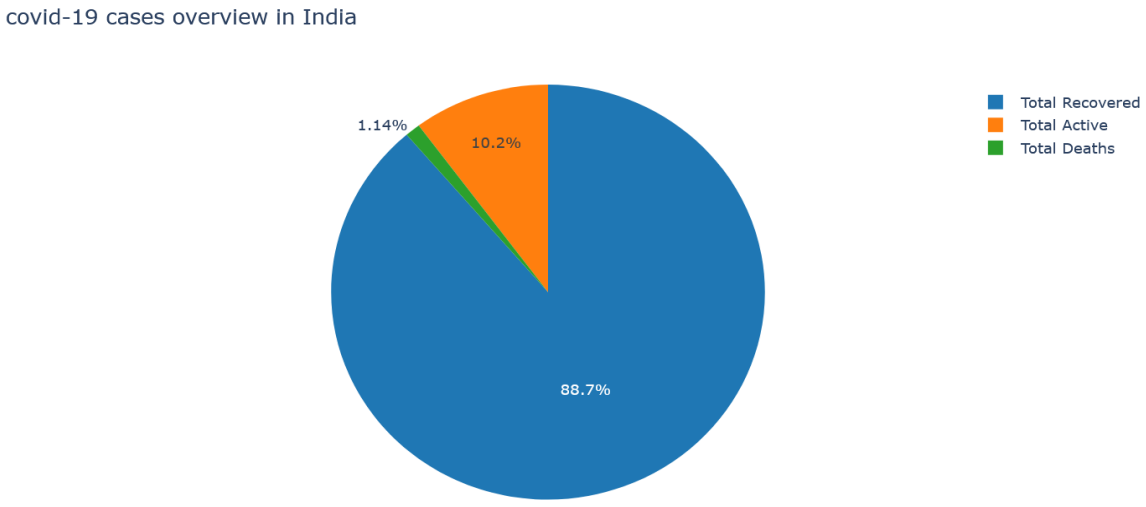
1. Correlation matrix



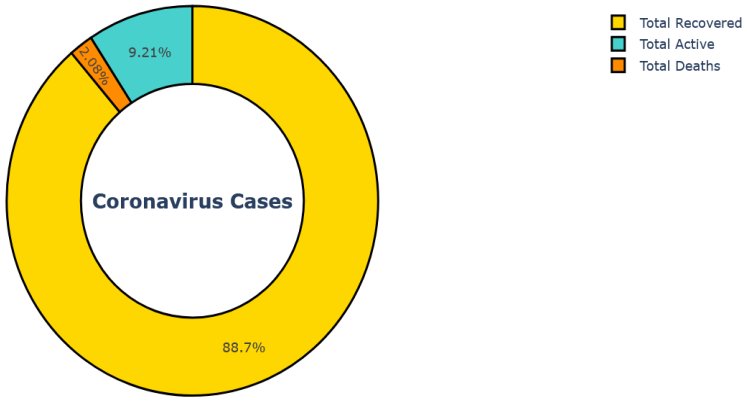
2. Pie Chart

```
def country(topic : str) :
    params =['Total Recovered', 'Total Active', 'Total Deaths']
    values =[df1[df1.country == topic].total_recovered.sum(), df1[df1.country == topic].active_cases.sum(), df1[df1.
    fig = px.pie(df1, values=values,title='covid-19 cases overview in '+topic, names=params, color_discrete_sequence
    fig.show()
```

```
country('India')
```



```
trace = go.Pie(labels=['Total Recovered', 'Total Active', 'Total Deaths'],
               values=[df1.total_recovered.sum(), df1.active_cases.sum(), df1.total_deaths.sum()],
               title="<b>Coronavirus Cases</b>",
               title_font_size=18,
               hovertemplate="<b>%{label}</b><br>%{value}<br><i>%{percent}</i>",
               #hoverinfo='percent+value+label',
               textinfo='percent',
               textposition='inside',
               hole=0.6,
               showlegend=True,
               marker=dict(colors=['gold', 'mediumturquoise', 'darkorange'],
                           line=dict(color='#000000',
                                     width=2),
                          ),
               name=""
              )
fig=go.Figure(data=[trace])
fig.show()
```

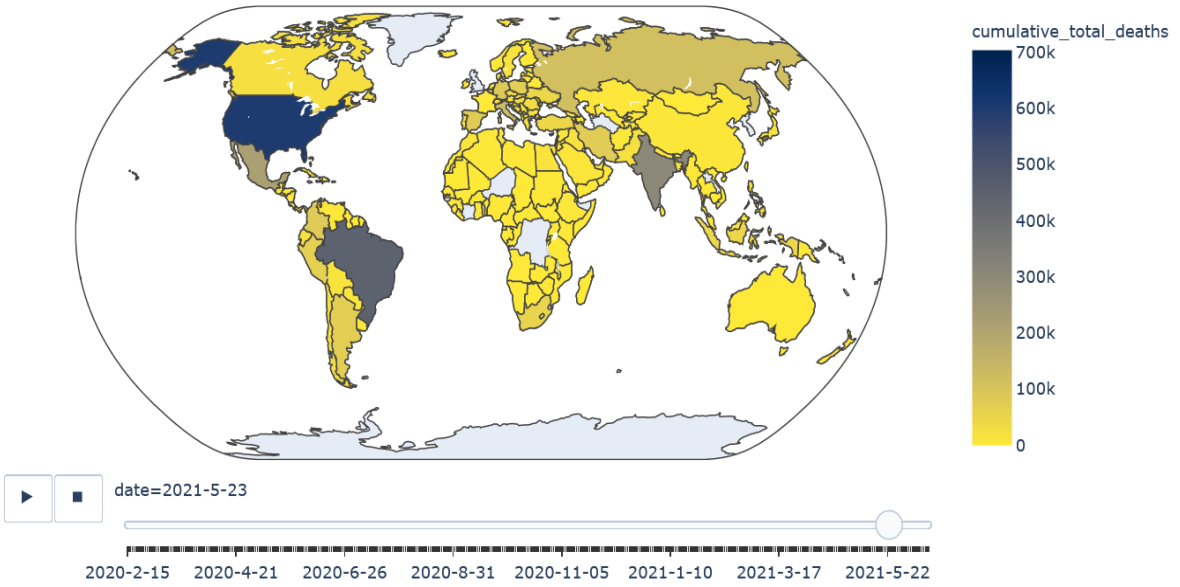


3. Geo Map

```
def geomap(topic: str):
    fig = px.choropleth(
        df,                             # Input Dataframe
        locations="iso_alpha",            # identify country code column
        color=topic,                      # identify representing column
        hover_name="country",# identify hover name
        animation_frame="date",
        color_continuous_scale= px.colors.sequential.Cividis_r,
        projection="natural earth",        # select projection
        range_color=[0,(df[topic].max())+100000],
        title='<span style="font-size:36px; font-family:Times New Roman"> '+ " ".join(topic.split('_')) +' per count
    )             # select range of dataset
    fig.layout.updatemenus[0].buttons[0].args[1]["frame"]["duration"] = 0.5
    fig.layout.updatemenus[0].buttons[0].args[1]["transition"]["duration"] = 0.5
    fig.layout.coloraxis.showscale = True
    fig.layout.sliders[0].pad.t = 10
    fig.layout.updatemenus[0].pad.t= 10
    return fig
```

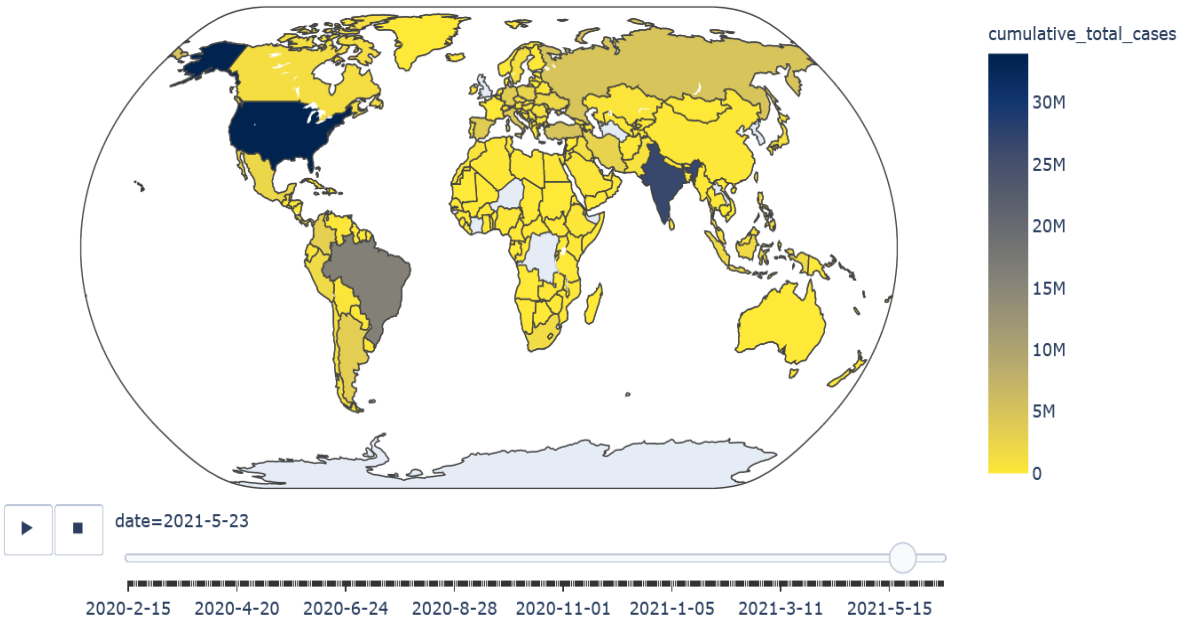
```
geomap('cumulative_total_deaths')
```

cumulative total deaths per country




```
geomap('cumulative_total_cases')
```

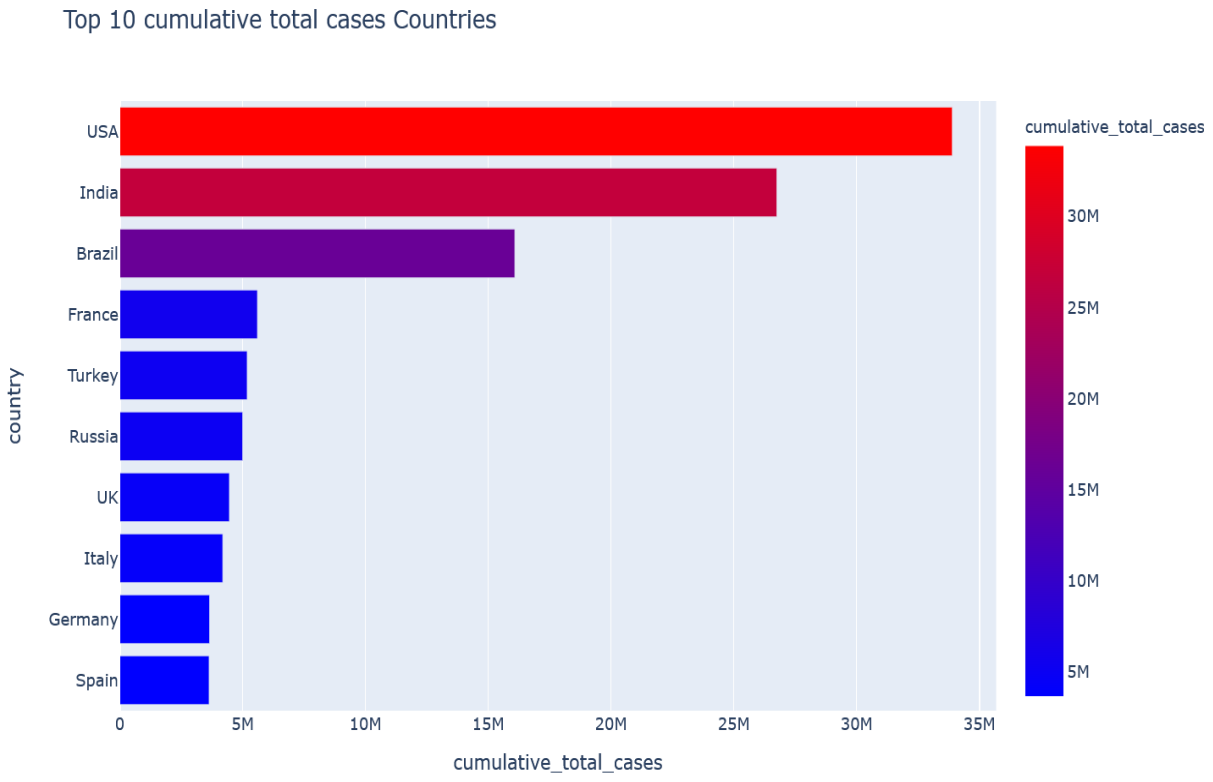
cumulative total cases per country



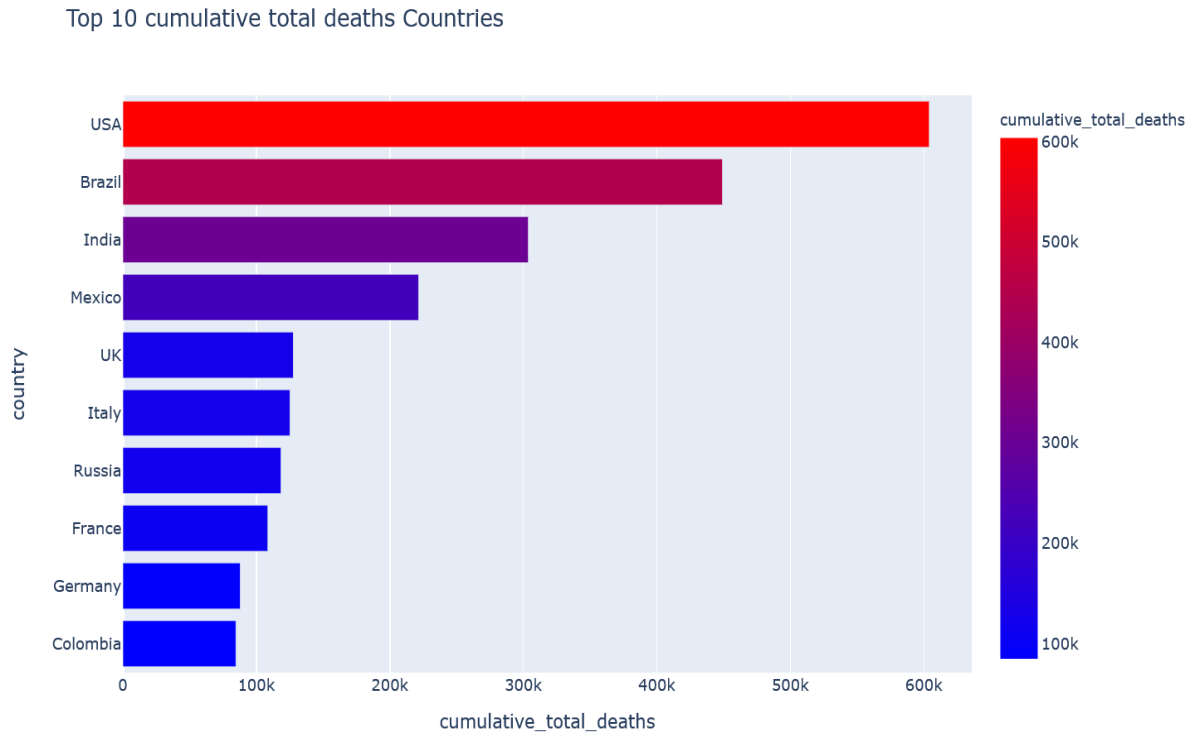
4. Bar Chart

```
def topten( topic : str ):
    top10 = pd.DataFrame(df.groupby('country')[topic].max().nlargest(10).sort_values(ascending = True))
    fig = px.bar(top10, x = topic, y = top10.index, height = 600, color = topic, orientation = 'h',
            color_continuous_scale = px.colors.sequential.Bluered, title = 'Top 10 ' +  " ".join(topic.split('_')) +
    return fig
```

```
topten('cumulative_total_cases')
```



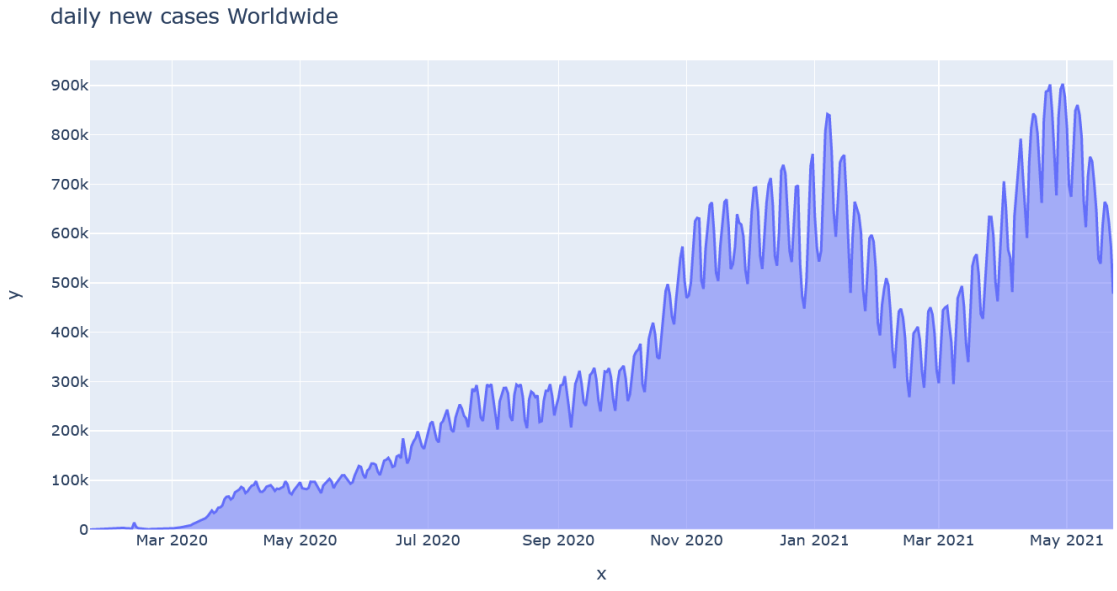

```
topten('cumulative_total_deaths')
```



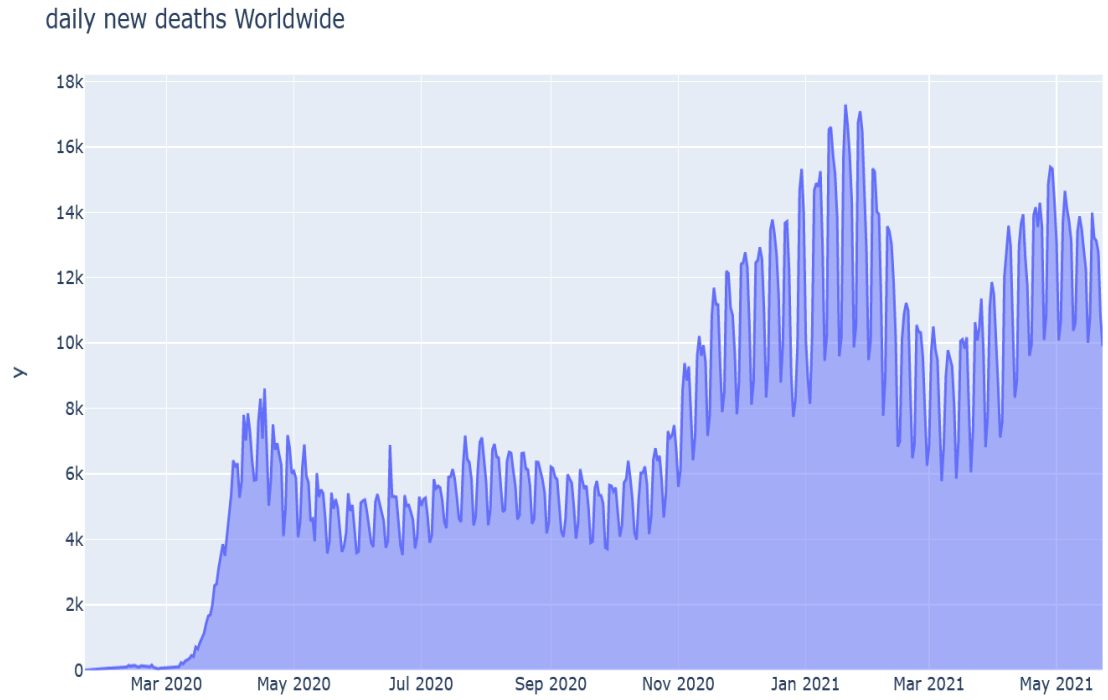
5. Area chart

```
def worldwide(topic :str):
    fig = go.Figure()
    #fig.add_trace(go.Scatter(x = df_dates.index, y = df_dates[topic],line_shape='linear', fill = 'tonexty',line_col
    fig = px.area(df,x = df_dates.index, y = df_dates[topic])
    fig.update_layout(title = " ".join(topic.split('_'))+' Worldwide')
    return fig
```

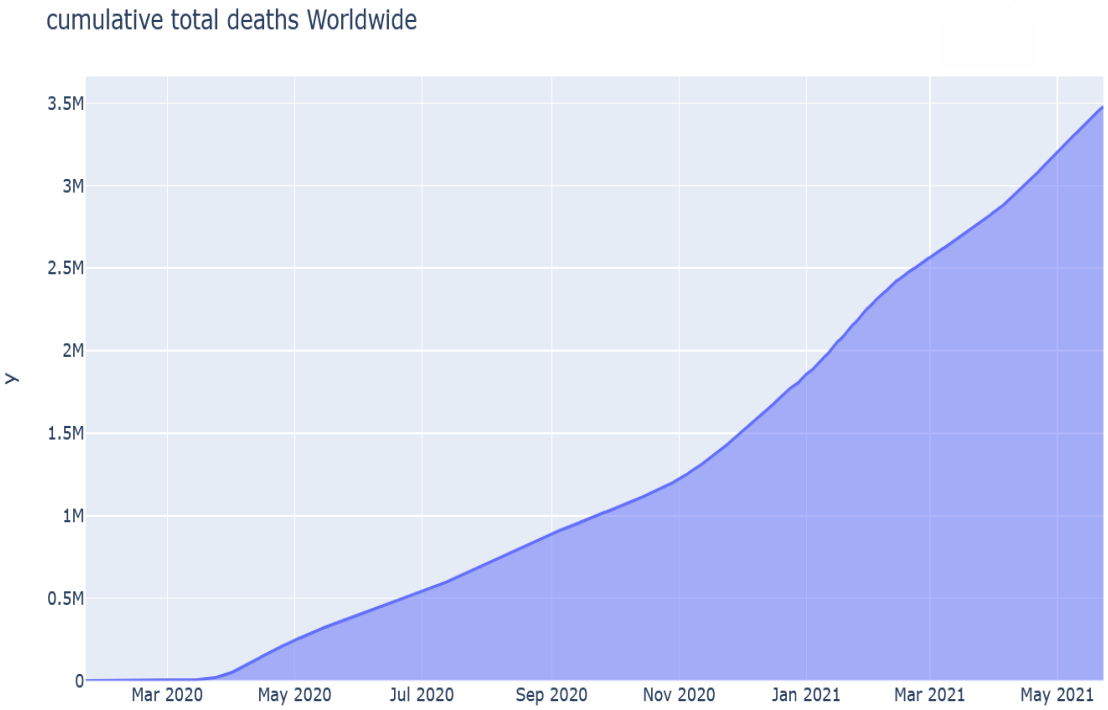
```
worldwide('daily_new_cases')
```



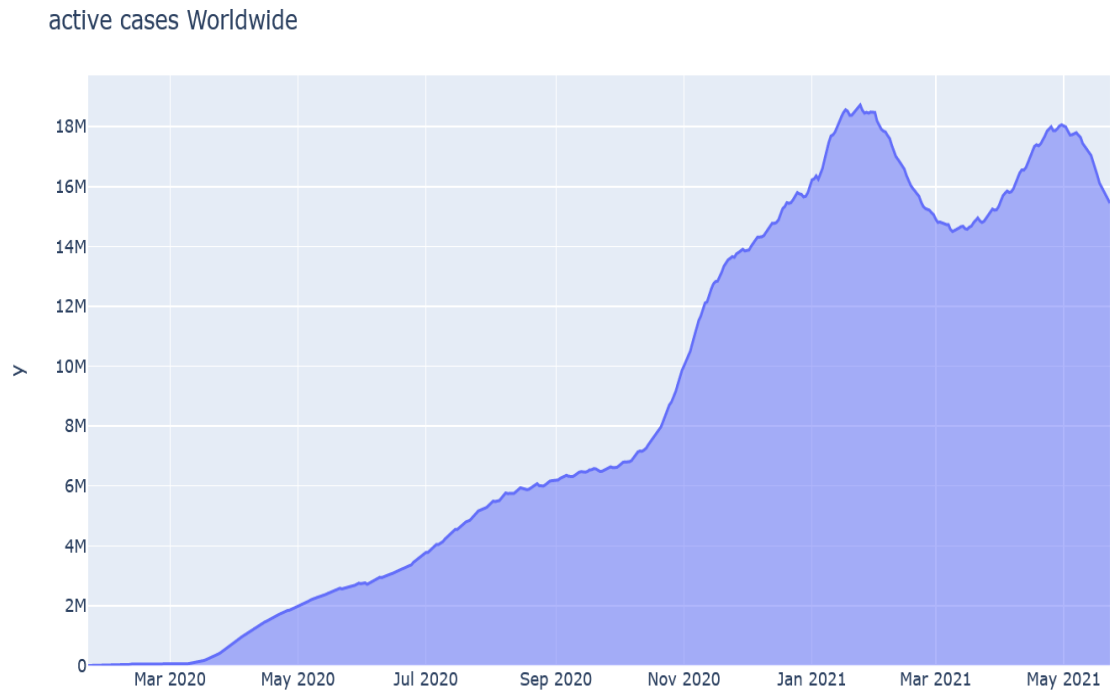
```
worldwide('daily_new_deaths')
```



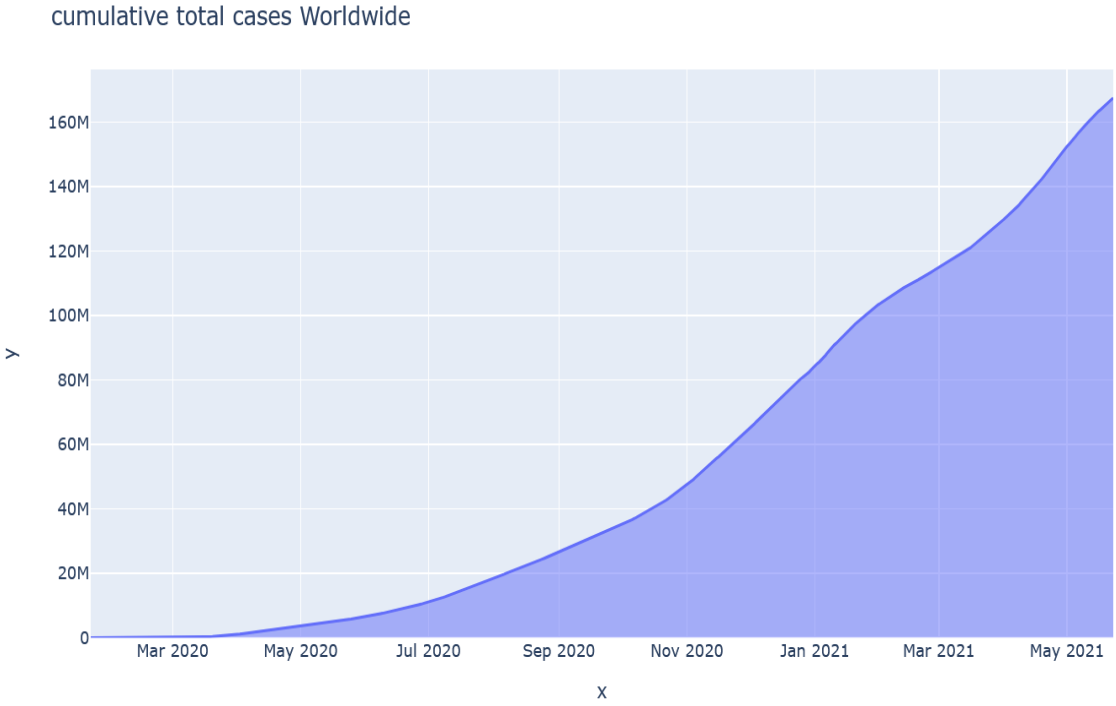

```
worldwide('cumulative_total_deaths')
```



```
worldwide('active_cases')
```



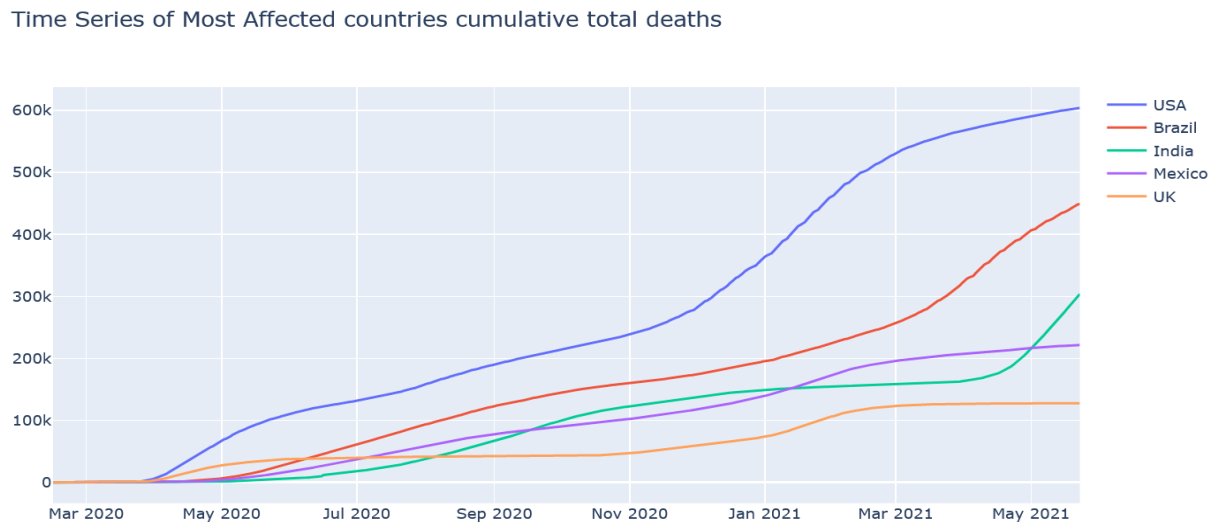

```
worldwide('cumulative_total_cases')
```



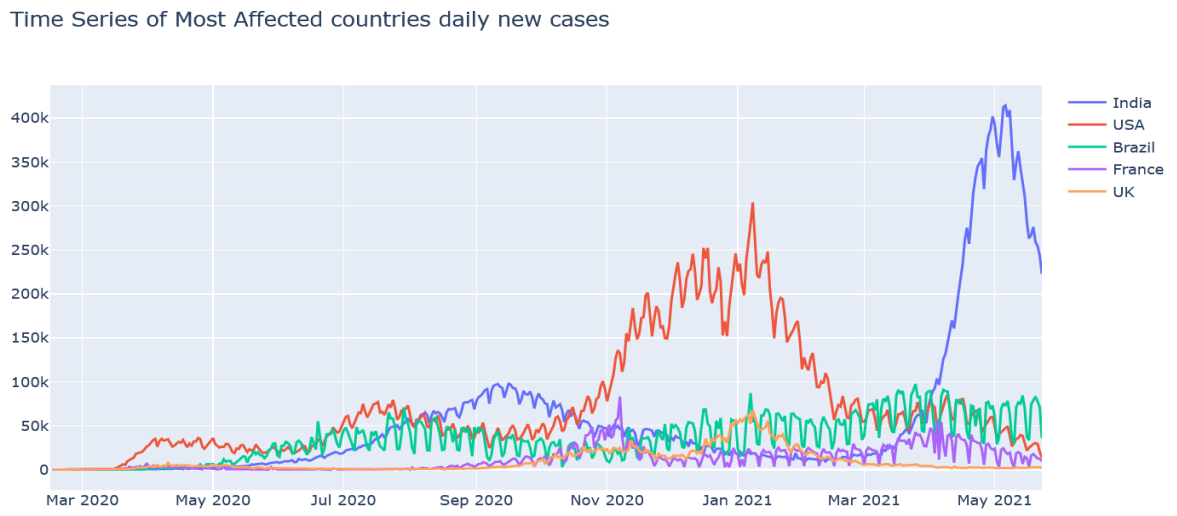
6. Line Graph

```
def mostaffected( topic : str ):
    top5 = pd.DataFrame(df.groupby('country')[topic].max().nlargest(5).sort_values(ascending = False))
    fig = go.Figure()
    for i in range(0,5):
        df_country = df['country'] == top5.index[i]
        df_country = df[df_country]
        fig.add_trace(go.Line(x = df_country['date'], y = df_country[topic], name = top5.index[i]))
        fig.update_layout(title = 'Time Series of Most Affected countries ' + " ".join(topic.split('_')))
    return fig
```

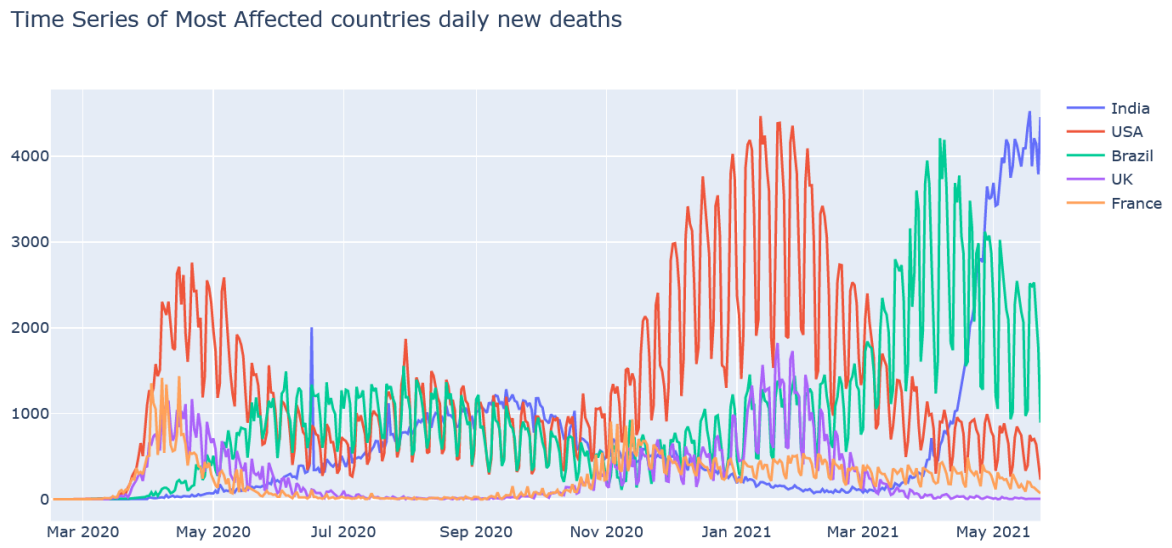
```
mostaffected('cumulative_total_deaths')
```




```
mostaffected('daily_new_cases')
```



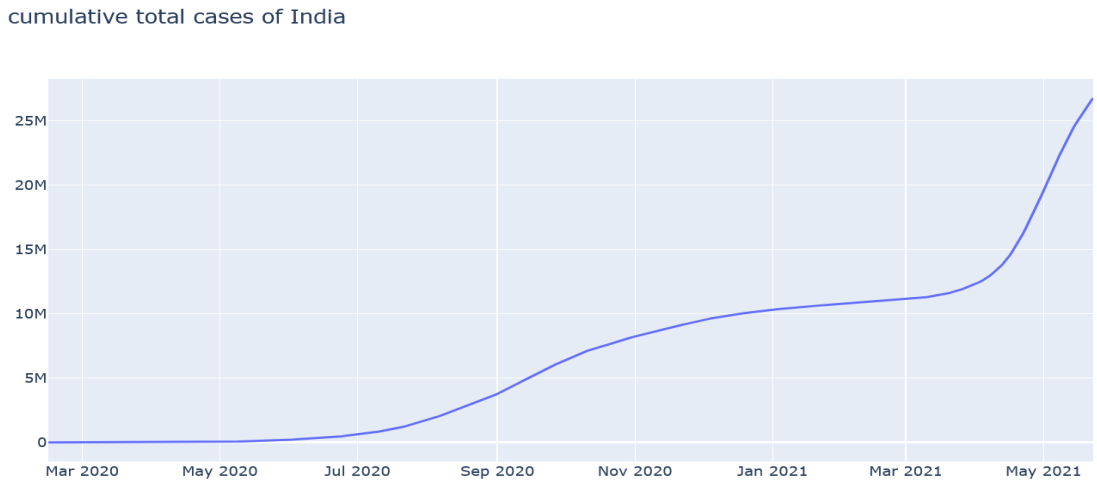
```
mostaffected('daily_new_deaths')
```



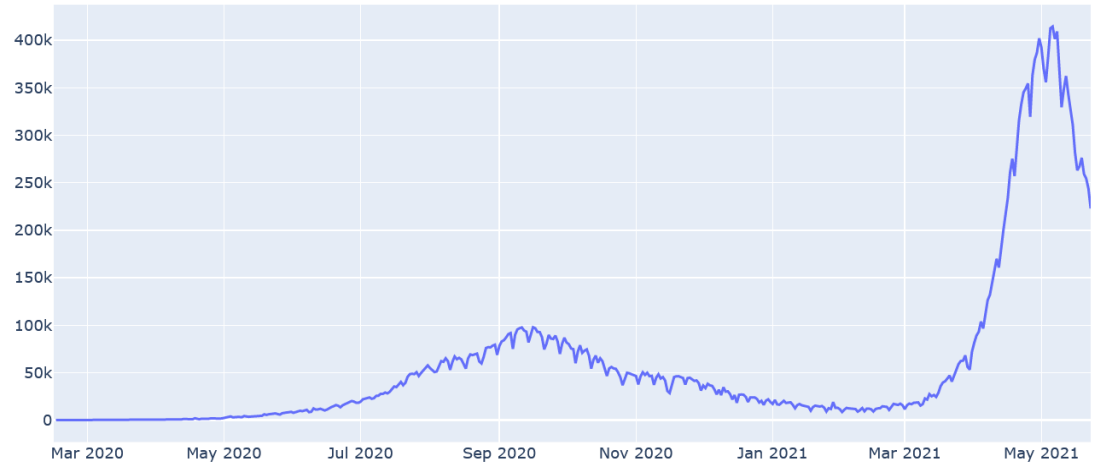

```
def plot(topic1:str,topic:str):
    total = pd.DataFrame(df.groupby('country')[topic1])
    df_country = df['country'] == topic
    df_country = df[df_country]
    fig = go.Figure()
    fig.add_trace(go.Line(x = df_country['date'], y = df_country[topic1], name = topic))
    fig.update_layout(title = " ".join(topic1.split('_'))+ ' of ' +topic)
    fig.show()
```

```
def country(topic : str):
    plot('cumulative_total_cases',topic)
    plot('daily_new_cases',topic)
    plot('active_cases',topic)
    plot('cumulative_total_deaths',topic)
    plot('daily_new_deaths',topic)
```

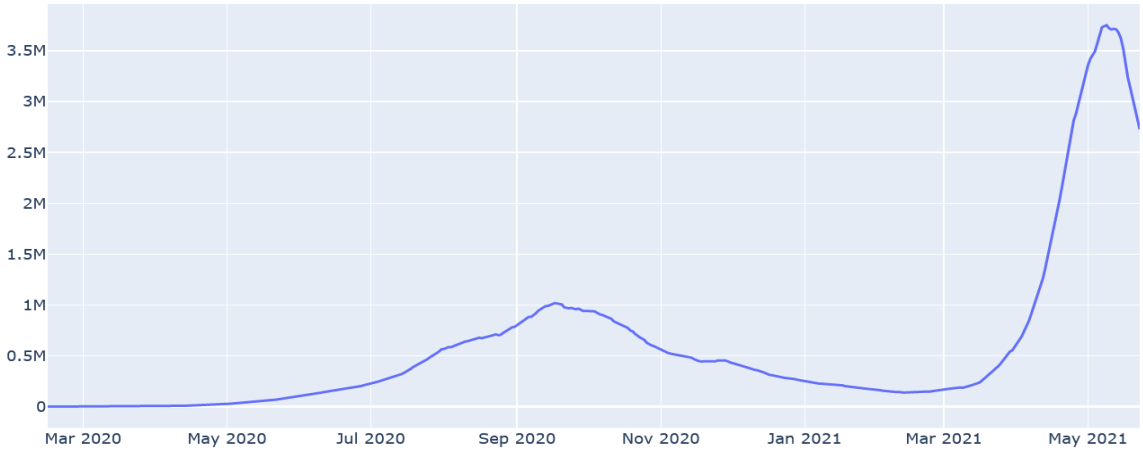
```
country('India')
```



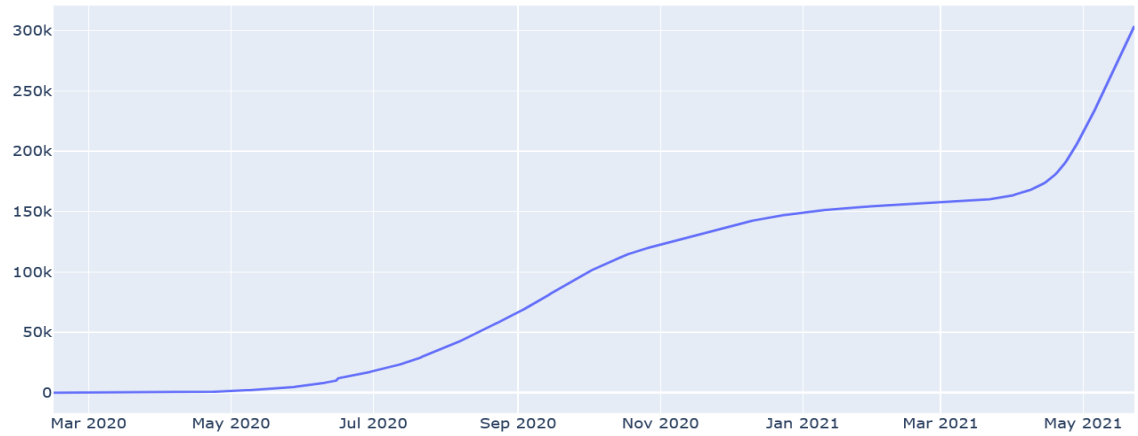
daily new cases of India



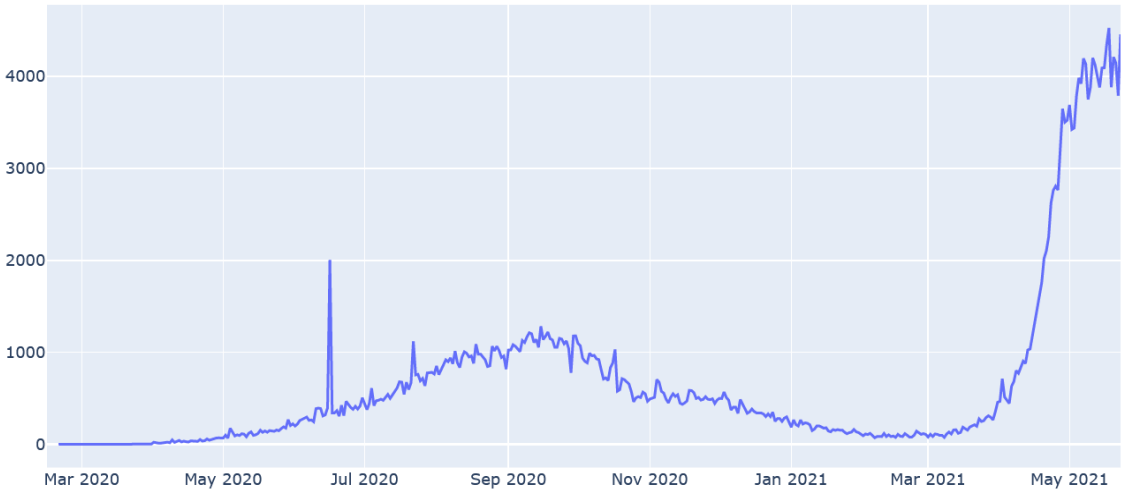
active cases of India



cumulative total deaths of India



daily new deaths of India

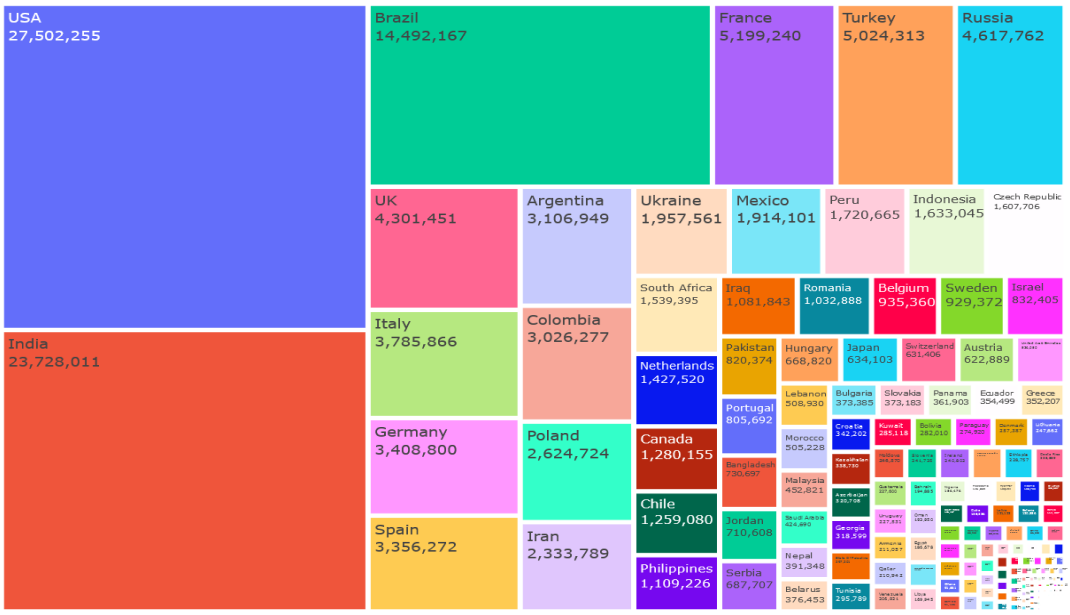


7. Tree Map

```
def tree(topic:str):
    fig = px.treemap(df1, path=["country"], values=topic, height = 750,title="<b>Total Coronavirus "+" ".join(topic.
    fig.update_traces(textinfo = "label+text+value")
    return fig
```

```
tree("total_recovered")
```

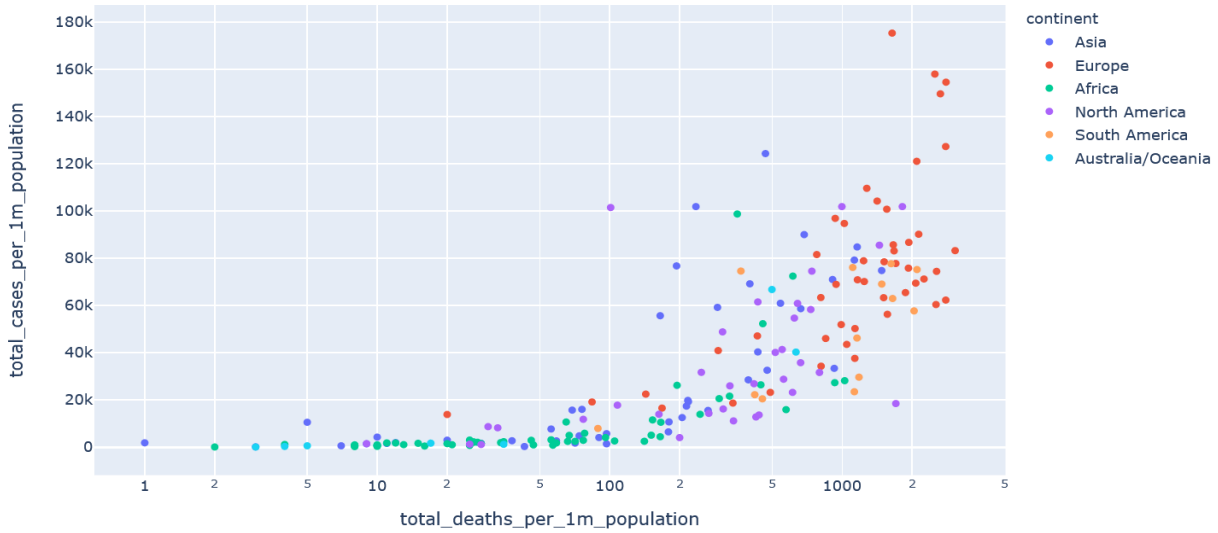
Total Coronavirus total recovered Breakdown by Country



8. Scatter Plot

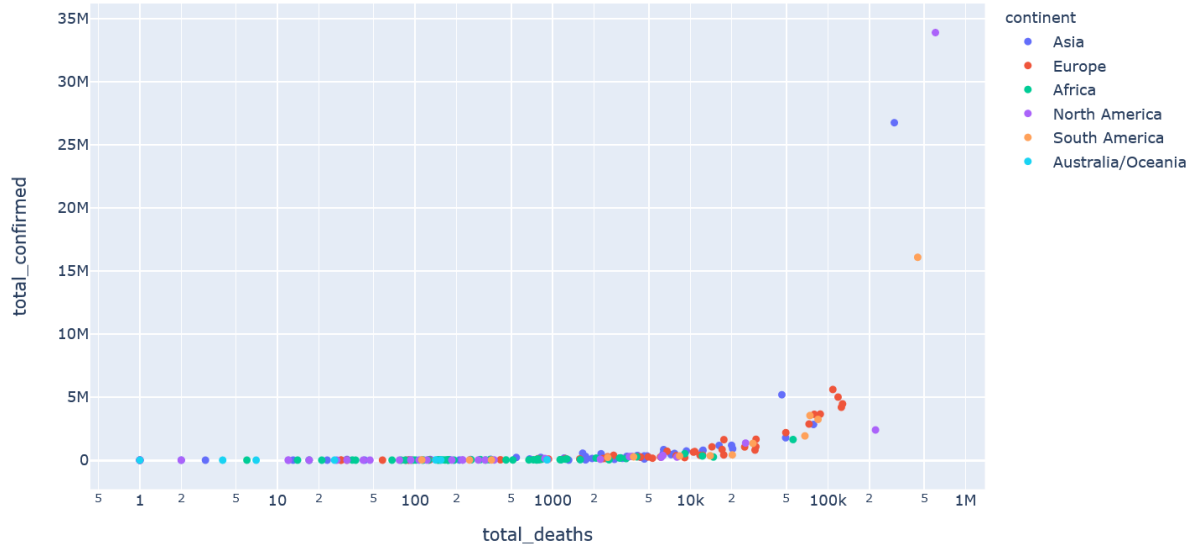
```
fig = px.scatter(df1, y="total_cases_per_1m_population", x="total_deaths_per_1m_population", log_x=True,
                 hover_name="country", hover_data=["continent"],color="continent")

fig.show()
```



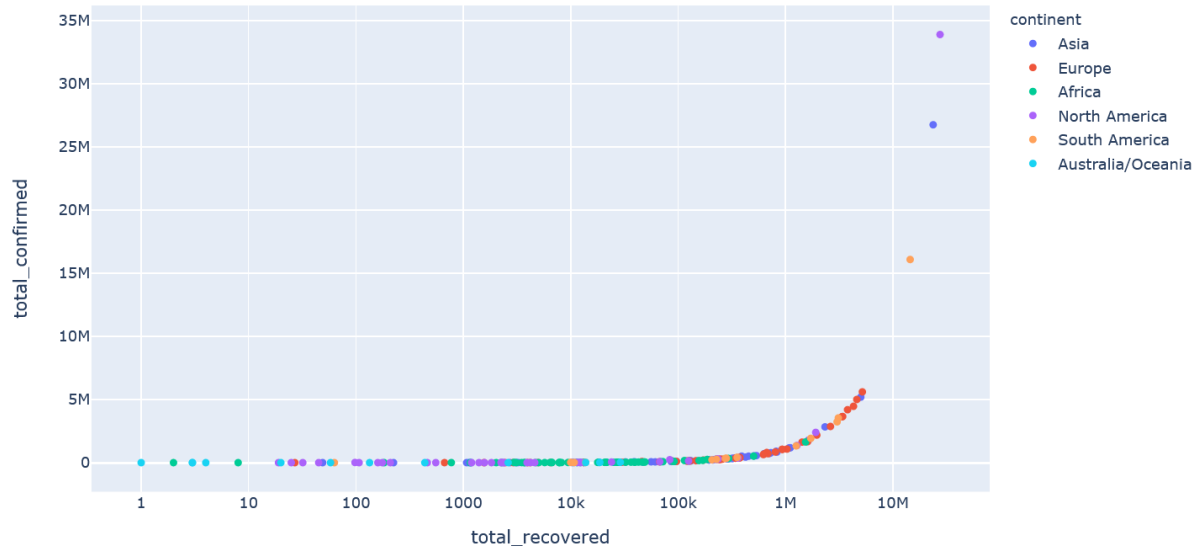

```
fig = px.scatter(df1, y="total_confirmed", x="total_deaths", log_x=True,
                 hover_name="country", hover_data=["continent"],color="continent")

fig.show()
```



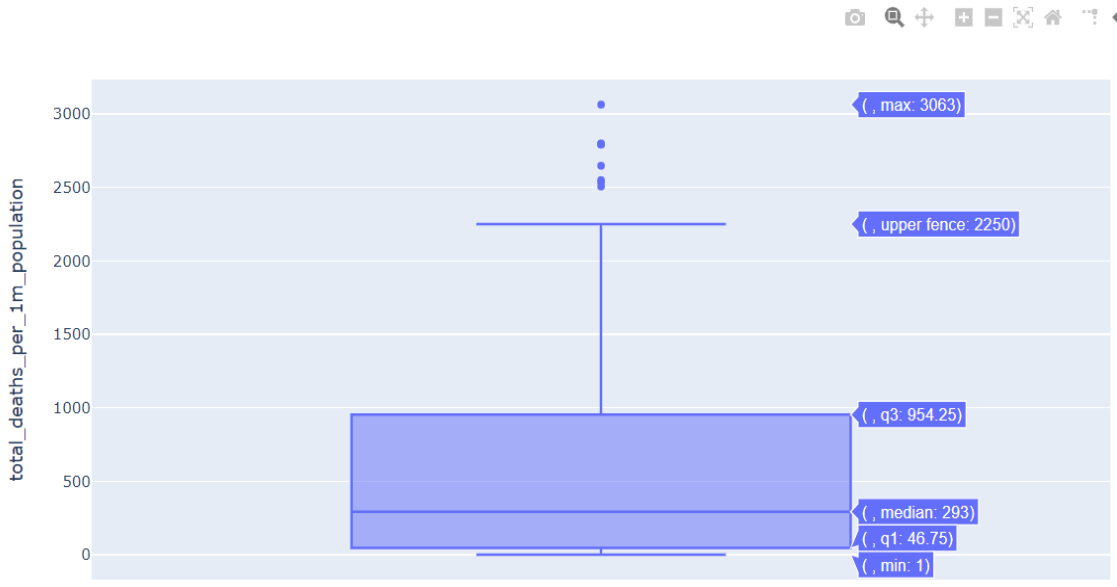
```
fig = px.scatter(df1, y="total_confirmed", x="total_recovered", log_x=True,
                 hover_name="country", hover_data=["continent"],color="continent")

fig.show()
```



9. Box Plot

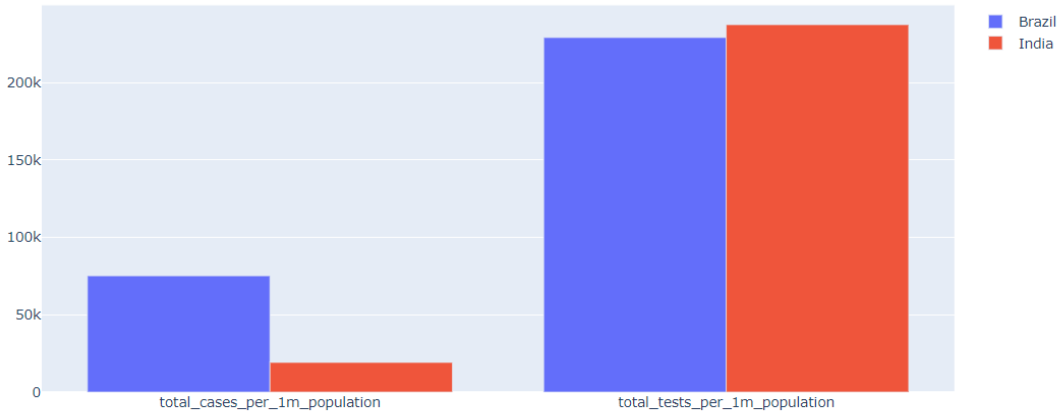
```
fig = px.box(df1, y="total_deaths_per_1m_population")
fig.show()
```



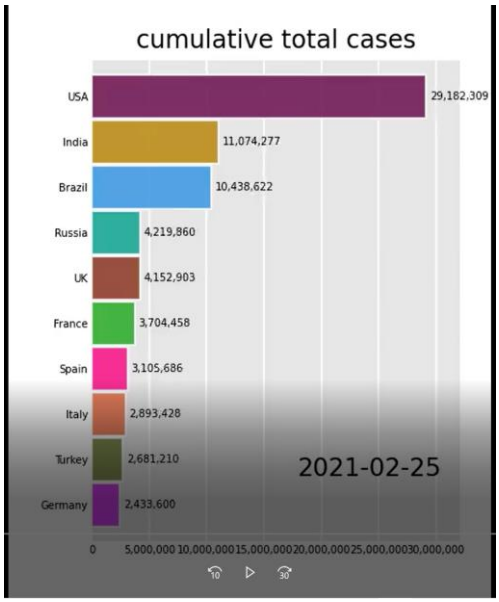
10. Stacked Bar

```
categories=['total_cases_per_1m_population','total_tests_per_1m_population']

fig = go.Figure(data=[
    go.Bar(name='Brazil', x=categories, y=result.iloc[0]),
    go.Bar(name='India', x=categories, y=result.iloc[1])
])
# Change the bar mode
fig.update_layout(barmode='group')
fig.show()
```



11. Bar race chart



12. Word Cloud

```
import seaborn as sns
import numpy as np
```

```
df2 = pd.DataFrame(df1,columns =['country','total_confirmed'])
df2.set_index('country',inplace=True)
```

```
from wordcloud import WordCloud
def wordcloud(topic : str):
    my_dictionary = df2.to_dict()[topic]
    wordcloud = WordCloud()
    wordcloud.generate_from_frequencies(frequencies=my_dictionary)
    plt.figure()
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.savefig('cloud.png')
    plt.show()
```


```
wordcloud('total_recovered')
```

1.5 Conclusion

We have learned how to visualize the data using Python and also learnt about different libraries which are used for data visualization. We also performed various plots to understand the dataset and obtained an overview of dataset and current pandemic situation, then discovering the correlation with a scatter plot Analyzing the categories with bar plots, pie plots and many more. Also plotted the various plots on considering different variables and obtained those plots on worldwide as well on particular countries to understand the covid-19 effect. So the cases were high in countries like USA, India, Brazil and their values on cases got decreased but later we have seen that there was an increase in the cases and deaths in those countries.

List of figures