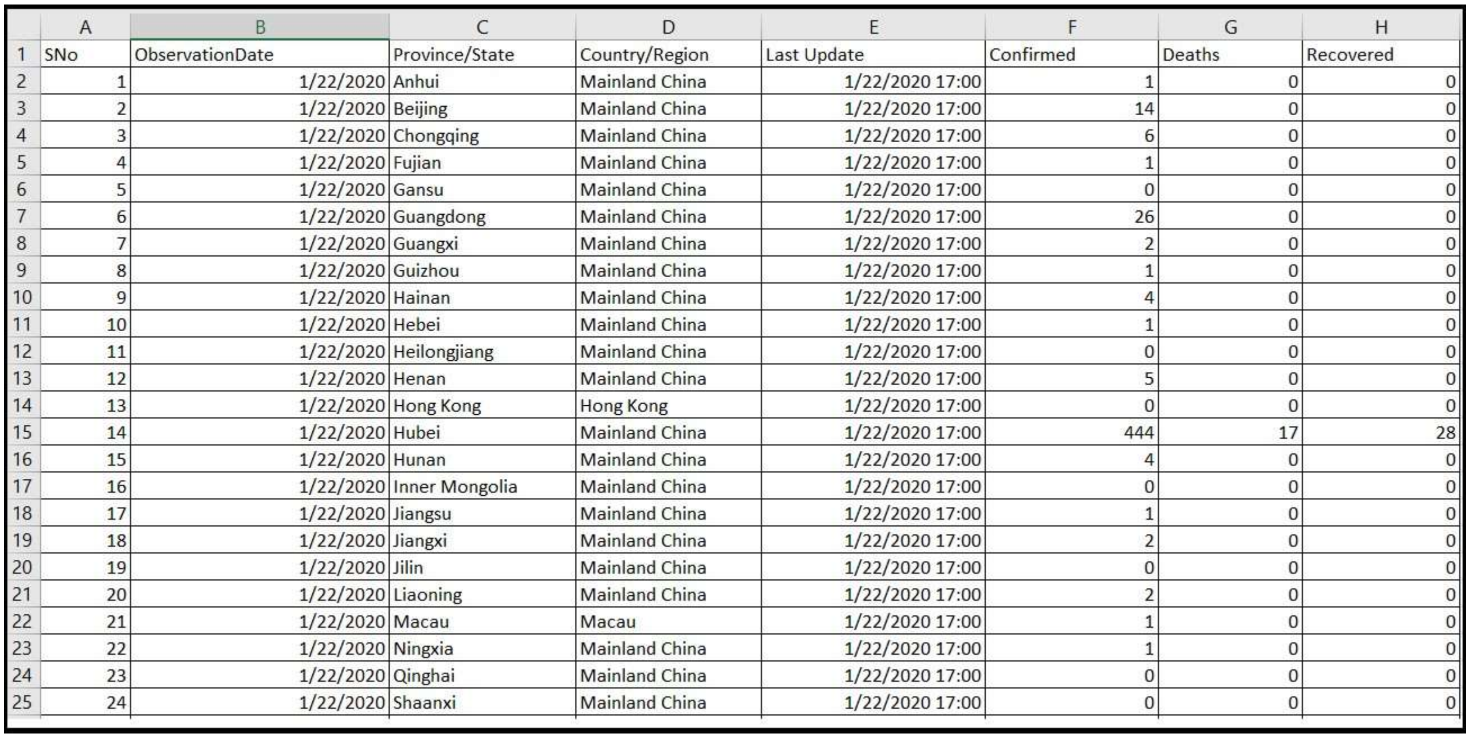
Everywhere on the internet such as blogs, news, social media, and, etc., we are witnessing some common clichés such as "2020 is a worst possible year", " The year which destroyed my career is 2020", "Living at my home for the past 5 months" or an unprofessional comment like "2020 sux", which is also used as a meme by most of the content creator in social media. Furthermore, after hearing any death of a celebrity, a natural disaster, or a deadly explosion, leads again to a global statement that "2020 is the worst year possible". While these things have happened before, so why does everyone feel like 2020 is one of the most detrimental years exist? One word is all it needs to answer this question. COVID!.

COVID-19 is a disease caused by SARS-CoV-2 that can trigger what doctors call a respiratory tract infection. COVID is spreading across the globe for the past six months, which gave rise to the pandemic situation which we are facing right now. So this is a blog where we analyze how COVID-19 has impacted across the globe by analyzing specific details such as the number of countries spread, active cases to months, closed cases, Mortality Rate vs Recovery Rate, Growth Factor, top 15 affected countries, and many other.

Before going into details regarding the analysis, we shall look closely at the dataset we have:

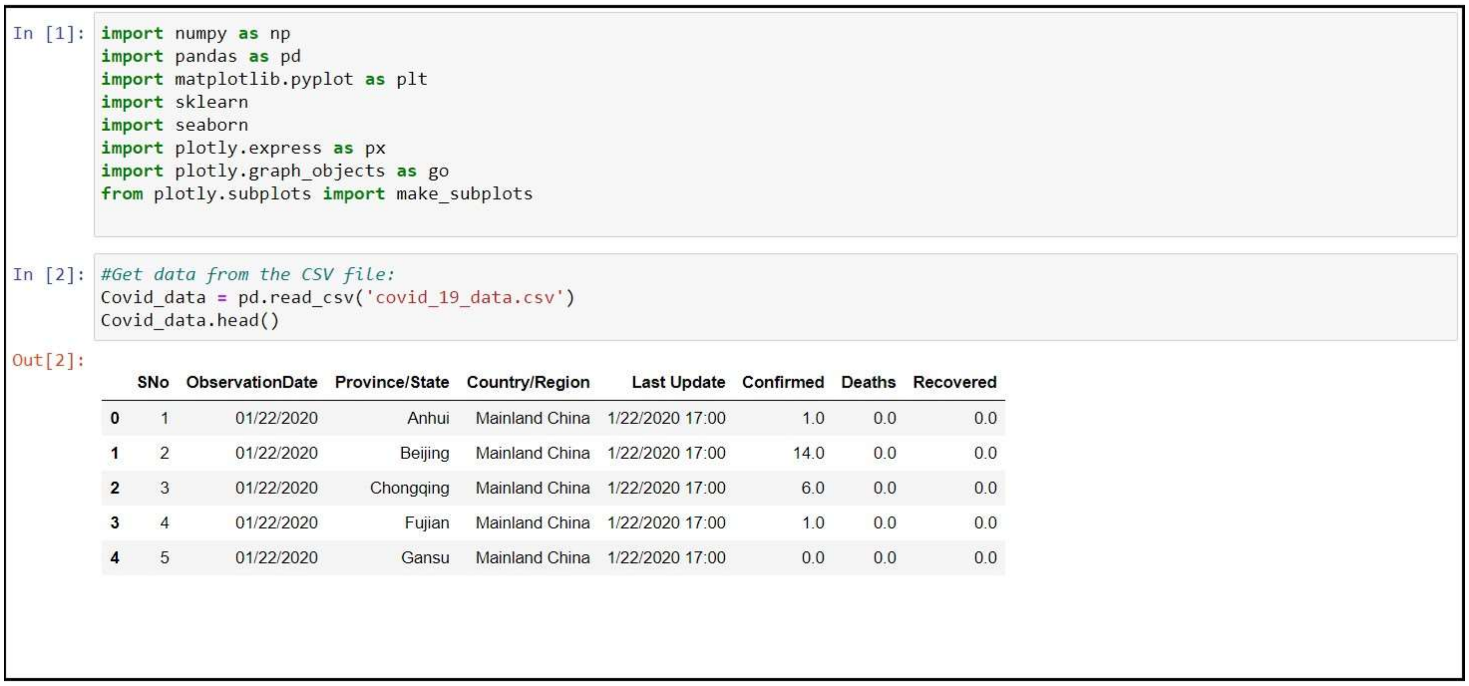


	A	B	C	D	E	F	G	H
1	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
2	1	1/22/2020	Anhui	Mainland China	1/22/2020 17:00	1	0	0
3	2	1/22/2020	Beijing	Mainland China	1/22/2020 17:00	14	0	0
4	3	1/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6	0	0
5	4	1/22/2020	Fujian	Mainland China	1/22/2020 17:00	1	0	0
6	5	1/22/2020	Gansu	Mainland China	1/22/2020 17:00	0	0	0
7	6	1/22/2020	Guangdong	Mainland China	1/22/2020 17:00	26	0	0
8	7	1/22/2020	Guangxi	Mainland China	1/22/2020 17:00	2	0	0
9	8	1/22/2020	Guizhou	Mainland China	1/22/2020 17:00	1	0	0
10	9	1/22/2020	Hainan	Mainland China	1/22/2020 17:00	4	0	0
11	10	1/22/2020	Hebei	Mainland China	1/22/2020 17:00	1	0	0
12	11	1/22/2020	Heilongjiang	Mainland China	1/22/2020 17:00	0	0	0
13	12	1/22/2020	Henan	Mainland China	1/22/2020 17:00	5	0	0
14	13	1/22/2020	Hong Kong	Hong Kong	1/22/2020 17:00	0	0	0
15	14	1/22/2020	Hubei	Mainland China	1/22/2020 17:00	444	17	28
16	15	1/22/2020	Hunan	Mainland China	1/22/2020 17:00	4	0	0
17	16	1/22/2020	Inner Mongolia	Mainland China	1/22/2020 17:00	0	0	0
18	17	1/22/2020	Jiangsu	Mainland China	1/22/2020 17:00	1	0	0
19	18	1/22/2020	Jiangxi	Mainland China	1/22/2020 17:00	2	0	0
20	19	1/22/2020	Jilin	Mainland China	1/22/2020 17:00	0	0	0
21	20	1/22/2020	Liaoning	Mainland China	1/22/2020 17:00	2	0	0
22	21	1/22/2020	Macau	Macau	1/22/2020 17:00	1	0	0
23	22	1/22/2020	Ningxia	Mainland China	1/22/2020 17:00	1	0	0
24	23	1/22/2020	Qinghai	Mainland China	1/22/2020 17:00	0	0	0
25	24	1/22/2020	Shaanxi	Mainland China	1/22/2020 17:00	0	0	0

As you can see, we have ObservationDate, Province/State, Country/Region, Last Update, Confirmed, Deaths & Recovered are the details which we will be utilizing for our analysis. The observation Date exists until July 8th, 2020.

So now, let's start our analysis using Jupyter Notebook- Python version 3.7.4.

We start by including the necessary libraries and importing the CSV data.



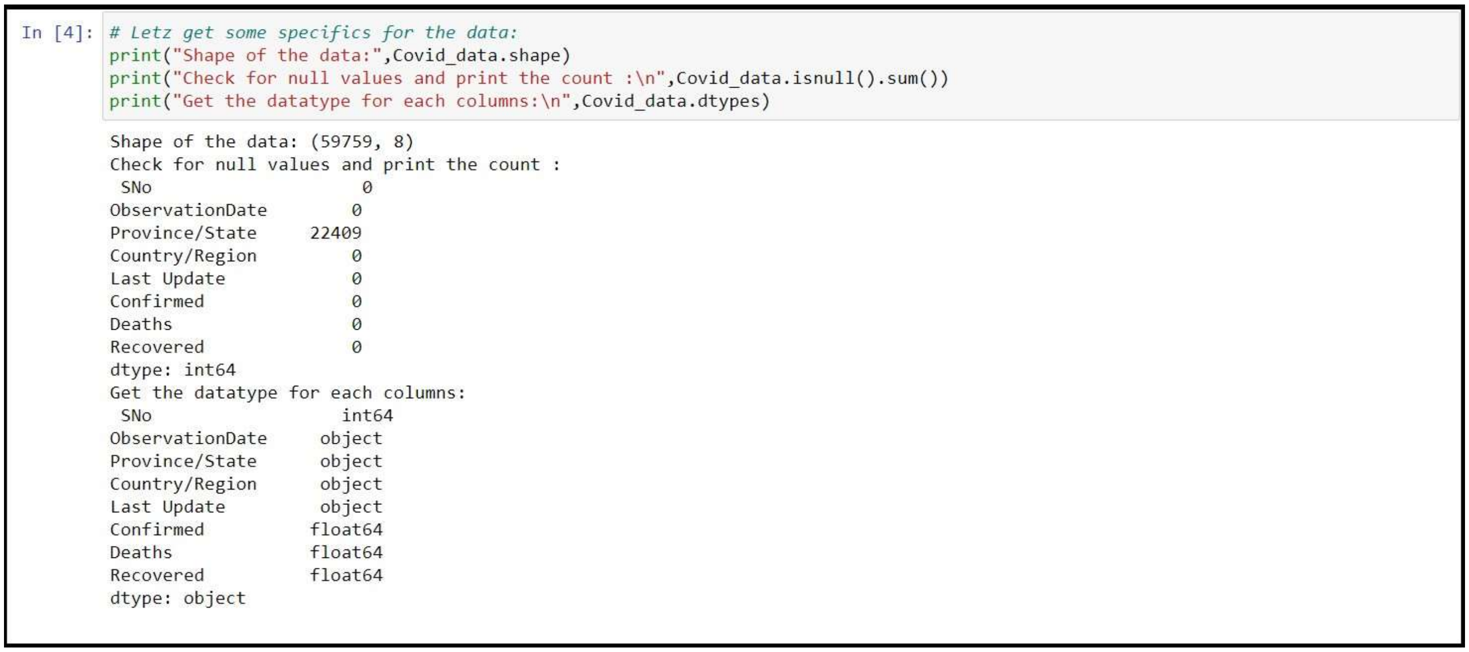
```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import sklearn
        import seaborn
        import plotly.express as px
        import plotly.graph_objects as go
        from plotly.subplots import make_subplots
```

```
In [2]: #Get data from the CSV file:
        Covid_data = pd.read_csv('covid_19_data.csv')
        Covid_data.head()
```

Out[2]:

	SNo	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	1	01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2	01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	3	01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	4	01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	5	01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0

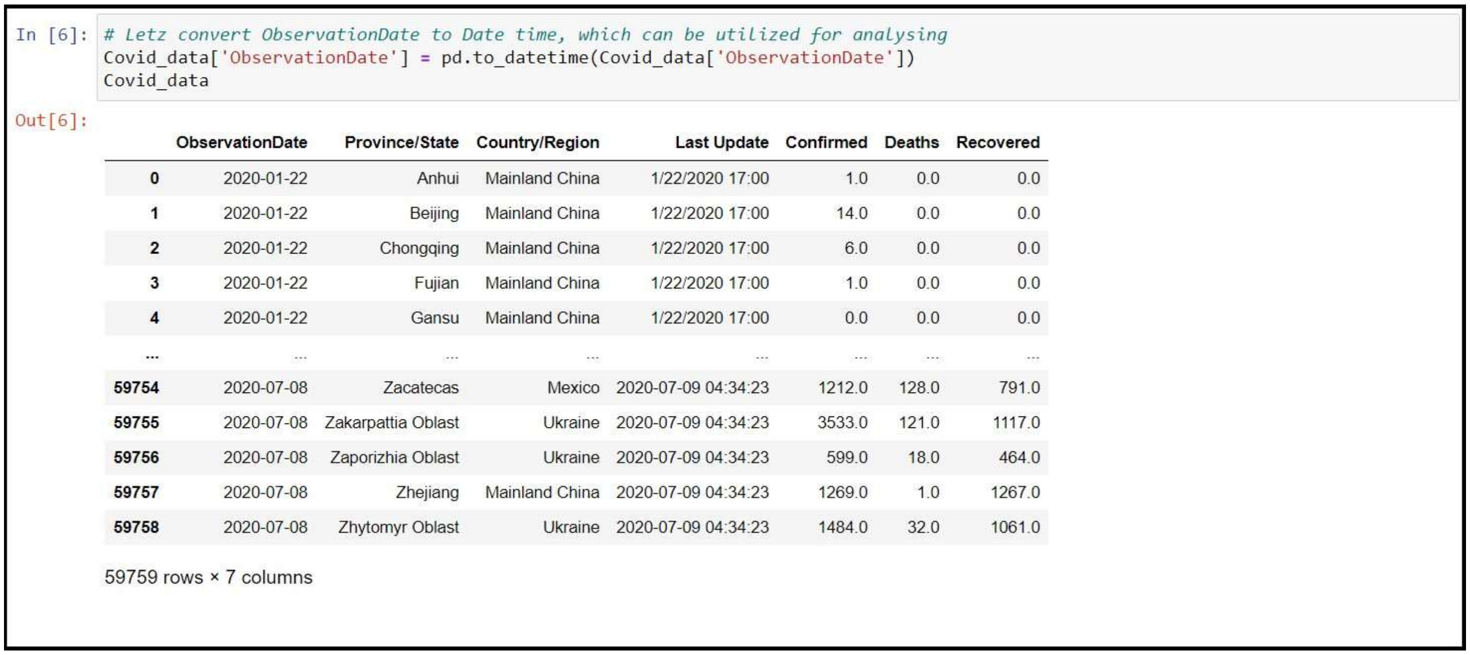
More information regarding the dataset:



```
In [4]: # Letz get some specifics for the data:
        print("Shape of the data:",Covid_data.shape)
        print("Check for null values and print the count :\n",Covid_data.isnull().sum())
        print("Get the datatype for each columns:\n",Covid_data.dtypes)
```

```
Shape of the data: (59759, 8)
Check for null values and print the count :
 SNo                   0
ObservationDate       0
Province/State    22409
Country/Region        0
Last Update           0
Confirmed             0
Deaths                0
Recovered             0
dtype: int64
Get the datatype for each columns:
 SNo                int64
ObservationDate   object
Province/State    object
Country/Region    object
Last Update       object
Confirmed        float64
Deaths           float64
Recovered        float64
dtype: object
```

We now convert the ObservationDate to Date time and group the data to Country & ObservationDate, which will help us to analyze concerning datewise.

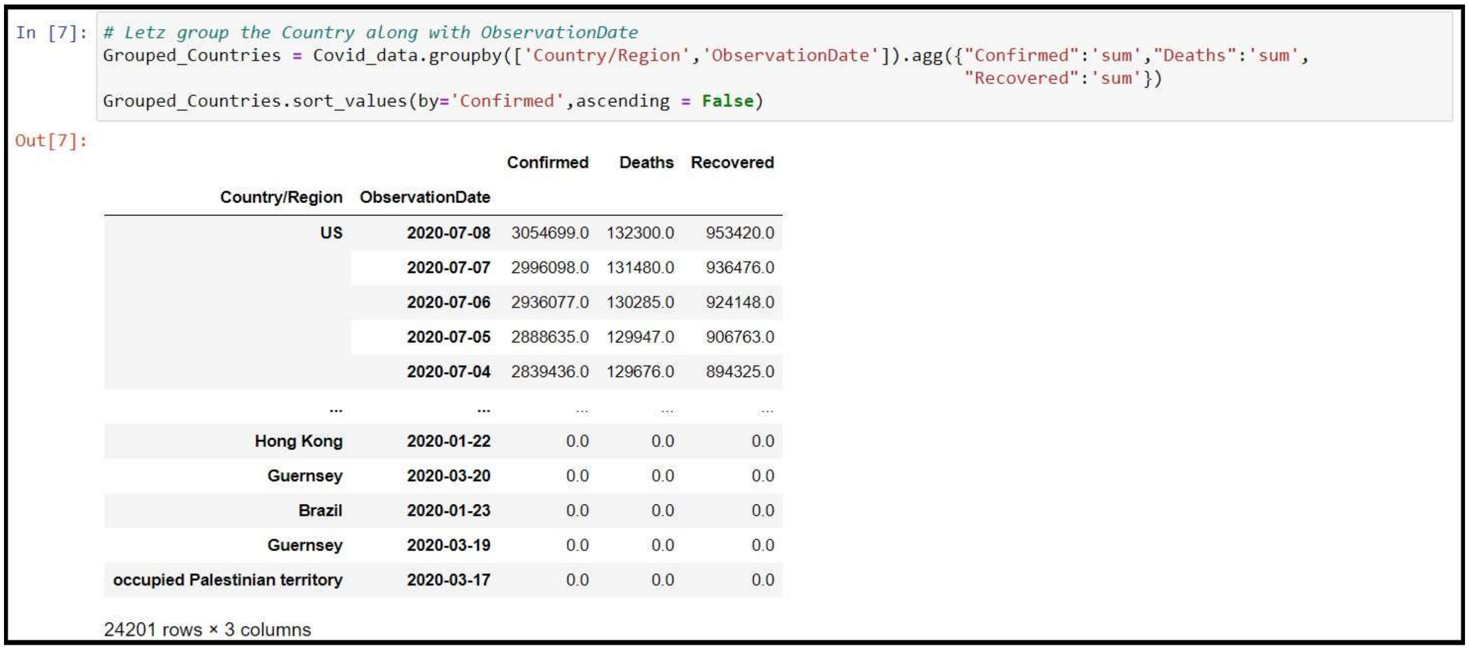


In [6]:
```python
# Letz convert ObservationDate to Date time, which can be utilized for analysing
Covid_data['ObservationDate'] = pd.to_datetime(Covid_data['ObservationDate'])
Covid_data
```

Out[6]:

	ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
0	2020-01-22	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
1	2020-01-22	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
2	2020-01-22	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
3	2020-01-22	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
4	2020-01-22	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
...	...	...	...	...	...	...	...
59754	2020-07-08	Zacatecas	Mexico	2020-07-09 04:34:23	1212.0	128.0	791.0
59755	2020-07-08	Zakarpattia Oblast	Ukraine	2020-07-09 04:34:23	3533.0	121.0	1117.0
59756	2020-07-08	Zaporizhia Oblast	Ukraine	2020-07-09 04:34:23	599.0	18.0	464.0
59757	2020-07-08	Zhejiang	Mainland China	2020-07-09 04:34:23	1269.0	1.0	1267.0
59758	2020-07-08	Zhytomyr Oblast	Ukraine	2020-07-09 04:34:23	1484.0	32.0	1061.0

59759 rows × 7 columns



In [7]:
```python
# Letz group the Country along with ObservationDate
Grouped_Countries = Covid_data.groupby(['Country/Region','ObservationDate']).agg({"Confirmed":'sum',"Deaths":'sum',
                                                                                 "Recovered":'sum'})
Grouped_Countries.sort_values(by='Confirmed',ascending = False)
```

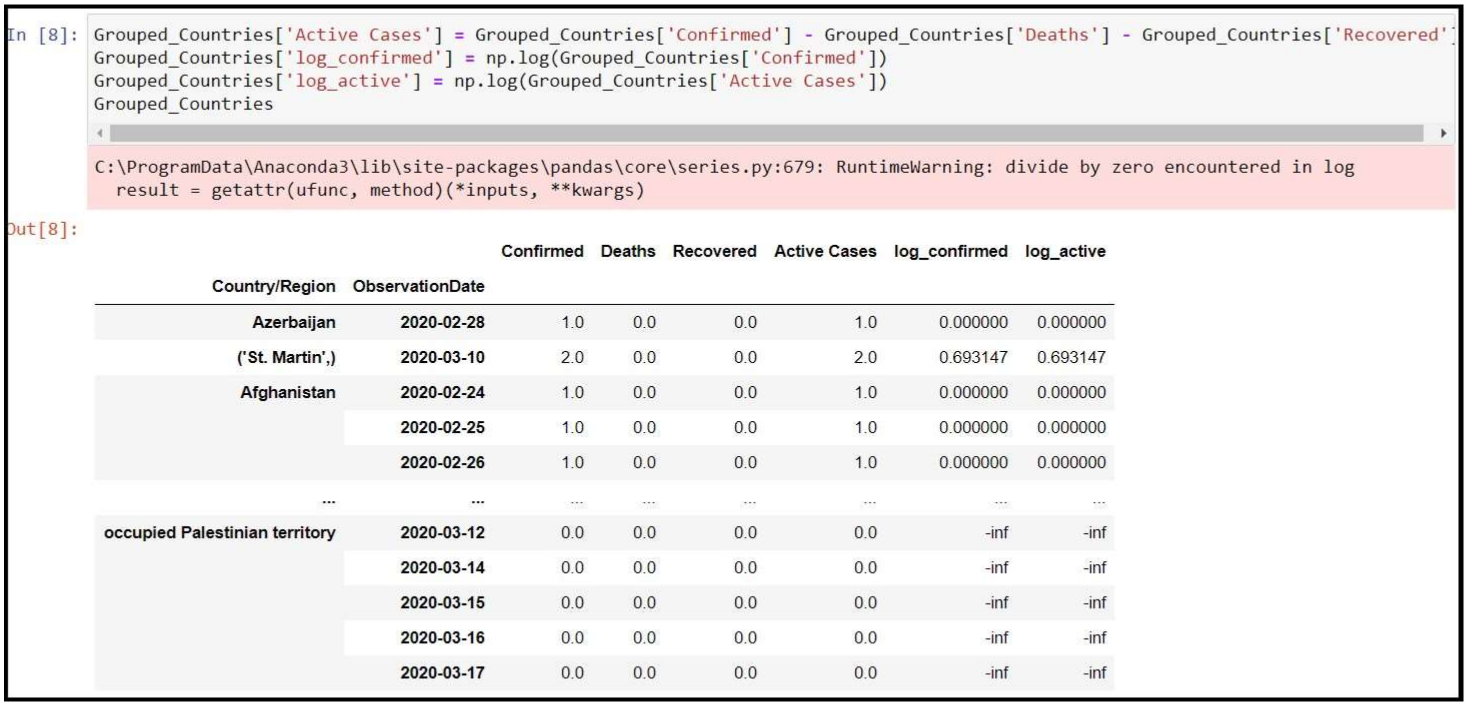
Out[7]:

		Confirmed	Deaths	Recovered
Country/Region	**ObservationDate**			
US	**2020-07-08**	3054699.0	132300.0	953420.0
	2020-07-07	2996098.0	131480.0	936476.0
	2020-07-06	2936077.0	130285.0	924148.0
	2020-07-05	2888635.0	129947.0	906763.0
	2020-07-04	2839436.0	129676.0	894325.0
...	**...**	...	...	...
Hong Kong	**2020-01-22**	0.0	0.0	0.0
Guernsey	**2020-03-20**	0.0	0.0	0.0
Brazil	**2020-01-23**	0.0	0.0	0.0
Guernsey	**2020-03-19**	0.0	0.0	0.0
occupied Palestinian territory	**2020-03-17**	0.0	0.0	0.0

24201 rows × 3 columns

We now calculate the active cases by subtracting the number of deaths & Recovered cases from Confirmed Cases.

I.e. Active Cases = Confirmed - [Deaths + Recovered]



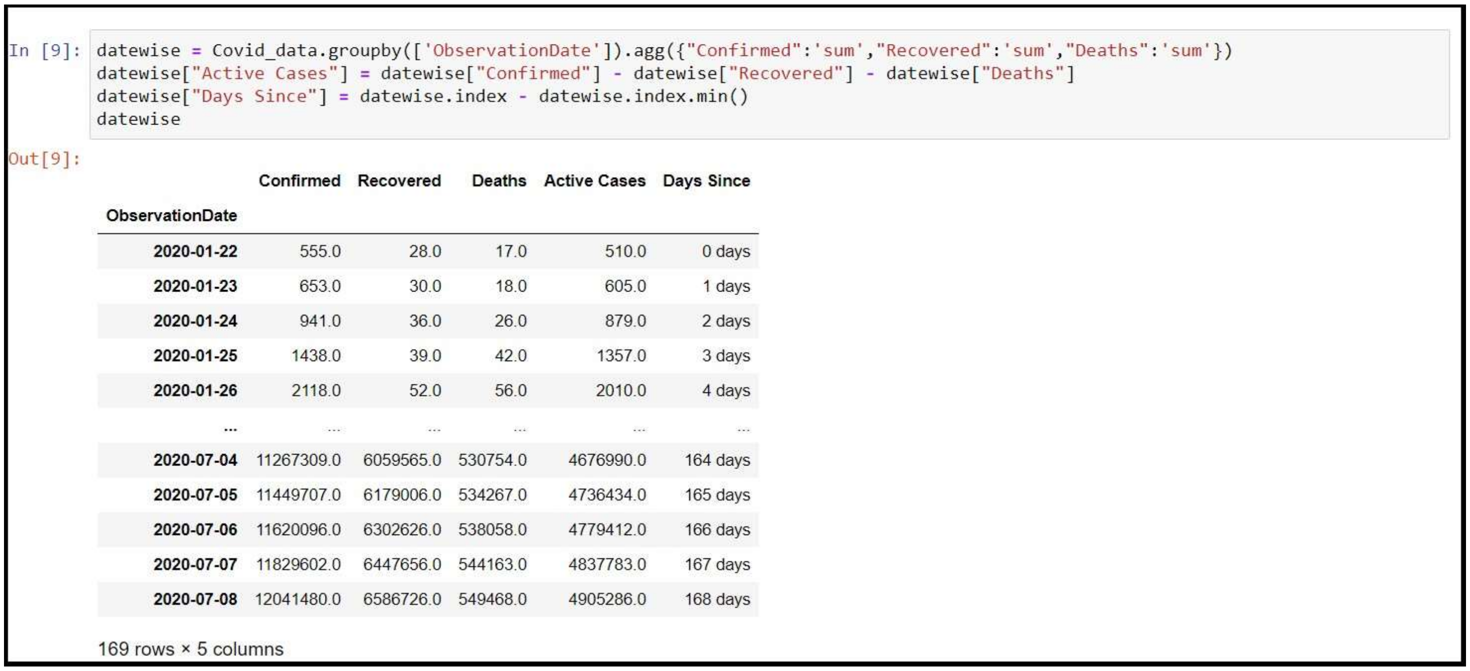
```
In [8]: Grouped_Countries['Active Cases'] = Grouped_Countries['Confirmed'] - Grouped_Countries['Deaths'] - Grouped_Countries['Recovered'
        Grouped_Countries['log_confirmed'] = np.log(Grouped_Countries['Confirmed'])
        Grouped_Countries['log_active'] = np.log(Grouped_Countries['Active Cases'])
        Grouped_Countries
```

```
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\series.py:679: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[8]:

		Confirmed	Deaths	Recovered	Active Cases	log_confirmed	log_active
Country/Region	**ObservationDate**						
Azerbaijan	**2020-02-28**	1.0	0.0	0.0	1.0	0.000000	0.000000
('St. Martin',)	**2020-03-10**	2.0	0.0	0.0	2.0	0.693147	0.693147
Afghanistan	**2020-02-24**	1.0	0.0	0.0	1.0	0.000000	0.000000
	2020-02-25	1.0	0.0	0.0	1.0	0.000000	0.000000
	2020-02-26	1.0	0.0	0.0	1.0	0.000000	0.000000
...	**...**	...	...	...	...	...	...
occupied Palestinian territory	**2020-03-12**	0.0	0.0	0.0	0.0	-inf	-inf
	2020-03-14	0.0	0.0	0.0	0.0	-inf	-inf
	2020-03-15	0.0	0.0	0.0	0.0	-inf	-inf
	2020-03-16	0.0	0.0	0.0	0.0	-inf	-inf
	2020-03-17	0.0	0.0	0.0	0.0	-inf	-inf

To analyze some basic pieces of information such as the Total number of affected Countries, Confirmed/ Recovered/ Death/ Active cases across the globe, an approx number of confirmed/recovered/death/active cases per day around, and many more. We begin by creating a new data frame as below:



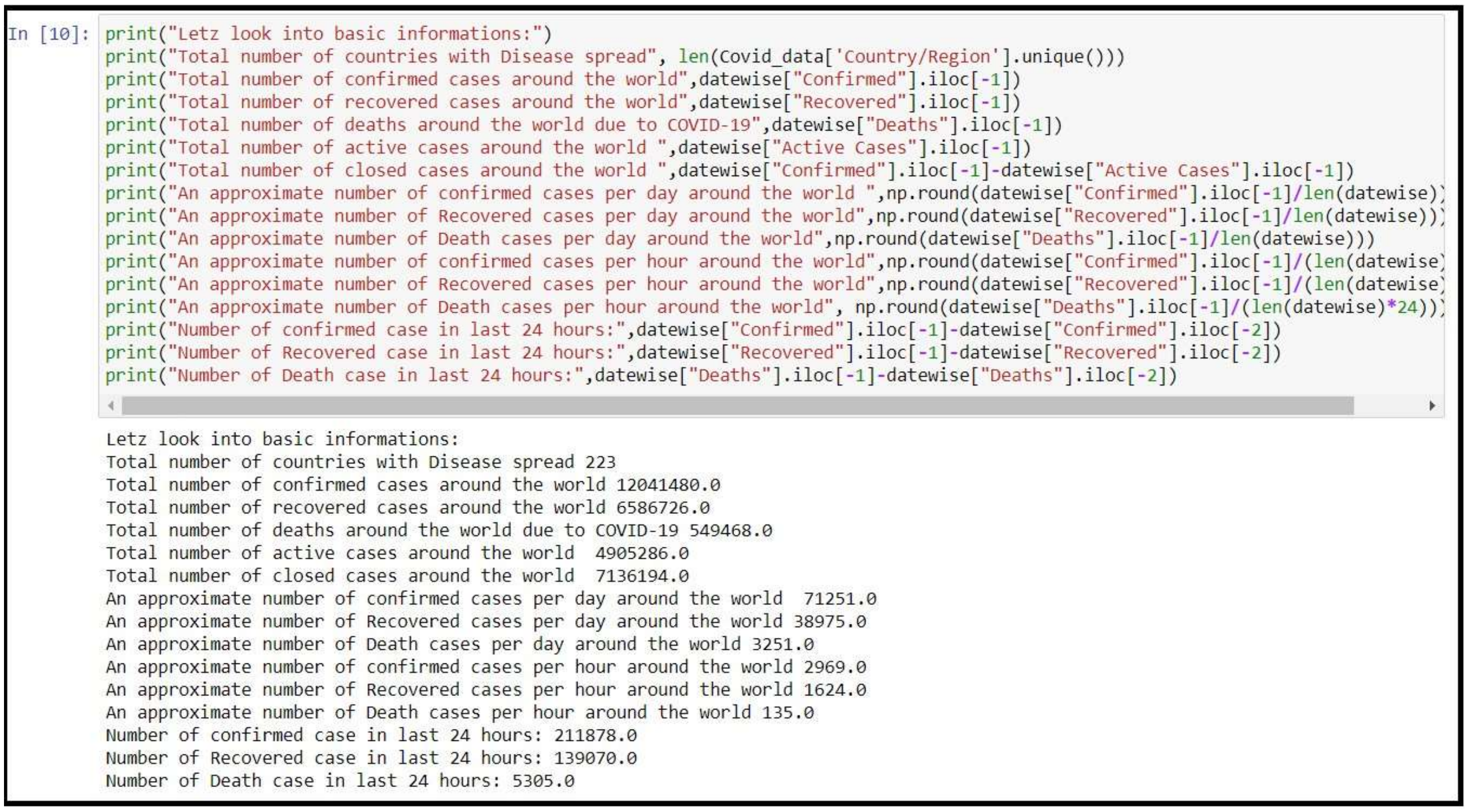
```
In [9]: datewise = Covid_data.groupby(['ObservationDate']).agg({"Confirmed":'sum',"Recovered":'sum',"Deaths":'sum'})
        datewise["Active Cases"] = datewise["Confirmed"] - datewise["Recovered"] - datewise["Deaths"]
        datewise["Days Since"] = datewise.index - datewise.index.min()
        datewise
```

Out[9]:

	Confirmed	Recovered	Deaths	Active Cases	Days Since
ObservationDate					
2020-01-22	555.0	28.0	17.0	510.0	0 days
2020-01-23	653.0	30.0	18.0	605.0	1 days
2020-01-24	941.0	36.0	26.0	879.0	2 days
2020-01-25	1438.0	39.0	42.0	1357.0	3 days
2020-01-26	2118.0	52.0	56.0	2010.0	4 days
...	...	...	...	...	...
2020-07-04	11267309.0	6059565.0	530754.0	4676990.0	164 days
2020-07-05	11449707.0	6179006.0	534267.0	4736434.0	165 days
2020-07-06	11620096.0	6302626.0	538058.0	4779412.0	166 days
2020-07-07	11829602.0	6447656.0	544163.0	4837783.0	167 days
2020-07-08	12041480.0	6586726.0	549468.0	4905286.0	168 days

169 rows × 5 columns

By utilizing the created Dataframe, we fetch the required analysis as below:

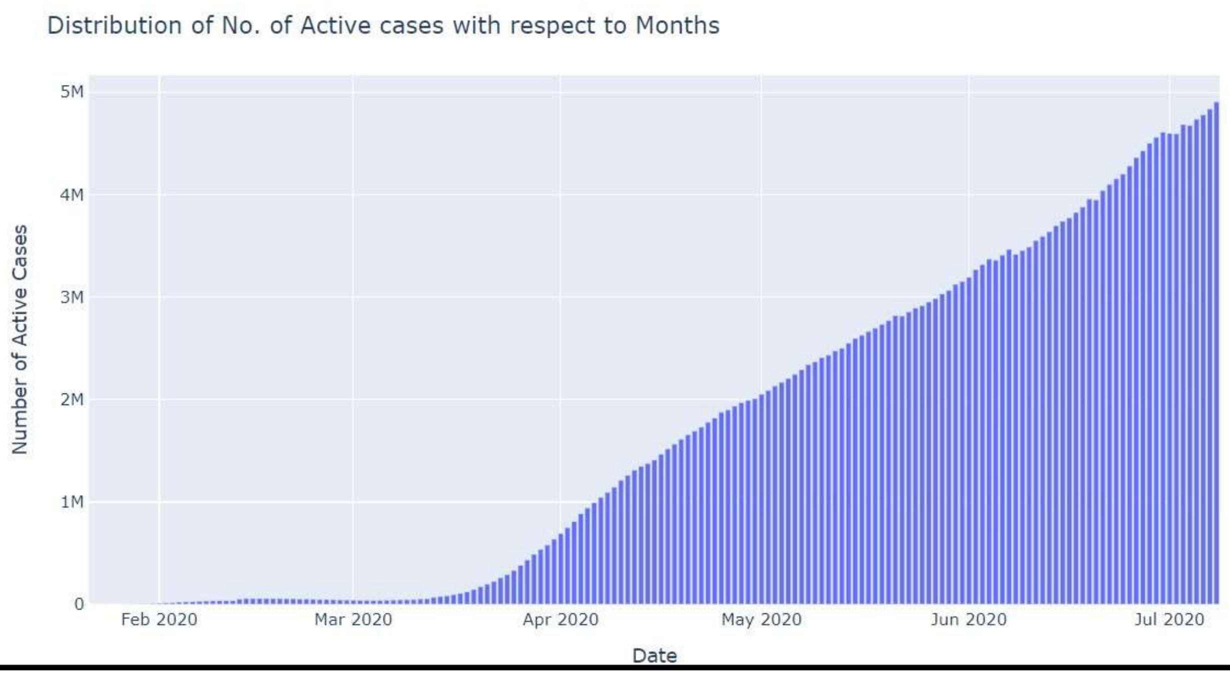


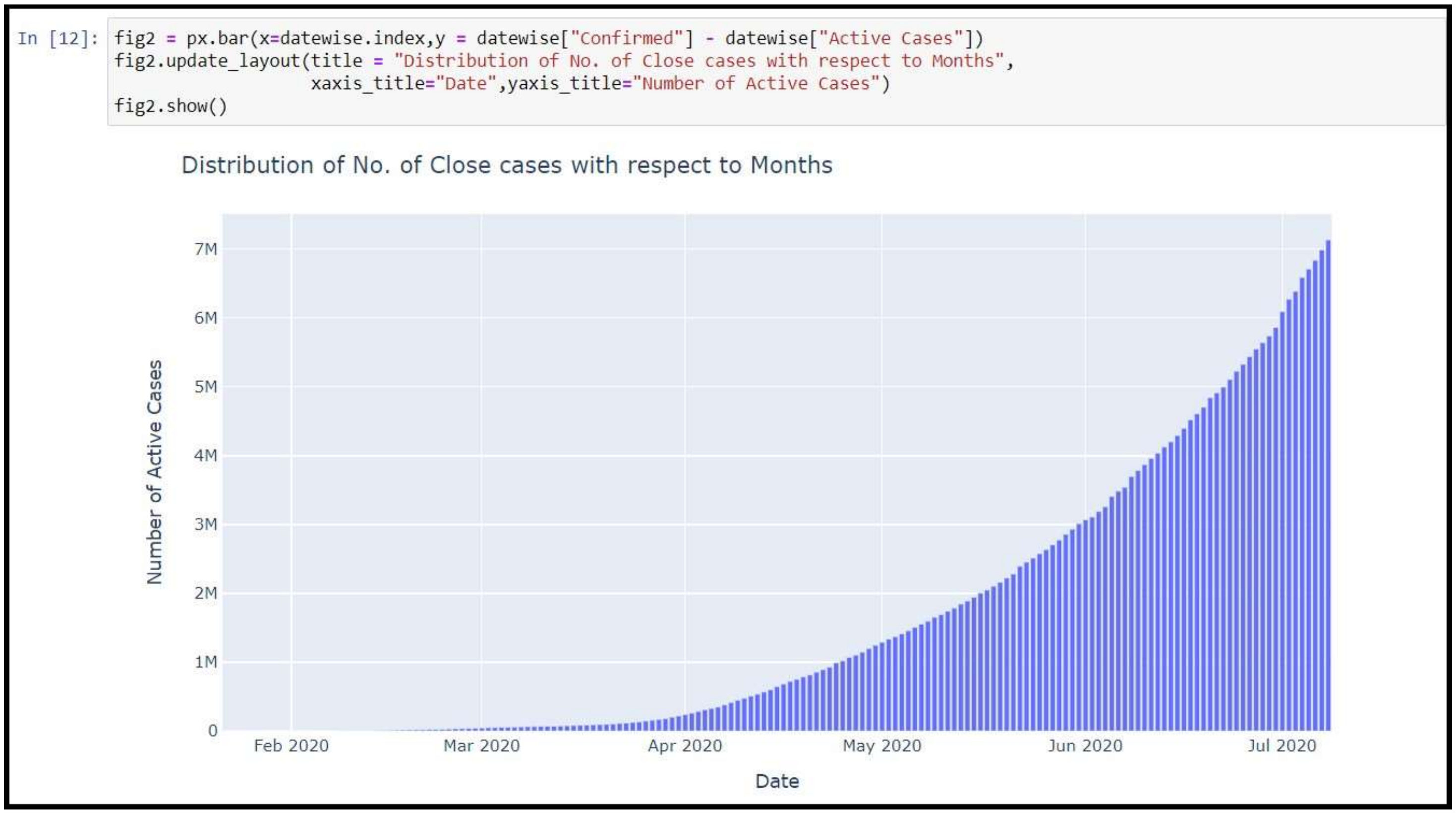
```
In [10]: print("Letz look into basic informations:")
         print("Total number of countries with Disease spread", len(Covid_data['Country/Region'].unique()))
         print("Total number of confirmed cases around the world",datewise["Confirmed"].iloc[-1])
         print("Total number of recovered cases around the world",datewise["Recovered"].iloc[-1])
         print("Total number of deaths around the world due to COVID-19",datewise["Deaths"].iloc[-1])
         print("Total number of active cases around the world ",datewise["Active Cases"].iloc[-1])
         print("Total number of closed cases around the world ",datewise["Confirmed"].iloc[-1]-datewise["Active Cases"].iloc[-1])
         print("An approximate number of confirmed cases per day around the world ",np.round(datewise["Confirmed"].iloc[-1]/len(datewise)
         print("An approximate number of Recovered cases per day around the world",np.round(datewise["Recovered"].iloc[-1]/len(datewise))
         print("An approximate number of Death cases per day around the world",np.round(datewise["Deaths"].iloc[-1]/len(datewise)))
         print("An approximate number of confirmed cases per hour around the world",np.round(datewise["Confirmed"].iloc[-1]/(len(datewise)
         print("An approximate number of Recovered cases per hour around the world",np.round(datewise["Recovered"].iloc[-1]/(len(datewise)
         print("An approximate number of Death cases per hour around the world", np.round(datewise["Deaths"].iloc[-1]/(len(datewise)*24))
         print("Number of confirmed case in last 24 hours:",datewise["Confirmed"].iloc[-1]-datewise["Confirmed"].iloc[-2])
         print("Number of Recovered case in last 24 hours:",datewise["Recovered"].iloc[-1]-datewise["Recovered"].iloc[-2])
         print("Number of Death case in last 24 hours:",datewise["Deaths"].iloc[-1]-datewise["Deaths"].iloc[-2])
```

```
Letz look into basic informations:
Total number of countries with Disease spread 223
Total number of confirmed cases around the world 12041480.0
Total number of recovered cases around the world 6586726.0
Total number of deaths around the world due to COVID-19 549468.0
Total number of active cases around the world  4905286.0
Total number of closed cases around the world  7136194.0
An approximate number of confirmed cases per day around the world  71251.0
An approximate number of Recovered cases per day around the world 38975.0
An approximate number of Death cases per day around the world 3251.0
An approximate number of confirmed cases per hour around the world 2969.0
An approximate number of Recovered cases per hour around the world 1624.0
An approximate number of Death cases per hour around the world 135.0
Number of confirmed case in last 24 hours: 211878.0
Number of Recovered case in last 24 hours: 139070.0
Number of Death case in last 24 hours: 5305.0
```

Now we visualize the Active Cases & Close cases for different months by plotting a bar graph for the number of respective cases and months which we grouped previously.

```
In [11]: fig = px.bar(x=datewise.index,y = datewise["Active Cases"])
         fig.update_layout(title = "Distribution of No. of Active cases with respect to Months",xaxis_title="Date",
                           yaxis_title="Number of Active Cases")
         fig.show()
```





```
In [12]: fig2 = px.bar(x=datewise.index,y = datewise["Confirmed"] - datewise["Active Cases"])
         fig2.update_layout(title = "Distribution of No. of Close cases with respect to Months",
                            xaxis_title="Date",yaxis_title="Number of Active Cases")
         fig2.show()
```

We follow a similar step to analyze only for the United States.

We start by creating a Dataframe named 'US_Data' which only has US data segregated from the original dataset.

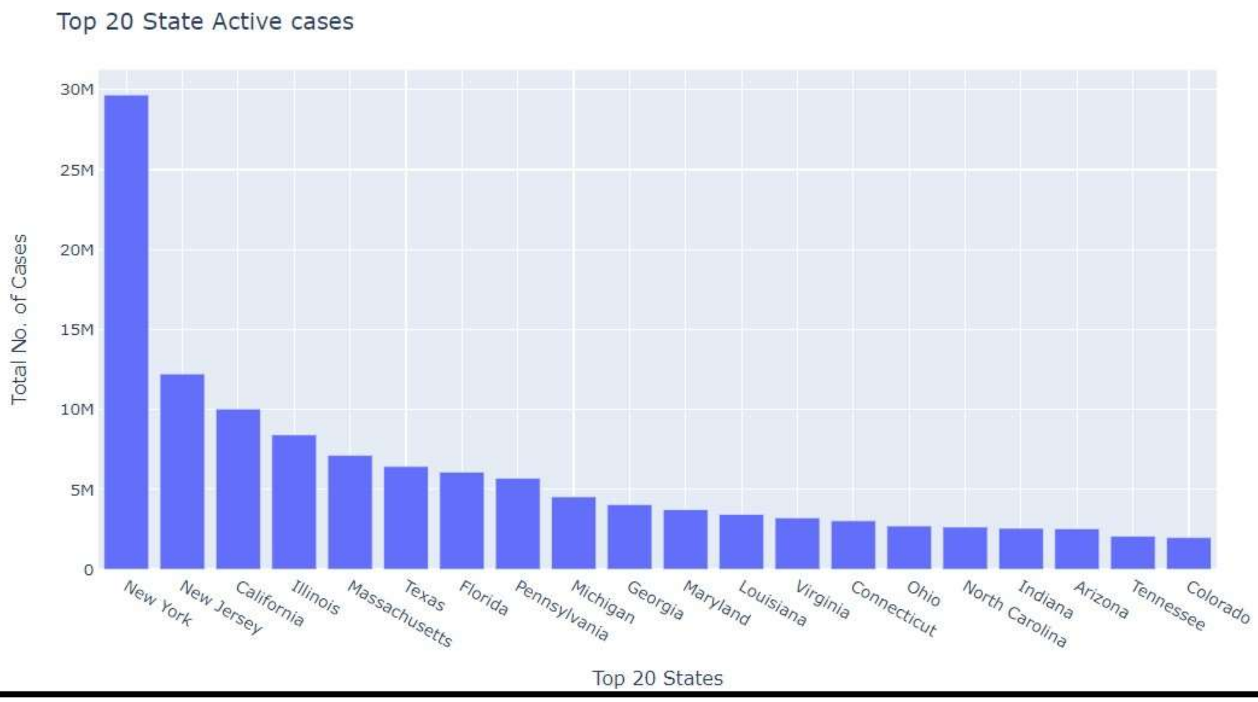
Much as before, we add a new column for Active Cases for United Stated by subtracting the Number of Deaths & Recovered from Confirmed Cases.

i.e. US_Data["Active Cases"] = US_Data["Confirmed"] -(US_Data["Deaths"] + US_Data["Recovered"])

Then we aggregate concerning Confirmed/Recovered/Deaths & Active Cases, also group by State, and create a new DataFrame named 'State_US_data'

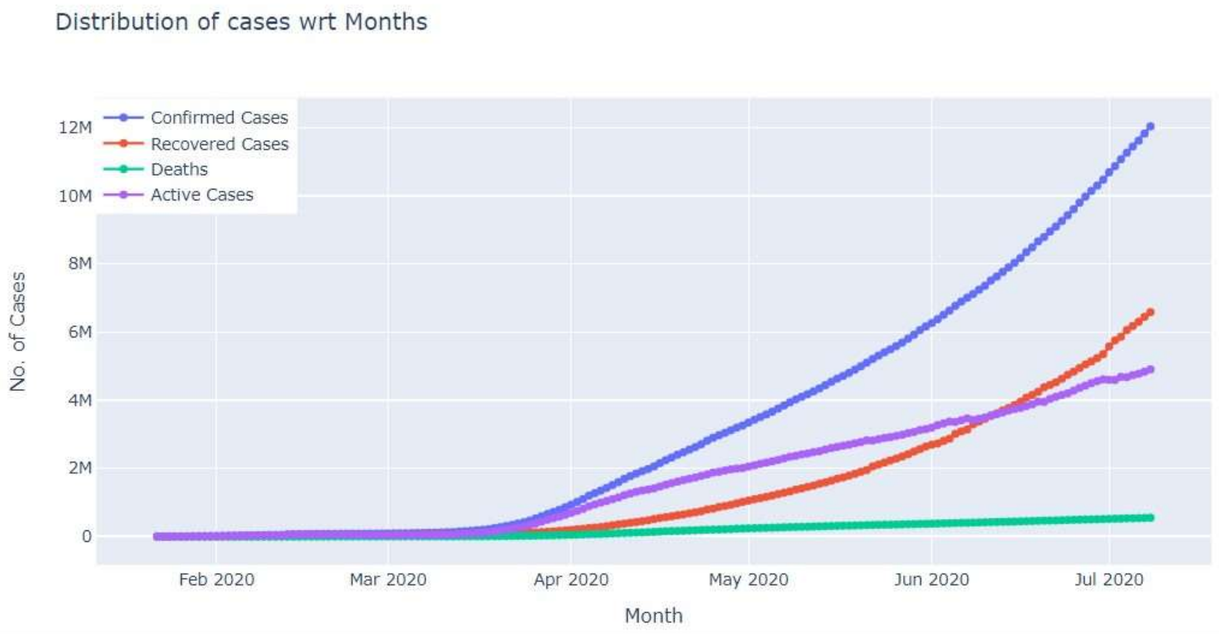
By using the DataFrame, we visualize the top 20 states where the active case is high by plotting a graph as below:

```
In [15]: y_label = State_US_data[0:20]#State_US_data['Confirmed']>1000]
         #y_label
         #State_US_data.index[0:20]
         fig4 = px.bar(x = State_US_data.index[0:20], y = State_US_data["Active Cases"].iloc[0:20])
         fig4.update_layout(title = "Top 20 State Active cases",xaxis_title = "Top 20 States",yaxis_title = "Total No. of Cases")
         #px.bar(x = US_Data["Province/State"], y= US_Data["Confirmed"])#- (US_Data['Recovered'] + US_Data['Deaths']))
         fig4.show()
```



Now we use the same Dataframe to plot for distribution of cases in the United Stated wrt Months.

```
In [16]: fig5 = go.Figure()
         fig5.add_trace(go.Scatter(x=datewise.index,y = datewise["Confirmed"],mode='lines+markers',name='Confirmed Cases'))
         fig5.add_trace(go.Scatter(x=datewise.index,y = datewise["Recovered"],mode='lines+markers',name="Recovered Cases"))
         fig5.add_trace(go.Scatter(x=datewise.index,y = datewise["Deaths"],mode='lines+markers',name='Deaths'))
         fig5.add_trace(go.Scatter(x=datewise.index,y=datewise["Active Cases"],mode='lines+markers',name='Active Cases'))
         fig5.update_layout(title='Distribution of cases wrt Months',xaxis_title='Month',yaxis_title='No. of Cases',legend = dict(x=0,y=1,
         fig5.show()
```



Mortality Rate vs Recovery Rate:

So firstly, what is Mortality Rate?

According to Wikipedia: Mortality Rate, or death rate, is a measure of the number of deaths in a particular population, scaled to the size of that population, per unit of time.

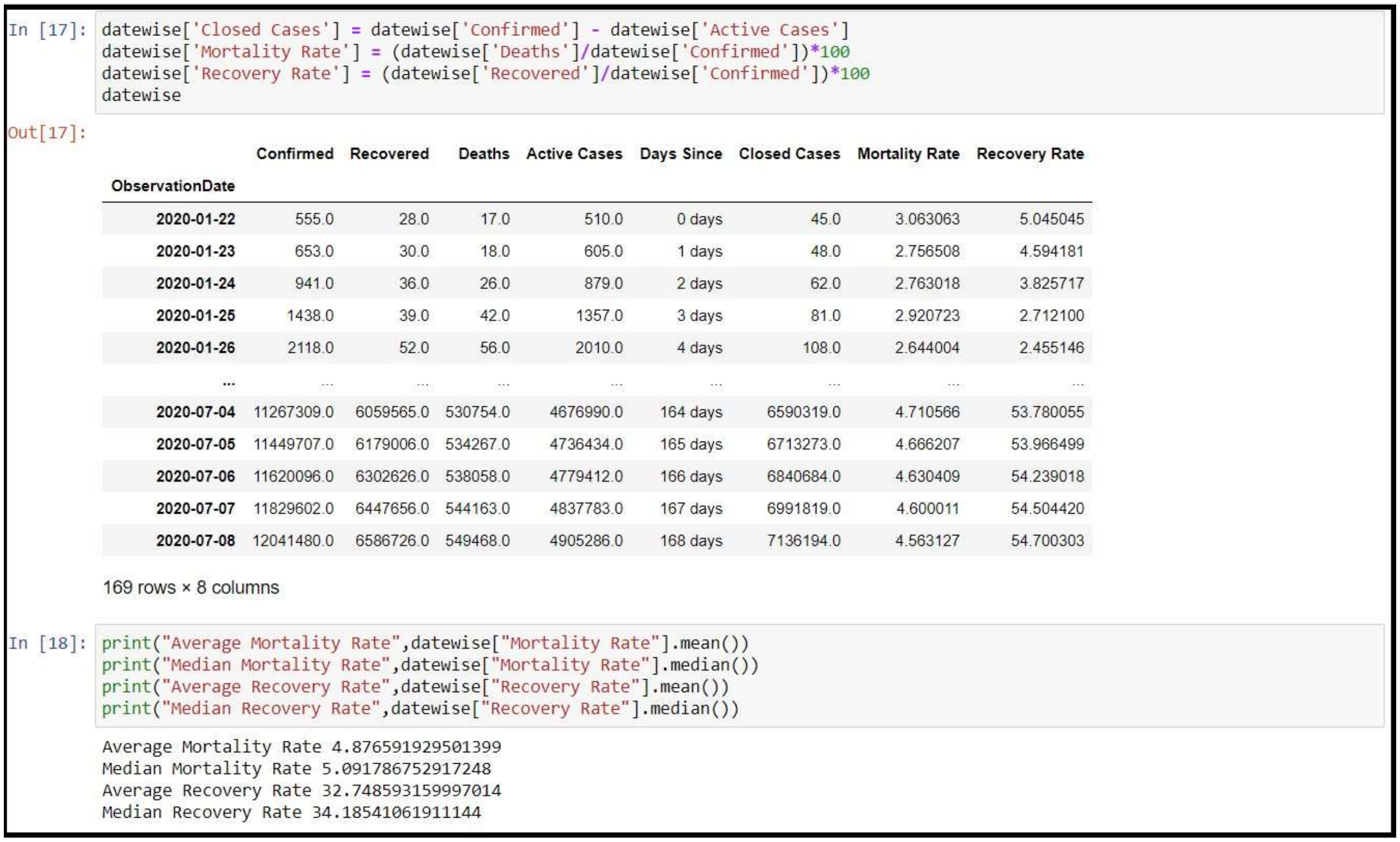
So, to give rise to a Dataframe where we can calculate Mortality Rate and visualize it in a graph.

We use the already existing datewise Dataframe and we add a new column named ‘Mortality Rate’ by diving confirmed cases with death cases and multiplying it by 100.

i.e.: Mortality Rate = (Deaths/Confirmed)*100

And similarly, we do for Recovery Rate.

i.e.: Recovery Rate = (Recovered/Confirmed)*100



```
In [17]: datewise['Closed Cases'] = datewise['Confirmed'] - datewise['Active Cases']
         datewise['Mortality Rate'] = (datewise['Deaths']/datewise['Confirmed'])*100
         datewise['Recovery Rate'] = (datewise['Recovered']/datewise['Confirmed'])*100
         datewise
```

Out[17]:

	Confirmed	Recovered	Deaths	Active Cases	Days Since	Closed Cases	Mortality Rate	Recovery Rate
ObservationDate								
2020-01-22	555.0	28.0	17.0	510.0	0 days	45.0	3.063063	5.045045
2020-01-23	653.0	30.0	18.0	605.0	1 days	48.0	2.756508	4.594181
2020-01-24	941.0	36.0	26.0	879.0	2 days	62.0	2.763018	3.825717
2020-01-25	1438.0	39.0	42.0	1357.0	3 days	81.0	2.920723	2.712100
2020-01-26	2118.0	52.0	56.0	2010.0	4 days	108.0	2.644004	2.455146
...	...	...	...	...	...	...	...	...
2020-07-04	11267309.0	6059565.0	530754.0	4676990.0	164 days	6590319.0	4.710566	53.780055
2020-07-05	11449707.0	6179006.0	534267.0	4736434.0	165 days	6713273.0	4.666207	53.966499
2020-07-06	11620096.0	6302626.0	538058.0	4779412.0	166 days	6840684.0	4.630409	54.239018
2020-07-07	11829602.0	6447656.0	544163.0	4837783.0	167 days	6991819.0	4.600011	54.504420
2020-07-08	12041480.0	6586726.0	549468.0	4905286.0	168 days	7136194.0	4.563127	54.700303

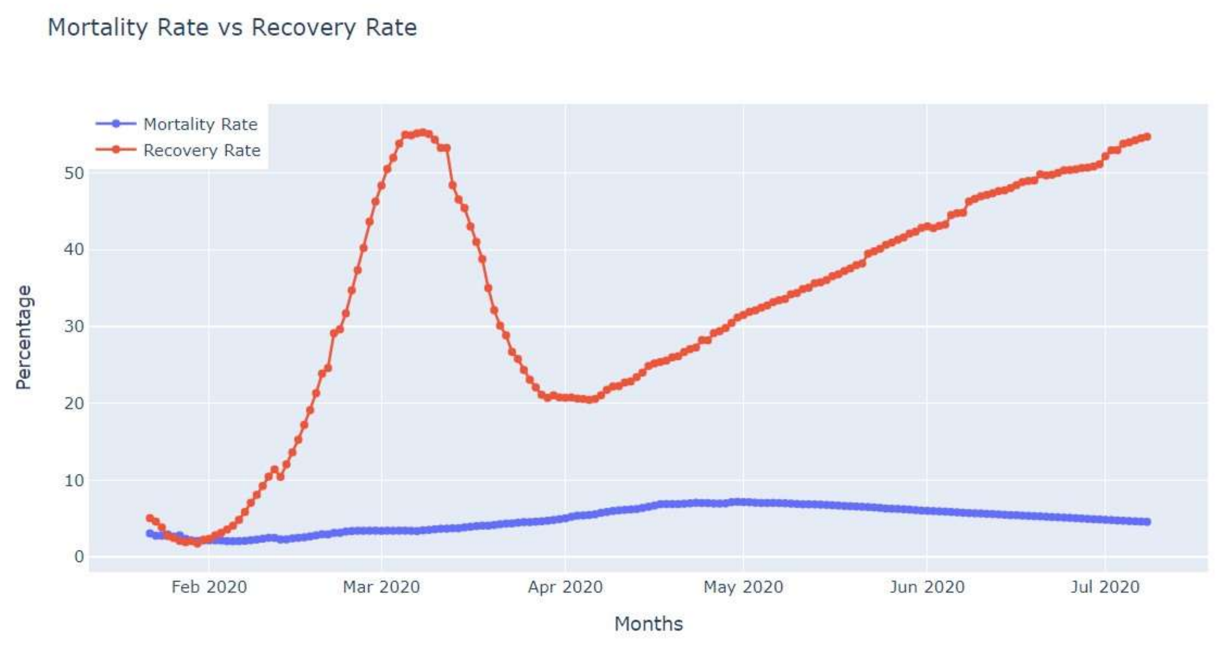
169 rows × 8 columns

```
In [18]: print("Average Mortality Rate",datewise["Mortality Rate"].mean())
         print("Median Mortality Rate",datewise["Mortality Rate"].median())
         print("Average Recovery Rate",datewise["Recovery Rate"].mean())
         print("Median Recovery Rate",datewise["Recovery Rate"].median())

         Average Mortality Rate 4.876591929501399
         Median Mortality Rate 5.091786752917248
         Average Recovery Rate 32.748593159997014
         Median Recovery Rate 34.18541061911144
```

Now let us visualize our analysis:

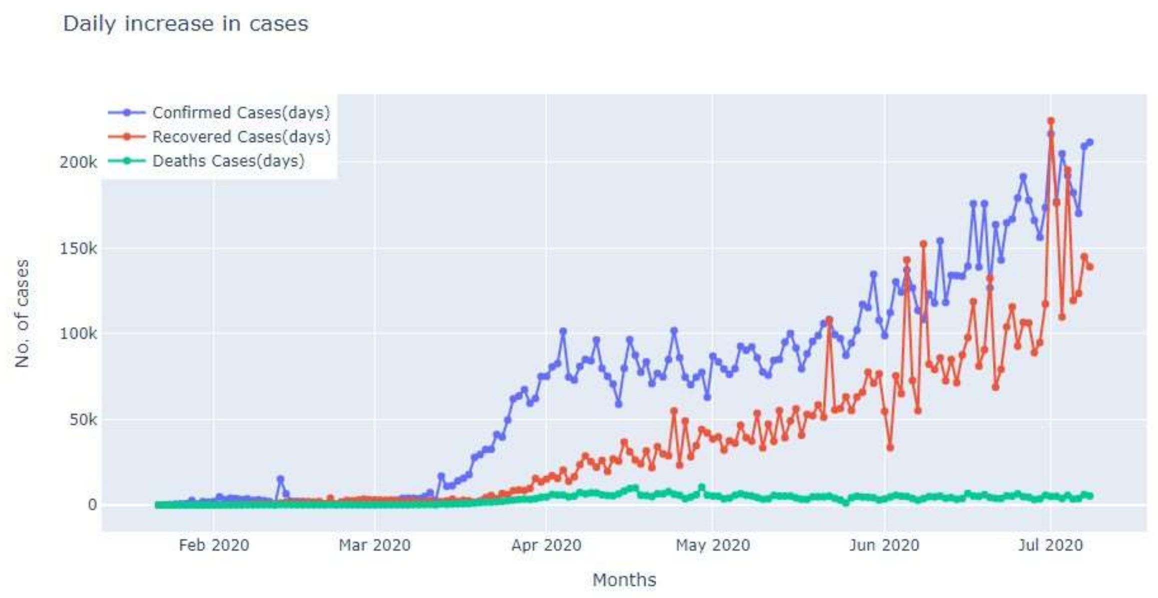
```
In [19]: fig6= go.Figure()
         fig6.add_trace(go.Scatter(x=datewise.index,y=datewise['Mortality Rate'],mode='lines+markers',name='Mortality Rate'))
         fig6.add_trace(go.Scatter(x=datewise.index,y=datewise['Recovery Rate'],mode='lines+markers',name='Recovery Rate'))
         fig6.update_layout(title="Mortality Rate vs Recovery Rate",xaxis_title="Months",yaxis_title="Percentage",legend = dict(x=0,y=1,tr
         fig6.show()
```



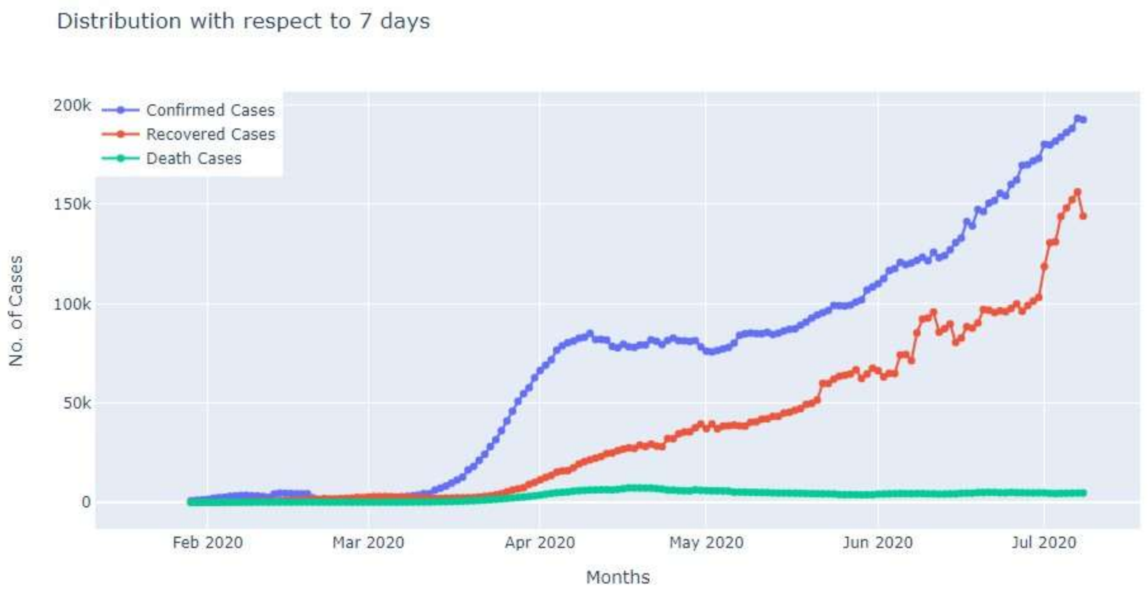
Now we plot for daily increasing cases & Cases distributed among 7 days I.e. per week.

```
In [20]: print("Average increase in number of confirmed cases (by days:)",datewise['Confirmed'].diff().fillna(0).mean())
         print("Average increase in number of recovered cases (by days:)",datewise['Recovered'].diff().fillna(0).mean())
         print("Average increase in number of Death cases (by days:)",datewise['Deaths'].diff().fillna(0).mean())
         fig7 = go.Figure()
         fig7.add_trace(go.Scatter(x=datewise.index,y=datewise['Confirmed'].diff().fillna(0),mode='lines+markers',name='Confirmed Cases(da
         fig7.add_trace(go.Scatter(x=datewise.index,y=datewise['Recovered'].diff().fillna(0),mode='lines+markers',name='Recovered Cases(da
         fig7.add_trace(go.Scatter(x=datewise.index,y=datewise['Deaths'].diff().fillna(0),mode='lines+markers',name='Deaths Cases(days)'))
         fig7.update_layout(title='Daily increase in cases',xaxis_title='Months',yaxis_title='No. of cases',legend=dict(x=0,y=1,traceorder
         fig7.show()
```

```
Average increase in number of confirmed cases (by days:) 71248.07692307692
Average increase in number of recovered cases (by days:) 38974.54437869822
Average increase in number of Death cases (by days:) 3251.189349112426
```



```
In [21]: fig8 = go.Figure()
         fig8.add_trace(go.Scatter(x=datewise.index,y=datewise['Confirmed'].diff().rolling(window=7).mean(),mode='lines+markers',name='Cor
         fig8.add_trace(go.Scatter(x=datewise.index,y=datewise['Recovered'].diff().rolling(window=7).mean(),mode='lines+markers',name='Rec
         fig8.add_trace(go.Scatter(x=datewise.index,y = datewise['Deaths'].diff().rolling(window=7).mean(),mode='lines+markers',name='Deat
         fig8.update_layout(title='Distribution with respect to 7 days',xaxis_title='Months',yaxis_title='No. of Cases',legend=dict(x=0,y=
         fig8.show()
```



GROWTHFACTOR:

The growth factor is the factor by which a quantity multiplies itself over time. The formula used is:

Formula: Every day's new (Confirmed,Recovered,Deaths) / new (Confirmed,Recovered,Deaths) on the previous day

```
In [22]: print("Average growth factor of number of Confirmed Cases: ",(datewise["Confirmed"]/datewise["Confirmed"].shift()).mean())
         print("Median growth factor of number of Confirmed Cases: ",(datewise["Confirmed"]/datewise["Confirmed"].shift()).median())
         print("Average growth factor of number of Recovered Cases: ",(datewise["Recovered"]/datewise["Recovered"].shift()).mean())
         print("Median growth factor of number of Recovered Cases: ",(datewise["Recovered"]/datewise["Recovered"].shift()).median())
         print("Average growth factor of number of Death Cases: ",(datewise["Deaths"]/datewise["Deaths"].shift()).mean())
         print("Median growth factor of number of Death Cases: ",(datewise["Deaths"]/datewise["Deaths"].shift()).median())

         Average growth factor of number of Confirmed Cases:  1.0655540406993327
         Median growth factor of number of Confirmed Cases:  1.024710962630524
         Average growth factor of number of Recovered Cases:  1.080868628448917
         Median growth factor of number of Recovered Cases:  1.0404390329401614
         Average growth factor of number of Death Cases:  1.0674768830995875
         Median growth factor of number of Death Cases:  1.0283354658536563
```