Covid-19 Data Visualization using Plotly Express (50 graphs)

For Fast Processing :Go to runtime(headbar)> Change runtime type > GPU

Task 1: Importing Necessary Libraries

Importing Pandas and Matplotlib

```
import pandas as pd       #Data analysis and Manipulation
import matplotlib.pyplot as plt   #Data Visualization
```

Importing Plotly

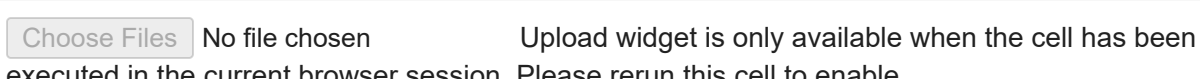
```
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go                # Importing Plotly
import plotly.express as px
```

Initializing Plotly (#Plotly Consumes a lot of computing power, so its default mode is off in Google Colab, Hence we need to initialize it)

```
import plotly.io as pio
pio.renderers.default = 'colab'     # To initialize plotly
```

Task 2: Importing the Datasets

```
from google.colab import files
files.upload()
```




```
df1= pd.read_csv("covid.csv")
df1.head()
```

	Country/Region	Continent	Population	TotalCases	NewCases	TotalDeaths	New
0	USA	North America	3.311981e+08	5032179	NaN	162804.0	
1	Brazil	South America	2.127107e+08	2917562	NaN	98644.0	
2	India	Asia	1.381345e+09	2025409	NaN	41638.0	

```
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving covid_grouped.csv to covid_grouped.csv
{'covid_grouped.csv': b"Date,Country/Region,Confirmed,Deaths,Recovered,Active,New
```

```
df2= pd.read_csv("covid_grouped.csv")
df2.head()
```

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	rec
0	2020-01-22	Afghanistan	0	0	0	0	0	0	
1	2020-01-22	Albania	0	0	0	0	0	0	
2	2020-01-22	Algeria	0	0	0	0	0	0	

```
df2.tail()
```

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths
35151	2020-07-27	West Bank and Gaza	10621	78	3752	6791	152	2
35152	2020-07-27	Western Sahara	10	1	8	1	0	0
35153	2020-07-27	Yemen	1691	483	833	375	10	4

Dataset 1 and Dataset 2 Columns

```
df1.columns
```

```
Index(['Country/Region', 'Continent', 'Population', 'TotalCases', 'NewCases',
       'TotalDeaths', 'NewDeaths', 'TotalRecovered', 'NewRecovered',
```

```
       'ActiveCases', 'Serious,Critical', 'Tot Cases/1M pop', 'Deaths/1M pop',
       'TotalTests', 'Tests/1M pop', 'WHO Region', 'iso_alpha'],
      dtype='object')
```

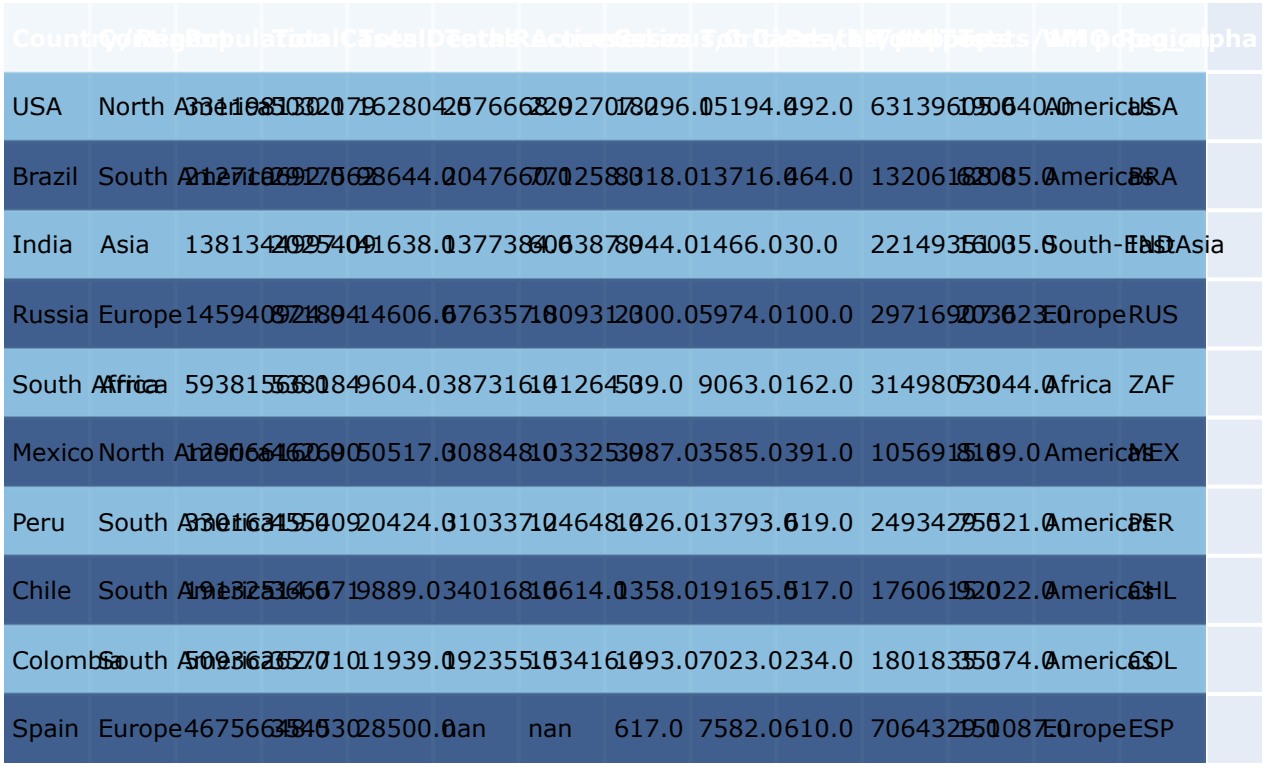
```
df1.drop(['NewCases','NewDeaths', 'NewRecovered'], axis=1, inplace=True)
```

```
df1.head()
```

	Country/Region	Continent	Population	TotalCases	TotalDeaths	TotalRecovere
0	USA	North America	3.311981e+08	5032179	162804.0	2576668.
1	Brazil	South America	2.127107e+08	2917562	98644.0	2047660.
2	India	Asia	1.381345e+09	2025409	41638.0	1377384.
3	Russia	Europe	1.459409e+08	871894	14606.0	676357.
4	South Africa	Africa	5.938157e+07	538184	9604.0	387316.

Creating Tables

```
from plotly.figure_factory import create_table
table=create_table(df1.head(10), colorscale="blues")
py.iplot(table)
```



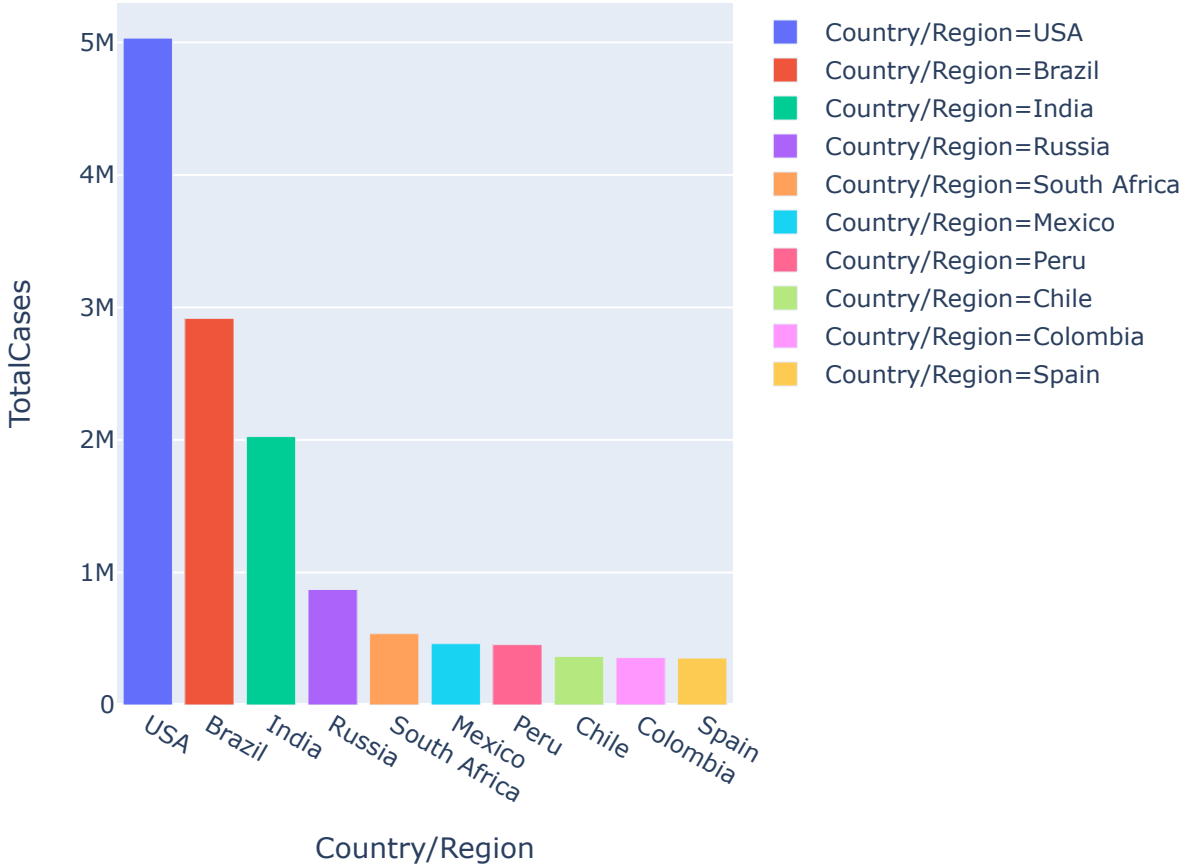
Task 3: Quick Visualizations with Custom Bar Charts

```
#dataset 1, x,y,color,height,hover_data
df1.columns
```

```
Index(['Country/Region', 'Continent', 'Population', 'TotalCases',
       'TotalDeaths', 'TotalRecovered', 'ActiveCases', 'Serious,Critical',
       'Tot Cases/1M pop', 'Deaths/1M pop', 'TotalTests', 'Tests/1M pop',
       'WHO Region', 'iso_alpha'],
      dtype='object')
```

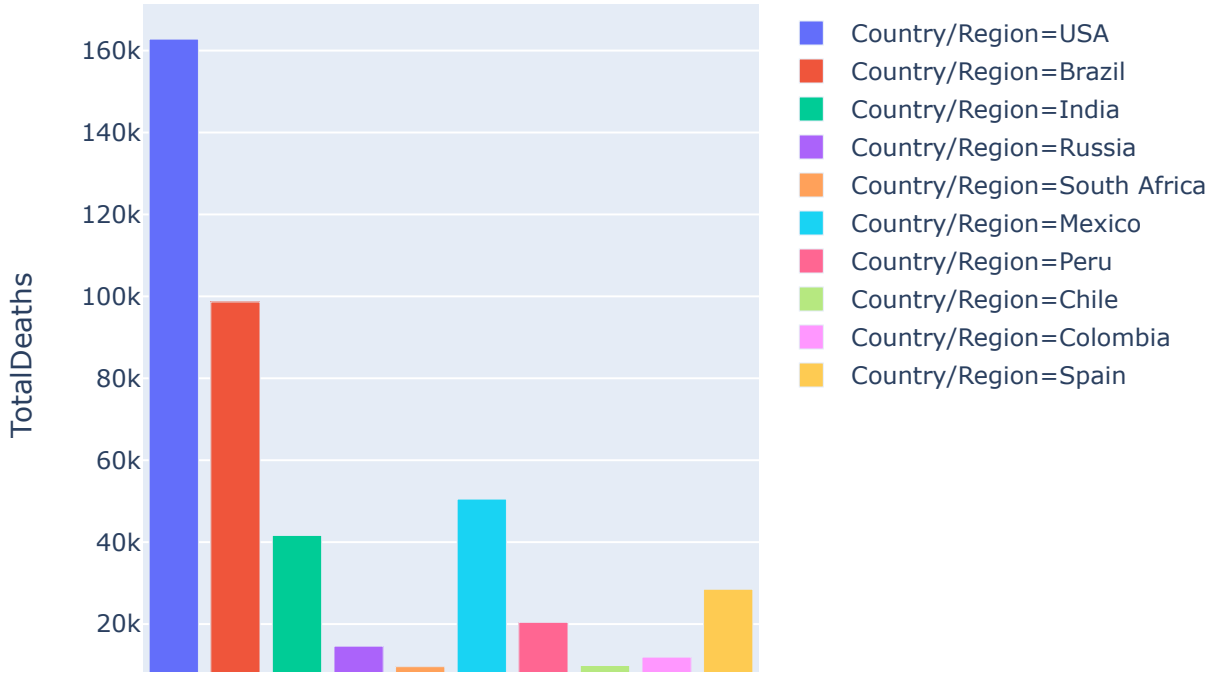
Total Cases vs Countries

```
px.bar(df1.head(10), x='Country/Region', y='TotalCases', color='Country/Region', height=500, hover_data
```



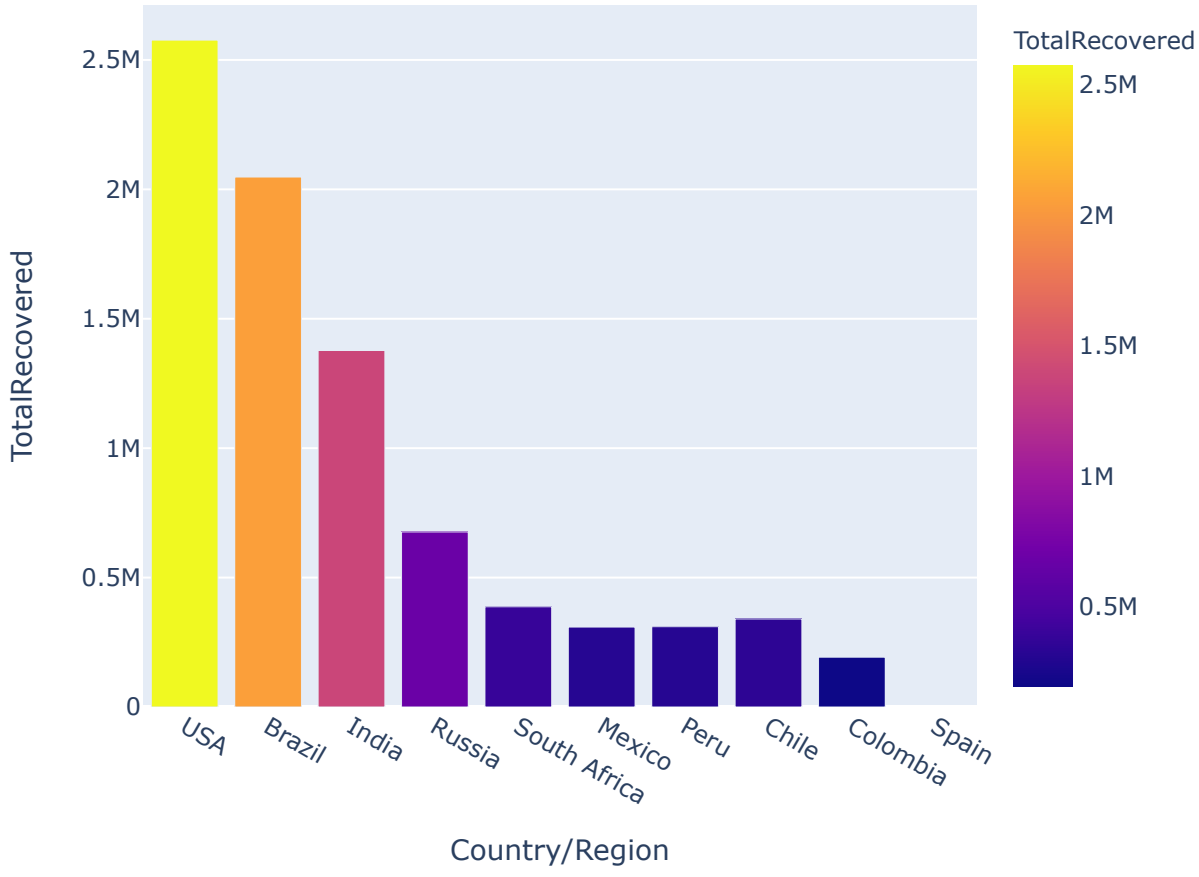
Total Death vs Countries

```
px.bar(df1.head(10), x='Country/Region', y='TotalDeaths', color='Country/Region', height=500, hover_dat
```



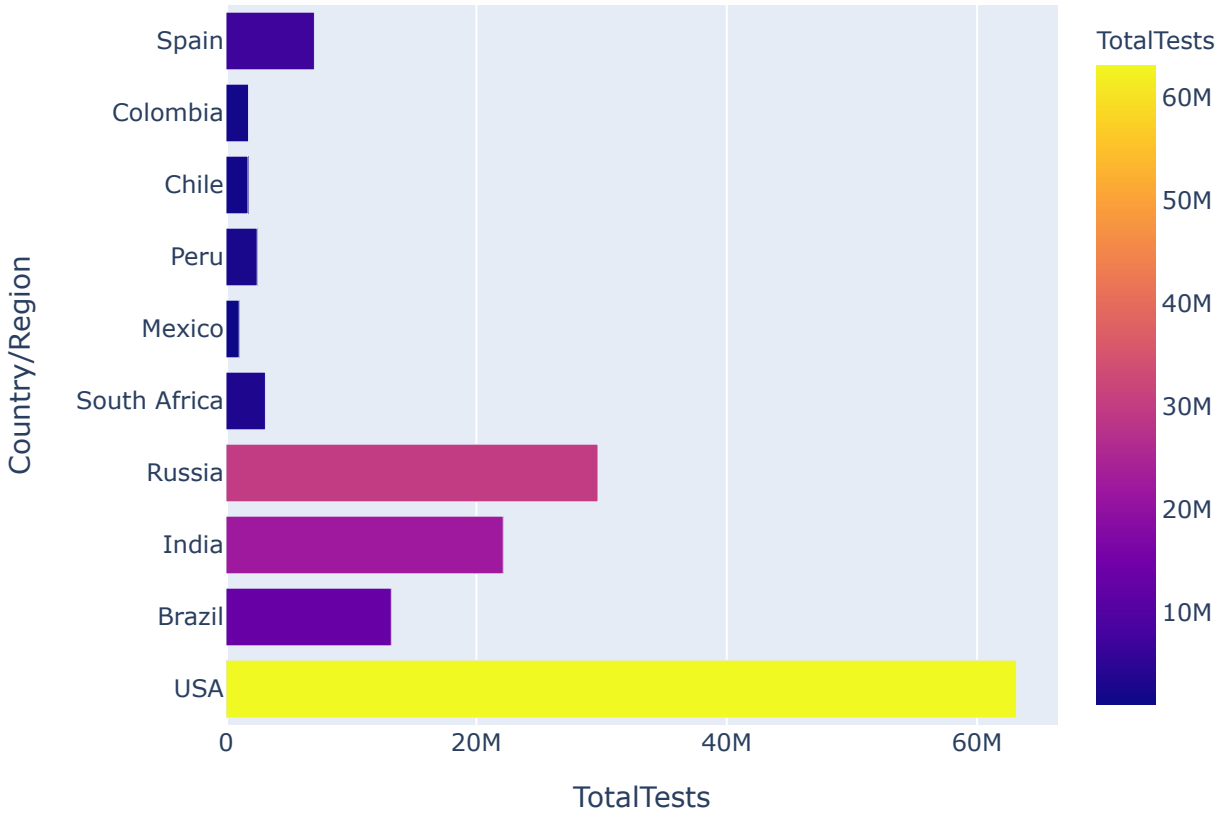
Total Recovered vs Countries

```
px.bar(df1.head(10), x='Country/Region', y='TotalRecovered', color='TotalRecovered', height=500, hover_
```



Total Tests vs Countries (Orientation)

```
px.bar(df1.head(10), x='TotalTests', y='Country/Region', color='TotalTests', orientation="h",height=500
```




```
px.bar(df1.head(10), x='TotalTests', y='Continent', color='TotalTests', orientation="h",height=500, hov
```

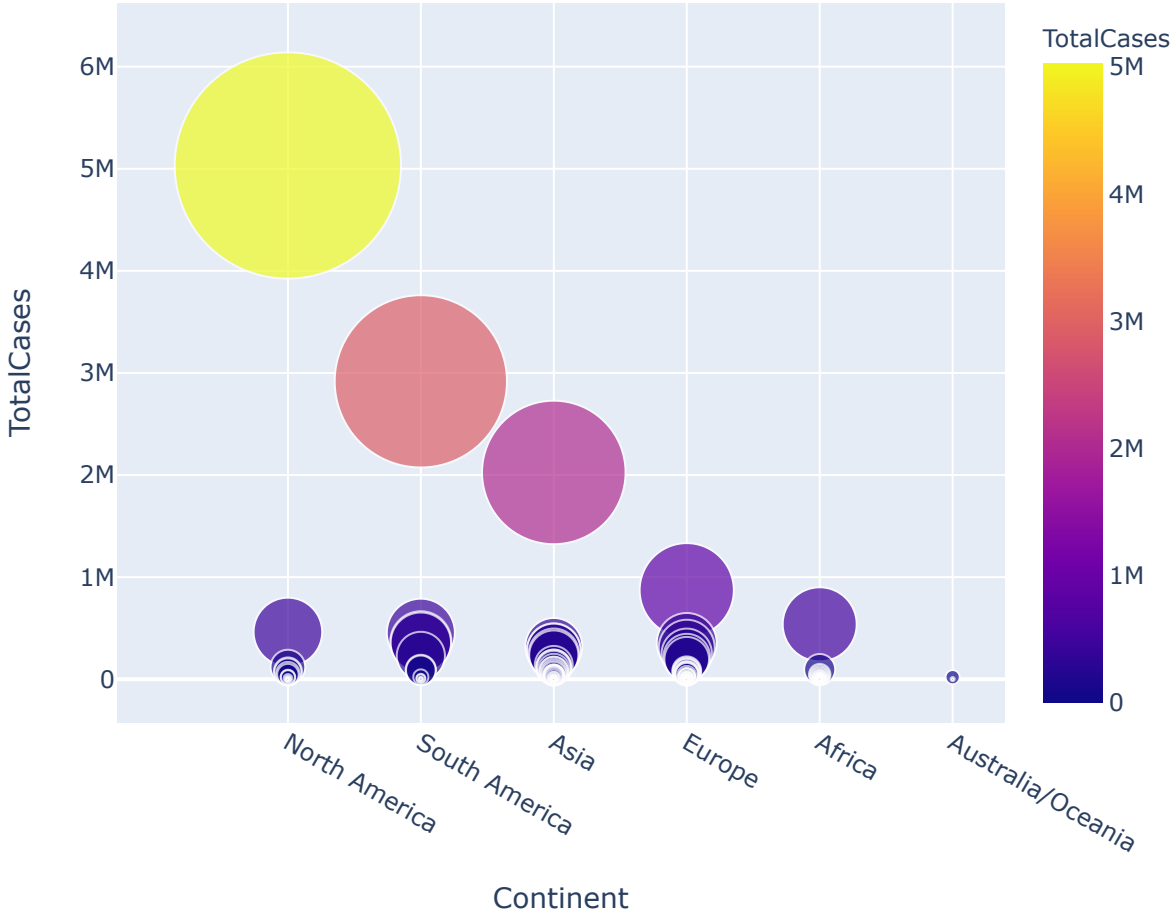
Task 4: Data Visualization using Bubble Chart


```
# px.scatter, x,y, size, color, hover_data, size_max, log_x,log_y
df1.columns
```

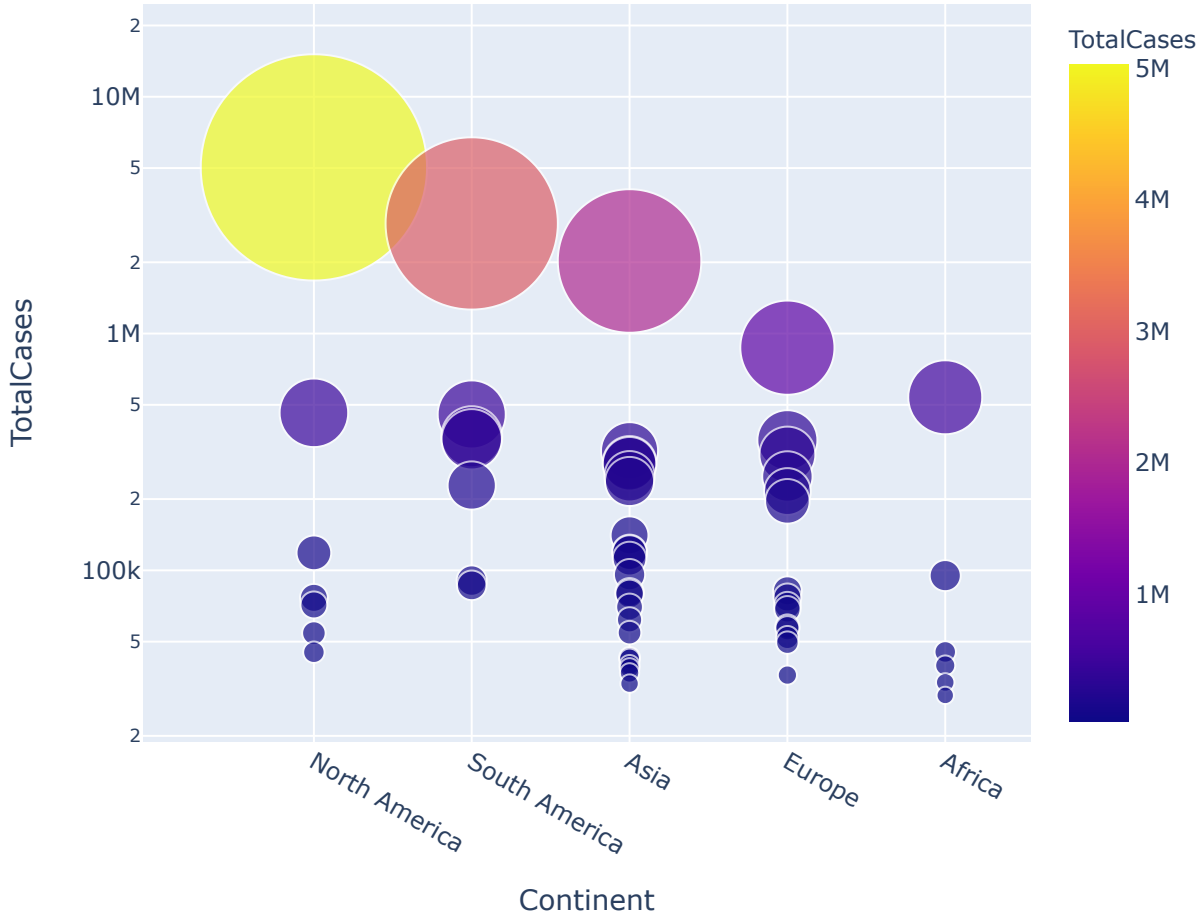
```
Index(['Country/Region', 'Continent', 'Population', 'TotalCases',
       'TotalDeaths', 'TotalRecovered', 'ActiveCases', 'Serious,Critical',
       'Tot Cases/1M pop', 'Deaths/1M pop', 'TotalTests', 'Tests/1M pop',
       'WHO Region', 'iso_alpha'],
      dtype='object')
```

Total Cases vs Continent (50 countries)

```
px.scatter(df1, x='Continent',y='TotalCases', hover_data=['Country/Region', 'Continent'],
           color='TotalCases', size='TotalCases', size_max=80)
```

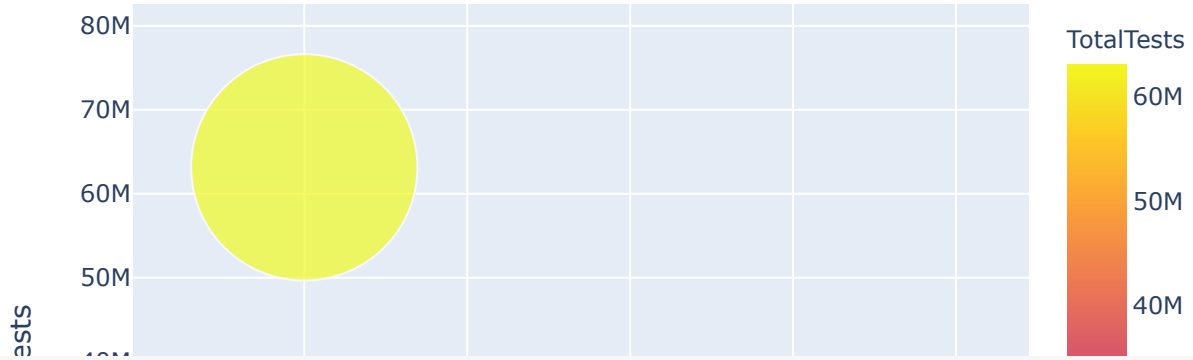


```
px.scatter(df1.head(57), x='Continent',y='TotalCases', hover_data=['Country/Region', 'Continent'],
           color='TotalCases', size='TotalCases', size_max=80, log_y=True)
```

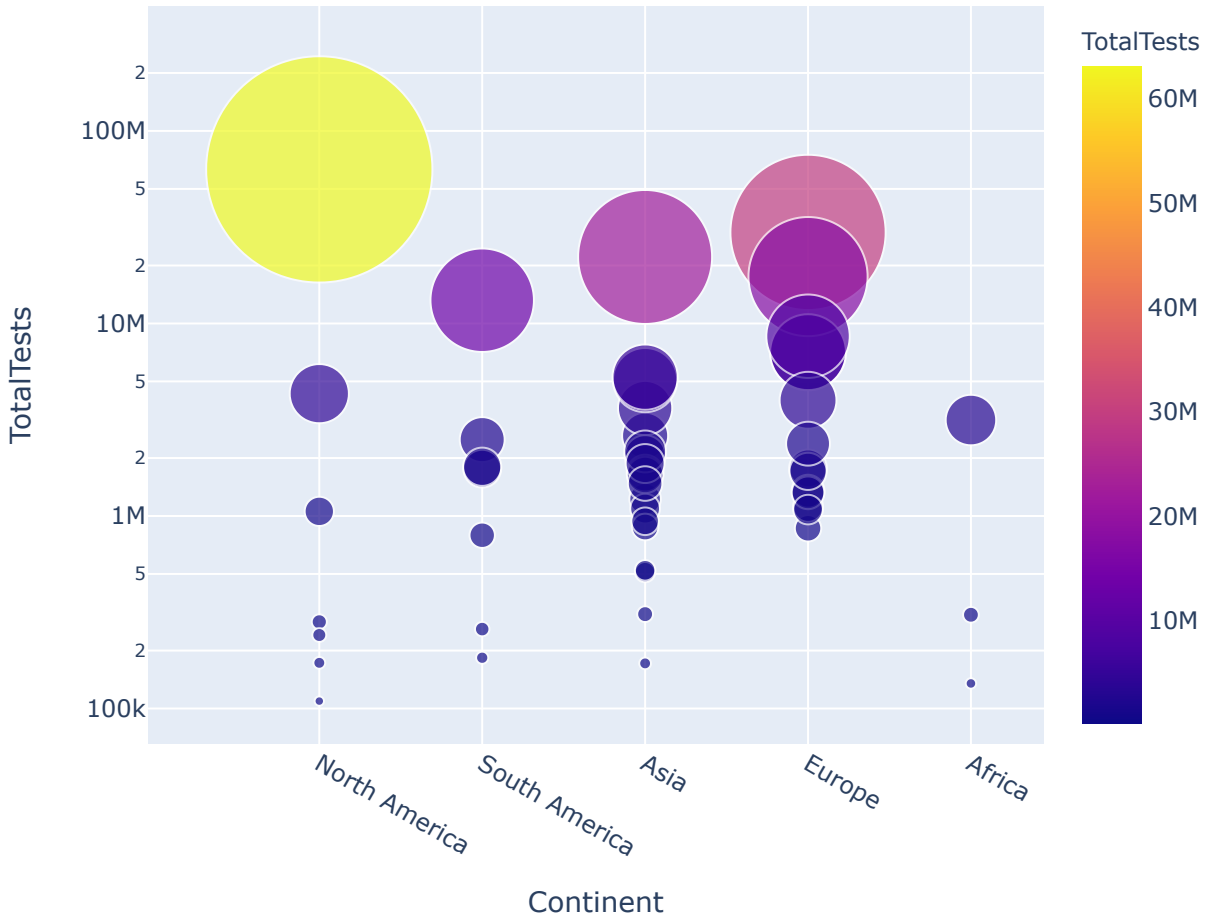


Total Tests vs Continent (50 countries)

```
px.scatter(df1.head(54), x='Continent',y='TotalTests', hover_data=['Country/Region', 'Continent'],
           color='TotalTests', size='TotalTests', size_max=80)
```

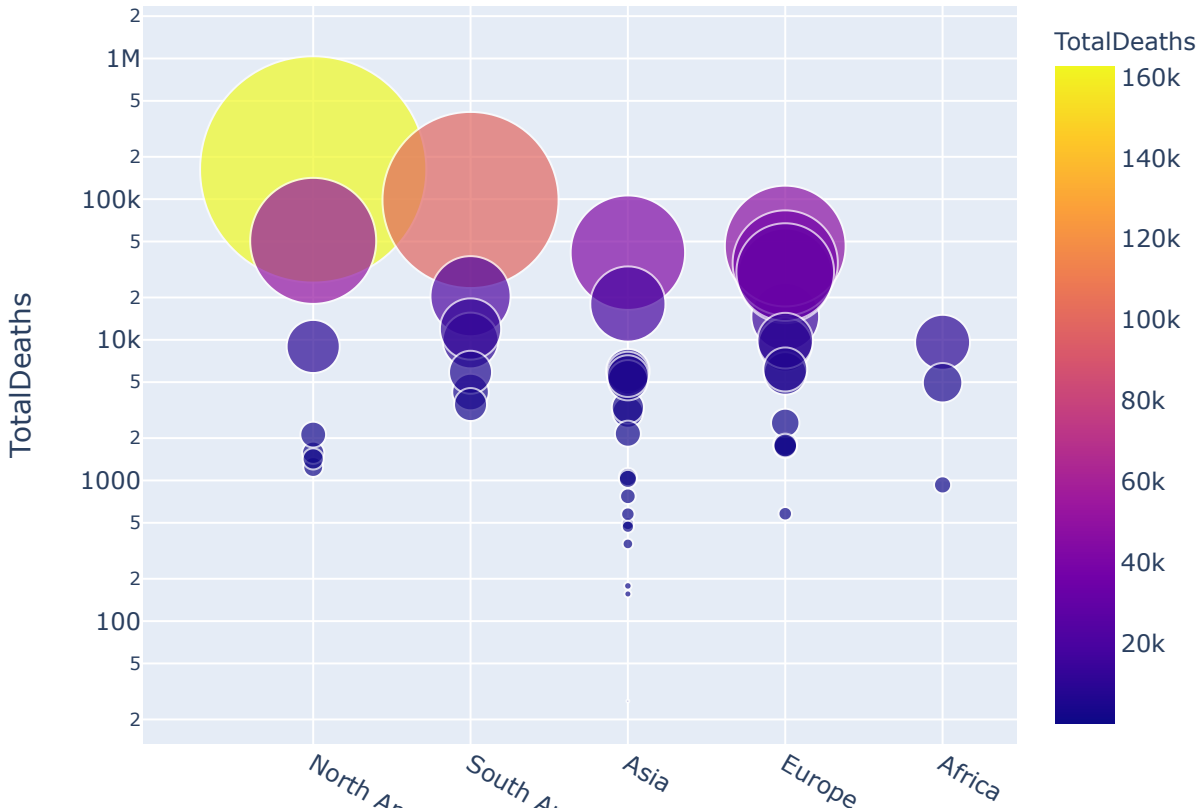



```
px.scatter(df1.head(50), x='Continent',y='TotalTests', hover_data=['Country/Region', 'Continent'],
           color='TotalTests', size='TotalTests', size_max=80, log_y=True)
```



Total Deaths vs Continent (20 countries)

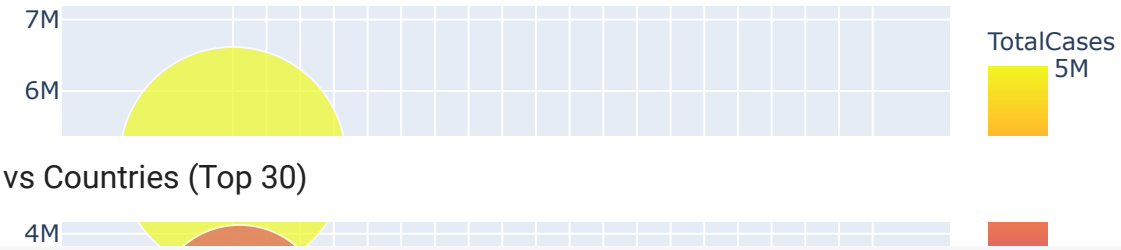
```
px.scatter(df1.head(50), x='Continent',y='TotalDeaths', hover_data=['Country/Region', 'Continent'],
           color='TotalDeaths', size='TotalDeaths', size_max=80, log_y=True)
```



Total Cases vs Countries (All Countries)

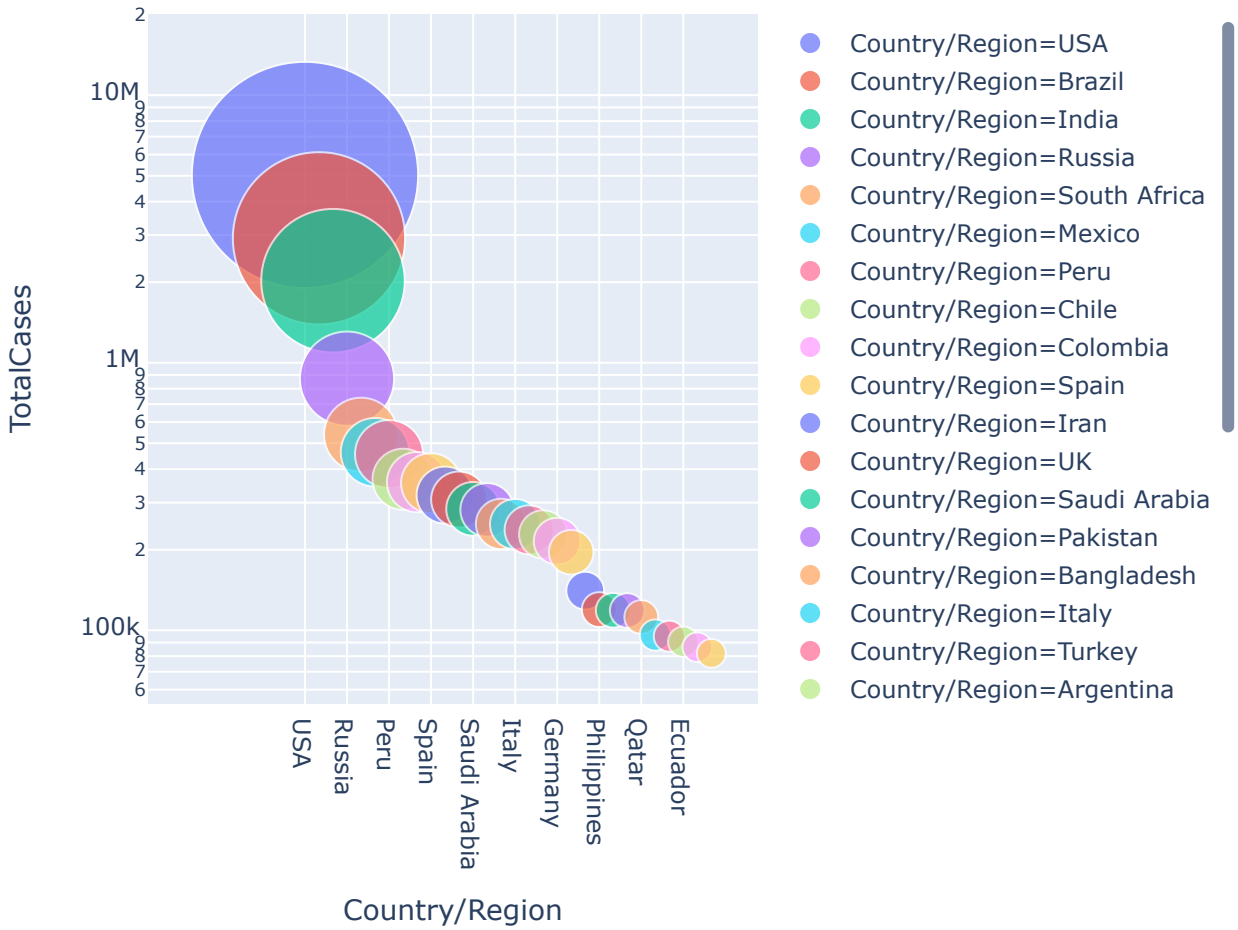
Continent

```
px.scatter(df1.head(100), x='Country/Region', y='TotalCases', hover_data=['Country/Region', 'Continent'
           color='TotalCases', size='TotalCases', size_max=80)
```



Total Cases vs Countries (Top 30)


```
px.scatter(df1.head(30), x='Country/Region', y='TotalCases', hover_data=['Country/Region', 'Continent']
           color='Country/Region', size='TotalCases', size_max=80, log_y=True)
```



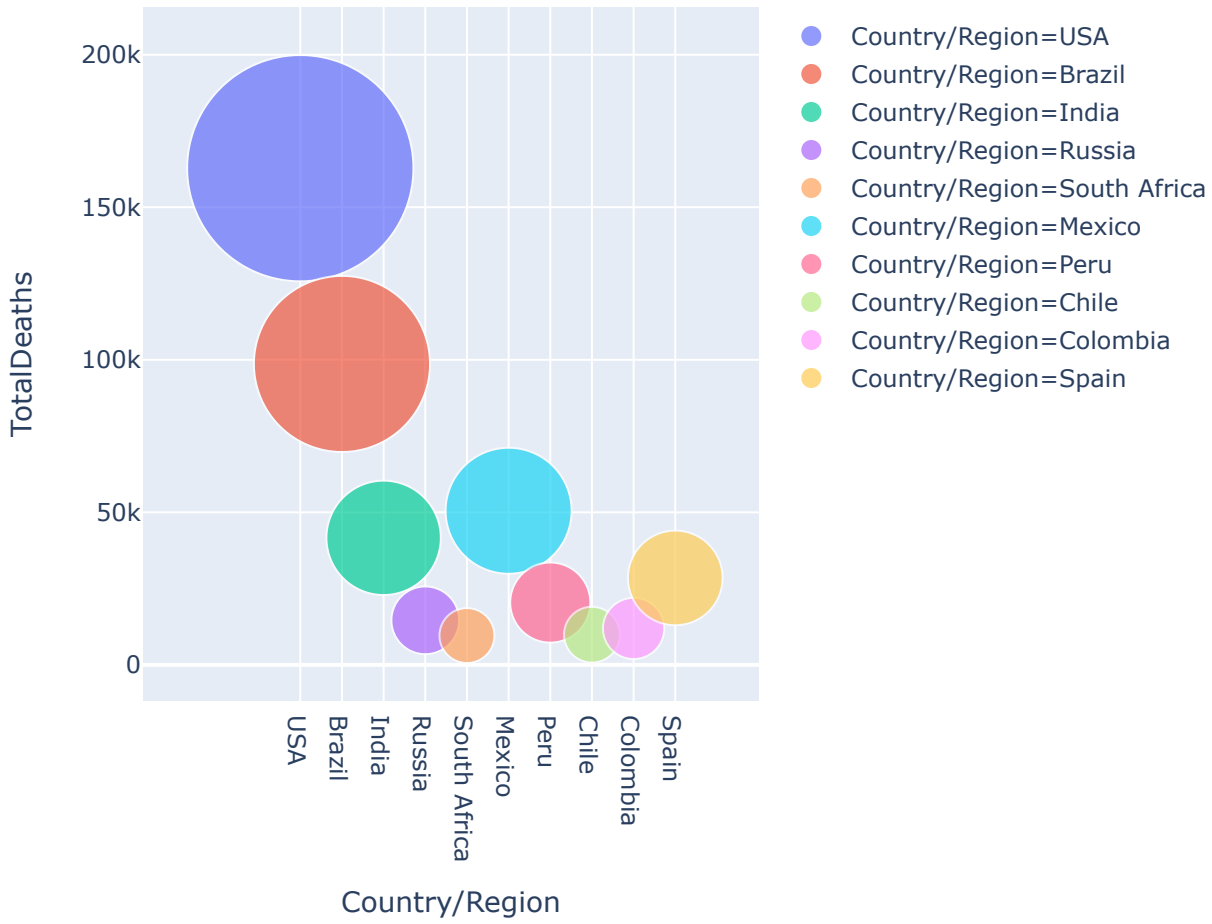

```
df1.columns
```

```
Index(['Country/Region', 'Continent', 'Population', 'TotalCases',
       'TotalDeaths', 'TotalRecovered', 'ActiveCases', 'Serious,Critical',
       'Tot Cases/1M pop', 'Deaths/1M pop', 'TotalTests', 'Tests/1M pop',
       'WHO Region', 'iso_alpha'],
      dtype='object')
```

Total death vs Countries (Top 10)

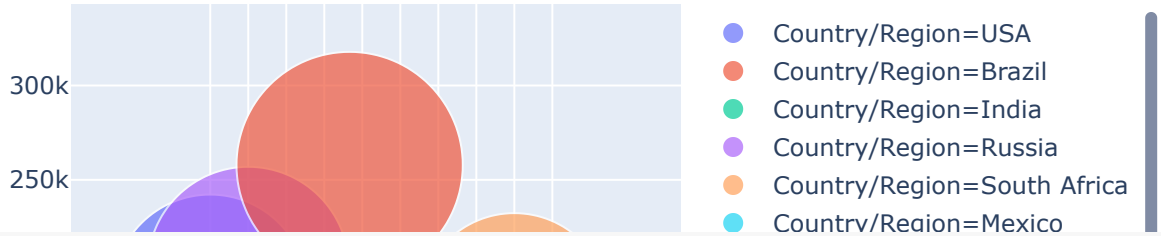
```
px.scatter(df1.head(10), x='Country/Region', y= 'TotalDeaths', hover_data=['Country/Region', 'Continent
```

```
        color='Country/Region', size= 'TotalDeaths', size_max=80)
```

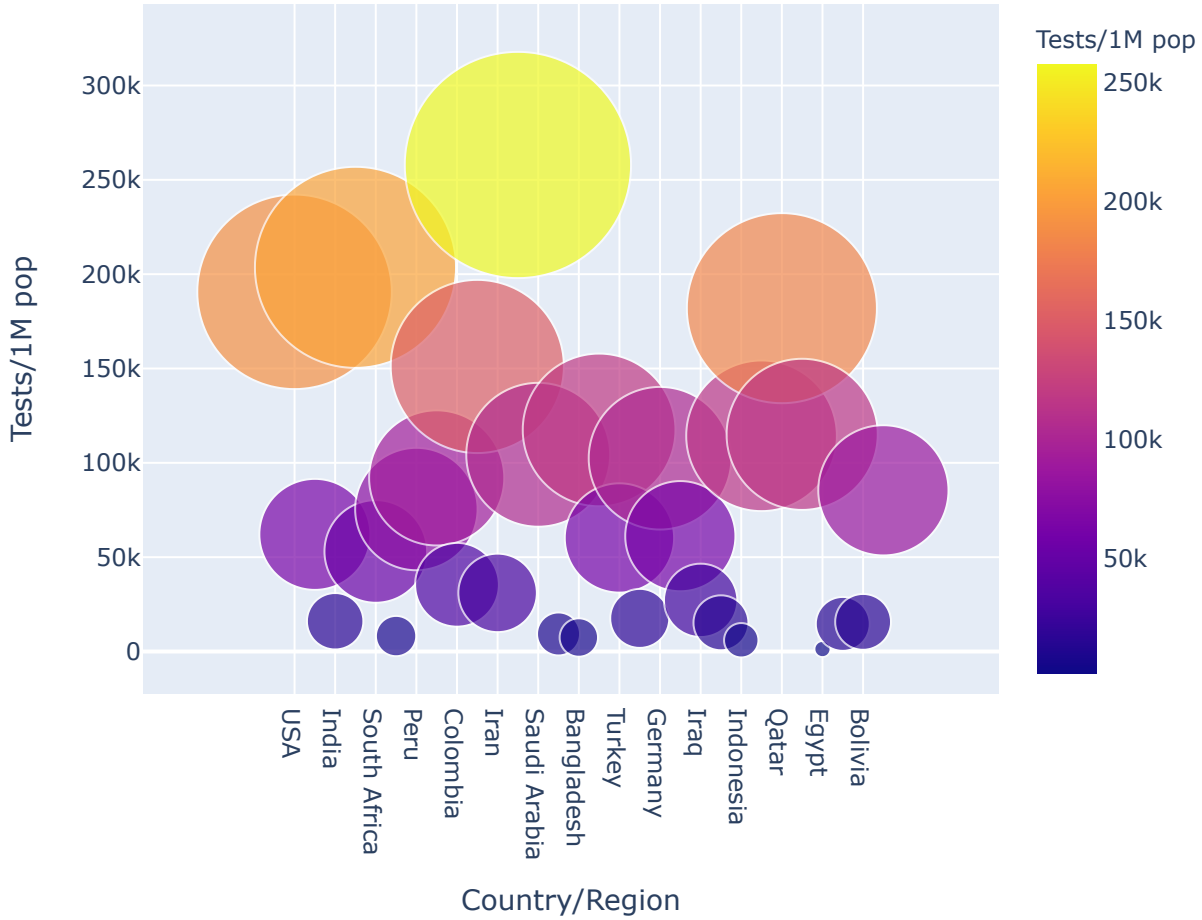


Total Tests/1M vs Countries (Top 50)

```
px.scatter(df1.head(30), x='Country/Region', y= 'Tests/1M pop', hover_data=['Country/Region', 'Continen
           color='Country/Region', size= 'Tests/1M pop', size_max=80)
```




```
px.scatter(df1.head(30), x='Country/Region', y= 'Tests/1M pop', hover_data=['Country/Region', 'Continen
           color='Tests/1M pop', size= 'Tests/1M pop', size_max=80)
```

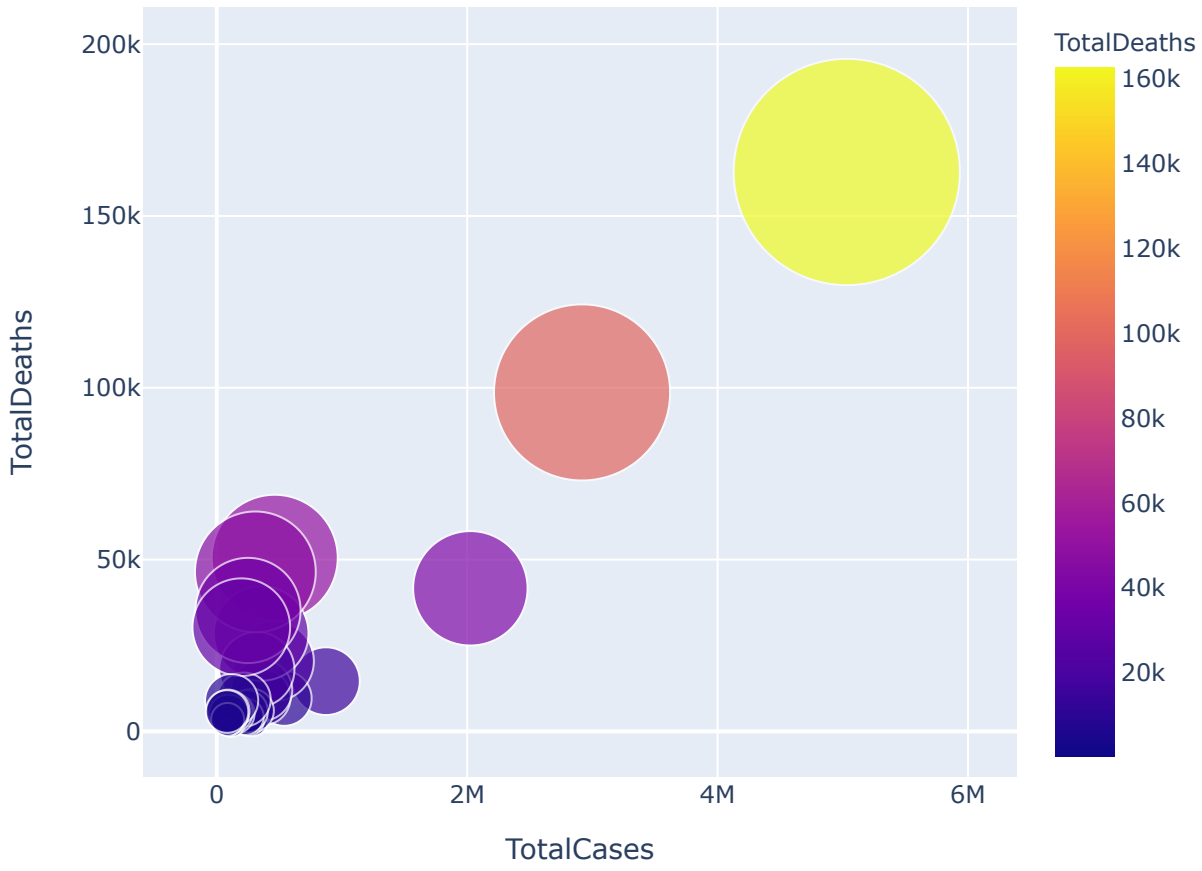



```
df1.columns
```

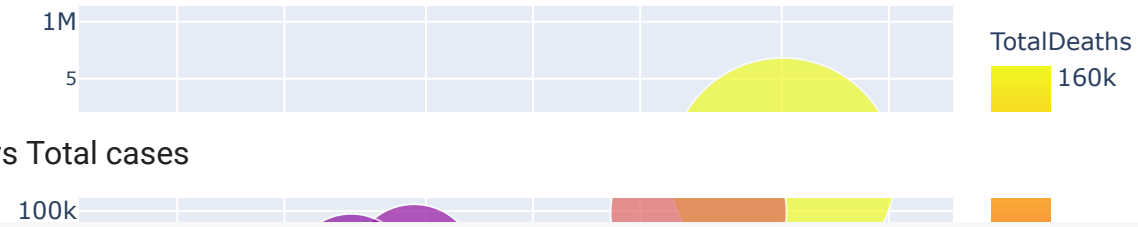
```
Index(['Country/Region', 'Continent', 'Population', 'TotalCases',
       'TotalDeaths', 'TotalRecovered', 'ActiveCases', 'Serious,Critical',
       'Tot Cases/1M pop', 'Deaths/1M pop', 'TotalTests', 'Tests/1M pop',
       'WHO Region', 'iso_alpha'],
      dtype='object')
```

Total case vs Total death

```
px.scatter(df1.head(30), x='TotalCases', y= 'TotalDeaths', hover_data=['Country/Region', 'Continent'],
           color='TotalDeaths', size= 'TotalDeaths', size_max=80)
```

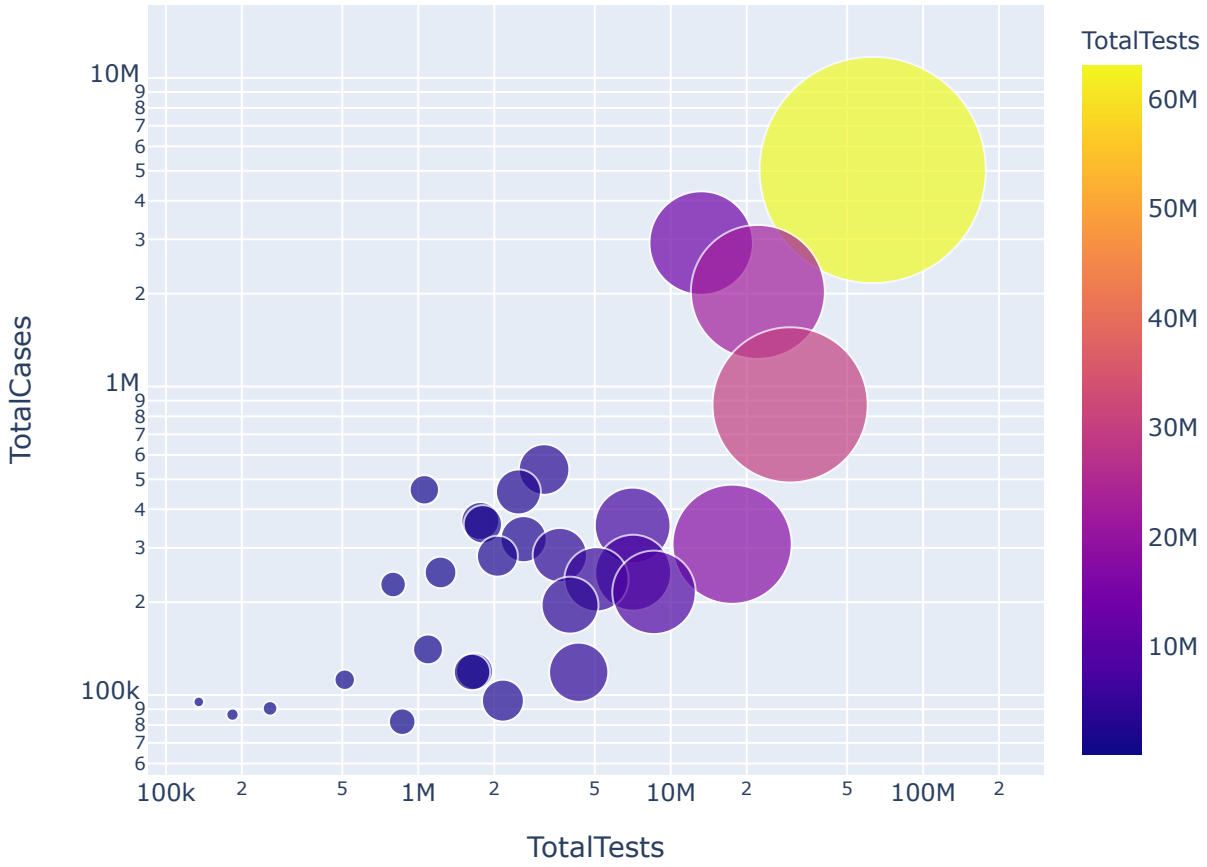



```
px.scatter(df1.head(30), x='TotalCases', y= 'TotalDeaths', hover_data=['Country/Region', 'Continent'],
           color='TotalDeaths', size= 'TotalDeaths', size_max=80, log_x=True, log_y=True)
```



Total test vs Total cases


```
px.scatter(df1.head(30), x='TotalTests', y= 'TotalCases', hover_data=['Country/Region', 'Continent'],
           color='TotalTests', size= 'TotalTests', size_max=80, log_x=True, log_y=True)
```



Advanced Data Visualization using Line graph & Bar graph (Dataset 2)

```
# This Dataset contains DATE column which makes it more appropriate for more advanced Data Visualizatio
```

```
df2.columns
```

```
Index(['Date', 'Country/Region', 'Confirmed', 'Deaths', 'Recovered', 'Active',
       'New cases', 'New deaths', 'New recovered', 'WHO Region', 'iso_alpha'],
      dtype='object')
```

```
df2.head()
```

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	rec
0	2020-01-22	Afghanistan	0	0	0	0	0	0	
1	2020-01-22	Albania	0	0	0	0	0	0	
2	2020-01-22	Algeria	0	0	0	0	0	0	

```
df2.tail()
```

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths
35151	2020-07-27	West Bank and Gaza	10621	78	3752	6791	152	2
35152	2020-07-27	Western Sahara	10	1	8	1	0	0
35153	2020-07-27	Yemen	1691	483	833	375	10	4

Date Vs Confirmed (All Countries)

```
px.bar(df2, x="Date", y="Confirmed", color="Confirmed", hover_data=["Confirmed", "Date", "Country/Regio
```

```
px.bar(df2, x="Date", y="Confirmed", color="Confirmed", hover_data=["Confirmed", "Date", "Country/Regio
```

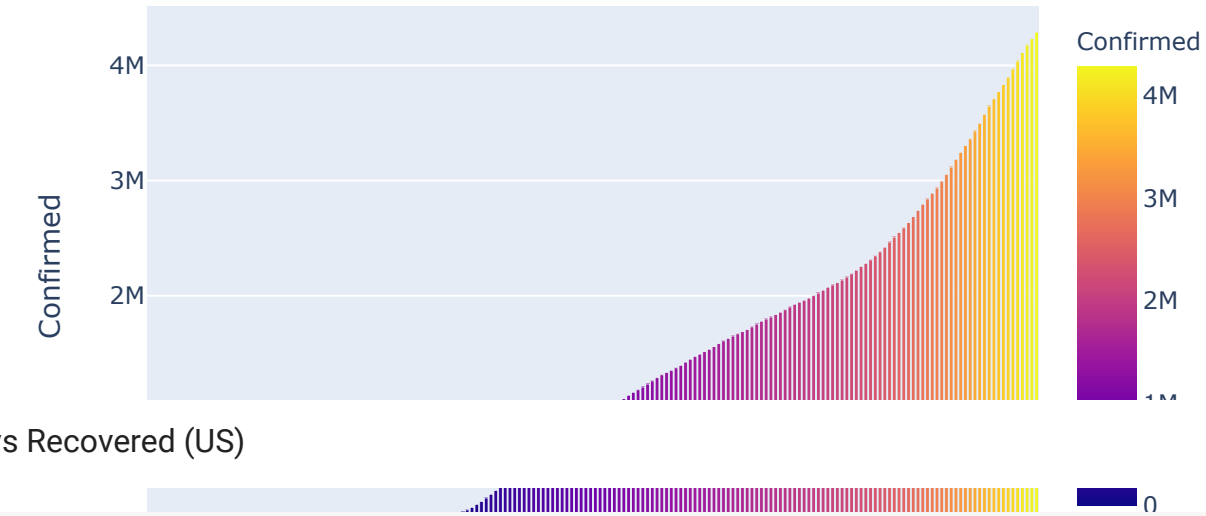
Date vs Death (All countries)[Line Graph]

```
px.bar(df2, x="Date", y="Deaths", color="Deaths", hover_data=["Confirmed", "Date", "Country/Region"],lo
```

```
df_US= df2.loc[df2["Country/Region"]=="US"]
```

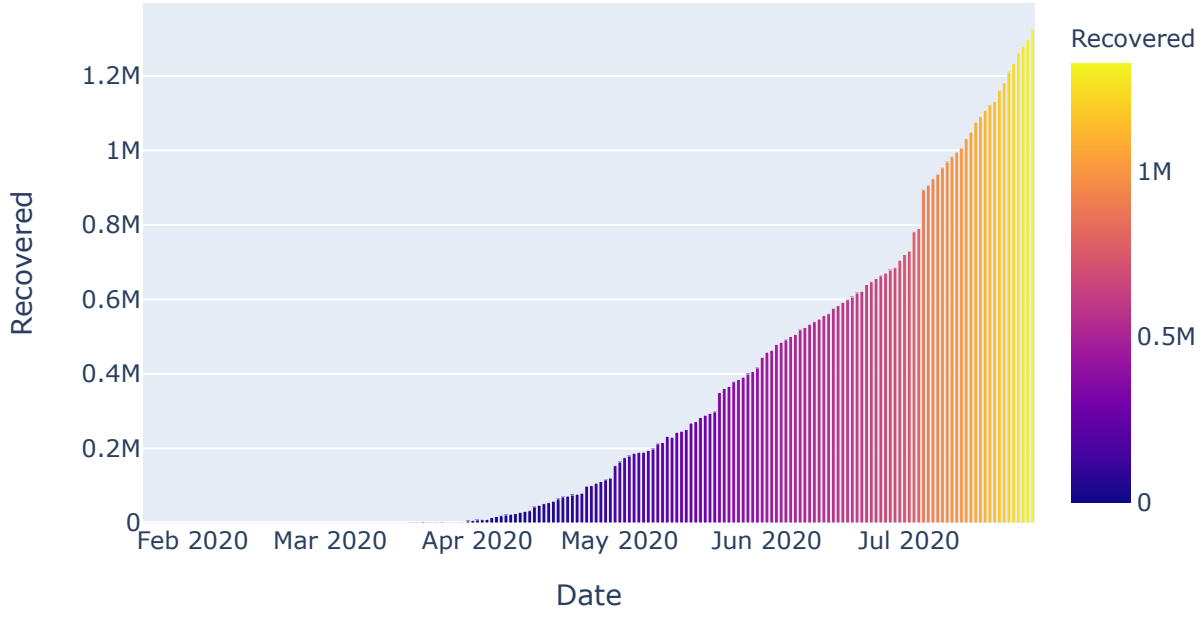
Date Vs Confirmed (US)

```
px.bar(df_US, x="Date", y="Confirmed", color="Confirmed", height=400)
```



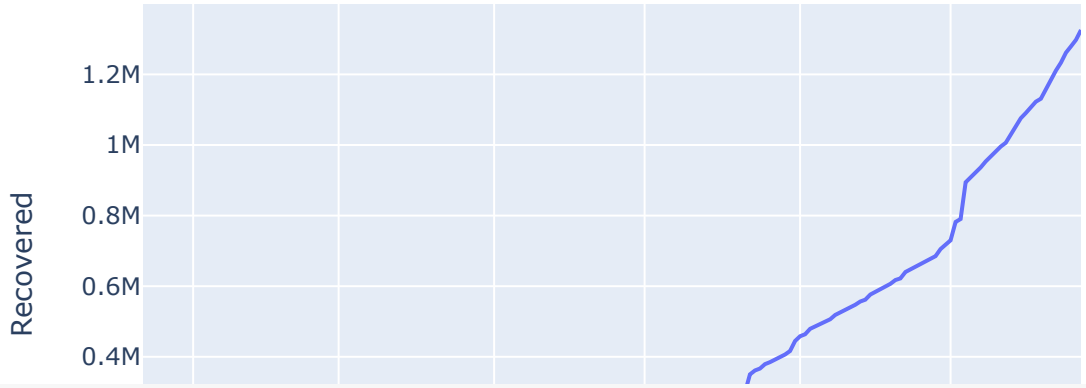
Date vs Recovered (US)

```
px.bar(df_US,x="Date", y="Recovered", color="Recovered", height=400)
```

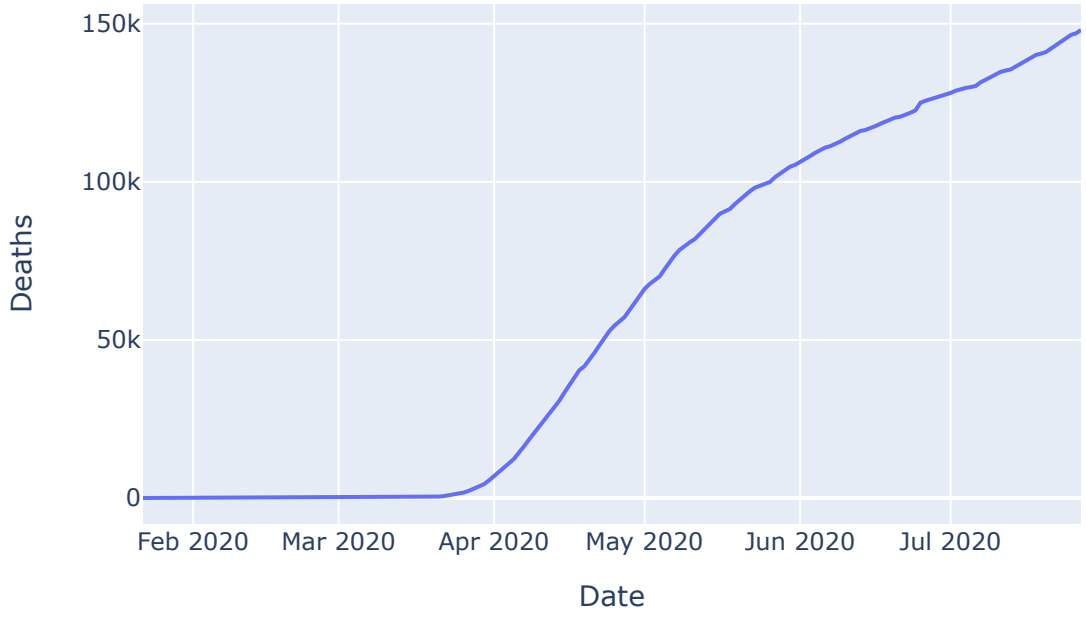


Date vs Death (US)

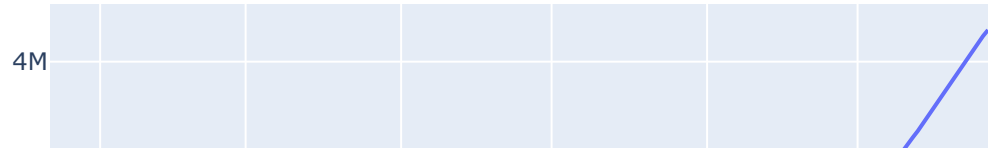
```
px.line(df_US,x="Date", y="Recovered", height=400)
```




```
px.line(df_US,x="Date", y="Deaths", height=400)
```

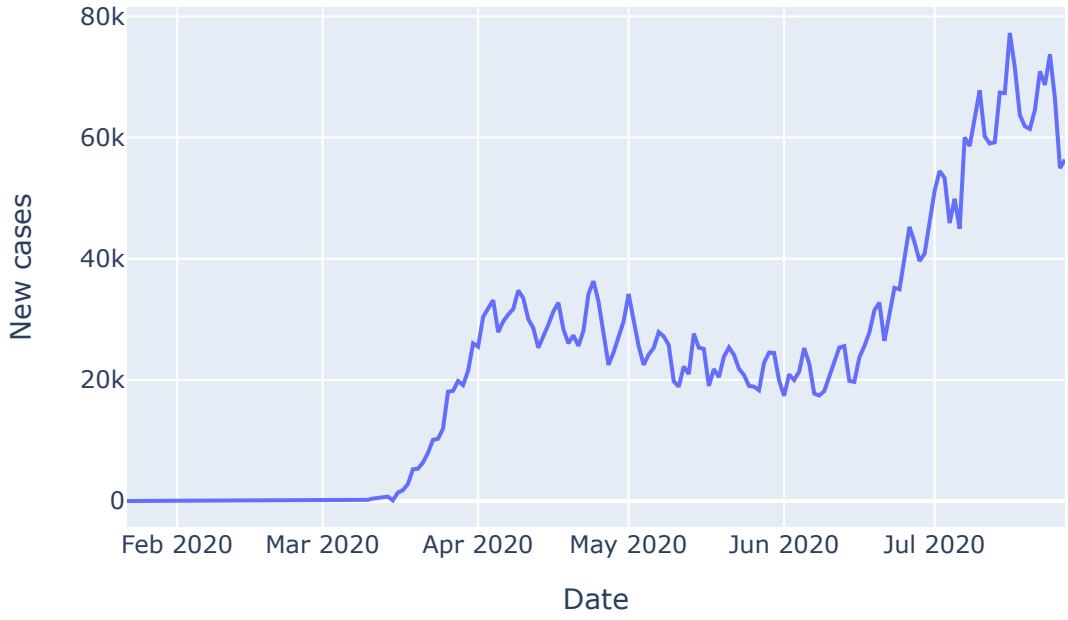



```
px.line(df_US,x="Date", y="Confirmed", height=400)
```

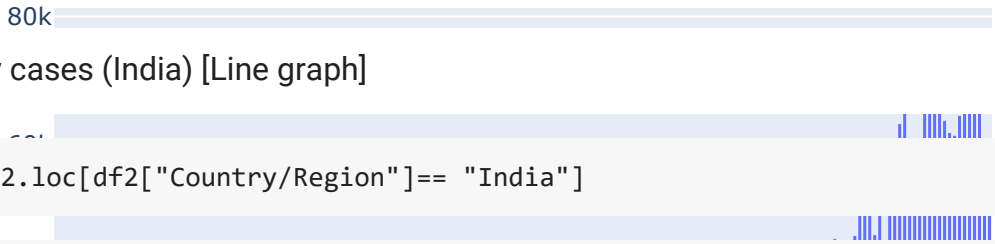


Date vs New Cases (US)[Line graph]

```
px.line(df_US,x="Date", y="New cases", height=400)
```



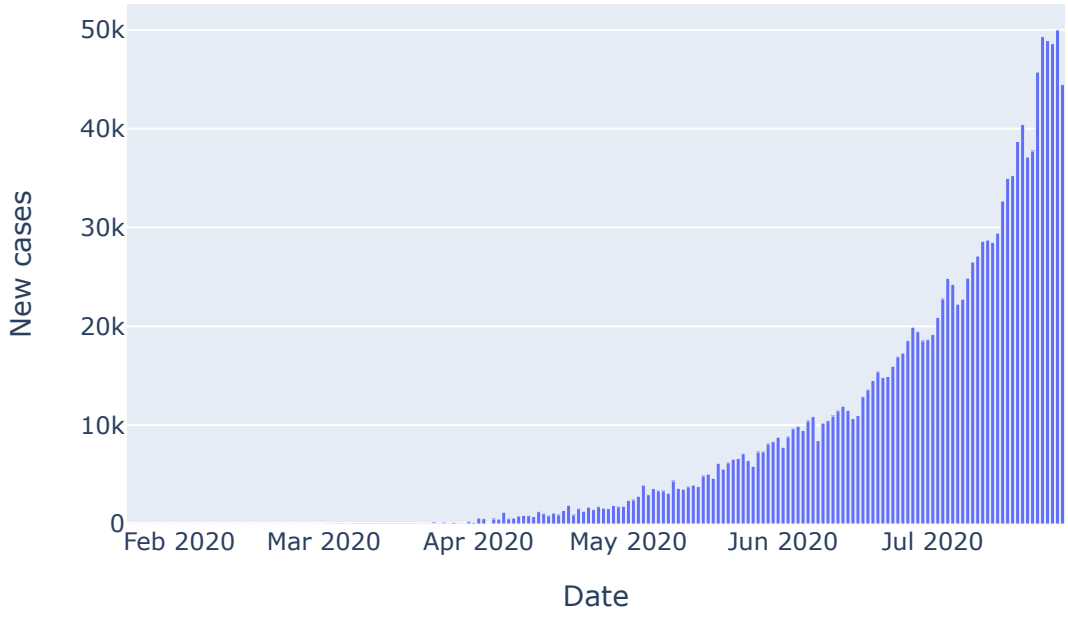

```
px.bar(df_US,x="Date", y="New cases", height=400)
```



Date Vs New cases (India) [Line graph]

```
df_india= df2.loc[df2["Country/Region"]== "India"]
```

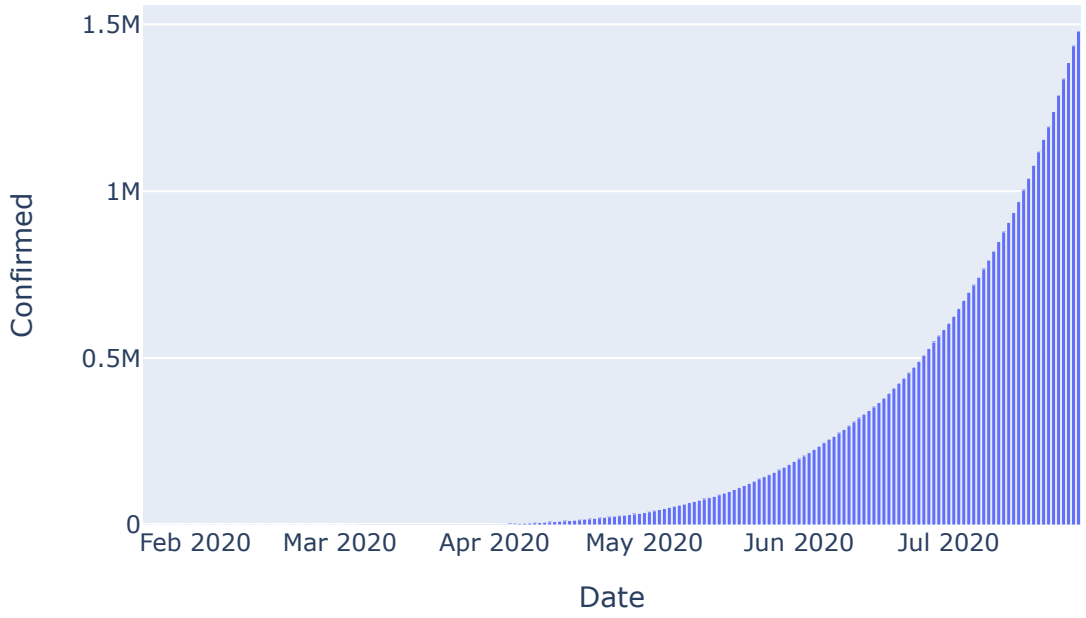
```
px.bar(df_india, x="Date", y="New cases", height=400)
```




```
px.line(df_india, x="Date", y="New cases", height=400)
```

Date Vs Confirmed (India)

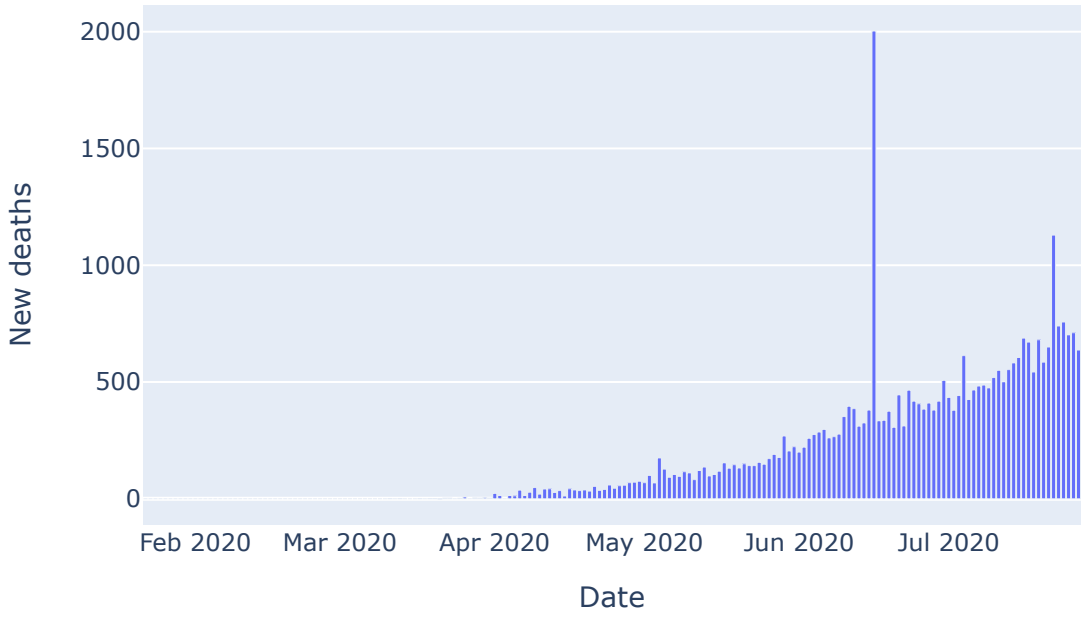
```
px.bar(df_india, x="Date", y="Confirmed", height=400)
```



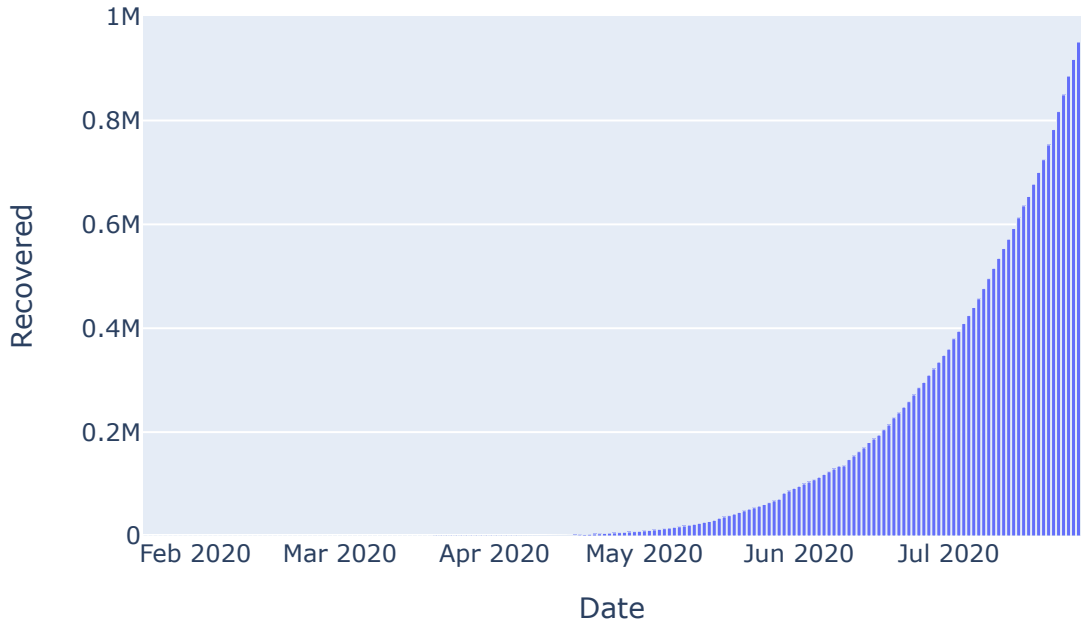
Date Vs New Death (India)

```
px.bar(df_india, x="Date", y="Deaths", height=400)
```

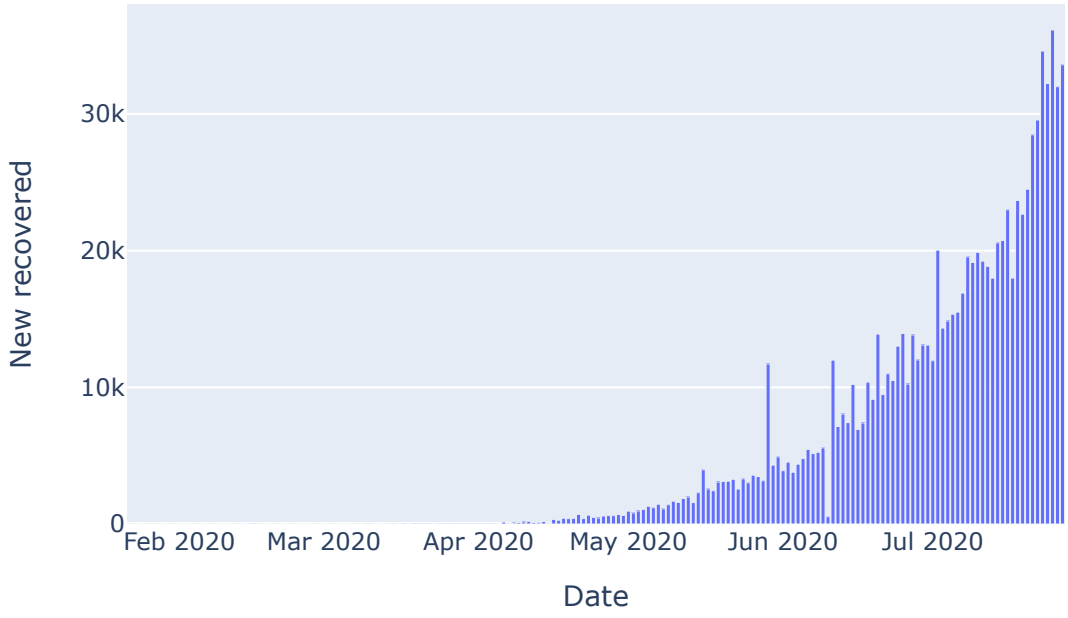
```
px.bar(df_india, x="Date", y="New deaths", height=400)
```



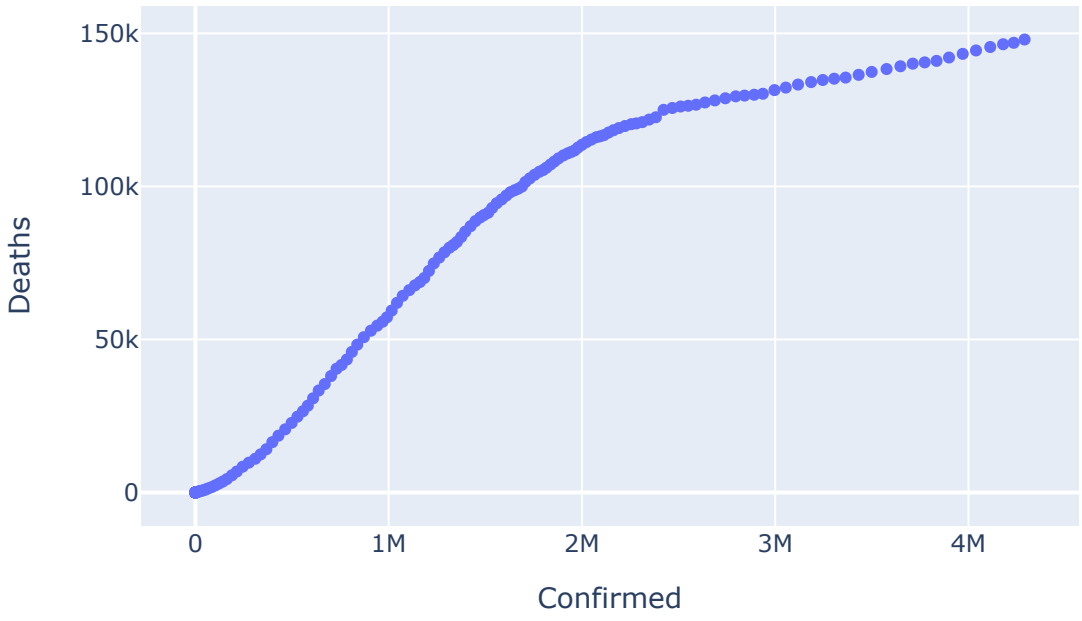

```
px.bar(df_india, x="Date", y="Recovered", height=400)
```



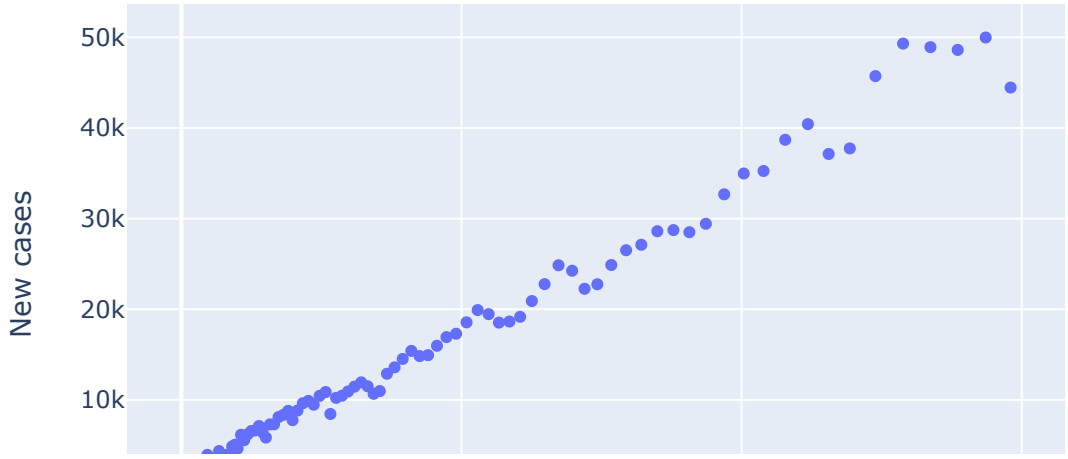

```
px.bar(df_india, x="Date", y="New recovered", height=400)
```




```
px.scatter(df_US, x="Confirmed", y="Deaths", height=400)
```




```
px.scatter(df_india, x="Confirmed", y="New cases", height=400)
```

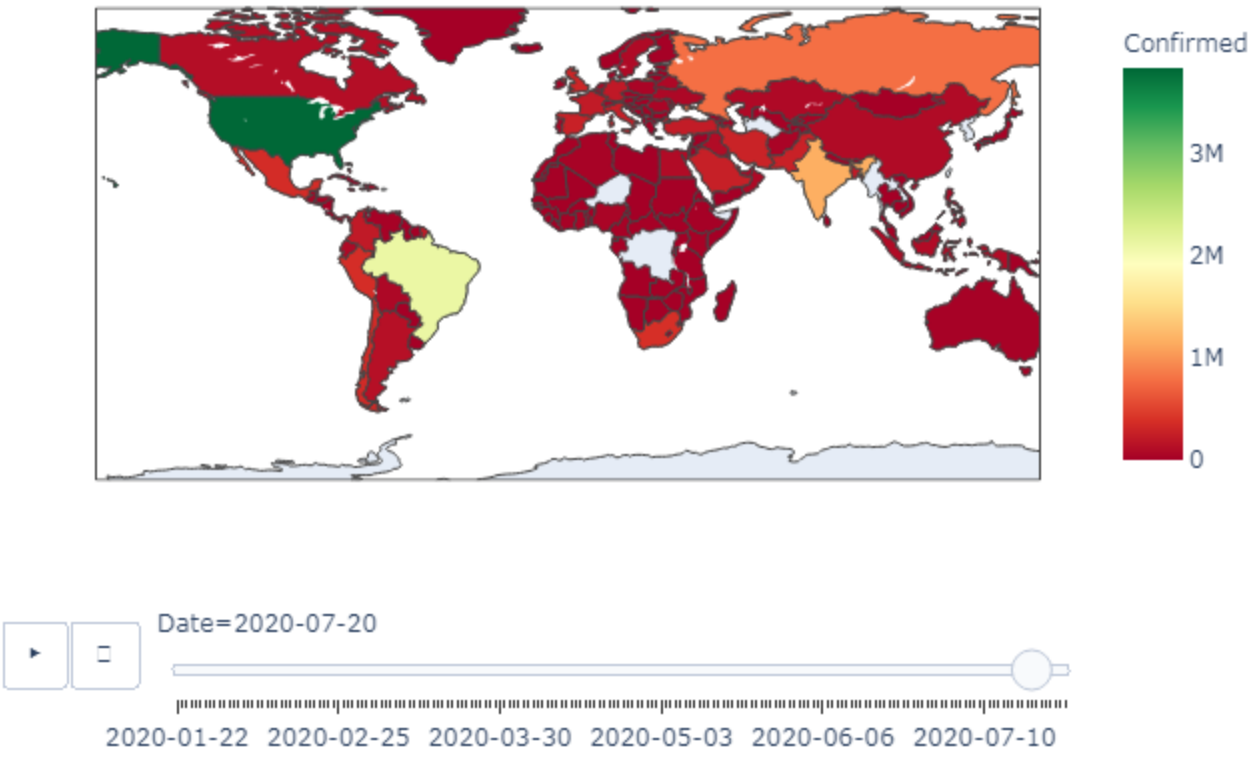


Task 6: Represent Geographic Data as Choropleth Maps

Confirmed

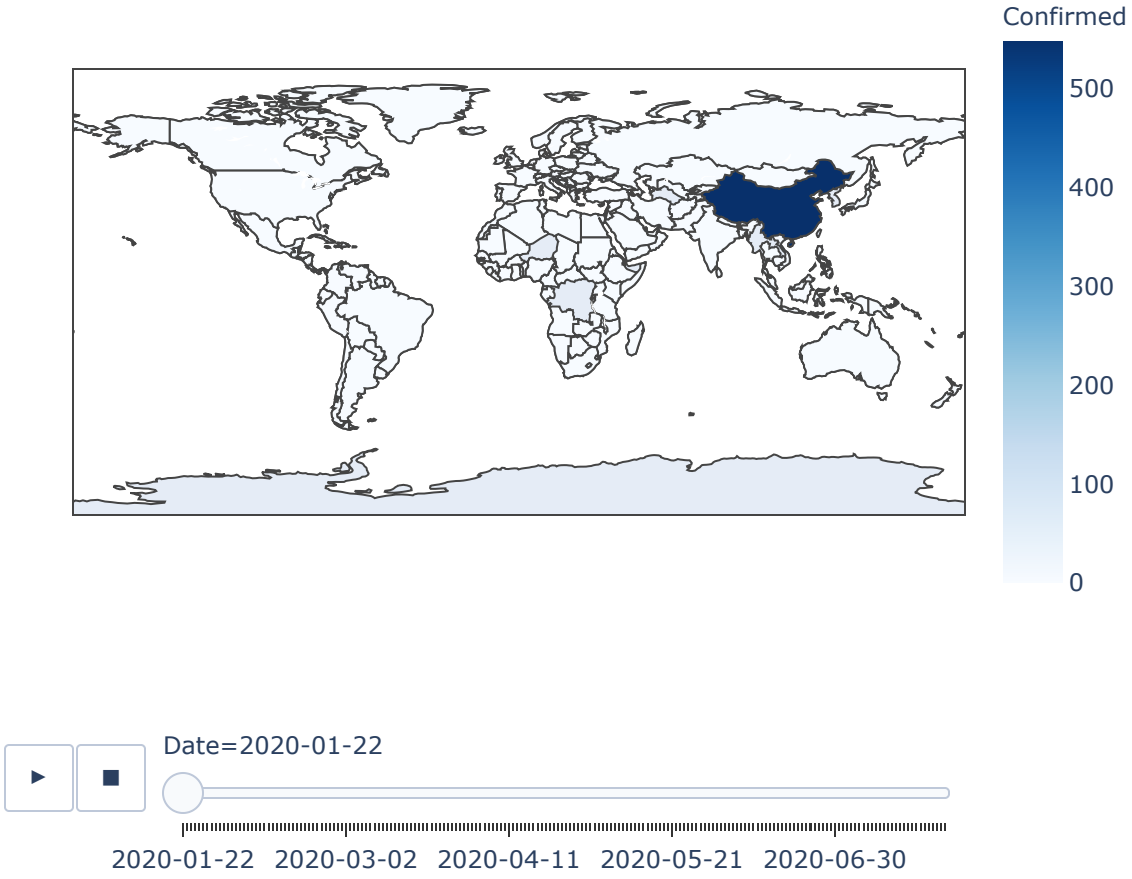
```
# A choropleth map displays divided geographical areas or regions that are coloured, shaded or patterne
#Dataset 2
#parameters= dataset, locations= ISOALPHA, color, hover_name, color_continuous_scale= [RdYlGn, Blues, V

#Amazing Representation of data in a map . Choropleth maps provide an easy way to visualize how a measu
```

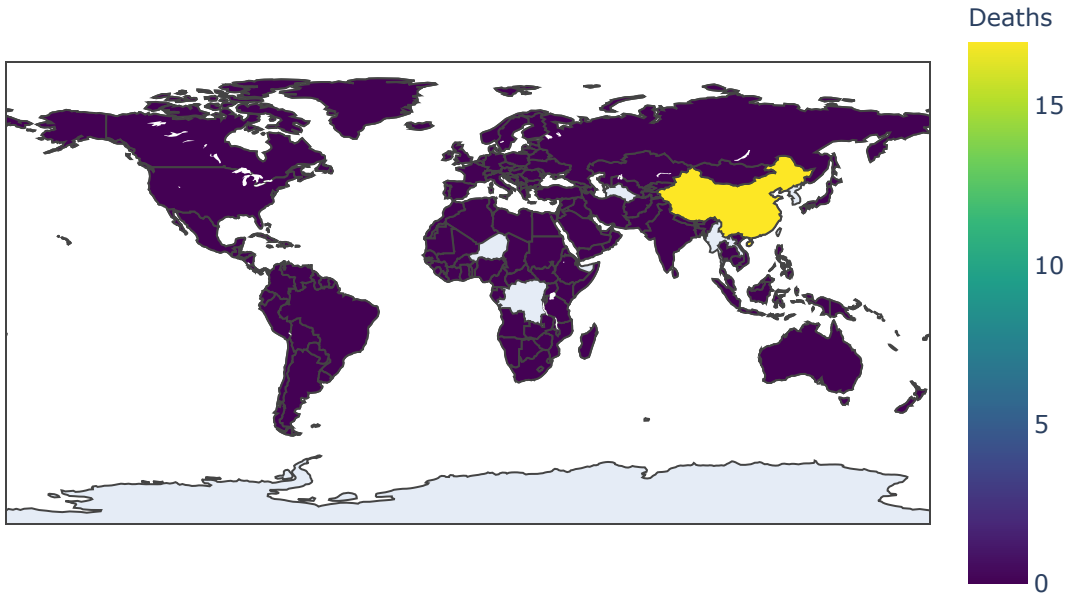


Equi-rectangular Projection: Total cases

```
px.choropleth(df2,
              locations="iso_alpha",
              color="Confirmed",
              hover_name="Country/Region",
              color_continuous_scale="Blues",
              animation_frame="Date")
```




```
px.choropleth(df2,
              locations='iso_alpha',
              color="Deaths",
              hover_name="Country/Region",
              color_continuous_scale="Viridis",
              animation_frame="Date" )
```

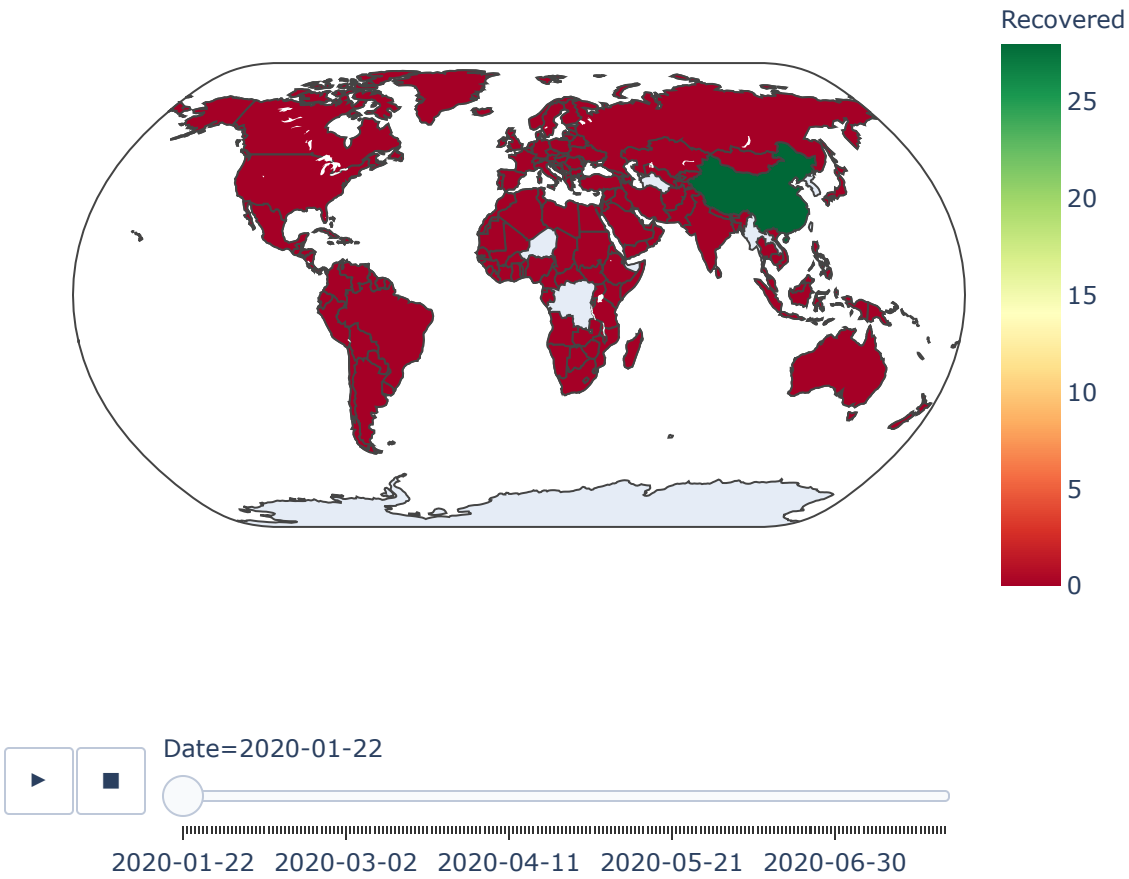


Orthographic Projection : Total Death

```
px.choropleth(df2,
              locations='iso_alpha',
              color="Deaths",
              hover_name="Country/Region",
              color_continuous_scale="Viridis",
              projection="orthographic",
              animation_frame="Date" )
```

Equirectangular Projection : Total Recovered

```
px.choropleth(df2,
              locations='iso_alpha',
              color="Recovered",
              hover_name="Country/Region",
              color_continuous_scale="RdYlGn",
              projection="natural earth",
              animation_frame="Date" )
```

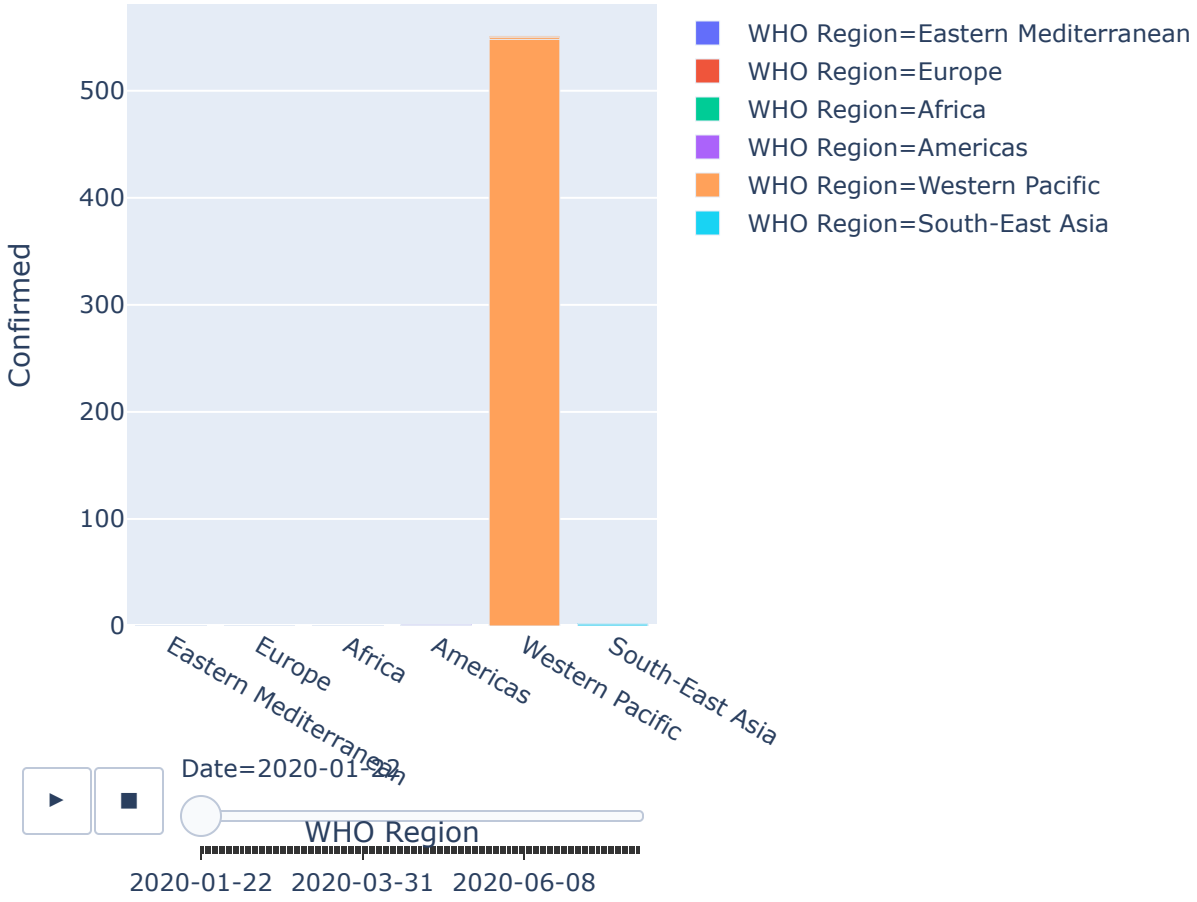


Task 7: Animations

```
#px.bar (data, x,y, color, hover_name, animation_frame)
#px.scatter (data, x,y, color, hover_name, animation_frame, size, size_max)
```

Bar Animation : Total Cases

```
px.bar(df2, x="WHO Region", y="Confirmed", color="WHO Region", animation_frame="Date", hover_name="Coun
```



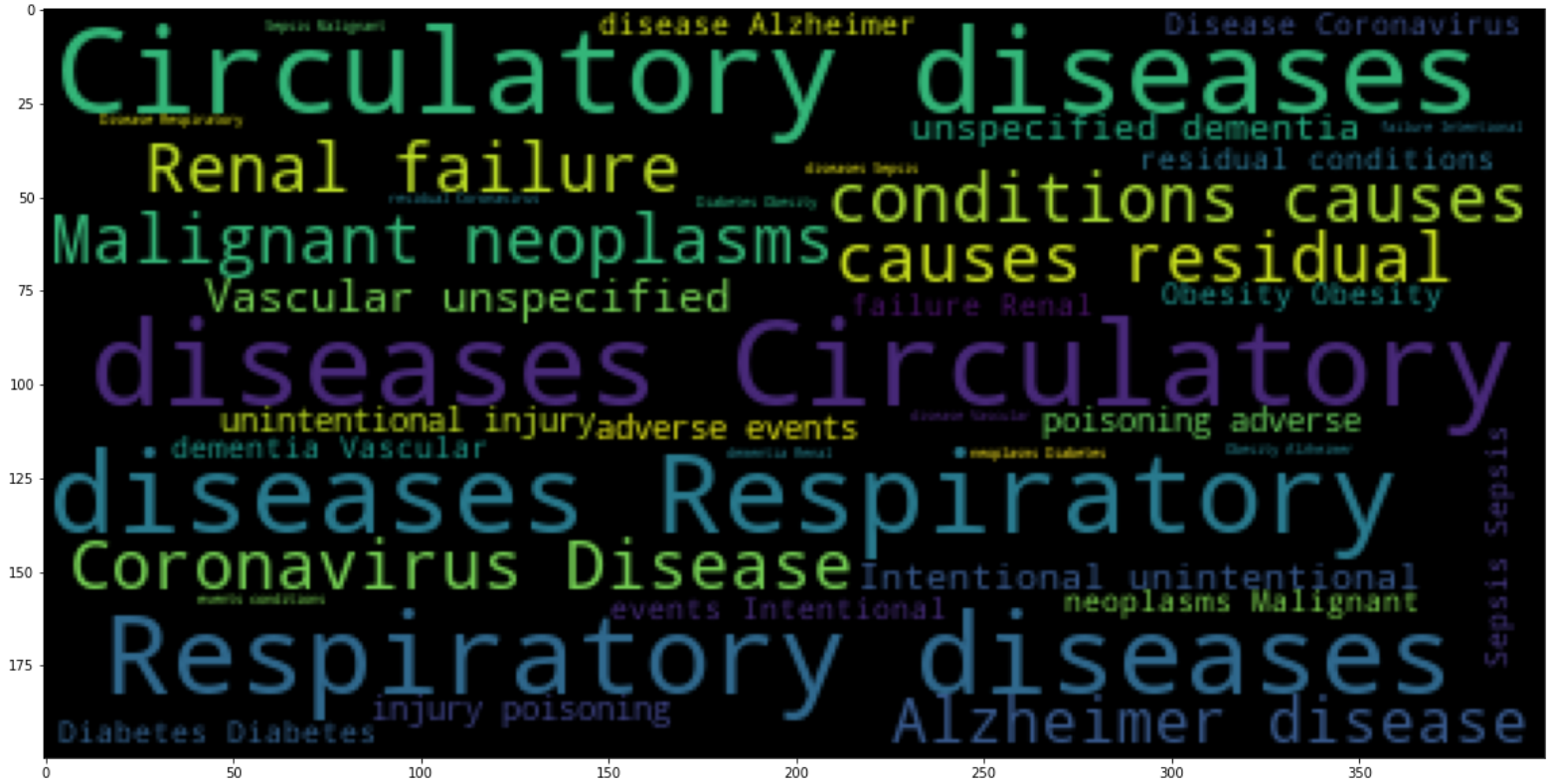
Bar Animation : New cases

```
px.bar(df2, x="WHO Region", y="New cases", color="WHO Region", animation_frame="Date", hover_name="Coun
```

Bonus Lecture : WordCloud (Reasons of Death) (New dataset-3)

```
#Step 1. Importing WordCloud and datasets
#Step 2. Exploring data  using pandas                          #NEW DATASET 3
#Step 3. Creating WordCloud
```

```
#Step3a= Convert the column with diseases count into list using tolist() function
#Step3b= Convert the list to one single string
#Step3x= Convert the string into WordCloud
```




```
from wordcloud import WordCloud
```

```
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving covid death.csv to covid death.csv
{'covid death.csv': b'Data as of,Start Week,End Week,State,Condition Group,Conditi
```

```
df3= pd.read_csv("covid death.csv")
df3.head()
```

	Data as of	Start Week	End Week	State	Condition Group	Condition	ICD10_codes	A Gro
0	08/30/2020	02/01/2020	08/29/2020	US	Respiratory diseases	Influenza and pneumonia	J09-J18	0-
1	08/30/2020	02/01/2020	08/29/2020	US	Respiratory	Influenza and	J09-J18	25

```
df3.tail()
```

	Data as of	Start Week	End Week	State	Condition Group	Condition	ICD10_codes
12255	08/30/2020	02/01/2020	08/29/2020	YC	Coronavirus Disease 2019	COVID-19	U071
12256	08/30/2020	02/01/2020	08/29/2020	YC	Coronavirus Disease 2019	COVID-19	U071
12257	08/30/2020	02/01/2020	08/29/2020	YC	Coronavirus Disease 2019	COVID-19	U071

```
df3.groupby(["Condition"]).count()
```

Condition	Data as of	Start Week	End Week	State	Condition Group	ICD10_codes	Age Group	Number of COVID-19 Deaths	F
Adult respiratory distress syndrome	540	540	540	540	540	540	540	272	
All other conditions and causes (residual)	540	540	540	540	540	540	540	363	
Alzheimer disease	530	530	530	530	530	530	530	144	
COVID-19	540	540	540	540	540	540	540	377	
Cardiac arrest	520	520	520	520	520	520	520	219	
Cardiac arrhythmia	540	540	540	540	540	540	540	192	
Cerebrovascular diseases	530	530	530	530	530	530	530	187	
Chronic lower respiratory diseases	540	540	540	540	540	540	540	229	
Diabetes	540	540	540	540	540	540	540	276	
Heart failure	540	540	540	540	540	540	540	204	
Hypertensive ..	540	540	540	540	540	540	540	264	

```
df3.groupby(["Condition Group"]).count()
```

	Data as of	Start Week	End Week	State	Condition	ICD10_codes	Age Group	Number of COVID-19 Deaths	Fla
Condition Group									

```
#Step 3 : WordCloud 1

#Step3a= Convert the column with diseases count into list using tolist() function
```

```
sentences= df3["Condition"].tolist()
```

```
#Step3b= Convert the list to one single string
```

```
sentences_as_a_string= ' '.join(sentences)
```

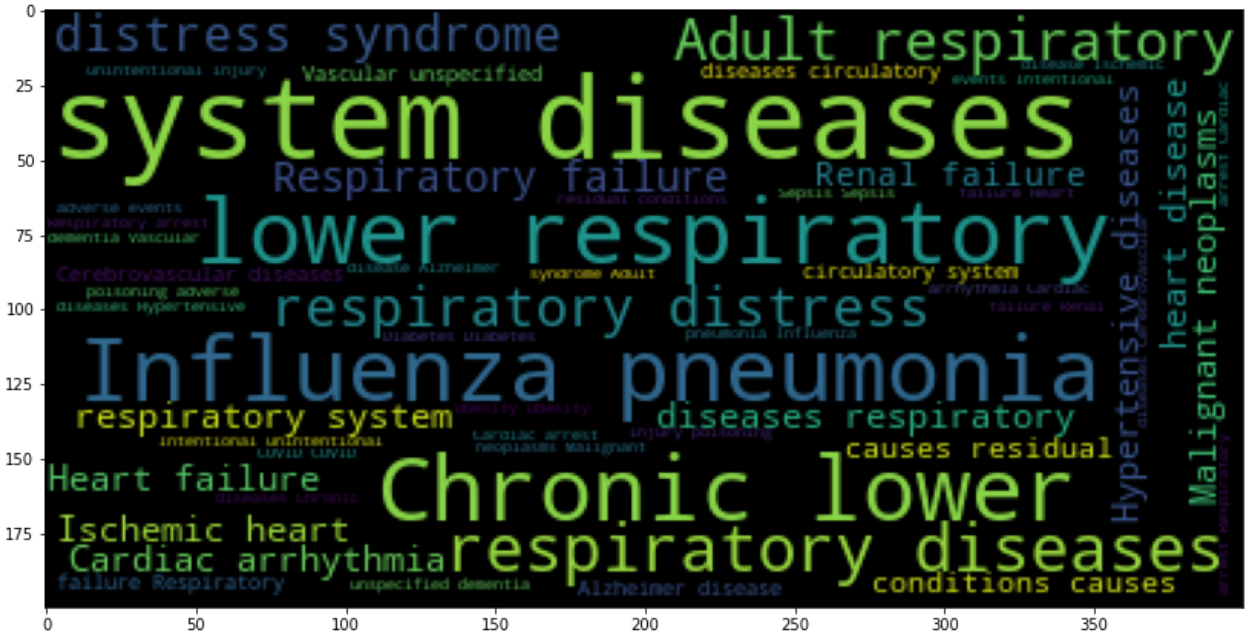
Diabetes	540	540	540	540	540	540	540	276	26

```
#Step3c= Convert the string into WordCloud
```

and

```
plt.figure(figsize=(14,14))
plt.imshow(WordCloud().generate(sentences_as_a_string))
```

```
<matplotlib.image.AxesImage at 0x7fdd10c99be0>
```




```
#Step 3 : WordCloud 2- "Condition Group" column
```

```
#Step3a= Convert the column with diseases count into list using tolist() function

column2_tolist= df3["Condition Group"].tolist()
```

```
#Step3b= Convert the list to one single string
```

```
column_to_string= " ".join(column2_tolist)
```

```
#Step3c= Convert the string into WordCloud
```

```
plt.figure(figsize=(20,20))
plt.imshow(WordCloud().generate(column_to_string))
```

```
<matplotlib.image.AxesImage at 0x7fdd10ba3c50>
```