Covid-19 Analysis and Visualization

This analysis summarizes the modeling, simulation, and analytics work around the COVID-19 outbreak around the world from the perspective of data science and visual analytics. It examines the impact of best practices and preventive measures in various sectors and enables outbreaks to be managed with available health resources.

Step 1: Importing Necessary Libraries

```
# Data analysis and Manipulation
import plotly.graph_objs as go
import plotly.io as pio
import plotly.express as px
import pandas as pd

# Data Visualization
import matplotlib.pyplot as plt

# Importing Plotly
import plotly.offline as py
py.init_notebook_mode(connected=True)

# Initializing Plotly
pio.renderers.default = 'colab'
```

Step 2: Importing the Datasets

Importing three datasets into this project

covid– This dataset contains 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.

covid_grouped– This dataset contains Date(from 20-01-22 to 20-07-27), Country/Region, Confirmed, Deaths, Recovered, Active, New cases, New deaths, New recovered, WHO Region, iso_alpha.

coviddeath– This dataset contains real-world examples of a number of Covid-19 deaths and the reasons behind the deaths.

```
# Importing Dataset1
dataset1 = pd.read_csv("covid.csv")
dataset1.head()  # returns first 5 rows
```

	Country/Region	Continent	Population	TotalCases	NewCases	TotalDeaths	NewDeaths	TotalRecovered	NewRecovered	ActiveCa
0	USA	North America	3.311981e+08	5032179	NaN	162804.0	NaN	2576668.0	NaN	22927
1	Brazil	South America	2.127107e+08	2917562	NaN	98644.0	NaN	2047660.0	NaN	7712
2	India	Asia	1.381345e+09	2025409	NaN	41638.0	NaN	1377384.0	NaN	6063
3	Russia	Europe	1.459409e+08	871894	NaN	14606.0	NaN	676357.0	NaN	1809
4	South Africa	Africa	5.938157e+07	538184	NaN	9604.0	NaN	387316.0	NaN	1412

Getting dataset information

```
# Returns tuple of shape (Rows, columns)
print(dataset1.shape)

# Returns size of dataframe
print(dataset1.size)

# Information about Dataset1
# return concise summary of dataframe
dataset1.info()
```

```
(209, 17)
3553
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 209 entries, 0 to 208
Data columns (total 17 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Country/Region    209 non-null    object
 1   Continent         208 non-null    object
 2   Population        208 non-null    float64
 3   TotalCases        209 non-null    int64
 4   NewCases          4 non-null      float64
 5   TotalDeaths       188 non-null    float64
 6   NewDeaths         3 non-null      float64
 7   TotalRecovered    205 non-null    float64
 8   NewRecovered      3 non-null      float64
 9   ActiveCases       205 non-null    float64
 10  Serious,Critical  122 non-null    float64
 11  Tot Cases/1M pop  208 non-null    float64
 12  Deaths/1M pop     187 non-null    float64
 13  TotalTests        191 non-null    float64
 14  Tests/1M pop      191 non-null    float64
 15  WHO Region        184 non-null    object
 16  iso_alpha         209 non-null    object
dtypes: float64(12), int64(1), object(4)
memory usage: 27.9+ KB
```

Importing dataset number 2

```
# Importing Dataset2
dataset2 = pd.read_csv("covid_grouped.csv")
dataset2.head()  # return first 5 rows of dataset2
```

	Date	Country/Region	Confirmed	Deaths	Recovered	Active	New cases	New deaths	New recovered	WHO Region	iso_alpha
0	2020-01-22	Afghanistan	0	0	0	0	0	0	0	Eastern Mediterranean	AFG
1	2020-01-22	Albania	0	0	0	0	0	0	0	Europe	ALB
2	2020-01-	Algeria	0	0	0	0	0	0	0	Africa	DZA

Getting dataset information

```
# Returns tuple of shape (Rows, columns)
print(dataset2.shape)

# Returns size of dataframe
print(dataset2.size)

# Information about Dataset2
dataset2.info()  # return concise summary of dataframe
```

```
(35156, 11)
386716
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35156 entries, 0 to 35155
Data columns (total 11 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Date            35156 non-null  object
 1   Country/Region  35156 non-null  object
 2   Confirmed       35156 non-null  int64
 3   Deaths          35156 non-null  int64
 4   Recovered       35156 non-null  int64
```

```
 5   Active          35156 non-null  int64
 6   New cases       35156 non-null  int64
 7   New deaths      35156 non-null  int64
 8   New recovered   35156 non-null  int64
 9   WHO Region      35156 non-null  object
 10  iso_alpha       35156 non-null  object
dtypes: int64(7), object(4)
memory usage: 3.0+ MB
```

Step 3: Dataset cleaning

```
# Columns labels of a Dataset1
dataset1.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')
```

We don't need 'NewCases', 'NewDeaths', 'NewRecovered' columns as they contains NaN values. So drop these columns by drop() function of pandas.

```
# Drop NewCases, NewDeaths, NewRecovered rows from dataset1

dataset1.drop(['NewCases', 'NewDeaths', 'NewRecovered'],
              axis=1, inplace=True)

# Select random set of values from dataset1
dataset1.sample(5)
```

	Country/Region	Continent	Population	TotalCases	TotalDeaths	TotalRecovered	ActiveCases	Serious,Critical	Tot Cases/1M pop	D
68	El Salvador	North America	6489514.0	19126	513.0	9236.0	9377.0	509.0	2947.0	
90	Zambia	Africa	18430129.0	7164	199.0	5786.0	1179.0	NaN	389.0	
196	Saint Lucia	North America	183712.0	25	NaN	24.0	1.0	NaN	136.0	
156	Diamond Princess	NaN	NaN	712	13.0	651.0	48.0	4.0	NaN	

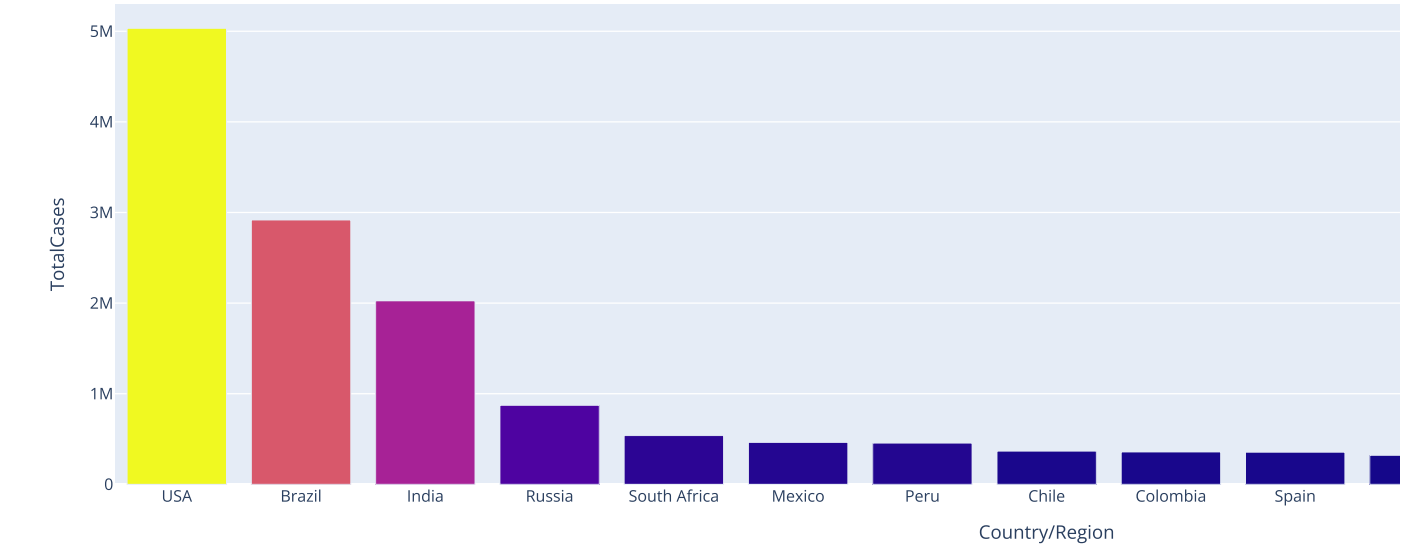
Creating table using plotly express

```
# Import create_table Figure Factory

from plotly.figure_factory import create_table

colorscale = [[0, '#4d004c'], [.5, '#f2e5ff'], [1, '#ffffff']]
table = create_table(dataset1.head(15), colorscale=colorscale)
py.iplot(table)
```

Country/Region	Continent	Population	TotalCases	TotalDeaths	TotalRecovered	ActiveCases	Serious,Critical	Tot Cases/1M pop
USA	North America	331198130.0	5032179	162804.0	2576668.0	2292707.0	18296.0	15194.0
Brazil	South America	212710692.0	2917562	98644.0	2047660.0	771258.0	8318.0	13716.0
India	Asia	1381344997.0	2025409	41638.0	1377384.0	606387.0	8944.0	1466.0
Russia	Europe	145940924.0	871894	14606.0	676357.0	180931.0	2300.0	5974.0
South Africa	Africa	59381566.0	538184	9604.0	387316.0	141264.0	539.0	9063.0
Mexico	North America	129066160.0	462690	50517.0	308848.0	103325.0	3987.0	3585.0
Peru	South America	33016319.0	455409	20424.0	310337.0	124648.0	1426.0	13793.0
Chile	South America	19132514.0	366671	9889.0	340168.0	16614.0	1358.0	19165.0
Colombia	South America	50936262.0	357710	11939.0	192355.0	153416.0	1493.0	7023.0
Spain	Europe	46756648.0	354530	28500.0	nan	nan	617.0	7582.0
Iran	Asia	84097623.0	320117	17976.0	277463.0	24678.0	4156.0	3806.0
UK	Europe	67922029.0	308134	46413.0	nan	nan	73.0	4537.0
Saudi Arabia	Asia	34865919.0	284226	3055.0	247089.0	34082.0	1915.0	8152.0
Pakistan	Asia	221295851.0	281863	6035.0	256058.0	19770.0	809.0	1274.0

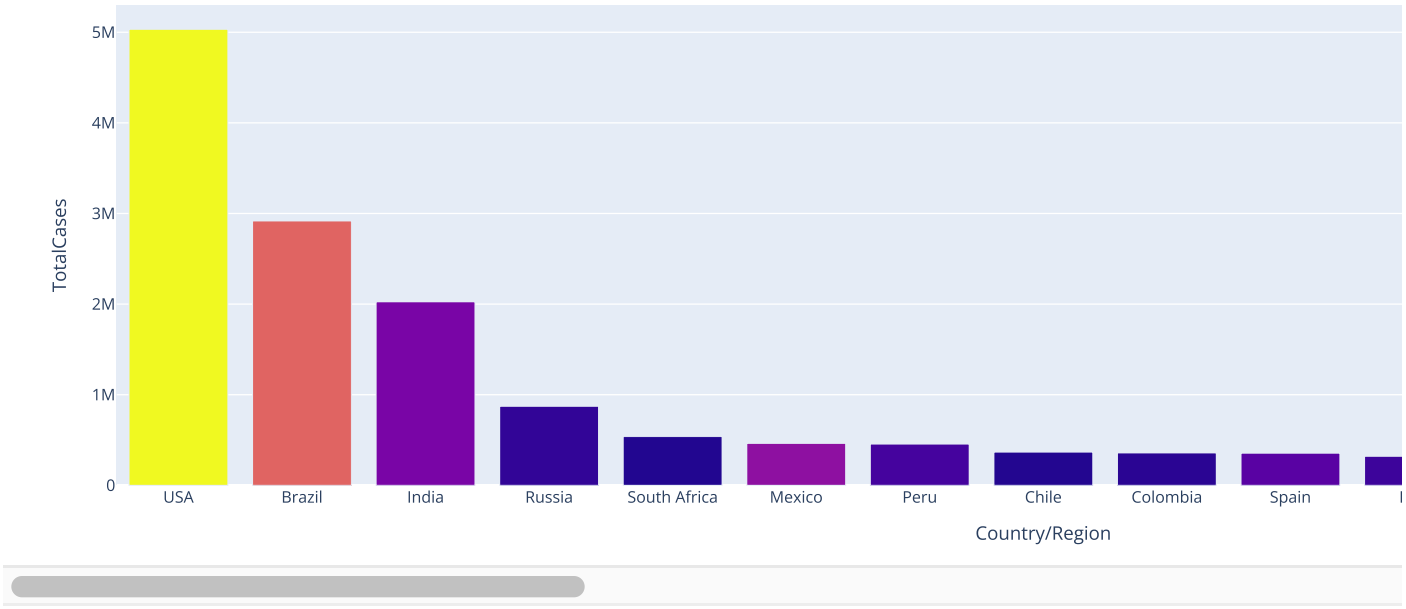
Step 4: Bar graphs- Comparisons between COVID infected countries in terms of total cases, total deaths, total recovered & total tests

```
px.bar(dataset1.head(15), x = 'Country/Region',
       y = 'TotalCases',color = 'TotalCases',
       height = 500,hover_data = ['Country/Region', 'Continent'])
```



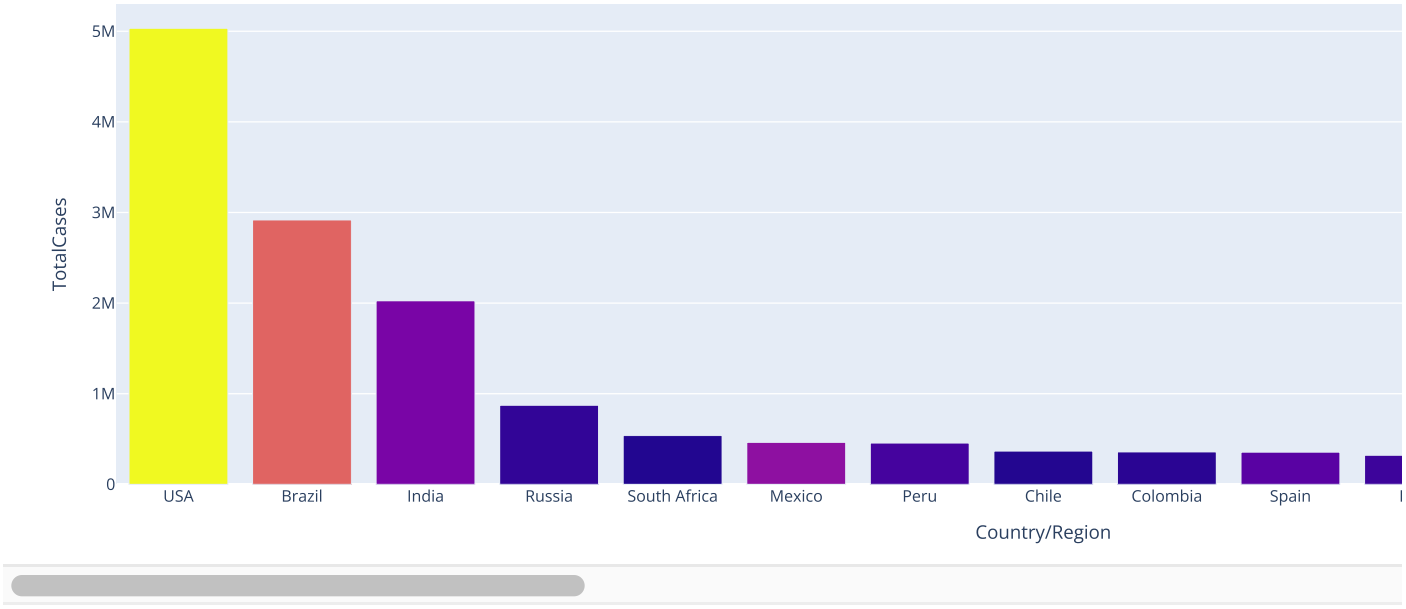
As the plot clearly shows the data for the top 15 countries, now again take the country with respect to the total number of cases from the top 15 countries, color the total deaths hover data as 'Country/Region', 'Continent' and analyze the visualization.

```
px.bar(dataset1.head(15), x = 'Country/Region', y = 'TotalCases',
       color = 'TotalDeaths', height = 500,
       hover_data = ['Country/Region', 'Continent'])
```



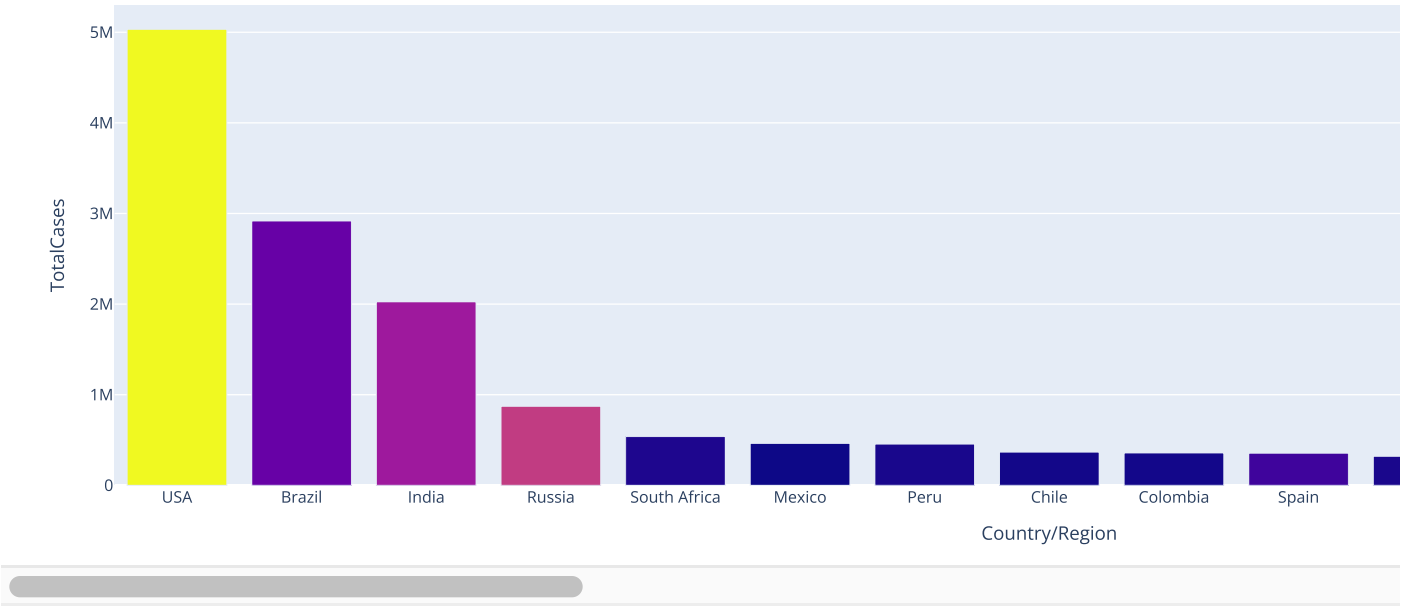
Let's analyze by coloring the total number of recovered cases

```
px.bar(dataset1.head(15), x = 'Country/Region', y = 'TotalCases',
       color = 'TotalDeaths', height = 500,
       hover_data = ['Country/Region', 'Continent'])
```



Visualize the same again by coloring the total number of tests.

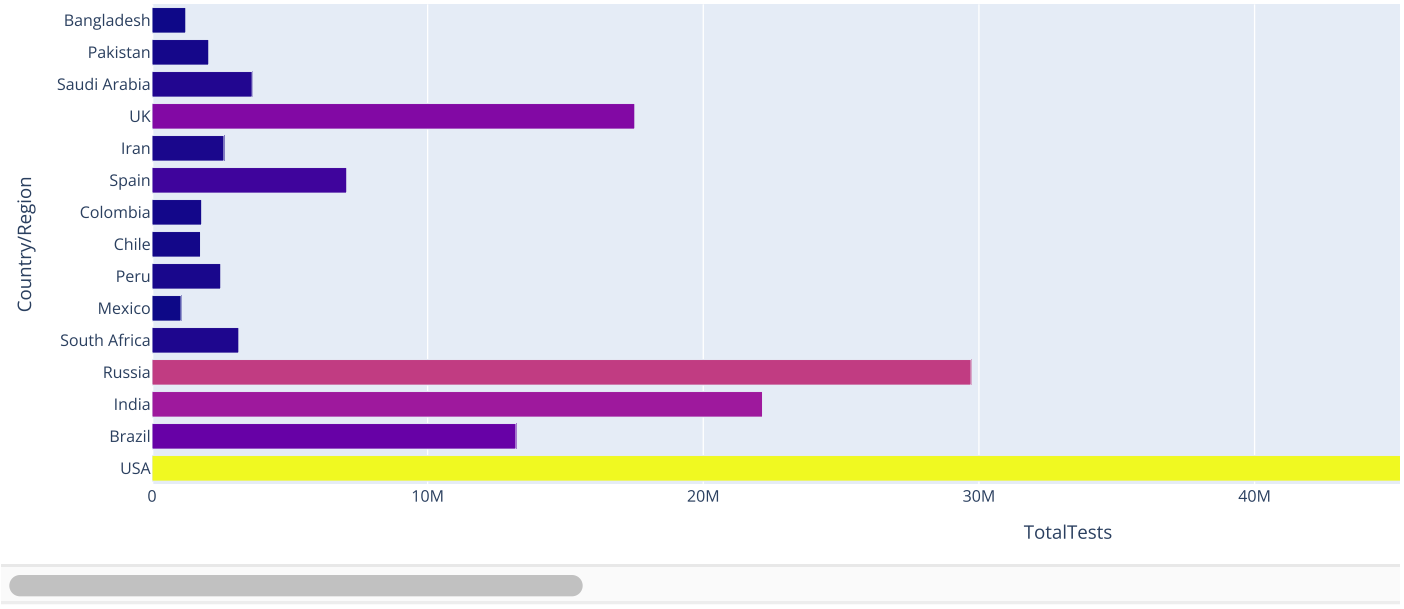
```
px.bar(dataset1.head(15), x = 'Country/Region', y = 'TotalCases',
       color = 'TotalTests', height = 500, hover_data = ['Country/Region', 'Continent'])
```



The visualization could be as we have done with the top 15 countries with total cases, deaths, recoveries, and tests. We can analyze the plot by looking at them.

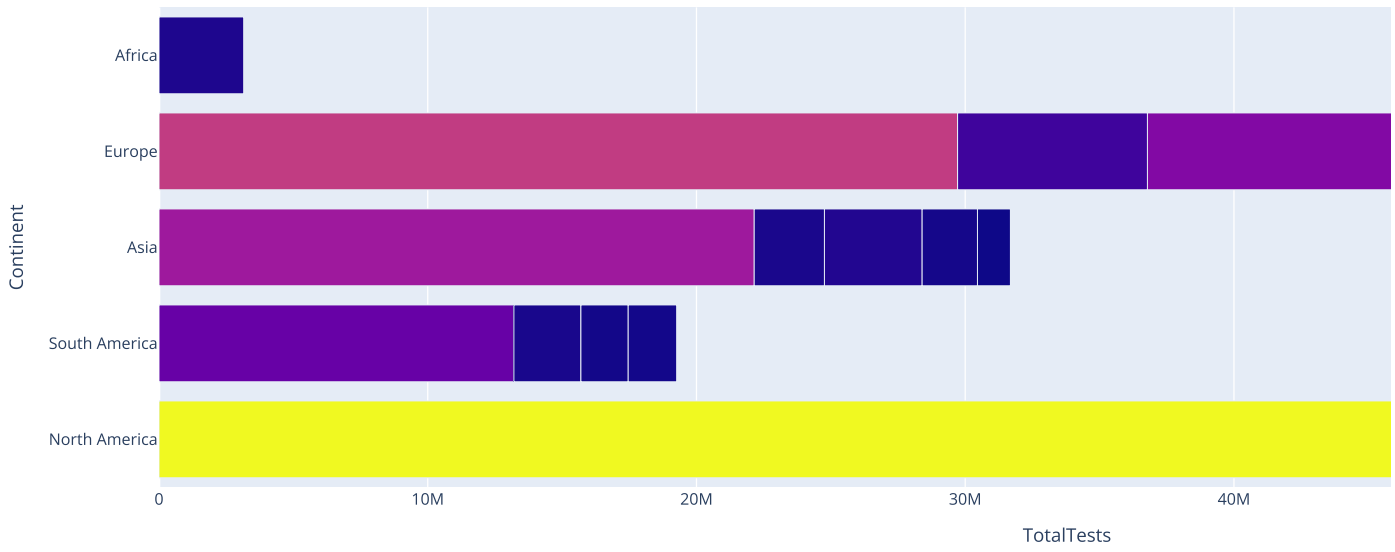
Let's create a horizontal orientation plot with X-axis as 'TotalTests' and Y-axis as 'Country/Region' with passing parameter orientation="h" and color the plot by 'TotalTests'.

```
px.bar(dataset1.head(15), x = 'TotalTests', y = 'Country/Region',
       color = 'TotalTests',orientation ='h',  height = 500,
       hover_data = ['Country/Region', 'Continent'])
```



Let's look at 'TotalTests' followed by 'Continent' and color the plot with 'Continent'.

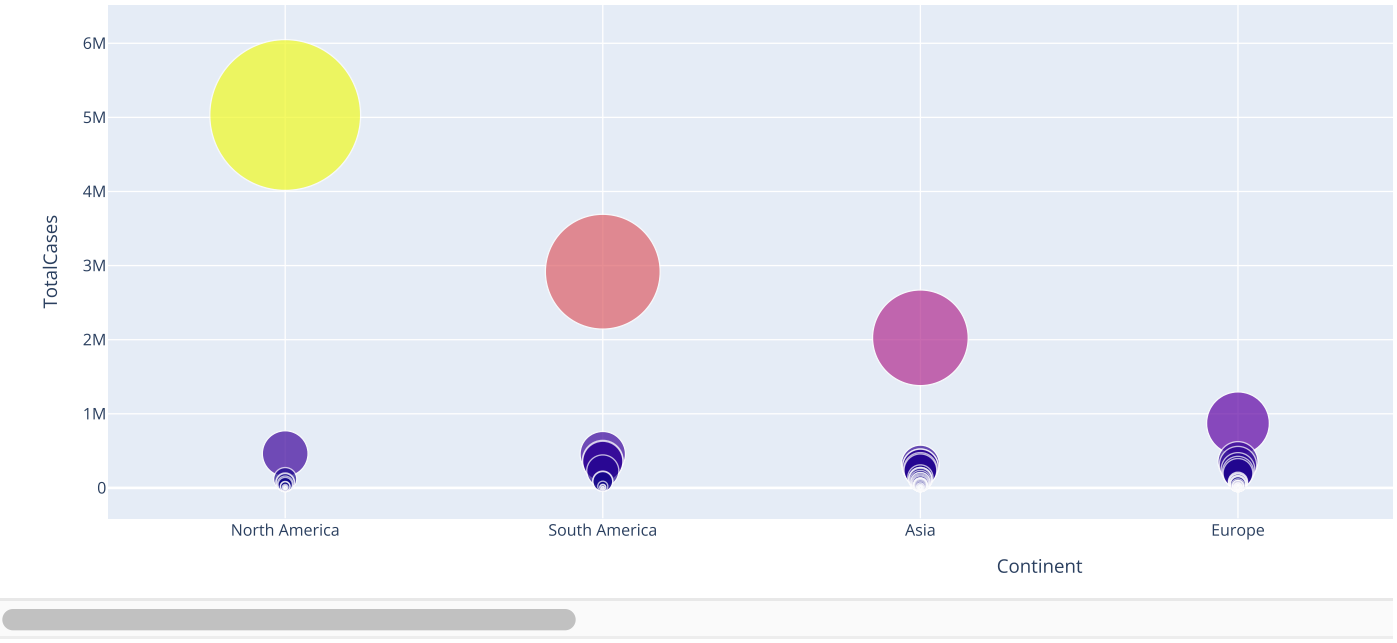
```
px.bar(dataset1.head(15), x = 'TotalTests', y = 'Continent',
       color = 'TotalTests',orientation ='h',  height = 500,
       hover_data = ['Country/Region', 'Continent'])
```



Step 5: Data Visualization through Bubble Charts-Continent Wise

Let's create a scatter plot and take a look at the continent's statistics, firstly look at the total number of cases by continent and take hover data as 'Country/Region', 'Continent'.

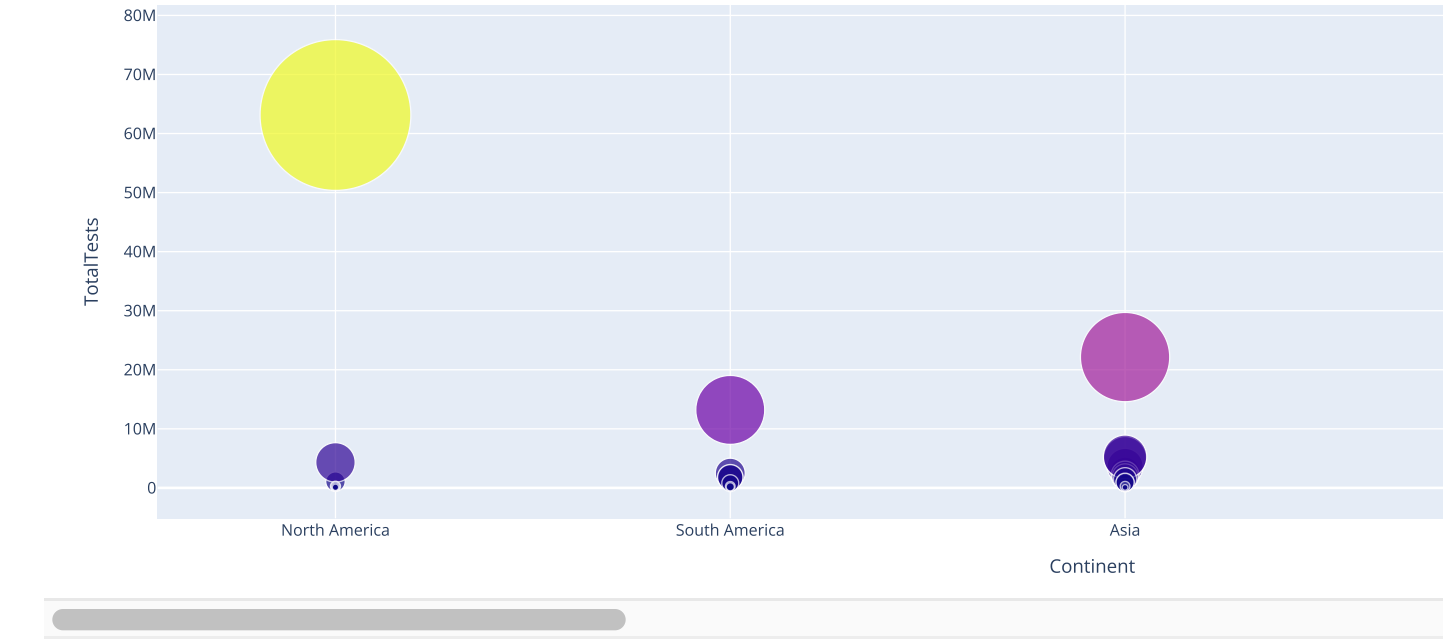
```
px.scatter(dataset1, x='Continent',y='TotalCases',
           hover_data=['Country/Region', 'Continent'],
           color='TotalCases', size='TotalCases', size_max=80)
```



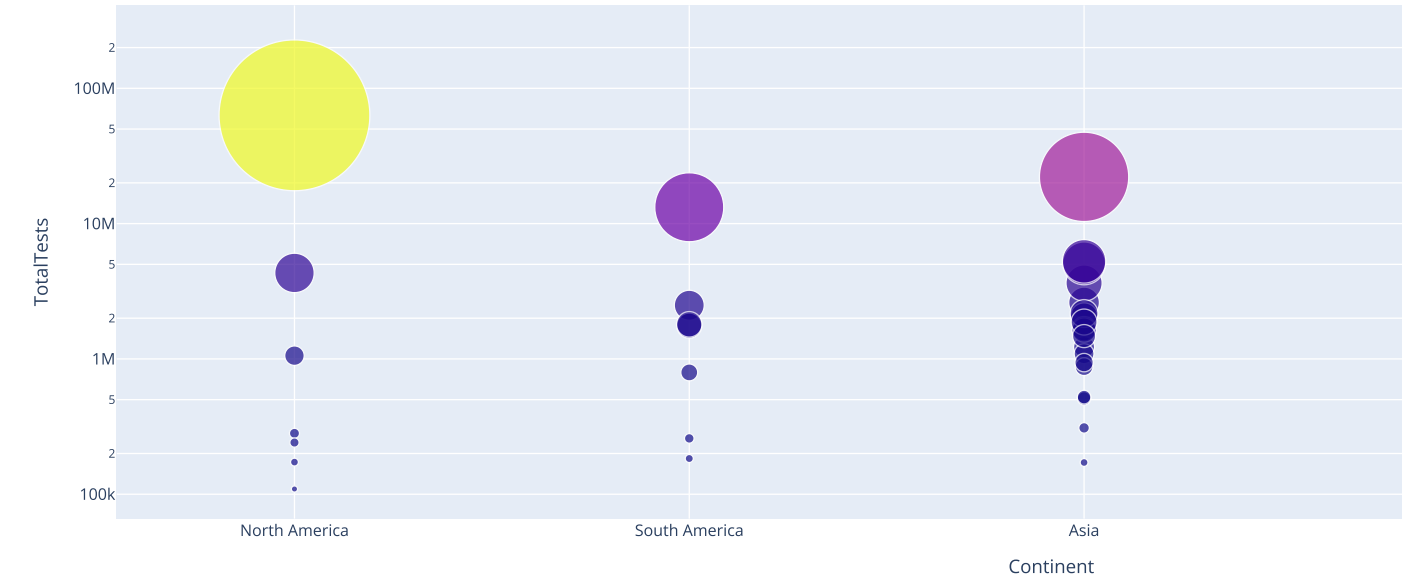
log_y= True, the histogram axis (not the returned parameter) is in log scale. The return parameter (n, bins), i.e. the values of bins and sides of bins are the same for log=True and log=False. This means both n==n2 and bins==bins2 are true

```
px.scatter(dataset1.head(57), x='Continent',y='TotalCases',
           hover_data=['Country/Region', 'Continent'],
           color='TotalCases', size='TotalCases', size_max=80, log_y=True)
```


```
px.scatter(dataset1.head(54), x='Continent',y='TotalTests',
           hover_data=['Country/Region', 'Continent'],
           color='TotalTests', size='TotalTests', size_max=80)
```



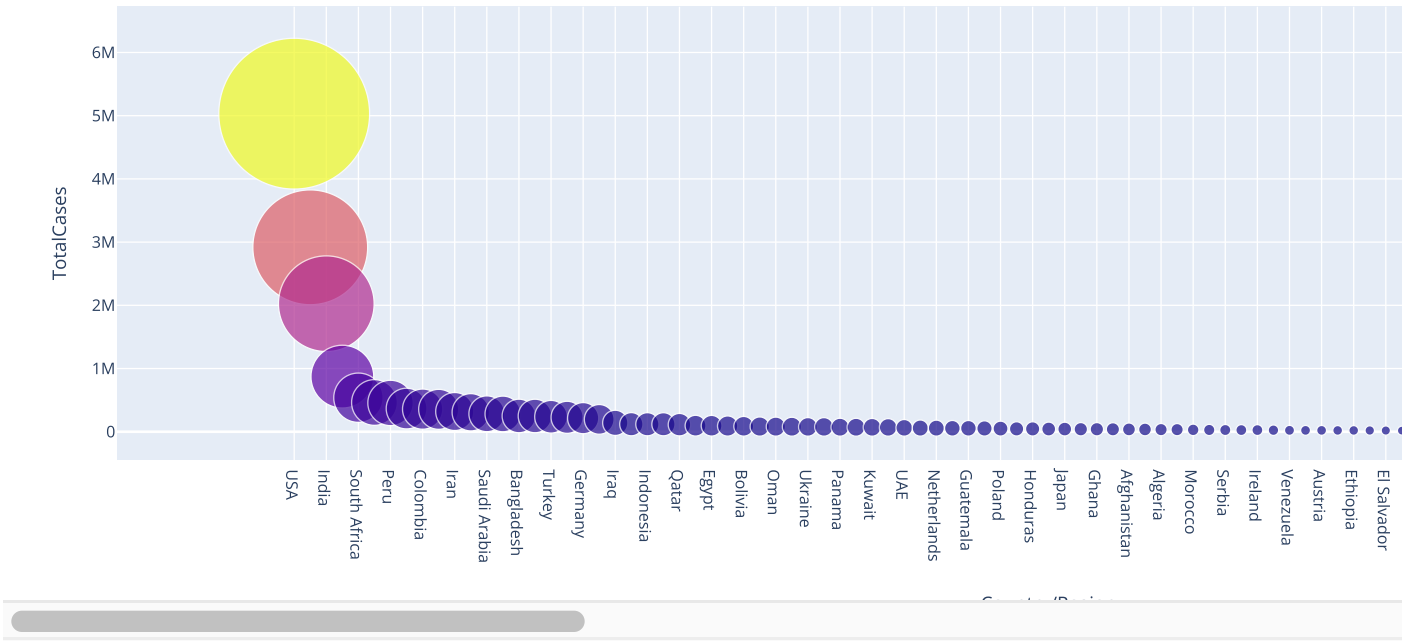

```
px.scatter(dataset1.head(50), x='Continent',y='TotalTests',
           hover_data=['Country/Region', 'Continent'],
           color='TotalTests', size='TotalTests', size_max=80, log_y=True)
```



Step 6: Data Visualization through Bubble Charts-Country Wise Let's take a look at the country-wise data visualization, first look at the continent with respect to the total number of deaths by the top 50 countries only and color the total number of deaths and take the hover data as 'Country/Region', 'Continent'.

Example: Bubble chart

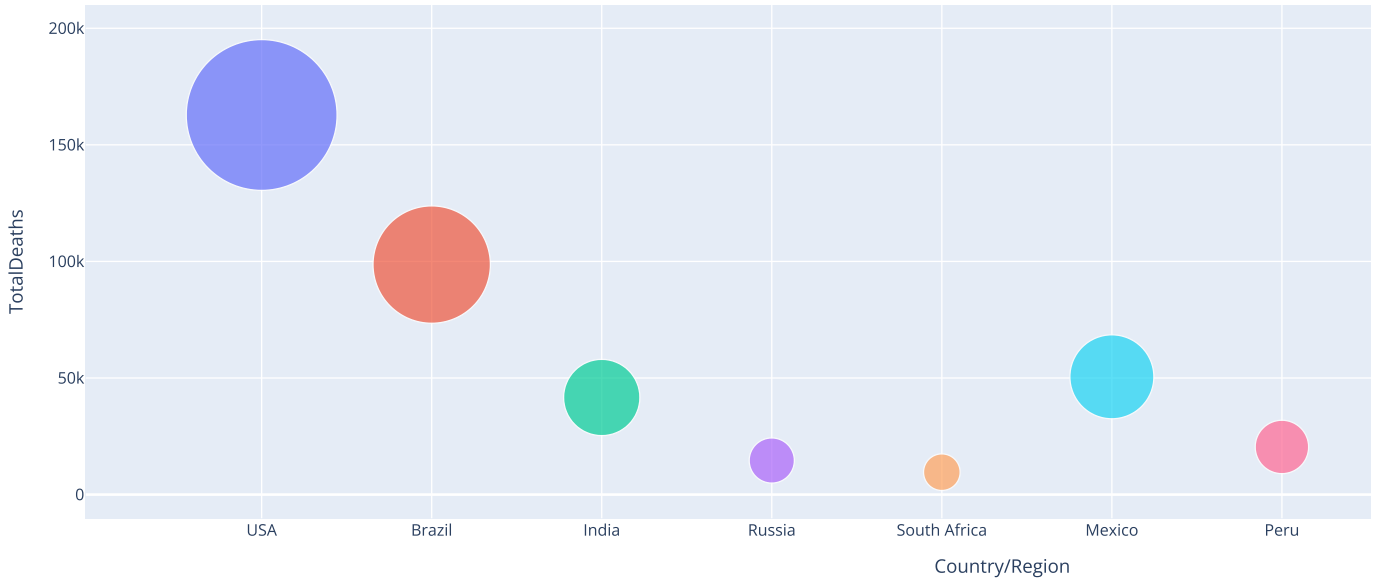
```
px.scatter(dataset1.head(100), x='Country/Region', y='TotalCases',
           hover_data=['Country/Region', 'Continent'],
           color='TotalCases', size='TotalCases', size_max=80)
```



Now format the image of the country/region in relation to the total number of deaths. And do the same for the other aspects of COVID-19 from dataset1.

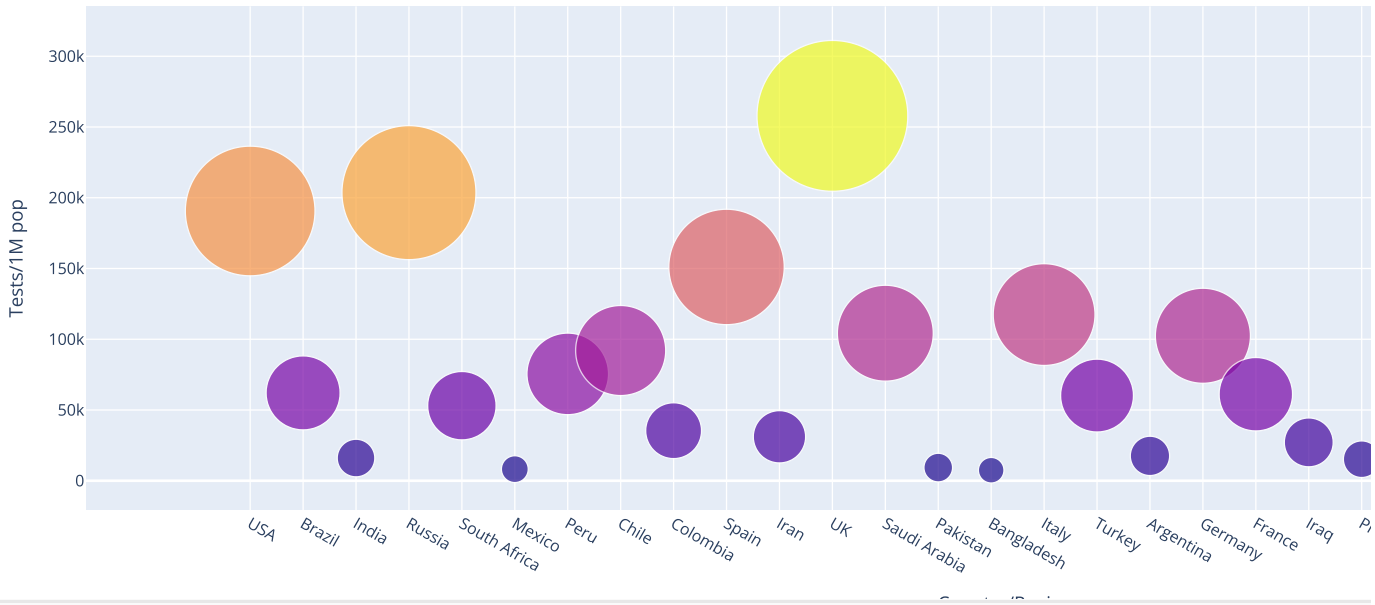
Example: Bubble chart

```
px.scatter(dataset1.head(10), x='Country/Region', y= 'TotalDeaths',
           hover_data=['Country/Region', 'Continent'],
           color='Country/Region', size= 'TotalDeaths', size_max=80)
```

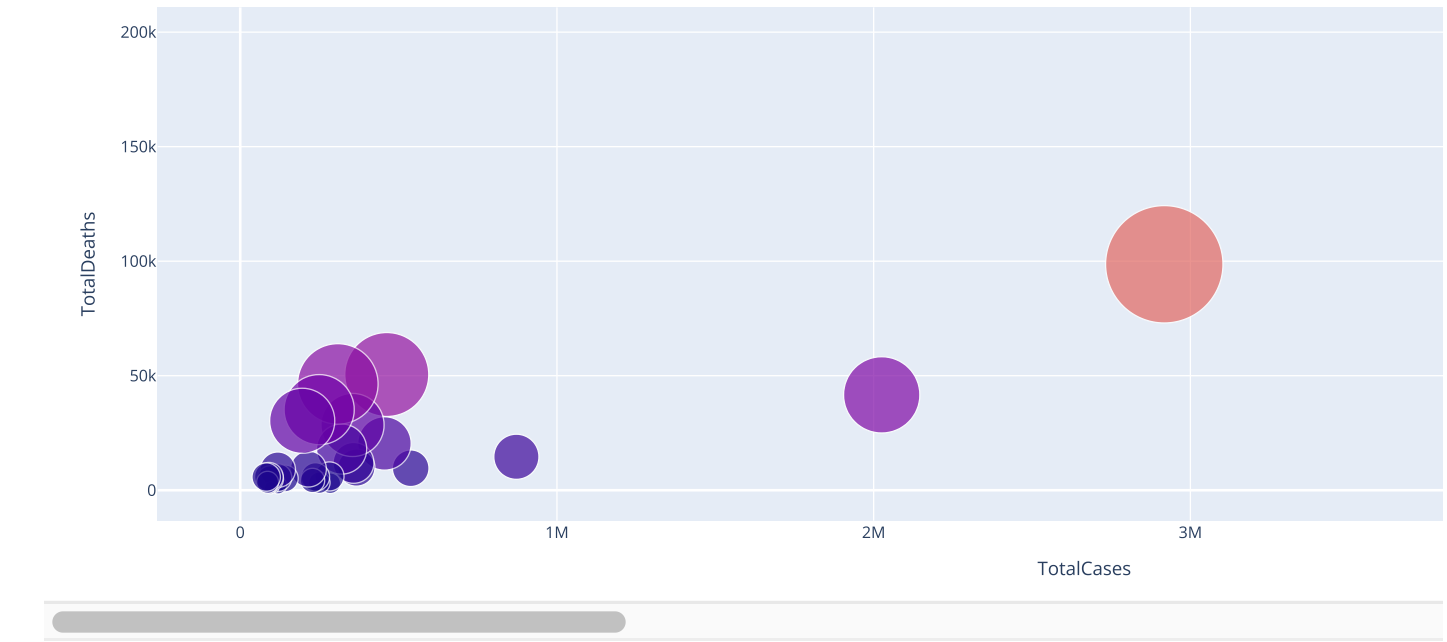


Example: Country/Region VS Tests/1M pop (color-scale of Tests/1M pop)

```
px.scatter(dataset1.head(30), x='Country/Region', y= 'Tests/1M pop',
           hover_data=['Country/Region', 'Continent'],
           color='Tests/1M pop', size= 'Tests/1M pop', size_max=80)
```

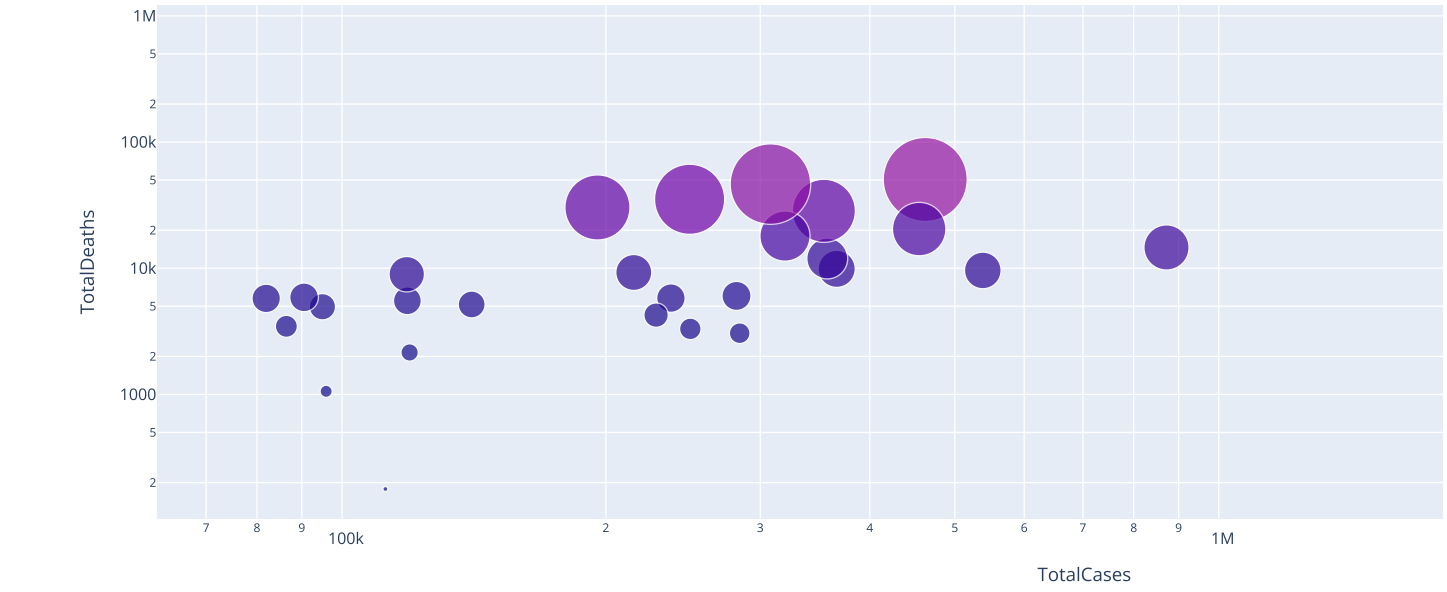



```
px.scatter(dataset1.head(30), x='TotalCases', y= 'TotalDeaths',
           hover_data=['Country/Region', 'Continent'],
           color='TotalDeaths', size= 'TotalDeaths', size_max=80)
```

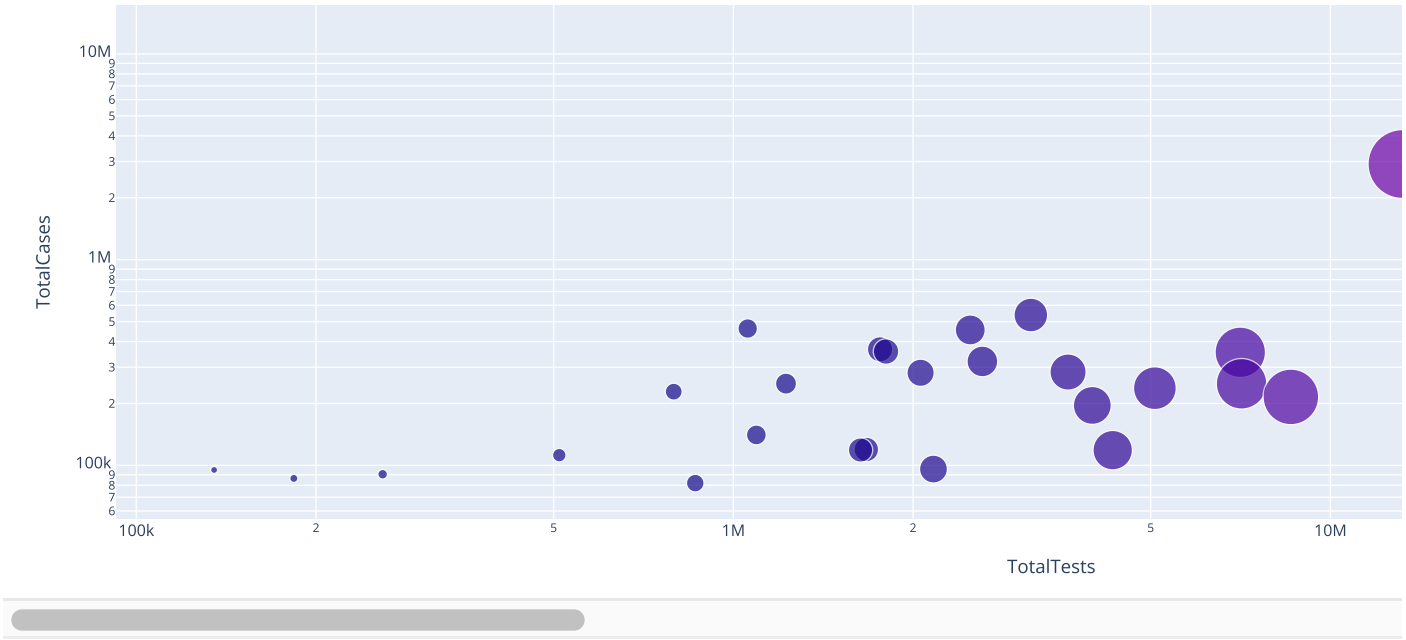


It is clear from the result that they have a linear relationship between the total number of cases and the total number of deaths. That means more cases, more deaths.

```
px.scatter(dataset1.head(30), x='TotalCases', y= 'TotalDeaths',
           hover_data=['Country/Region', 'Continent'],
           color='TotalDeaths', size= 'TotalDeaths', size_max=80,
           log_x=True, log_y=True)
```




```
px.scatter(dataset1.head(30), x='TotalTests', y= 'TotalCases',
           hover_data=['Country/Region', 'Continent'],
           color='TotalTests', size= 'TotalTests', size_max=80,
           log_x=True, log_y=True)
```

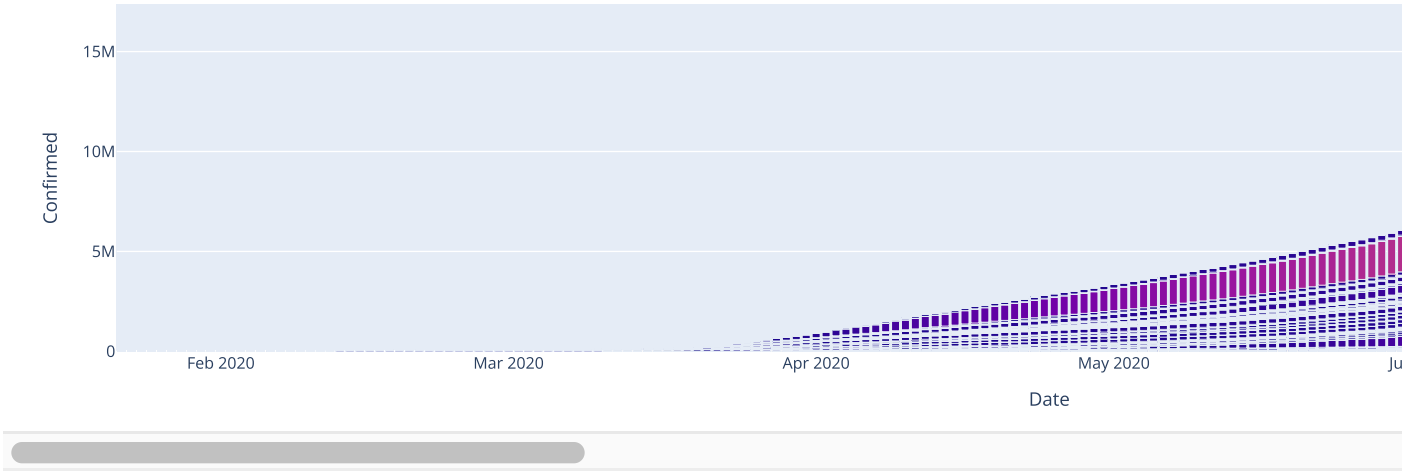


Step 7: Advanced Data Visualization-Bar graphs for All top infected Countries

In this task, we will explore covid-19 data using bar graphs and charts and use dataset2 as it has date column.

Example: Bar chart

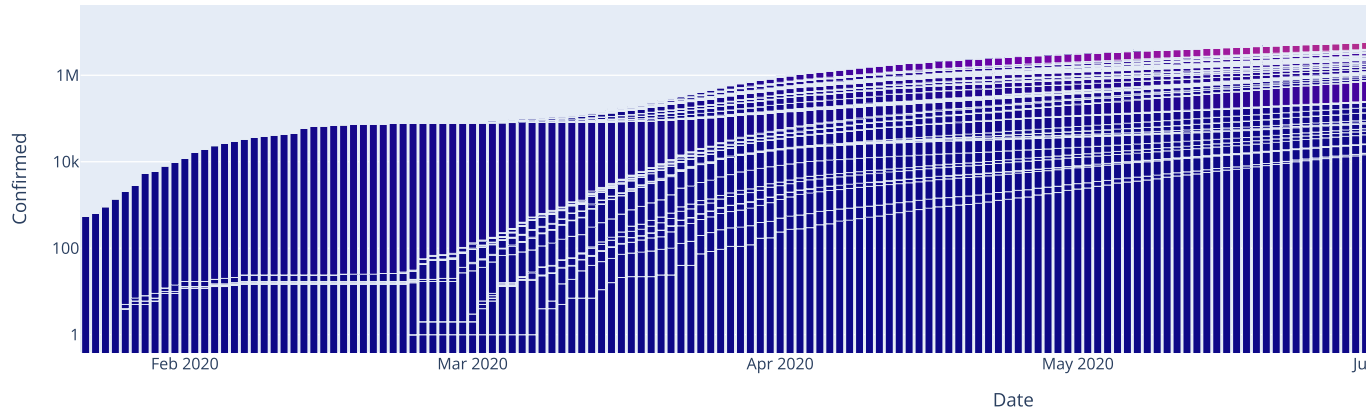
```
px.bar(dataset2, x="Date", y="Confirmed", color="Confirmed",
       hover_data=["Confirmed", "Date", "Country/Region"], height=400)
```



The above graph we get as output which includes all countries with respect to recovered cases. we can imagine the exponential growth of corona cases by date. We can use log function for this to be more clear.

Example: Bar chart

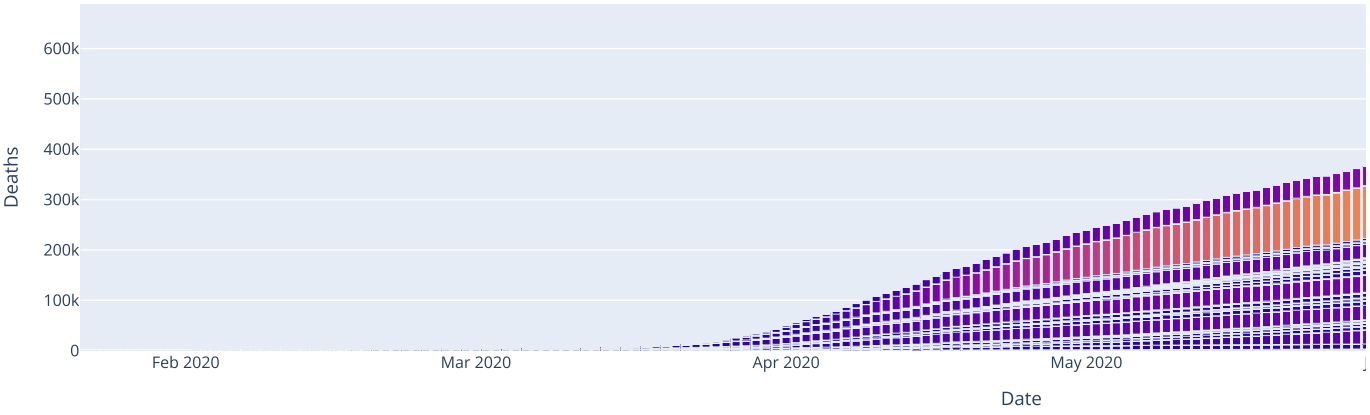
```
px.bar(dataset2, x="Date", y="Confirmed", color="Confirmed",
       hover_data=["Confirmed", "Date", "Country/Region"],log_y=True, height=400)
```



Let's imagine death instead of confirmation with the same and color it by date.

Example: Bar chart

```
px.bar(dataset2, x="Date", y="Deaths", color="Deaths",
       hover_data=["Confirmed", "Date", "Country/Region"],
       log_y=False, height=400)
```



Step 8: Countries Specific COVID Data Visualization: (United States)

In this specific task, we will analyze data of the USA country.

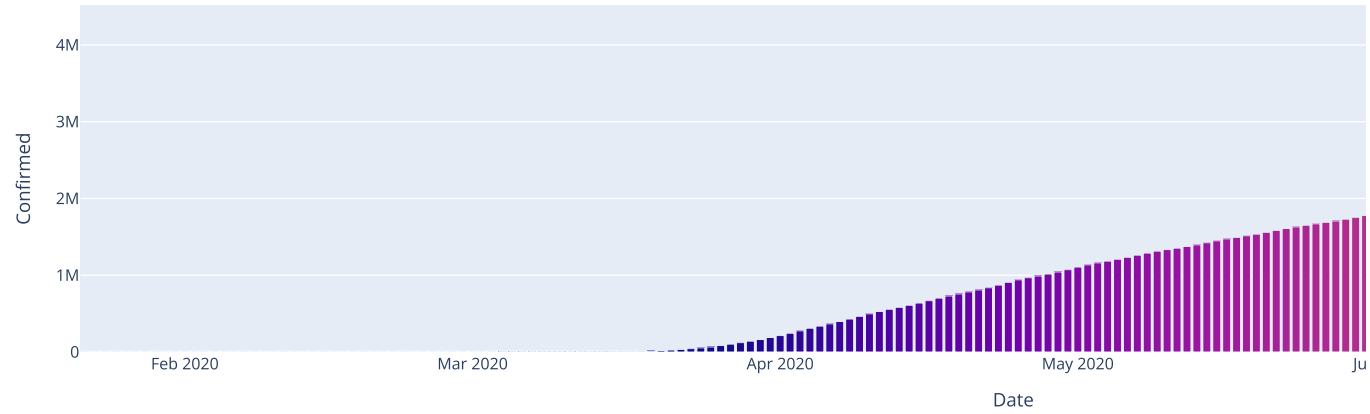
Example: Refining dataset to get only USA data

```
df_US= dataset2.loc[dataset2["Country/Region"]=="US"]
```

Now let us plot and study the covid situation in the USA.

Example: Bar chart

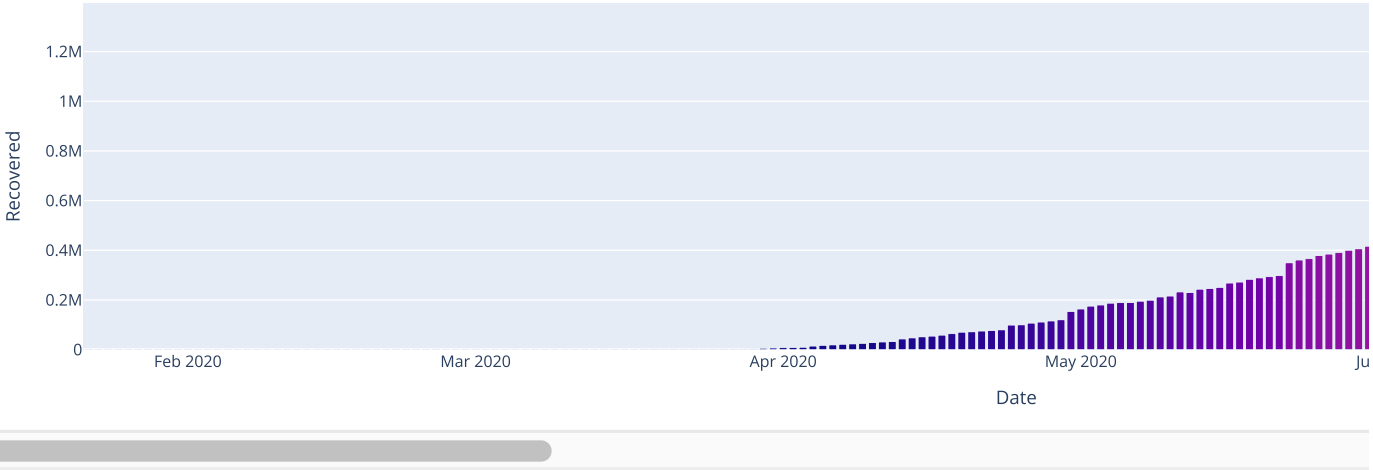
```
px.bar(df_US, x="Date", y="Confirmed", color="Confirmed", height=400)
```



Here we can clearly see how the confirmed cases increased in the United States with respect to time (January 2020 to July 2020). Similarly, we can check the same for recovered cases, tests and deaths.

Example: Bar chart

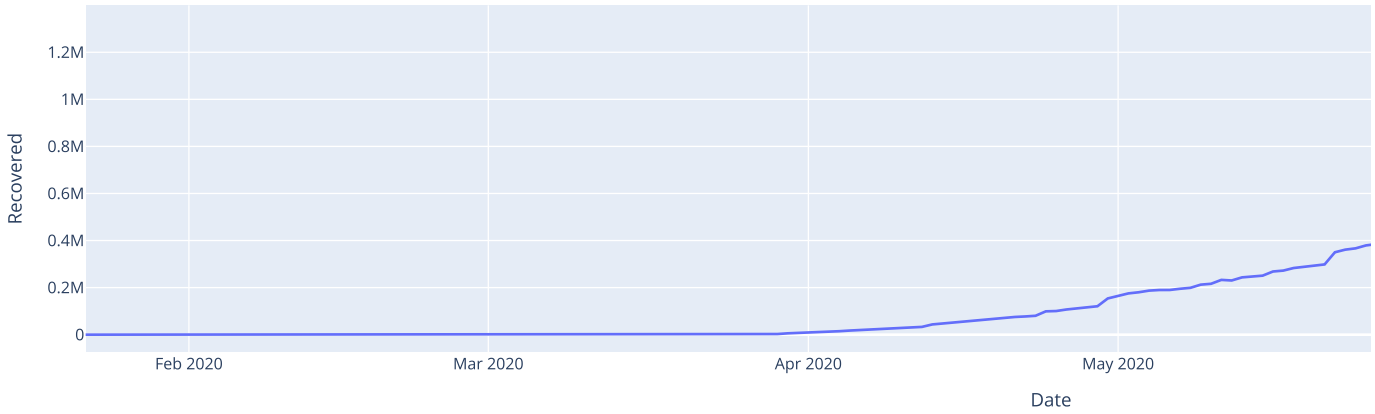
```
px.bar(df_US,x="Date", y="Recovered", color="Recovered", height=400)
```



Similarly, we can analyze the data in all the ways to generate the line graph for the same.

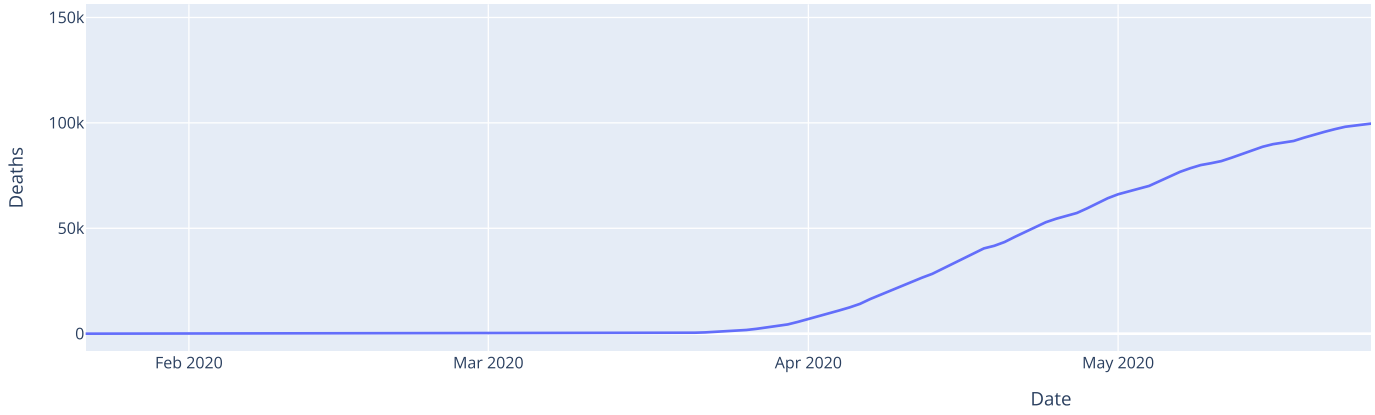
Example: Line plot

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

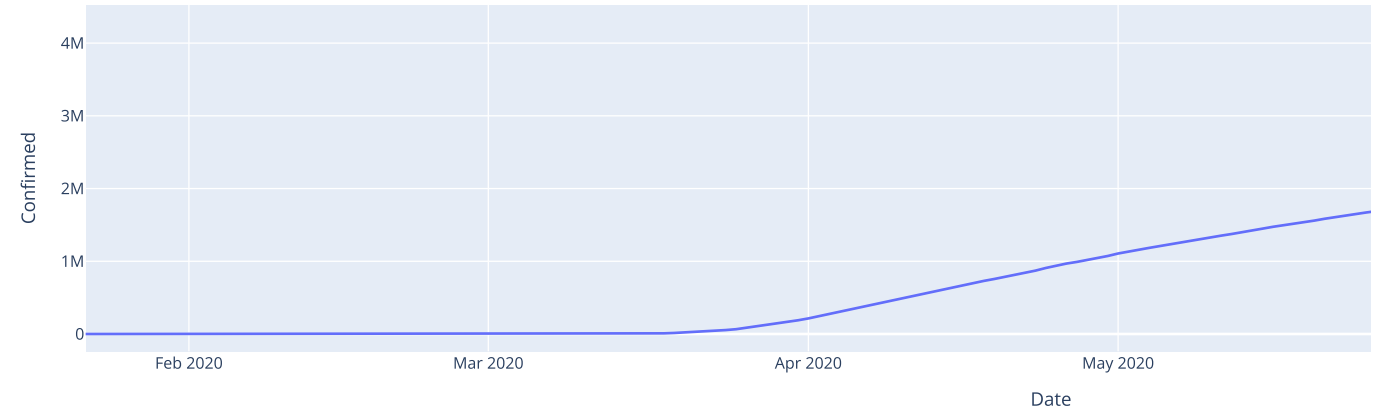


Example: Line plot

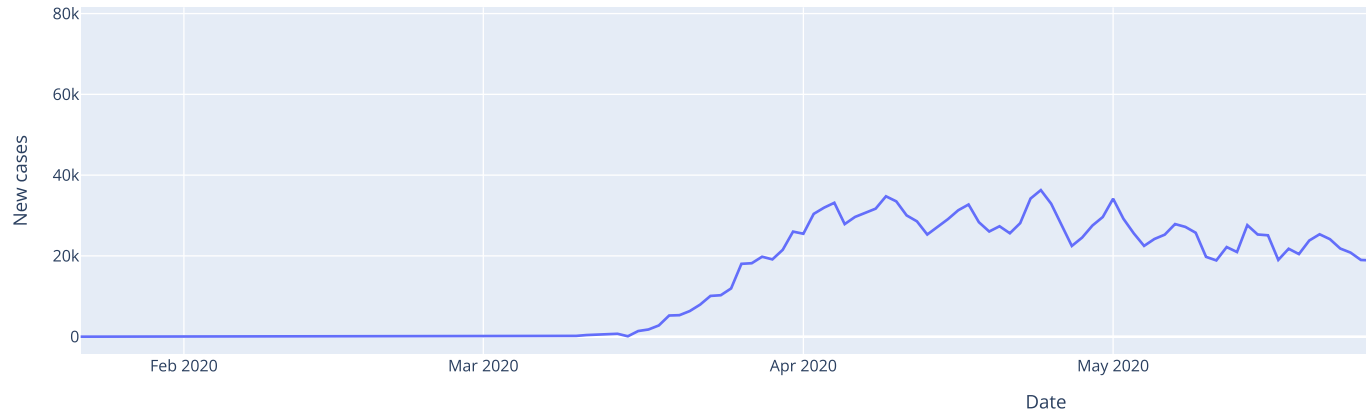
```
px.line(df_US,x="Date", y="Deaths", height=400)
```




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



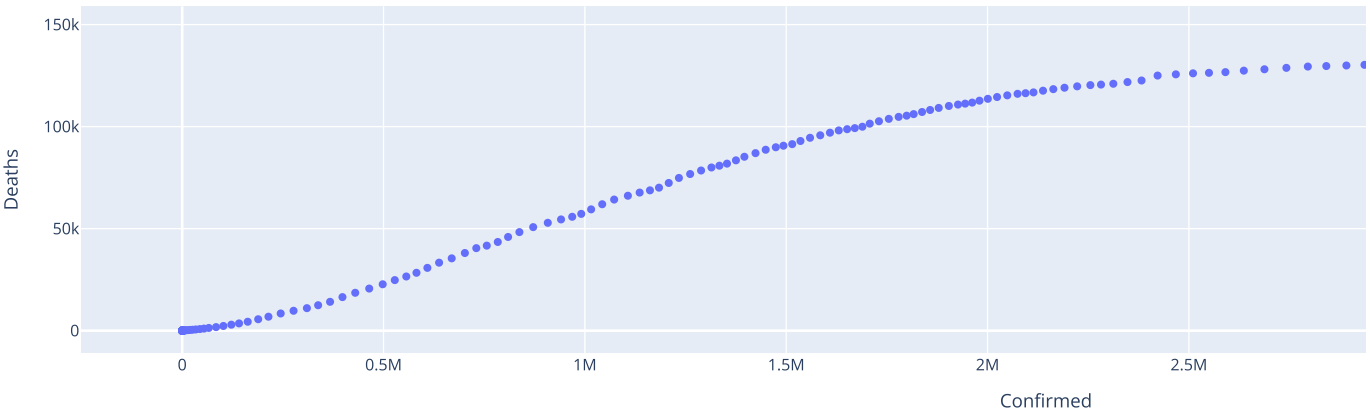

```
px.line(df_US,x="Date", y="New cases", height=400)
```



Similarly, let us also plot a line plot.

Example: Scatter plot

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



Step 9: Visualization of Data in terms of Maps We can use choropleth to visualize the data in terms of maps, with maps usually being the predominant way of visualizing the data. Since COVID-19 is a global phenomenon and so we look through and fix them in terms of wall maps. Ortho-graphics, rectangular and natural earth projection to visualize the data With dataset2 for the purpose as it has Dates column. It will look at the growth of Covid-19 (from Jan to July 2020) as in how the virus reached across the world.

Choropleth is an amazing representation of data on a map. Choropleth maps provide an easy way to visualize how a measurement varies across a geographic areal-Life

Project Application in Real choropleth map displays divided geographical areas or regions that are colored, shaded or patterned in relation to a data variable.

Equi-rectangular Projection:

```
px.choropleth(dataset2,
              locations="iso_alpha",
              color="Confirmed",
              hover_name="Country/Region",
              color_continuous_scale="Blues",
              animation_frame="Date")
```

This creates an animation containing visualizations from January to July 2020. Playing this animation will make it more clear how the virus spread around the world. The darker the color, the higher the confirmed cases are.

Creating map

```
px.choropleth(dataset2,
              locations='iso_alpha',
              color="Deaths",
              hover_name="Country/Region",
              color_continuous_scale="Viridis",
              animation_frame="Date" )
```

This code creates an animation of death cases by date. By playing this animation it will be shown how deaths increase around the world.

Natural Earth projection is a compromise pseudo-cylindrical map projection for world maps.

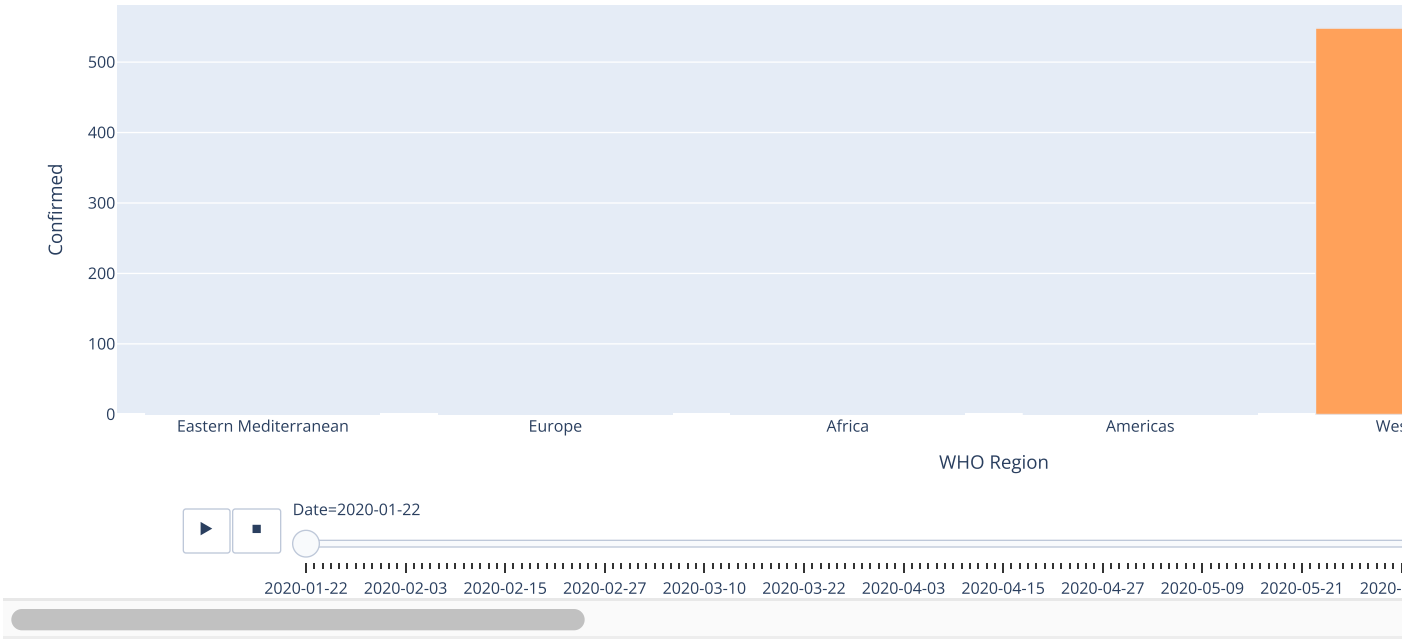
Example: Natural earth projection

```
px.choropleth(dataset2,
              locations='iso_alpha',
              color="Recovered",
              hover_name="Country/Region",
              color_continuous_scale="RdYlGn",
              projection="natural earth",
              animation_frame="Date" )
```

By running the output, things start to become more clear about how the recovery rate changes with respect to the date. Lets also look at how an animation can be added to a bar graph. We can convert the bar graph into animation using Dates column that is in dataset2.

Example: Bar graph animation

```
px.bar(dataset2, x="WHO Region", y="Confirmed", color="WHO Region",
       animation_frame="Date", hover_name="Country/Region")
```



When running the output, the animation will run from January to July 2020. It will show 6 different bar graphs, each continent has its own color representing the confirmed cases.

Step 10: Visualize text using Word Cloud Visualize the causes of death due to covid-19, as covid-19 affects people in different ways, hence creating a word cloud to visualize the leading cause of covid-19 deaths. To visualize the text the steps need to be followed are-

Used to convert data elements of an array into list. Convert the string to one single string. Convert the string into word cloud Dataset3: This dataset contains real world examples of number of Covid-19 deaths and the reasons behind the deaths.

Example: Importing dataset

```
dataset3= pd.read_csv("coviddeath.csv")
dataset3.head()
```

	Data as of	Start Week	End Week	State	Condition Group	Condition	ICD10_codes	Age Group	Number of COVID-19 Deaths	Flag
0	08/30/2020	02/01/2020	08/29/2020	US	Respiratory diseases	Influenza and pneumonia	J09-J18	0-24	122.0	NaN
1	08/30/2020	02/01/2020	08/29/2020	US	Respiratory diseases	Influenza and pneumonia	J09-J18	25-34	596.0	NaN
2	08/30/2020	02/01/2020	08/29/2020	US	Respiratory	Influenza and	J09-J18	35-44	1521.0	NaN

Getting dataset information

```
dataset3.groupby(["Condition"]).count()
```

Condition	Data as of	Start Week	End Week	State	Condition Group	ICD10_codes	Age Group	Number of COVID-19 Deaths	Flag
Adult respiratory distress syndrome	540	540	540	540	540	540	540	272	268
All other conditions and causes (residual)	540	540	540	540	540	540	540	363	177
Alzheimer disease	530	530	530	530	530	530	530	144	386
COVID-19	540	540	540	540	540	540	540	377	163
Cardiac arrest	520	520	520	520	520	520	520	219	301
Cardiac arrhythmia	540	540	540	540	540	540	540	192	348
Cerebrovascular diseases	530	530	530	530	530	530	530	187	343
Chronic lower respiratory diseases	540	540	540	540	540	540	540	229	311
Diabetes	540	540	540	540	540	540	540	276	264
Heart failure	540	540	540	540	540	540	540	204	336
Hypertensive diseases	540	540	540	540	540	540	540	264	276
Influenza and pneumonia	540	540	540	540	540	540	540	331	209
Intentional and unintentional injury, poisoning, and other adverse events	520	520	520	520	520	520	520	188	332
Ischemic heart disease	540	540	540	540	540	540	540	224	316
Malignant neoplasms	540	540	540	540	540	540	540	198	342
Obesity	530	530	530	530	530	530	530	182	348
Other diseases of the circulatory system	530	530	530	530	530	530	530	213	317
Other diseases of the respiratory system	540	540	540	540	540	540	540	188	352
Renal failure	540	540	540	540	540	540	540	238	302
Respiratory arrest	480	480	480	480	480	480	480	111	369
Respiratory failure	540	540	540	540	540	540	540	320	220

Creating wordcloud

```
# import word cloud
from wordcloud import WordCloud

sentences = dataset3["Condition"].tolist()
sentences_as_a_string = ' '.join(sentences)


# Convert the string into WordCloud
plt.figure(figsize=(20, 20))
plt.imshow(WordCloud().generate(sentences_as_a_string))
```

```
<matplotlib.image.AxesImage at 0x7f44b34c1f00>
```