# Covid 19 World Status

## Introduction

Covid 19 outbreak, a subject that is not indifferent to anybody, has affected all levels of the society in many ways: economy, education, tourism, employment, entertainment, and the list goes on and on. Since early 2020, we have been having a global health crisis unlike any era in the 75-year United Nations history. A crisis that has cost lives, suffering and, in the least of the cases, upending people's lives.

With massive data volumes being generated by both professionals and researchers have an opportunity to fight the pandemic by analyzing the data so world leaders make accurate and effective countermeasures to fight the pandemic.

## Objetive

The fist step in order to solve a problem, is to understand it, and we can do this by deep dive into the data and ask us questions. For this particular project we want to solve the following ones:

- How has the virus spread across the world?
- How confirmed cases, deaths and recovery has fluctuated across the world?
- What's the current status at each country?

## Data

We are going to use data from the Johns Hopkins Resource Center that can be found here in a tidy format.

In [1]:
```
import pandas as pd
import plotly.express as px
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from plotly.figure_factory import ff
import folium
import warnings
warnings.filterwarnings("ignore")

data_url = 'https://raw.githubusercontent.com/datasets/covid-19/master/data/countries-aggregated.csv'
df = pd.read_csv(data_url)
```

In [2]:
```
fig = px.choropleth(
    df,
    locations = 'Country',
    locationmode='country names',
    color='Confirmed',
    color='Country',
    animation_frame='Date')
fig.update_layout(
    title_text = 'Covid-19 Global Spread',
    title_x = 0.5,
    geo=dict(
        showframe = False,
        showcoastlines = False,
    ))
fig.layout.updatemenus[0].buttons[0].args[1]['frame']['duration'] = 10
fig.layout.updatemenus[0].buttons[0].args[1]['transition']['duration'] = 10
fig.update_geos(projection_type="equirectangular", visible=True, resolution=110)
fig.show()
```


The virus by January would slowly start to shut down much of the planet and consequently getting to the point of costing more than 3.5 million people in the most disruptive global health disaster since the influenza pandemic of 1918.

In [3]:
```
data = list(df['Date'].unique())

World_infection_rate = []
World_death_rate = []
World_recovery_rate = []
for d in data:
    World = df[df.Date == day].Country=='World']
    World_infection_rate.append(World)
    World_death_rate.append(World.Deaths.sum())
    World_death_rate.append(World)
    World_recovery_rate.append(World.Recovered.sum())
    World_recovery_rate.append(World)

World=pd.DataFrame()
World['Date']=data
World['World Infection']=World_infection_rate
World['World Death']=World_death_rate.diff()
World['World Deaths']=World_death_rate
World['World Death Rate']=World['World Deaths'].diff()
World['World Recovery']=World_recovery_rate
World['World Recovery Rate']=World['World Recovery'].diff()

World.at[World['Date']=='2020-12-14', 'World Recovery Rate']= np.nan
World.at[World['Date']=='2020-12-13', 'World Recovery Rate']= np.nan
World.at[World['Date']=='2020-12-14', 'World Infection Rate']= np.nan
World['World Infection Rate']=World['World Infection Rate'].interpolate();
World['World Recovery Rate']=World['World Recovery Rate'].interpolate(); 1

World['World Infection Rate Avg']=World['World Infection Rate'].rolling(5).mean()
World['World Death Rate Avg']=World['World Death Rate'].rolling(5).mean()
World['World Recovery Rate Avg']=World['World Recovery Rate'].rolling(5).mean()
```

In [4]:
```
fig=px.line(World, x='Date', y= ['World Infection Rate Avg', 'World Recovery Rate Avg', 'World Death Rate Avg'])
fig.update_layout(
    hovermode='x unified',
    updatemenus=[
        dict(
            type = "buttons",
            direction = "left",
            buttons=list([
                dict(
                    args=["yaxis.type", "linear"],
                    label="LINEAR",
                    method="relayout"
                ),

                dict(
                    args=["yaxis.type", "log"],
                    label="LOG",
                    method="relayout"
                )
            ]),
        )
    ])
fig.show()
```


As you can see the spread have been uprising March, even with the start of the quarantine at the end of March in many countries. A constant increase is been observed trought 2020 having the December the highest peak of that year, probably due to holidays and people not following social distancing policies, family meetings and unadequate customs. Since that point things start to be gotter until March when the infection rate starts rising again.

In [5]:
```
countries = list(df['Country'].unique())

Confirmed_cases = []
Infection_rate = []
Total_Deaths = []

for country in countries:
    Confirmed_cases.append(df[df.Country == country].Confirmed.max())
    Confirmed_cases.append(Total_cases)
    Max_rate = df[df.Country == country].Confirmed.diff().max()
    Infection_rate.append(Max_rate)
    Total_Deaths = df[df.Country == country].Deaths.max()
    Total_Deaths.append(Deaths)

df_World_cases=pd.DataFrame()
df_World_cases['Country']=countries
df_World_cases['Total Cases']=Confirmed_cases
df_World_cases['Infection_rate']=Infection_rate
df_World_cases['Total Deaths']=Total_Deaths

df_World_cases=df_World_cases.sort_values('Total Cases', ascending=False).head(10)
df_World_cases.at[df_World_cases['Country']=='US', 'Country']='United States'
df_World_cases.at[df_World_cases['Country']=='Russia', 'Country']='Russia'
```

In order to get to know the situation of an specific country, you can select a country and use the code below to have a broad view of its status. With the help of some funtions, you can now reproduce the process we just saw for any country of your selection. The graphs you are going to get include:

- Covid 19 impact
- Death Rate
- Active Cases
- Total Accumulated Cases

For this example we are going to see Japan's situation across the pandemic.

In [6]:
```
print('Select a country:')
print(sorted(df['Country'].unique()))
df['Country'].unique()
```


```
Select a country:
array(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola',
       'Antigua and Barbuda', 'Argentina', 'Armenia', 'Australia',
       'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh',
       'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bhutan',
       'Bolivia', 'Bosnia and Herzegovina', 'Botswana', 'Brazil',
       'Brunei', 'Bulgaria', 'Burkina Faso', 'Burma', 'Burundi',
       'Cabo Verde', 'Cambodia', 'Cameroon', 'Canada',
       'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia',
       'Comoros', 'Congo (Brazzaville)', 'Congo (Kinshasa)', 'Costa Rica',
       'Cote d\'Ivoire', 'Croatia', 'Cuba', 'Cyprus', 'Czechia',
       'Diamond Princess', 'Djibouti', 'Dominica', 'Dominican Republic',
       'Ecuador', 'Egypt', 'El Salvador', 'Equatorial Guinea', 'Eritrea',
       'Estonia', 'Eswatini', 'Ethiopia', 'Fiji', 'Finland', 'France',
       'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Greece',
       'Grenada', 'Guatemala', 'Guinea', 'Guinea-Bissau', 'Guyana',
       'Haiti', 'Holy See', 'Honduras', 'Hungary', 'Iceland', 'India',
       'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Israel', 'Italy',
       'Jamaica', 'Japan', 'Jordan', 'Kazakhstan', 'Kenya', 'Korea, South',
       'Kosovo', 'Kuwait', 'Kyrgyzstan', 'Laos', 'Latvia', 'Lebanon',
       'Lesotho', 'Liberia', 'Libya', 'Liechtenstein', 'Lithuania',
       'Luxembourg', 'MS Zaandam', 'Madagascar', 'Malawi', 'Malaysia',
       'Maldives', 'Mali', 'Malta', 'Mauritania', 'Mauritius', 'Mexico',
       'Moldova', 'Monaco', 'Mongolia', 'Montenegro', 'Morocco', 'Mozambique',
       'Namibia', 'Nepal', 'Netherlands', 'New Zealand', 'Nicaragua',
       'Niger', 'Nigeria', 'North Macedonia', 'Norway', 'Oman',
       'Pakistan', 'Panama', 'Papua New Guinea', 'Paraguay', 'Peru',
       'Philippines', 'Poland', 'Portugal', 'Qatar', 'Romania', 'Russia',
       'Rwanda', 'Saint Kitts and Nevis', 'Saint Lucia',
       'Saint Vincent and the Grenadines', 'Samoa', 'San Marino',
       'Sao Tome and Principe', 'Saudi Arabia', 'Senegal', 'Serbia',
       'Seychelles', 'Sierra Leone', 'Singapore', 'Slovakia', 'Slovenia',
       'Solomon Islands', 'Somalia', 'South Africa', 'South Sudan',
       'Spain', 'Sri Lanka', 'Sudan', 'Suriname', 'Sweden', 'Switzerland',
       'Syria', 'Taiwan*', 'Tajikistan', 'Tanzania', 'Thailand', 'Timor-Leste',
       'Togo', 'Trinidad and Tobago', 'Tunisia', 'Turkey', 'Uganda',
       'US', 'Ukraine', 'Uruguay', 'Uzbekistan', 'Venezuela',
       'Vietnam', 'West Bank and Gaza', 'Yemen', 'Zambia', 'Zimbabwe'],
      dtype=object)
```


In [7]:
```
country = 'Japan'
```

In [8]:
```
def selected_country(country):
    df_country = df.loc[df.Country == country]
    df_country['Infection Rate'] = df_country.Confirmed.diff()
    df_country['Mortality Rate'] = df_country.Deaths.diff()
    df_country['Recovered Rate'] = df_country.Recovered.diff()
    df_country['Infection Rate Avg'] = df_country['Infection Rate'].rolling(5).mean()
    df_country['Mortality Rate Avg'] = df_country['Mortality Rate'].rolling(5).mean()
    df_country['Recovery Rate Avg'] = df_country['Recovery Rate'].rolling(5).mean()
    df_country['Active Cases'] = df_country['Confirmed'] - df_country['Recovered']
    df_country['Active Cases Avg'] = df_country['Active Cases'].rolling(5).mean()

    return country, df_country

name_country = selected_country(country)[0]
df_c = selected_country(country)[1]
```

In [9]:
```
def impact(df_country):
    fig = px.line(df_country, title='Covid 19 impact on {}'.format(country), x
    return fig

def death_cases(df_country):
    fig2 = px.line(df_country, title='Death Cases in {}'.format(country), x='Dat
    return fig2

def active_cases(df_country):
    fig3 = px.line(df_country, title='Active Cases in {}'.format(country), x='Da
    return fig3

def accumulated(df_country):
    fig4 = px.line(df_country, title='Total Accumulated Cases in {}'.format(coun
    fig4.update_layout(
    hovermode='x unified',
    updatemenus=[
        dict(
            type = "buttons",
            direction = "left",
            buttons=list([
                dict(
                    args=["yaxis.type", "linear"],
                    label="LINEAR",
                    method="relayout"
                ),

                dict(
                    args=["yaxis.type", "log"],
                    label="LOG",
                    method="relayout"
                )
            ]),
        )
    ])
    return fig4
```

In [10]:
```
Impact = impact(df_c)
Death = death_cases(df_c)
Active = active_cases(df_c)
Ac = accumulated(df_c)
```

In [11]:
```
Impact
```


In [12]:
```
Death
```


In [13]:
```
Active
```




In [14]:
```
Ac
```




## Conclussions and Recommendations

Get to know the Covid-19 status around the world or in a particular country can help governments to apply prudent policies to fight the virus. Either social distancing, business closing, borders closing or a general lockdown it is important to be well informed about the topic using powerful tools like python to make right decisions in the data-driven world we live in.