Covid-19 Vaccines Analysis

Team member

723721205018 : GOKULA KANNAN M

Phase-2 Document Submission

Introduction:

The development and distribution of COVID-19 vaccines have been pivotal in the global response to the ongoing pandemic caused by the SARS-CoV-2 virus. Since the emergence of the virus in late 2019, researchers, healthcare professionals, and policymakers have been working tirelessly to understand, evaluate, and optimize various COVID-19 vaccines.

Time series data analysis plays a crucial role in this effort. It involves tracking and analyzing vaccine distribution, administration rates, and the evolution of the pandemic over time. Time series data allows us to monitor the effectiveness of vaccination campaigns, identify trends, and make informed decisions about public health measures. This analysis aims to explore key aspects of COVID vaccine research and deployment, including vaccine efficacy, safety, distribution, and their role in curbing the spread of the

virus. It will incorporate time series data to provide a dynamic view of how vaccination efforts have evolved over months or even years. Additionally, it will highlight the significance of ongoing data collection and research efforts in monitoring the effectiveness of these vaccines in the face of evolving viral variants and the long-term impact of vaccination on public health.

Program Code:

Importing packages and libraries:

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from plotly.offline import init_notebook_mode, iplot, plot
import plotly as py
init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.express as px
from pandas_profiling import ProfileReport
```

Look Into the Data:

Here we are going to create a function called 'getDf' which is used to return the original datasets. We can check the Head of the data and get the description and information about it.

Now for the whole information:

```
In [7]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74961 entries, 0 to 74960
Data columns (total 15 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   country                               74961 non-null  object
 1   iso_code                              74961 non-null  object
 2   date                                  74961 non-null  object
 3   total_vaccinations                    39262 non-null  float64
 4   people_vaccinated                     37222 non-null  float64
 5   people_fully_vaccinated               34488 non-null  float64
 6   daily_vaccinations_raw                32891 non-null  float64
 7   daily_vaccinations                    74636 non-null  float64
 8   total_vaccinations_per_hundred        39262 non-null  float64
 9   people_vaccinated_per_hundred         37222 non-null  float64
 10  people_fully_vaccinated_per_hundred   34488 non-null  float64
 11  daily_vaccinations_per_million        74636 non-null  float64
 12  vaccines                              74961 non-null  object
 13  source_name                           74961 non-null  object
 14  source_website                        74961 non-null  object
dtypes: float64(9), object(6)
memory usage: 8.6+ MB
```

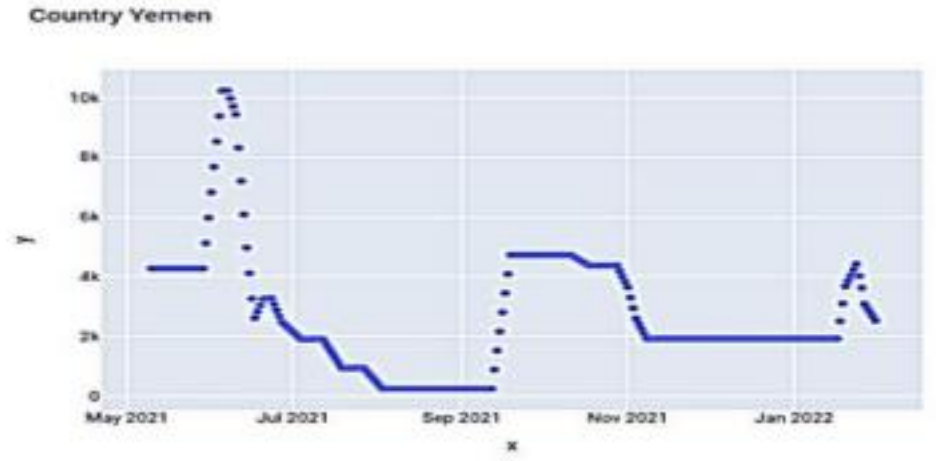
```
In [8]: df.describe()
```

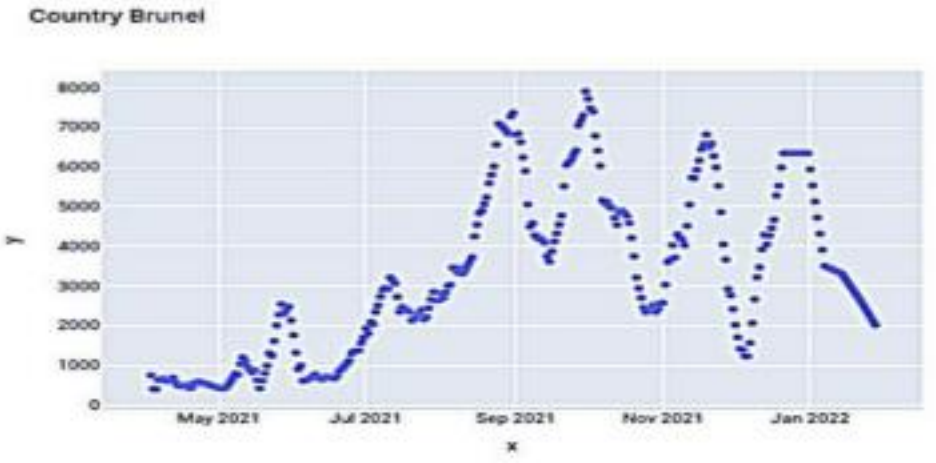
Out[8]:

	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinatio
count	3.926200e+04	3.722200e+04	3.448800e+04	3.289100e+04	7.463600e+04
mean	3.874683e+07	1.324878e+07	1.131666e+07	2.759083e+05	1.273043e+05
std	1.932234e+08	8.089730e+07	4.978853e+07	1.247145e+06	8.077875e+05
min	0.000000e+00	0.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00
25%	4.595625e+05	3.060028e+05	1.997862e+05	5.374500e+03	9.827500e+01
50%	3.031700e+06	1.698688e+06	1.347336e+06	2.673400e+04	8.019600e+03
75%	1.435591e+07	7.874247e+06	5.974702e+06	1.276480e+05	4.592526e+04
max	3.002806e+09	1.268070e+09	1.227387e+09	2.474100e+07	2.342429e+07

The Plotting:

```
In [7]:
def dateVacination(countryCode):
    df = getDf()

    dfA = df[df['iso_code'] == countryCode]
    liste = list(set(dfA['country']))

    fig = px.scatter(x=dfA['date'] , y=dfA['daily_vaccinations'],title='Country
{}'.format(liste[0]))

    fig.show()


liste = list(set(df['iso_code']))

for i in range(3):
    source_code = liste[i]
    dateVacination(source_code)
```





In the above graphs ww can see some missing values.

Data Analysis:

Here we are going to make analysis of data using different types of plots. Let's take the date 2022-02-04.

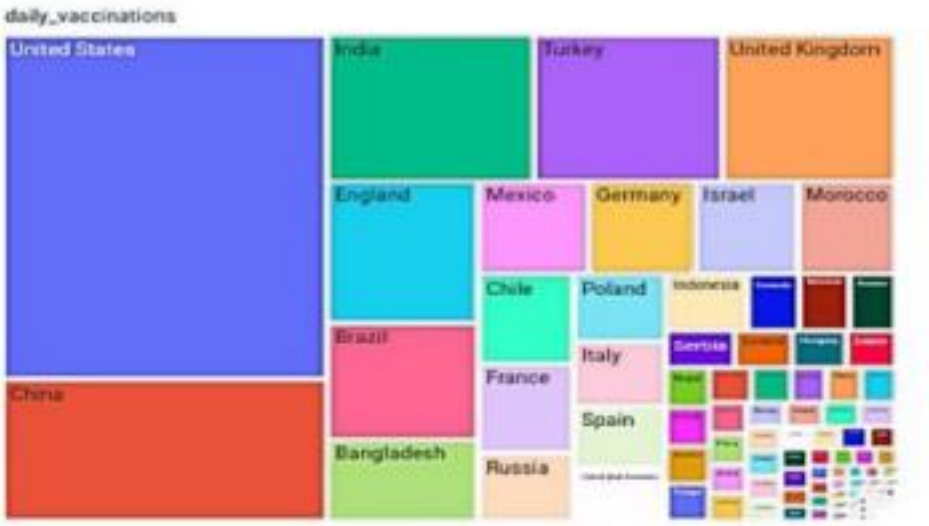
In [8]:
```
df = df[ df['date'] == '2022-02-04']
df = df[['country','daily_vaccinations']].sort_values(by='daily_vaccinations', a
scending=False)
```

In [9]:
```
fig = px.treemap(df, path=[px.Constant('daily_vaccinations'),'country'], values
='daily_vaccinations',
                 hover_data=['country'])
fig.show()
```

Compare It with The First Day:

In [10]:
```
df = getDf()

df = df[ df['date'] == '2021-02-22']
df = df[['country','daily_vaccinations']].sort_values(by='daily_vaccinations', a
scending=False)
```

In [11]:
```
fig = px.treemap(df, path=[px.Constant('daily_vaccinations'),'country'], values
='daily_vaccinations',
                 hover_data=['country'])
fig.show()
```



Note: INDIA to the place is USA.

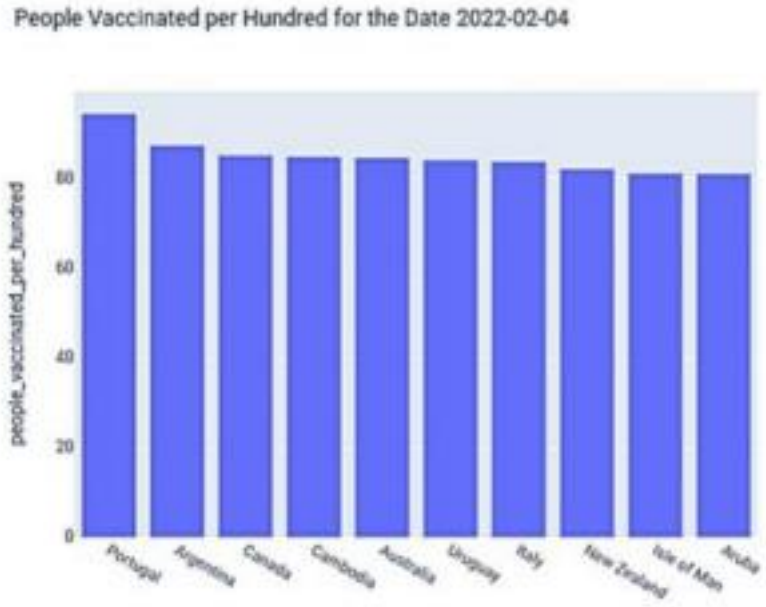
Deeper Analysis:

The dataframes gives us the ratio percent, vaccines/ Population which will be helpful.

In [12]:
```
df = getDf()
```

In [13]:
```
df = df[ df['date'] == '2022-02-04']
df = df.sort_values(by='people_vaccinated_per_hundred',ascending=False)
```

In [14]:
```
px.bar(df.head(10), x='country', y='people_vaccinated_per_hundred',
                    title='People Vaccinated per Hundred for the Date 2022-02-0
4')
```



We can say often that these countries have population between 5 to 40 million.

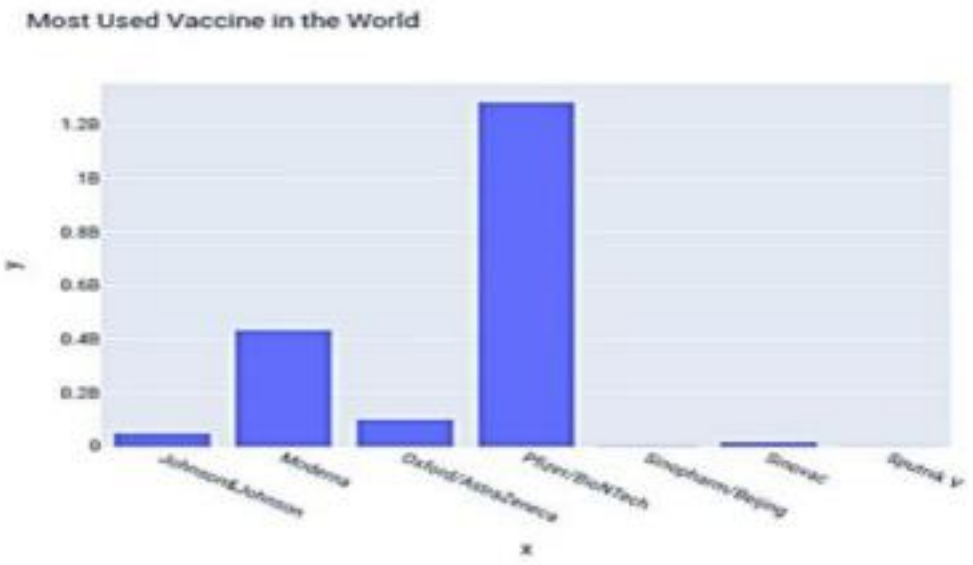
Vaccination:

Belgium uses the vaccine named 'BioNTech' . The statistics For countries like India, USA, China

are not available.

In [28]:

```
total  = dfA.groupby('vaccine').sum()

px.bar(x=total.index, y=total['total_vaccinations'],
                   title='Most Used Vaccine in the World')
```

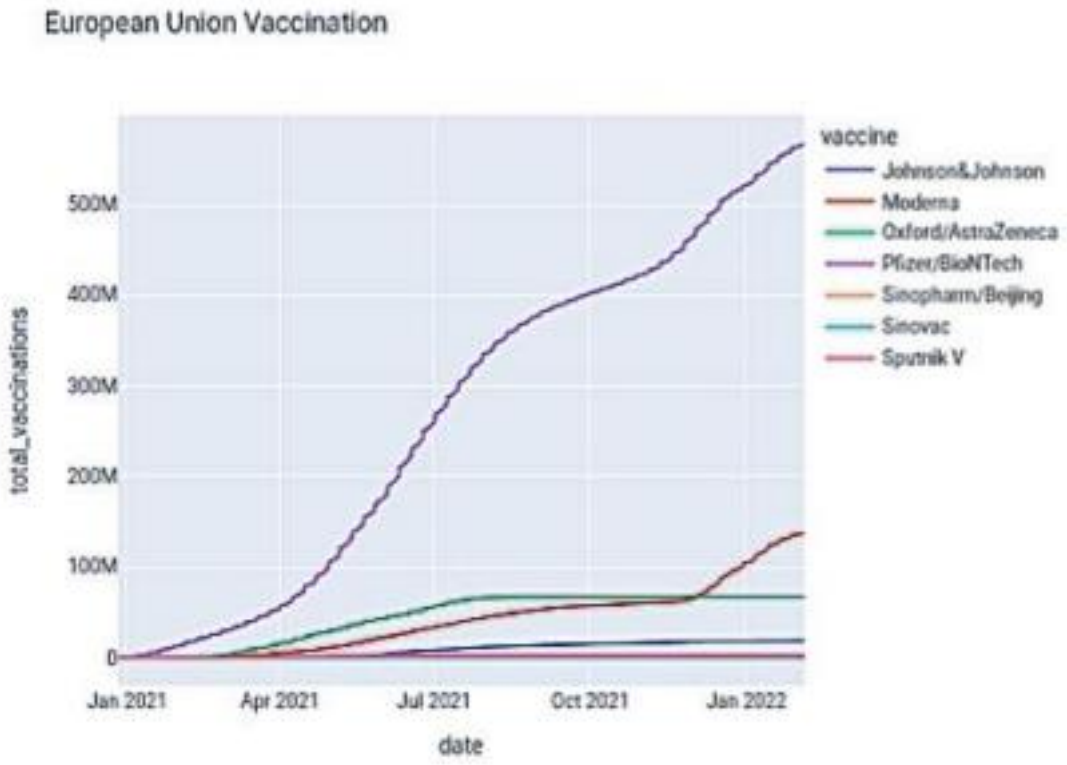


The most used vaccine is BioNTech .

Let's see the European Union graph:

In [23]:
```
europeanUnion = dfA[dfA['location'] ==  'European Union']
```

In [24]:
```
px.line(europeanUnion, x="date", y="total_vaccinations",color='vaccine',title='E
uropean Union Vaccination')
```



We can also create a function to view any country's Graph.

In [25]:
```
def vaccinationDuration(location):

    dfA = getDfA()

    europeanUnion = dfA[dfA['location'] ==  location]

    fig = px.line(europeanUnion, x="date", y="total_vaccinations",color='vaccin
e',title='{} Vaccination'.format(location))

    fig.show()
```

In [26]:
```
import random
listeOfCountry = list(set(dfA['location']))

for i in range(3):
    location = random.choice(listeOfCountry)
    vaccinationDuration(location)
```

Conclusion:

In conclusion, the analysis of COVID-19 vaccines and their impact on the ongoing pandemic has been highlighted . Time series data analysis has been instrumental in tracking the progress of vaccination campaigns, enabling us to monitor trends and make data-driven decisions to mitigate the spread of the virus.However, the fight against COVID-19 is ongoing, with the emergence of new variants requiring continuous monitoring and potential vaccine adjustments. The success of vaccination efforts also relies on equitable distribution and addressing vaccine hesitancy.