**Phase 3 : Covid World Vaccination Progress**

|  |  |  |
| --- | --- | --- |
| **NAME** | **REG NO:** | **EMAIL ID** |
| **Jeyasurya S** | **312821243010** | **21aids13@act.edu.in** |



The Covid World Vaccination Progress **(https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-progress)** dataset on Kaggle is a comprehensive collection of information pertaining to the ongoing COVID-19 vaccination drive across various countries worldwide.

This dataset provides substantial insight into each country's trajectory in its vaccination campaign. Here are some of the main aspects and types of information you can find in this dataset:

**1. Country:** The name of the country for which the vaccination information is provided.

**2. Date:** The date for the corresponding entry.

**3. Total number of vaccinations:** This is the absolute count of total immunizations in the country.

**4. Total number of people vaccinated:** This is the total number of people who received at least one vaccine dose.

**5. Total number of people fully vaccinated:** This represents the number of people who received all doses prescribed by the vaccination protocol.

**6. Daily vaccinations (raw):** This is the count for vaccinations given on the actual day.

**7. Daily vaccinations:** This holds the count for vaccinations for a specific date, which may differ from 'daily vaccinations (raw)' due to certain calculation methods.

**8. Total vaccinations per hundred:** This is the ratio (in percent) of total vaccinations per total population up to the date in the country.

**9. Total number of people vaccinated per hundred:** The ratio (in percent) of population that received at least one vaccine dose.

**10. Total number of people fully vaccinated per hundred:** The ratio (in percent) indicating the total number of people fully vaccinated.

# **Loading the dataset:**

**An explanation of each library's purpose and how they can be used in data analysis and visualization:**

**1. Pandas:**

Purpose: Pandas is a popular data manipulation and analysis library for Python. It provides data structures such as DataFrames and Series, making it easy to load, manipulate, and analyze tabular data.

**2. Plotly Express and Plotly Graph Objects:**

Purpose: Plotly is a library for interactive and visually appealing data visualization. Plotly Express provides a high-level API for creating plots, while Plotly Graph Objects offers a lower-level, more customizable approach.

**3. Folium:**

Purpose: Folium is a Python library for creating interactive maps. It integrates with the Leaflet JavaScript library and allows you to visualize geographical data.

**4. Seaborn:**

Purpose: Seaborn is a data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics.

**import pandas as pd**

**import plotly.express as px**

**import plotly.graph\_objects as go**

**from folium.features import Choropleth**

**import folium**

**from folium.features import Tooltip**

**import seaborn as sns**

**Preprocessing datasets:**

**The code you've provided is related to data analysis using the Pandas library in Python. Let's break down what each line of code does:**

**1. df = pd.read\_csv("/kaggle/input/covid-world-vaccination-progress/country\_vaccinations\_by\_manufacturer.csv"):**

This line reads a CSV file located at the given file path ("/kaggle/input/covid-world-vaccination-progress/country\_vaccinations\_by\_manufacturer.csv") and loads it into a Pandas DataFrame. The DataFrame is stored in the variable df.

**2. df.head(10):**

This line displays the first 10 rows of the DataFrame `df`. It's a way to quickly inspect the data and see what it looks like.

**3. df["location"].nunique():**

This line calculates the number of unique values in the "location" column of the DataFrame `df`. It provides the count of unique countries or locations in the dataset.

**4. df.isnull().sum():**

This line checks for missing (null) values in each column of the DataFrame and returns the sum of missing values for each column. It helps you identify which columns have missing data.

**5. df.dtypes:**

This line returns the data types of each column in the DataFrame. It tells you whether each column contains numerical data, strings (objects), dates, or other data types.

**6. df['date'] = pd.to\_datetime(df['date']):**

This line converts the "date" column in the DataFrame `df` to a datetime data type using the `pd.to\_datetime` function. It's a common step in data analysis to ensure that date columns are treated as dates, which allows for time-based operations.

**A brief explanation of why converting the "date" column to a datetime data type is important:**

When the "date" column is in a datetime format, you can perform date-related operations and analyses, such as filtering data for a specific date range or extracting components like day, month, or year.

**Preforming Different Analysis Code and Explaination:**

**Visualization tasks using the Pandas, Plotly, Seaborn, and Folium libraries. Let's break down what each part of the code does:**

**1. data\_2 = pd.DataFrame(columns=['Country', 'Vaccine']):**

This line creates an empty DataFrame data\_2 with columns 'Country' and 'Vaccine'.

1. **data["Total\_vaccine"] =**

**pd.to\_numeric(data["Total\_vaccine"], errors="coerce"):**

This line converts the "Total\_vaccine" column in the DataFrame data to a numeric data type, coercing non-numeric values to NaN.

**3. data\_2.head():**

Displays the first few rows of the data\_2 DataFrame.

**4. data\_2["Vaccine"].value\_counts().plot(kind="bar", color=["Red", "Gray", "Gray", "Gray"]):**

This code generates a bar plot showing the counts of each unique value in the "Vaccine" column in the data\_2 DataFrame.

**5. number\_of\_days = (df["date"].max() - df["date"].min()).days:**

Calculates the number of days between the maximum and minimum dates in the "date" column of the DataFrame df.

**6.dtfrm["average\_vaccination\_count"].sort\_values(ascending=False).head(10).plot(kind="bar", color=color):**

This code generates a bar plot showing the top 10 countries with the highest "average\_vaccination\_count" values, using custom colors specified in the color variable.

**7. dtfrm.head(10):**

Displays the first 10 rows of the DataFrame dtfrm.

**8. dtfrm.set\_index("Country", inplace=True):**

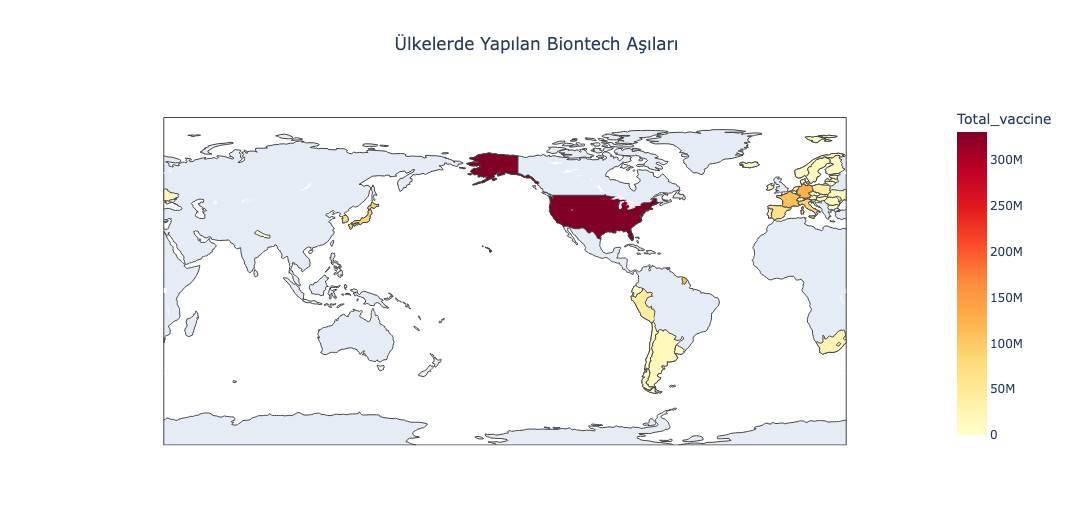
Sets the "Country" column as the index of the DataFrame

dtfrm.

**9. fig.update\_layout(title\_x=0.5):**

Adjusts the layout of the Plotly figure to center the title.

**Output:**

****