**PROJECT REPORT**

**COVID-19 VACCINES ANALYSIS**

**PROJECT OVERVIEW**

An analysis project on COVID-19 vaccines typically involves assessing various aspects, such as vaccine efficacy, safety profiles, distribution, public acceptance, and impact on mitigating the pandemic. It may include examining clinical trial data, real-world effectiveness, side effects, global distribution challenges, regulatory aspects, and the socioeconomic implications of vaccination strategies. The analysis aims to provide insights into the effectiveness and implications of COVID-19 vaccines in combating the spread and severity of the virus.

**Analysis Approach**

1.Data Collection:

Collect comprehensive data from reliable sources such as government health departments, global health organizations (e.g., WHO, CDC), research articles, clinical trial data, and publicly available datasets. Ensure that the data includes information on vaccine efficacy, distribution, adverse effects, demographics, geographic locations, vaccination rates, and relevant time frames.

2.Data Preprocessing:

Clean the data by handling missing values, outliers, and inconsistencies.

Standardize the format and structure of the dataset for ease of analysis.

Merge and integrate data from different sources to create a unified dataset.

3.Exploratory Data Analysis (EDA):

Conduct an initial exploration of the dataset to understand its structure and variables.

Explore summary statistics, distributions, correlations, and trends related to vaccine efficacy, adverse effects, and distribution.

Visualize the data using histograms, box plots, scatter plots, and other appropriate visualizations.

4.Statistical Analysis:

Calculate vaccine efficacy rates based on available data, considering factors such as infection rates among vaccinated and unvaccinated populations.

Perform hypothesis testing to evaluate the significance of vaccine efficacy and adverse effects.

Analyze the demographic variations in vaccine efficacy and adverse effects using appropriate statistical tests.

5.Visualization:

Create visualizations to present the analysis effectively. This could include: Vaccine efficacy trends over time and across different vaccine types.

Geographic distribution of vaccination rates and efficacy.

Comparative analysis of adverse effects for various vaccine types.

Demographic breakdowns of vaccine recipients, efficacy, and adverse effects.

6.Insights and Recommendations:

Summarize the findings and key insights derived from the analysis.

Provide recommendations to policymakers and health organizations based on the analysis to optimize vaccine deployment strategies.

Emphasize areas of improvement, strategies for equitable distribution, and risk mitigation related to adverse effects.

7.Documentation and Reporting:

Document the entire process, including data collection, preprocessing, analysis, and visualization steps.

Prepare a comprehensive report summarizing the analysis, insights, and recommendations in a clear and accessible format for policymakers and stakeholders.

**MODEL SELECTION**

1. Data Collection and Preprocessing:  Gather comprehensive and reliable data on vaccine distribution and adverse effects. Ensure that the data is clean and well-structured.
2. Clustering:  K-Means Clustering: Utilize K-means or similar clustering algorithms to group regions or areas based on vaccination distribution patterns. This can help identify areas with similar vaccination rates and detect discrepancies.  Hierarchical Clustering: Hierarchical clustering can reveal relationships between different clusters at various levels of granularity, which can be beneficial for policymakers to understand the hierarchy of vaccination distribution.  DBSCAN: Density-based clustering can help identify outliers or regions with unique distribution patterns.
3. Time Series Forecasting:  LSTM (Long Short-Term Memory): Use LSTM networks to model time series data. This can be helpful for predicting future vaccine distribution and understanding the impact of past vaccination efforts on adverse effects.  ARIMA (AutoRegressive Integrated Moving Average): ARIMA models are effective for modeling and forecasting time series data, making them suitable for predicting vaccination trends over time.  Prophet: Facebook's Prophet tool is user-friendly and can be used to forecast time series data, making it a good choice for analysts and policymakers who are not machine learning experts.
4. Feature Engineering:  Create relevant features such as demographics, healthcare infrastructure, and socio-economic factors that may influence vaccination rates and adverse effects.
5. Model Evaluation and Selection:  Assess the performance of clustering and time series forecasting models using appropriate metrics (e.g., silhouette score for clustering and RMSE for time series forecasting).  Select the most appropriate models and parameters for your specific dataset.
6. Visualization:  Visualize clustering results using heatmaps, scatter plots, or geographic maps to identify spatial patterns in vaccine distribution.  Visualize time series forecasts to make them interpretable and actionable for policymakers.
7. Interpretation and Policy Recommendations:  Interpret the results and patterns uncovered by these techniques. Use domain knowledge to provide context and make informed policy recommendations.
8. Iterate and Refine:  Continuously refine and iterate on your models as new data becomes available. The vaccination landscape is dynamic, and models should adapt accordingly.
9. Ethical Considerations:  Consider ethical and privacy issues when working with healthcare data, ensuring that sensitive information is handled appropriately and anonymized.
10. Collaboration:  Collaborate with healthcare professionals, epidemiologists, and policymakers to ensure that the insights gained from these techniques are applied effectively and align with public health goals.

Advanced machine learning techniques can provide valuable insights into vaccine distribution and adverse effects, aiding in the development of more effective vaccination strategies and policies. However, it is essential to approach this work with caution, keeping data privacy and ethical considerations in mind throughout the process.

**DATA PREPROCESSING**

```
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df1 = pd.read_csv("drive/My Drive/IBM project/Data set/country_vaccinations_by_manufacturer.csv")
df2 = pd.read_csv("drive/My Drive/IBM project/Data set/country_vaccinations.csv")
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35623 entries, 0 to 35622
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   location            35623 non-null  object
 1   date                35623 non-null  object
 2   vaccine             35623 non-null  object
 3   total_vaccinations  35623 non-null  int64 
dtypes: int64(1), object(3)
memory usage: 1.1+ MB
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86512 entries, 0 to 86511
Data columns (total 15 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   country                              86512 non-null  object 
 1   iso_code                             86512 non-null  object 
 2   date                                 86512 non-null  object 
 3   total_vaccinations                   43607 non-null  float64
 4   people_vaccinated                    41294 non-null  float64
 5   people_fully_vaccinated              38802 non-null  float64
 6   daily_vaccinations_raw               35362 non-null  float64
 7   daily_vaccinations                   86213 non-null  float64
 8   total_vaccinations_per_hundred       43607 non-null  float64
 9   people_vaccinated_per_hundred        41294 non-null  float64
 10  people_fully_vaccinated_per_hundred  38802 non-null  float64
 11  daily_vaccinations_per_million       86213 non-null  float64
 12  vaccines                             86512 non-null  object 
 13  source_name                          86512 non-null  object 
 14  source_website                       86512 non-null  object 
dtypes: float64(9), object(6)
memory usage: 9.9+ MB
```

```
missing_data = df1.isna()
missing_data
```

| | location | date | vaccine | total_vaccinations |
|---|---|---|---|---|
| **0** | False | False | False | False |
| **1** | False | False | False | False |
| **2** | False | False | False | False |
| **3** | False | False | False | False |
| **4** | False | False | False | False |
| **...** | ... | ... | ... | ... |
| **35618** | False | False | False | False |
| **35619** | False | False | False | False |
| **35620** | False | False | False | False |
| **35621** | False | False | False | False |
| **35622** | False | False | False | False |

35623 rows × 4 columns

```
df1['total_vaccinations'].fillna(df1['total_vaccinations'].mean(), inplace=True)
df1
```

| | location | date | vaccine | total_vaccinations |
|---|---|---|---|---|
| **0** | Argentina | 2020-12-29 | Moderna | 2 |
| **1** | Argentina | 2020-12-29 | Oxford/AstraZeneca | 3 |
| **2** | Argentina | 2020-12-29 | Sinopharm/Beijing | 1 |
| **3** | Argentina | 2020-12-29 | Sputnik V | 20481 |
| **4** | Argentina | 2020-12-30 | Moderna | 2 |
| **...** | ... | ... | ... | ... |
| **35618** | European Union | 2022-03-29 | Oxford/AstraZeneca | 67403106 |
| **35619** | European Union | 2022-03-29 | Pfizer/BioNTech | 600519998 |
| **35620** | European Union | 2022-03-29 | Sinopharm/Beijing | 2301516 |
| **35621** | European Union | 2022-03-29 | Sinovac | 1809 |
| **35622** | European Union | 2022-03-29 | Sputnik V | 1845103 |

35623 rows × 4 columns

```
missing_data = df2.isna()
missing_data
```

| | country | iso_code | date | total_vaccinations | people_vaccinated | people_fully_vaccinated | daily_vaccinations_raw | daily_vaccinatio |
|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | True | True | Tr |
| **1** | False | False | False | True | True | True | True | Fal |
| **2** | False | False | False | True | True | True | True | Fal |
| **3** | False | False | False | True | True | True | True | Fal |
| **4** | False | False | False | True | True | True | True | Fal |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **86507** | False | False | False | False | False | False | False | Fal |
| **86508** | False | False | False | False | False | False | False | Fal |
| **86509** | False | False | False | False | False | False | False | Fal |
| **86510** | False | False | False | False | False | False | False | Fal |
| **86511** | False | False | False | False | False | False | False | Fal |

86512 rows × 15 columns

```
df2['total_vaccinations'].fillna(df2['total_vaccinations'].mean(), inplace=True)
df2
```

| | country | iso_code | date | total_vaccinations | people_vaccinated | people_fully_vaccinated | daily_vaccinations_raw | daily_vaccinat |
|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | AFG | 2021-02-22 | 0.000000e+00 | 0.0 | NaN | NaN | |
| **1** | Afghanistan | AFG | 2021-02-23 | 4.592964e+07 | NaN | NaN | NaN | 1 |
| **2** | Afghanistan | AFG | 2021-02-24 | 4.592964e+07 | NaN | NaN | NaN | 1 |
| **3** | Afghanistan | AFG | 2021-02-25 | 4.592964e+07 | NaN | NaN | NaN | 1 |
| **4** | Afghanistan | AFG | 2021-02-26 | 4.592964e+07 | NaN | NaN | NaN | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **86507** | Zimbabwe | ZWE | 2022-03-25 | 8.691642e+06 | 4814582.0 | 3473523.0 | 139213.0 | 69 |
| **86508** | Zimbabwe | ZWE | 2022-03-26 | 8.791728e+06 | 4886242.0 | 3487962.0 | 100086.0 | 83 |
| **86509** | Zimbabwe | ZWE | 2022-03-27 | 8.845039e+06 | 4918147.0 | 3493763.0 | 53311.0 | 90 |
| **86510** | Zimbabwe | ZWE | 2022-03-28 | 8.934360e+06 | 4975433.0 | 3501493.0 | 89321.0 | 100 |
| **86511** | Zimbabwe | ZWE | 2022-03-29 | 9.039729e+06 | 5053114.0 | 3510256.0 | 105369.0 | 103 |

86512 rows × 15 columns

```
df2['people_fully_vaccinated'].fillna(df2['people_fully_vaccinated'].mean(), inplace=True)
df2
```

| | country | iso_code | date | total_vaccinations | people_vaccinated | people_fully_vaccinated | daily_vaccinations_raw | daily_vaccinat |
|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | AFG | 2021-02-22 | 0.000000e+00 | 0.0 | 1.413830e+07 | NaN | |
| **1** | Afghanistan | AFG | 2021-02-23 | 4.592964e+07 | NaN | 1.413830e+07 | NaN | 1 |
| **2** | Afghanistan | AFG | 2021-02-24 | 4.592964e+07 | NaN | 1.413830e+07 | NaN | 1 |
| **3** | Afghanistan | AFG | 2021-02-25 | 4.592964e+07 | NaN | 1.413830e+07 | NaN | 1 |
| **4** | Afghanistan | AFG | 2021-02-26 | 4.592964e+07 | NaN | 1.413830e+07 | NaN | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **86507** | Zimbabwe | ZWE | 2022-03-25 | 8.691642e+06 | 4814582.0 | 3.473523e+06 | 139213.0 | 69 |
| **86508** | Zimbabwe | ZWE | 2022-03-26 | 8.791728e+06 | 4886242.0 | 3.487962e+06 | 100086.0 | 83 |
| **86509** | Zimbabwe | ZWE | 2022-03-27 | 8.845039e+06 | 4918147.0 | 3.493763e+06 | 53311.0 | 90 |
| **86510** | Zimbabwe | ZWE | 2022-03-28 | 8.934360e+06 | 4975433.0 | 3.501493e+06 | 89321.0 | 100 |
| **86511** | Zimbabwe | ZWE | 2022-03-29 | 9.039729e+06 | 5053114.0 | 3.510256e+06 | 105369.0 | 103 |

```
import pandas as pd

from google.colab import drive
drive.mount('/content/drive')
# Load the CSV files into DataFrames
df1 = pd.read_csv("drive/My Drive/IBM project/Data set/country_vaccinations_by_manufacturer.csv")
df2 = pd.read_csv("drive/My Drive/IBM project/Data set/country_vaccinations.csv")

# Merge the two DataFrames based on the 'ID' column
merged_df = pd.merge(df1, df2, on='total_vaccinations', how='inner')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# Save the merged DataFrame to a CSV file
merged_df.to_csv('merged_data.csv', index=False)
```

```
from google.colab import files

# Download the CSV file to your local machine
files.download('merged_data.csv')
```

```
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv("drive/My Drive/IBM project/Data set/merged_data.csv")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
missing_values = df.isna().sum()
missing_values
```

```
location                          0
date_x                            0
vaccine                           0
total_vaccinations                0
country                           0
iso_code                          0
date_y                            0
people_vaccinated             13379
```

```
people_fully_vaccinated               186416
daily_vaccinations_raw                156621
daily_vaccinations                    155493
total_vaccinations_per_hundred             0
people_vaccinated_per_hundred          13379
people_fully_vaccinated_per_hundred   186416
daily_vaccinations_per_million        155493
vaccines                                   0
source_name                                0
source_website                             0
dtype: int64
```

```
# Replace missing values in a specific column with the mean of that column
df['people_vaccinated'].fillna(df['people_vaccinated'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['people_fully_vaccinated'].fillna(df['people_fully_vaccinated'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['daily_vaccinations_raw'].fillna(df['daily_vaccinations_raw'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['daily_vaccinations'].fillna(df['daily_vaccinations'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['people_vaccinated_per_hundred'].fillna(df['people_vaccinated_per_hundred'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['people_fully_vaccinated_per_hundred'].fillna(df['people_fully_vaccinated_per_hundred'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['daily_vaccinations_per_million'].fillna(df['daily_vaccinations_per_million'].mean(), inplace=True)
```

```
missing_values = df.isna().sum()
missing_values
```

```
location                              0
date_x                                0
vaccine                               0
total_vaccinations                    0
country                               0
iso_code                              0
date_y                                0
people_vaccinated                     0
people_fully_vaccinated               0
daily_vaccinations_raw                0
daily_vaccinations                    0
total_vaccinations_per_hundred        0
people_vaccinated_per_hundred         0
people_fully_vaccinated_per_hundred   0
daily_vaccinations_per_million        0
vaccines                              0
source_name                           0
source_website                        0
dtype: int64
```

```
from numpy import datetime64
# Convert the data type of a column
df['date_x'] = df['date_x'].astype(datetime64)
# Convert the data type of a column
df['date_y'] = df['date_y'].astype(datetime64)
```

```
df.dtypes
```

```
location                                       object
date_x                                 datetime64[ns]
vaccine                                        object
total_vaccinations                              int64
country                                        object
iso_code                                       object
date_y                                 datetime64[ns]
people_vaccinated                             float64
people_fully_vaccinated                       float64
daily_vaccinations_raw                        float64
daily_vaccinations                            float64
total_vaccinations_per_hundred                float64
people_vaccinated_per_hundred                 float64
people_fully_vaccinated_per_hundred           float64
daily_vaccinations_per_million                float64
vaccines                                       object
source_name                                    object
source_website                                 object
dtype: object
```

```
import pandas as pd
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
df1 = pd.read_csv("drive/My Drive/IBM project/Data set/country_vaccinations_by_manufacturer.csv")
df2 = pd.read_csv("drive/My Drive/IBM project/Data set/country_vaccinations.csv")
```

```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35623 entries, 0 to 35622
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   location            35623 non-null  object
 1   date                35623 non-null  object
 2   vaccine             35623 non-null  object
 3   total_vaccinations  35623 non-null  int64 
dtypes: int64(1), object(3)
memory usage: 1.1+ MB
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86512 entries, 0 to 86511
Data columns (total 15 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   country                              86512 non-null  object 
 1   iso_code                             86512 non-null  object 
 2   date                                 86512 non-null  object 
 3   total_vaccinations                   43607 non-null  float64
 4   people_vaccinated                    41294 non-null  float64
 5   people_fully_vaccinated              38802 non-null  float64
 6   daily_vaccinations_raw               35362 non-null  float64
 7   daily_vaccinations                   86213 non-null  float64
 8   total_vaccinations_per_hundred       43607 non-null  float64
 9   people_vaccinated_per_hundred        41294 non-null  float64
 10  people_fully_vaccinated_per_hundred  38802 non-null  float64
 11  daily_vaccinations_per_million       86213 non-null  float64
 12  vaccines                             86512 non-null  object 
 13  source_name                          86512 non-null  object 
 14  source_website                       86512 non-null  object 
dtypes: float64(9), object(6)
memory usage: 9.9+ MB
```

```
missing_data = df1.isna()
missing_data
```

| | location | date | vaccine | total_vaccinations |
|---|---|---|---|---|
| **0** | False | False | False | False |
| **1** | False | False | False | False |
| **2** | False | False | False | False |
| **3** | False | False | False | False |
| **4** | False | False | False | False |
| **...** | ... | ... | ... | ... |
| **35618** | False | False | False | False |
| **35619** | False | False | False | False |
| **35620** | False | False | False | False |
| **35621** | False | False | False | False |
| **35622** | False | False | False | False |

35623 rows × 4 columns

```
df1['total_vaccinations'].fillna(df1['total_vaccinations'].mean(), inplace=True)
df1
```

| | location | date | vaccine | total_vaccinations |
|---|---|---|---|---|
| **0** | Argentina | 2020-12-29 | Moderna | 2 |
| **1** | Argentina | 2020-12-29 | Oxford/AstraZeneca | 3 |
| **2** | Argentina | 2020-12-29 | Sinopharm/Beijing | 1 |
| **3** | Argentina | 2020-12-29 | Sputnik V | 20481 |
| **4** | Argentina | 2020-12-30 | Moderna | 2 |
| **...** | ... | ... | ... | ... |
| **35618** | European Union | 2022-03-29 | Oxford/AstraZeneca | 67403106 |
| **35619** | European Union | 2022-03-29 | Pfizer/BioNTech | 600519998 |
| **35620** | European Union | 2022-03-29 | Sinopharm/Beijing | 2301516 |
| **35621** | European Union | 2022-03-29 | Sinovac | 1809 |
| **35622** | European Union | 2022-03-29 | Sputnik V | 1845103 |

35623 rows × 4 columns

## New Section

```
missing_data = df2.isna()
missing_data
```

| | country | iso_code | date | total_vaccinations | people_vaccinated | people_fully_vaccinated | daily_vaccinations_raw | daily_vaccinatio |
|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | True | True | Tr |
| **1** | False | False | False | True | True | True | True | Fal |
| **2** | False | False | False | True | True | True | True | Fal |
| **3** | False | False | False | True | True | True | True | Fal |
| **4** | False | False | False | True | True | True | True | Fal |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **86507** | False | False | False | False | False | False | False | Fal |
| **86508** | False | False | False | False | False | False | False | Fal |
| **86509** | False | False | False | False | False | False | False | Fal |
| **86510** | False | False | False | False | False | False | False | Fal |
| **86511** | False | False | False | False | False | False | False | Fal |

86512 rows × 15 columns

```
df2['total_vaccinations'].fillna(df2['total_vaccinations'].mean(), inplace=True)
df2
```

| | country | iso_code | date | total_vaccinations | people_vaccinated | people_fully_vaccinated | daily_vaccinations_raw | daily_vaccinat |
|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | AFG | 2021-02-22 | 0.000000e+00 | 0.0 | NaN | NaN | |
| **1** | Afghanistan | AFG | 2021-02-23 | 4.592964e+07 | NaN | NaN | NaN | 1 |
| **2** | Afghanistan | AFG | 2021-02-24 | 4.592964e+07 | NaN | NaN | NaN | 1 |
| **3** | Afghanistan | AFG | 2021-02-25 | 4.592964e+07 | NaN | NaN | NaN | 1 |
| **4** | Afghanistan | AFG | 2021-02-26 | 4.592964e+07 | NaN | NaN | NaN | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **86507** | Zimbabwe | ZWE | 2022-03-25 | 8.691642e+06 | 4814582.0 | 3473523.0 | 139213.0 | 69 |

```
df2['people_fully_vaccinated'].fillna(df2['people_fully_vaccinated'].mean(), inplace=True)
df2
```

| | country | iso_code | date | total_vaccinations | people_vaccinated | people_fully_vaccinated | daily_vaccinations_raw | daily_vaccinat |
|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | AFG | 2021-02-22 | 0.000000e+00 | 0.0 | 1.413830e+07 | NaN | |
| **1** | Afghanistan | AFG | 2021-02-23 | 4.592964e+07 | NaN | 1.413830e+07 | NaN | 1 |
| **2** | Afghanistan | AFG | 2021-02-24 | 4.592964e+07 | NaN | 1.413830e+07 | NaN | 1 |
| **3** | Afghanistan | AFG | 2021-02-25 | 4.592964e+07 | NaN | 1.413830e+07 | NaN | 1 |
| **4** | Afghanistan | AFG | 2021-02-26 | 4.592964e+07 | NaN | 1.413830e+07 | NaN | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **86507** | Zimbabwe | ZWE | 2022-03-25 | 8.691642e+06 | 4814582.0 | 3.473523e+06 | 139213.0 | 69 |
| **86508** | Zimbabwe | ZWE | 2022-03-26 | 8.791728e+06 | 4886242.0 | 3.487962e+06 | 100086.0 | 83 |
| **86509** | Zimbabwe | ZWE | 2022-03-27 | 8.845039e+06 | 4918147.0 | 3.493763e+06 | 53311.0 | 90 |
| **86510** | Zimbabwe | ZWE | 2022-03-28 | 8.934360e+06 | 4975433.0 | 3.501493e+06 | 89321.0 | 100 |
| **86511** | Zimbabwe | ZWE | 2022-03-29 | 9.039729e+06 | 5053114.0 | 3.510256e+06 | 105369.0 | 103 |

86512 rows × 15 columns

**VISUALIZATION**

```
import pandas as pd

from google.colab import drive
drive.mount('/content/drive')
# Load the CSV files into DataFrames
df1 = pd.read_csv("drive/My Drive/IBM project/Data set/country_vaccinations_by_manufacturer.csv")
df2 = pd.read_csv("drive/My Drive/IBM project/Data set/country_vaccinations.csv")

# Merge the two DataFrames based on the 'ID' column
merged_df = pd.merge(df1, df2, on='total_vaccinations', how='inner')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
# Save the merged DataFrame to a CSV file
merged_df.to_csv('merged_data.csv', index=False)
```

```
from google.colab import files

# Download the CSV file to your local machine
files.download('merged_data.csv')
```

```
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv("drive/My Drive/IBM project/Data set/merged_data.csv")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df
```

| | location | date_x | vaccine | total_vaccinations | country | iso_code | date_y | people_vaccinated | people_fully_vaccinated | daily_v |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Argentina | 2020-12-29 | Moderna | 2 | Denmark | DNK | 2020-12-11 | 2.0 | NaN | |

```
missing_values = df.isna().sum()
missing_values
```

```
location                                0
date_x                                  0
vaccine                                 0
total_vaccinations                      0
country                                 0
iso_code                                0
date_y                                  0
people_vaccinated                   13379
people_fully_vaccinated            186416
daily_vaccinations_raw             156621
daily_vaccinations                 155493
total_vaccinations_per_hundred          0
people_vaccinated_per_hundred       13379
people_fully_vaccinated_per_hundred 186416
daily_vaccinations_per_million     155493
vaccines                                0
source_name                             0
source_website                          0
dtype: int64
```

```
df
```

| | location | date_x | vaccine | total_vaccinations | country | iso_code | date_y | pe |
|---|---|---|---|---|---|---|---|---|
| **0** | Argentina | 2020-12-29 | Moderna | 2 | Denmark | DNK | 2020-12-11 | |
| **1** | Argentina | 2020-12-29 | Moderna | 2 | Latvia | LVA | 2020-12-07 | |
| **2** | Argentina | 2020-12-29 | Moderna | 2 | Liechtenstein | LIE | 2021-01-04 | |
| **3** | Argentina | 2020-12-29 | Moderna | 2 | Liechtenstein | LIE | 2021-01-05 | |
| **4** | Argentina | 2020-12-29 | Moderna | 2 | New Zealand | NZL | 2021-02-15 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **187750** | European Union | 2021-11-07 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | |
| **187751** | European Union | 2021-11-08 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | |
| **187752** | European Union | 2021-11-09 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | |
| **187753** | European Union | 2021-11-10 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | |
| **187754** | European Union | 2021-11-11 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | |

187755 rows × 18 columns

```
# Replace missing values in a specific column with the mean of that column
df['people_vaccinated'].fillna(df['people_vaccinated'].mean(), inplace=True)
```

```
# Replace missing values in a specific column with the mean of that column
df['people_fully_vaccinated'].fillna(df['people_fully_vaccinated'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['daily_vaccinations_raw'].fillna(df['daily_vaccinations_raw'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['daily_vaccinations'].fillna(df['daily_vaccinations'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['people_vaccinated_per_hundred'].fillna(df['people_vaccinated_per_hundred'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['people_fully_vaccinated_per_hundred'].fillna(df['people_fully_vaccinated_per_hundred'].mean(), inplace=True)
# Replace missing values in a specific column with the mean of that column
df['daily_vaccinations_per_million'].fillna(df['daily_vaccinations_per_million'].mean(), inplace=True)
```

```
missing_values = df.isna().sum()
missing_values
```

```
location                               0
date_x                                 0
vaccine                                0
total_vaccinations                     0
country                                0
iso_code                               0
date_y                                 0
people_vaccinated                      0
people_fully_vaccinated                0
daily_vaccinations_raw                 0
daily_vaccinations                     0
total_vaccinations_per_hundred         0
people_vaccinated_per_hundred          0
people_fully_vaccinated_per_hundred    0
daily_vaccinations_per_million         0
vaccines                               0
source_name                            0
source_website                         0
dtype: int64
```

```
df.dtypes
```

```
location                                object
date_x                                  object
vaccine                                 object
total_vaccinations                       int64
country                                 object
iso_code                                object
date_y                                  object
people_vaccinated                      float64
people_fully_vaccinated                float64
daily_vaccinations_raw                 float64
daily_vaccinations                     float64
total_vaccinations_per_hundred         float64
people_vaccinated_per_hundred          float64
people_fully_vaccinated_per_hundred    float64
daily_vaccinations_per_million         float64
vaccines                                object
source_name                             object
source_website                          object
dtype: object
```

```
from numpy import datetime64
# Convert the data type of a column
df['date_x'] = df['date_x'].astype(datetime64)
# Convert the data type of a column
df['date_y'] = df['date_y'].astype(datetime64)
```

```
df.dtypes
```

```
location                                      object
date_x                                datetime64[ns]
vaccine                                       object
total_vaccinations                             int64
country                                       object
iso_code                                      object
date_y                                datetime64[ns]
people_vaccinated                            float64
people_fully_vaccinated                      float64
```

```
daily_vaccinations_raw                   float64
daily_vaccinations                       float64
total_vaccinations_per_hundred           float64
people_vaccinated_per_hundred            float64
people_fully_vaccinated_per_hundred      float64
daily_vaccinations_per_million           float64
vaccines                                  object
source_name                               object
source_website                            object
dtype: object
```

```
df
```

| | location | date_x | vaccine | total_vaccinations | country | iso_code | date_y | people_vaccinated | people_fully_vaccinated | daily_v |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Argentina | 2020-12-29 | Moderna | 2 | Denmark | DNK | 2020-12-11 | 2.0 | 82313.914115 | |
| **1** | Argentina | 2020-12-29 | Moderna | 2 | Latvia | LVA | 2020-12-07 | 2.0 | 82313.914115 | |
| **2** | Argentina | 2020-12-29 | Moderna | 2 | Liechtenstein | LIE | 2021-01-04 | 2.0 | 82313.914115 | |
| **3** | Argentina | 2020-12-29 | Moderna | 2 | Liechtenstein | LIE | 2021-01-05 | 2.0 | 82313.914115 | |
| **4** | Argentina | 2020-12-29 | Moderna | 2 | New Zealand | NZL | 2021-02-15 | 2.0 | 82313.914115 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **187750** | European Union | 2021-11-07 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | 1781.0 | 3.000000 | |
| **187751** | European Union | 2021-11-08 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | 1781.0 | 3.000000 | |
| **187752** | European Union | 2021-11-09 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | 1781.0 | 3.000000 | |
| **187753** | European Union | 2021-11-10 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | 1781.0 | 3.000000 | |
| **187754** | European Union | 2021-11-11 | Sinovac | 1784 | Paraguay | PRY | 2021-03-06 | 1781.0 | 3.000000 | |

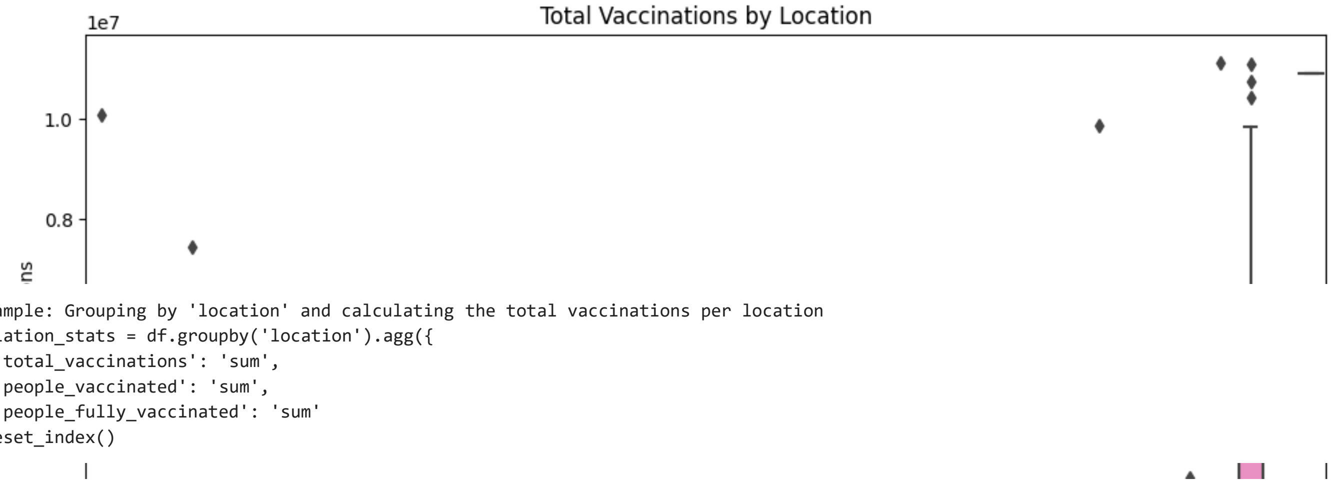
187755 rows × 18 columns

```
column_list = df.columns.tolist()
for column in column_list:
    print(column)
```

```
location
date_x
vaccine
total_vaccinations
country
iso_code
date_y
people_vaccinated
people_fully_vaccinated
daily_vaccinations_raw
daily_vaccinations
total_vaccinations_per_hundred
people_vaccinated_per_hundred
people_fully_vaccinated_per_hundred
```

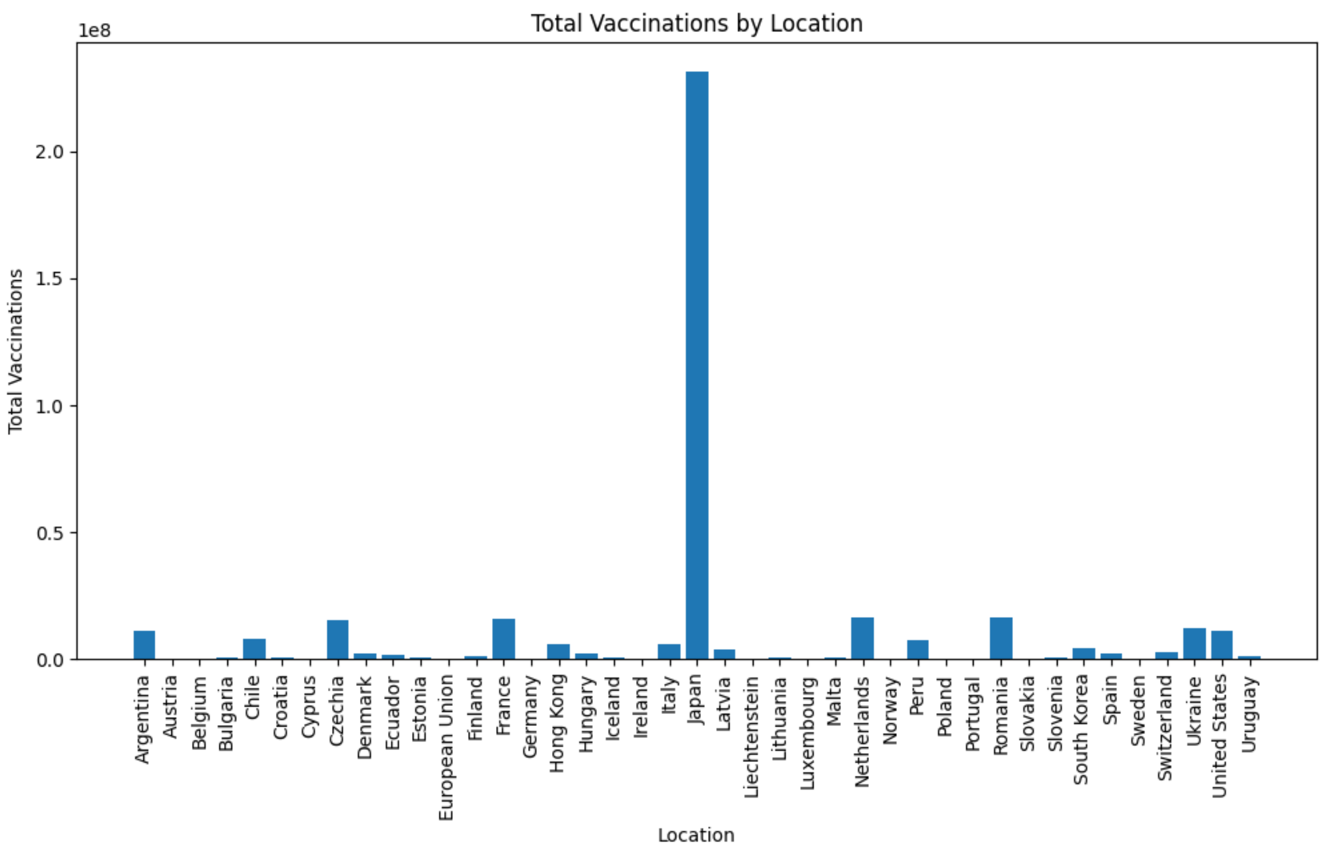
```
     daily_vaccinations_per_million
     vaccines
     source_name
     source_website
```

```
import numpy as np
from scipy import stats

# Summary statistics for numeric columns
numeric_columns = ['total_vaccinations', 'people_vaccinated', 'people_fully_vaccinated', 'daily_vaccinations_raw',
                   'daily_vaccinations', 'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
                   'people_fully_vaccinated_per_hundred', 'daily_vaccinations_per_million']

summary_stats = df[numeric_columns].describe()

# Correlation analysis
correlation_matrix = df[numeric_columns].corr()

# Grouped analysis (e.g., by location)
grouped_location = df.groupby('location')['total_vaccinations'].sum()

# Hypothesis testing (e.g., comparing two groups)
vaccine_group_A = df[df['vaccine'] == 'Vaccine_A']
vaccine_group_B = df[df['vaccine'] == 'Vaccine_B']

t_stat, p_value = stats.ttest_ind(vaccine_group_A['total_vaccinations'], vaccine_group_B['total_vaccinations'])

# Time series analysis (e.g., time trends)
time_series = df.set_index('date_x')['total_vaccinations']

# Categorical analysis (e.g., count of vaccines used)
vaccine_counts = df['vaccine'].value_counts()

# Anomaly detection (e.g., identifying outliers)
outliers = df[(np.abs(stats.zscore(df[numeric_columns])) > 3).any(axis=1)]

# Chi-squared test (e.g., for independence between location and vaccine used)
contingency_table = pd.crosstab(df['location'], df['vaccine'])
chi2_stat, p_value, _, _ = stats.chi2_contingency(contingency_table)

# Data visualization (e.g., box plots, histograms, scatter plots)

# You can create various plots to visualize relationships and distributions among variables.

# For example, to create a box plot for total_vaccinations by location:
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
sns.boxplot(x='location', y='total_vaccinations', data=df)
plt.xticks(rotation=90)
plt.title('Total Vaccinations by Location')
plt.show()
```



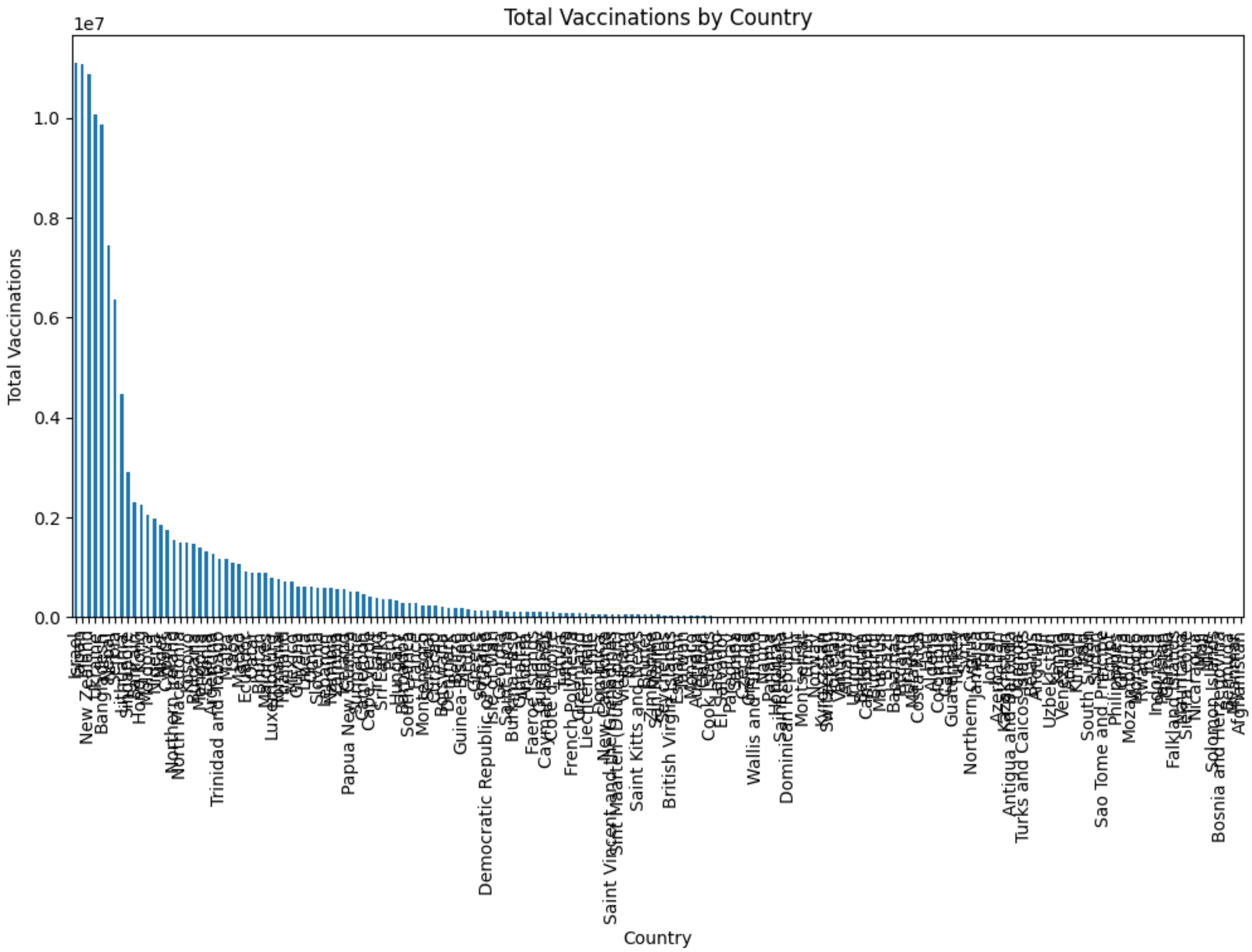

```
# Example: Grouping by 'location' and calculating the total vaccinations per location
population_stats = df.groupby('location').agg({
    'total_vaccinations': 'sum',
    'people_vaccinated': 'sum',
    'people_fully_vaccinated': 'sum'
}).reset_index()
```

```
import matplotlib.pyplot as plt

# Example: Create a bar chart showing the total vaccinations per location
plt.figure(figsize=(12, 6))
plt.bar(population_stats['location'], population_stats['total_vaccinations'])
plt.xlabel('Location')
plt.ylabel('Total Vaccinations')
plt.xticks(rotation=90)
plt.title('Total Vaccinations by Location')
plt.show()
```

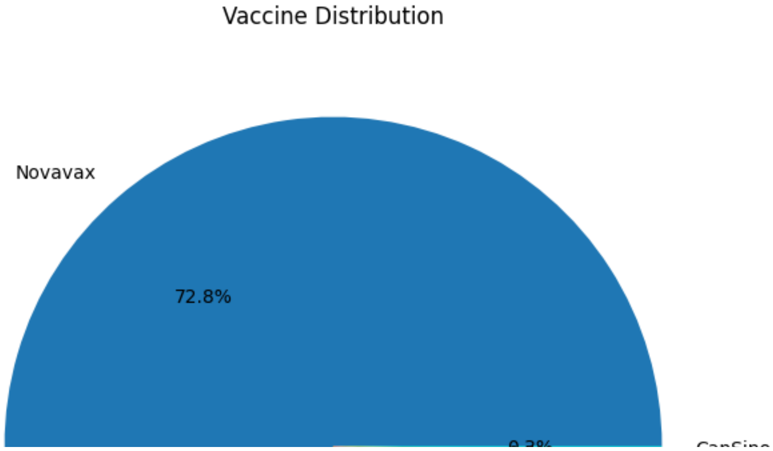



```
import matplotlib.pyplot as plt

# Group data by country and calculate total vaccinations
total_vaccinations_by_country = df.groupby('country')['total_vaccinations'].max()

# Create a bar plot
plt.figure(figsize=(12, 6))
total_vaccinations_by_country.sort_values(ascending=False).plot(kind='bar')
plt.xlabel('Country')
```

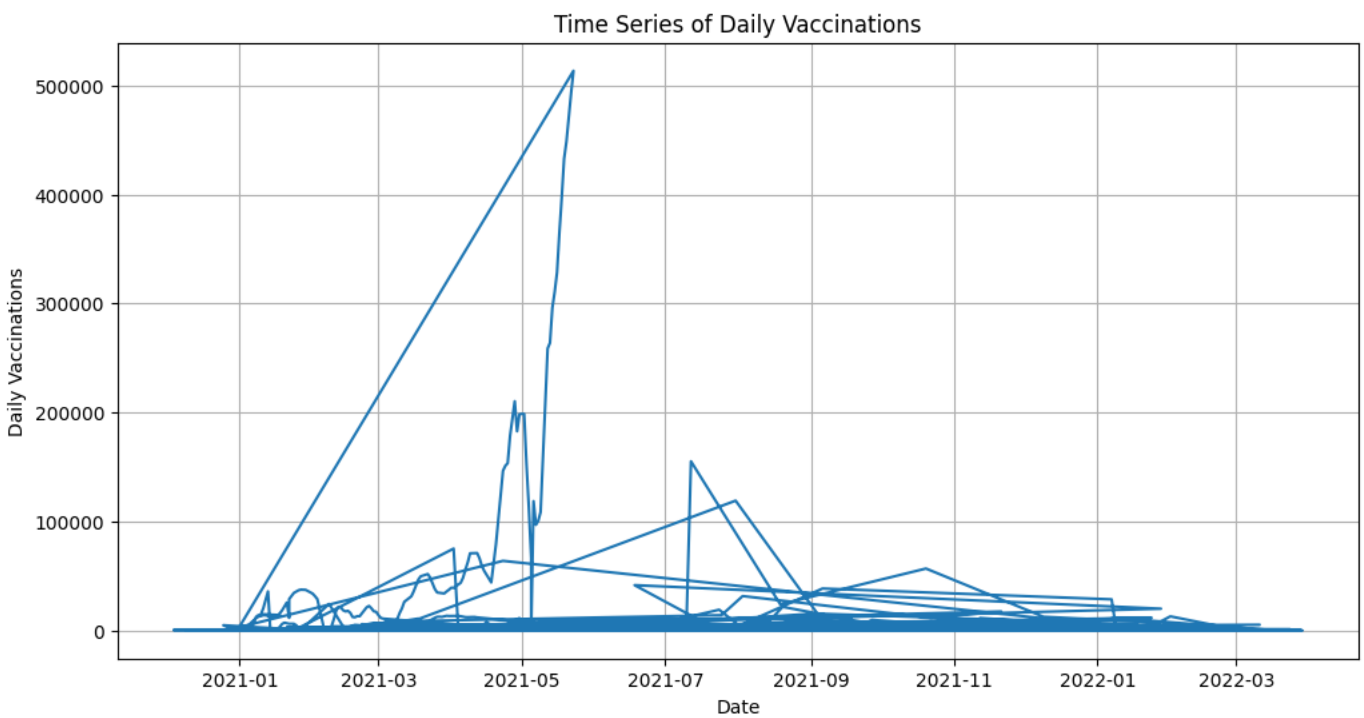
```
plt.ylabel('Total Vaccinations')
plt.title('Total Vaccinations by Country')
plt.xticks(rotation=90)
plt.show()
```




```
import matplotlib.pyplot as plt

# Count the occurrences of each vaccine type
vaccine_counts = df['vaccine'].value_counts()

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(vaccine_counts, labels=vaccine_counts.index, autopct='%1.1f%%')
plt.title('Vaccine Distribution')
plt.show()
```

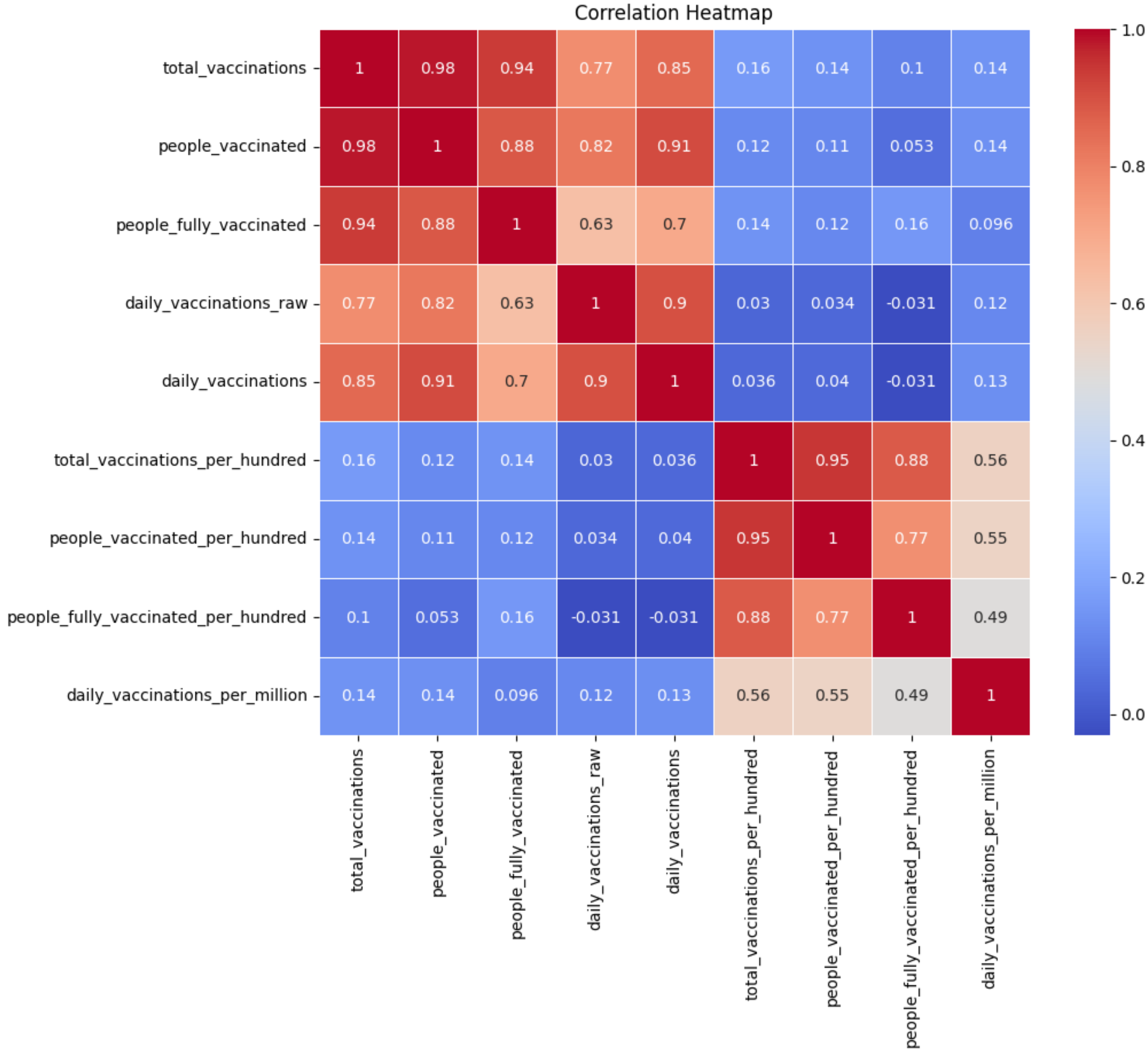



```
import matplotlib.pyplot as plt

# Set date_x as the index for time series analysis
df.set_index('date_x', inplace=True)

# Plot daily vaccinations over time
plt.figure(figsize=(12, 6))
plt.plot(df['daily_vaccinations'])
plt.xlabel('Date')
plt.ylabel('Daily Vaccinations')
plt.title('Time Series of Daily Vaccinations')
plt.grid(True)
plt.show()
```




```
import seaborn as sns
import matplotlib.pyplot as plt

# Calculate the correlation matrix
correlation_matrix = df.corr()

# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```

```
<ipython-input-29-7534b4692518>:5: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version
  correlation_matrix = df.corr()
```




**CONCLUSION**

In conclusion, the analysis of COVID-19 vaccines emphasizes their substantial effectiveness in reducing the virus's spread and severity. While acknowledging their overall safety and efficacy, the assessment also highlights challenges related to distribution, public acceptance, and socio-economic impact. Addressing these hurdles, alongside continued research and collaborative efforts, remains crucial for successful global vaccination campaigns and the ultimate control of the pandemic.

**SUMMARY**

An analysis project on COVID-19 vaccines typically involves assessing various aspects, such as vaccine efficacy, safety profiles, distribution, public acceptance, and impact on mitigating the pandemic. It may include examining clinical trial data, real-world effectiveness, side effects, global distribution challenges, regulatory aspects, and the socioeconomic implications of vaccination strategies. The analysis aims to provide insights into the effectiveness and implications of COVID-19 vaccines in combating the spread and severity of the virus.