Covid-19 Vaccine analysis

01/11/2023

IBM

NAAN MUDHALVAN

PHASE-5

Vaccine Analysis:

In-depth analysis of COVID-19 vaccine data offers critical insights to inform policymakers and health organizations in optimizing vaccine deployment strategies. The study focuses on three key aspects: vaccine efficacy, distribution, and adverse effects. Data from authoritative sources are collected, preprocessed, and subjected to extensive exploration, statistical analysis, and visualization. The analysis begins by defining clear objectives, leading to the examination of vaccine efficacy through clinical trial data, which is complemented by an assessment of vaccine distribution patterns across geographical regions and demographic groups. Additionally, the study investigates reported adverse effects, emphasizing their prevalence and severity across different vaccines. Through exploratory data analysis, statistical testing, and insightful visualizations, this research uncovers nuanced trends and patterns. The findings enable evidence-based decision-making for policymakers, highlighting the strengths and weaknesses of various vaccination strategies. Ultimately, this comprehensive analysis contributes to the ongoing global effort to combat COVID-19, guiding the effective allocation of resources and vaccines, ensuring equitable distribution, and enhancing public confidence in vaccination programs.

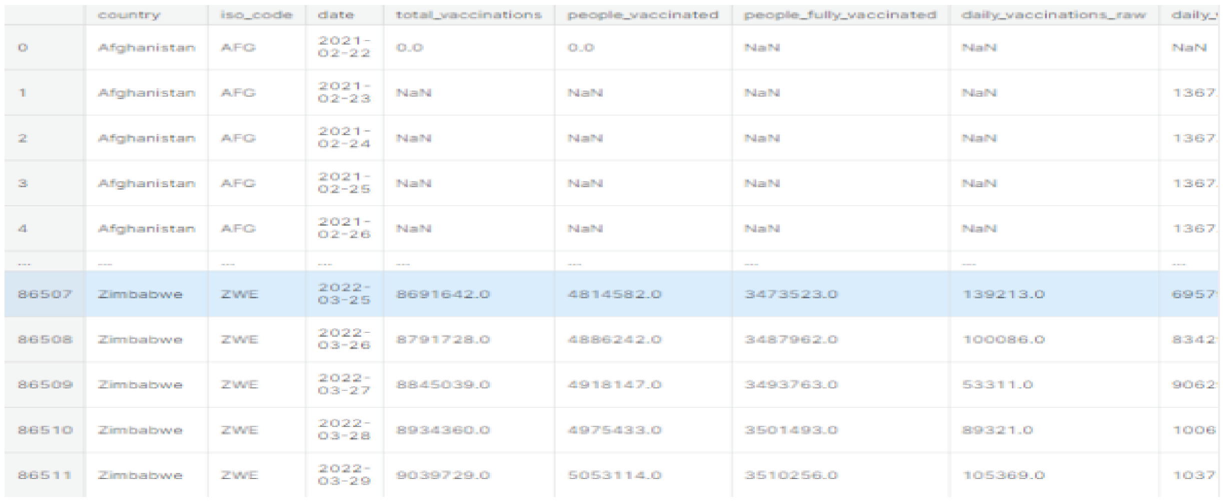
STEP-1 SETUP JUPITER NOTEBOOK, MYSQL, SPREATSHEET:

- For installing jupiter notebook, by running the following command in command prompt “pip install notebook”.
- For installing SQL, by browsing mysql.com and download the latest version of MYSQL workbench.
- For Verifying the intallation complition by running the following code.

```
mysql --version

notebook --version
```

STEP-2 PREPROCESSING THE DATA:

- Data Preprocessing: On the other hand, is the vital step where raw vaccine-related data is transformed and cleaned. This process encompasses data cleaning, transformation, and feature engineering to ensure that the data is suitable for further analysis
- Missing Data: Another common issue that we face in real-world data is the absence of data points. Most machine learning models can’t handle missing values in the data, so you need to intervene and adjust the data to be properly used inside the model.
- For Given dataset

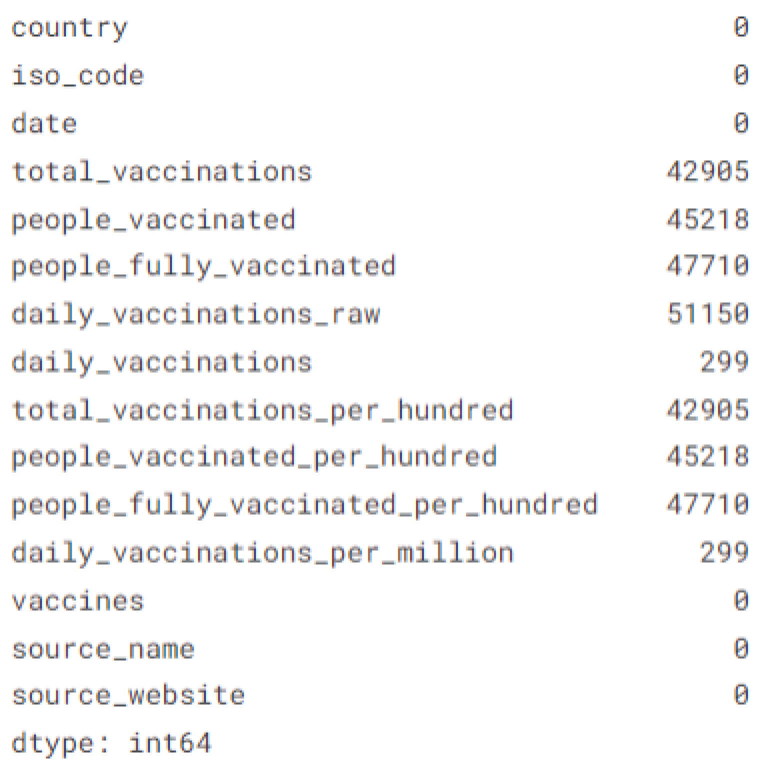


	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	NaN	NaN
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	NaN	1367
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	NaN	1367
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN	NaN	1367
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN	NaN	1367
...	...	...	...	...	...	...	...	...
86507	Zimbabwe	ZWE	2022-03-25	8691642.0	4814582.0	3473523.0	139213.0	6957
86508	Zimbabwe	ZWE	2022-03-26	8791728.0	4886242.0	3487962.0	100086.0	8342
86509	Zimbabwe	ZWE	2022-03-27	8845039.0	4918147.0	3493763.0	53311.0	9062
86510	Zimbabwe	ZWE	2022-03-28	8934360.0	4975433.0	3501493.0	89321.0	1006
86511	Zimbabwe	ZWE	2022-03-29	9039729.0	5053114.0	3510256.0	105369.0	1037

- Loading the dataset: Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
- Identify the dataset: The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service. Load the Dataset: Load your dataset into a Pandas DataFrame. The quality and reliability of data can significantly impact the outcomes of vaccine analysis, making it imperative to have robust data loading procedures in place.

- Program:
- vaccine_df = pd.read_csv('/content/drive/MyDrive/country_vaccinations.csv')
- vaccine_df
- Exploring data: Perform EDA to understand your data better. This includes checking for missing values, exploring the data's statistics, and visualizing it to identify patterns.
- Program: vaccine_df.isnull().sum() # Check for missing values.

Output

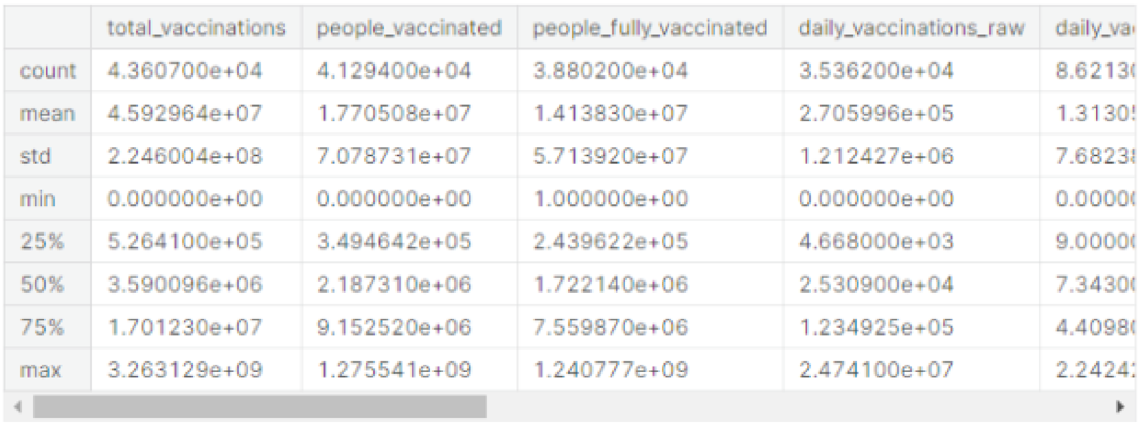


```
country                                   0
iso_code                                  0
date                                      0
total_vaccinations                    42905
people_vaccinated                     45218
people_fully_vaccinated               47710
daily_vaccinations_raw                51150
daily_vaccinations                      299
total_vaccinations_per_hundred        42905
people_vaccinated_per_hundred         45218
people_fully_vaccinated_per_hundred   47710
daily_vaccinations_per_million          299
vaccines                                  0
source_name                               0
source_website                            0
dtype: int64
```

Explore statistics

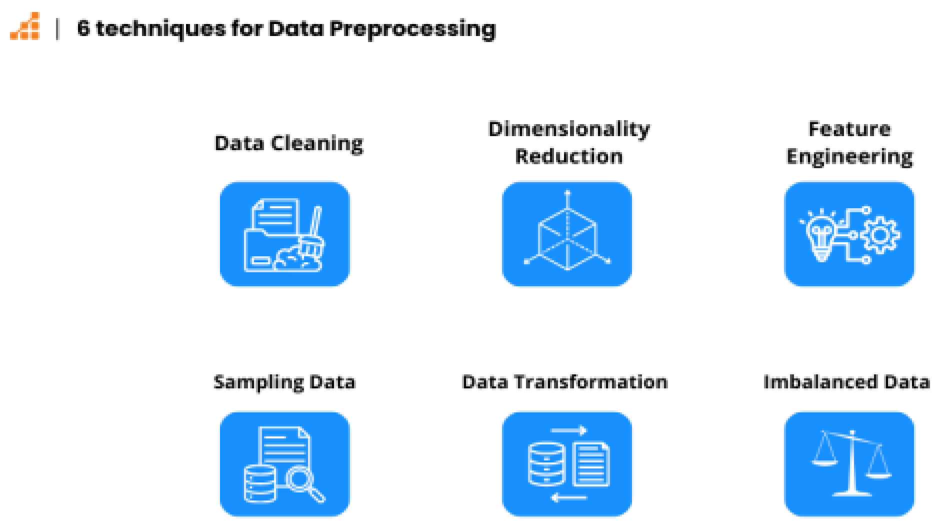
vaccine_df.describe()

Output



	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_va
count	4.360700e+04	4.129400e+04	3.880200e+04	3.536200e+04	8.6213
mean	4.592964e+07	1.770508e+07	1.413830e+07	2.705996e+05	1.3130
std	2.246004e+08	7.078731e+07	5.713920e+07	1.212427e+06	7.6823
min	0.000000e+00	0.000000e+00	1.000000e+00	0.000000e+00	0.0000
25%	5.264100e+05	3.494642e+05	2.439622e+05	4.668000e+03	9.0000
50%	3.590096e+06	2.187310e+06	1.722140e+06	2.530900e+04	7.3430
75%	1.701230e+07	9.152520e+06	7.559870e+06	1.234925e+05	4.4098
max	3.263129e+09	1.275541e+09	1.240777e+09	2.474100e+07	2.2424

- Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model. This may involve cleaning the data, transforming the data into a suitable format.

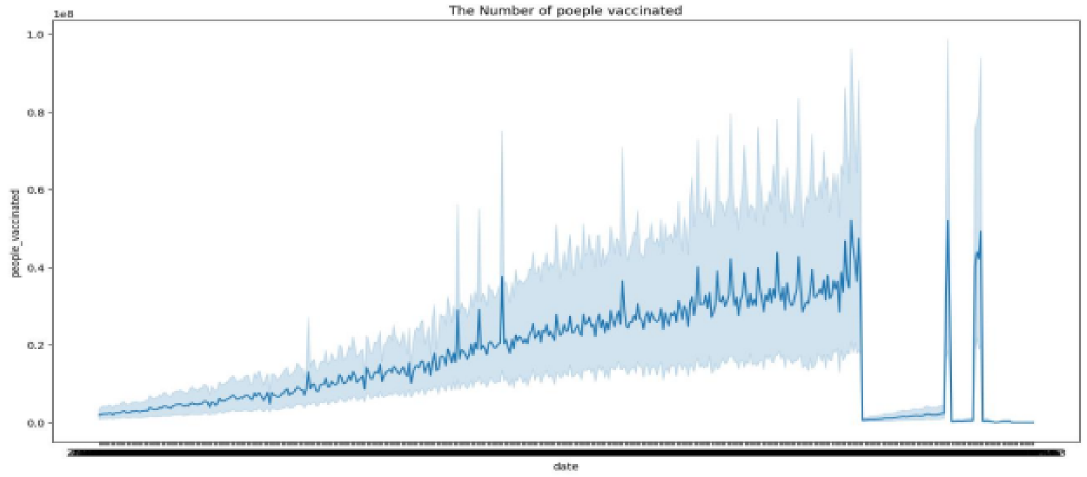


STEP-3 DATA CLEANING:

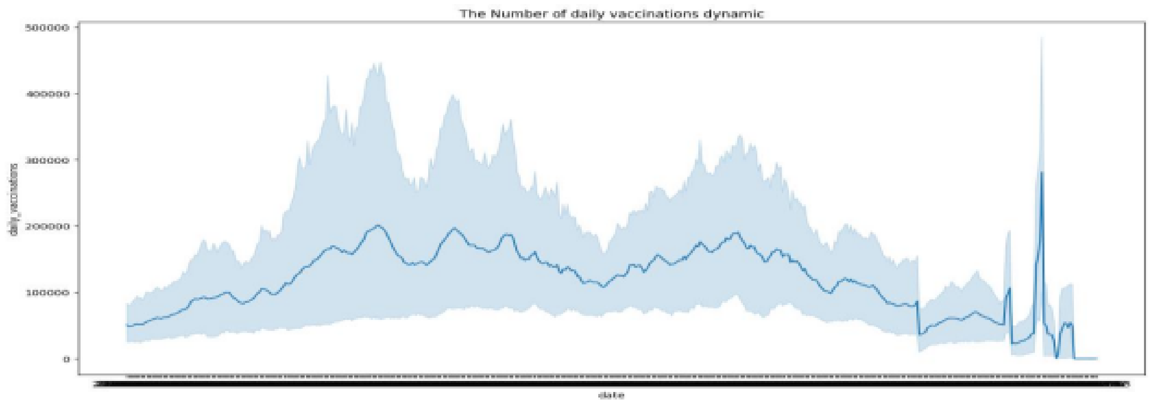
- This involves identifying and correcting errors and inconsistencies in the data. For example, this may involve removing duplicate records, correcting typos, and filling in missing values.
- Feature Scaling: Normalize or standardize numerical features to bring them to a common scale. Common methods include Min-Max scaling (scaling features to a specific range) and z-score normalization (scaling features to have a mean of 0 and a standard deviation of 1).
- Feature Engineering: Create new features or modify existing ones to capture more meaningful information from the data. This may involve mathematical transformations, interaction terms, or aggregations.
- Data transformation: It is a critical aspect of data preprocessing that involves converting and modifying the data to make it more suitable for analysis. It can help improve the performance of machine learning models, enhance the interpretability of the data, and ensure that it aligns with the assumptions of certain statistical techniques.

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(16,8))

sns.lineplot(x=vaccine_df.date, y=vaccine_df.people_vaccinated)

plt.title('The Number of poeple vaccinated')

plt.show()
```

Output

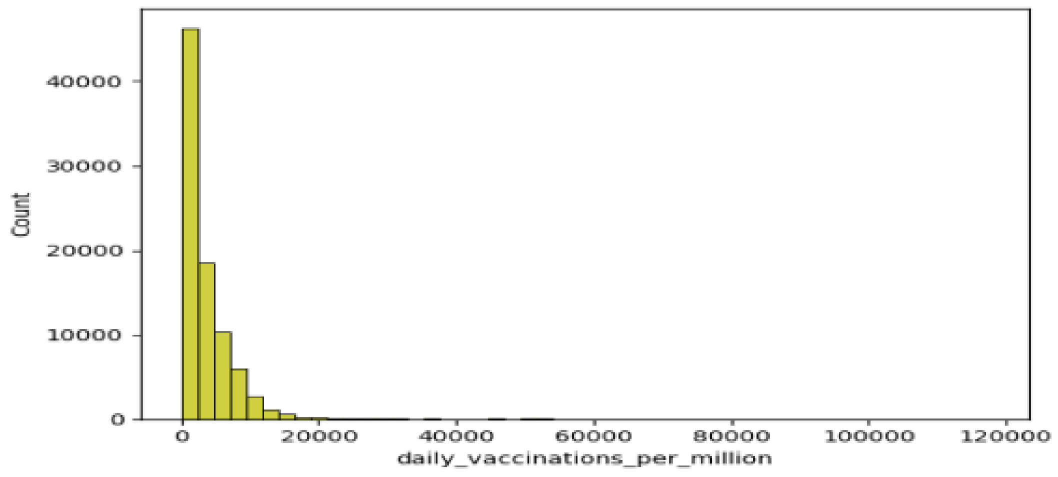


```
plt.figure(figsize=(16,8))
sns.lineplot(x=vaccine_df.date, y=vaccine_df.daily_vaccinations)
plt.title('The Number of daily vaccinations dynamic')
plt.show()
```



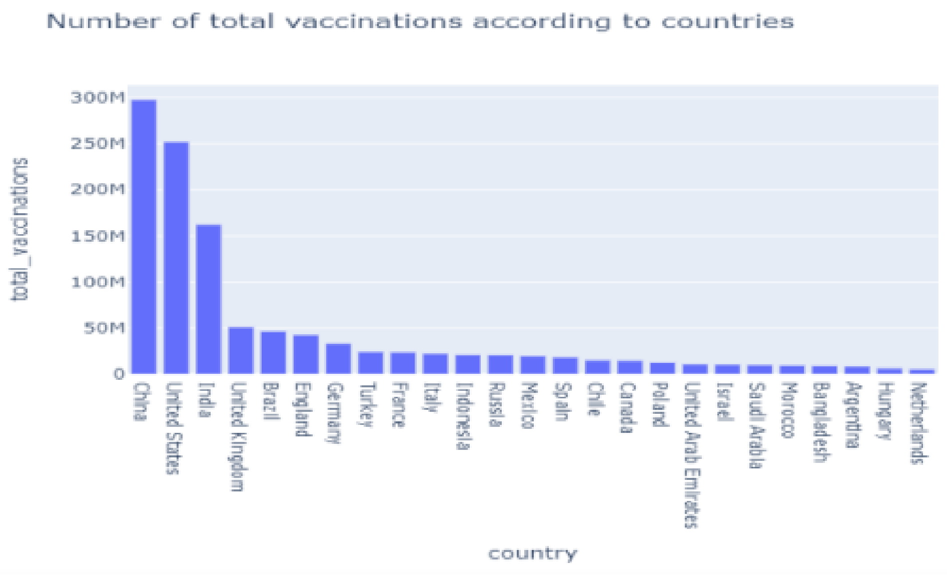
```
sns.histplot(vaccine_df, x='daily_vaccinations_per_million', bins=50,
color='y')
```

Output



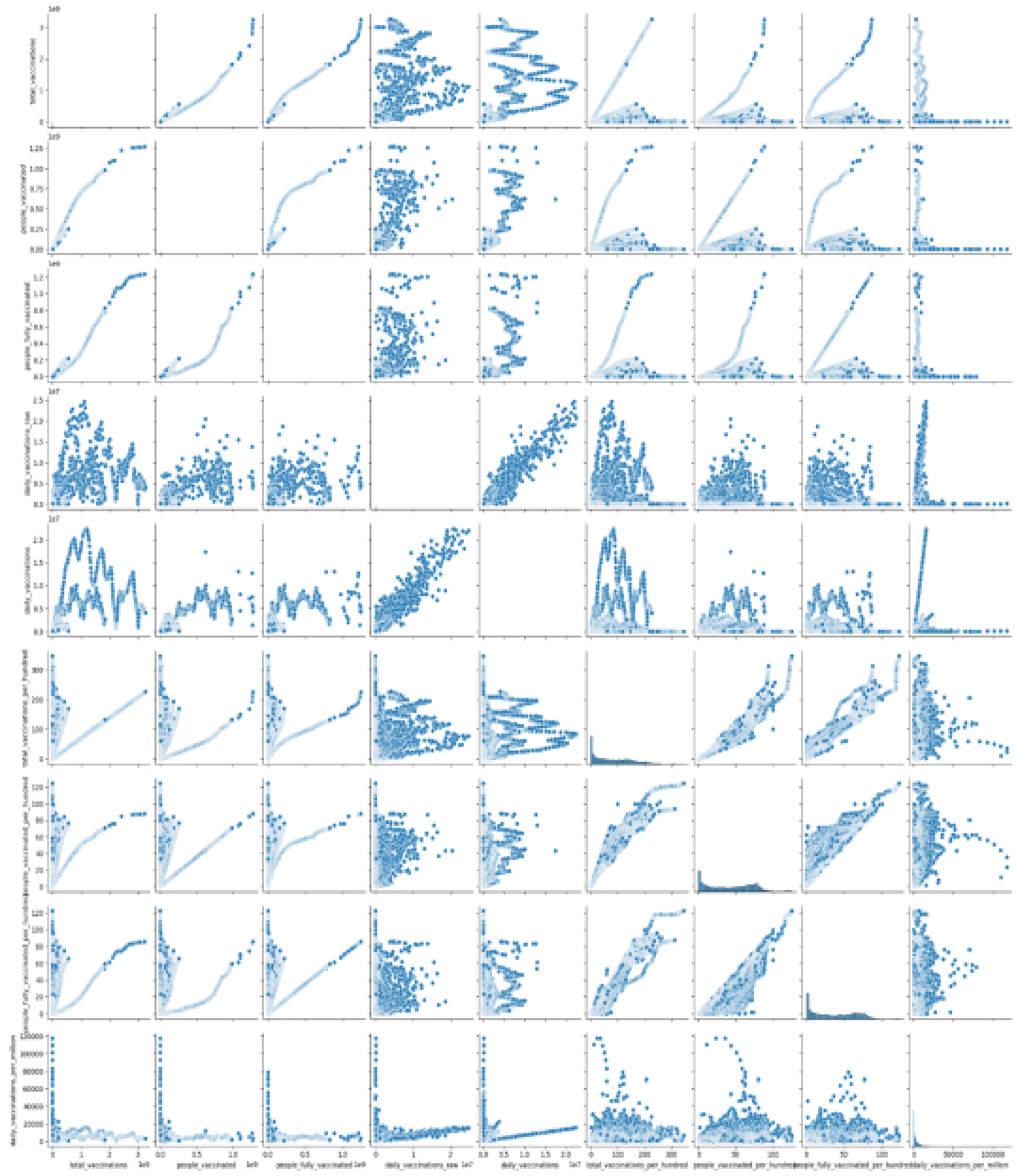
```
data =
vaccine_df[['country','total_vaccinations']].nlargest(25,'total_vaccinations')
 fig = px.bar(data, x = 'country',y = 'total_vaccinations',title="Number of total
vaccinations according to countries",)
fig.show()
```

Output

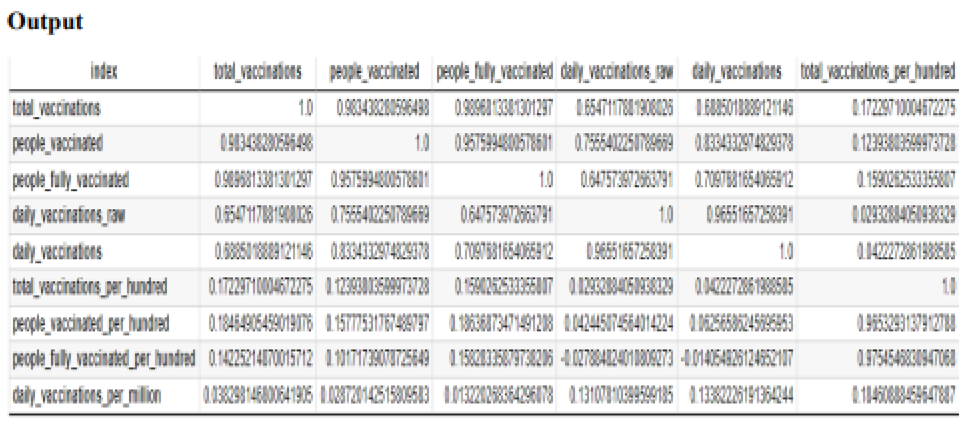


```
plt.figure(figsize=(12,8))

sns.pairplot(vaccine_df)
```

OUTPUT:



vaccine_df.corr(numeric_only=True)



Output

index	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred
total_vaccinations	1.0	0.983438280596498	0.9896813381301297	0.6547117881908026	0.6885018889121146	0.17229710004672275
people_vaccinated	0.983438280596498	1.0	0.9575994800578601	0.7555402250789669	0.8334332974829378	0.12393803599973728
people_fully_vaccinated	0.9896813381301297	0.9575994800578601	1.0	0.6475739726963791	0.7097881654065912	0.1590262533355807
daily_vaccinations_raw	0.6547117881908026	0.7555402250789669	0.6475739726963791	1.0	0.96551657258391	0.02932884050938329
daily_vaccinations	0.6885018889121146	0.8334332974829378	0.7097881654065912	0.96551657258391	1.0	0.0422272861988585
total_vaccinations_per_hundred	0.17229710004672275	0.12393803599973728	0.1590262533355807	0.02932884050938329	0.0422272861988585	1.0
people_vaccinated_per_hundred	0.18464905459019076	0.15777531767489797	0.18636873471491208	0.042445074564014224	0.06256586245695953	0.9653293137912788
people_fully_vaccinated_per_hundred	0.14225214870015712	0.10171739078725649	0.15828335879738206	-0.027884824010808273	-0.014054826124652107	0.9754546830947068
daily_vaccinations_per_million	0.038298146800641905	0.028720142515809583	0.013220268364296078	0.13107810399599185	0.13382226191364244	0.18460888459647887

Predictions for confirmed coronavirus cases worldwide

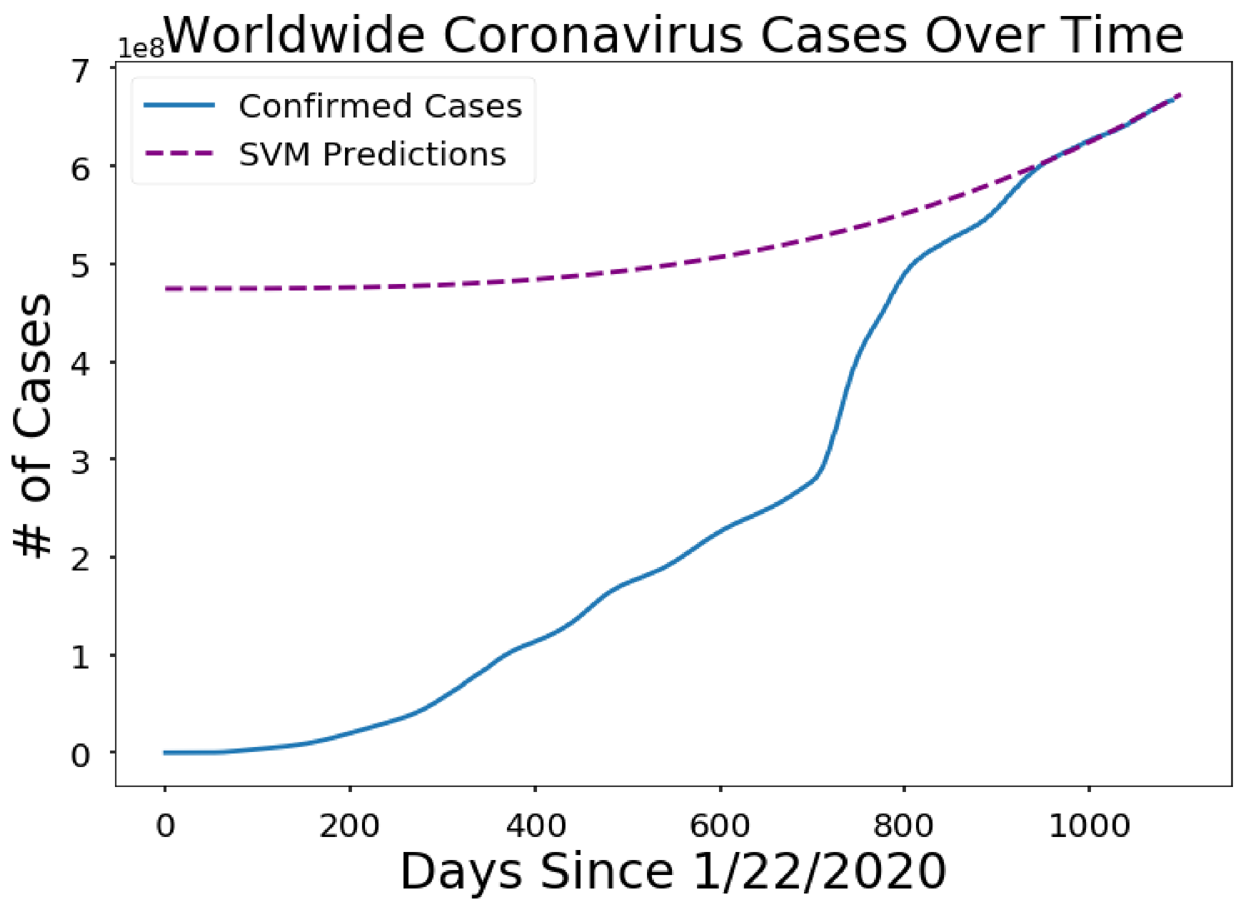
These three models predict future covid-19 cases on a global level. These are constructed to use the latest window of data to predict the current trend. The start date for the prediction data is 10/1/2022 to the current date.

The prediction models include

- Support Vector Machine
- Polynomial Regression
- Bayesian Ridge Regression

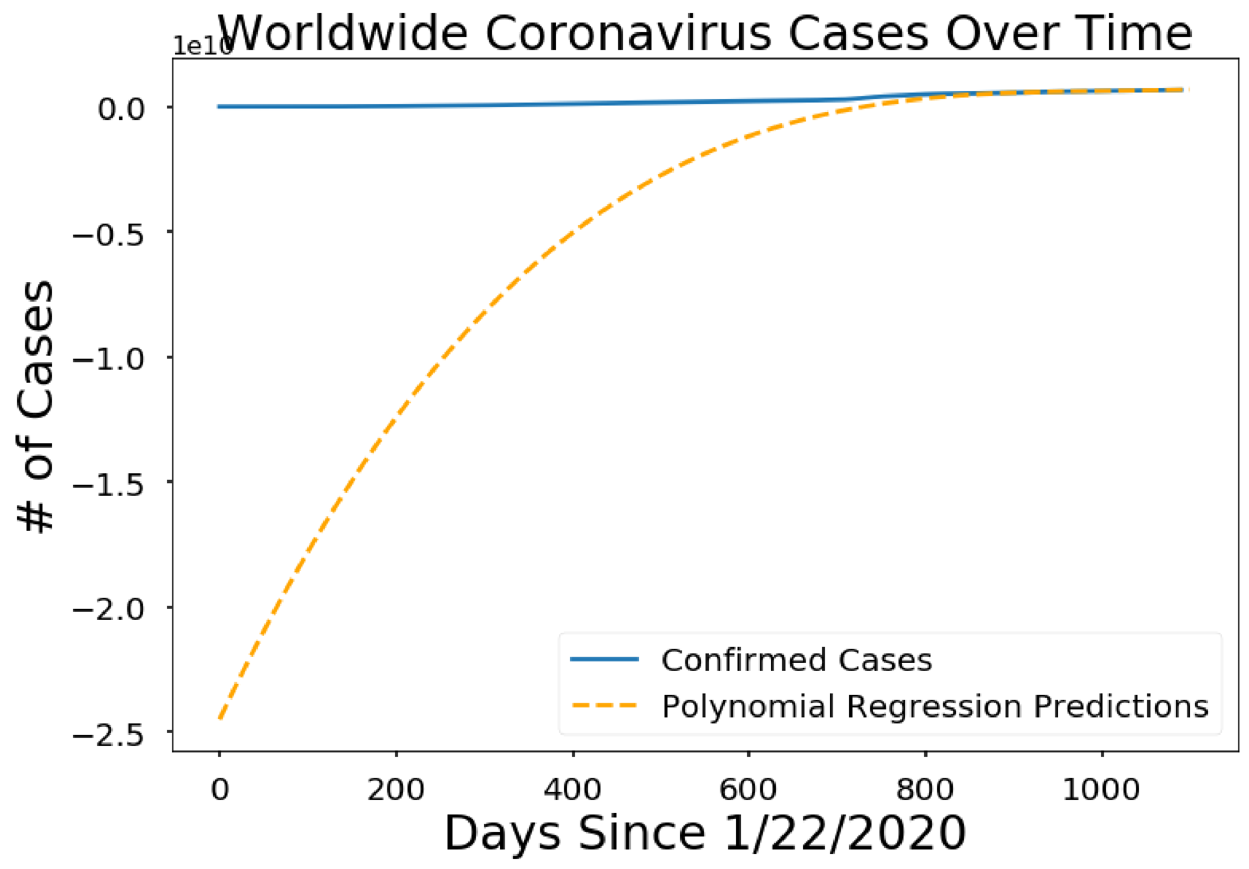
SVM Prediction:

```
plot_predictions(adjusted_dates, world_cases, svm_pred, 'SVM Predictions', 'purple')
```



Polynomial prediction:

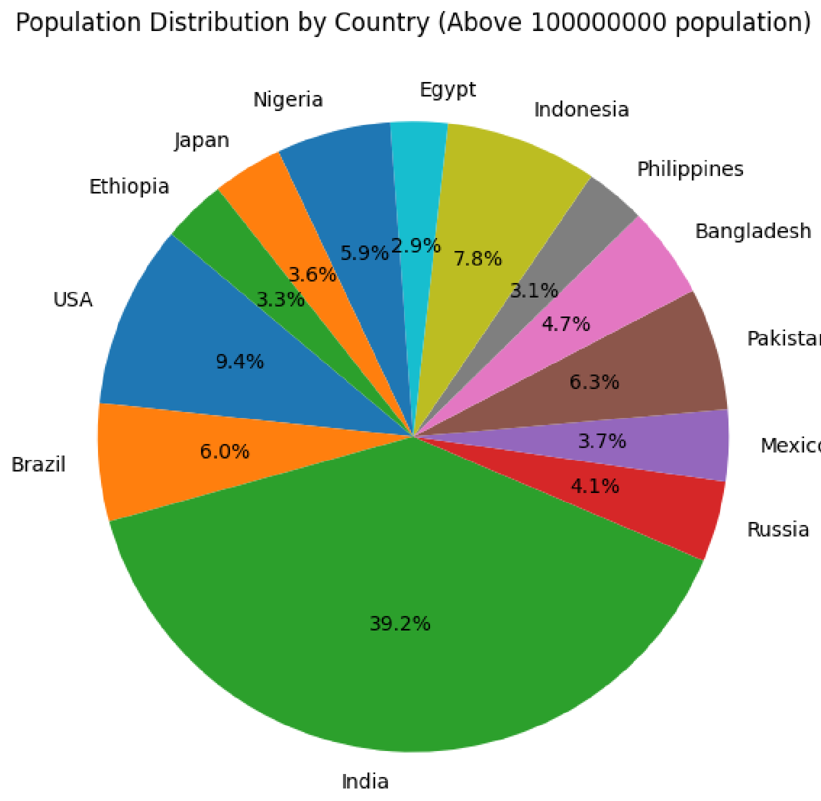
```
plot_predictions(adjusted_dates, world_cases, linear_pred, 'Polynomial Regression Predictions', 'orange')
```



Population:

```
import pandas as pd

import matplotlib.pyplot as plt


# Load your dataset

df1 = pd.read_csv("C:\\Users\\Imaya\\Desktop\\online courses\\naan
mudhalvan\\Dataset1\\worldometer_data.csv")


# Set a threshold for population

population_threshold = 100000000  # For example, 100 million
```

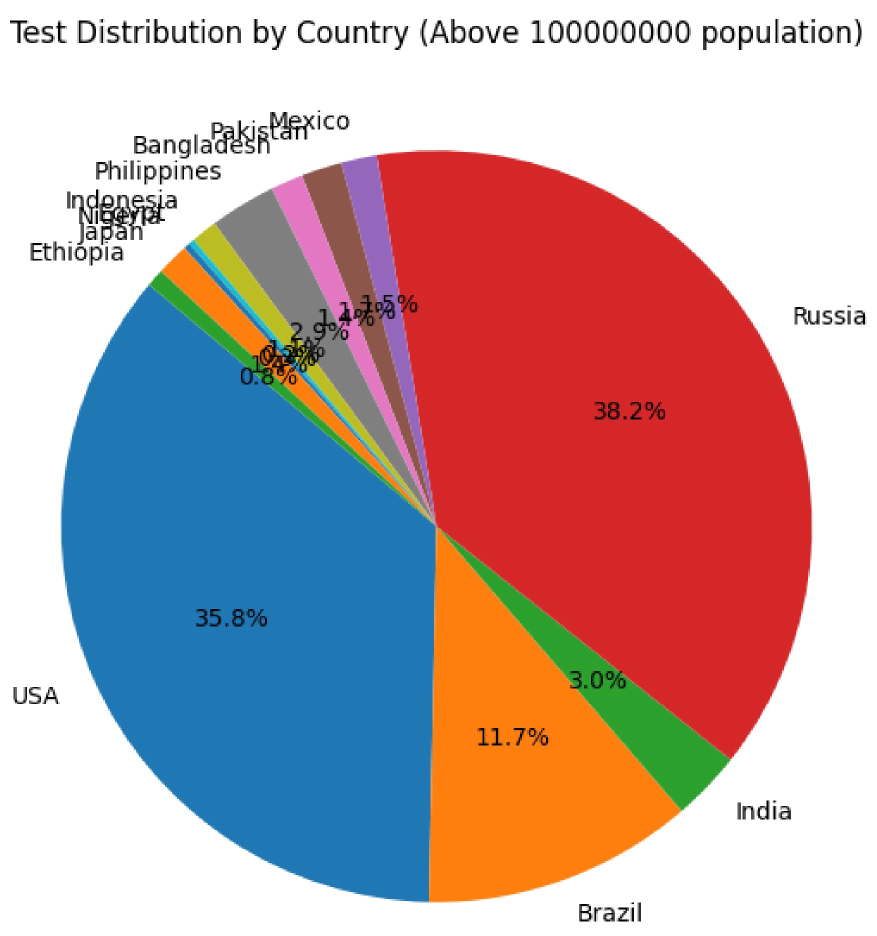
```
# Filter the data to include only countries with populations above the threshold
filtered_df = df1[df1['Population'] > population_threshold]

# Get the countries and populations for the filtered data
countries = filtered_df['Country/Region']
populations = filtered_df['Population']

# Create a pie chart for the filtered data
plt.figure(figsize=(10, 10))
plt.pie(populations, labels=countries, autopct='%1.1f%%', startangle=140)
plt.title('Population Distribution by Country (Above {}
population)'.format(population_threshold))
plt.show()
```



Covid-19 Testing:

```
import pandas as pd

import matplotlib.pyplot as plt


# Load your dataset

df1 = pd.read_csv("C:\\Users\\Imaya\\Desktop\\online courses\\naan
mudhalvan\\Dataset1\\worldometer_data.csv")


# Set a threshold for population

population_threshold = 100000000  # For example, 100 million
```

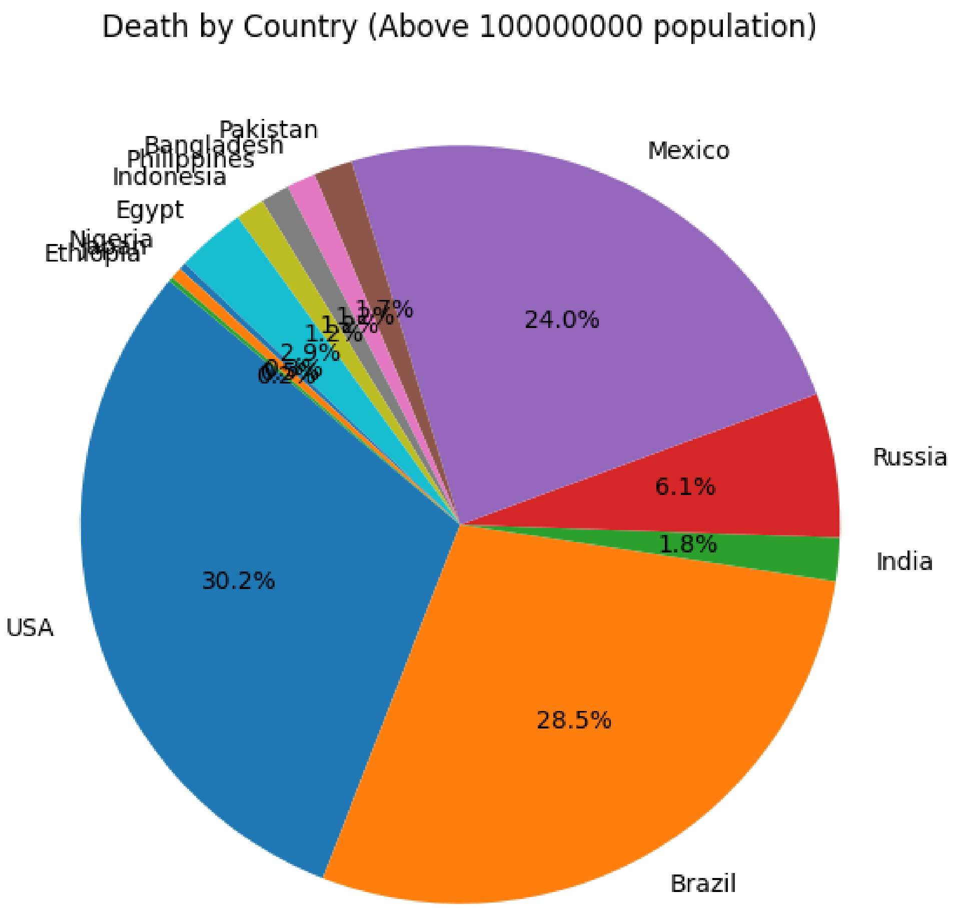
```
# Filter the data to include only countries with populations above the threshold
filtered_df = df1[df1['Population'] > population_threshold]

# Get the countries and populations for the filtered data
countries = filtered_df['Country/Region']
Tests = filtered_df['Tests/1M pop']

# Create a pie chart for the filtered data
plt.figure(figsize=(10, 10))
plt.pie(Tests, labels=countries, autopct='%1.1f%%', startangle=140)
plt.title('Test Distribution by Country (Above {}
population)'.format(population_threshold))
plt.show()
```



Covid-19 Deaths:

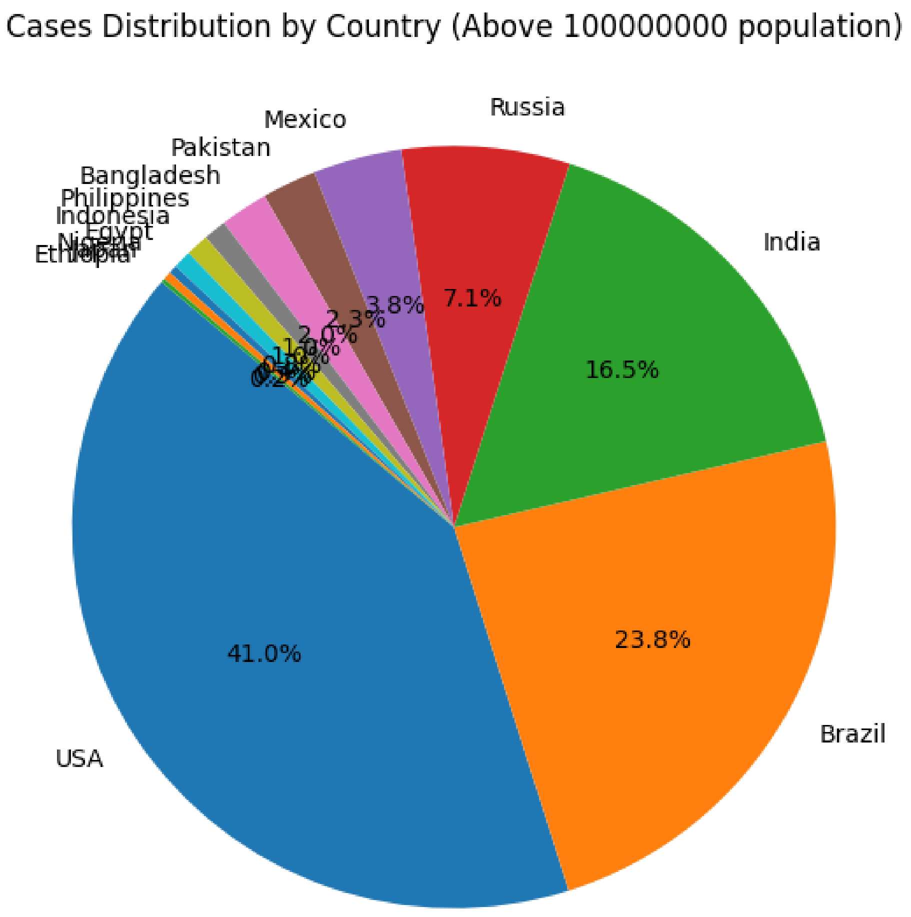
```
import pandas as pd

import matplotlib.pyplot as plt


# Load your dataset

df1 = pd.read_csv("C:\\Users\\Imaya\\Desktop\\online courses\\naan
mudhalvan\\Dataset1\\worldometer_data.csv")
```

```
# Set a threshold for population
population_threshold = 100000000  # For example, 100 million

# Filter the data to include only countries with populations above the threshold
filtered_df = df1[df1['Population'] > population_threshold]

# Get the countries and populations for the filtered data
countries = filtered_df['Country/Region']
Deaths = filtered_df['Deaths/1M pop']

# Create a pie chart for the filtered data
plt.figure(figsize=(10, 10))
plt.pie(Deaths, labels=countries, autopct='%1.1f%%', startangle=140)
plt.title('Death by Country (Above {} population)'.format(population_threshold))
plt.show()
```



```
import pandas as pd

import matplotlib.pyplot as plt


# Load your dataset

df1 = pd.read_csv("C:\\Users\\Imaya\\Desktop\\online courses\\naan
mudhalvan\\Dataset1\\worldometer_data.csv")


# Set a threshold for population

population_threshold = 100000000  # For example, 100 million
```

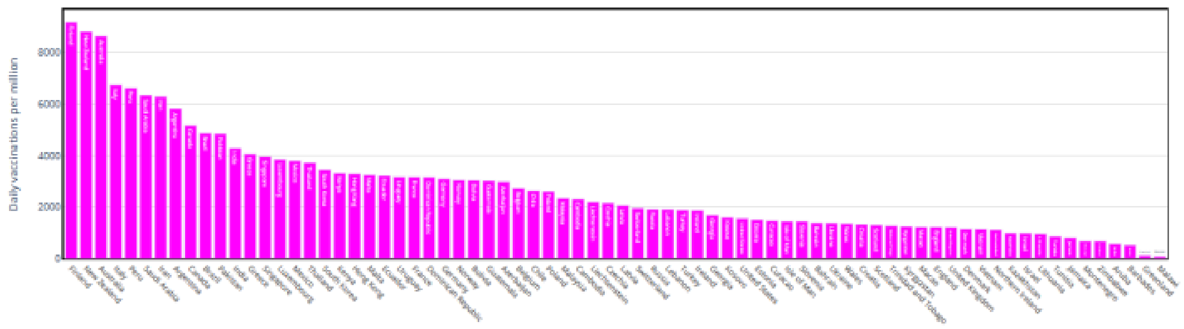
```
# Filter the data to include only countries with populations above the threshold
filtered_df = df1[df1['Population'] > population_threshold]

# Get the countries and populations for the filtered data
countries = filtered_df['Country/Region']
Cases = filtered_df['TotalCases']

# Create a pie chart for the filtered data
plt.figure(figsize=(10, 10))
plt.pie(Cases, labels=countries, autopct='%1.1f%%', startangle=140)
plt.title('Cases Distribution by Country (Above {}
population)'.format(population_threshold))
plt.show()
```



Country-Wise Daily Vaccination per Million:

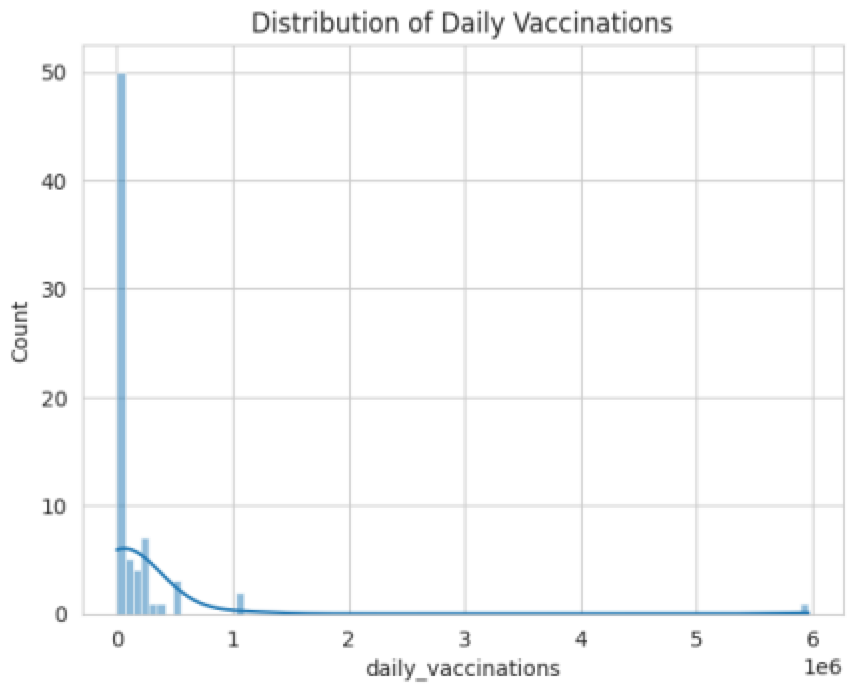
```
def trace_bar(data, feature, title, xlab, ylab, color):
 data = data.sort_values(feature, ascending=False)
 trace = go.Bar(
 x=data['country'],
 y=data[feature],
 marker=dict(color=color),
 text=data['country']
 )
```

```
data = [trace]
layout = dict(
title=title,
xaxis=dict(
title=xlab,
showticklabels=True,
tickangle=45,
zeroline=True,
zerolinewidth=1,
zerolinecolor='grey',
showline=True,
linewidth=2,
linecolor='black',
mirror=True,
tickfont=dict(size=10, color='black'),
),
yaxis=dict(
title=ylab,
gridcolor='lightgrey',
zeroline=True,
zerolinewidth=1,
zerolinecolor='grey',
showline=True,
linewidth=2,
 linecolor='black',
 mirror=True
 ),
 plot_bgcolor='rgba(0, 0, 0, 0)',
 paper_bgcolor='rgba(0, 0, 0, 0)',
 hovermode='closest'
 )
 fig = dict(data=data, layout=layout)
 iplot(fig)
trace_bar(df_vaccinations, 'daily_vaccinations_per_million', 'Daily Vaccinations
per Million per
Country', 'Country', 'Daily Vaccinations per Million', 'blue')
```



Distribution of Daily Vaccine:

```
sns.histplot(df_vaccinations['daily_vaccinations'], kde=True)
plt.title("Distribution of Daily Vaccinations")
plt.show()
```



Outcome:

1. Data Understanding:

• The documentation begins with importing necessary libraries and loading the data sets, providing insight into the tools and datasets used.

2. Visualization:

• Various visualizations are included to provide a graphical representation of the data. This includes heatmaps, bar plots, and charts showcasing top countries in vaccination, preferred vaccines, and more.

• The visualizations aim to make complex data more understandable and highlight trends and patterns.

3. Statistical Analysis:

• Statistical analysis is performed on key attributes, providing summary statistics such as mean, standard deviation, and quartiles.

• This section enables a quantitative understanding of the data distribution and central tendencies.

4. Insights and Interpretation:

• Throughout the documentation, insights are provided, such as the top countries in vaccination, most commonly used vaccines, and statistical summaries.

• These insights aid in drawing meaningful conclusions from the data, supporting decision-making processes.

5. Customization:

• The documentation is designed to be customizable based on specific requirements. Users can adapt and extend the documentation to suit their analysis goals or share it with others to facilitate collaboration.

CONCLUSION:

Data loading and preprocessing for vaccine analysis serve as the critical initial steps that empower researchers and healthcare professionals to make informed decisions about vaccine development, distribution, and safety. The quality, accuracy, and suitability of the data at this stage are pivotal in determining the success of subsequent analyses. Through diligent and systematic data handling, we can harness the power of data-driven insights to address public health challenges and contribute to the betterment of global health