COVID Vaccines Analysis

Phase 4: Development Part 2

Objective:

The goal is to advance the COVID vaccine analysis through exploratory data analysis (EDA), statistical examination, and visualization. The focus is on uncovering patterns, trends, and relationships within the data to inform evidence-based insights. By employing a combination of statistical tools and visualizations, the aim is to provide a comprehensive understanding of vaccination dynamics, identify global trends, and contribute valuable insights for strategic decision-making.

1. Importing Required Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objs as go
from plotly.offline import init_notebook_mode, iplot, plot
```

2. Loading Data Sets

Read data from CSV files

```
df_manufacturer = pd.read_csv('/content/drive/MyDrive/Naan Mudalvan/country_vaccinations_by_manufacturer.csv')
df_vaccinations = pd.read_csv('/content/drive/MyDrive/Naan Mudalvan/country_vaccinations.csv')
```

Columns Present in Given Data Sets

```
df_manufacturer.columns
```

Output:

```
['location', 'date', 'vaccine', 'total_vaccinations']
```

df_vaccinations.columns

Output:

['country', 'iso_code', 'date', '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', 'vaccines', 'source_name', 'source_website']

Shape of DataFrames

df_manufacturer.shape

Output:

(35623, 4)

df_vaccinations.shape

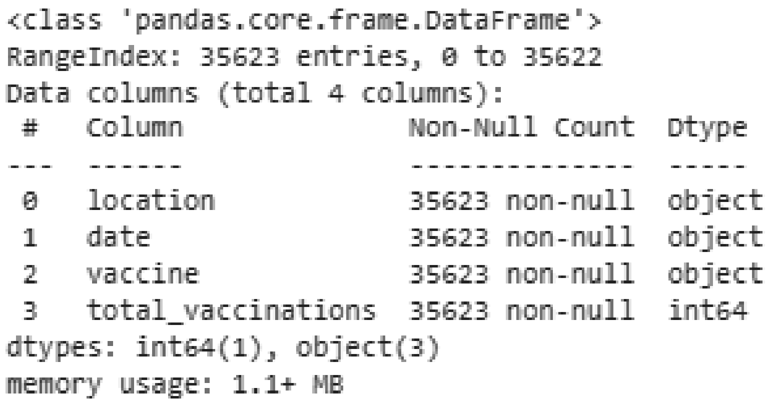
Output:

(86512, 15)

Information about Given Data Sets

df_manufacturer.info()

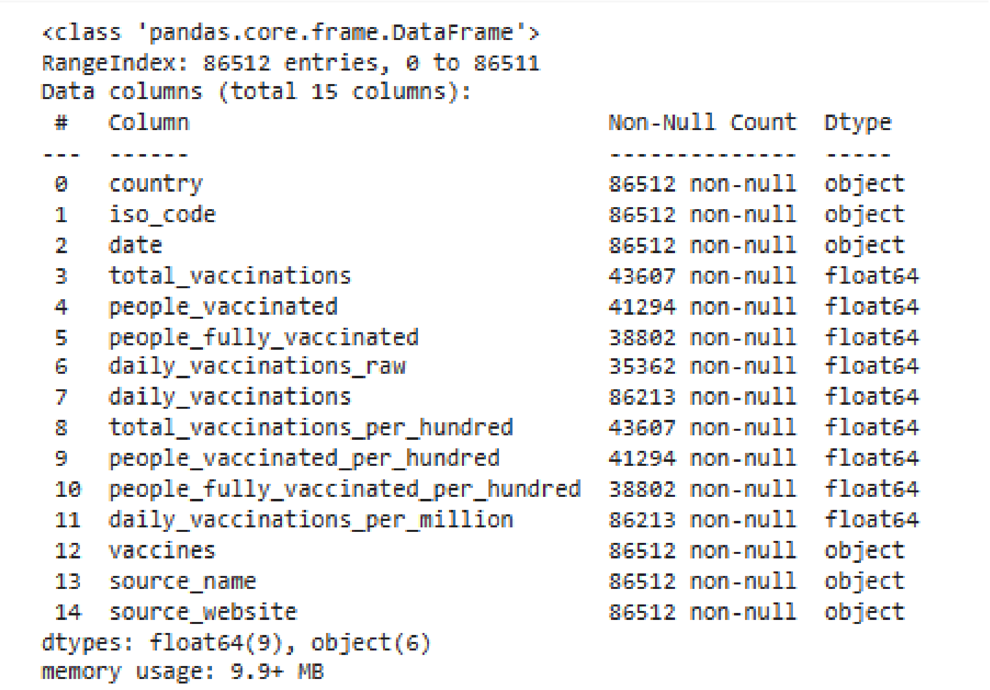
Output:



```
<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
```

df_vaccinations.info()

Output:



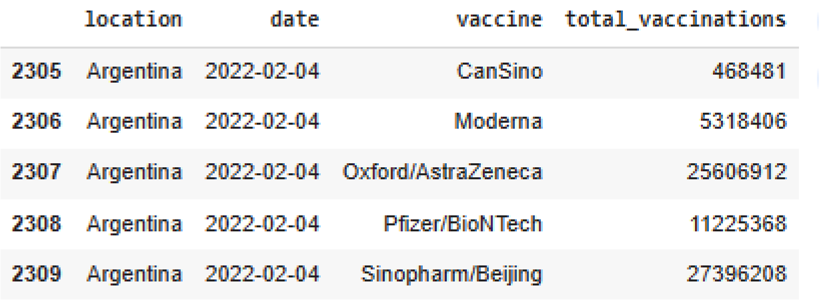
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86512 entries, 0 to 86511
Data columns (total 15 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   country                              86512 non-null  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
```

Vaccines Manufactured on a Particular Date

df_manufacturer = df_manufacturer[df_manufacturer.date == '2022-02-04']

df_manufacturer.head()

Output:



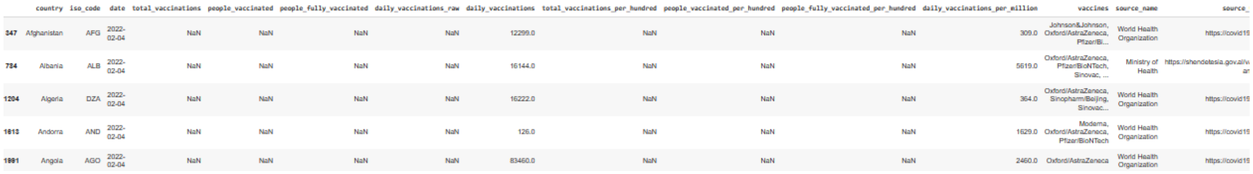
	location	date	vaccine	total_vaccinations
2305	Argentina	2022-02-04	CanSino	468481
2306	Argentina	2022-02-04	Moderna	5318406
2307	Argentina	2022-02-04	Oxford/AstraZeneca	25606912
2308	Argentina	2022-02-04	Pfizer/BioNTech	11225368
2309	Argentina	2022-02-04	Sinopharm/Beijing	27396208

Country-Wise Vaccination Status on a Particular Date

df_vaccinations = df_vaccinations[df_vaccinations.date == '2022-02-04']

df_vaccinations.head()

Output:



	country	iso_code	date	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	vaccines	source_name	source_
347	Afghanistan	AFG	2022- 02-04	NaN	NaN	NaN	NaN	12299.0	NaN	NaN	NaN	309.0	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...	World Health Organization	https://covid19
784	Albania	ALB	2022- 02-04	NaN	NaN	NaN	NaN	16144.0	NaN	NaN	NaN	5619.0	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, ...	Ministry of Health	https://shendetesia.gov.al/v an
1204	Algeria	DZA	2022- 02-04	NaN	NaN	NaN	NaN	16222.0	NaN	NaN	NaN	364.0	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...	World Health Organization	https://covid19
1613	Andorra	AND	2022- 02-04	NaN	NaN	NaN	NaN	126.0	NaN	NaN	NaN	1629.0	Moderna, Oxford/AstraZeneca, Pfizer/BioNTech	World Health Organization	https://covid19
1891	Angola	AGO	2022- 02-04	NaN	NaN	NaN	NaN	83460.0	NaN	NaN	NaN	2460.0	Oxford/AstraZeneca	World Health Organization	https://covid19

3. **Preprocessing Data**

   I. **Null Values:**

- Null values, often represented as NaN (Not a Number) in Python, indicate missing or undefined data.
- It's crucial to identify and handle null values, as they can affect the accuracy of our analysis or machine learning models.

Common methods to handle null values include:

a) Removing Rows: If a small percentage of rows have null values and removing them won't significantly impact our analysis.

b) Imputation: Fill null values with the mean, median, or mode of the respective column.

c) Forward/Backward Fill: Use the values from the previous or next row to fill null values.

d) Interpolation: Estimate missing values based on the values of other rows using methods like linear interpolation.

   II. **Missing Values:**

- Missing values can occur due to various reasons such as data collection errors or intentional gaps.
- Techniques for handling missing values are similar to those for null values.

   III. **Outliers:**

- Outliers are data points that deviate significantly from the rest of the data.
- Identifying outliers is crucial for accurate analysis and modeling.

Common methods for detecting outliers include:

a) Visual Inspection: Plotting the data and looking for points that deviate from the overall pattern.

b) Statistical Methods: Using measures like Z-scores or IQR (Interquartile Range) to identify values that are significantly different from the mean or median.

c) Machine Learning Models: Some models are sensitive to outliers, so detecting them during model training can be beneficial.

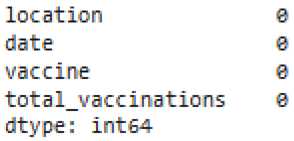
IV. Removing Duplicates:

- Duplicate values in a dataset can arise from data entry errors or other issues.
- Removing duplicates ensures that each data point is unique.
- In Python, you can use the `drop_duplicates` method for DataFrames in pandas to remove duplicate rows.
- Columns can also be checked for duplicates using methods like `duplicated()`.

Checking for Missing Values

df_manufacturer.isna().sum()

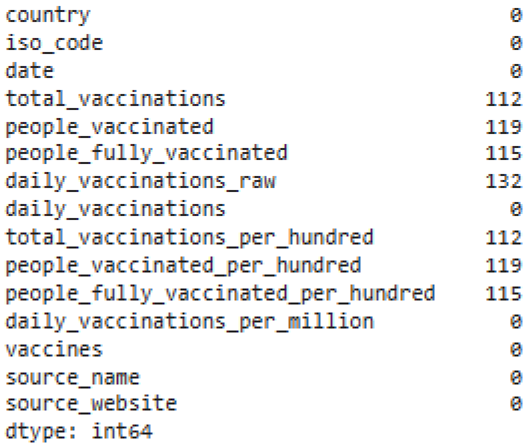
Output:



```
location              0
date                  0
vaccine               0
total_vaccinations    0
dtype: int64
```

df_vaccinations.isna().sum()

Output:

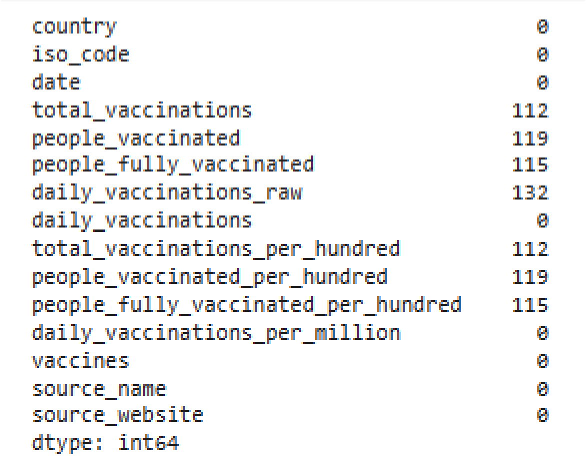


```
country                                  0
iso_code                                 0
date                                     0
total_vaccinations                     112
people_vaccinated                      119
people_fully_vaccinated                115
daily_vaccinations_raw                 132
daily_vaccinations                       0
total_vaccinations_per_hundred         112
people_vaccinated_per_hundred          119
people_fully_vaccinated_per_hundred    115
daily_vaccinations_per_million           0
vaccines                                 0
source_name                              0
source_website                           0
dtype: int64
```

Dropping Missing Values

df_vaccinations.isna().sum()

Output:



```
country                                 0
iso_code                                0
date                                    0
total_vaccinations                    112
people_vaccinated                     119
people_fully_vaccinated               115
daily_vaccinations_raw                132
daily_vaccinations                      0
total_vaccinations_per_hundred        112
people_vaccinated_per_hundred         119
people_fully_vaccinated_per_hundred   115
daily_vaccinations_per_million          0
vaccines                                0
source_name                             0
source_website                          0
dtype: int64
```

df_vaccinations =

df_vaccinations.drop(df_vaccinations[df_vaccinations.total_vaccinations.isna()].index)

df_vaccinations =

df_vaccinations.drop(df_vaccinations[df_vaccinations.people_vaccinated.isna()].index)

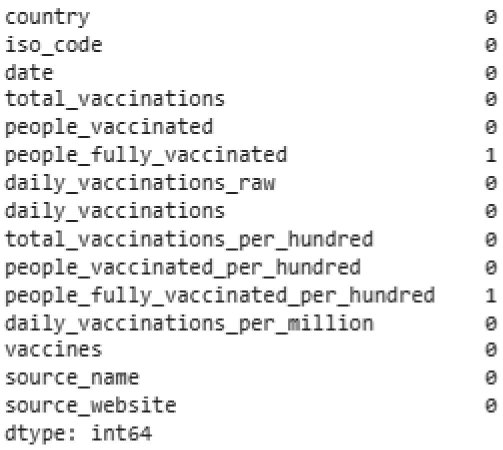
df_vaccinations =

df_vaccinations.drop(df_vaccinations[df_vaccinations.daily_vaccinations_raw.isna()].index)

Checking for Null Values

df_vaccinations.isnull().sum()

Output:



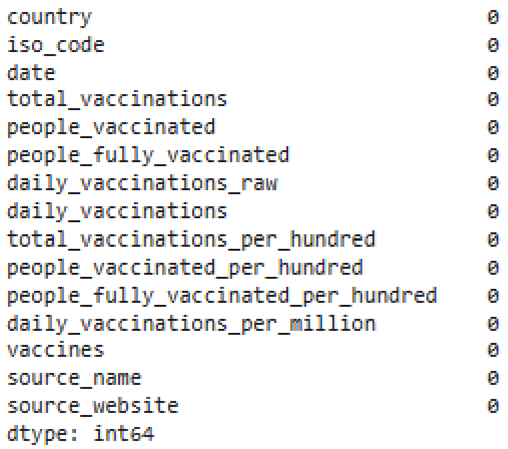
```
country                                0
iso_code                               0
date                                   0
total_vaccinations                     0
people_vaccinated                      0
people_fully_vaccinated                1
daily_vaccinations_raw                 0
daily_vaccinations                     0
total_vaccinations_per_hundred         0
people_vaccinated_per_hundred          0
people_fully_vaccinated_per_hundred    1
daily_vaccinations_per_million         0
vaccines                               0
source_name                            0
source_website                         0
dtype: int64
```

Filling Mean Values

df_vaccinations = df_vaccinations.fillna(df_vaccinations.mean())

df_vaccinations.isnull().sum()

Output:



```
country                                0
iso_code                               0
date                                   0
total_vaccinations                     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
```

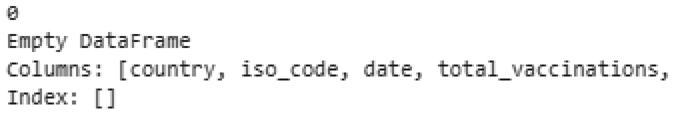
Checking for Duplicated Records

duplicate_rows = df_vaccinations[df_vaccinations.duplicated()]

print(len(duplicate_rows))

print(duplicate_rows)

Output:



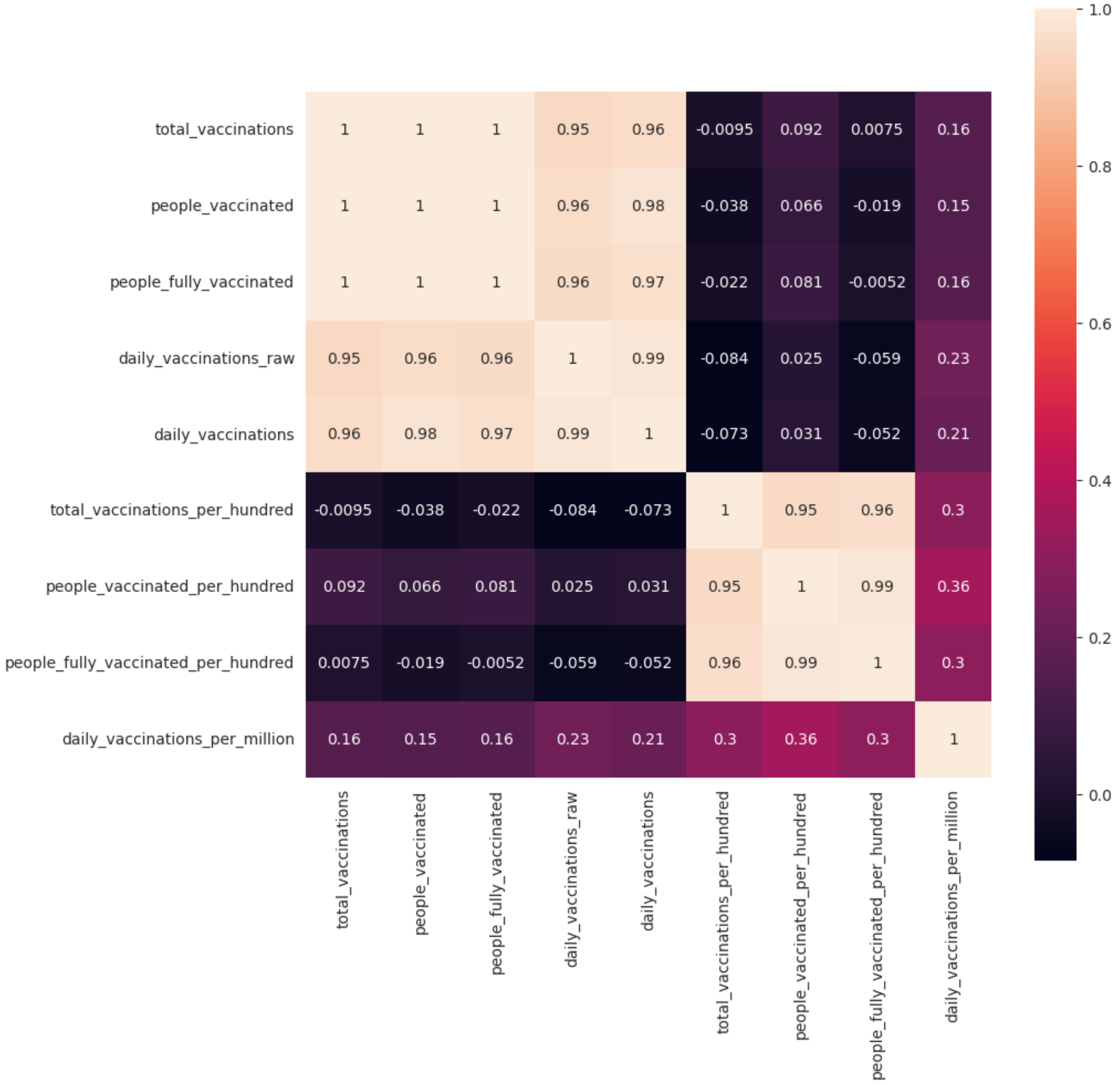
```
0
Empty DataFrame
Columns: [country, iso_code, date, total_vaccinations,
Index: []
```

4. **Visualization and Statistical Analysis of Data**

Visualization techniques, such as heatmaps, bar plots, and histograms, are employed to illustrate correlations, top countries in vaccination utilization, and distribution of daily vaccinations. The analysis go through statistical summaries of key attributes, including total vaccinations, people vaccinated, and daily vaccinations. Additionally, country-specific analyses, such as preferred vaccines in India and daily vaccinations per million in top countries, offer targeted insights.

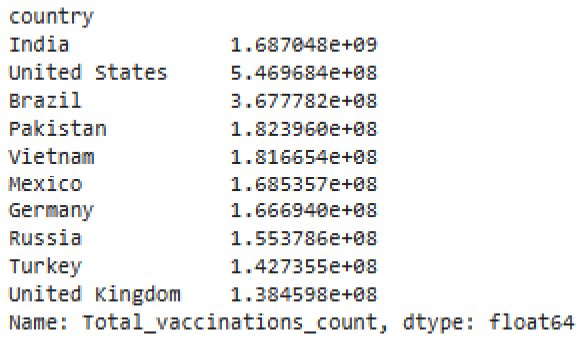
Heatmap Visualization to Check Correlation Between Attributes

```
plt.subplots(figsize=(10, 10))
sns.heatmap(df_vaccinations.corr(), annot=True, square=True)
plt.show()
```



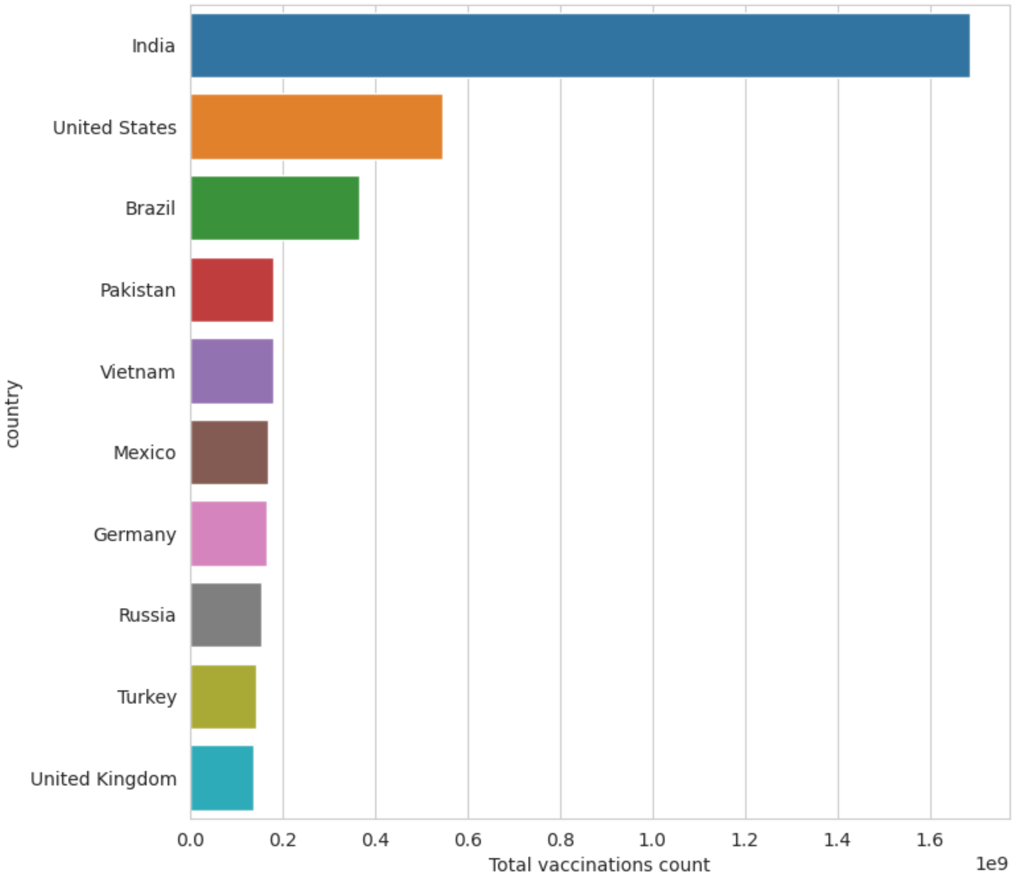
Top Countries in Vaccination Utilization

```
df_vaccinations["Total_vaccinations_count"] =
df_vaccinations.groupby("country").total_vaccinations.tail(1)
```



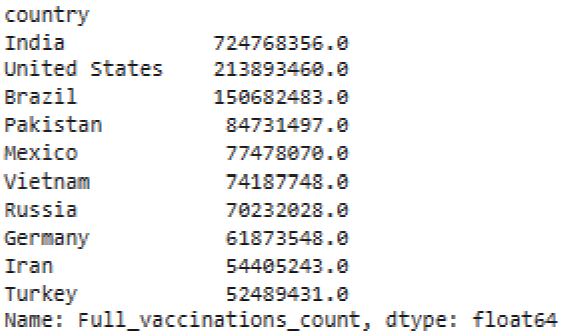
```
country
India             1.687048e+09
United States     5.469684e+08
Brazil            3.677782e+08
Pakistan          1.823960e+08
Vietnam           1.816654e+08
Mexico            1.685357e+08
Germany           1.666940e+08
Russia            1.553786e+08
Turkey            1.427355e+08
United Kingdom    1.384598e+08
Name: Total_vaccinations_count, dtype: float64
```

```
x = df_vaccinations.groupby("country")["Total_vaccinations_count"].mean().sort_values(ascending=False).head(10)
sns.set_style("whitegrid")
plt.figure(figsize=(8, 8))
ax = sns.barplot(x=x.values, y=x.index)
ax.set_xlabel("Total vaccinations count")
plt.show()
```



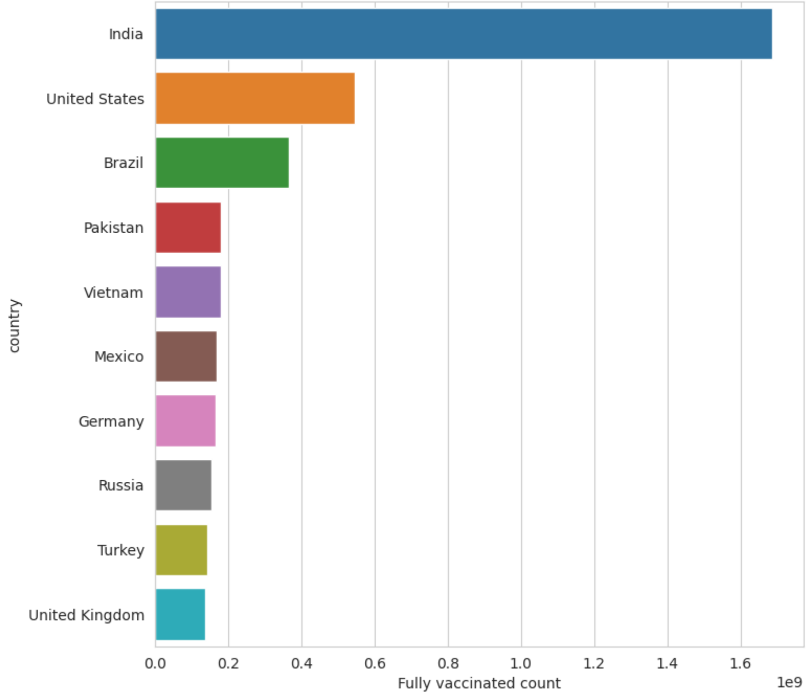
Fully Vaccinated Count

```
df_vaccinations["Full_vaccinations_count"] =
df_vaccinations.groupby("country").people_fully_vaccinated.tail(1)
```



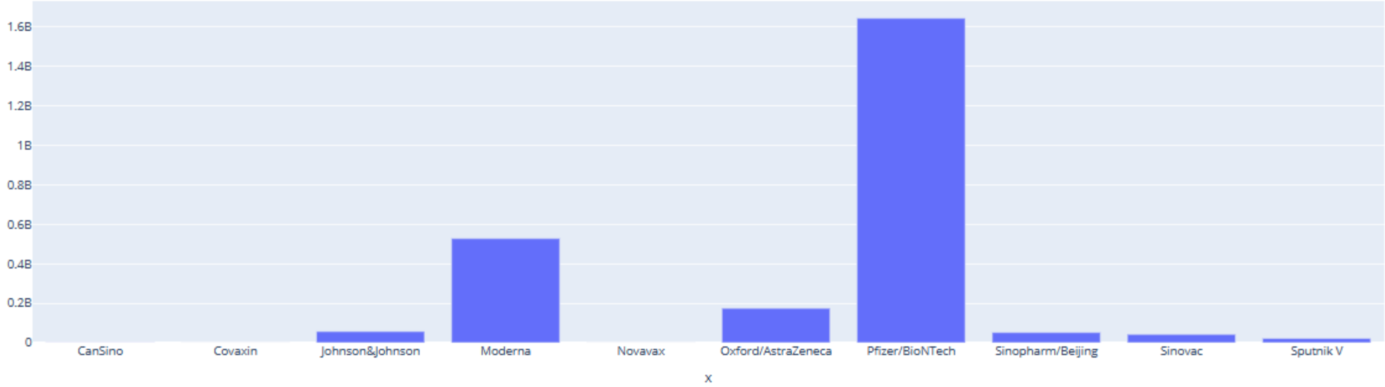
```
country
India             724768356.0
United States     213893460.0
Brazil            150682483.0
Pakistan           84731497.0
Mexico             77478070.0
Vietnam            74187748.0
Russia             70232028.0
Germany            61873548.0
Iran               54405243.0
Turkey             52489431.0
Name: Full_vaccinations_count, dtype: float64
```

```
x =
df_vaccinations.groupby("country")["Full_vaccinations_count"].mean().sort_values(ascending=
False).head(10)
sns.set_style("whitegrid")
plt.figure(figsize=(8, 8))
ax = sns.barplot(x=x.values, y=x.index)
ax.set_xlabel("Fully vaccinated count")
plt.show()
```



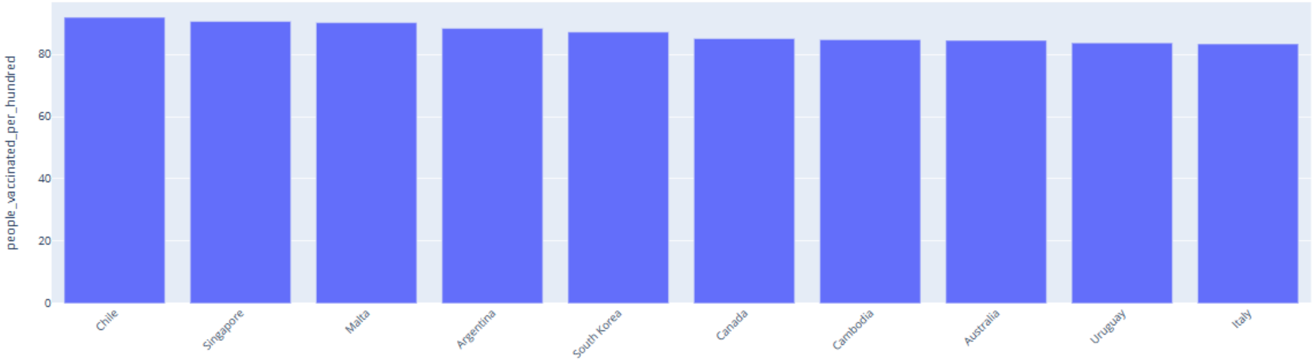
Most Commonly Used Vaccines in the World

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



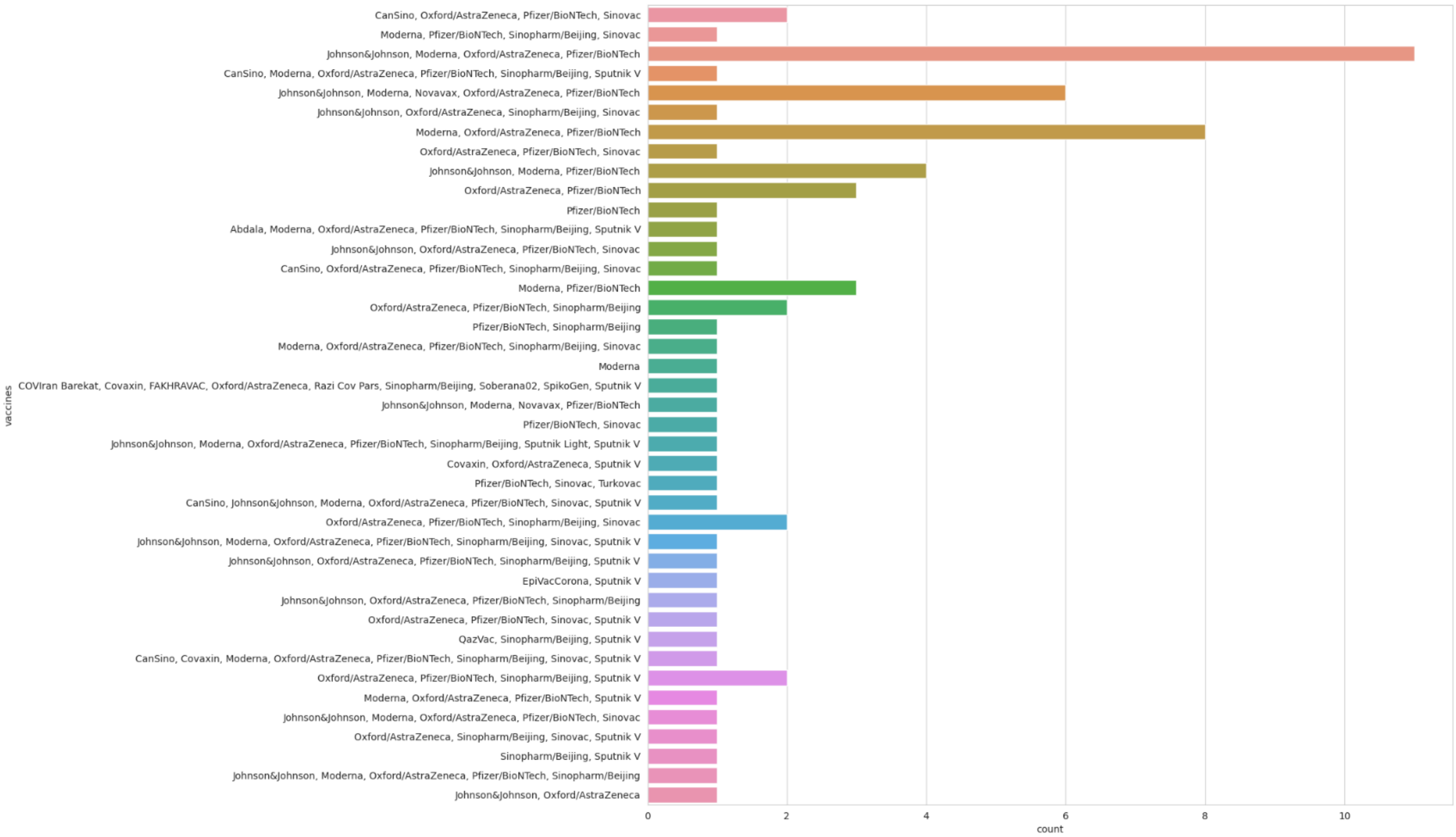
People Vaccinated per Hundred for the Date 2022-02-04

```
df_vaccinations = df_vaccinations[df_vaccinations['date'] == '2022-02-04']
df_vaccinations = df_vaccinations.sort_values(by='people_vaccinated_per_hundred',
ascending=False)
fig = px.bar(df_vaccinations.head(10), x='country', y='people_vaccinated_per_hundred',
title='People Vaccinated per Hundred for the Date 2022-02-04')
fig.update_layout(xaxis_tickangle=-45)
fig.show()
```



Type of Vaccine Utilized vs Count

```
plt.figure(figsize=(15, 15))
sns.countplot(y="vaccines", data=df_vaccinations)
plt.show()
```



Vaccination per Hundred Top Countries

df_vaccinations["Total_vaccinations_per_hundred"] =

df_vaccinations.groupby("country").total_vaccinations_per_hundred.tail(1)

x =

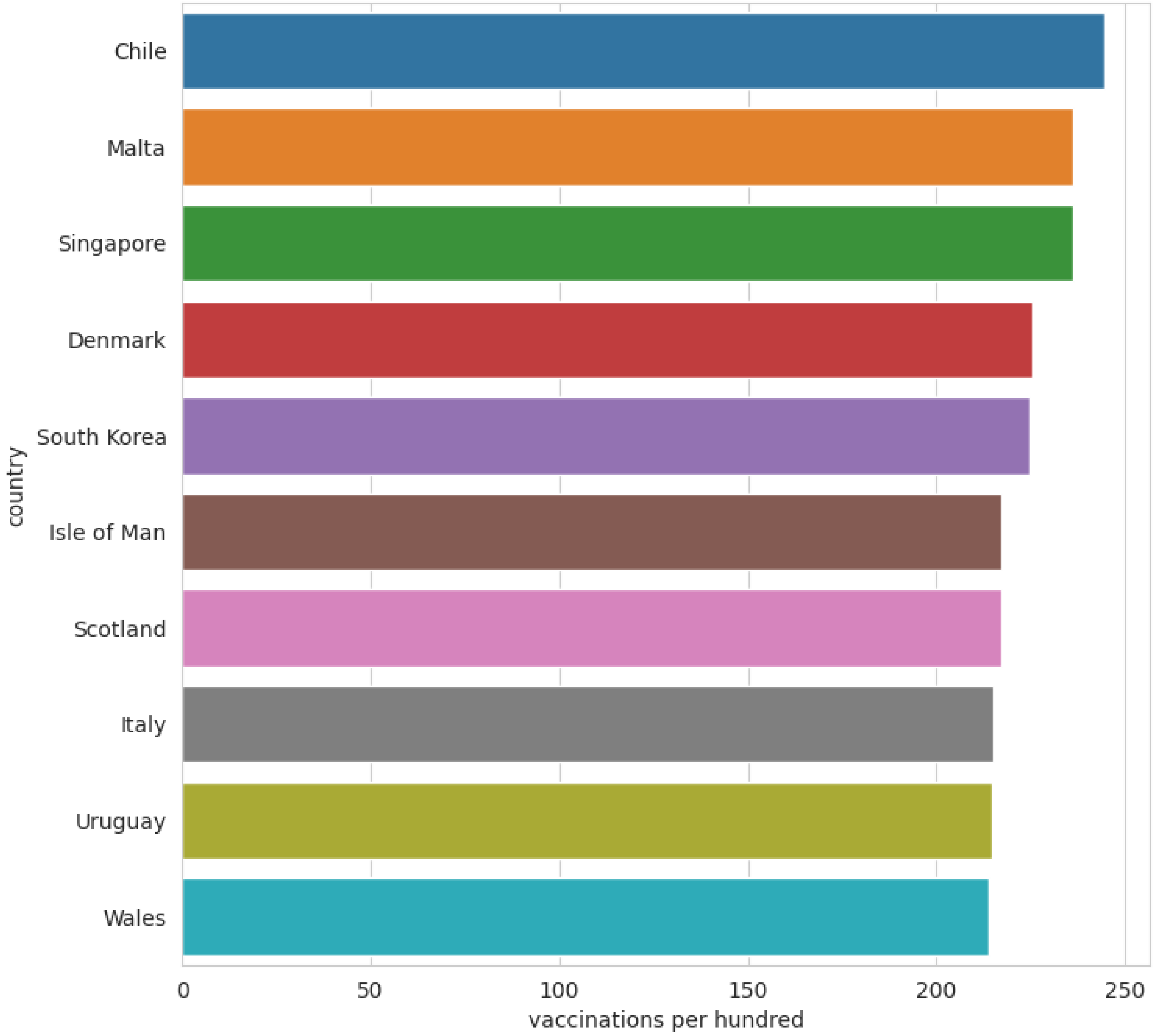
df_vaccinations.groupby("country")["Total_vaccinations_per_hundred"].mean().sort_values(ascending=False).head(10)

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

ax = sns.barplot(x=x.values, y=x.index)

ax.set_xlabel("Vaccinations per hundred")

plt.show()

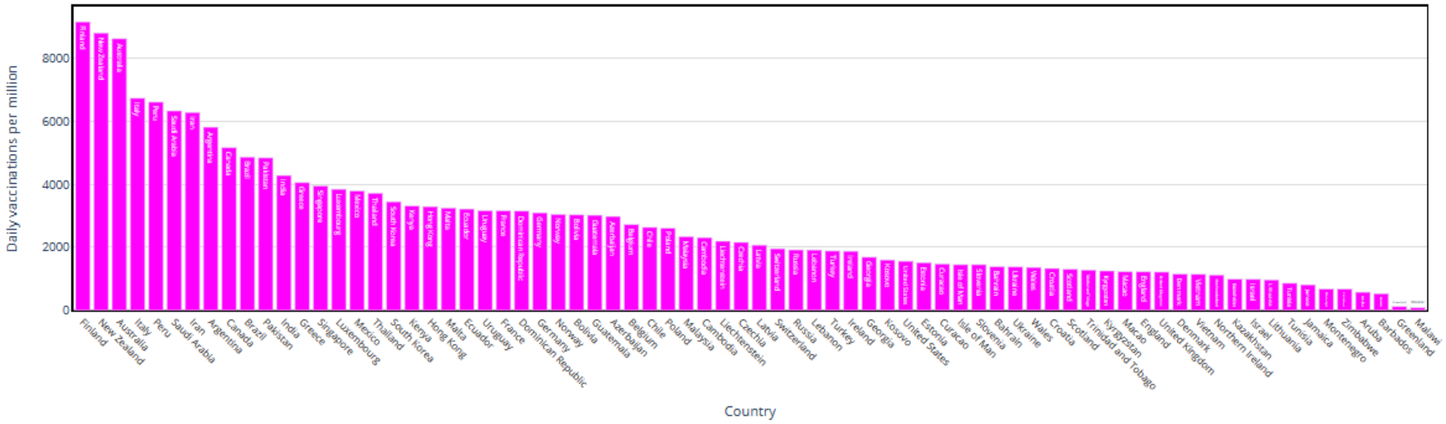


Country-Wise Daily Vaccination per Million

```
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']
    )
```

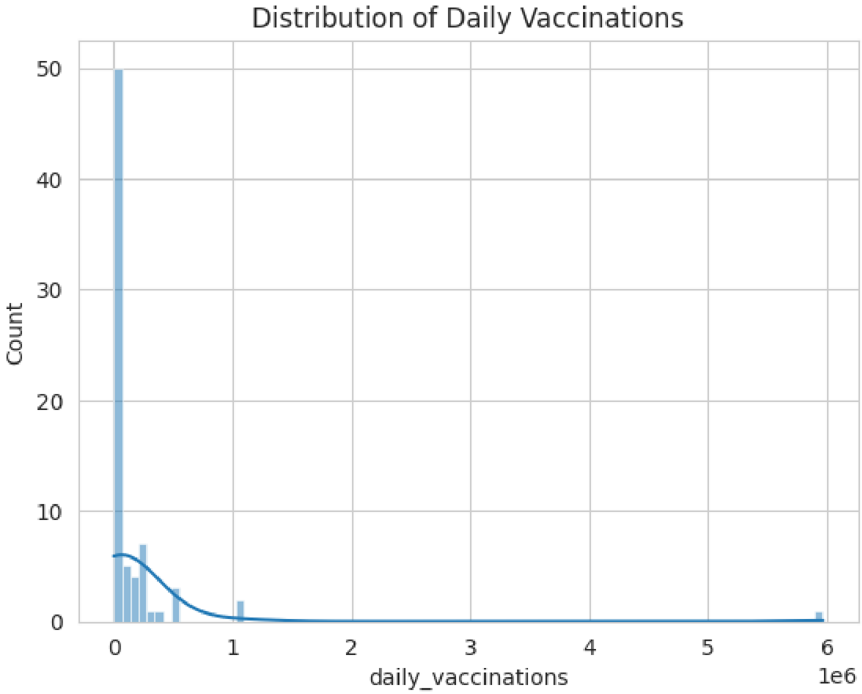
```
data = [trace]
layout = dict(
    title=title,
    xaxis=dict(
        title=xlab,
        showticklabels=True,
        tickangle=45,
        zeroline=True,
        zerolinewidth=1,
        zerolinecolor='grey',
        showline=True,
        linewidth=2,
        linecolor='black',
        mirror=True,
        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()
```



Statistical Analysis of Given Data Sets

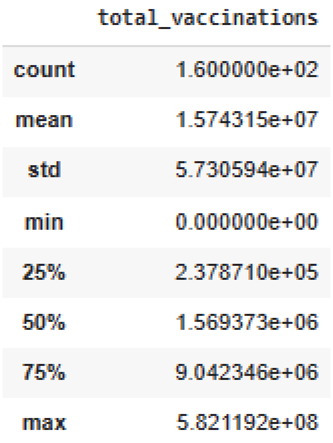
Descriptive Statistics:

- *Mean:* The average of a set of values.
- *Median:* The middle value of a sorted dataset.
- *Standard Deviation:* A measure of the amount of variation or dispersion in a set of values.

Total Vaccinations Statistical Analysis

df_manufacturer['total_vaccinations'].describe()

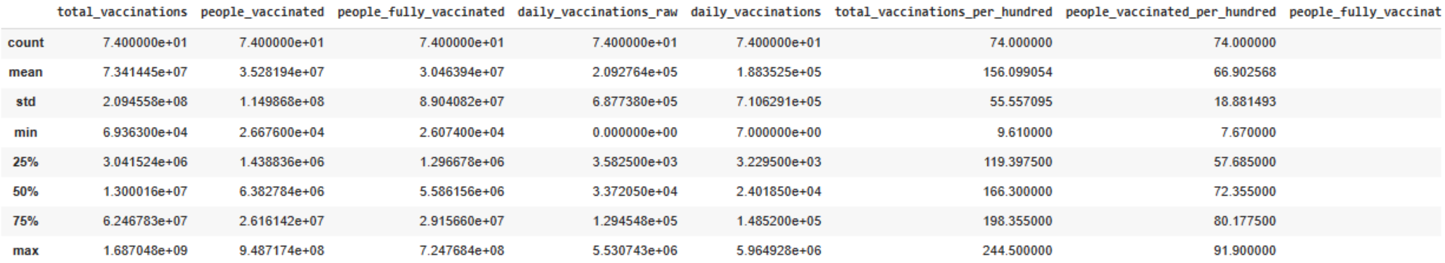
Output:



	total_vaccinations
count	1.600000e+02
mean	1.574315e+07
std	5.730594e+07
min	0.000000e+00
25%	2.378710e+05
50%	1.569373e+06
75%	9.042346e+06
max	5.821192e+08

df_vaccinations.describe()

Output:

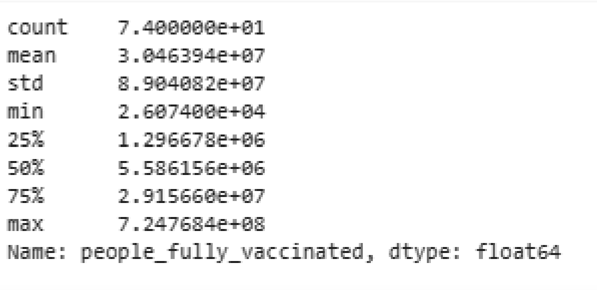


	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred	people_vaccinated_per_hundred	people_fully_vaccinat
count	7.400000e+01	7.400000e+01	7.400000e+01	7.400000e+01	7.400000e+01	74.000000	74.000000	
mean	7.341445e+07	3.528194e+07	3.046394e+07	2.092764e+05	1.883525e+05	156.099054	66.902568	
std	2.094558e+08	1.149868e+08	8.904082e+07	6.877380e+05	7.106291e+05	55.557095	18.881493	
min	6.936300e+04	2.667600e+04	2.607400e+04	0.000000e+00	7.000000e+00	9.610000	7.670000	
25%	3.041524e+06	1.438836e+06	1.296678e+06	3.582500e+03	3.229500e+03	119.397500	57.685000	
50%	1.300016e+07	6.382784e+06	5.586156e+06	3.372050e+04	2.401850e+04	166.300000	72.355000	
75%	6.246783e+07	2.616142e+07	2.915660e+07	1.294548e+05	1.485200e+05	198.355000	80.177500	
max	1.687048e+09	9.487174e+08	7.247684e+08	5.530743e+06	5.964928e+06	244.500000	91.900000	

People Fully Vaccinated Statistical Analysis

df_manufacturer['people_fully_vaccinated'].describe()

Output:

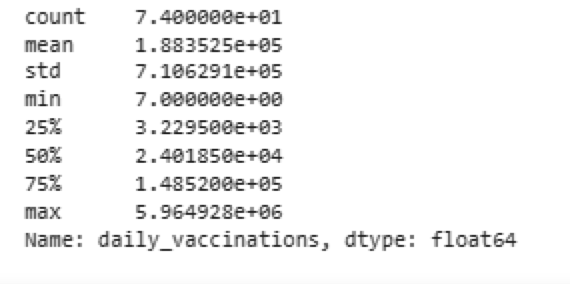


```
count    7.400000e+01
mean     3.046394e+07
std      8.904082e+07
min      2.607400e+04
25%      1.296678e+06
50%      5.586156e+06
75%      2.915660e+07
max      7.247684e+08
Name: people_fully_vaccinated, dtype: float64
```

Daily Vaccinations Statistical Analysis

df_manufacturer['daily_vaccinations'].describe()

Output:

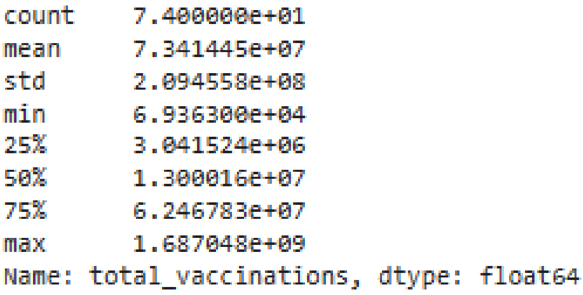


```
count    7.400000e+01
mean     1.883525e+05
std      7.106291e+05
min      7.000000e+00
25%      3.229500e+03
50%      2.401850e+04
75%      1.485200e+05
max      5.964928e+06
Name: daily_vaccinations, dtype: float64
```

Total Vaccinations in Country Statistical Analysis

df_vaccinations['total_vaccinations'].describe()

Output:

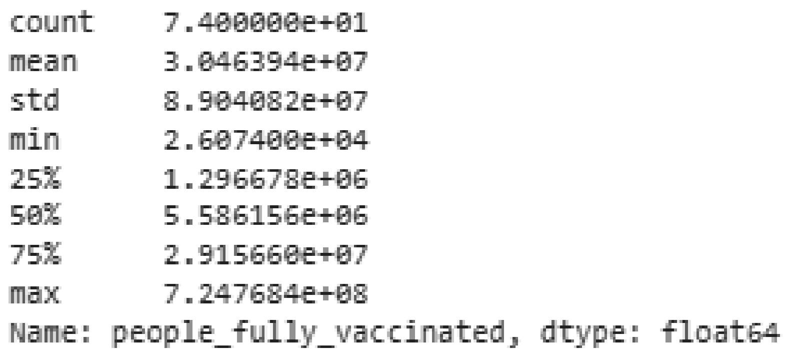


```
count    7.400000e+01
mean     7.341445e+07
std      2.094558e+08
min      6.936300e+04
25%      3.041524e+06
50%      1.300016e+07
75%      6.246783e+07
max      1.687048e+09
Name: total_vaccinations, dtype: float64
```

People Fully Vaccinated in Country Statistical Analysis

df_vaccinations['people_fully_vaccinated'].describe()

Output:

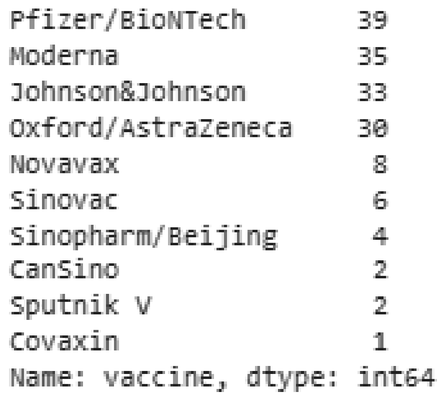


```
count    7.400000e+01
mean     3.046394e+07
std      8.904082e+07
min      2.607400e+04
25%      1.296678e+06
50%      5.586156e+06
75%      2.915660e+07
max      7.247684e+08
Name: people_fully_vaccinated, dtype: float64
```

Most Used Vaccine in the World

df_manufacturer['vaccine'].value_counts()

Output:

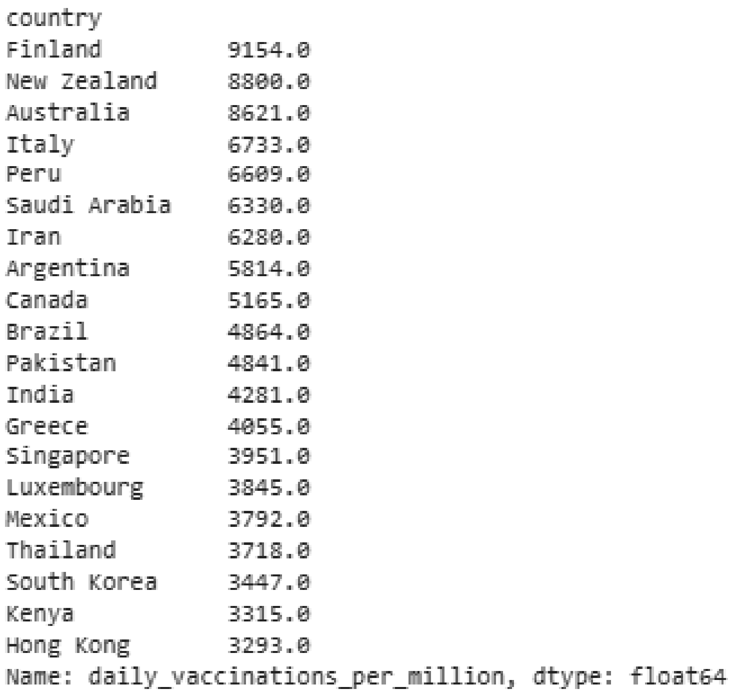


```
Pfizer/BioNTech       39
Moderna               35
Johnson&Johnson       33
Oxford/AstraZeneca    30
Novavax                8
Sinovac                6
Sinopharm/Beijing      4
CanSino                2
Sputnik V              2
Covaxin                1
Name: vaccine, dtype: int64
```

Daily Vaccinations per Million Top Countries

df_vaccinations.groupby("country")["daily_vaccinations_per_million"].mean().sort_values(ascending=False).head(20)

Output:



```
country
Finland         9154.0
New Zealand     8800.0
Australia       8621.0
Italy           6733.0
Peru            6609.0
Saudi Arabia    6330.0
Iran            6280.0
Argentina       5814.0
Canada          5165.0
Brazil          4864.0
Pakistan        4841.0
India           4281.0
Greece          4055.0
Singapore       3951.0
Luxembourg      3845.0
Mexico          3792.0
Thailand        3718.0
South Korea     3447.0
Kenya           3315.0
Hong Kong       3293.0
Name: daily_vaccinations_per_million, dtype: float64
```

Preferred Vaccine in India

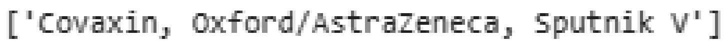
x = df_vaccinations[df_vaccinations["country"] == "India"]

z = x.vaccines.value_counts()

c = list(z.index)

print(c)

Output:



```
['Covaxin, Oxford/AstraZeneca, Sputnik V']
```

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. **Exploratory Data Analysis (EDA):**
    - The EDA section explores the structure and content of the data, presenting key information such as columns, shapes, and data types.

- It also includes the preprocessing steps, checking and handling missing values, and ensuring data quality.

3. **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.
4. **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.
5. **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.
6. **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.

Implementation - Phase 4 of COVID Vaccine Analysis:

https://colab.research.google.com/drive/18iN5o_u16y5msl0uB-RTOKWbWn3n8W33?usp=sharing