COVID-19-World-Vaccination-Progress-Exploratory-Data-Analysis-Armaan-Wadhwa

About *I conducted a preliminary exploration of the data using the Pandas and Matplotlib libraries. The dataset I used was obtained from Kaggle and focused on the progress of COVID-19 vaccinations worldwide between December 2020 and March 2022. During the analysis, I performed tasks such as cleaning and organizing the data, as well as visualizing it using various charts and graphs. The purpose was to investigate specific hypotheses and gain insights from the dataset through this initial exploration.*

EDA *Exploratory Data Analysis is an approach used in data science and statistics to analyze and summarize datasets in order to gain insights and understanding about the data. EDA involves examining the main characteristics of the data and identifying patterns and trends.*

Data Source *This dataframe includes country level vaccination data which is gathered and assembled in one single file. This dataframe was taken from Kaggle. URL: https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-progress*

Dataset Information *The dataset includes the following information:*

1. *Country: The country for which the vaccination information is provided.*
2. *Country ISO Code: The ISO code representing the country.*
3. *Date: The date of the data entry. For some dates, only the daily vaccinations are available, while for others, only the cumulative total is provided.*
4. *Total number of vaccinations: The absolute number of total immunizations in the country.*
5. *Total number of people vaccinated: The number of individuals who have received one or more vaccines. This number may be higher than the total population, considering that some people receive multiple doses.*
6. *Total number of people fully vaccinated: The number of individuals who have received the complete set of recommended vaccine doses.*
7. *Daily vaccinations (raw): The number of vaccinations administered on a particular date in a specific country, but information has not yet been processed by a machine or individual.*
8. *Daily vaccinations: The number of vaccinations administered on a particular date in a specific country.*
9. *Total vaccinations per hundred: The ratio (in percentage) of the vaccination number to the total population up to the given date in the country.*
10. *Total number of people vaccinated per hundred: The ratio (in percentage) of the immunized population to the total population up to the given date in the country.*
11. *Total number of people fully vaccinated per hundred: The ratio (in percentage) of the fully immunized population to the total population up to the given date in the country.*
12. *Daily vaccinations per million: The ratio (in parts per million) of the vaccination number to the total population on the current date in the country.*
13. *Vaccines used in the country: The total number of different vaccines used in the country up to the given date.*
14. *Source name: The name of the source providing the information (e.g., national authority, international organization, local organization, etc.).*
15. *Source website: The website of the source providing the information.*

Why is an EDA about this Specific Dataset Useful?

- *Understanding global vaccination progress: The dataset provides information on COVID-19 vaccination progress across different countries. EDA allows us to analyze and summarize this data, enabling a better* *understanding of the global vaccination landscape*

- *Identifying trends and patterns: EDA helps identify trends and patterns in the vaccination data*
- *Overall, the EDA of this dataset provides valuable insights into the progress, challenges, and impact of COVID-19 vaccination efforts worldwide, contributing to a better understanding of the global response to the pandemic and guiding future strategies for public health*

Downloading Dataset

```
pip install opendatasets --upgrade --quiet
```

```
import opendatasets as op
url_download = 'https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-progress
op.download(url_download)
```

```
Skipping, found downloaded files in "./covid-world-vaccination-progress" (use
force=True to force download)
```

```
file_name = './covid-world-vaccination-progress/country_vaccinations.csv'
```

Installing and Importing Packages

```
# Install the pandas_profiling library
!pip install pandas_profiling --quiet
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 324.4/324.4 kB 6.9 MB/s eta 0:00:00a
0:00:01
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 352.3/352.3 kB 30.0 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 102.7/102.7 kB 14.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 679.5/679.5 kB 63.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 296.5/296.5 kB 35.8 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 455.4/455.4 kB 48.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.7/4.7 MB 66.7 MB/s eta 0:00:00
  Building wheel for htmlmin (setup.py) ... done
```

```
# Import the pandas library for data manipulation and analysis
import pandas as pd
# Import the pandas_profiling library for generating data reports
from pandas_profiling import ProfileReport
```

```
<ipython-input-9-048fe6347e0c>:4: DeprecationWarning: `import pandas_profiling` is
going to be deprecated by April 1st. Please use `import ydata_profiling` instead.
  from pandas_profiling import ProfileReport
```

```
# Load the necessary libraries
import matplotlib.pyplot as plt
```

```python
# Import the Plotly Express library as px for easy and interactive data visualization
import plotly.express as px
```

Overview of Data

```python
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(file_name)
```

```python
df.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
```

```python
df.head()
```

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	NaN
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	NaN
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	NaN

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN	NaN
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN	NaN

```
df.tail()
```

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_r
86507	Zimbabwe	ZWE	2022-03-25	8691642.0	4814582.0	3473523.0	13921
86508	Zimbabwe	ZWE	2022-03-26	8791728.0	4886242.0	3487962.0	10008
86509	Zimbabwe	ZWE	2022-03-27	8845039.0	4918147.0	3493763.0	5331
86510	Zimbabwe	ZWE	2022-03-28	8934360.0	4975433.0	3501493.0	8932
86511	Zimbabwe	ZWE	2022-03-29	9039729.0	5053114.0	3510256.0	10536

```
# Perform data profiling and generate a report
Data_Report = ProfileReport(df)

# Display the generated data report
Data_Report
```

```
Output hidden; open in https://colab.research.google.com to view.
```

Data Preperation What does it Mean? *Data preparation refer to the process of transforming raw data into a clean, structured, and usable format for analysis. Data preparation is crucial because the quality of the data directly affects the reliability and validity of any insights or conclusions drawn from it.*

Steps it Involves

1. *Data cleaning: This involves identifying and handling missing values, outliers, duplicates, and inconsistencies in the data. Missing values may be filled or imputed using appropriate methods, outliers may be treated or analyzed separately, duplicates may be removed, and inconsistencies may be resolved through standardization or data correction techniques.*
2. *Data transformation: Data transformation involves converting data into a suitable format for analysis.*
3. *Data integration: If multiple datasets are involved, data integration ensures that the datasets are combined or linked appropriately. But, as we only have one dataset this step won't be neccessary.*
4. *Data validation: Data validation involves checking the quality and integrity of the data by performing checks for logical inconsistencies or errors. This step ensures that the data is reliable and suitable for analysis. We will be checking our steps throughout.*

Data Cleaning and Validation

Dropping Coloumns that are not Neccessary for my Analysis *The reason for dropping these columns are listed below:-*

- *Source Website - Does not contain data neccessary for analysis*
- *Source Name - Just required to verify information, not for analysis*
- *Daily Vaccination Raw - Contains a lot of missing data and might contain invalid information*
- *Daily Vaccinations per Million - Using Daily Vaccinations instead*
- *Total Vaccinations per Hundred - Using Total vaccinations instead*
- *People Fully Vaccinated per Hundred - Using People Fully Vaccinated instead*
- *People Vaccinated per Hundred - Using People Vaccinated instead*

```
# Dropping unnecessary columns from the DataFrame
df = df.drop(['source_website','source_name','daily_vaccinations_per_million',
         'people_fully_vaccinated_per_hundred','people_vaccinated_per_hundred',
         'total_vaccinations_per_hundred','daily_vaccinations_raw',
         ],axis = 1)
```

Visualizing Null Values

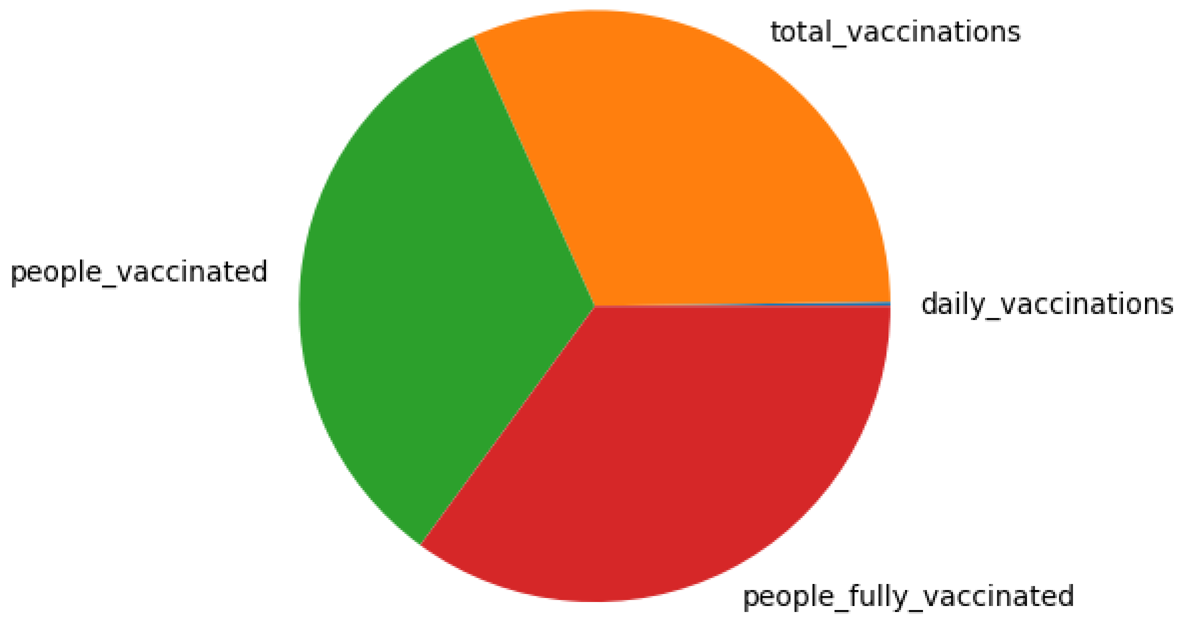
We will address the missing values in order to enhance the ease and clarity of data visualization

```
# Calculate the number of missing values in each column and sort them in ascending orde
missing_values = df.isna().sum().sort_values()

# Display the resulting missing values count
missing_values
```

```
country                        0
iso_code                       0
date                           0
vaccines                       0
daily_vaccinations           299
total_vaccinations         42905
people_vaccinated          45218
people_fully_vaccinated    47710
dtype: int64
```

```
# Plot a pie chart to visualize the distribution of missing values where the values are
missing_values[missing_values != 0].plot(kind='pie')
```

```
<Axes: >
```



Let's first drop the rows that include missing values in the total_vaccinations column

```
# Drop rows where total_vaccinations is missing
df = df.drop(df[df.total_vaccinations.isna()].index)

# Check for missing values in the DataFrame
df.isna().sum()
```

```
country                       0
iso_code                      0
date                          0
total_vaccinations            0
people_vaccinated          2717
people_fully_vaccinated    5097
daily_vaccinations          223
vaccines                      0
dtype: int64
```

Let's now work on the daily vaccinations column and fill the missing values with '0' as we will assume that no vaccinations were used that day

```
# Fill missing values in the 'daily_vaccinations' column with 0
df.daily_vaccinations = df.daily_vaccinations.fillna(0)

# Check the count of missing values in each column
df.isna().sum()
```

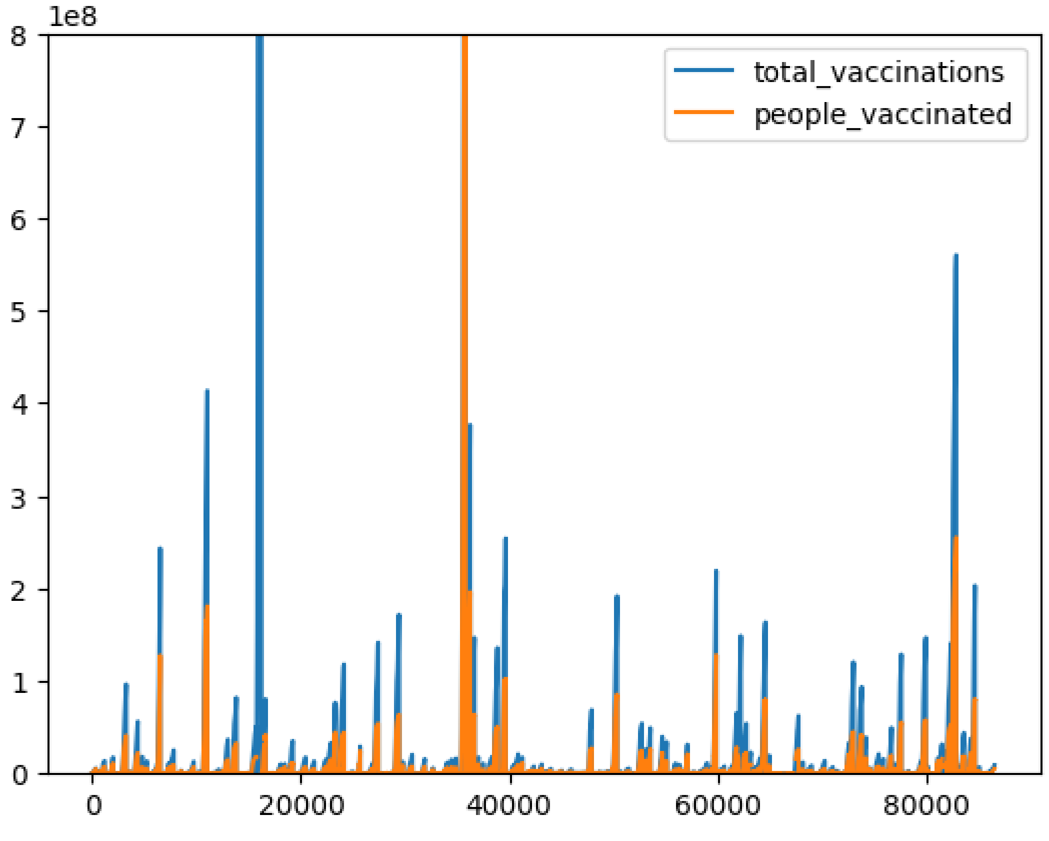
```
country                       0
iso_code                      0
date                          0
total_vaccinations            0
```

```
people_vaccinated          2717
people_fully_vaccinated    5097
daily_vaccinations            0
vaccines                      0
dtype: int64
```

Next, as you can see from the line chart, the data of the total_vaccinations column and the people_vaccinated column look almost the same

Therefore, we can calculate the difference between the mean value of the "total_vaccinations" column and the mean value of the "people_vaccinated" column and subtract this from the corresponding total_vaccinations to fill in an estimated value in exchange for the missing values in the 'people_vaccinated' column.

```
# Plotting the columns 'total_vaccinations' and 'people_vaccinated' on the same chart
df.plot(y=['total_vaccinations', 'people_vaccinated'])

# Setting the y-axis limits from 0 to 800,000,000
plt.ylim(0, 800000000)

# Displaying the chart
plt.show()
```



```
# Remove rows with missing values in the 'people_vaccinated' column
new_df = df.drop(df[df.people_vaccinated.isna()].index)

# Calculate the difference between the mean of 'total_vaccinations' and 'people_vaccina
diff = new_df.total_vaccinations.mean() - new_df.people_vaccinated.mean()

# Fill the missing values in 'people_vaccinated' column with 'total_vaccinations' minus
```

```
df.people_vaccinated = df.people_vaccinated.fillna(df.total_vaccinations - diff)

# Check the number of missing values in each column
df.isna().sum()
```

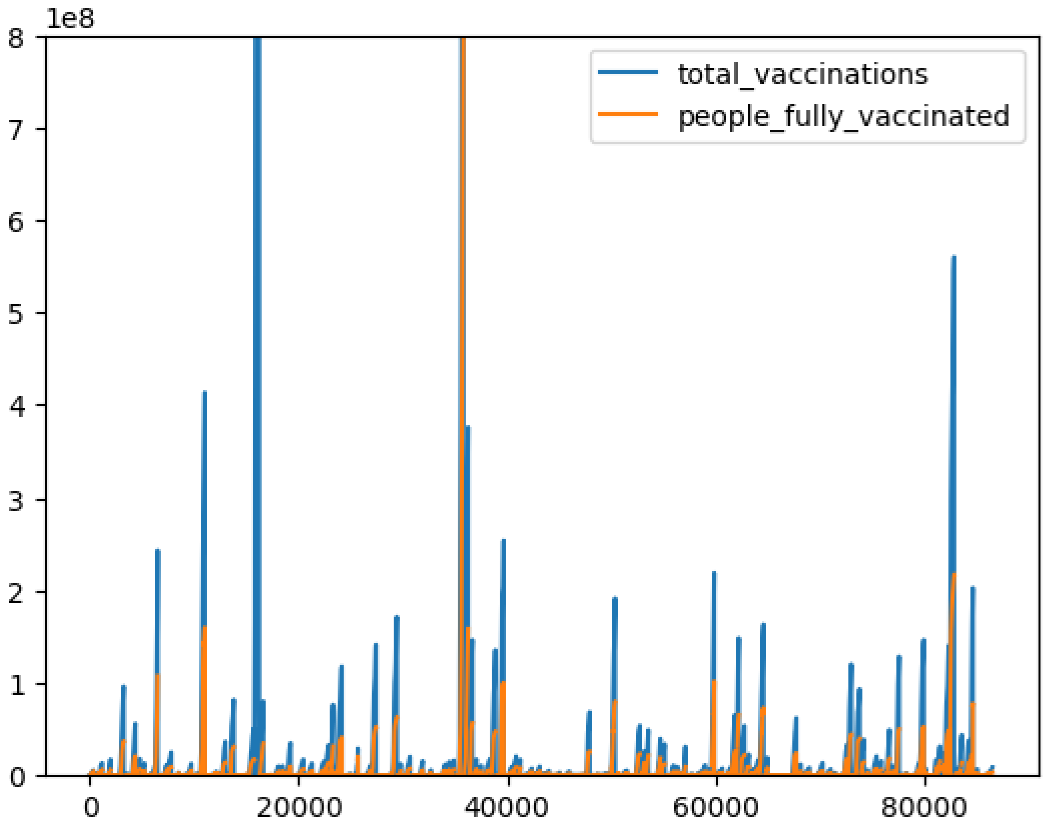
```
country                      0
iso_code                     0
date                         0
total_vaccinations           0
people_vaccinated            0
people_fully_vaccinated   5097
daily_vaccinations           0
vaccines                     0
dtype: int64
```

We have now filled all the missing values in the 'people_vaccinated' column

As you can see from the line chart, the data of the total_vaccinations column and the people_fully_vaccinated column look almost the same

Therefore, we can calculate the difference between the mean value of the "total_vaccinations" column and the mean value of the "people_fully_vaccinated" column and subtract this from the corresponding total_vaccinations to fill in an estimated value in exchange for the missing values in the 'people_fully_vaccinated' column

```
# Plot the columns 'total_vaccinations' and 'people_fully_vaccinated' from the DataFram
df.plot(y=['total_vaccinations','people_fully_vaccinated'])

# Set the y-axis limits to be between 0 and 800,000,000
plt.ylim(0,800000000)

# Show the plot
plt.show()
```



```
# Drop rows where 'people_fully_vaccinated' is missing
new_df = df.drop(df[df.people_fully_vaccinated.isna()].index)

# Calculate the difference between the mean of 'total_vaccinations' and 'people_fully_v
diff = new_df.total_vaccinations.mean() - new_df.people_fully_vaccinated.mean()

# Fill missing values in 'people_fully_vaccinated' with 'total_vaccinations' minus the
df.people_fully_vaccinated = df.people_fully_vaccinated.fillna(df.total_vaccinations -

# Check for missing values in the dataframe
df.isna().sum()
```

```
country                    0
iso_code                   0
date                       0
total_vaccinations         0
people_vaccinated          0
people_fully_vaccinated    0
daily_vaccinations         0
vaccines                   0
dtype: int64
```

As you can see, there is no more missing data in our dataframe, it has been replaced or cleared and now we can move forward without worrying about it affecting our analysis

Data Transformation

ISO_CODES *ISO Codes are internationally recognized codes used to represent countries, territories, and regions. These codes are standardized by the International Organization for Standardization (ISO). Below, we can see that* *in this dataframe there are 223 independent ISO codes, although the U.N only recognises 195 countries. But this dataframe includes both recognized sovereign countries and some dependent territories, that are not universally recognized as independent nations. To clarify, this dataframe includes not only the 195 recognized sovereign countries but also several dependent territories*

```
df.iso_code.unique().size
```

```
223
```

```
# Display unique ISO codes
display_iso_codes = df.iso_code.unique()
display_iso_codes
```

```
array(['AFG', 'ALB', 'DZA', 'AND', 'AGO', 'AIA', 'ATG', 'ARG', 'ARM',
       'ABW', 'AUS', 'AUT', 'AZE', 'BHS', 'BHR', 'BGD', 'BRB', 'BLR',
       'BEL', 'BLZ', 'BEN', 'BMU', 'BTN', 'BOL', 'BES', 'BIH', 'BWA',
       'BRA', 'VGB', 'BRN', 'BGR', 'BFA', 'BDI', 'KHM', 'CMR', 'CAN',
       'CPV', 'CYM', 'CAF', 'TCD', 'CHL', 'CHN', 'COL', 'COM', 'COG',
       'COK', 'CRI', 'CIV', 'HRV', 'CUB', 'CUW', 'CYP', 'CZE', 'COD',
       'DNK', 'DJI', 'DMA', 'DOM', 'ECU', 'EGY', 'SLV', 'OWID_ENG', 'GNQ',
       'EST', 'SWZ', 'ETH', 'FRO', 'FLK', 'FJI', 'FIN', 'FRA', 'PYF',
       'GAB', 'GMB', 'GEO', 'DEU', 'GHA', 'GIB', 'GRC', 'GRL', 'GRD',
       'GTM', 'GGY', 'GIN', 'GNB', 'GUY', 'HTI', 'HND', 'HKG', 'HUN',
       'ISL', 'IND', 'IDN', 'IRN', 'IRQ', 'IRL', 'IMN', 'ISR', 'ITA',
       'JAM', 'JPN', 'JEY', 'JOR', 'KAZ', 'KEN', 'KIR', 'OWID_KOS', 'KWT',
       'KGZ', 'LAO', 'LVA', 'LBN', 'LSO', 'LBR', 'LBY', 'LIE', 'LTU',
       'LUX', 'MAC', 'MDG', 'MWI', 'MYS', 'MDV', 'MLI', 'MLT', 'MRT',
       'MUS', 'MEX', 'MDA', 'MCO', 'MNG', 'MNE', 'MSR', 'MAR', 'MOZ',
       'MMR', 'NAM', 'NRU', 'NPL', 'NLD', 'NCL', 'NZL', 'NIC', 'NER',
       'NGA', 'NIU', 'MKD', 'OWID_CYN', 'OWID_NIR', 'NOR', 'OMN', 'PAK',
       'PSE', 'PAN', 'PNG', 'PRY', 'PER', 'PHL', 'PCN', 'POL', 'PRT',
       'QAT', 'ROU', 'RUS', 'RWA', 'SHN', 'KNA', 'LCA', 'VCT', 'WSM',
       'SMR', 'STP', 'SAU', 'OWID_SCT', 'SEN', 'SRB', 'SYC', 'SLE', 'SGP',
       'SXM', 'SVK', 'SVN', 'SLB', 'SOM', 'ZAF', 'KOR', 'SSD', 'ESP',
       'LKA', 'SDN', 'SUR', 'SWE', 'CHE', 'SYR', 'TWN', 'TJK', 'TZA',
       'THA', 'TLS', 'TGO', 'TKL', 'TON', 'TTO', 'TUN', 'TUR', 'TKM',
       'TCA', 'TUV', 'UGA', 'UKR', 'ARE', 'GBR', 'USA', 'URY', 'UZB',
       'VUT', 'VEN', 'VNM', 'OWID_WLS', 'WLF', 'YEM', 'ZMB', 'ZWE'],
      dtype=object)
```

Next, as we can see in the above iso codes, all the codes are in alpha 3 format, but some are not, so lets replace them with the correct format. For example, OWID_ENG is for England, so we will replace it with ENG

```
# Replace 'OWID_ENG' with 'ENG'
df.iso_code = df.iso_code.replace('OWID_ENG', 'ENG')
 # Replace 'OWID_NIR' with 'NIR'
df.iso_code = df.iso_code.replace('OWID_NIR', 'NIR')
# Replace 'OWID_WLS' with 'WLS'
df.iso_code = df.iso_code.replace('OWID_WLS', 'WLS')
# Replace 'OWID_SCT' with 'SCT'
```

```
df.iso_code = df.iso_code.replace('OWID_SCT', 'SCT')
# Replace 'OWID_KOS' with 'KOS'
df.iso_code = df.iso_code.replace('OWID_KOS', 'KOS')
# Replace 'OWID_CYN' with 'CYN'
df.iso_code = df.iso_code.replace('OWID_CYN', 'CYN')
```

```
df.iso_code.unique()
```

```
array(['AFG', 'ALB', 'DZA', 'AND', 'AGO', 'AIA', 'ATG', 'ARG', 'ARM',
       'ABW', 'AUS', 'AUT', 'AZE', 'BHS', 'BHR', 'BGD', 'BRB', 'BLR',
       'BEL', 'BLZ', 'BEN', 'BMU', 'BTN', 'BOL', 'BES', 'BIH', 'BWA',
       'BRA', 'VGB', 'BRN', 'BGR', 'BFA', 'BDI', 'KHM', 'CMR', 'CAN',
       'CPV', 'CYM', 'CAF', 'TCD', 'CHL', 'CHN', 'COL', 'COM', 'COG',
       'COK', 'CRI', 'CIV', 'HRV', 'CUB', 'CUW', 'CYP', 'CZE', 'COD',
       'DNK', 'DJI', 'DMA', 'DOM', 'ECU', 'EGY', 'SLV', 'ENG', 'GNQ',
       'EST', 'SWZ', 'ETH', 'FRO', 'FLK', 'FJI', 'FIN', 'FRA', 'PYF',
       'GAB', 'GMB', 'GEO', 'DEU', 'GHA', 'GIB', 'GRC', 'GRL', 'GRD',
       'GTM', 'GGY', 'GIN', 'GNB', 'GUY', 'HTI', 'HND', 'HKG', 'HUN',
       'ISL', 'IND', 'IDN', 'IRN', 'IRQ', 'IRL', 'IMN', 'ISR', 'ITA',
       'JAM', 'JPN', 'JEY', 'JOR', 'KAZ', 'KEN', 'KIR', 'KOS', 'KWT',
       'KGZ', 'LAO', 'LVA', 'LBN', 'LSO', 'LBR', 'LBY', 'LIE', 'LTU',
       'LUX', 'MAC', 'MDG', 'MWI', 'MYS', 'MDV', 'MLI', 'MLT', 'MRT',
       'MUS', 'MEX', 'MDA', 'MCO', 'MNG', 'MNE', 'MSR', 'MAR', 'MOZ',
       'MMR', 'NAM', 'NRU', 'NPL', 'NLD', 'NCL', 'NZL', 'NIC', 'NER',
       'NGA', 'NIU', 'MKD', 'CYN', 'NIR', 'NOR', 'OMN', 'PAK', 'PSE',
       'PAN', 'PNG', 'PRY', 'PER', 'PHL', 'PCN', 'POL', 'PRT', 'QAT',
       'ROU', 'RUS', 'RWA', 'SHN', 'KNA', 'LCA', 'VCT', 'WSM', 'SMR',
       'STP', 'SAU', 'SCT', 'SEN', 'SRB', 'SYC', 'SLE', 'SGP', 'SXM',
       'SVK', 'SVN', 'SLB', 'SOM', 'ZAF', 'KOR', 'SSD', 'ESP', 'LKA',
       'SDN', 'SUR', 'SWE', 'CHE', 'SYR', 'TWN', 'TJK', 'TZA', 'THA',
       'TLS', 'TGO', 'TKL', 'TON', 'TTO', 'TUN', 'TUR', 'TKM', 'TCA',
       'TUV', 'UGA', 'UKR', 'ARE', 'GBR', 'USA', 'URY', 'UZB', 'VUT',
       'VEN', 'VNM', 'WLS', 'WLF', 'YEM', 'ZMB', 'ZWE'], dtype=object)
```

Date *Let's now correct the format for the date column*

```
# Convert the 'date' column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Sort the DataFrame by the 'date' column in ascending order
df = df.sort_values(by="date", ascending=True)

# Convert the 'date' column back to a string format with the desired date format ("%Y-%
df["date"] = df["date"].dt.strftime("%Y-%m-%d")

# Display the first few rows of the DataFrame to check the changes
df.head()
```

country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations	
58517	Norway	NOR	2020-12-02	0.0	0.0	-2.108195e+07	0.0	
58518	Norway	NOR	2020-12-03	0.0	0.0	-2.108195e+07	0.0	
43117	Latvia	LVA	2020-12-04	1.0	1.0	-2.108195e+07	0.0	Jc M
58519	Norway	NOR	2020-12-04	0.0	0.0	-2.108195e+07	0.0	
58520	Norway	NOR	2020-12-05	0.0	0.0	-2.108195e+07	0.0	

```
df.tail()
```

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations
19609	Curacao	CUW	2022-03-29	246566.0	107539.0	98489.0	59.0
72973	South Korea	KOR	2022-03-29	120604515.0	44948629.0	44482876.0	37544.0
4017	Aruba	ABW	2022-03-29	169231.0	87884.0	81347.0	77.0
62750	Poland	POL	2022-03-29	53908363.0	22585418.0	22343826.0	14141.0
86511	Zimbabwe	ZWE	2022-03-29	9039729.0	5053114.0	3510256.0	103751.0

Therefore, we can confirm that this dataframe includes data from December 2020 to March 2022

Exploratory Analysis and Visualization What Does it Mean? *Exploratory Data Analysis (EDA) involves analyzing and summarizing data to gain insights and understand the main characteristics, patterns, and trends within the dataset. Visualizations are an essential component of EDA as they provide a way to visually represent the data and aid in understanding its patterns and relationship.*

Daily Vaccinations

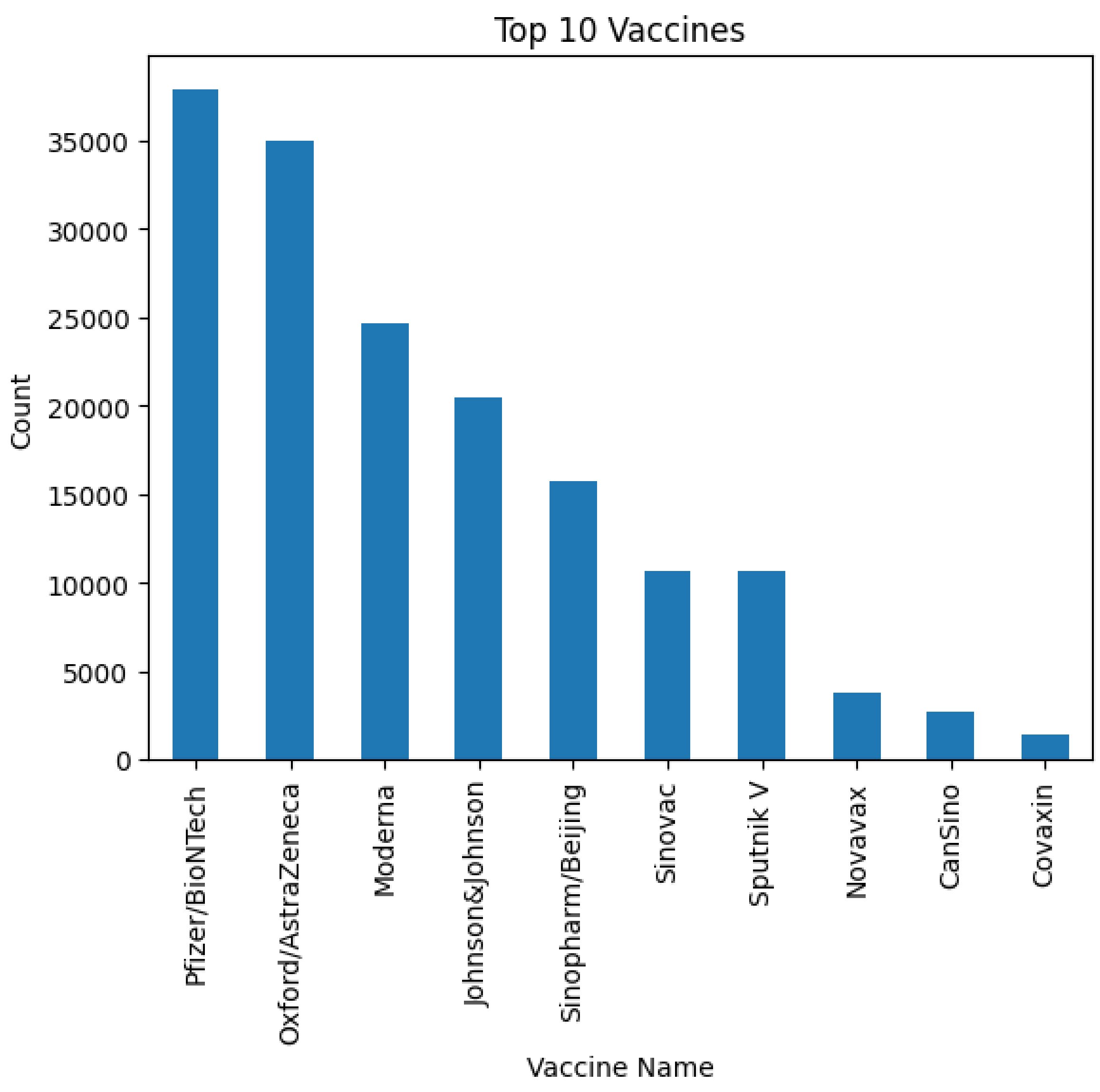
Let's first display the increase/decrease of daily vaccinations in all the countries over the years

```
# Create a choropleth map to visualize daily vaccinations per country over time
world_fig_1 = px.choropleth(df,locations="iso_code", color="daily_vaccinations", animat
    color_continuous_scale=["Pink", "cyan", "purple", "orange"], projection="natural ea
    range_color=[0, 1200000], title="Daily Vaccinations Per Country from 12-20 to 03-22

# Display the choropleth map
world_fig_1.show()
```

Vaccines

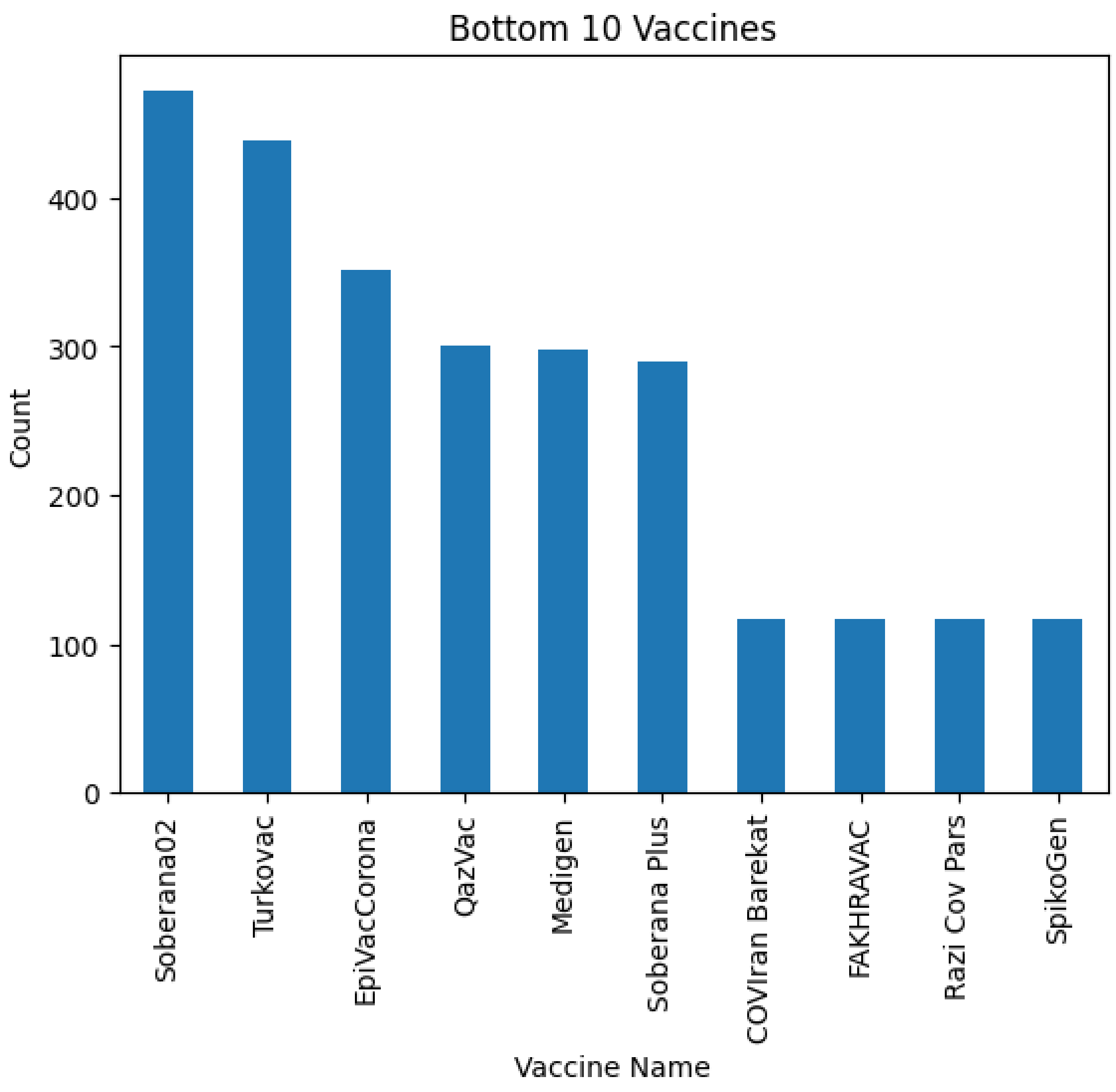
Top 10 most used vaccines around the world to get an idea about their effectiveness

```
# Splitting the 'vaccines' column into individual vaccines
df['individual_vaccines'] = df['vaccines'].str.split(', ')

# Explode the dataframe based on individual vaccines
df_exploded = df.explode('individual_vaccines')

# Count the occurrences of each individual vaccine
vaccine_counts = df_exploded['individual_vaccines'].value_counts()

# Select the top 10 most used vaccines
top_10_vaccines = vaccine_counts.head(10)

# Plotting a bar chart of the top 10 vaccines
top_10_vaccines.plot(kind='bar')

# Adding labels and title to the plot
plt.xlabel('Vaccine Name')
plt.ylabel('Count')
plt.title('Top 10 Vaccines')

# Display the plot
plt.show()
```



Top 10 least used vaccines around the world to get an idea about their effectiveness

```python
# Select the last 10 vaccines from the vaccine_counts dataframe
last_10_vaccines = vaccine_counts.tail(10)

# Plot a bar chart of the last 10 vaccines
last_10_vaccines.plot(kind='bar')

# Add labels and title to the plot
plt.xlabel('Vaccine Name')
plt.ylabel('Count')
plt.title('Bottom 10 Vaccines')

# Display the plot
plt.show()
```



CONCLUSIONS

1. *The most popular vaccine is Pfizer/BioNTech and it has been adopted in most countries, simmilarly the least popular vaccines are COVIran Barekat, FAKHRAVAC, Razi COV Pars and SpikoGen and these vaccines have been adopted in minimal countries*
2. *We can also conclude that, Pfizer, Oxford and Moderna are the top 3 most effective COVID-19 vaccines*
3. *Most developing and developed countries use a combination of the top 10 vaccines*

Countries

Now, to get an idea of the most active countries to provide vaccinations during COVID-19, let us take a look at the top 25 countries with the highest number of total vaccinations

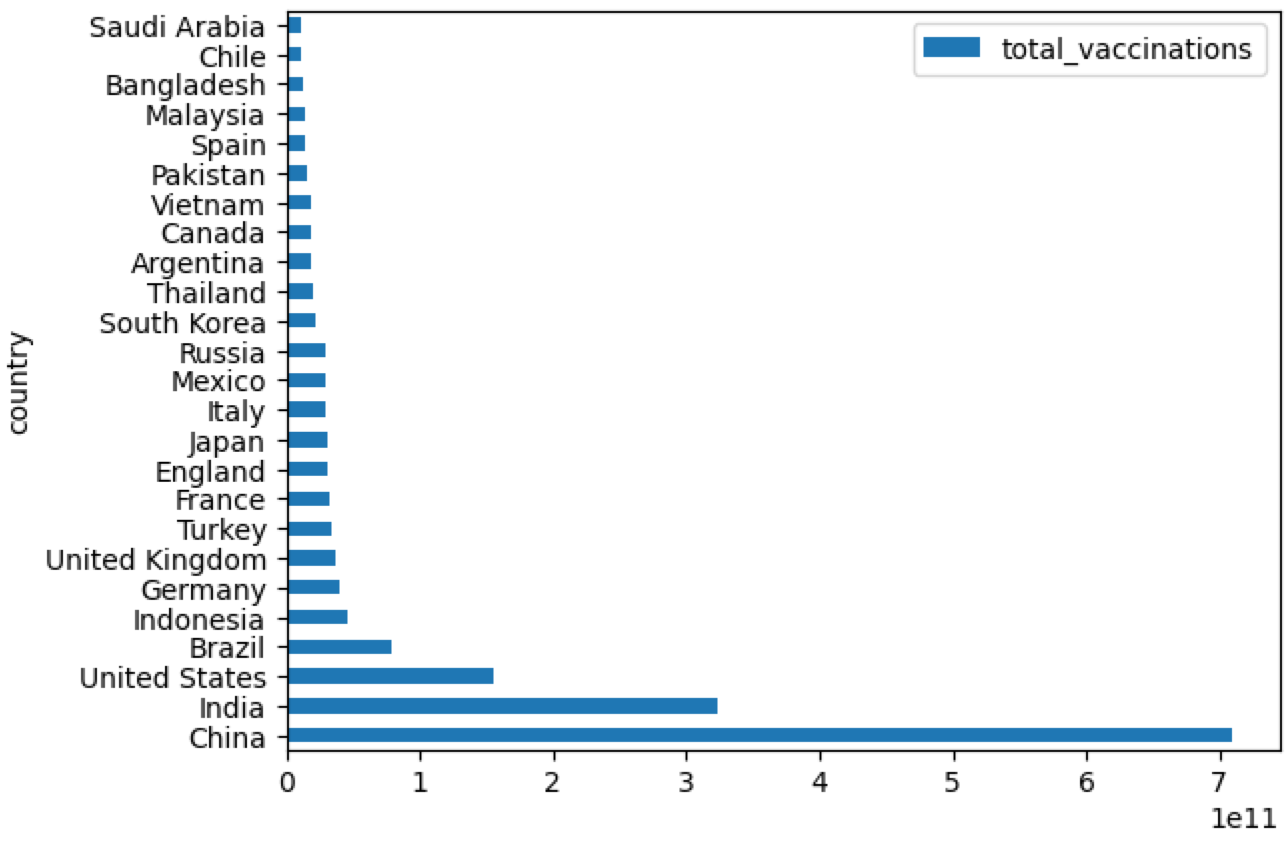
```
# Group the data by country and calculate the sum of total_vaccinations for each countr
countries = df[['total_vaccinations', 'country']].groupby('country').sum()

# Sort the countries based on the total_vaccinations in descending order
countries = countries.sort_values(by='total_vaccinations', ascending=False)

# Select the top 25 countries with the highest total_vaccinations
top_countries = countries[:25]
```

```
# Print the resulting DataFrame with the top 25 countries and their total_vaccinations
top_countries
```

	total_vaccinations
country	
China	7.094527e+11
India	3.234403e+11
United States	1.550139e+11
Brazil	7.906717e+10
Indonesia	4.521462e+10
Germany	3.898283e+10
United Kingdom	3.685164e+10
Turkey	3.392310e+10
France	3.217704e+10
England	3.103711e+10
Japan	3.030213e+10
Italy	2.971305e+10
Mexico	2.851547e+10
Russia	2.838456e+10
South Korea	2.183141e+10
Thailand	2.001835e+10
Argentina	1.867895e+10
Canada	1.822827e+10
Vietnam	1.791450e+10
Pakistan	1.567789e+10
Spain	1.386324e+10
Malaysia	1.329751e+10
Bangladesh	1.320508e+10
Chile	1.176962e+10
Saudi Arabia	1.145684e+10

```
top_countries.plot(kind='barh')
```

```
<Axes: ylabel='country'>
```



Conclusions *Interesting as India being the most populated country is second but china despite being 2nd most populated is 1st. This can be due to several factors like saccination strategies and policies, vaccine availability and production or even because of the fact that COVID-19 started in China. Another interesting point is that although United Kingdom was the first country to start administrating its citizens with a fully trialled and tested COVID-19 vaccine, it is still 7th on the graph above. This is due to the fact that United Kingdom is the 21st most populated country in the world. The rest of the data is as expected, as these countries come in the top 50 most populated countries in the world and as they are higher in population as compared to the rest of the countries, they with have a higher number of total vaccinations.*

SMART *SMART is an acronym that stands for Specific, Measurable, Achievable, Relevant, and Time-bound. It is a framework commonly used for setting clear and effective goals or objectives. When applied to data analysis, the SMART criteria help ensure that the analysis goals are well-defined, feasible, and aligned with the desired outcomes.* ***Framework***

- *Specific: How many countries are included in the dataset?*
- *Measurable: What is the total number of vaccinations administered globally?*
- *Achievable: Can we identify the top five countries with the highest total vaccinations?*
- *Relevant: Are there any notable spikes or drops in the daily vaccination numbers for specific countries?*
- *Time-bound: Identify any trends or patterns in the total and daily vaccinations progress over time?*

####***Specific #1***

```
Total_Countries = df.country.unique().size
Total_Countries
```

```
223
```

Measureable #2

```
# Calculate the total number of vaccinations administered globally
Total_Vaccines_Administered = (countries.total_vaccinations.sum().astype(float)/1000)

# Format the total number of vaccinations with commas and zero decimal places
Total_Vaccines_Administered = "{:,.0f}".format(Total_Vaccines_Administered)

# Print the total number of vaccinations administered globally
print("Total number of vaccinations administered globally:", Total_Vaccines_Administere
```

```
Total number of vaccinations administered globally: 2,002,854,014 Billion Vaccinations.
```

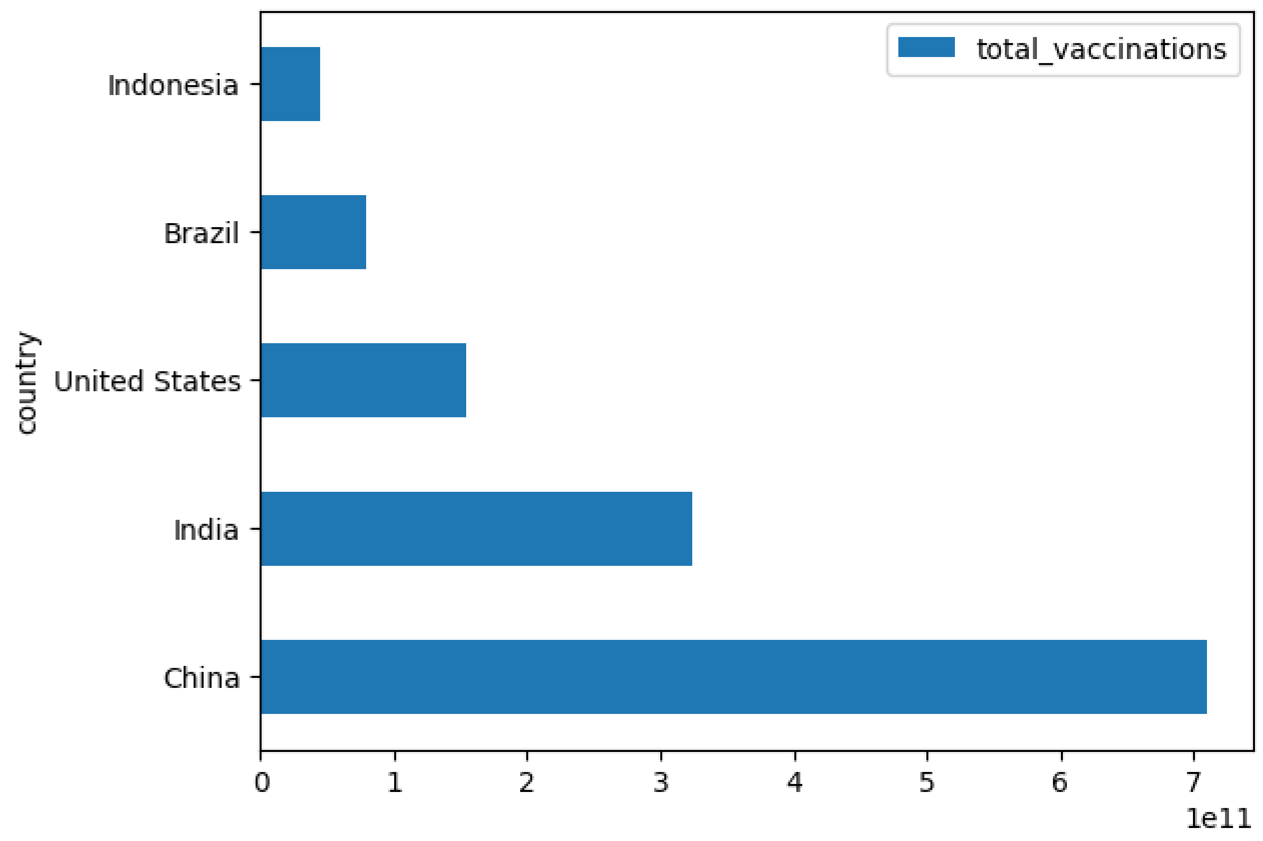
```
# Convert the 'daily_vaccinations' column to numeric, handling any errors by replacing
df['daily_vaccinations'] = pd.to_numeric(df['daily_vaccinations'], errors='coerce')

# Group the data by date and calculate the sum of daily vaccinations for each date
daily_vaccinations_sum = df.groupby('date')['daily_vaccinations'].sum()

# Calculate the average daily vaccination rate
average_daily_vaccination_rate = daily_vaccinations_sum.mean()

# Format the average daily vaccination rate with thousand separators
average_daily_vaccination_rate = "{:,.0f}".format(average_daily_vaccination_rate)

# Print the average daily vaccination rate worldwide
print("Average daily vaccination rate worldwide:", average_daily_vaccination_rate, "mil
```

```
Average daily vaccination rate worldwide: 20,467,680 million vaccinations daily
worldwide.
```

Achievable #3

```
# Selecting the top five countries based on total vaccinations and sorting them in desc
Top_Five = countries[:5].sort_values(by='total_vaccinations', ascending=False)

# Creating a horizontal bar chart to visualize the top five countries
Top_Five_Visual = Top_Five.plot(kind='barh')

# Displaying the bar chart
Top_Five_Visual
```

```
<Axes: ylabel='country'>
```



```
df.info()
```

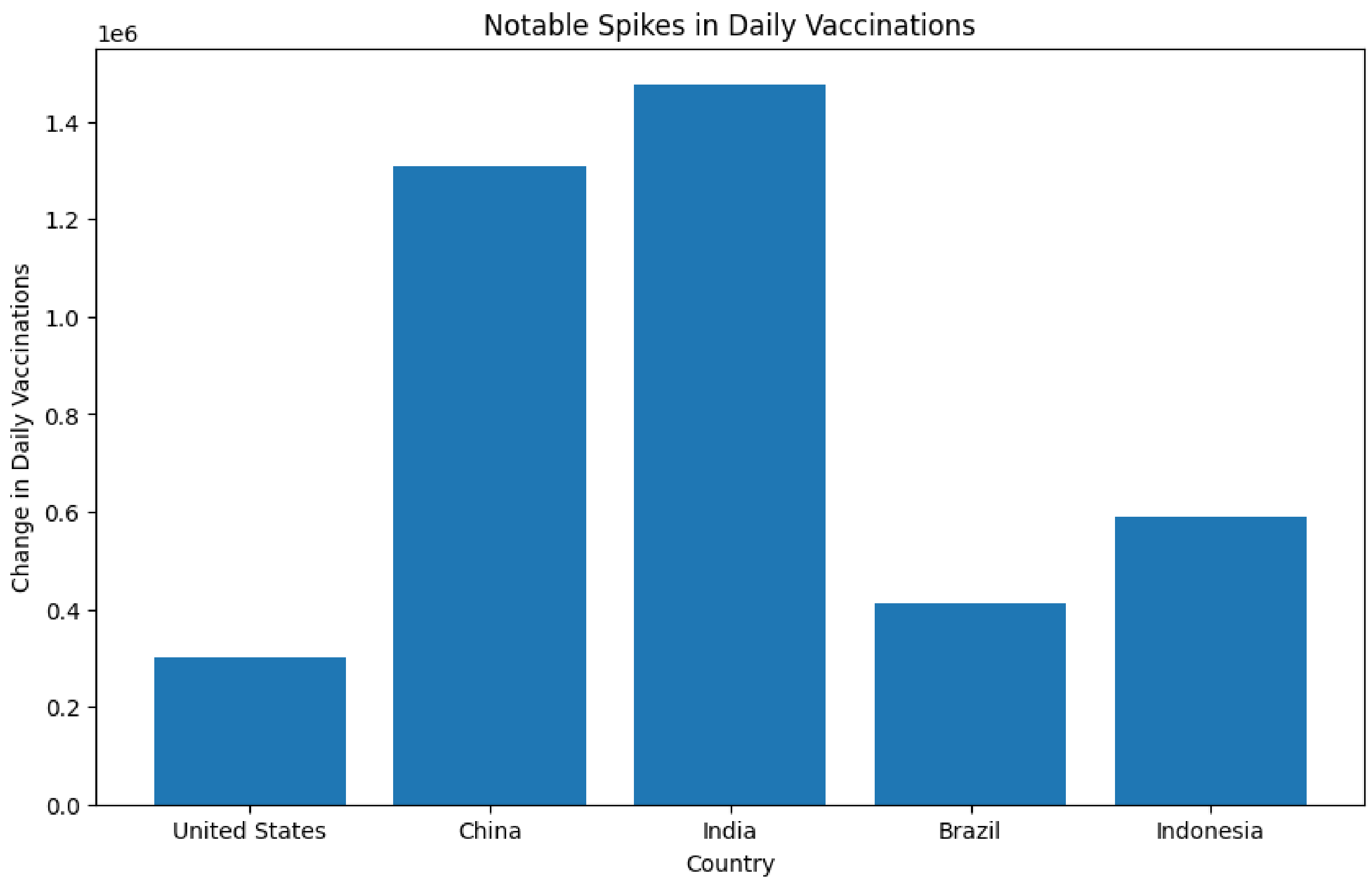
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 43607 entries, 58517 to 86511
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   country                  43607 non-null  object
 1   iso_code                 43607 non-null  object
 2   date                     43607 non-null  object
 3   total_vaccinations       43607 non-null  float64
 4   people_vaccinated        43607 non-null  float64
 5   people_fully_vaccinated  43607 non-null  float64
 6   daily_vaccinations       43607 non-null  float64
 7   vaccines                 43607 non-null  object
 8   individual_vaccines      43607 non-null  object
dtypes: float64(4), object(5)
memory usage: 3.3+ MB
```

Relevant #4

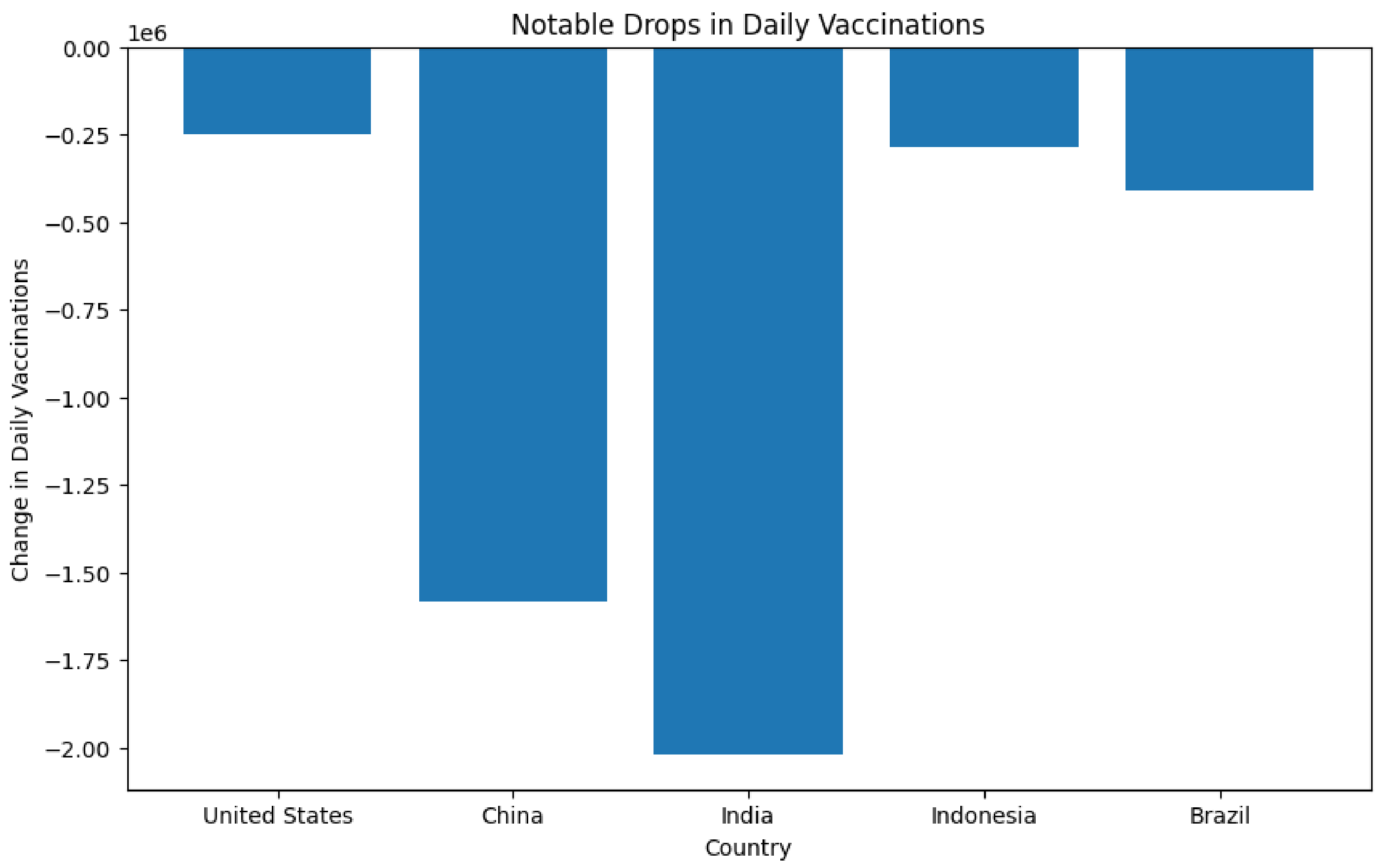
```
# Selecting data for specific countries of interest
countries_of_interest = ['China', 'India', 'United States', 'Brazil', 'Indonesia']
selected_data = df[df['country'].isin(countries_of_interest)]

# Calculating the daily change in vaccinations for each country
```

```
selected_data['daily_change'] = selected_data.groupby('country')['daily_vaccinations'].

# Define a threshold for what constitutes a notable spike
spike_threshold = 10000

# Define a threshold for what constitutes a notable drop
drop_threshold = -10000

# Selecting data points that exceed the spike threshold
spikes = selected_data[selected_data['daily_change'] > spike_threshold]

# Selecting data points that fall below the drop threshold
drops = selected_data[selected_data['daily_change'] < drop_threshold]

# Plotting the notable spikes in daily vaccinations
plt.figure(figsize=(10, 6))
plt.bar(spikes['country'], spikes['daily_change'])
plt.xlabel('Country')
plt.ylabel('Change in Daily Vaccinations')
plt.title('Notable Spikes in Daily Vaccinations')
plt.show()
```

```
<ipython-input-42-f655f3e699dc>:6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

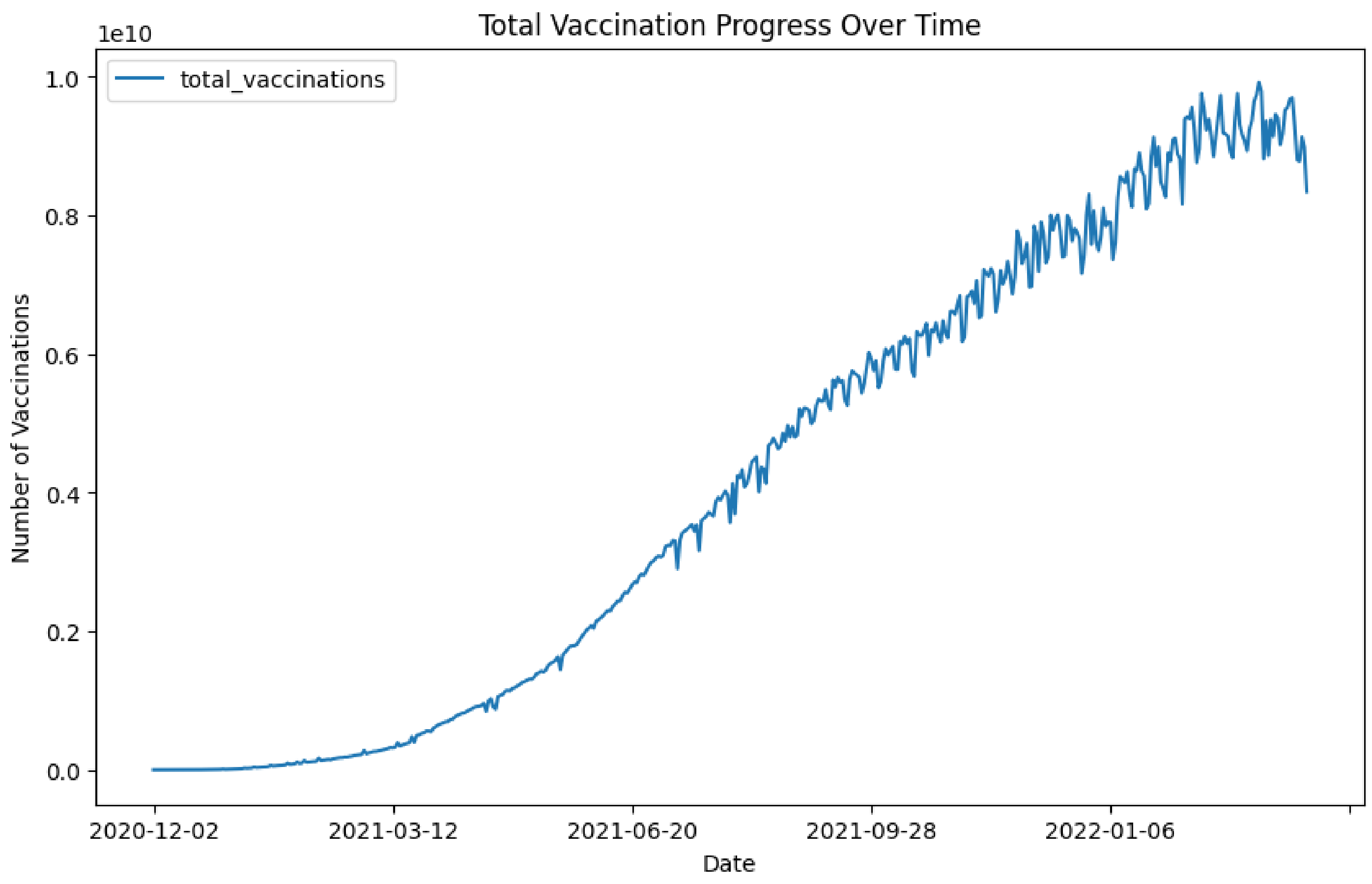


```python
# Plotting the notable drops in daily vaccinations
plt.figure(figsize=(10, 6))
plt.bar(drops['country'], drops['daily_change'])
plt.xlabel('Country')
plt.ylabel('Change in Daily Vaccinations')
plt.title('Notable Drops in Daily Vaccinations')
plt.show()
```



Time-Bound #5

```
# Group the data by date and calculate the sum of total_vaccinations for each date
grouped_data = df.groupby('date').agg({'total_vaccinations':'sum'})

# Plot the total_vaccinations over time using an area chart
grouped_data.plot(y=['total_vaccinations'], figsize=(10, 6))

# Set the x-axis label
plt.xlabel('Date')

# Set the y-axis label
plt.ylabel('Number of Vaccinations')

# Set the title of the plot
plt.title('Total Vaccination Progress Over Time')

# Show the legend
plt.legend()

# Display the plot
plt.show()
```

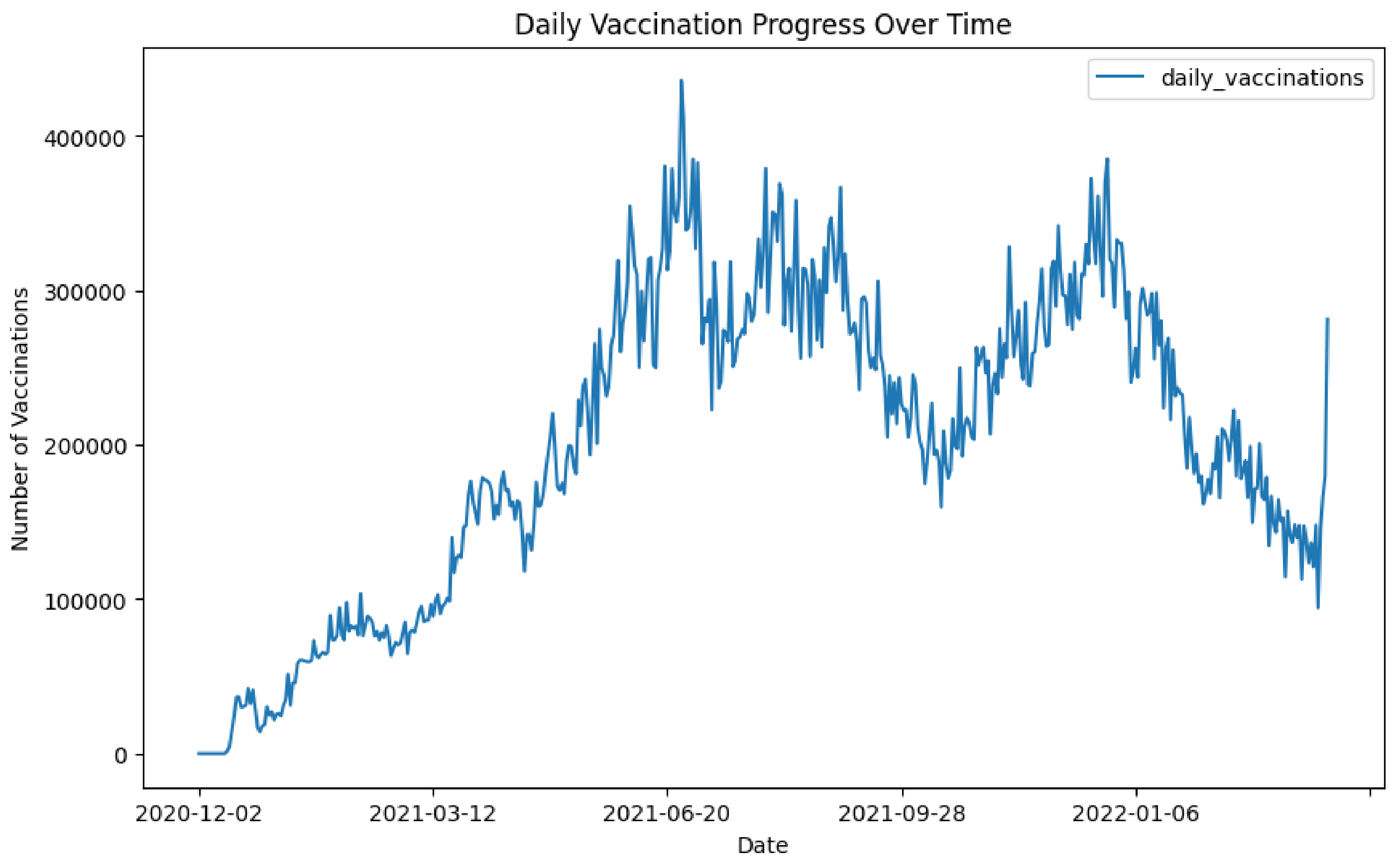


Insights

1. *The chart shows an upward trend in the data over the analyzed period*
2. *The upward movement of the line demonstrates an increase in the data points with significant fluctuations*
3. *We can see a rapid increase after 2021-05 as that was the time when the COVID vaccine was available to the general public in most countries*
4. *Although the line is ascending with significant fluctuations, it maintains a relatively stable trajectory without significant drops*
5. *The consistent upward trend, even in a static motion, suggests a steady growth pattern that indicates positive progress regarding COVID-19 vaccines*

```
# Group the data by date and calculate the mean of daily vaccinations
grouped_data = df.groupby('date').agg({'daily_vaccinations':'mean'})

# Plot the daily vaccinations over time
grouped_data.plot(y=['daily_vaccinations'], figsize=(10, 6))

# Set the x-axis label
plt.xlabel('Date')

# Set the y-axis label
plt.ylabel('Number of Vaccinations')

# Set the title of the plot
plt.title('Daily Vaccination Progress Over Time')
```

```
# Display the legend
plt.legend()

# Show the plot
plt.show()
```



###*Insights*

1. *The chart first shows an upward trend in the data from 2020-12 to 2021-06, the data then drops from 2021-06 till 2021-09 and it again follows the same trend throughout.*
2. *The upward and downward movement of the line demonstrates an increase in the data points with significant fluctuations and drops*
3. *The number of daily vaccinations globally was at its peak in 2021-06*

Summarizing Main Findings

Insights

1. *The total number of vaccinations administered globally: 2,002,854,014 Billion Vaccinations.*
2. *Average daily vaccination rate worldwide: 20,467,680 million vaccinations daily worldwide.*
3. *The top 5 countries with the highest number of vaccinations are China, India, United States, Brazil and Indonesia.*
4. *It was interesting that India being the most populated country is second but china despite being 2nd most populated is 1st. *
5. *Another interesting point is that although United Kingdom was the first country to start administrating its citizens with a fully trialled and tested COVID-19 vaccine, it is still 7th when it comes to the number of total vaccinations and this is probably because it is the 21st most populated country in the world.*

6. *The most popular vaccine is Pfizer/BioNTech and the least popular vaccines are COVIran Barekat, FAKHRAVAC, Razi COV Pars and SpikoGen.*
7. *We can also conclude that, Pfizer, Oxford and Moderna are the top 3 most effective COVID-19 vaccines*
8. *In this dataframe there are 223 independent ISO codes, although the U.N only recognises 195 countries. But this dataframe includes both recognized sovereign countries and some dependent territories, that are not universally recognized as independent nations.*