COLLEGE CODE-4212
421221243022-PHASE 5

COVID VACCINES ANALYSIS

DATA ANALYTICS WITH COGNOS -GROUP 2

INTRODUCTION

The COVID-19 pandemic has presented an unprecedented global challenge, and the development and distribution of vaccines have emerged as critical tools in the fight against the virus. This project aims to undertake a comprehensive analysis of COVID-19 vaccine data, focusing on three key aspects: vaccine efficacy, distribution, and adverse effects. The analysis is driven by the urgent need to provide evidence-based insights to support policymakers, healthcare authorities, and researchers in making informed decisions regarding vaccine deployment and safety monitoring.

In the following sections, we will outline the step-by-step approach to conducting this analysis, beginning with data collection and preprocessing, followed by exploratory data analysis, statistical modeling, and data visualization. Our ultimate goal is to extract meaningful insights from the available data that can contribute to the development of optimized vaccination strategies, equitable distribution, and enhanced safety surveillance. This project not only addresses the immediate challenges posed by the pandemic but also contributes to the broader knowledge base surrounding vaccine development and deployment in the face of global health crises.

PROBLEM STATEMENT

The problem is to conduct an in-depth analysis of Covid-19 vaccine data, focusing on vaccine efficacy, distribution, and adverse effects. The goal is to provide insights that aid policymakers and health organizations in optimizing vaccine deployment strategies. This project involves data collection, data preprocessing, exploratory data analysis, statistical analysis, and visualization.

DESIGN THINKING:

To design a model for your COVID-19 vaccine data analysis project using data analysis techniques, you can follow these step-by-step guidelines:

1.Define the Problem:

The problem is to analyze COVID-19 vaccine data to assess vaccine efficacy, distribution patterns, and adverse effects, with the goal of informing evidence-based decision-making for effective pandemic management.

2. Data Collection:

Use relevant data from trusted sources, including vaccine efficacy trials, vaccination campaign data, and adverse event reports. Ensure that the data is comprehensive and up-to-date.

3. Data Preprocessing:

Clean the data by handling missing values, outliers, and inconsistencies. Normalize or scale numerical features if necessary. Encode categorical variables and create dummy variables. Merge and join datasets as needed to create a consolidated dataset for analysis.

4. Exploratory Data Analysis (EDA):

Perform EDA to gain an initial understanding of the data. This includes: Visualizing data distributions with histograms, box plots, and scatter plots. Calculating summary statistics like mean, median, and standard deviation. Identifying correlations and relationships among variables. Detecting outliers and anomalies. Creating geographical or regional visualizations if relevant.

5. Hypothesis Formulation:

Based on your EDA findings, formulate hypotheses related to vaccine efficacy, distribution disparities, or adverse effects. These hypotheses will guide your subsequent statistical analyses.

6. Statistical Analysis:

Conduct statistical tests and analyses to test your hypotheses. This may include: T-tests or ANOVA for comparing vaccine efficacy rates between groups. Regression analysis to understand factors influencing vaccine

distribution. Chi-squared tests for analyzing associations between adverse effects and demographic variables.

7. Data Visualization:

Create clear and informative data visualizations to communicate your findings. Utilize libraries like matplotlib, seaborn, or Plotly for plotting graphs and charts. Generate geographic visualizations (e.g., choropleth maps) to illustrate regional disparities in vaccine distribution.

8. Documentation:

Document your analysis process thoroughly, including code, data sources, and methodology. Ensure that your work is transparent and replicable.

DEVELOPMENT PHASE

IMPORT NEEDED LIBRARIES

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

LOAD DATA

In [2]:
```
vaccinations_by_manufacturer = pd.read_csv("D:/ibm naan mudhalvan/country_vacc
vaccinations_data = pd.read_csv("D:/ibm naan mudhalvan/country_vaccinations.cs
```

In [3]:
```
vaccinations_data
```

Out[3]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinate
0	Afghanistan	AFG	2021-02-22	0.0	0.0	Na
1	Afghanistan	AFG	2021-02-23	NaN	NaN	Na
2	Afghanistan	AFG	2021-02-24	NaN	NaN	Na
3	Afghanistan	AFG	2021-02-25	NaN	NaN	Na
4	Afghanistan	AFG	2021-02-26	NaN	NaN	Na
...	...	...	...	...	...	.
86507	Zimbabwe	ZWE	2022-03-25	8691642.0	4814582.0	3473523.
86508	Zimbabwe	ZWE	2022-03-26	8791728.0	4886242.0	3487962.
86509	Zimbabwe	ZWE	2022-03-27	8845039.0	4918147.0	3493763.
86510	Zimbabwe	ZWE	2022-03-28	8934360.0	4975433.0	3501493.
86511	Zimbabwe	ZWE	2022-03-29	9039729.0	5053114.0	3510256.

86512 rows × 15 columns

In [4]:
```
vaccinations_by_manufacturer
```

Out[4]:

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

35623 rows × 4 columns

DATA PREPROCESSING AND EDA

In [5]:
```
vaccinations_by_manufacturer.info()
vaccinations_data.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
<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
```

In [6]:
```
vaccinations_by_manufacturer.rename(columns={'location': 'country'}, inplace=T
```

In [7]:
```
# Convert the 'date' column to datetime
vaccinations_by_manufacturer['date'] = pd.to_datetime(vaccinations_by_manufact
vaccinations_data['date'] = pd.to_datetime(vaccinations_data['date'])

# Check for common columns for merging
common_columns = ['country', 'date']

# Merge the datasets on the common columns
merged_df = pd.merge(vaccinations_data, vaccinations_by_manufacturer, on=commo
```

In [8]:
```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 28954 entries, 0 to 28953
Data columns (total 17 columns):
 #   Column                               Non-Null Count  Dtype         
---  ------                               --------------  -----         
 0   country                              28954 non-null  object        
 1   iso_code                             28954 non-null  object        
 2   date                                 28954 non-null  datetime64[ns]
 3   total_vaccinations_x                 27018 non-null  float64       
 4   people_vaccinated                    26356 non-null  float64       
 5   people_fully_vaccinated              26442 non-null  float64       
 6   daily_vaccinations_raw               26497 non-null  float64       
 7   daily_vaccinations                   28923 non-null  float64       
 8   total_vaccinations_per_hundred       27018 non-null  float64       
 9   people_vaccinated_per_hundred        26356 non-null  float64       
 10  people_fully_vaccinated_per_hundred  26442 non-null  float64       
 11  daily_vaccinations_per_million       28923 non-null  float64       
 12  vaccines                             28954 non-null  object        
 13  source_name                          28954 non-null  object        
 14  source_website                       28954 non-null  object        
 15  vaccine                              28954 non-null  object        
 16  total_vaccinations_y                 28954 non-null  int64         
dtypes: datetime64[ns](1), float64(9), int64(1), object(6)
memory usage: 4.0+ MB
```

In [9]:
```
print(vaccinations_data.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  datetime64[ns]
 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: datetime64[ns](1), float64(9), object(5)
memory usage: 9.9+ MB
None
```

In [10]:
```
print(vaccinations_by_manufacturer.describe())
```

```
       total_vaccinations
count        3.562300e+04
mean         1.508357e+07
std          5.181768e+07
min          0.000000e+00
25%          9.777600e+04
50%          1.305506e+06
75%          7.932423e+06
max          6.005200e+08
```

In [11]:
```
print(merged_df.describe())
```

```
       total_vaccinations_x  people_vaccinated  people_fully_vaccinated  \
count          2.701800e+04       2.635600e+04              2.644200e+04
mean           4.145588e+07       2.173495e+07              1.781408e+07
std            7.839012e+07       3.948117e+07              3.427411e+07
min            1.000000e+00       1.000000e+00              1.000000e+00
25%            2.568560e+06       1.526178e+06              1.001473e+06
50%            1.175507e+07       6.129479e+06              5.180098e+06
75%            4.690237e+07       2.884636e+07              1.870244e+07
max            5.601818e+08       2.553624e+08              2.174990e+08

       daily_vaccinations_raw  daily_vaccinations  \
count            2.649700e+04        2.892300e+04
mean             2.011788e+05        1.876768e+05
std              3.716247e+05        3.296631e+05
min              0.000000e+00        0.000000e+00
25%              1.525700e+04        1.749550e+04
50%              6.908900e+04        7.218400e+04
75%              2.302590e+05        2.239550e+05
max              6.586453e+06        3.506960e+06

       total_vaccinations_per_hundred  people_vaccinated_per_hundred  \
count                    27018.000000                   26356.000000
mean                        88.995541                      45.428177
std                         67.574722                      29.328559
min                          0.000000                       0.000000
25%                         23.945000                      15.550000
50%                         85.180000                      52.430000
75%                        140.610000                      71.880000
max                        261.820000                      92.960000

       people_fully_vaccinated_per_hundred  daily_vaccinations_per_million  \
count                         26442.000000                    28923.000000
mean                             37.919123                     4407.306573
std                              28.897718                     3218.155886
min                               0.000000                        0.000000
25%                               7.380000                     1850.000000
50%                              38.760000                     3611.000000
75%                              64.970000                     6471.000000
max                              91.530000                    19828.000000

       total_vaccinations_y
count          2.895400e+04
mean           1.048228e+07
std            3.127265e+07
min            0.000000e+00
25%            1.349192e+05
50%            1.162808e+06
75%            7.418144e+06
max            3.306080e+08
```

In [12]:
```
print(vaccinations_data.isnull().sum())
```

```
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
```

In [13]:
```
print(vaccinations_by_manufacturer.isnull().sum())
```

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

In [14]:
```
print(merged_df.isnull().sum())
```

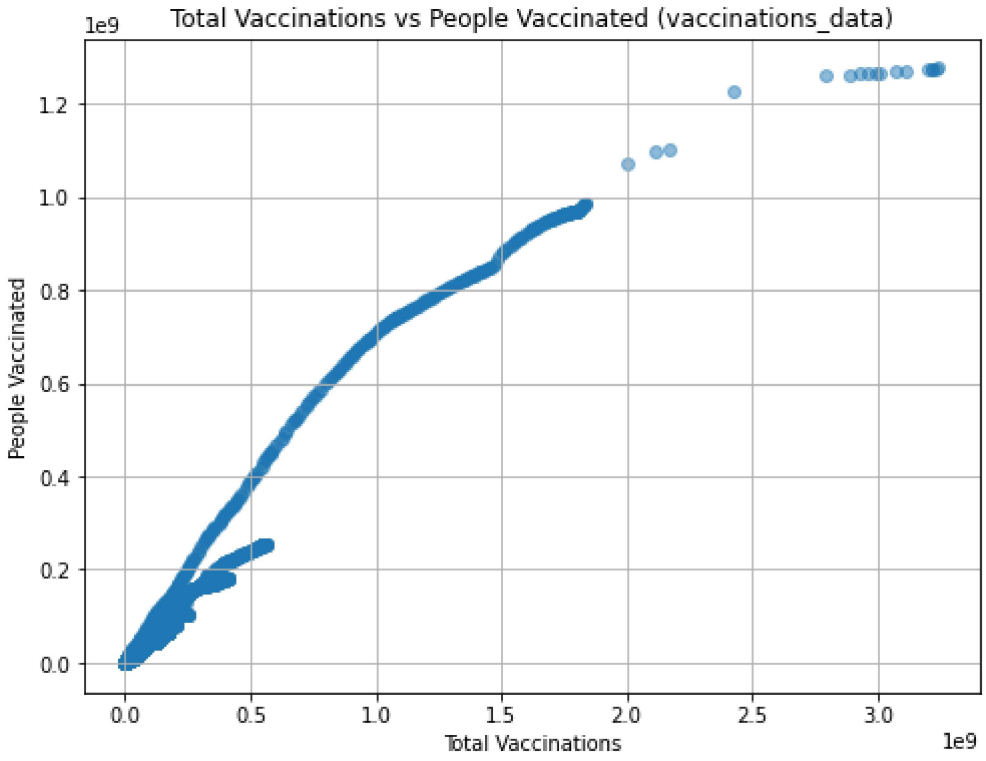
```
country                                0
iso_code                               0
date                                   0
total_vaccinations_x               1936
people_vaccinated                  2598
people_fully_vaccinated            2512
daily_vaccinations_raw             2457
daily_vaccinations                   31
total_vaccinations_per_hundred     1936
people_vaccinated_per_hundred      2598
people_fully_vaccinated_per_hundred    2512
daily_vaccinations_per_million       31
vaccines                               0
source_name                            0
source_website                         0
vaccine                                0
total_vaccinations_y                   0
dtype: int64
```

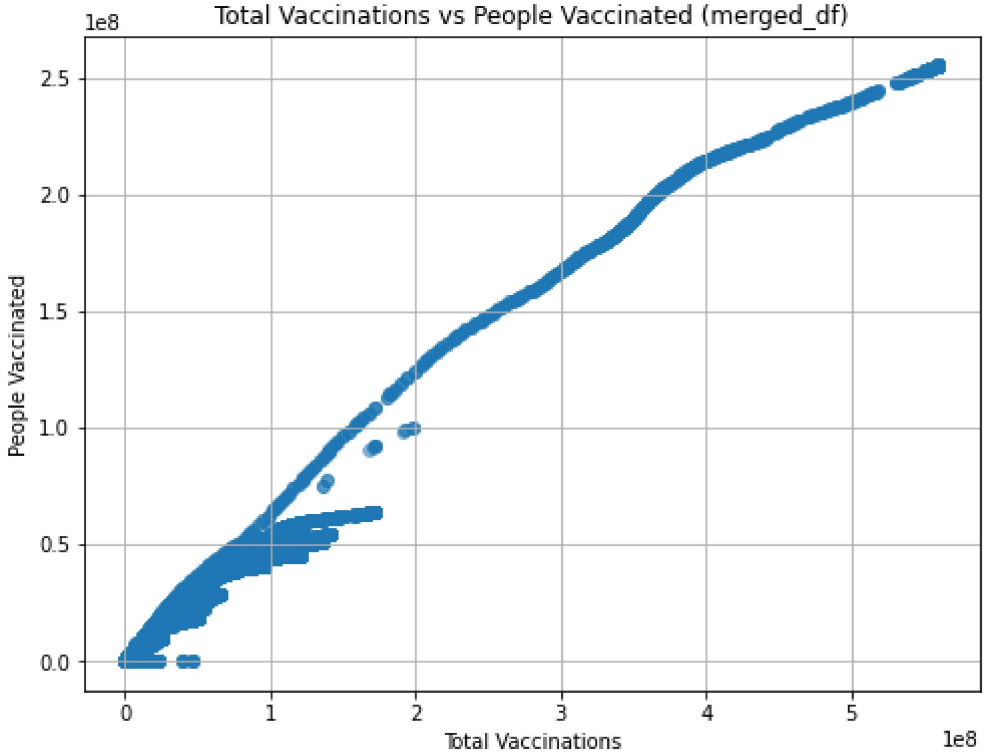
In [15]:
```
# Handle missing values
merged_df.dropna(subset=['total_vaccinations_x'], inplace=True)
merged_df.fillna(0, inplace=True)  # Replace NaN with 0 for certain columns
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 27018 entries, 0 to 28953
Data columns (total 17 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   country                               27018 non-null  object
 1   iso_code                              27018 non-null  object
 2   date                                  27018 non-null  datetime64[ns]
 3   total_vaccinations_x                  27018 non-null  float64
 4   people_vaccinated                     27018 non-null  float64
 5   people_fully_vaccinated               27018 non-null  float64
 6   daily_vaccinations_raw                27018 non-null  float64
 7   daily_vaccinations                    27018 non-null  float64
 8   total_vaccinations_per_hundred        27018 non-null  float64
 9   people_vaccinated_per_hundred         27018 non-null  float64
 10  people_fully_vaccinated_per_hundred   27018 non-null  float64
 11  daily_vaccinations_per_million        27018 non-null  float64
 12  vaccines                              27018 non-null  object
 13  source_name                           27018 non-null  object
 14  source_website                        27018 non-null  object
 15  vaccine                               27018 non-null  object
 16  total_vaccinations_y                  27018 non-null  int64
dtypes: datetime64[ns](1), float64(9), int64(1), object(6)
memory usage: 3.7+ MB
```

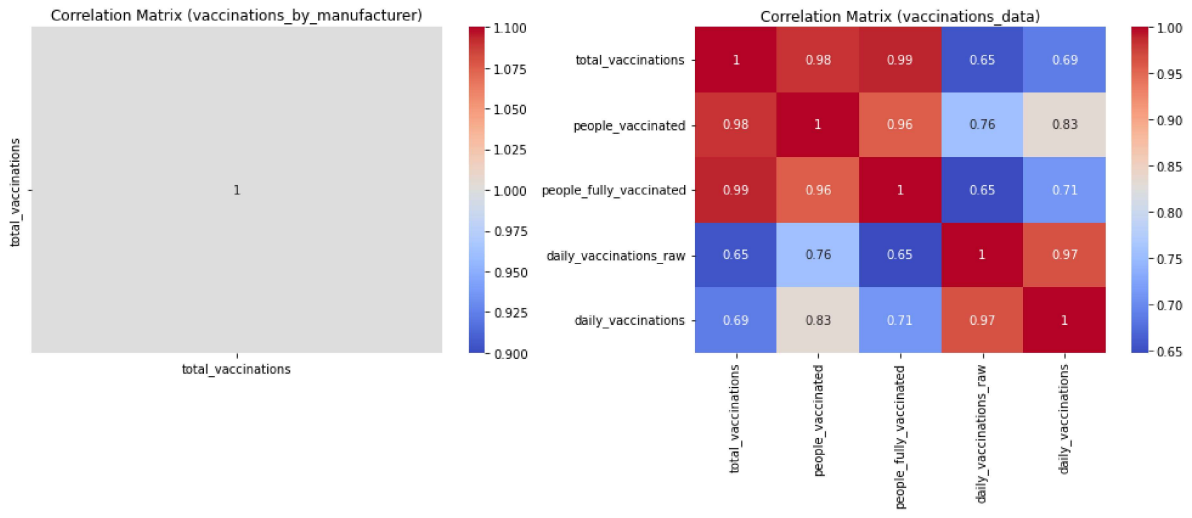
VISUALIZATION

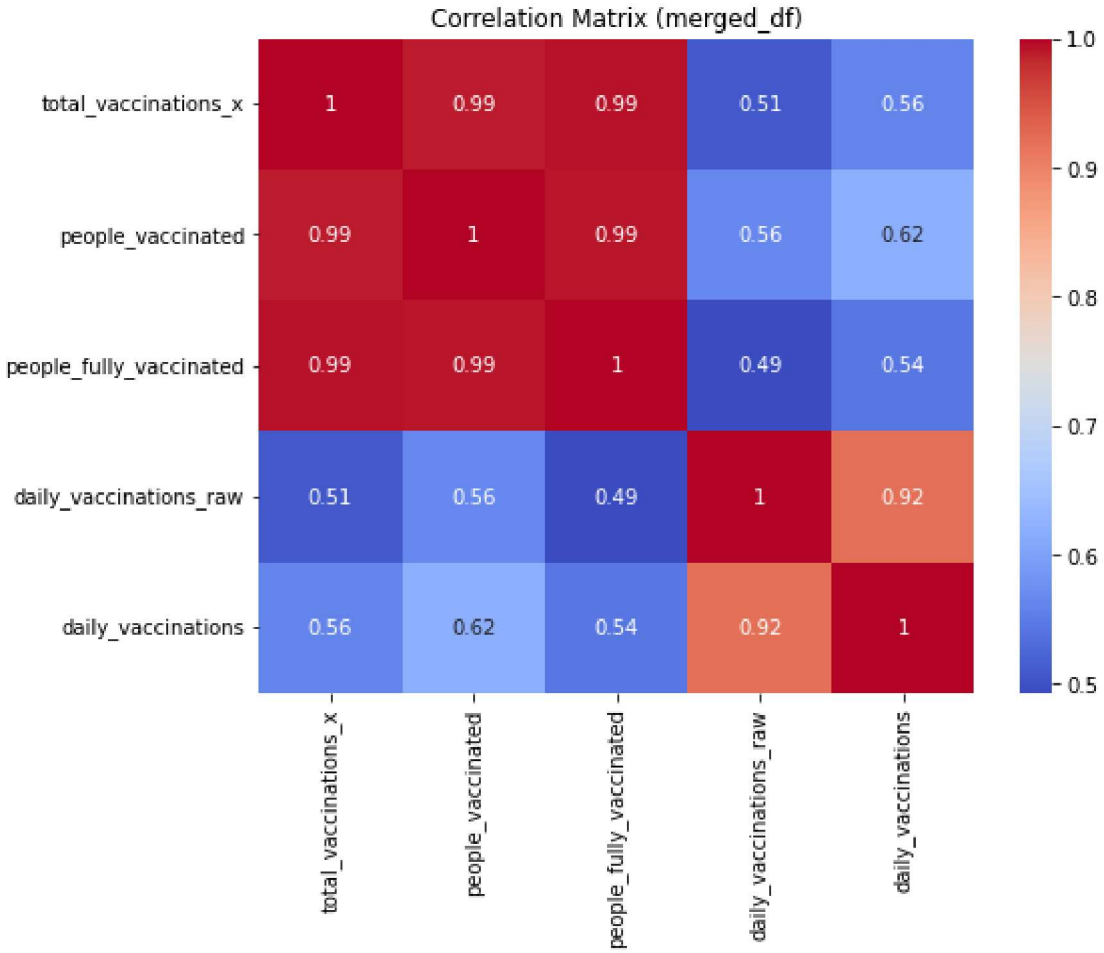
In [16]:
```
# Create a scatter plot for 'total_vaccinations' vs 'people_vaccinated' in 'va
plt.figure(figsize=(8, 6))
plt.scatter(vaccinations_data['total_vaccinations'], vaccinations_data['people
plt.title('Total Vaccinations vs People Vaccinated (vaccinations_data)')
plt.xlabel('Total Vaccinations')
plt.ylabel('People Vaccinated')
plt.grid(True)
plt.show()


plt.figure(figsize=(8, 6))
plt.scatter(merged_df['total_vaccinations_x'], merged_df['people_vaccinated'],
plt.title('Total Vaccinations vs People Vaccinated (merged_df)')
plt.xlabel('Total Vaccinations')
plt.ylabel('People Vaccinated')
plt.grid(True)
plt.show()
```



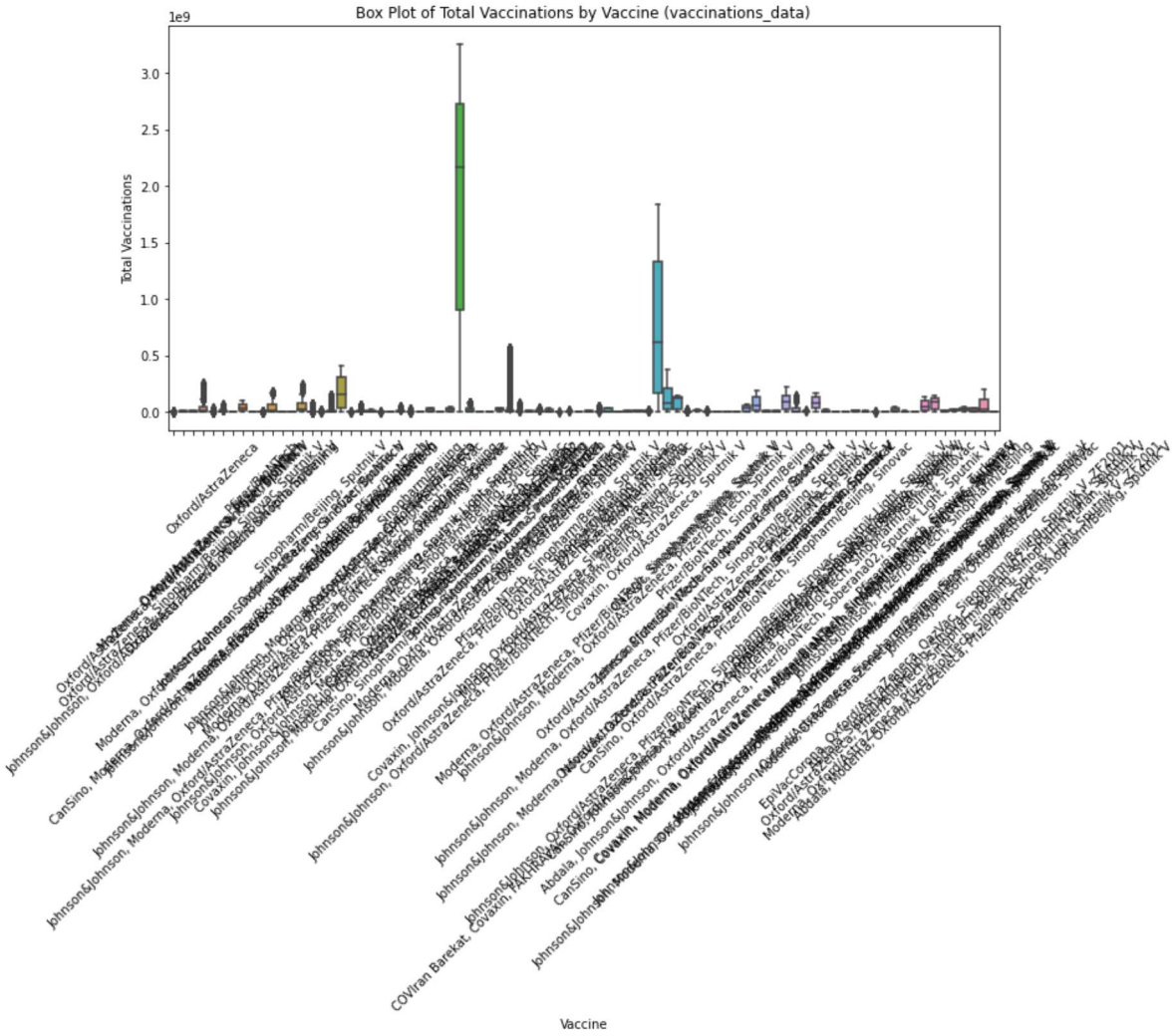

Total Vaccinations vs People Vaccinated (merged_df)
1e8
2.5
2.0
1.5
1.0
0.5
0.0
People Vaccinated
0
1
2
3
4
5
Total Vaccinations
1e8

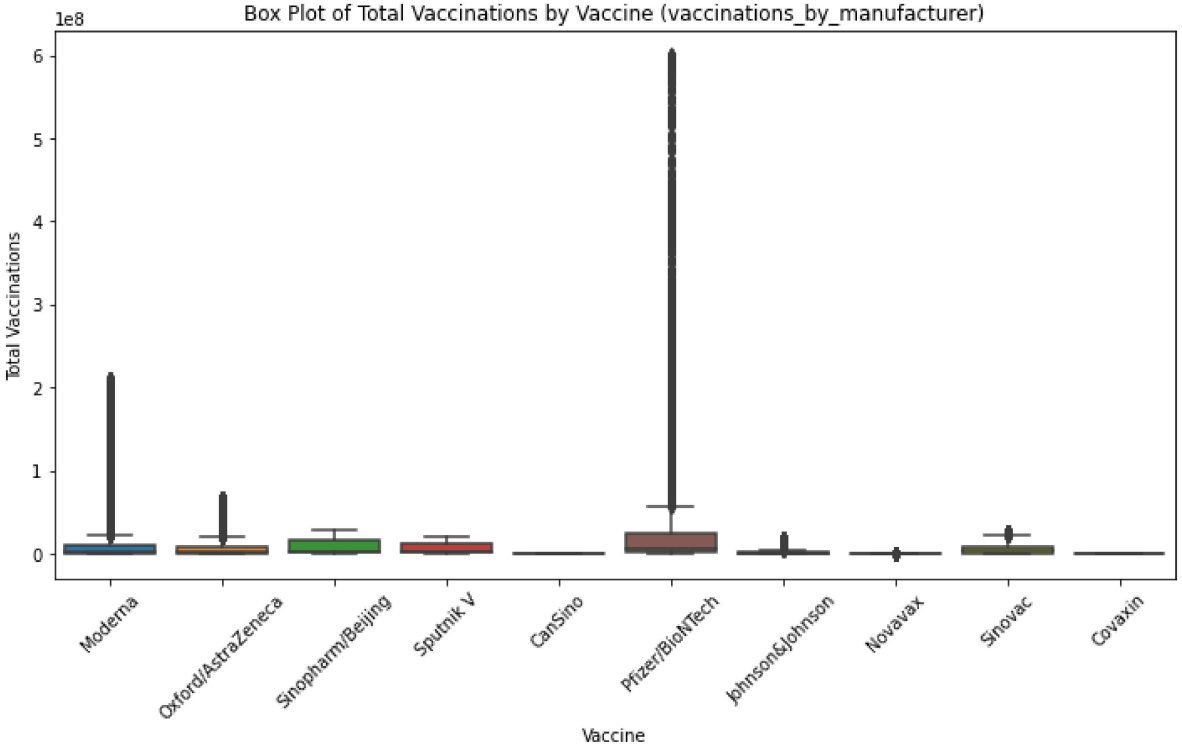
In [17]:
```
relevant_cols_manufacturer = ['total_vaccinations']
relevant_cols_data = ['total_vaccinations', 'people_vaccinated', 'people_fully

# Create a correlation matrix for 'vaccinations_by_manufacturer'
corr_matrix_manufacturer = vaccinations_by_manufacturer[relevant_cols_manufact

# Create a correlation matrix for 'vaccinations_data'
corr_matrix_data = vaccinations_data[relevant_cols_data].corr()

# Create a figure with subplots
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# Plot the correlation matrix for 'vaccinations_by_manufacturer'
sns.heatmap(corr_matrix_manufacturer, annot=True, cmap='coolwarm', ax=axes[0])
axes[0].set_title('Correlation Matrix (vaccinations_by_manufacturer)')
plt.tight_layout()

# Plot the correlation matrix for 'vaccinations_data'
sns.heatmap(corr_matrix_data, annot=True, cmap='coolwarm', ax=axes[1])
axes[1].set_title('Correlation Matrix (vaccinations_data)')
plt.tight_layout()

plt.show()


relevant_cols_merged = ['total_vaccinations_x', 'people_vaccinated', 'people_f

# Create a correlation matrix for 'merged_df'
corr_matrix_merged = merged_df[relevant_cols_merged].corr()

# Create a figure with a subplot
fig, ax = plt.subplots(figsize=(8, 6))

# Plot the correlation matrix for 'merged_df'
sns.heatmap(corr_matrix_merged, annot=True, cmap='coolwarm', ax=ax)
ax.set_title('Correlation Matrix (merged_df)')

plt.show()
```

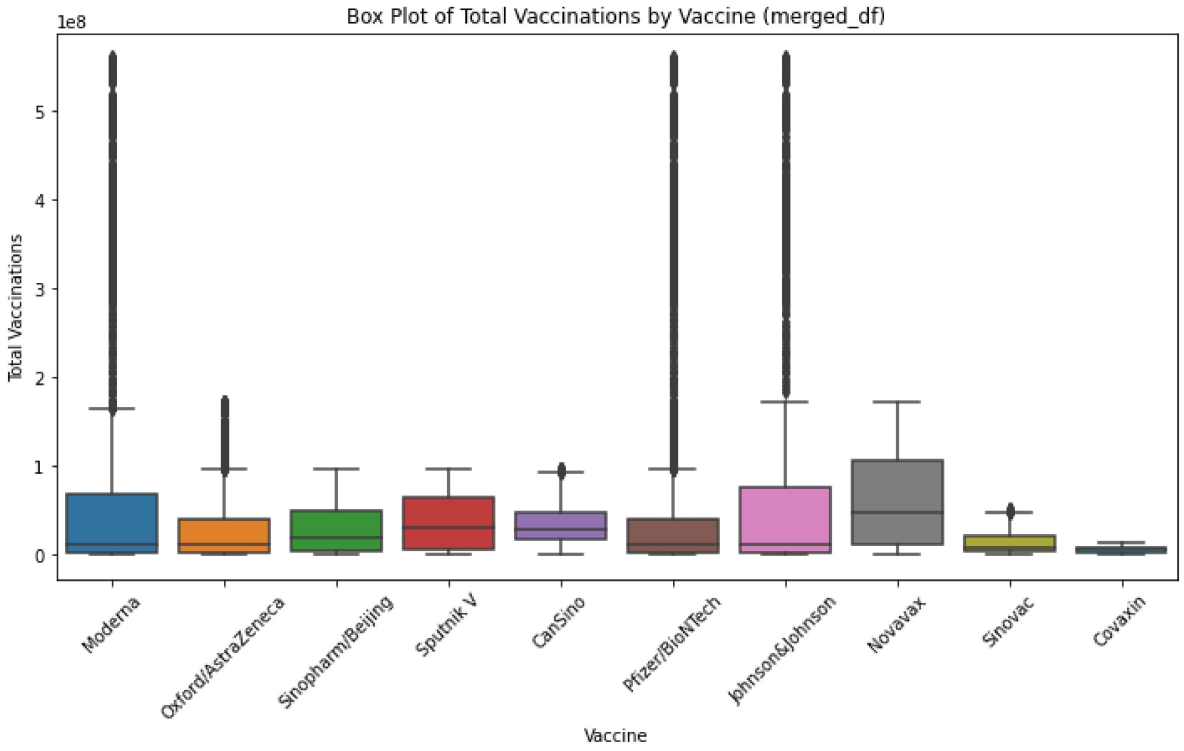



Correlation Matrix (merged_df)
total_vaccinations_x
people_vaccinated
people_fully_vaccinated
daily_vaccinations_raw
daily_vaccinations
1
0.99
0.99
0.51
0.56
0.99
1
0.99
0.56
0.62
0.99
0.99
1
0.49
0.54
0.51
0.56
0.49
1
0.92
0.56
0.62
0.54
0.92
1
1.0
0.9
0.8
0.7
0.6
0.5

In [18]:
```
# Create a box plot to identify outliers for 'total_vaccinations' by vaccine c
plt.figure(figsize=(12, 6))
sns.boxplot(data=vaccinations_data, x='vaccines', y='total_vaccinations')
plt.title('Box Plot of Total Vaccinations by Vaccine (vaccinations_data)')
plt.xlabel('Vaccine')
plt.ylabel('Total Vaccinations')
plt.xticks(rotation=45)
plt.show()


# Create a box plot to identify outliers for 'total_vaccinations' by vaccine c
plt.figure(figsize=(12, 6))
sns.boxplot(data=vaccinations_by_manufacturer, x='vaccine', y='total_vaccinati
plt.title('Box Plot of Total Vaccinations by Vaccine (vaccinations_by_manufact
plt.xlabel('Vaccine')
plt.ylabel('Total Vaccinations')
plt.xticks(rotation=45)
plt.show()



plt.figure(figsize=(12, 6))
sns.boxplot(data=merged_df, x='vaccine', y='total_vaccinations_x')
plt.title('Box Plot of Total Vaccinations by Vaccine (merged_df)')
plt.xlabel('Vaccine')
plt.ylabel('Total Vaccinations')
plt.xticks(rotation=45)
plt.show()
```

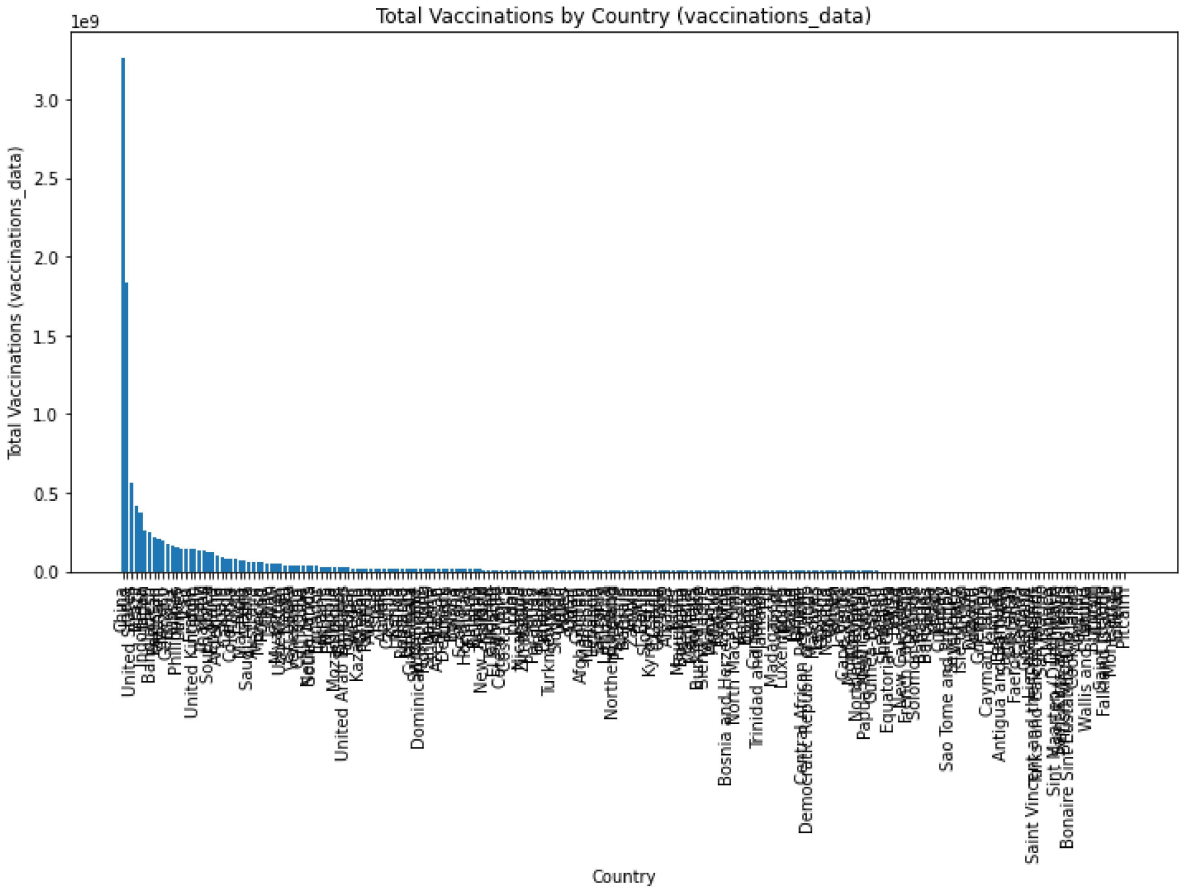

Box Plot of Total Vaccinations by Vaccine (vaccinations_data)
1e9
Total Vaccinations
Vaccine

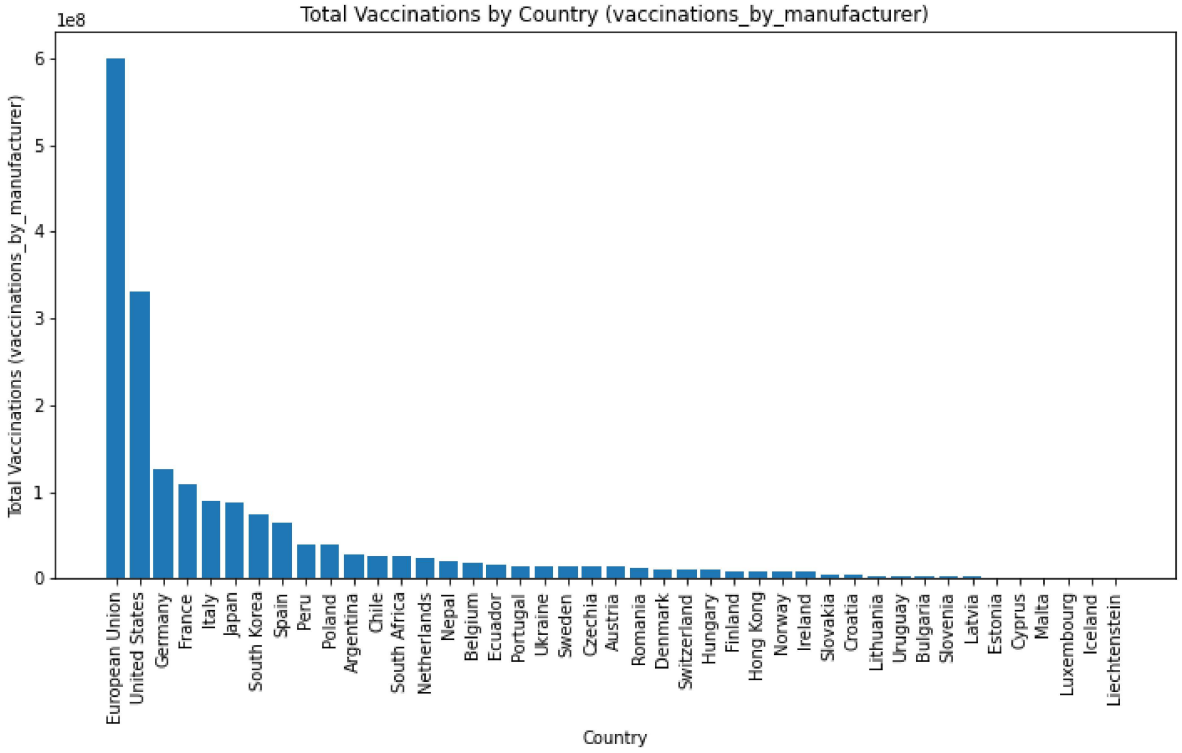

Box Plot of Total Vaccinations by Vaccine (vaccinations_by_manufacturer)
1e8
Total Vaccinations
Moderna
Oxford/AstraZeneca
Sinopharm/Beijing
Sputnik V
CanSino
Pfizer/BioNTech
Johnson&Johnson
Novavax
Sinovac
Covaxin
Vaccine

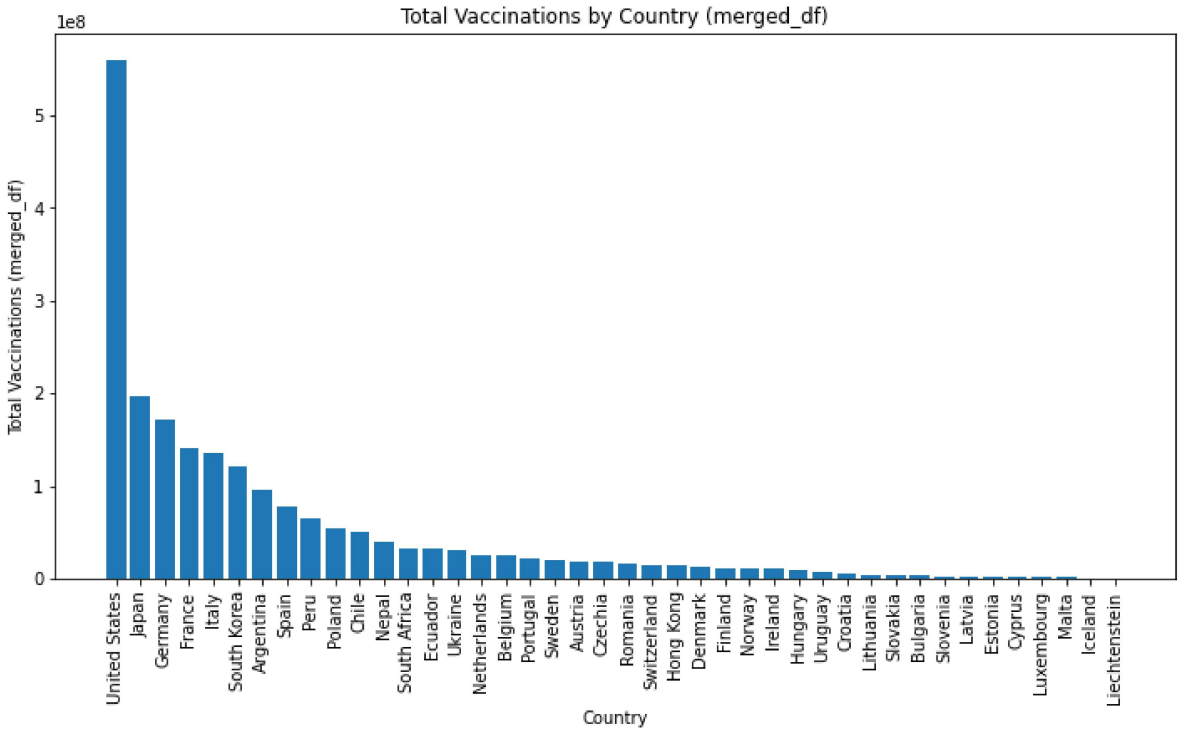

Box Plot of Total Vaccinations by Vaccine (merged_df)
1e8
Total Vaccinations
0
1
2
3
4
5
Moderna
Oxford/AstraZeneca
Sinopharm/Beijing
Sputnik V
CanSino
Pfizer/BioNTech
Johnson&Johnson
Novavax
Sinovac
Covaxin
Vaccine

In [19]:

```
# Group the data by 'country' and calculate the total vaccinations for each co
country_totals_data = vaccinations_data.groupby('country')['total_vaccinations

# Sort the data by total vaccinations in descending order
country_totals_data = country_totals_data.sort_values(by='total_vaccinations',

# Create a bar chart for 'total_vaccinations' in 'vaccinations_data'
plt.figure(figsize=(12, 6))
plt.bar(country_totals_data['country'], country_totals_data['total_vaccination

plt.xlabel('Country')
plt.ylabel('Total Vaccinations (vaccinations_data)')
plt.title('Total Vaccinations by Country (vaccinations_data)')

plt.xticks(rotation=90)
plt.show()


country_totals_manufacturer = vaccinations_by_manufacturer.groupby('country')[

country_totals_manufacturer = country_totals_manufacturer.sort_values(by='tota

# Create a bar chart for 'total_vaccinations' in 'vaccinations_by_manufacturer
plt.figure(figsize=(12, 6))
plt.bar(country_totals_manufacturer['country'], country_totals_manufacturer['t

plt.xlabel('Country')
plt.ylabel('Total Vaccinations (vaccinations_by_manufacturer)')
plt.title('Total Vaccinations by Country (vaccinations_by_manufacturer)')

plt.xticks(rotation=90)
plt.show()

country_totals_merged = merged_df.groupby('country')['total_vaccinations_x'].m

country_totals_merged = country_totals_merged.sort_values(by='total_vaccinatio

# Create a bar chart for 'total_vaccinations_x' in 'merged_df'
plt.figure(figsize=(12, 6))
plt.bar(country_totals_merged['country'], country_totals_merged['total_vaccina

plt.xlabel('Country')
plt.ylabel('Total Vaccinations (merged_df)')
plt.title('Total Vaccinations by Country (merged_df)')

plt.xticks(rotation=90)
plt.show()
```


Total Vaccinations by Country (vaccinations_data)
1e9
Total Vaccinations (vaccinations_data)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Country


Total Vaccinations by Country (vaccinations_by_manufacturer)
1e8
Total Vaccinations (vaccinations_by_manufacturer)
6
5
4
3
2
1
0
European Union
United States
Germany
France
Italy
Japan
South Korea
Spain
Peru
Poland
Argentina
Chile
South Africa
Netherlands
Nepal
Belgium
Ecuador
Portugal
Ukraine
Sweden
Czechia
Austria
Romania
Denmark
Switzerland
Hungary
Finland
Hong Kong
Norway
Ireland
Slovakia
Croatia
Lithuania
Uruguay
Bulgaria
Slovenia
Latvia
Estonia
Cyprus
Malta
Luxembourg
Iceland
Liechtenstein
Country



STATISTICAL ANALYSIS

In [20]:
```
# Example: T-test for vaccine efficacy comparison

import scipy.stats as stats


vaccine_a_data = vaccinations_data[vaccinations_data['vaccines'] == 'Vaccine A
vaccine_b_data = vaccinations_data[vaccinations_data['vaccines'] == 'Vaccine B


# Perform a t-test for comparing the means of "Vaccine A" and "Vaccine B"
t_statistic, p_value = stats.ttest_ind(vaccine_a_data, vaccine_b_data, equal_v

# Print the results
if p_value < 0.05:  # You can adjust the significance level (alpha)
    print("There is a significant difference in vaccine efficacy between Vacci
else:
    print("There is no significant difference in vaccine efficacy between Vacc
```

```
There is no significant difference in vaccine efficacy between Vaccine A and
Vaccine B.
```

In [21]:
```
country_a = 'Country A'
country_b = 'Country B'


total_vaccinations_country_a = vaccinations_data[vaccinations_data['country']
total_vaccinations_country_b = vaccinations_data[vaccinations_data['country']

# Perform a two-sample t-test to compare the means of the two countries
t_statistic, p_value = stats.ttest_ind(total_vaccinations_country_a, total_vac

# Print the t-statistics and p-value
print(f'T-Statistic: {t_statistic}')
print(f'P-Value: {p_value}')

# Print the results
if p_value < 0.05:  # You can adjust the significance level (alpha)
    print(f"There is a significant difference in total vaccinations between {c
else:
    print(f"There is no significant difference in total vaccinations between {
```

```
T-Statistic: nan
P-Value: nan
There is no significant difference in total vaccinations between Country A an
d Country B.
```

VACCINES EFFICACY

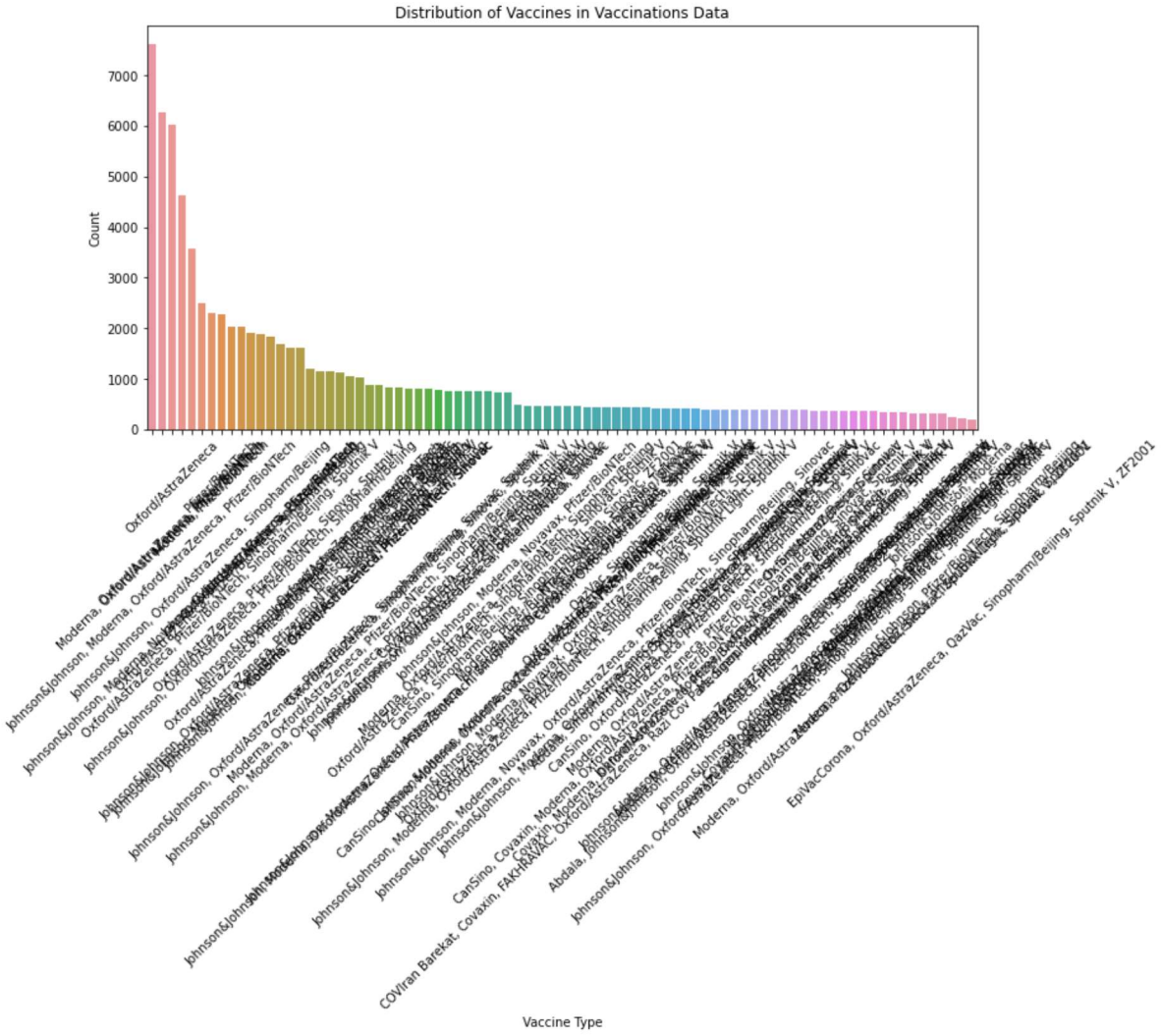
In [22]:
```
# Calculate attack rates
merged_df['attack_rate_vaccinated'] = merged_df['people_fully_vaccinated'] / m
merged_df['attack_rate_unvaccinated'] = (merged_df['total_vaccinations_x'] - m

# Calculate vaccine efficacy
merged_df['vaccine_efficacy'] = (1 - (merged_df['attack_rate_unvaccinated'] /

# Display the dataset with vaccine efficacy
print(merged_df[['country', 'vaccine', 'date', 'vaccine_efficacy']])
```

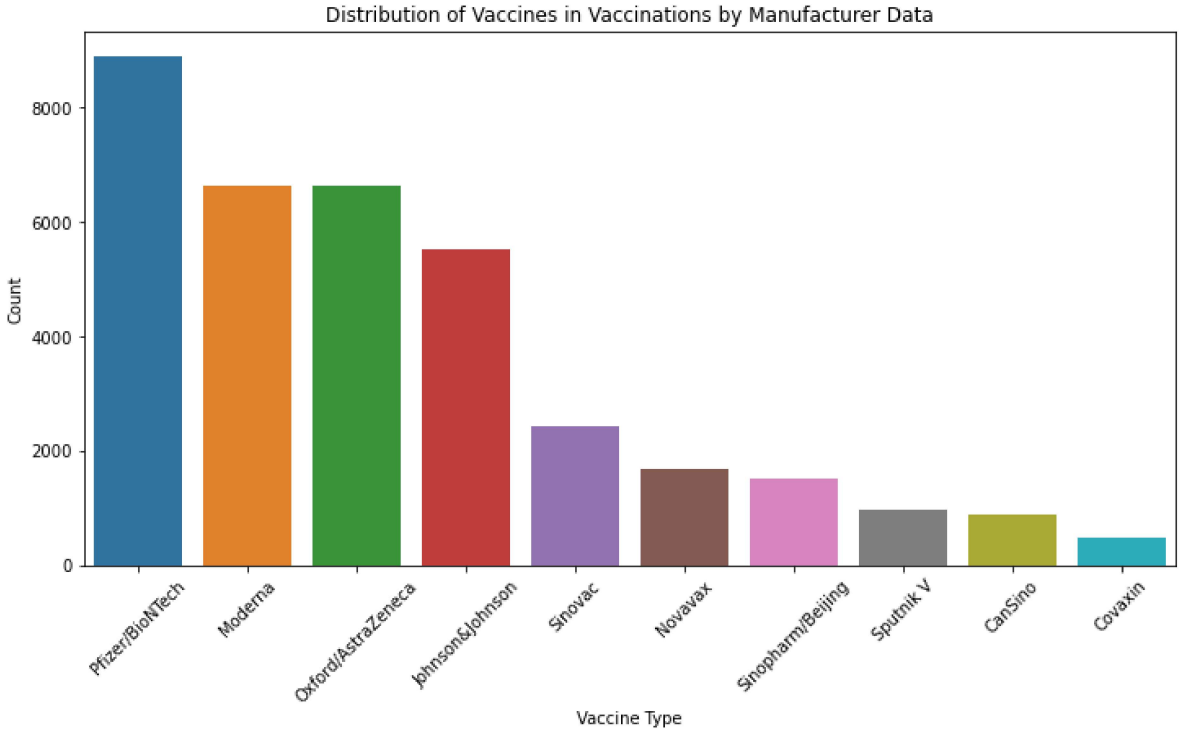
```
        country             vaccine       date  vaccine_efficacy
0     Argentina             Moderna 2020-12-29    -409540.000000
1     Argentina  Oxford/AstraZeneca 2020-12-29    -409540.000000
2     Argentina   Sinopharm/Beijing 2020-12-29    -409540.000000
3     Argentina           Sputnik V 2020-12-29    -409540.000000
4     Argentina             Moderna 2020-12-30    -405690.000000
...         ...                 ...        ...              ...
28949   Uruguay     Pfizer/BioNTech 2022-03-28        -82.106766
28950   Uruguay             Sinovac 2022-03-28        -82.106766
28951   Uruguay  Oxford/AstraZeneca 2022-03-29        -82.420469
28952   Uruguay     Pfizer/BioNTech 2022-03-29        -82.420469
28953   Uruguay             Sinovac 2022-03-29        -82.420469

[27018 rows x 4 columns]
```

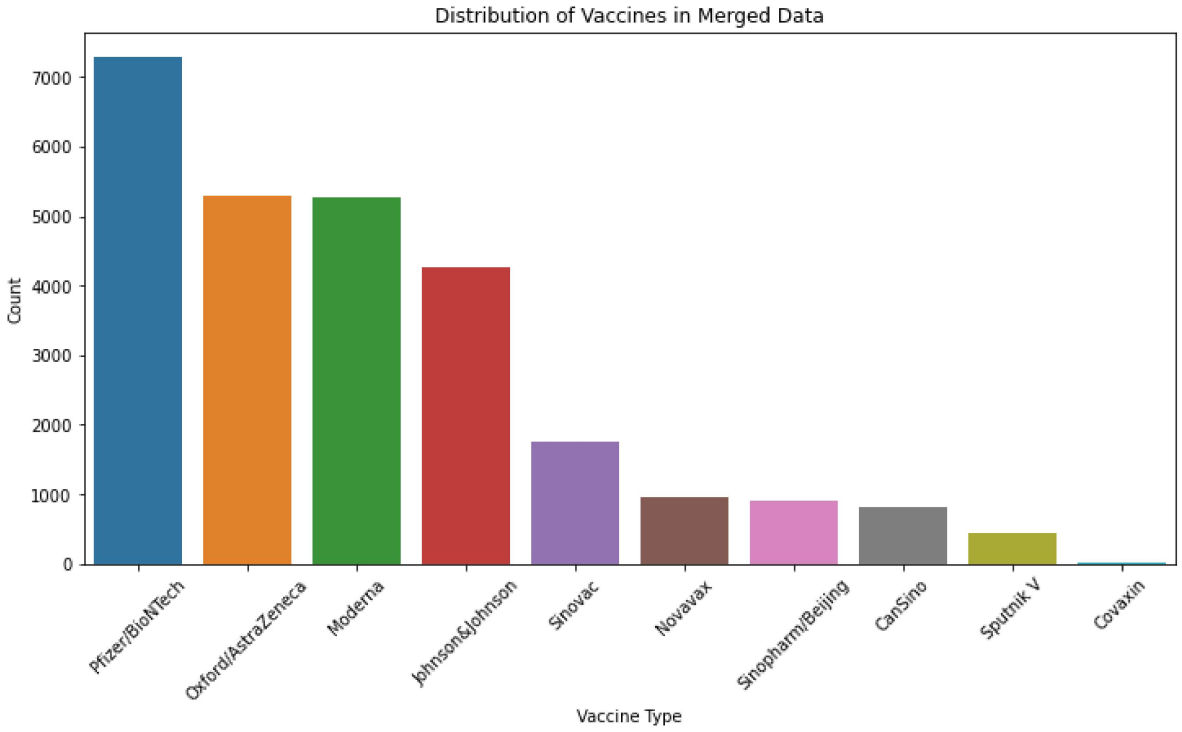
DISTRIBUTIONS

In [23]:

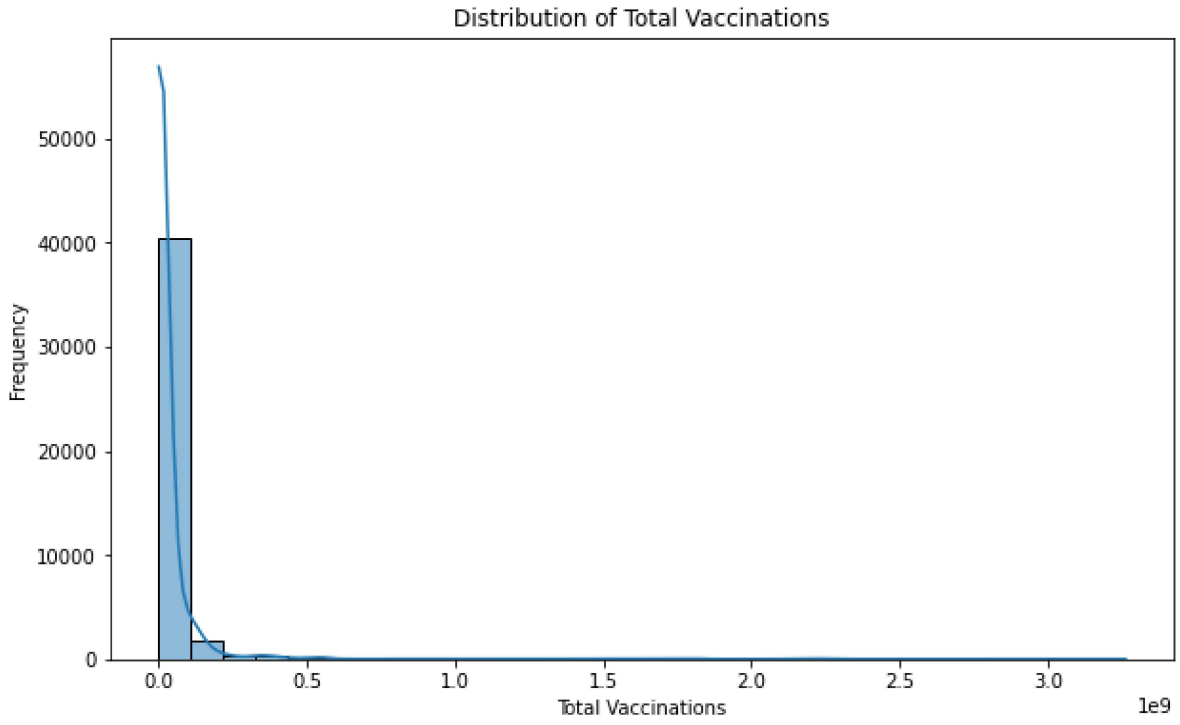
```
# Create a count plot for the distribution of vaccines in 'vaccinations_data'
plt.figure(figsize=(12, 6))
sns.countplot(data=vaccinations_data, x='vaccines', order=vaccinations_data['v
plt.title('Distribution of Vaccines in Vaccinations Data')
plt.xlabel('Vaccine Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()


# Create a count plot for the distribution of vaccines in 'vaccinations_by_man
plt.figure(figsize=(12, 6))
sns.countplot(data=vaccinations_by_manufacturer, x='vaccine', order=vaccinatio
plt.title('Distribution of Vaccines in Vaccinations by Manufacturer Data')
plt.xlabel('Vaccine Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()

# Create a count plot for the distribution of vaccines in 'merged_df'
plt.figure(figsize=(12, 6))
sns.countplot(data=merged_df, x='vaccine', order=merged_df['vaccine'].value_co
plt.title('Distribution of Vaccines in Merged Data')
plt.xlabel('Vaccine Type')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

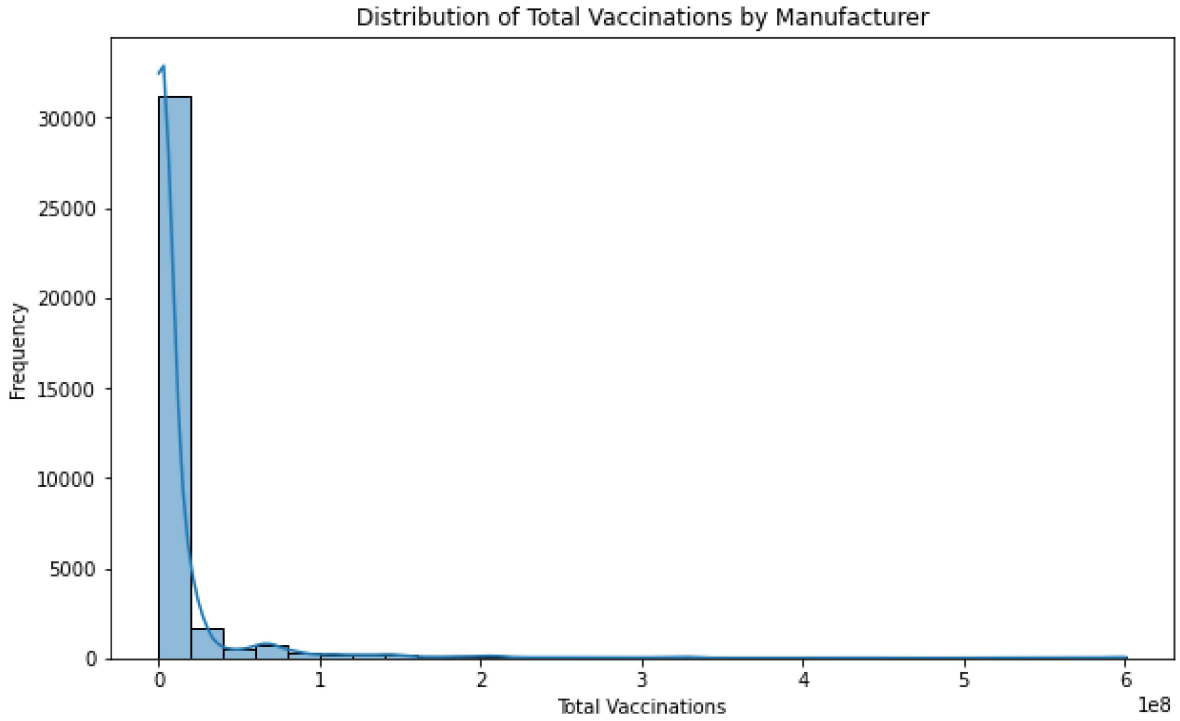

Distribution of Vaccines in Vaccinations Data
Count
Vaccine Type

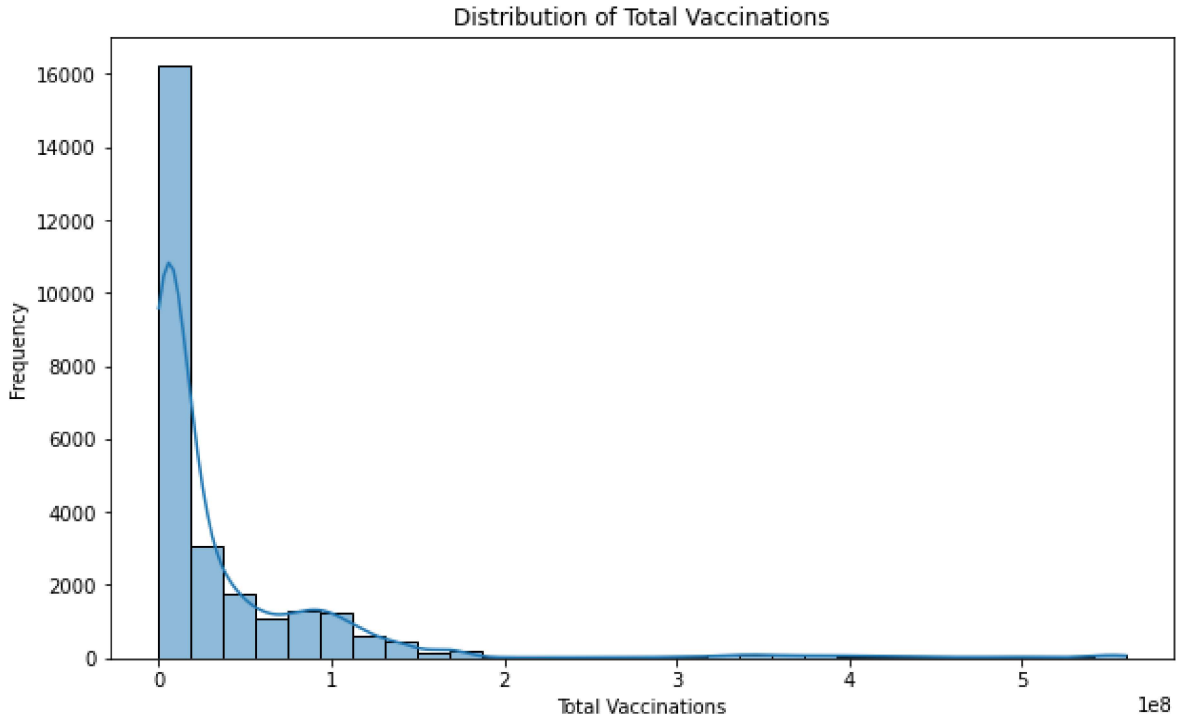

Distribution of Vaccines in Vaccinations by Manufacturer Data
Count
Pfizer/BioNTech
Moderna
Oxford/AstraZeneca
Johnson&Johnson
Sinovac
Novavax
Sinopharm/Beijing
Sputnik V
CanSino
Covaxin
Vaccine Type


Distribution of Vaccines in Merged Data
Count
0
1000
2000
3000
4000
5000
6000
7000
Pfizer/BioNTech
Oxford/AstraZeneca
Moderna
Johnson&Johnson
Sinovac
Novavax
Sinopharm/Beijing
CanSino
Sputnik V
Covaxin
Vaccine Type

In [24]:
```
plt.figure(figsize=(10, 6))
sns.histplot(data=vaccinations_data, x='total_vaccinations', bins=30, kde=True
plt.title('Distribution of Total Vaccinations')
plt.xlabel('Total Vaccinations')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(data=vaccinations_by_manufacturer, x='total_vaccinations', bins=3
plt.title('Distribution of Total Vaccinations by Manufacturer')
plt.xlabel('Total Vaccinations')
plt.ylabel('Frequency')
plt.show()


# Create a histogram for 'total_vaccinations' in merged_df
plt.figure(figsize=(10, 6))
sns.histplot(data=merged_df, x='total_vaccinations_x', bins=30, kde=True)
plt.title('Distribution of Total Vaccinations')
plt.xlabel('Total Vaccinations')
plt.ylabel('Frequency')
plt.show()
```



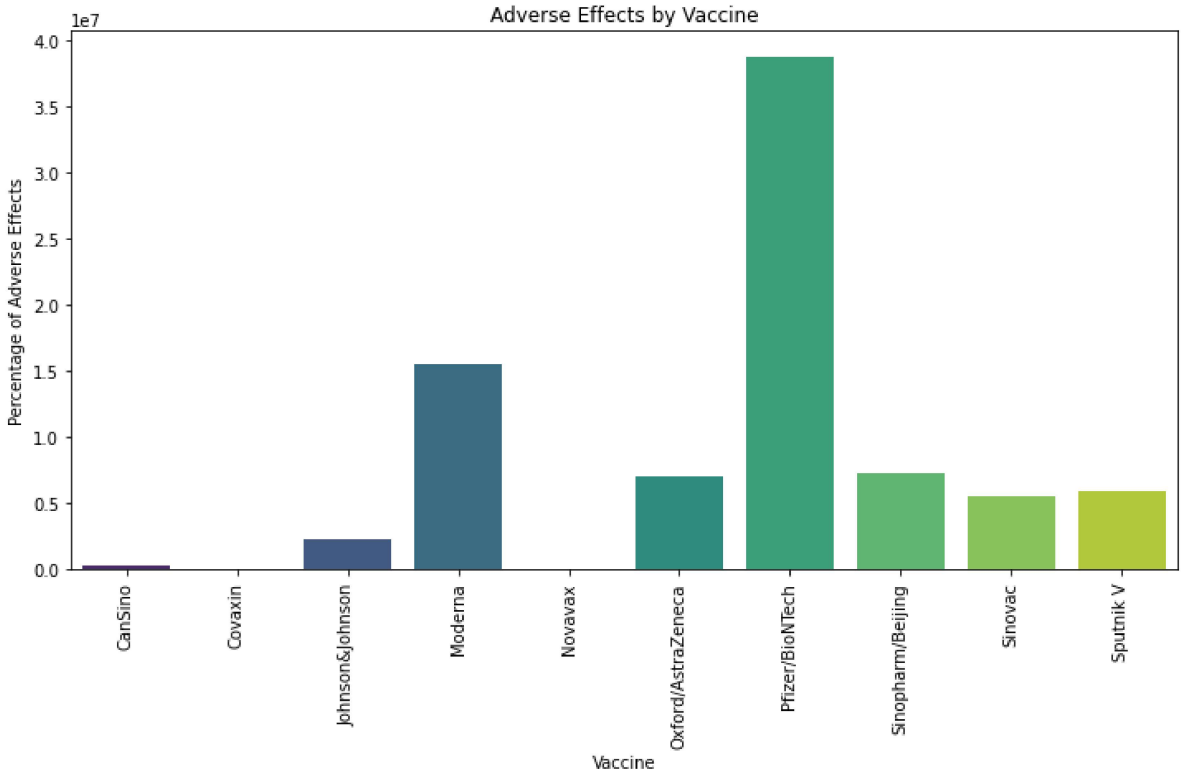




ADVERSE EFFECTS

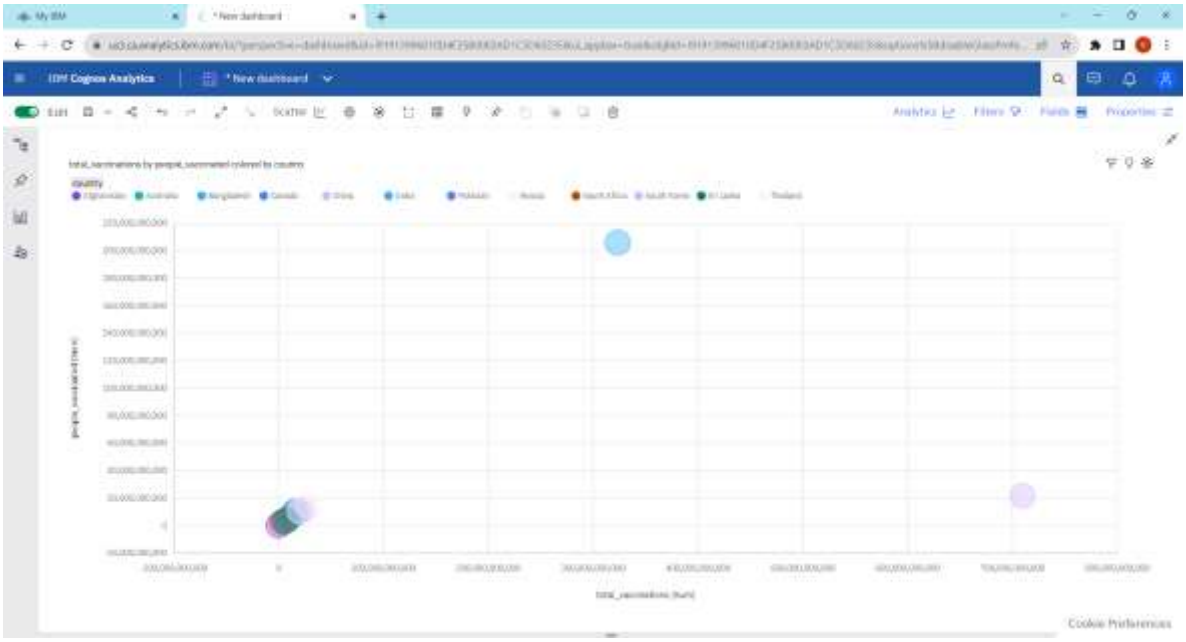
In [25]:
```
vaccinations_data.rename(columns={'vaccines': 'vaccine'}, inplace=True)
```

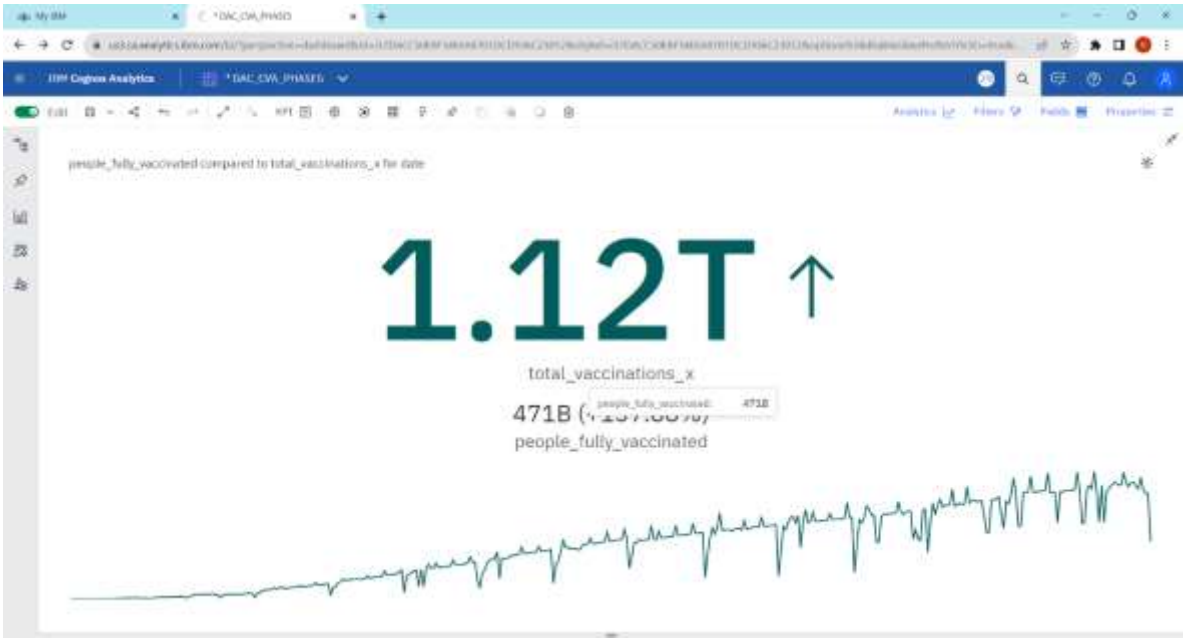
In [26]:
```
merged_data_adverse=vaccinations_by_manufacturer.merge(vaccinations_data, on=[


adverse_effects_by_vaccine = merged_data_adverse.groupby('vaccine')['total_vac

plt.figure(figsize=(12, 6))
sns.barplot(x=adverse_effects_by_vaccine.index, y=adverse_effects_by_vaccine.v
plt.xlabel('Vaccine')
plt.ylabel('Percentage of Adverse Effects')
plt.title('Adverse Effects by Vaccine')
plt.xticks(rotation=90)
plt.show()
```



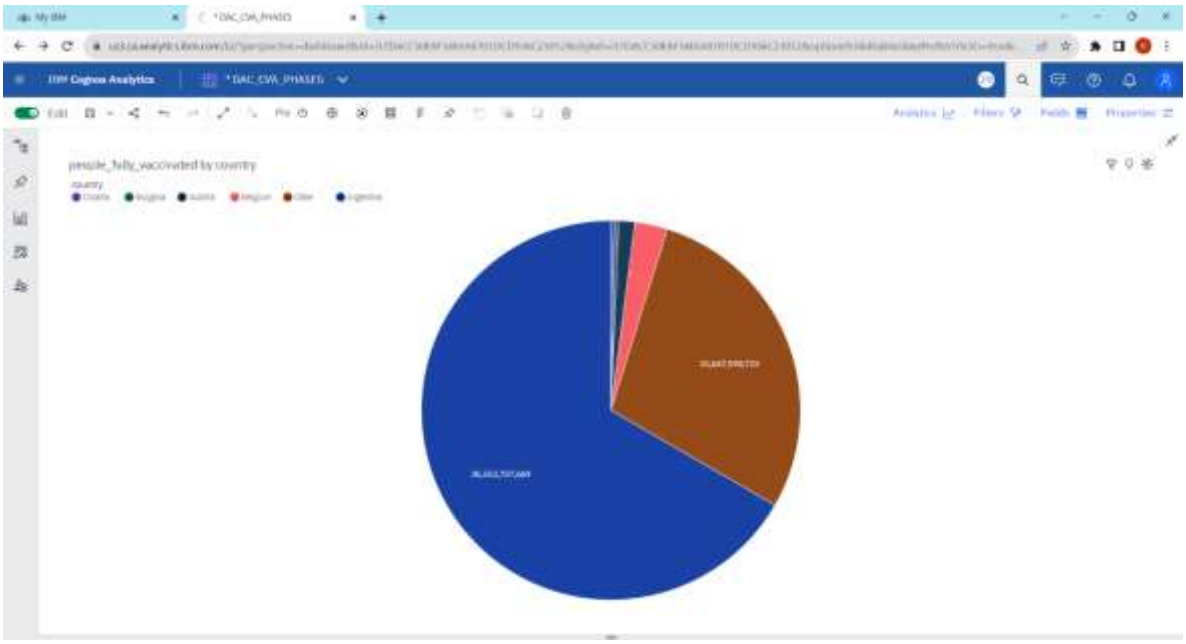
COLLEGE CODE-4212
421221243022-PHASE 5

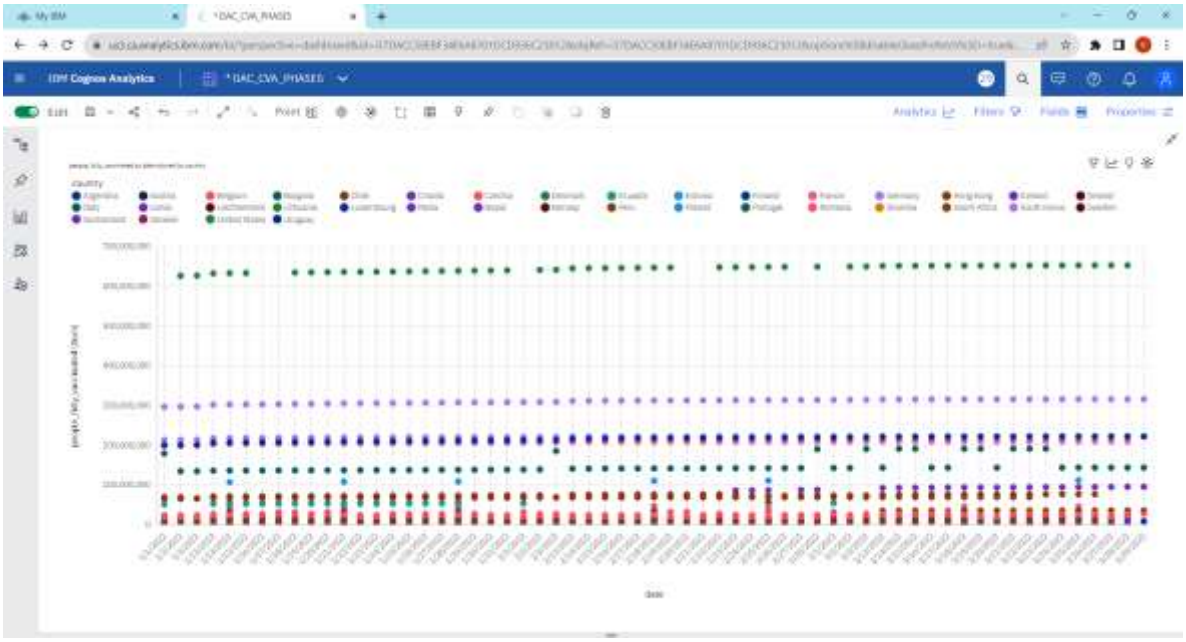
VISUALIZATION USING IBM COGNOS ANALYTICS



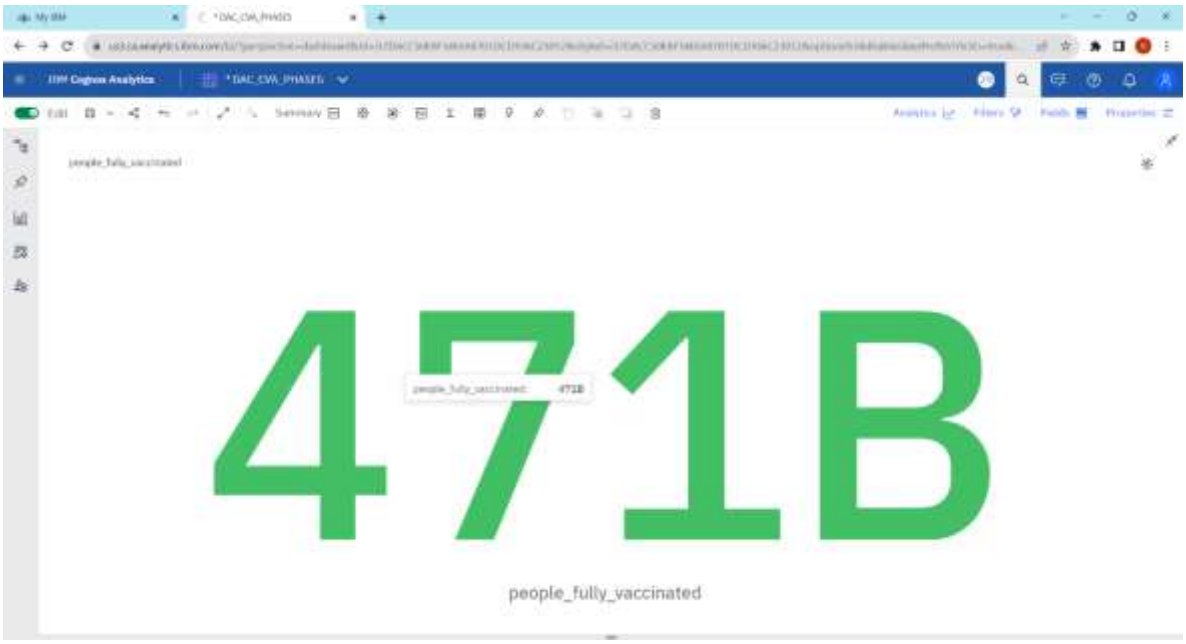


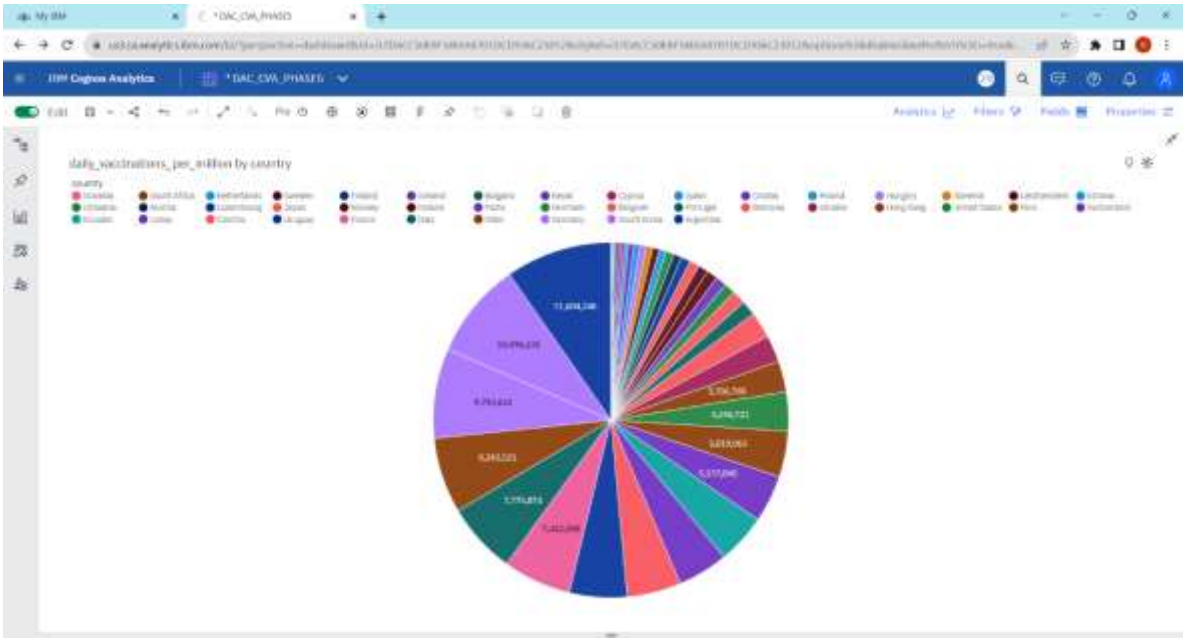
COLLEGE CODE-4212
421221243022-PHASE 5



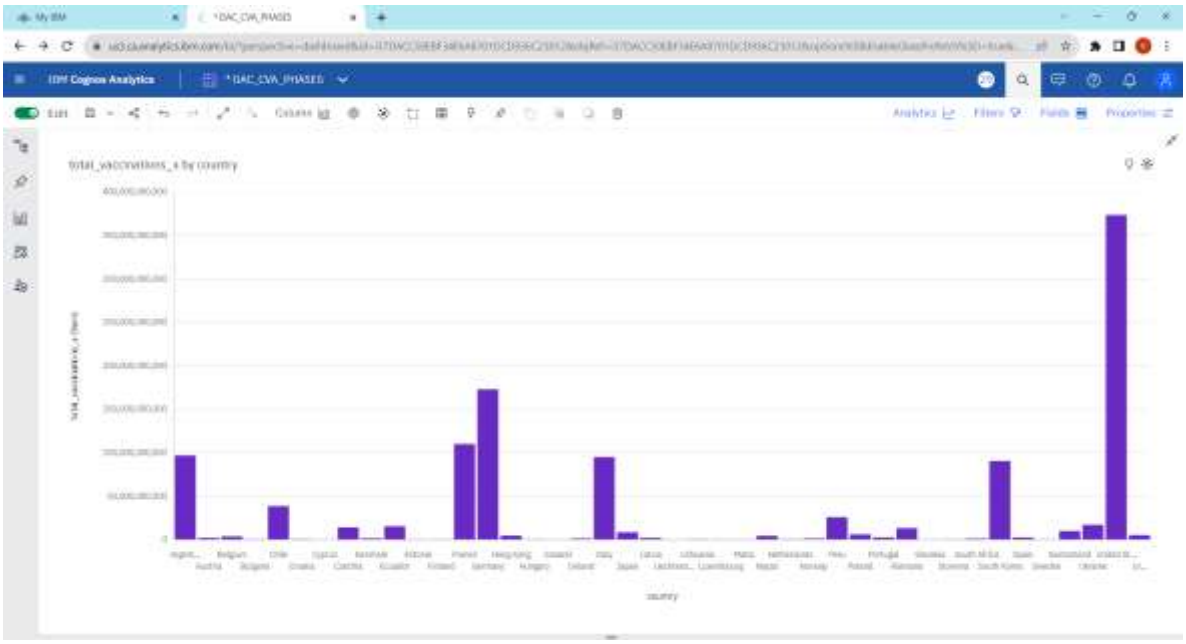


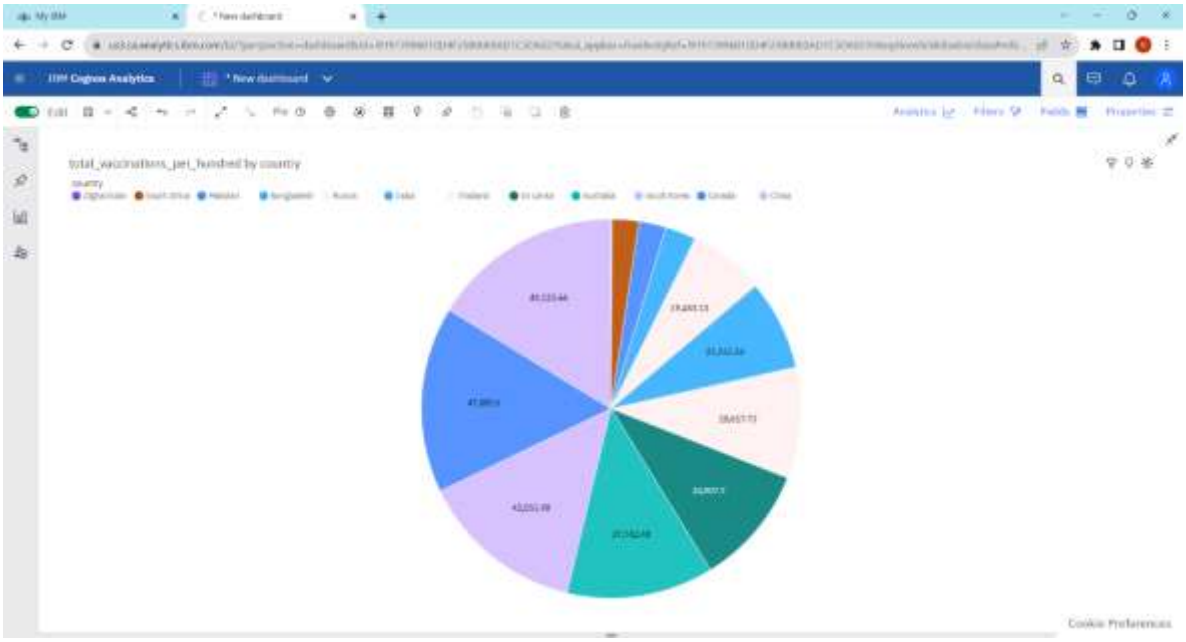
COLLEGE CODE-4212
421221243022-PHASE 5





COLLEGE CODE-4212
421221243022-PHASE 5





COLLEGE CODE-4212
421221243022-PHASE 5