

COVID VACCINES ANALYSIS PHASE 4-DATA ANALYTICS WITH COGNOS GROUP 2 DEVELOPMENT PART 2

In the previous phases we have discussed the step-by-step process, Design thinking and at the phase3 we have discussed the data preprocessing techniques and many more in the last steps and in this step, we have given some problem statements to solve.

In this part we will continue building your project. Continue conducting the Covid-19 vaccines analysis by performing:

- Exploratory data analysis
- Statistical analysis
- Visualization

IBM NAAN MUDHALVAN

Development Part 2 - COVID VACCINES ANALYSIS

Exploratory data analysis

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

```
In [2]: # Load your Covid-19 vaccine data into a DataFrame
data = pd.read_csv("D:/ibm naan mudhalvan/country_vaccinations.csv") # Replace with your dataset's filename
```

```
In [3]: # Display basic information about the dataset
print(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  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
None
```

```
In [4]: # Display summary statistics
print(data.describe())
```

```

      total_vaccinations  people_vaccinated  people_fully_vaccinated  \
count      4.360700e+04      4.129400e+04      3.880200e+04
mean      4.592964e+07      1.770508e+07      1.413830e+07
std       2.246004e+08      7.078731e+07      5.713920e+07
min       0.000000e+00      0.000000e+00      1.000000e+00
25%       5.264100e+05      3.494642e+05      2.439622e+05
50%       3.590096e+06      2.187310e+06      1.722140e+06
75%       1.701230e+07      9.152520e+06      7.559870e+06
max       3.263129e+09      1.275541e+09      1.240777e+09

      daily_vaccinations_raw  daily_vaccinations  \
count      3.536200e+04      8.621300e+04
mean      2.705996e+05      1.313055e+05
std       1.212427e+06      7.682388e+05
min       0.000000e+00      0.000000e+00
25%       4.668000e+03      9.000000e+02
50%       2.530900e+04      7.343000e+03
75%       1.234925e+05      4.409800e+04
max       2.474100e+07      2.242429e+07

      total_vaccinations_per_hundred  people_vaccinated_per_hundred  \
count      43607.000000      41294.000000
mean       80.188543      40.927317
std       67.913577      29.290759
min        0.000000      0.000000
25%       16.050000      11.370000
50%       67.520000      41.435000
75%      132.735000      67.910000
max      345.370000      124.760000

      people_fully_vaccinated_per_hundred  daily_vaccinations_per_million
count      38802.000000      86213.000000
mean       35.523243      3257.049157
std       28.376252      3934.312440
min        0.000000      0.000000
25%       7.020000      636.000000
50%       31.750000      2050.000000
75%       62.080000      4682.000000
max      122.370000      117497.000000

```

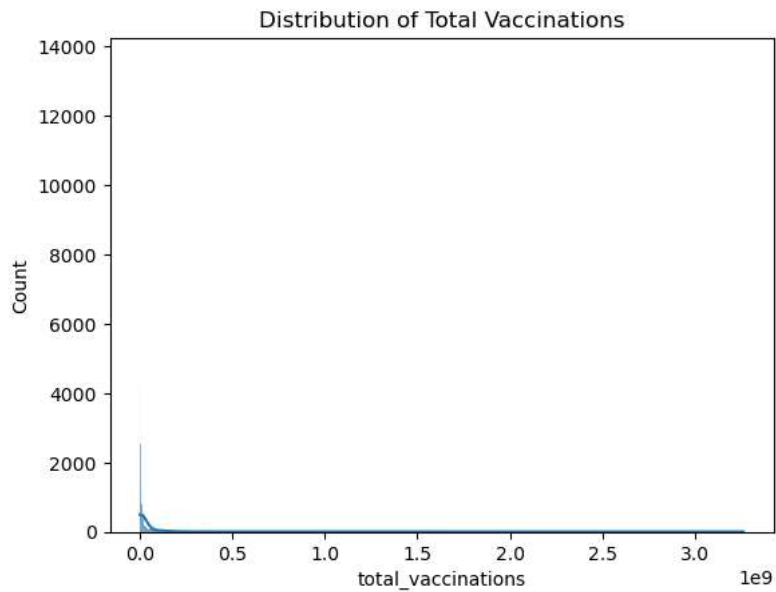
```
In [5]: # Check for missing values
print(data.isnull().sum())
```

```

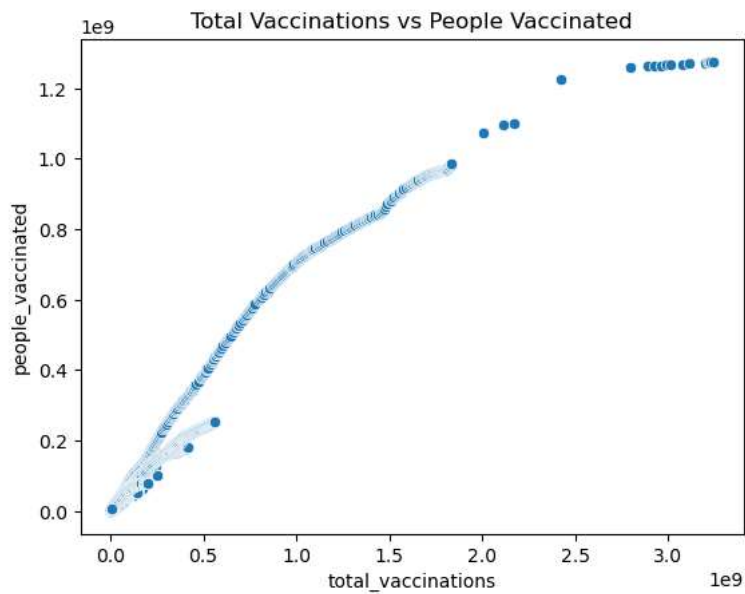
country      0
iso_code     0
date         0
total_vaccinations      42905
people_vaccinated      45218
people_fully_vaccinated  47710
daily_vaccinations_raw  51150
daily_vaccinations      299
total_vaccinations_per_hundred      42905
people_vaccinated_per_hundred      45218
people_fully_vaccinated_per_hundred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

```

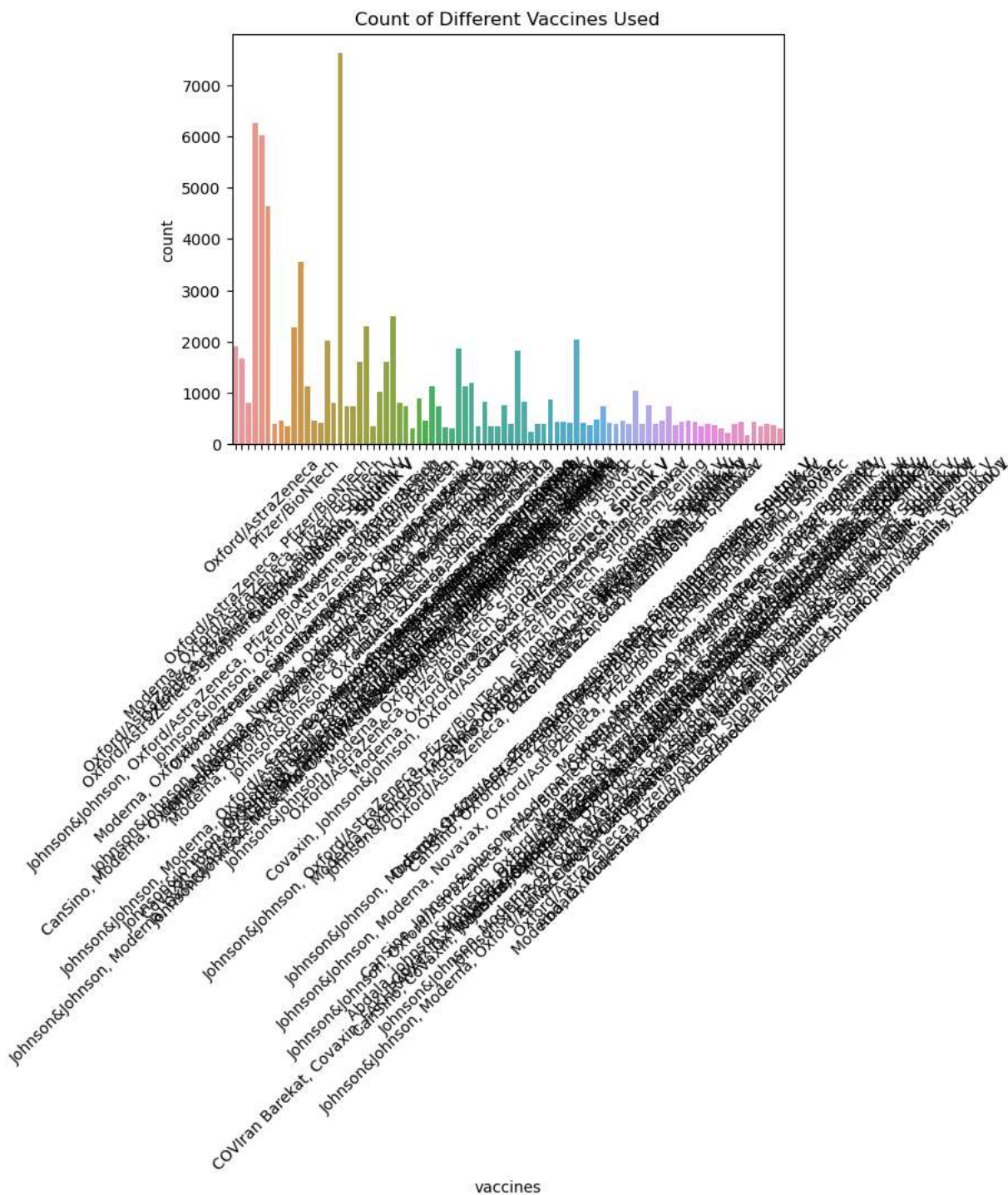
```
In [6]: # Visualize the distribution of total_vaccinations
sns.histplot(data['total_vaccinations'].dropna(), kde=True)
plt.title('Distribution of Total Vaccinations')
plt.show()
```



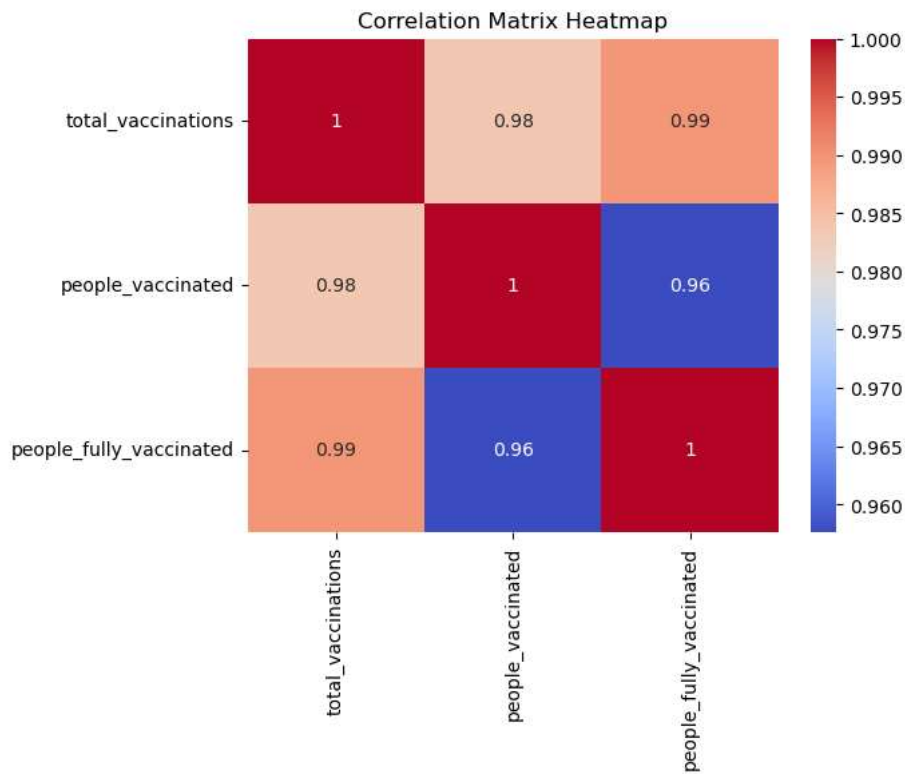
```
In [7]: # Visualize the relationship between total_vaccinations and people_vaccinated
sns.scatterplot(x='total_vaccinations', y='people_vaccinated', data=data)
plt.title('Total Vaccinations vs People Vaccinated')
plt.show()
```



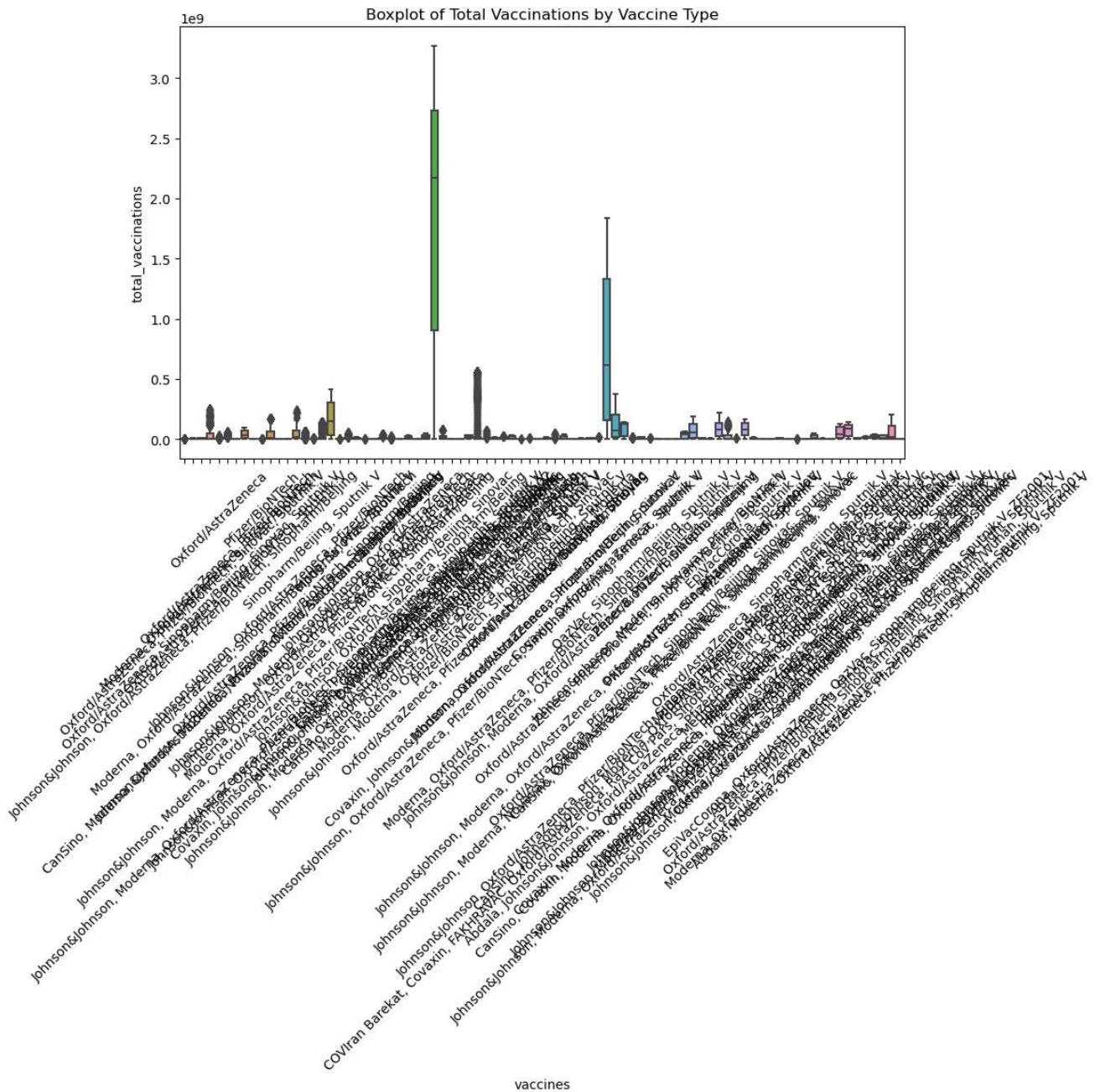
```
In [8]: # Visualize the distribution of vaccines used
sns.countplot(x='vaccines', data=data)
plt.title('Count of Different Vaccines Used')
plt.xticks(rotation=45)
plt.show()
```



```
In [9]: # Correlation matrix heatmap for relevant numerical variables
numerical_columns = ['total_vaccinations', 'people_vaccinated', 'people_fully_vaccinated']
correlation_matrix = data[numerical_columns].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix Heatmap')
plt.show()
```



```
In [10]: # Boxplot to identify outliers in 'total_vaccinations' for each 'vaccines' category
plt.figure(figsize=(10, 6)) # Adjust the figure size if needed
sns.boxplot(x='vaccines', y='total_vaccinations', data=data)
plt.title('Boxplot of Total Vaccinations by Vaccine Type')
plt.xticks(rotation=45)
plt.show()
```



Statistical Analysis

```
In [11]: #Statistical Analysis

# Extract data for two countries of interest
country1_data = data[data['country'] == 'Country1']
country2_data = data[data['country'] == 'Country2']

# Perform a two-sample t-test to compare total vaccinations in the two countries
t_stat, p_value = stats.ttest_ind(country1_data['total_vaccinations'].dropna(), country2_data['total_vaccinations'].dropna())

# Display the results
print(f"t-statistic: {t_stat}")
print(f"P-value: {p_value}")

# Determine the significance of the test
alpha = 0.05 # Set your significance level
if p_value < alpha:
    print("Reject the null hypothesis: There is a significant difference in total vaccinations between the two countries.")
else:
    print("Fail to reject the null hypothesis: There is no significant difference in total vaccinations between the two countries")

t-statistic: nan
P-value: nan
Fail to reject the null hypothesis: There is no significant difference in total vaccinations between the two countries.
```


Visualization

```
In [12]: #Visualization

#Bar chart

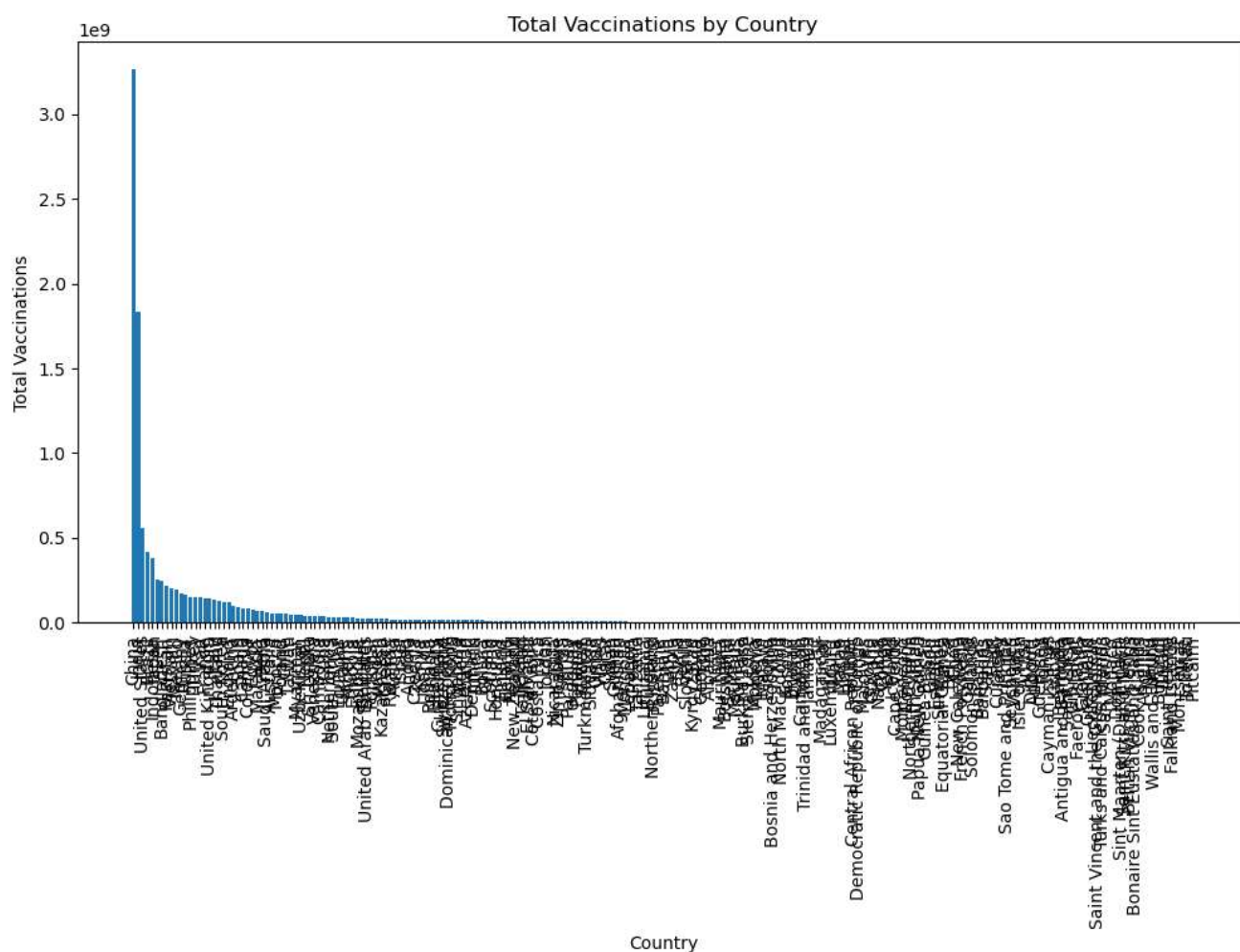
# Group the data by 'country' and calculate the total vaccinations for each country
country_totals = data.groupby('country')['total_vaccinations'].max().reset_index()

# Sort the data by total vaccinations in descending order
country_totals = country_totals.sort_values(by='total_vaccinations', ascending=False)

# Create a bar chart
plt.figure(figsize=(12, 6)) # Adjust the figure size if needed
plt.bar(country_totals['country'], country_totals['total_vaccinations'])

# Add Labels and a title
plt.xlabel('Country')
plt.ylabel('Total Vaccinations')
plt.title('Total Vaccinations by Country')

# Show the chart
plt.xticks(rotation=90)
plt.show()
```



```
In [13]: #Line chart

# Assuming 'date' is in datetime format, if not, convert it to datetime
data['date'] = pd.to_datetime(data['date'])

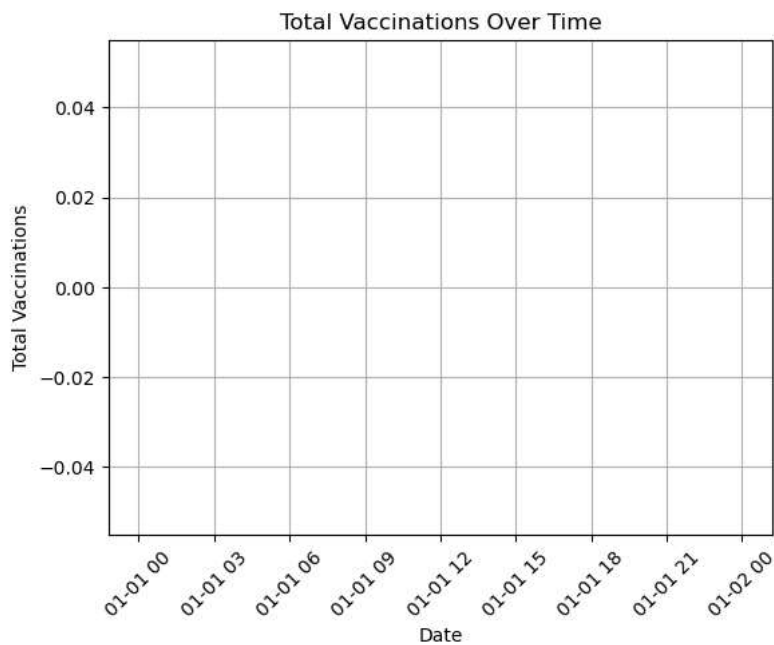
# Sort the data by date
data.sort_values(by='date', inplace=True)

# Extract data for a specific country (you can change the country)
country_data = data[data['country'] == 'Your_Country_Name']

# Create a Line chart
plt.plot(country_data['date'], country_data['total_vaccinations'], marker='o', linestyle='-')

# Add Labels and a title
plt.xlabel('Date')
plt.ylabel('Total Vaccinations')
plt.title('Total Vaccinations Over Time')

# Show the chart
plt.grid(True) # Add grid Lines
plt.xticks(rotation=45)
plt.show()
```



In []:

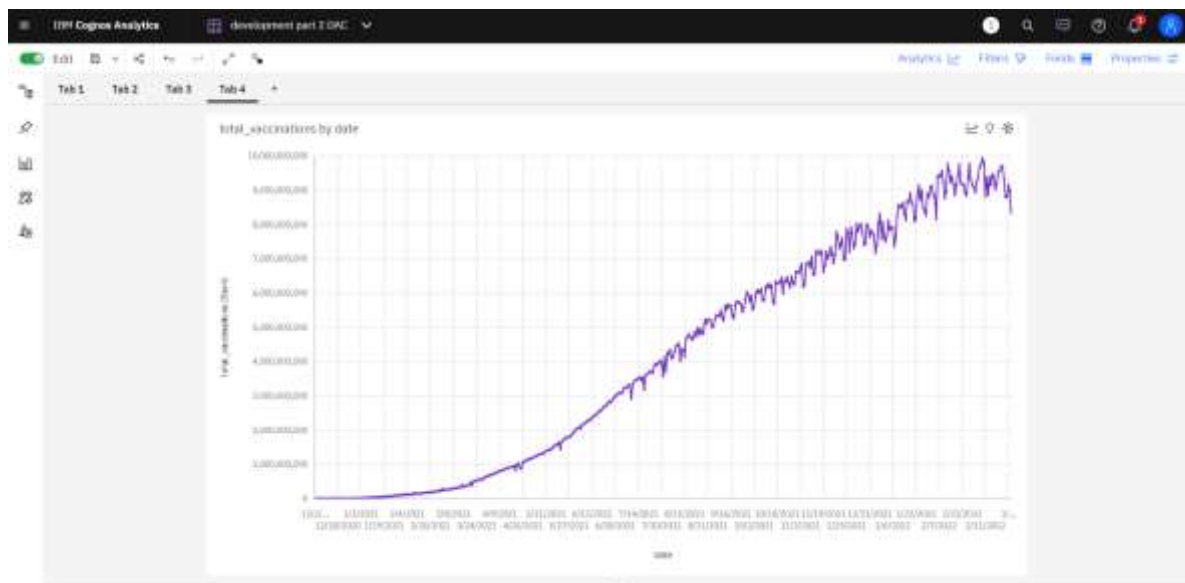
VISUALIZATION USING IBM COGNOS

SCATTER PLOT

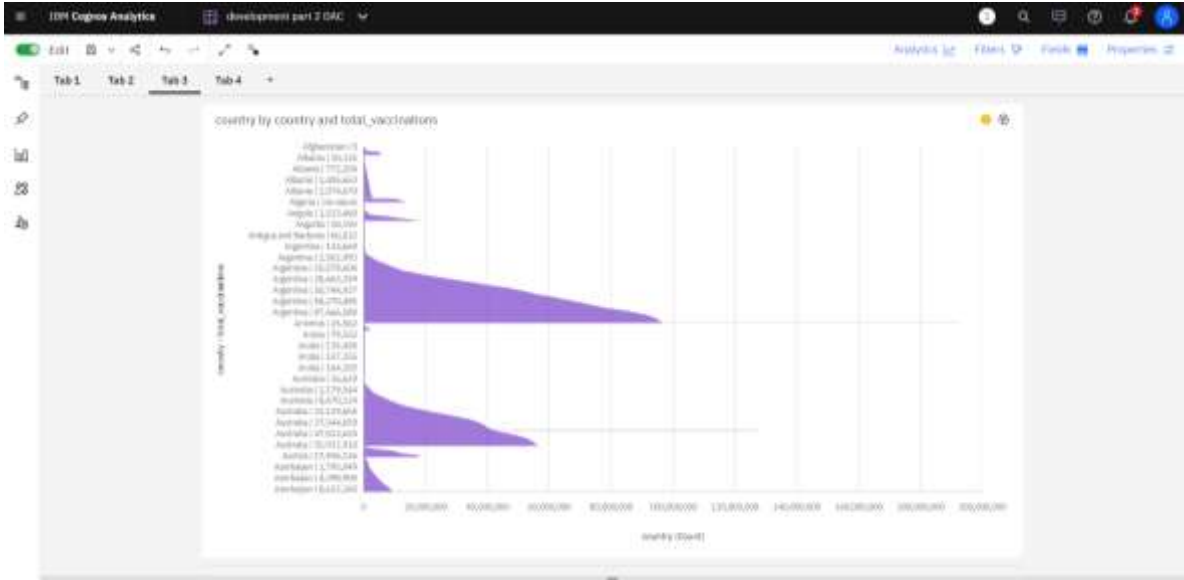


IBM NAAN MUDHALVAN

LINE CHART



BAR CHART



IBM NAAN MUDHALVAN

BOX PLOT

