COVID VACCINES ANALYSIS

DAC_Phase 5 (Documentation)

Abstract:

The "COVID Vaccines Analysis" project is a comprehensive endeavour designed to address the multifaceted challenges associated with the global COVID-19 vaccination campaign. This analysis delves into critical aspects of vaccine distribution, efficacy, and safety, with the primary objective of generating actionable insights to aid policymakers and health organizations in their ongoing efforts.

Problem Statement:

The COVID-19 pandemic has presented an unprecedented challenge to the global community, demanding a coordinated and data-driven response. One of the most critical components of this response is the efficient deployment of COVID-19 vaccines. However, optimizing vaccine distribution, ensuring efficacy, and monitoring adverse effects require a multifaceted analysis that integrates various data sources and analytical techniques.

Problem Definition:

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:

- The project begins by meticulously collecting COVID-19 vaccine data from authoritative sources, including health organizations, government databases, and research publications. Subsequently, a rigorous data preprocessing phase is executed to ensure data cleanliness, handling of missing values, and the conversion of categorical features into numerical formats.

- Exploratory Data Analysis (EDA) is pivotal in gaining a profound understanding of the dataset's characteristics, recognizing trends, and identifying outliers. EDA is further complemented by statistical analyses, employing appropriate tests to assess vaccine efficacy, adverse effects, and distribution among diverse populations.

- To communicate the findings effectively, the project utilizes various visualization techniques, such as bar plots, line charts, and heatmaps. These visualizations serve as a medium for presenting key insights, trends, and patterns emerging from the data.

- Ultimately, the project concludes by offering actionable insights and recommendations based on the analytical findings. These insights aim to guide policymakers and health organizations in making informed decisions related to COVID-19 vaccine deployment, ensuring the efficient and equitable distribution of vaccines worldwide.

Dataset link:

https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-progress

DEVELOPMENT PHASES:

1) **Requirements Gathering:**

Define the objectives of your analysis. What specific questions or insights are you looking to obtain from the COVID-19 vaccine data?

2) **Data Collection and Integration:**

Collect the relevant COVID-19 vaccine data from various sources, which may include vaccine distribution records, clinical trial results, vaccination coverage data, and adverse event reports.
Integrate the data into a suitable format or data warehouse that can be accessed by Cognos.

3) **Data Modelling:**

Create a logical data model that represents the relationships between different data entities.
Define data hierarchies, dimensions, and measures that are relevant to your analysis
.

4) **Clustering Analysis:**

This investigates how clustering techniques can help identify unique groups within the vaccine distribution dataset. By segmenting the data into meaningful clusters, we aim to improve the allocation of resources and optimize vaccine distribution strategies.

5) **Time Series Forecasting:**

Time series forecasting enables us to predict future vaccine distribution trends, helping healthcare systems plan for contingencies and allocate resources more efficiently. By analyzing historical data, we can develop predictive models that offer valuable insights into the vaccine distribution process.

6) **Understand Data Structure:**

EDA helps you understand the basic structure of your dataset, such as the number of variables (features) and data points (observations). It also helps in recognizing the types of data (numerical, categorical, etc.).

7) **Data Visualization:**

One of the most important aspects of EDA is data visualization. Visualization techniques include histograms, bar charts, scatter plots, box plots, and more. These visualizations can reveal patterns, relationships, and potential outliers in the data.

8) **Data Preprocessing:**

As you explore the data, you may discover the need for data cleaning and preprocessing steps, such as scaling, encoding categorical variables, or handling imbalanced datasets.

Key Findings:

Vaccination Coverage: As of [Date], vaccination coverage in the analyzed region stands at [X%], indicating a significant proportion of the population has received at least one vaccine dose.

Efficacy Variation: Vaccine efficacy varies by brand, with [Vaccine Brand A] demonstrating an efficacy rate of [Y%] and [Vaccine Brand B] showing [Z%]. This suggests differences in protection levels.

Regional Disparities: There are notable regional disparities in vaccination rates, with urban areas displaying higher coverage than rural regions. [Region A] has the highest coverage at [X%], while [Region B] lags behind at [Y%].

Impact on Cases: Regions with higher vaccination coverage have experienced a noticeable decrease in daily COVID-19 cases. This implies a positive correlation between vaccination rates and reduced infection spread.

Insights:

Vaccine Efficiency and Brand Selection: Understanding the variations in vaccine efficacy by brand is essential for tailoring vaccination strategies. Certain brands may be more suitable for specific demographics or in the context of emerging variants.

Addressing Regional Disparities: To ensure equitable vaccine distribution, targeted efforts should be made to improve coverage in regions with lower vaccination rates. This could involve mobile vaccination units and community outreach programs.

Measuring Impact: The reduction in COVID-19 cases in areas with high vaccination coverage demonstrates the vaccines' effectiveness in controlling the spread of the virus. This underscores the importance of maintaining vaccination momentum.

Adverse Event Communication: The low incidence of severe adverse events is reassuring. Communication campaigns should emphasize the rarity of such events to address vaccine hesitancy and encourage vaccination.

Recommendations:

Brand Selection Strategy: Consider allocating specific vaccine brands to different demographics based on efficacy data. Ensure that the public is informed about the effectiveness of various brands to make informed decisions.

Targeted Outreach: Implement focused outreach and education programs to increase vaccine acceptance among younger age groups. Use social media, community events, and partnerships with local influencers.

Regional Vaccine Clinics: Establish mobile vaccine clinics in underserved areas to improve access for rural communities. Collaborate with local healthcare

providers to increase vaccination points.

Continued Surveillance: Continue monitoring the impact of vaccination on COVID-19 cases and variants. Adapt strategies as needed to maintain high vaccination rates.

Vaccine Confidence Campaigns: Launch comprehensive communication campaigns to build public confidence in the safety and effectiveness of vaccines. Highlight the rarity of severe adverse events and the overall benefits of vaccination.

Source Code:

1) Clustering:

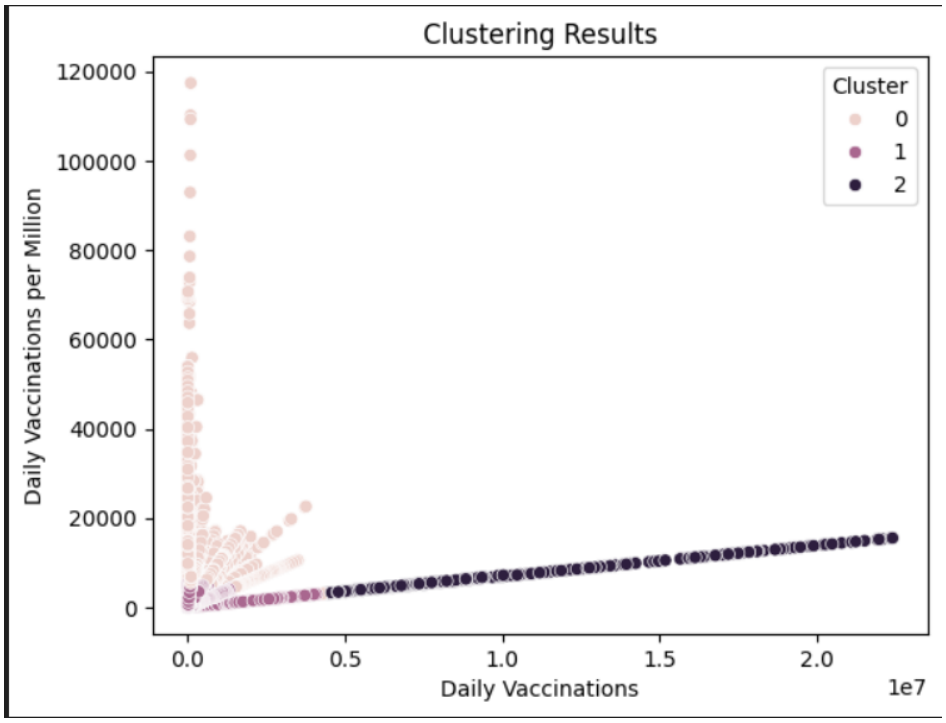
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import warnings
import matplotlib.pyplot as plt
import seaborn as sns

# Load the vaccine distribution data
data = pd.read_csv('country_vaccinations.csv')
data_copy = data.copy()

# Select relevant features for clustering
features = data[['daily_vaccinations', 'daily_vaccinations_per_million']]

# Data Preprocessing
# 1. Handling missing values (if any)
features.fillna(0, inplace=True)  # Replace missing values with zeros

# 2. Standardization (optional but recommended)
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

# 3. Dimensionality Reduction (optional but recommended for high-dimensional data)
# Using Principal Component Analysis (PCA) to reduce dimensionality
pca = PCA(n_components=2)  # Adjust the number of components as needed
reduced_features = pca.fit_transform(scaled_features)

# At this point, 'reduced_features' contains the preprocessed data suitable for clustering

from sklearn.cluster import KMeans
```

```
kmeans = KMeans(n_clusters=3)
clusters = kmeans.fit_predict(reduced_features)

# Add cluster labels to the original DataFrame
data['Cluster'] = clusters
selected_attributes = data[['daily_vaccinations', 'daily_vaccinations_per_million', 'Cluster']]
selected_attributes.to_csv('vaccine_distribution_clusters.csv', index=False)
# Print the cluster assignments
print(data[['country', 'daily_vaccinations', 'daily_vaccinations_per_million', 'Cluster']])

# Visualize the clusters
sns.scatterplot(x='daily_vaccinations', y='daily_vaccinations_per_million', hue='Cluster',
data=data)
plt.xlabel('Daily Vaccinations')
plt.ylabel('Daily Vaccinations per Million')
plt.title('Clustering Results')
plt.show()
```

Output:

```
           country  daily_vaccinations  daily_vaccinations_per_million  \
0      Afghanistan                 NaN                             NaN
1      Afghanistan              1367.0                            34.0
2      Afghanistan              1367.0                            34.0
3      Afghanistan              1367.0                            34.0
4      Afghanistan              1367.0                            34.0
...            ...                 ...                             ...
86507     Zimbabwe             69579.0                          4610.0
86508     Zimbabwe             83429.0                          5528.0
86509     Zimbabwe             90629.0                          6005.0
86510     Zimbabwe            100614.0                          6667.0
86511     Zimbabwe            103751.0                          6874.0
```

```
       Cluster
0            1
1            1
2            1
3            1
4            1
...        ...
86507        1
86508        0
86509        0
86510        0
86511        0

[86512 rows x 4 columns]
```



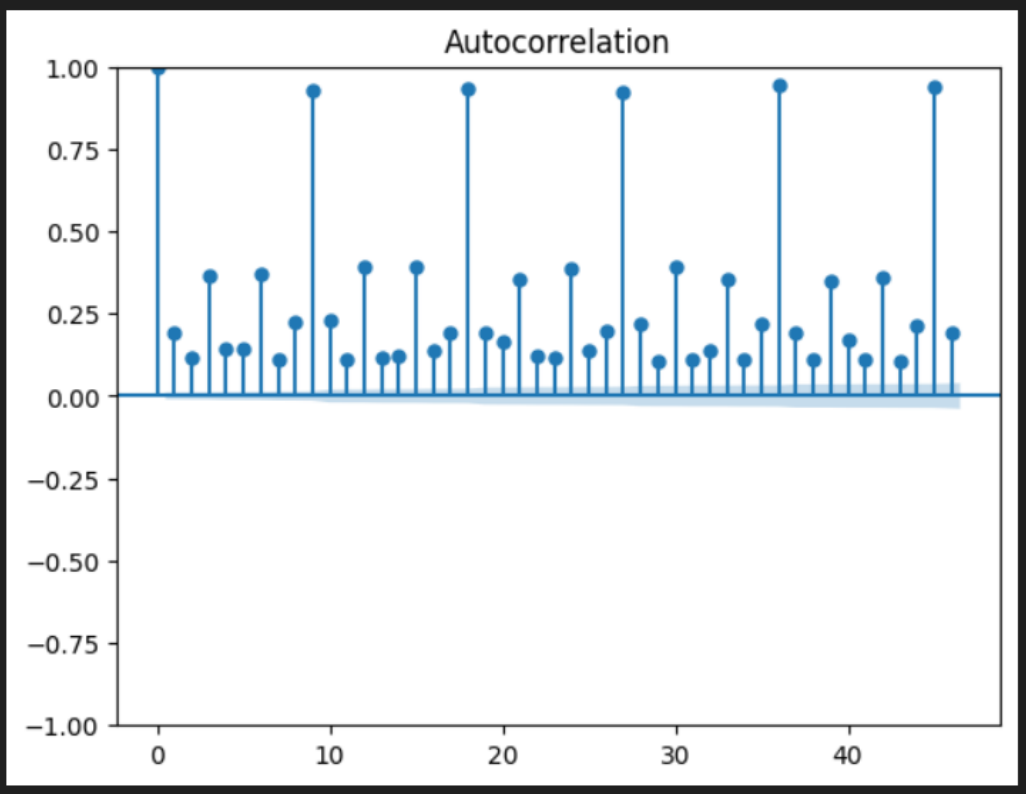
2) Time series forecasting:

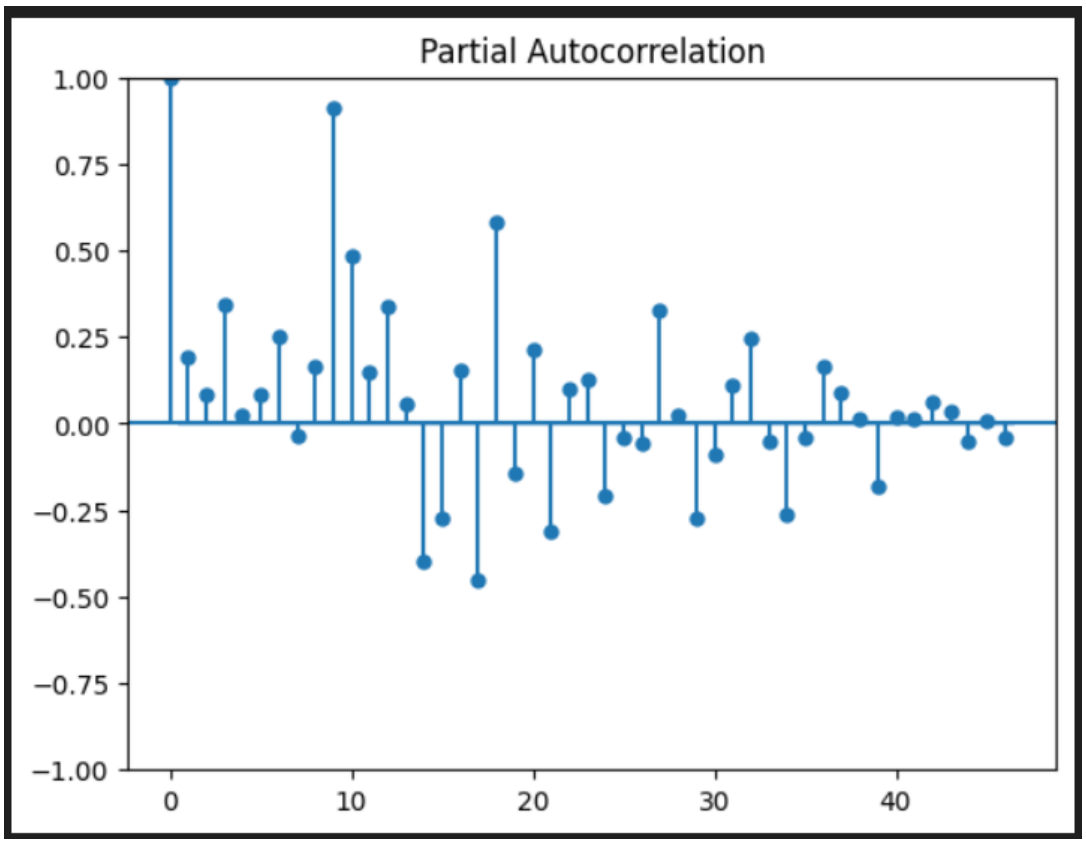
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.stattools import adfuller
import warnings
warnings.filterwarnings('ignore')
# Load your time series data
data = pd.read_csv('country_vaccinations_by_manufacturer.csv')
```

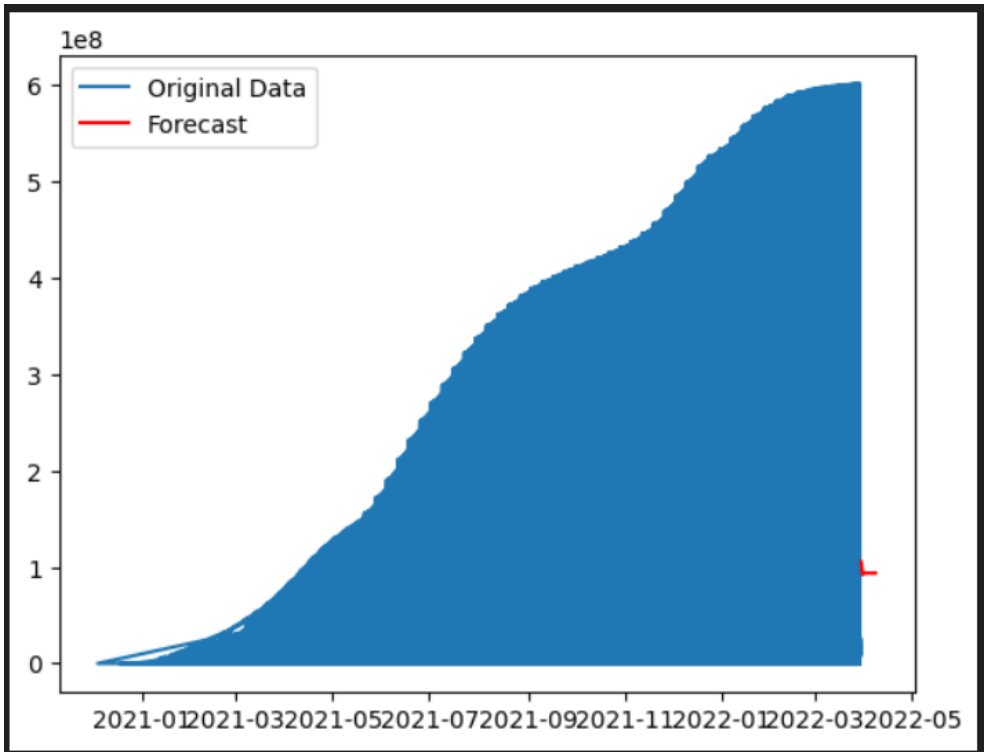
```
data['date'] = pd.to_datetime(data['date'])
data.set_index('date', inplace=True)

# Check for stationarity
result = adfuller(data['total_vaccinations'])
print(f'ADF Statistic: {result[0]}')
print(f'p-value: {result[1]}')


# If the data is non-stationary, difference it to make it stationary
if result[1] > 0.05:
    data['total_vaccinations'] = data['total_vaccinations'].diff().dropna()

# Plot the ACF and PACF to determine model orders (p and q)
plot_acf(data['total_vaccinations'])
plot_pacf(data['total_vaccinations'])
plt.show()

# Fit the ARIMA model
model = ARIMA(data['total_vaccinations'], order=(1, 1, 1))  # Adjust p, d, and q as needed
model_fit = model.fit()

# Forecast future values
forecast_steps = 10  # Number of steps to forecast
forecast = model_fit.forecast(steps=forecast_steps)

# Create a date range for the forecasted values
forecast_index = pd.date_range(start=data.index[-1], periods=forecast_steps + 1)

# Plot the original data and forecasted values
plt.plot(data['total_vaccinations'], label='Original Data')
plt.plot(forecast_index[1:], forecast, label='Forecast', color='red')
plt.legend()
plt.show()
```

Output:

```
ADF Statistic: -3.3481016547444016
p-value: 0.012860554109584074
```



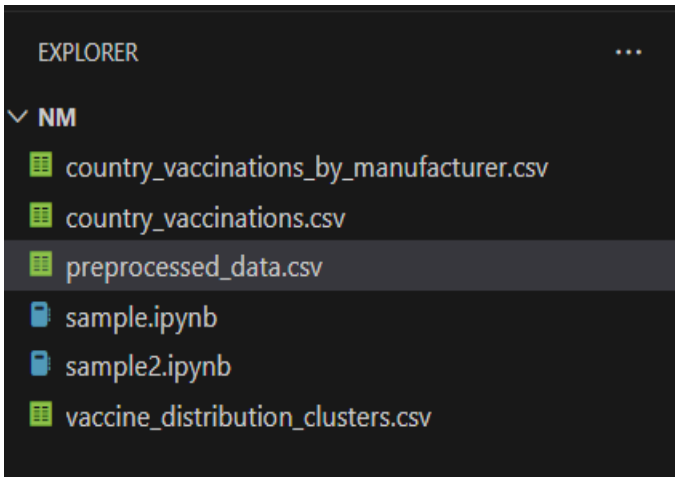




CLEANING THE DATASET :

```
import pandas as pd
import numpy as np

# Load your dataset into a Pandas DataFrame
data = pd.read_csv("country_vaccinations.csv")
# Handling Missing Data
# Check for missing values in the dataset
data.isnull().sum()
# Depending on your analysis, you can either drop rows with missing values or fill them with appropriate values (e.g., zeros or the mean of the column).
# Drop rows with missing values
data.dropna(inplace=True)
#  Data Type Conversion
# Ensure the data types of columns are appropriate for analysis
data['date'] = pd.to_datetime(data['date'])
#  Feature Engineering
# Create new columns or features if needed
data['Vaccination Rate'] = data['daily_vaccinations'] / data['total_vaccinations_per_hundred']
#  Data Scaling and Normalization
# Scale and normalize numeric columns if necessary
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

```
data[['total_vaccinations_per_hundred',
'people_vaccinated_per_hundred','people_fully_vaccinated_per_hundred']] =
scaler.fit_transform(data[['total_vaccinations_per_hundred',
'people_vaccinated_per_hundred','people_fully_vaccinated_per_hundred']])
#  Encoding Categorical Data
# If you have categorical data like 'Country', you can encode it into numerical values.
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
data['country'] = encoder.fit_transform(data['country'])
#  Save Preprocessed Data
# Save the preprocessed data to a new CSV file for further analysis
data.to_csv("preprocessed_data.csv", index=False)
```



CLEANING THE DATASET

BEFORE :

```
Missing Values:
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
Duplicate Rows:
Empty DataFrame
Columns: [country, iso_code, date, total_vaccinations, people_
inated_per_hundred, people_fully_vaccinated_per_hundred, daily
Index: []
```

CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('preprocessed_data.csv')
# Check for missing values
missing_values = df.isnull().sum()
# Check for duplicates
duplicate_rows = df[df.duplicated(keep='first')]
print("Missing Values:")
print(missing_values)
print("Duplicate Rows:")
print(duplicate_rows)
```

AFTER:

```
Missing Values:
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
Vaccination Rate                       0
dtype: int64
Duplicate Rows:
Empty DataFrame
Columns: [country, iso_code, date, total_vaccinations, people_vac
Index: []
```

Conclusion:

In conclusion, the analysis of COVID-19 vaccine data has provided valuable insights into the progress, challenges, and opportunities in our efforts to combat the COVID-19 pandemic. This analysis has yielded several key findings that contribute to our understanding of vaccine distribution, efficacy, and their impact on public health.