

Covid-19 Vaccines Analysis

Phase 4: Development Part 2

Topic: Continue Building the project Covid-19 Vaccine Analysis by Performing exploratory data analysis, Statistical analysis and Visualization



Overview of the process:

The following is the overview of the Covid-19 Vaccine Analysis project by performing exploratory data analysis, Statistical analysis and Visualization. Analyzing COVID-19 vaccine data through exploratory data analysis (EDA), statistical analysis, and visualization is a critical aspect of understanding the vaccine's effectiveness and distribution.

1.Data Collection:

The first step in this project is to gather the relevant data. This data can include information on vaccine distribution, administration, vaccine types (e.g., Pfizer, Moderna, Johnson & Johnson), demographics of vaccinated individuals, adverse events, and more. Sources of data can include government health agencies, research institutions, and vaccination clinics.'

2. Data Cleaning and Preprocessing:

Raw data often contains errors or missing values. Data cleaning involves handling missing data, removing duplicates, and converting data into a format suitable for analysis.

3. Exploratory Data Analysis (EDA):

- **Descriptive Statistics:** Compute basic statistics like mean, median, standard deviation, and percentiles to summarize the data.

- **Data Visualization:** Create visualizations such as bar charts, histograms, and box plots to understand the distribution of key variables. For instance, you might visualize the age distribution of vaccine recipients or the regional distribution of vaccinations.
- **Correlation Analysis:** Identify relationships between variables. For example, you can analyze the correlation between vaccine coverage and COVID-19 case rates.
- **Time Series Analysis:** Examine how vaccine distribution and COVID-19 cases have changed over time.

4. Statistical Analysis:

Hypothesis Testing: Formulate hypotheses and conduct statistical tests to determine if there are significant differences or associations.

For example, you can test whether there's a significant difference in vaccine efficacy between age groups.

5. Data Visualization:

Create more complex visualizations to present key insights. For instance, you can use choropleth maps to show vaccination rates by region, or you can create time series plots to visualize vaccine rollout progress.

Exploratory Data Analysis (EDA):

It is an approach to analyzing and visualizing data sets to summarize their main characteristics, often with the help of statistical graphics and various data visualization techniques. Here's a general outline of the steps typically involved in EDA along with their source code and output:

1.Data Collection:

In this stage we simply collect the dataset that we are in need to Analyse. This can be done by making using of the dataset link that is given below:

["https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-progress"](https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-progress)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	country	iso_code	date	total_vacc	people_va	people_fu	daily_vacc	daily_vacc	total_vacc	people_va	people_fu	daily_vaccination	vaccines	source_na	source_website							
2	Afghanistan	AFG	22-02-2021	0	0				0	0			Johnson&J World Hea	https://covid19.who.int/								
3	Afghanistan	AFG	23-02-2021					1367					34 Johnson&J World Hea	https://covid19.who.int/								
4	Afghanistan	AFG	24-02-2021					1367					34 Johnson&J World Hea	https://covid19.who.int/								
5	Afghanistan	AFG	25-02-2021					1367					34 Johnson&J World Hea	https://covid19.who.int/								
6	Afghanistan	AFG	26-02-2021					1367					34 Johnson&J World Hea	https://covid19.who.int/								
7	Afghanistan	AFG	27-02-2021					1367					34 Johnson&J World Hea	https://covid19.who.int/								
8	Afghanistan	AFG	28-02-2021	8200	8200			1367	0.02	0.02			34 Johnson&J World Hea	https://covid19.who.int/								
9	Afghanistan	AFG	01-03-2021					1580					40 Johnson&J World Hea	https://covid19.who.int/								
10	Afghanistan	AFG	02-03-2021					1794					45 Johnson&J World Hea	https://covid19.who.int/								
11	Afghanistan	AFG	03-03-2021					2008					50 Johnson&J World Hea	https://covid19.who.int/								
12	Afghanistan	AFG	04-03-2021					2221					56 Johnson&J World Hea	https://covid19.who.int/								
13	Afghanistan	AFG	05-03-2021					2435					61 Johnson&J World Hea	https://covid19.who.int/								
14	Afghanistan	AFG	06-03-2021					2649					66 Johnson&J World Hea	https://covid19.who.int/								
15	Afghanistan	AFG	07-03-2021					2862					72 Johnson&J World Hea	https://covid19.who.int/								
16	Afghanistan	AFG	08-03-2021					2862					72 Johnson&J World Hea	https://covid19.who.int/								
17	Afghanistan	AFG	09-03-2021					2862					72 Johnson&J World Hea	https://covid19.who.int/								
18	Afghanistan	AFG	10-03-2021					2862					72 Johnson&J World Hea	https://covid19.who.int/								
19	Afghanistan	AFG	11-03-2021					2862					72 Johnson&J World Hea	https://covid19.who.int/								
20	Afghanistan	AFG	12-03-2021					2862					72 Johnson&J World Hea	https://covid19.who.int/								
21	Afghanistan	AFG	13-03-2021					2862					72 Johnson&J World Hea	https://covid19.who.int/								
22	Afghanistan	AFG	14-03-2021					2862					72 Johnson&J World Hea	https://covid19.who.int/								
23	Afghanistan	AFG	15-03-2021					2862					72 Johnson&J World Hea	https://covid19.who.int/								
24	Afghanistan	AFG	16-03-2021	54000	54000			2862	0.14	0.14			72 Johnson&J World Hea	https://covid19.who.int/								
25	Afghanistan	AFG	17-03-2021					2882					72 Johnson&J World Hea	https://covid19.who.int/								
26	Afghanistan	AFG	18-03-2021					2902					73 Johnson&J World Hea	https://covid19.who.int/								
27	Afghanistan	AFG	19-03-2021					2921					73 Johnson&J World Hea	https://covid19.who.int/								
28	Afghanistan	AFG	20-03-2021					2941					74 Johnson&J World Hea	https://covid19.who.int/								

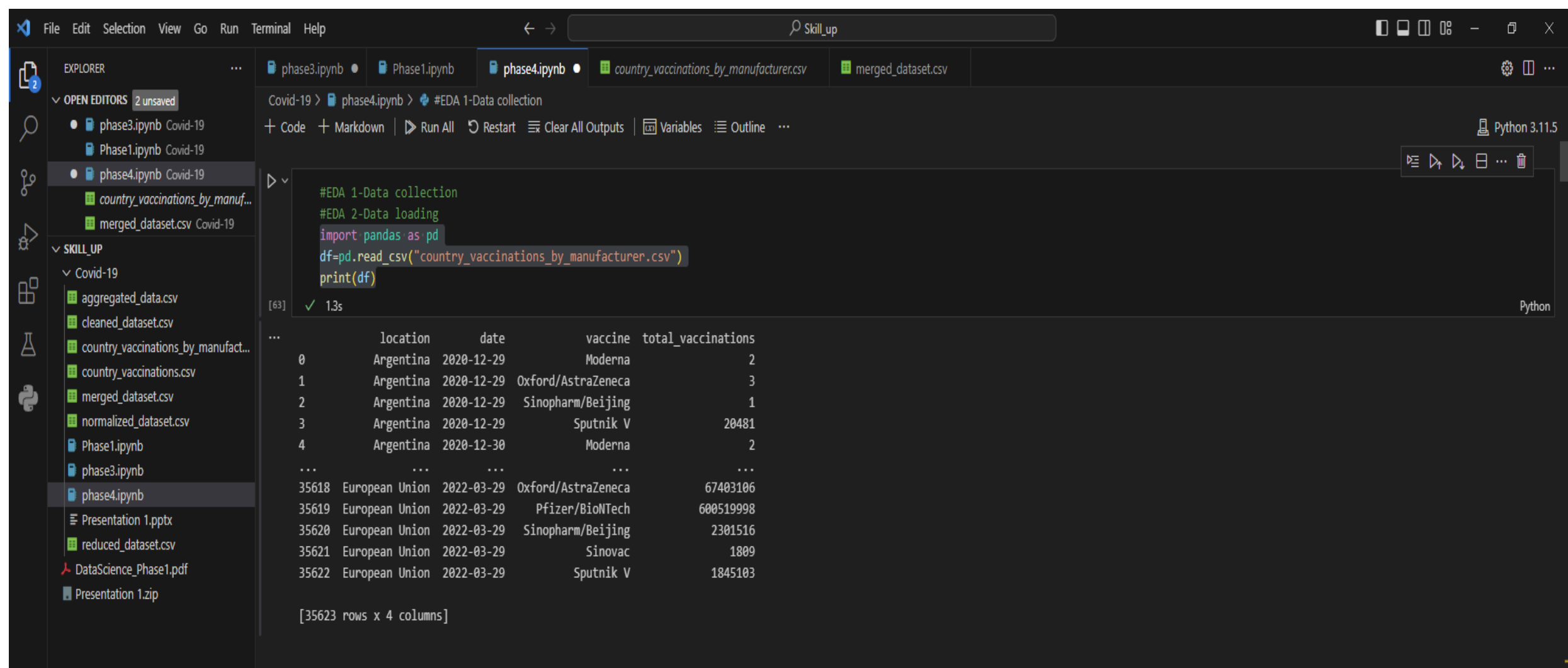
2.Data Loading:

To load a dataset in Python, we make use of various libraries, such as Pandas, NumPy, and scikit-learn, depending on the dataset's format and our specific requirements. The term "dataset" is quite broad, so the method that we use may vary depending on whether we have a CSV file, Excel file, SQL database, or some other format. Since we are given a csv file, we make use of the `read_csv()` method to read the given dataset file.

Program:


```
import pandas as pd
df=pd.read_csv("country_vaccinations_by_manufacturer.csv")
print(df)
```

Output:



```
#EDA 1-Data collection
#EDA 2-Data loading
import pandas as pd
df=pd.read_csv("country_vaccinations_by_manufacturer.csv")
print(df)
```

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

[35623 rows x 4 columns]

3.Data Inspection:

Data inspection is an important step in the Exploratory Data Analysis (EDA) process. It involves examining and understanding the raw data we are working with before diving into more complex analysis. This can be started by inspecting the dataset to understand its basic characteristics. To do it so we make use of commands like `head()`, `info()`, and `describe()` in pandas to check the first few rows, data types, and summary statistics.

Program:

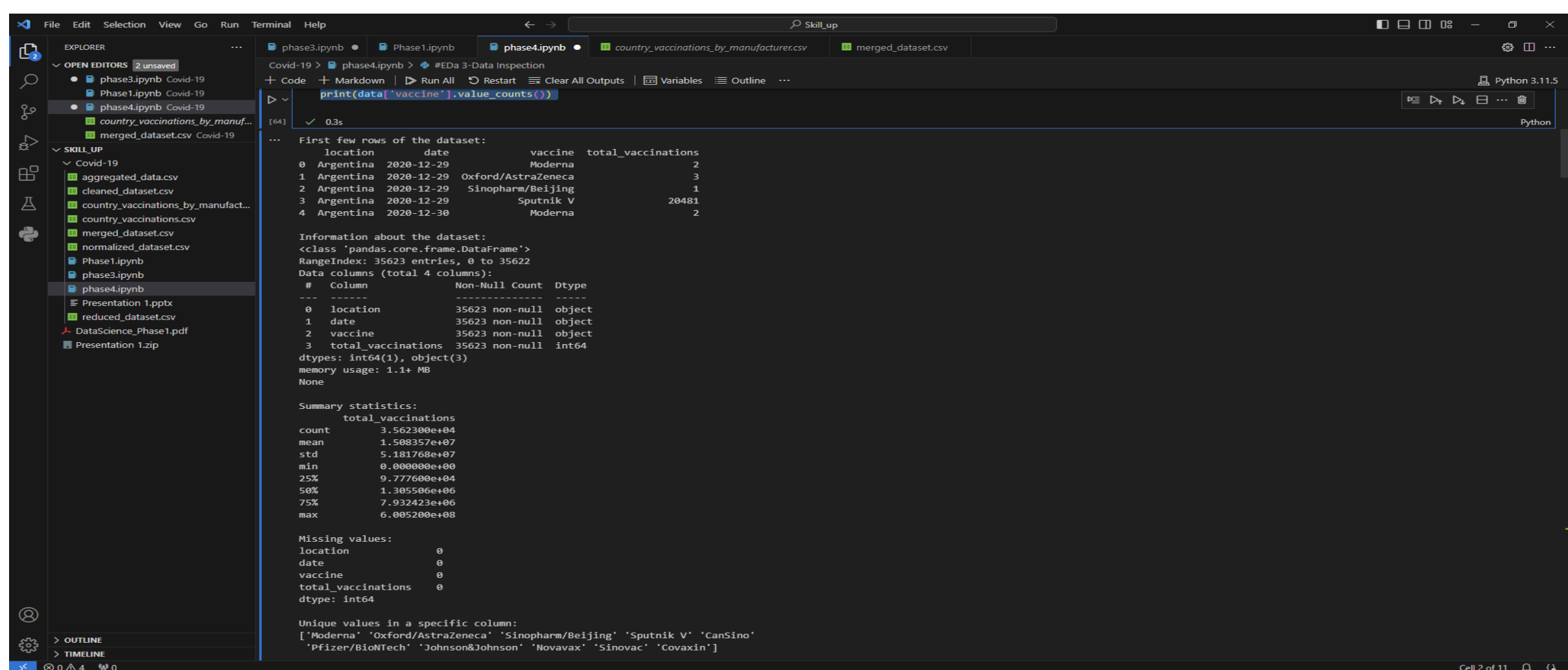
```
import pandas as pd
data =
pd.read_csv('country_vaccinations_by_manufacturer.csv')
print("First few rows of the dataset:")
```

```

print(data.head())
print("\nInformation about the dataset:")
print(data.info())
print("\nSummary statistics:")
print(data.describe())
print("\nMissing values:")
print(data.isnull().sum())
print("\nUnique values in a specific column:")
print(data['vaccine'].unique())
print("\nCount of unique values in a specific column:")
print(data['vaccine'].value_counts())

```

Output:



```

print(data["vaccine"].value_counts())

```

First few rows of the dataset:

	location	date	vaccine	total_vaccinations
0	Argentina	2020-12-29	Moderna	2
1	Argentina	2020-12-29	Oxford/AstraZeneca	3
2	Argentina	2020-12-29	Sinopharm/Beijing	1
3	Argentina	2020-12-29	Sputnik V	20481
4	Argentina	2020-12-30	Moderna	2

Information about the dataset:

```

<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
None

```

Summary statistics:

```

total_vaccinations
count      3.562300e+04
mean       1.508357e+07
std        5.181768e+07
min        0.000000e+00
25%       9.777600e+04
50%       1.305500e+06
75%       7.932422e+06
max        6.005200e+08

```

Missing values:

```

location      0
date          0
vaccine       0
total_vaccinations  0
dtype: int64

```

Unique values in a specific column:

```

['Moderna' 'Oxford/AstraZeneca' 'Sinopharm/Beijing' 'Sputnik V' 'CanSino'
 'Pfizer/BioNTech' 'Johnson&Johnson' 'Novavax' 'Sinovac' 'Covaxin']

```

4. Handling Missing Data:

Identify and address missing data. Use methods like dropping rows with missing values or imputing missing values with appropriate strategies.

Check for missing or null values in the dataset. Missing data can impact your analysis, and you may need to decide how to handle it.

Functions like `isnull()` or `isna()` in Pandas can be used to identify missing values.

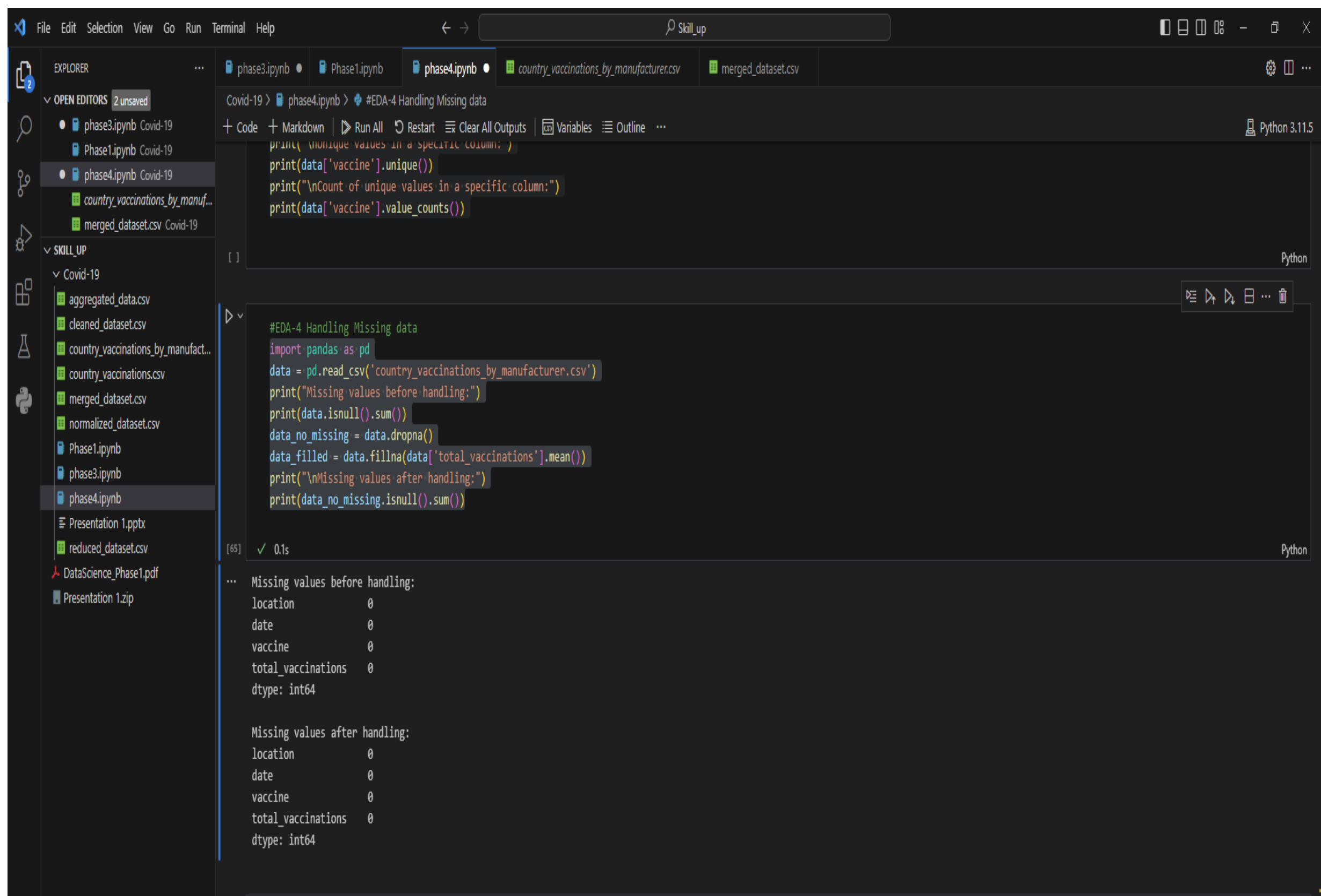
Handling missing data is a critical aspect of data analysis and is often necessary during the data preprocessing phase of an analysis project.

Missing data can arise for various reasons, including errors in data collection, data entry issues, or non-responses in surveys.

Program:

```
import pandas as pd
data = pd.read_csv('country_vaccinations_by_manufacturer.csv')
print("Missing values before handling:")
print(data.isnull().sum())
data_no_missing = data.dropna()
data_tilled = data.fillna(data['total_vaccinations'].mean())
print("\nMissing values after handling:")
print(data_no_missing.isnull().sum())
```

Output:



The screenshot shows a Jupyter Notebook titled "#EDA-4 Handling Missing data" in a VS Code editor. The notebook is open to the "phase4.ipynb" file. The code in the notebook is as follows:

```
#EDA-4 Handling Missing data
import pandas as pd
data = pd.read_csv('country_vaccinations_by_manufacturer.csv')
print("Missing values before handling:")
print(data.isnull().sum())
data_no_missing = data.dropna()
data_filled = data.fillna(data['total_vaccinations'].mean())
print("\nMissing values after handling:")
print(data_no_missing.isnull().sum())
```

The output of the code is displayed in the console below the code cell. It shows the missing values before and after handling:

```
Missing values before handling:
location      0
date          0
vaccine       0
total_vaccinations  0
dtype: int64

Missing values after handling:
location      0
date          0
vaccine       0
total_vaccinations  0
dtype: int64
```

5. Handling Outliers:

Detect and deal with outliers, which are extreme values that can affect your analysis. Visualizations like box plots or scatter plots can help in identifying outliers. Handling outliers is an important step in data analysis to ensure that extreme or erroneous data points do not unduly influence the results of statistical analyses or data visualizations. Outliers can be caused by data entry errors, measurement errors, or rare but legitimate observations

Program:

```

import pandas as pd
import numpy as np
data = pd.read_csv('country_vaccinations_by_manufacturer.csv')
def handle_outliers(data, column_name):
    mean = data[column_name].mean()
    std = data[column_name].std()
    z_score_threshold = 3
    z_scores = (data[column_name] - mean) / std
    is_outlier = np.abs(z_scores) > z_score_threshold
    data_no_outliers = data[~is_outlier]
    data_no_outliers = data.copy()
    data_no_outliers.loc[is_outlier, column_name] =
data[column_name].median()
    return data_no_outliers
outlier_column = 'total_vaccinations'
data_no_outliers = handle_outliers(data, outlier_column)
print("Number of outliers in the column:",
sum(data_no_outliers[outlier_column] != data[outlier_column]))

```

Output:

```

#ED-4 Handling outliers
import pandas as pd
import numpy as np
data = pd.read_csv('country_vaccinations_by_manufacturer.csv')
def handle_outliers(data, column_name):
    mean = data[column_name].mean()
    std = data[column_name].std()
    z_score_threshold = 3
    z_scores = (data[column_name] - mean) / std
    is_outlier = np.abs(z_scores) > z_score_threshold
    data_no_outliers = data[~is_outlier]
    data_no_outliers = data.copy()
    data_no_outliers.loc[is_outlier, column_name] = data[column_name].median()
    return data_no_outliers
outlier_column = 'total_vaccinations'
data_no_outliers = handle_outliers(data, outlier_column)
print("Number of outliers in the column:", sum(data_no_outliers[outlier_column] != data[outlier_column]))

```

[13] ✓ 0.0s

... Number of outliers in the column: 686

6.Data Transformation:

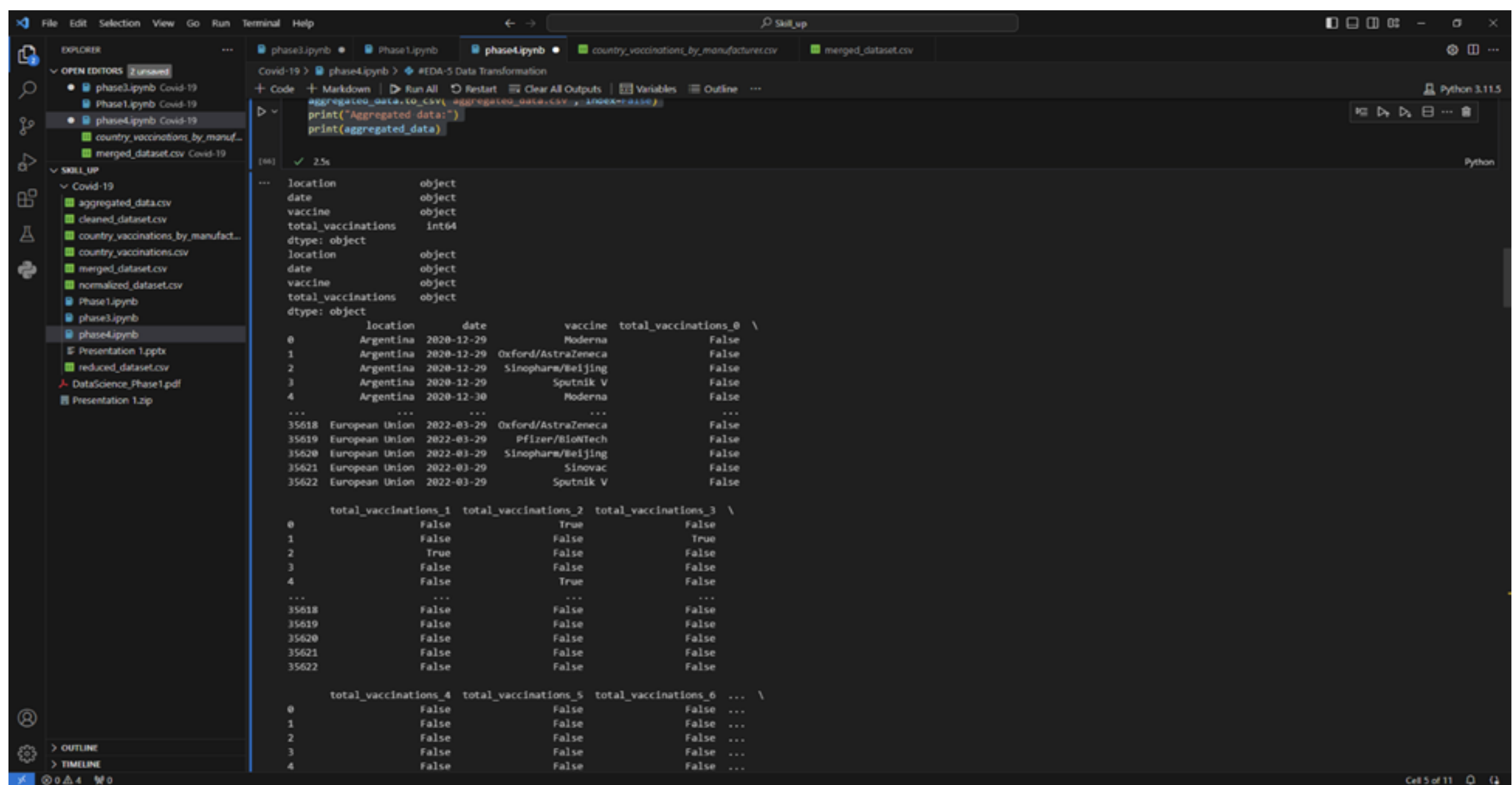
Data transformation is a fundamental step in data preprocessing and analysis. It involves modifying the

original data in various ways to make it more suitable for analysis, modeling, and visualization. Data transformation can help address issues like skewness, non-linearity, and heteroscedasticity, among others.

Program:

```
import pandas as pd
df =
pd.read_csv('country_vaccinations_by_manufacturer.csv')
print(df.dtypes)
df['total_vaccinations'] =
df['total_vaccinations'].astype(object)
print(df.dtypes)
df = pd.get_dummies(df, columns=['total_vaccinations'])
print(df)
import pandas as pd
from sklearn.preprocessing import StandardScaler
df =
pd.read_csv('country_vaccinations_by_manufacturer.csv')
columns_to_normalize = ['total_vaccinations']
scaler = StandardScaler()
df[columns_to_normalize] =
scaler.fit_transform(df[columns_to_normalize])
df.to_csv('normalized_dataset.csv', index=False)
print("Normalized dataset:")
print(df.head())
import pandas as pd
df =
pd.read_csv('country_vaccinations_by_manufacturer.csv')
grouped_data = df.groupby('date')
aggregated_data =
grouped_data['total_vaccinations'].mean().reset_index()
aggregated_data.to_csv('aggregated_data.csv', index=False)
print("Aggregated data:")
print(aggregated_data)
```

Output:



8.Univariant Analysis:

Explore individual variables using summary statistics and visualizations. Create histograms, box plots, or bar charts to understand the distribution of each variable. Univariate analysis is a statistical and data analysis technique that focuses on analyzing and summarizing the characteristics of a single variable in isolation. In other words, it examines one variable at a time to understand its distribution, central tendency, dispersion, and other key characteristics. Univariate analysis is often the first step in exploratory data analysis (EDA) and provides fundamental insights into the data.

Program:

```

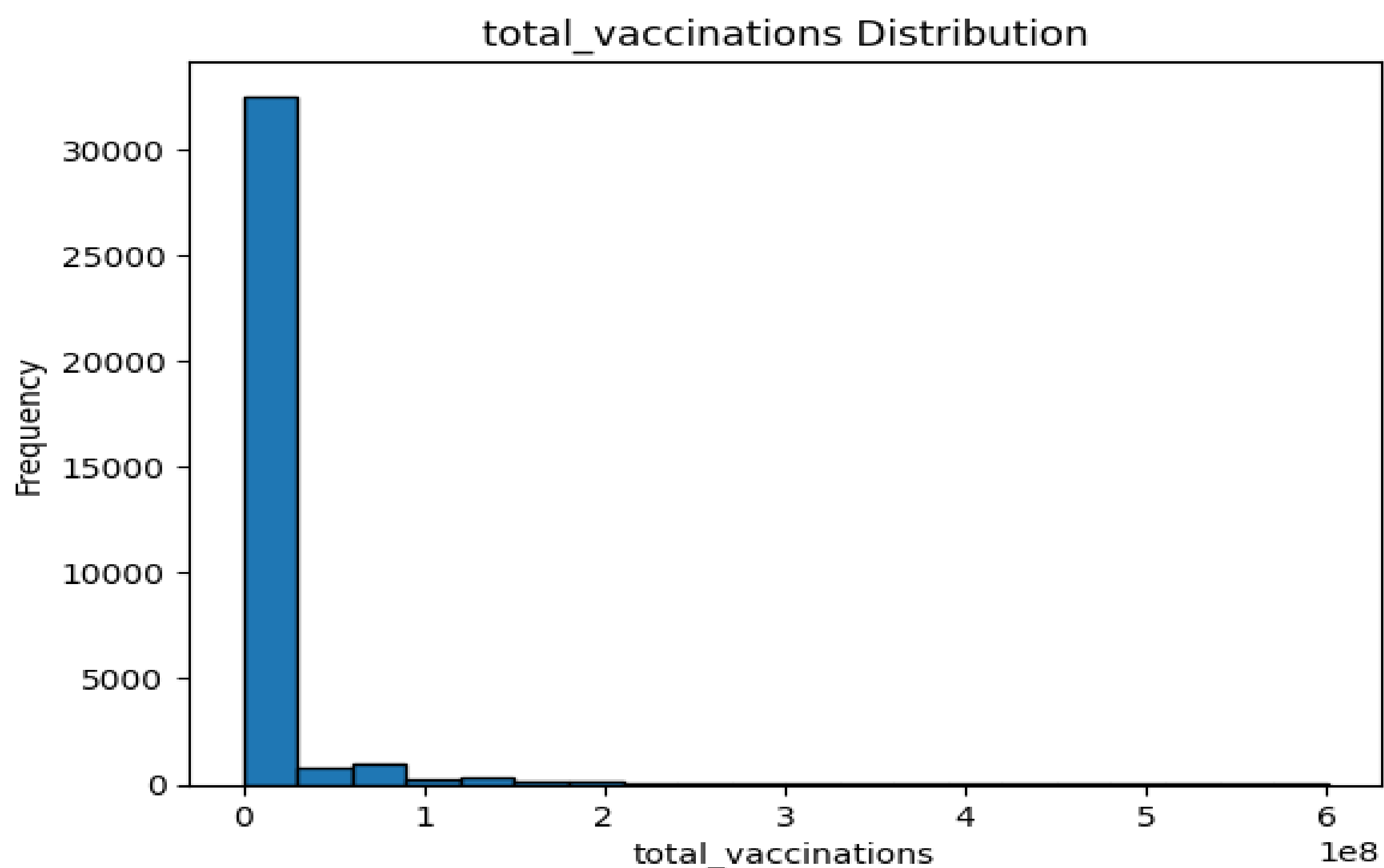
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("country_vaccinations_by_manufacturer.csv")
column_name = "total_vaccinations"
summary = df[column_name].describe()

```

```
print(f"Summary statistics for {column_name}")
{column_name}:\n{summary}\n")
plt.hist(df[column_name], bins=20, edgecolor='k')
plt.xlabel(column_name)
plt.ylabel('Frequency')
plt.title(f'{column_name} Distribution')
plt.show()
```

Output:

```
... Summary statistics for 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
Name: total_vaccinations, dtype: float64
```



Basic Exploratory Analysis:

Basic exploratory data analysis (EDA) is an essential initial step in data analysis that involves summarizing, visualizing, and understanding the main characteristics of your dataset.

exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods. It helps determine how best to manipulate data sources to get the answers you need, making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv("country_vaccinations.csv",encoding="unico
de_escape")
print(df.head())
print(df.describe())
print(df['people_vaccinated'].value_counts())
df.hist()
plt.show()
from pandas.plotting import scatter_matrix
scatter_matrix(df, figsize=(10, 10))
plt.show()
```


Output:

sense of complex information, identifying patterns, and drawing meaningful insights. The process typically begins with data collection, followed by data cleaning and preprocessing to ensure data quality. Descriptive statistics are employed to provide an initial summary of the dataset, revealing central tendencies and variability. Inferential statistics, on the other hand, are used to make predictions and test hypotheses about the population from which the data was collected. This branch of analysis encompasses a wide array of methods, including hypothesis testing, regression analysis, and analysis of variance, among others. Statistical analysis is a cornerstone in fields ranging from science and business to healthcare and social sciences, aiding in decision-making, problem-solving, and evidence-based reasoning.

Program:

```
import pandas as pd
import numpy as np
from scipy import stats
df = pd.read_csv("merged_dataset.csv")
df=df.tail(10)
summary = df.describe()
group1_data = df['total_vaccinations_x']
group2_data = df['people_vaccinated']
t_statistic, p_value = stats.ttest_ind(group1_data,
group2_data)
correlation_coefficient =
df['people_fully_vaccinated'].corr(df['daily_vaccinations_raw']
)
print("Descriptive Statistics:")
```

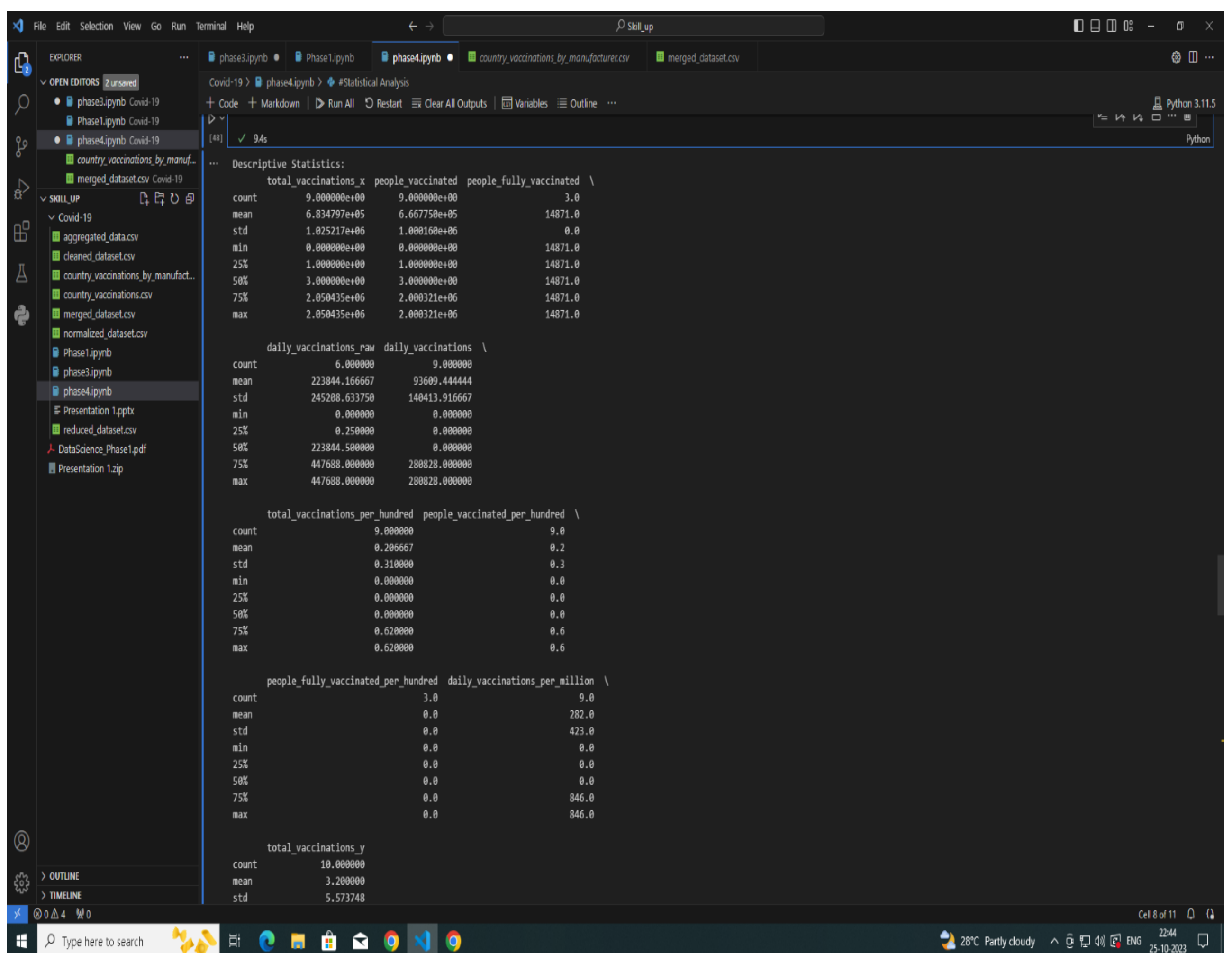


```

print(summary)
print("\nT-Test Results:")
print(f"T-Statistic: {t_statistic}")
print(f"P-Value: {p_value}")
print("\nPearson Correlation Coefficient:")
print(correlation_coefficient)

```

Output:



Data Visualization:

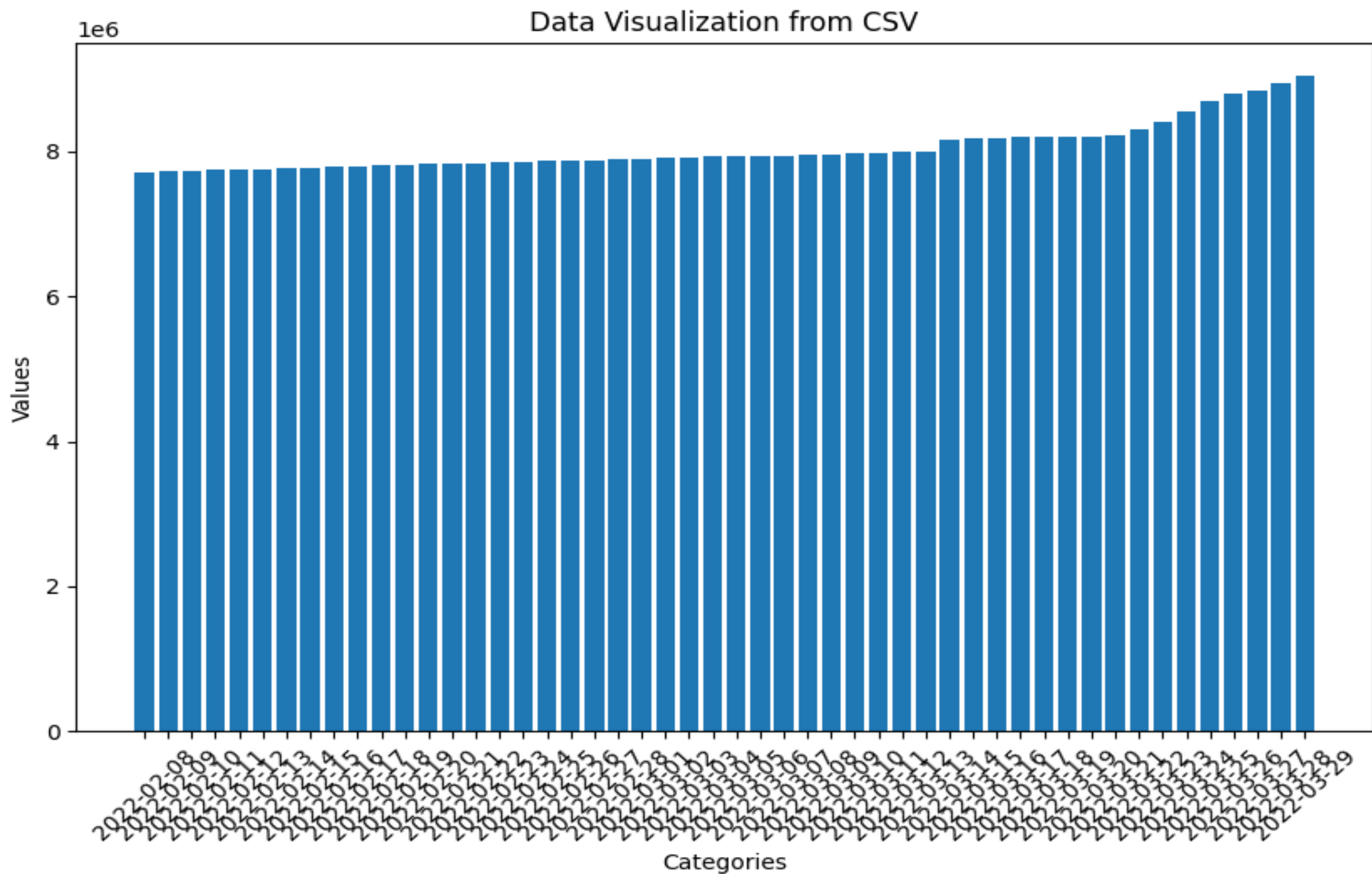
Data visualization is a powerful method of representing complex data in a visual and understandable format. It involves the creation of charts, graphs, and

diagrams to illustrate patterns, trends, and relationships within the data. By presenting information visually, data visualization makes it easier for individuals to comprehend and interpret large datasets. Data visualizations can take many forms, from simple bar charts and pie graphs to intricate heat maps and interactive dashboards. They are particularly useful for identifying outliers, correlations, and data distributions, making data more accessible and actionable. In an era of data abundance, data visualization has become an indispensable tool for turning data into knowledge and communicating findings effectively to both experts and non-experts.

Program:

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv("country_vaccinations.csv")
df=df.tail(50)
categories = df["date"]
values = df["total_vaccinations"]
plt.figure(figsize=(10, 6))
plt.bar(categories, values)
plt.xlabel("Categories")
plt.ylabel("Values")
plt.title("Data Visualization from CSV")
plt.xticks(rotation=45)
plt.show()
```

Output:



Program2:

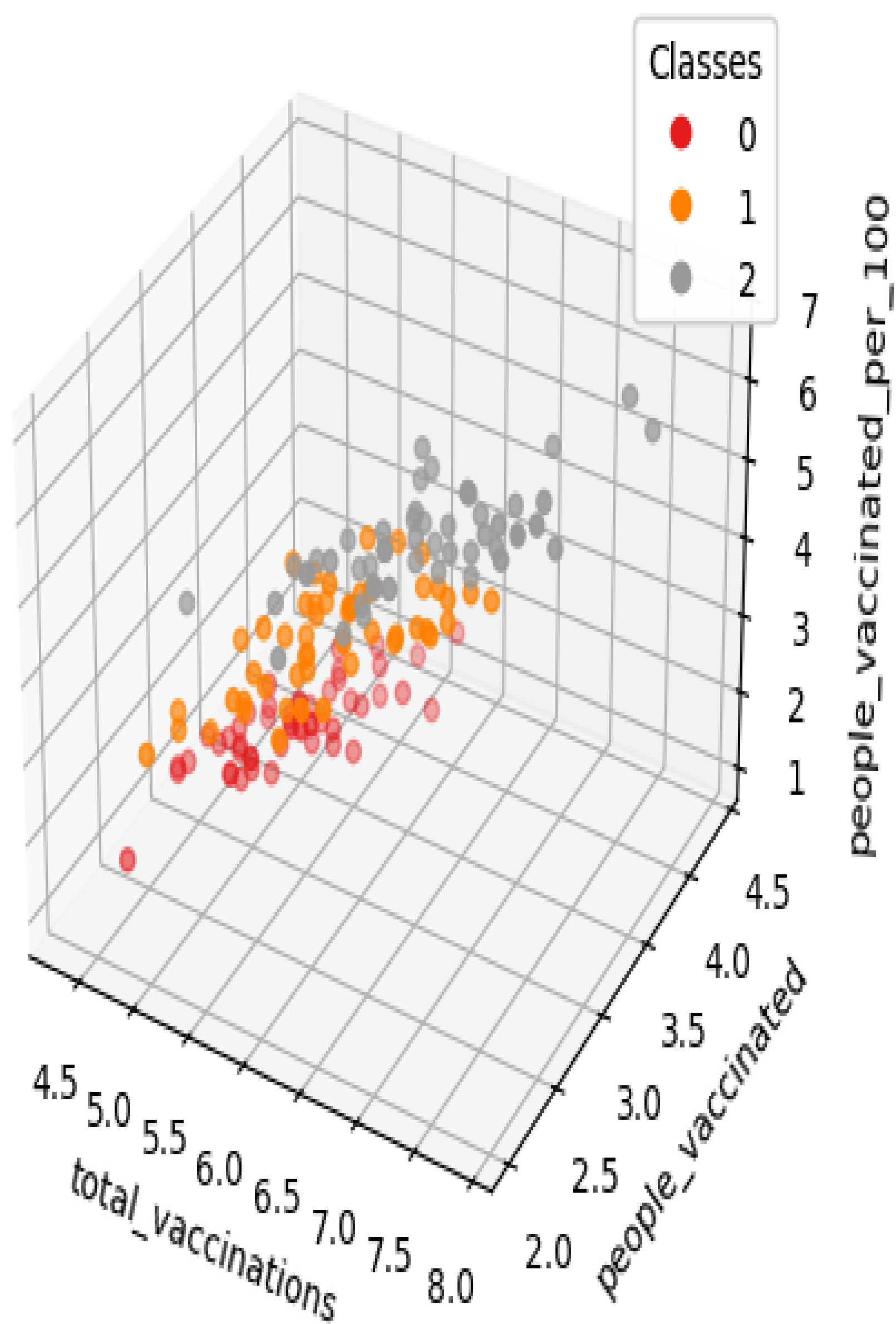
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.datasets import load_iris
df =
pd.read_csv("country_vaccinations_by_manufacturer.csv")
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y,
cmap=plt.cm.Set1)
ax.set_xlabel('total_vaccinations')
ax.set_ylabel('people_vaccinated')
ax.set_zlabel('people_vaccinated_per_100')
legend = ax.legend(*scatter.legend_elements(),
title="Classes")
ax.add_artist(legend)
```



```
ax.set_title('3D Scatter Plot of  
Country_vaccinations_by_manufacturerDataset')  
plt.show()
```

Output:

3D Scatter Plot of Country_vaccinations_by_manufacturerDataset



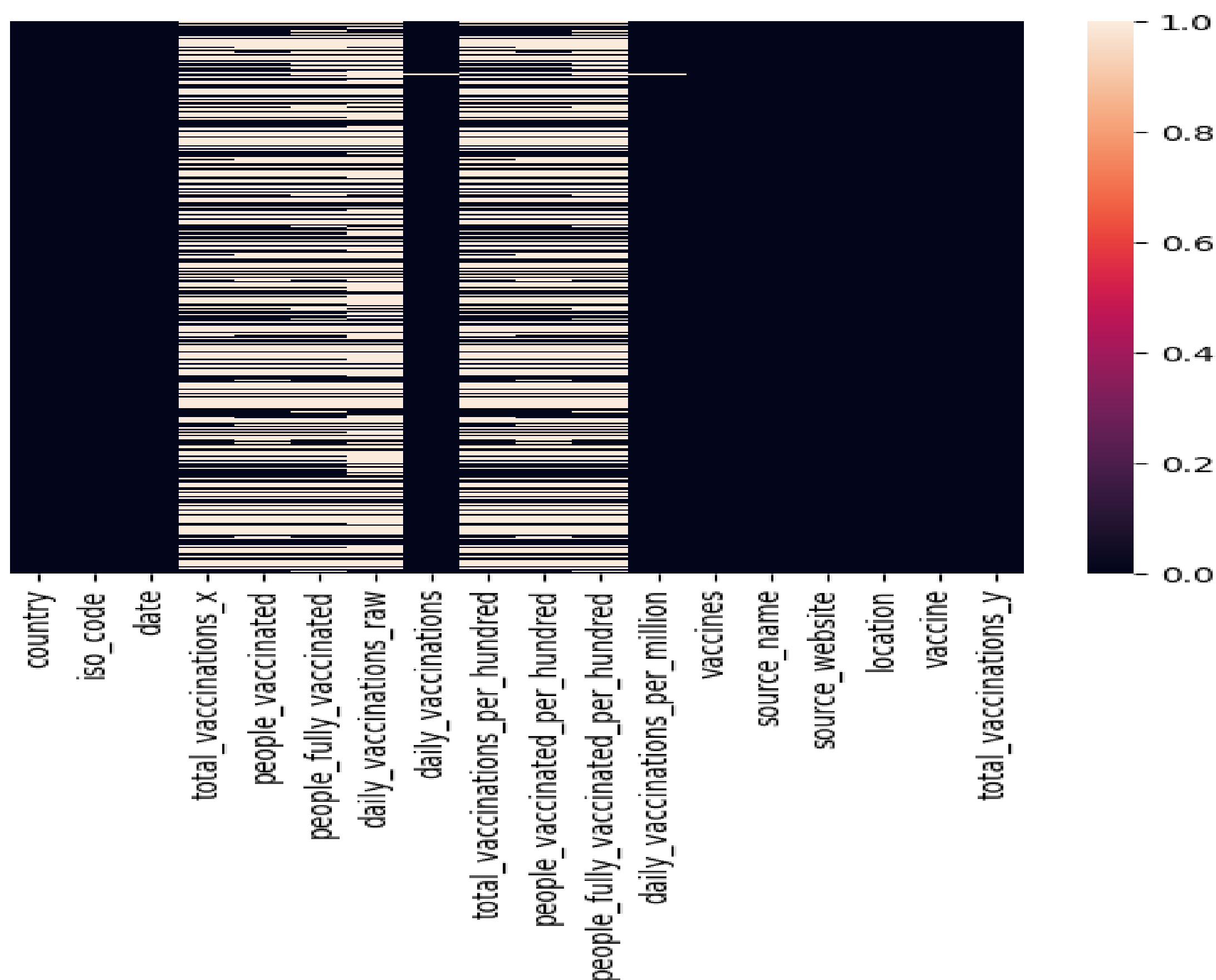
Program 3:

```

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib as plt
df=pd.read_csv("merged_dataset.csv",encoding="unicode_e
scape")
missing_values=["N/a","na",np.nan]
df=pd.read_csv("merged_dataset.csv",na_values=missing_v
alues,encoding="unicode_escape")
print(sns.heatmap(df.isnull(),yticklabels=False))

```

Output:



Conclusion:

In conclusion, the comprehensive analysis of COVID-19 vaccines through a combination of exploratory data analysis (EDA), statistical analysis, and visualization has proven instrumental in understanding the pandemic's trajectory and the impact of vaccination efforts. EDA allowed for the initial assessment of the vaccine's distribution, demographics of recipients, and adverse events, shedding light on important patterns and disparities. Statistical analysis enabled hypothesis testing, identification of vaccine efficacy, and the exploration of associations, offering valuable insights for public health decision-makers. Furthermore, data visualization played a pivotal role in making complex information accessible and aiding in the communication of critical findings to the public. The holistic approach of these analytical methods has not only contributed to informed decision-making but has also underscored the importance of data-driven strategies in addressing global health crises. As we continue to navigate the challenges posed by COVID-19, the integration of these analytical techniques remains essential in monitoring, evaluating, and optimizing vaccination campaigns, ultimately striving for a safer and healthier world.