

Covid-19 Vaccine Analysis

Phase 5 document submission

By:

Vignesh.M	4 20421106045
Sanjay harish.S	4 20421106034
Chitaardhan.S	4 20421106009
Santhosh.K	4 20421106035



Intoduction:

The COVID-19 vaccine has indeed been a significant turning point during the global quarantine and pandemic. Covid-19 Vaccines has played some of the crucial roles during Quarantine which are described as below.

- ❖ **Global Solidarity:**The pandemic and vaccine distribution have underscored the importance of global solidarity. Many countries have donated vaccines to less fortunate nations to ensure that everyone has a fair chance at protection.
- ❖ **Research and Development:**The rapid development of COVID-19 vaccines showcased the power of scientific collaboration and innovation. It has also paved the way for advancements in vaccine technology and research.
- ❖ **Travel and Mobility:**Vaccination has facilitated international travel and mobility. Many countries have implemented vaccine passport systems, allowing vaccinated individuals to move more freely and engage in activities that were previously restricted.
- ❖ **The analysis of COVID-19 vaccines** involves a multifaceted assessment of their development, clinical trials, regulatory approvals, safety and efficacy profiles, distribution, challenges, and broader public health impact.
- ❖ **As we embark on this journey into the realm of machine learning for**

Covid-19 Analysis, we will explore the various techniques, data sources, and challenges involved.

Dataset Link:

The dataset for the project can be downloaded from the given data set link below.

“ <https://www.kaggle.com/datasets/vedavyasv/usa-housing> ”

country	iso_code	date	total_vacc	people_vacc	daily_vacc	source_na
Afghanistan	AFG	22-02-2021	0	0		Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	23-02-2021			1367	34 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	24-02-2021			1367	34 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	25-02-2021			1367	34 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	26-02-2021			1367	34 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	27-02-2021			1367	34 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	28-02-2021	8200	8200	1367	34 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	01-03-2021			1580	40 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	02-03-2021			1794	45 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	03-03-2021			2008	50 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	04-03-2021			2221	56 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	05-03-2021			2435	61 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	06-03-2021			2649	66 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	07-03-2021			2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	08-03-2021			2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	09-03-2021			2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	10-03-2021			2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	11-03-2021			2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	12-03-2021			2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	13-03-2021			2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	14-03-2021			2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	15-03-2021			2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	16-03-2021	54000	54000	2862	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	17-03-2021			2882	72 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	18-03-2021			2902	73 Johnson&J World Hea https://covid19.who.int/
Afghanistan	AFG	19-03-2021			2921	73 Johnson&J World Hea https://covid19.who.int/

Here's a list of tools and software commonly used in the process:

1. Programming Language:

Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy, pandas, scikit-learn, and more.

2. Integrated Development Environment (IDE):

Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, Google Colab, or traditional IDEs like PyCharm.

3. Machine Learning Libraries:

You'll need various machine learning libraries, including:

- scikit-learn for building and evaluating machine learning models.
- TensorFlow or PyTorch for deep learning, if needed.
- XGBoost, LightGBM, or CatBoost for gradient boosting models.

4. Data Visualization Tools:

Tools like Matplotlib, Seaborn, or Plotly are essential for data exploration and visualization.

5. Data Preprocessing Tools:

Libraries like pandas help with data cleaning, manipulation, and preprocessing.

6. Data Collection and Storage:

- Depending on your data source, you might need web scraping tools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite, PostgreSQL) for data storage.

DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT

1. Empathize:

- Understand the needs and challenges of all stakeholders involved in the house price prediction process, including homebuyers, sellers, real estate professionals, appraisers, and investors.

- Conduct interviews and surveys to gather insights on what users value in property valuation and what information is most critical for their decision-making.

2. Define:

- Clearly articulate the problem statement, such as "How might we predict house prices more accurately and transparently using machine learning?"
- Identify the key goals and success criteria for the project, such as increasing prediction accuracy, reducing bias, or improving user trust in the valuation process.

3. Ideate:

- Brainstorm creative solutions and data sources that can enhance the accuracy and transparency of house price predictions.
- Encourage interdisciplinary collaboration to generate a wide range of ideas, including the use of alternative

data, new algorithms, or improved visualization techniques.

4. Prototype:

- Create prototype machine learning models based on the ideas generated during the ideation phase.
- Test and iterate on these prototypes to determine which approaches are most promising in terms of accuracy and usability.

5. Test:

- Gather feedback from users and stakeholders by testing the machine learning models with real-world data and scenarios.
- Assess how well the models meet the defined goals and success criteria, and make adjustments based on user feedback.

6. Implement:

- Develop a production-ready machine learning solution for predicting house prices, integrating the best-performing algorithms and data sources.
- Implement transparency measures, such as model interpretability tools, to ensure users understand how predictions are generated.

7. Evaluate:

- Continuously monitor the performance of the machine learning model after implementation to ensure it remains accurate and relevant in a changing real estate market.
- Gather feedback and insights from users to identify areas for improvement.

DESIGN INTO INNOVATION

1. Data Collection:

Gather a comprehensive dataset that includes features such as location, size, age, amenities, nearby schools, crime rates, and other relevant variables.

2. Data Preprocessing:

Loading and preprocessing the dataset is an important first step in building up any machine learning model. By loading and preprocessing the dataset, we can ensure the machine learning algorithm is able to learn from the data effectively and accurately.

1.Loading the dataset:

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:

Since we are given two datasets, we are going to load both of these datasets separately.

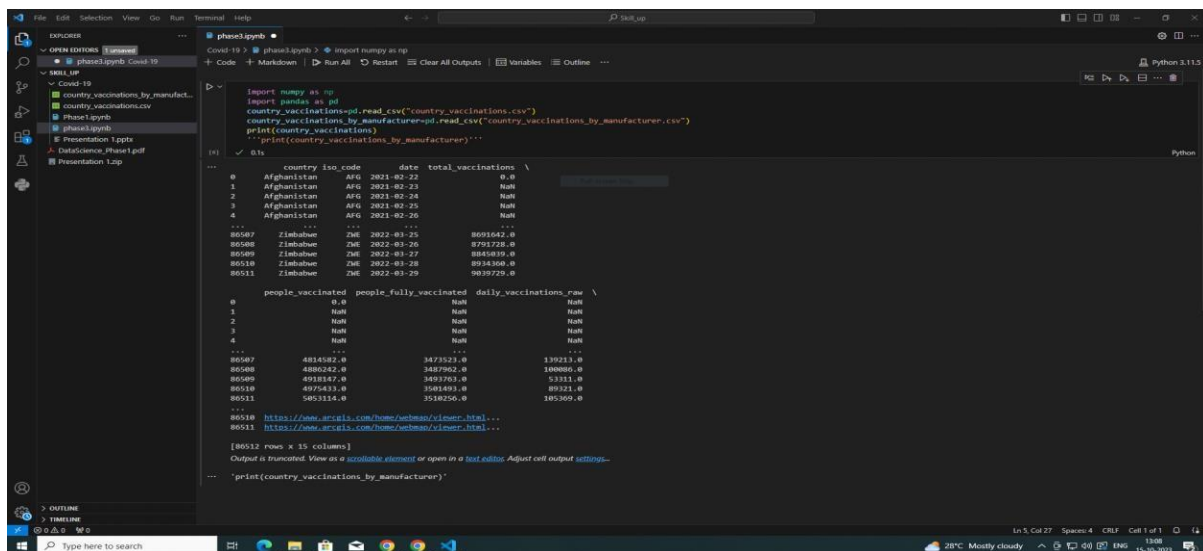
1.Dataset named country_vaccinations:

```
import numpy as np
import pandas as pd
country_vaccinations=pd.read_csv("country_vaccinations.csv")
print(country_vaccinations)
```

2. Dataset named country_vaccinations_by_manufacturer:

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

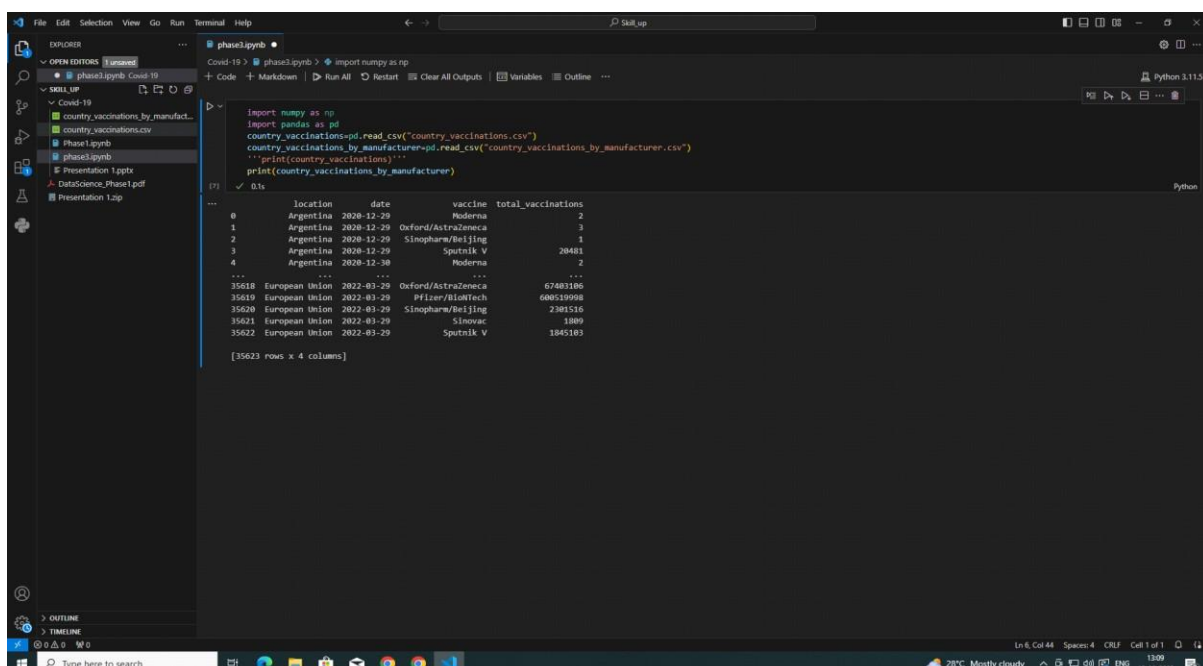
OUTPUT:



```
import numpy as np
import pandas as pd
country_vaccinations=pd.read_csv("country_vaccinations.csv")
country_vaccinations_by_manufacturer=pd.read_csv("country_vaccinations_by_manufacturer.csv")
print(country_vaccinations)
print(country_vaccinations_by_manufacturer)'''
```

country_iso_code	date	total_vaccinations
0	Afghanistan AFG 2021-02-22	0.0
1	Afghanistan AFG 2021-02-23	NaN
2	Afghanistan AFG 2021-02-24	NaN
3	Afghanistan AFG 2021-02-25	NaN
4	Afghanistan AFG 2021-02-26	NaN
...
80507	Zimbabwe ZWE 2022-03-25	8605642.0
80508	Zimbabwe ZWE 2022-03-26	87951228.0
80509	Zimbabwe ZWE 2022-03-27	88450839.0
80510	Zimbabwe ZWE 2022-03-28	89343068.0
80511	Zimbabwe ZWE 2022-03-29	90207229.0

```
...
people_vaccinated people_fully_vaccinated daily_vaccinations_raw
0 0.0 NaN NaN
1 NaN NaN NaN
2 NaN NaN NaN
3 NaN NaN NaN
4 NaN NaN NaN
...
80507 4814582.0 3473523.0 139213.0
80508 4880242.0 3487982.0 180860.0
80509 4918147.0 3493763.0 53311.0
80510 4975433.0 3501493.0 89321.0
80511 5053115.0 3510206.0 105369.0
...
[80512 rows x 5 columns]
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
'''
print(country_vaccinations_by_manufacturer)'''
```



```
import numpy as np
import pandas as pd
country_vaccinations=pd.read_csv("country_vaccinations.csv")
country_vaccinations_by_manufacturer=pd.read_csv("country_vaccinations_by_manufacturer.csv")
print(country_vaccinations)
print(country_vaccinations_by_manufacturer)'''
```

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	67483186
35619	European Union 2022-03-29	Pfizer/BioNTech	600519998
35620	European Union 2022-03-29	Sinopharm/Beijing	2301536
35621	European Union 2022-03-29	Sinovac	1889
35622	European Union 2022-03-29	Sputnik V	1845183

```
...
[35623 rows x 4 columns]
```


2.Preprocessing the dataset:

- Data preprocessing is the process of cleaning, transforming, and integrating the data in order to make it ready for analysis.
- This may involve removing errors and inconsistencies, handling missing values, transforming the data into a consistent format, and scaling the data to a suitable range.

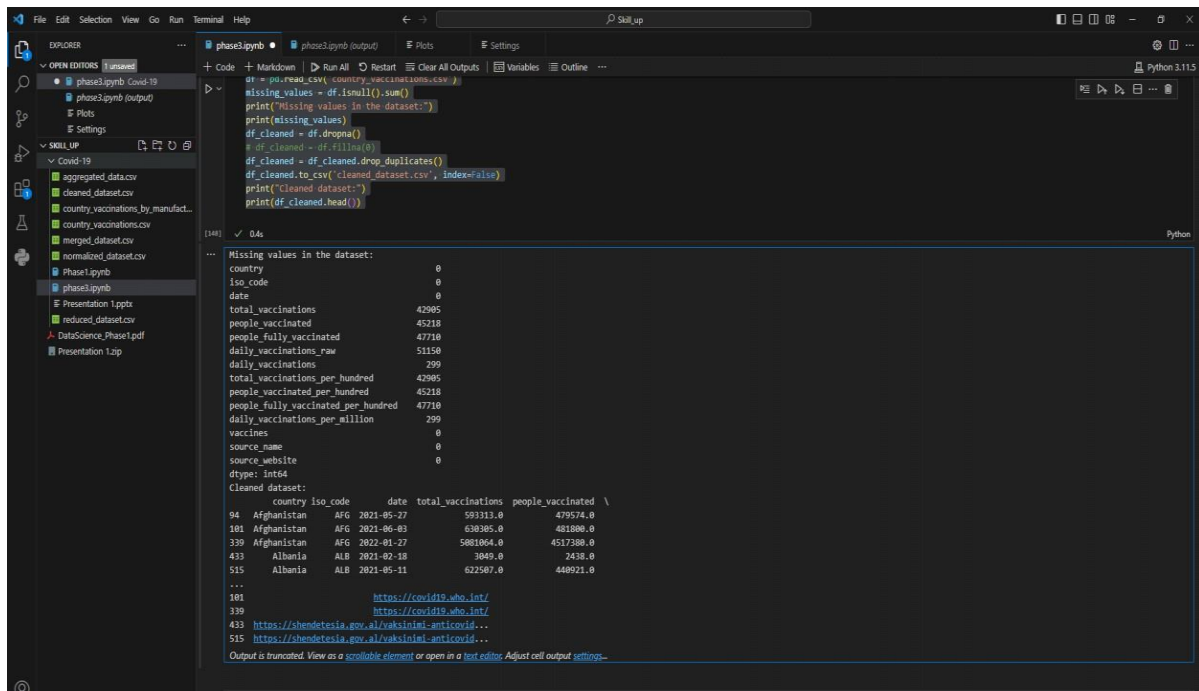
2.1 Data Cleansing:

1) Data cleansing for the dataset country_vaccinations are represented pictorially below with their source code. It Includes removing missing values in the dataset, dropping those missing values.

Program:

```
import pandas as pd
df = pd.read_csv('country_vaccinations.csv')
missing_values = df.isnull().sum()
print("Missing values in the dataset:")
print(missing_values)
df_cleaned = df.dropna()
# df_cleaned = df.fillna(0)
df_cleaned = df_cleaned.drop_duplicates()
df_cleaned.to_csv('cleaned_dataset.csv', index=False)
print("Cleaned dataset:")
print(df_cleaned.head())
```

Output:



The screenshot shows a Jupyter Notebook environment. The left sidebar displays the file explorer with a project named 'COVID-19' containing various CSV files and a Jupyter Notebook 'phase3.ipynb'. The main area shows the code from 'phase3.ipynb' being executed. The code reads a CSV file, checks for missing values, cleans the data by dropping NA values and duplicates, and saves the cleaned dataset. The output shows the missing values for each column, the cleaned dataset's dtypes, and a preview of the cleaned data.

```
df = pd.read_csv('country_vaccinations.csv')
missing_values = df.isnull().sum()
print("Missing values in the dataset:")
print(missing_values)
df_cleaned = df.dropna()
# df_cleaned = df.fillna(0)
df_cleaned = df_cleaned.drop_duplicates()
df_cleaned.to_csv('cleaned_dataset.csv', index=False)
print("Cleaned dataset:")
print(df_cleaned)
```

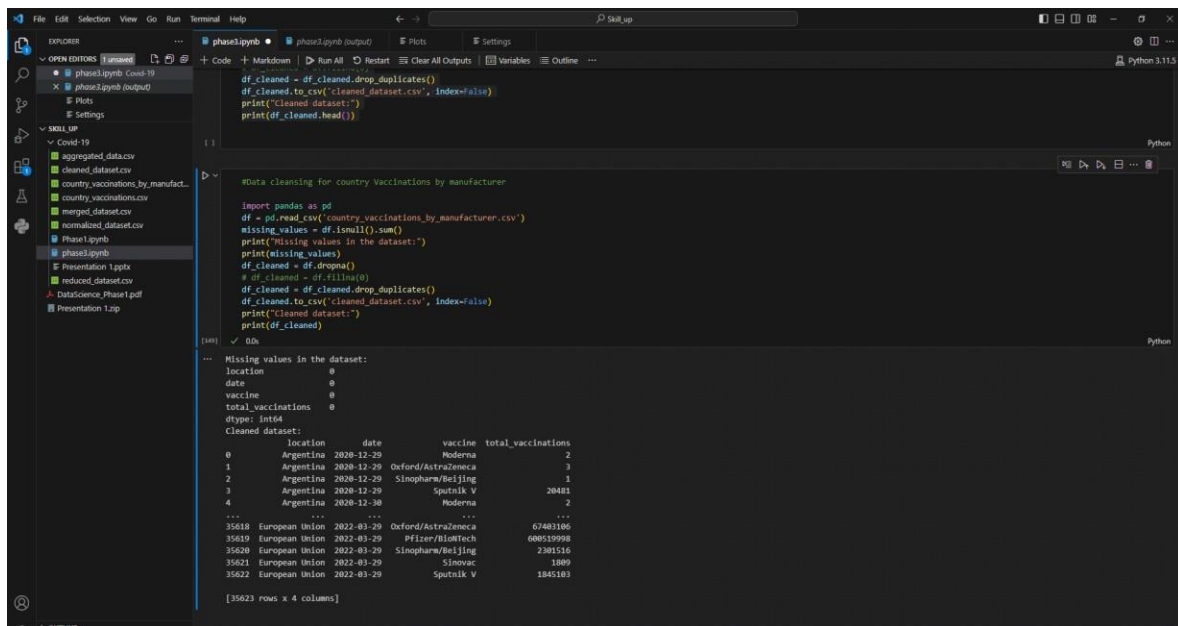
Output:

```
Missing values in the dataset:
country          0
iso_code         0
date            0
total_vaccinations    42985
people_vaccinated    45218
people_fully_vaccinated 47718
daily_vaccinations_raw 51150
daily_vaccinations    299
total_vaccinations_per_hundred    42985
people_vaccinated_per_hundred    45218
people_fully_vaccinated_per_hundred 47718
daily_vaccinations_per_million    299
vaccines          0
source_name       0
source_website    0
dtype: int64
Cleaned dataset:
   country iso_code  date  total_vaccinations  people_vaccinated \
94  Afghanistan  AFG  2021-05-27          593113.0          479574.0
101 Afghanistan  AFG  2021-06-03          639385.0          481880.0
339 Afghanistan  AFG  2022-01-27          5881864.0         4517388.0
433  Albania    ALB  2021-02-18           3049.0           2438.0
515  Albania    ALB  2021-05-11          622587.0          448921.0
...
101  https://covid19.who.int/
339  https://covid19.who.int/
433  https://shendetesia.gov.al/yakiniini-anticovid...
515  https://shendetesia.gov.al/yakiniini-anticovid...
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

2) Now we will perform Data cleansing for the dataset `country_vaccinations_by_manufacture` is performed using the following code and represented pictorially below with their output:

```
import pandas as pd
df =
pd.read_csv('country_vaccinations_by_manufacturer.csv')
missing_values = df.isnull().sum()
print("Missing values in the dataset:")
print(missing_values)
df_cleaned = df.dropna()
# df_cleaned = df.fillna(0)
df_cleaned = df_cleaned.drop_duplicates()
df_cleaned.to_csv('cleaned_dataset.csv', index=False)
print("Cleaned dataset:")
print(df_cleaned)
```

Output:



The screenshot shows a Jupyter Notebook with two cells. The first cell contains code to clean a dataset by dropping duplicates and saving it to a CSV file. The second cell contains code to clean a dataset for country vaccinations by manufacturer, including handling missing values and dropping duplicates. The output of the second cell shows the missing values in the dataset and the cleaned dataset.

```
df_cleaned = df_cleaned.drop_duplicates()
df_cleaned.to_csv('cleaned_dataset.csv', index=False)
print(df_cleaned.head())
```

```
#Data cleansing for country Vaccinations by manufacturer

import pandas as pd
df = pd.read_csv('country_vaccinations_by_manufacturer.csv')
missing_values = df.isnull().sum()
print("Missing values in the dataset:")
print(missing_values)
df_cleaned = df.dropna()
df_cleaned = df_cleaned.drop_duplicates()
df_cleaned.to_csv('cleaned_dataset.csv', index=False)
print("Cleaned dataset:")
print(df_cleaned)
```

```
Missing values in the dataset:
location      0
date          0
vaccine       0
total_vaccinations  0
dtype: int64
Cleaned 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
...      ...      ...      ...
35618  European Union  2022-03-29  Oxford/AstraZeneca  67483106
35619  European Union  2022-03-29  Pfizer/BioNTech  608519998
35620  European Union  2022-03-29  Sinopharm/Beijing  23051516
35621  European Union  2022-03-29  Sinovac                1809
35622  European Union  2022-03-29  Sputnik V       1845183
[35623 rows x 4 columns]
```

2.2 Data Integration:

Data integration, as a vital component of data preprocessing, involves merging and harmonizing data from diverse sources into a unified data set. Imagine a sample dataset related to COVID-19 vaccinations, where information comes from multiple channels like healthcare providers, government agencies, and research institutions. Each source may have its own data format, terminology, and structure. Data integration in this context involves bringing together these heterogeneous datasets, aligning data fields, and ensuring consistency. It helps in harmonizing the data to form a single, coherent dataset. This integrated dataset can then be used to analyze vaccination trends, vaccine effectiveness, and public health outcomes.

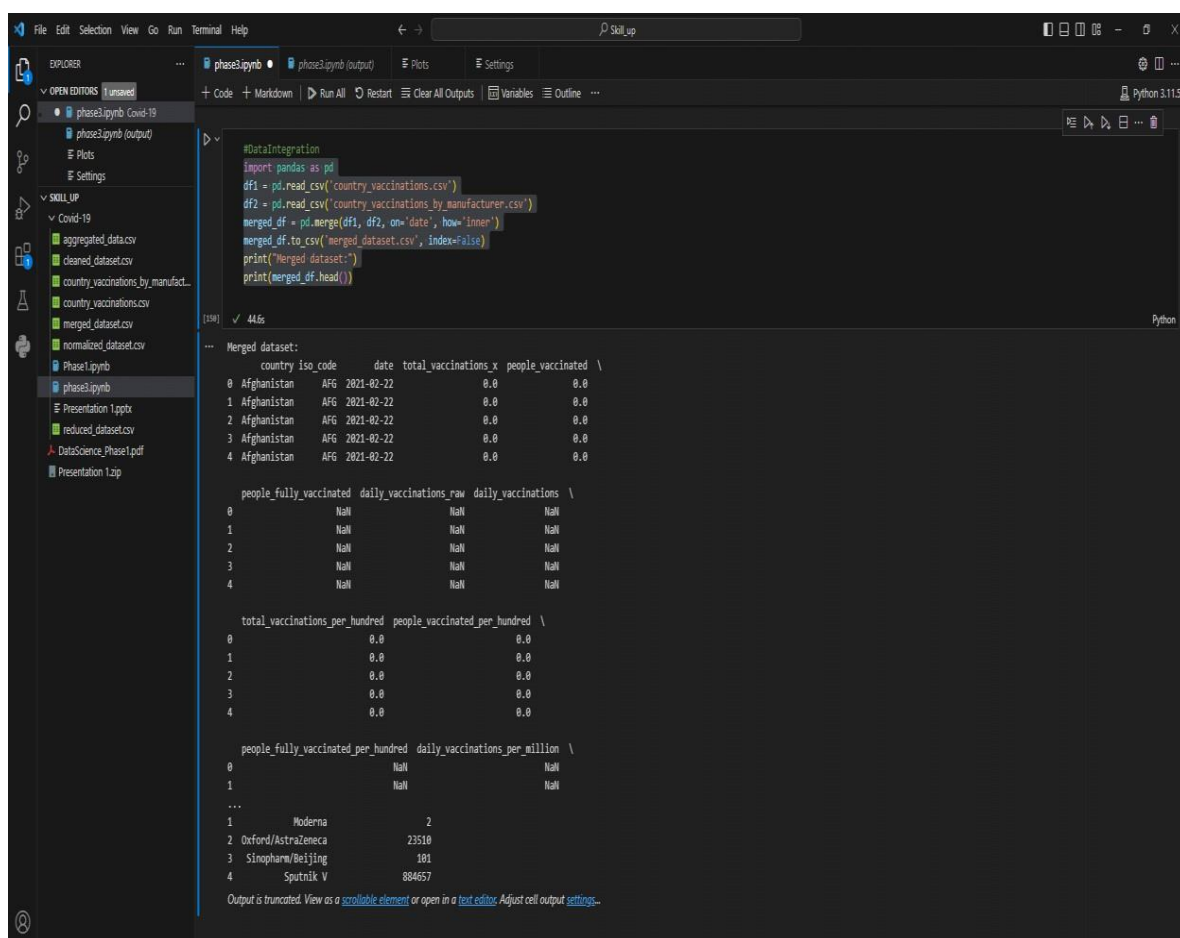
So, we have created an integrated dataset by making use of the two different datasets that we are given. The integrated data sets are created separately and stored as csv files.

The pictorial representation of integrated dataset's output is given below along with the source code:

Program:

```
import pandas as pd
df1 = pd.read_csv('country_vaccinations.csv')
df2 =
pd.read_csv('country_vaccinations_by_manufacturer.csv')
merged_df = pd.merge(df1, df2, on='date', how='inner')
merged_df.to_csv('merged_dataset.csv', index=False)
print("Merged dataset:")
print(merged_df.head())
```

Output:



The screenshot shows a Jupyter Notebook interface with a dark theme. The left sidebar contains a file explorer with various files and folders, including 'Covid-19', 'aggregated_data.csv', 'cleaned_dataset.csv', 'country_vaccinations_by_manufacturer.csv', 'country_vaccinations.csv', 'merged_dataset.csv', 'normalized_dataset.csv', 'Phase1.ipynb', 'phase1.ipynb', 'Presentation 1.pptx', 'reduced_dataset.csv', 'DataScience_Phase1.pdf', and 'Presentation 1.zip'. The main area displays a Python script for data integration. The script imports pandas, reads two CSV files, merges them on the 'date' column using an inner join, saves the merged dataset to a new CSV file, and prints the first five rows. The output shows the merged dataset with columns: country, iso_code, date, total_vaccinations_x, and people_vaccinated. The first five rows all show data for Afghanistan on 2021-02-22, with zero values for the other columns. Below this, there are three more tables showing vaccination data for different countries and manufacturers, with some values truncated.

```
#DataIntegration
import pandas as pd
df1 = pd.read_csv('country_vaccinations.csv')
df2 = pd.read_csv('country_vaccinations_by_manufacturer.csv')
merged_df = pd.merge(df1, df2, on='date', how='inner')
merged_df.to_csv('merged_dataset.csv', index=False)
print("Merged dataset:")
print(merged_df.head())
```

(1/1) ✓ 446s Python

Merged dataset:

	country	iso_code	date	total_vaccinations_x	people_vaccinated \
0	Afghanistan	AFG	2021-02-22	0.0	0.0
1	Afghanistan	AFG	2021-02-22	0.0	0.0
2	Afghanistan	AFG	2021-02-22	0.0	0.0
3	Afghanistan	AFG	2021-02-22	0.0	0.0
4	Afghanistan	AFG	2021-02-22	0.0	0.0

	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations \
0	NaN	NaN	NaN
1	NaN	NaN	NaN
2	NaN	NaN	NaN
3	NaN	NaN	NaN
4	NaN	NaN	NaN

	total_vaccinations_per_hundred	people_vaccinated_per_hundred \
0	0.0	0.0
1	0.0	0.0
2	0.0	0.0
3	0.0	0.0
4	0.0	0.0

	people_fully_vaccinated_per_hundred	daily_vaccinations_per_million \
0	NaN	NaN
1	NaN	NaN
...		
1	Moderna	2
2	Oxford/AstraZeneca	23510
3	Sinopharm/Beijing	101
4	Sputnik V	894657

Output is truncated. View as a [scrollable element](#) or open in a [text editor](#). Adjust cell output [settings](#)...

2.3 Data Transformation:

Data transformation, a key step in data preprocessing, involves converting and reshaping data to make it suitable for analysis or modeling. This process may include changing data types, scaling, encoding categorical variables, and aggregating data. In a given dataset, data transformation helps ensure data consistency and prepares it for advanced analytics. For example, you can transform textual descriptions into numerical features, normalize numerical values, and aggregate data for summary statistics. Data transformation is essential for extracting meaningful insights and patterns from the data, making it a fundamental aspect of data preprocessing.

Data Transformation includes various subprocess which are listed as below:

- Changing Data Types
- Encoding Categorical Variables
- Normalizing data in dataset
- Aggregating data

2.3.1 Changing Data Types:

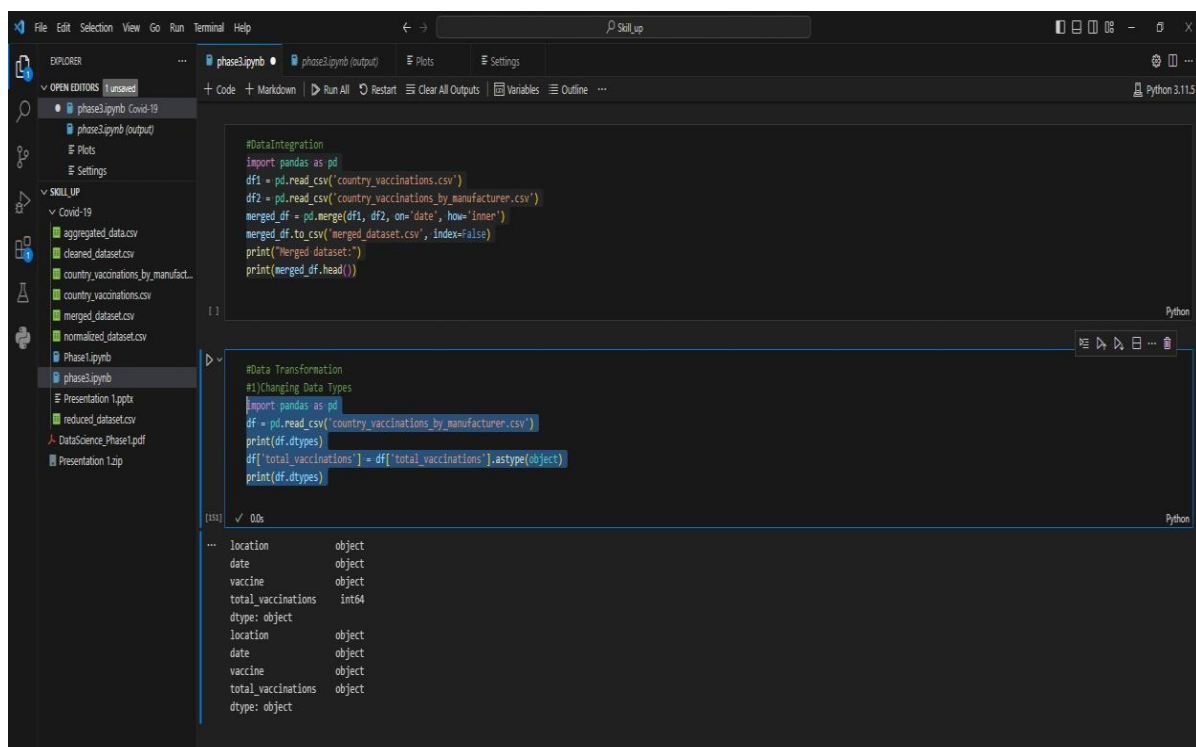
Changing data types, as part of data transformation, involves converting the type of data in a dataset from one format to another. This process is essential when dealing with a dataset that contains data in formats that are not suitable for the analysis or tasks you want to perform. Changing data types ensures that the dataset's content is in the right format for analysis and modeling, preventing issues like data type

incompatibility and enabling more effective data processing. Here in the data set `country_vaccinations_by_manufacturer` we have changed the data type of the column `total_vaccination` into 'object' from 'int64' by making use of the method `astype()` suffixing the column name before it.

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)
```

Output:



```
#Data Integration
import pandas as pd
df1 = pd.read_csv('country_vaccinations.csv')
df2 = pd.read_csv('country_vaccinations_by_manufacturer.csv')
merged_df = pd.merge(df1, df2, on='date', how='inner')
merged_df.to_csv('merged_dataset.csv', index=False)
print("Merged dataset:")
print(merged_df.head())

#Data Transformation
#1) Changing Data Types
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)
```

```
location      object
date          object
vaccine       object
total_vaccinations int64
dtype: object
location      object
date          object
vaccine       object
total_vaccinations object
dtype: object
```

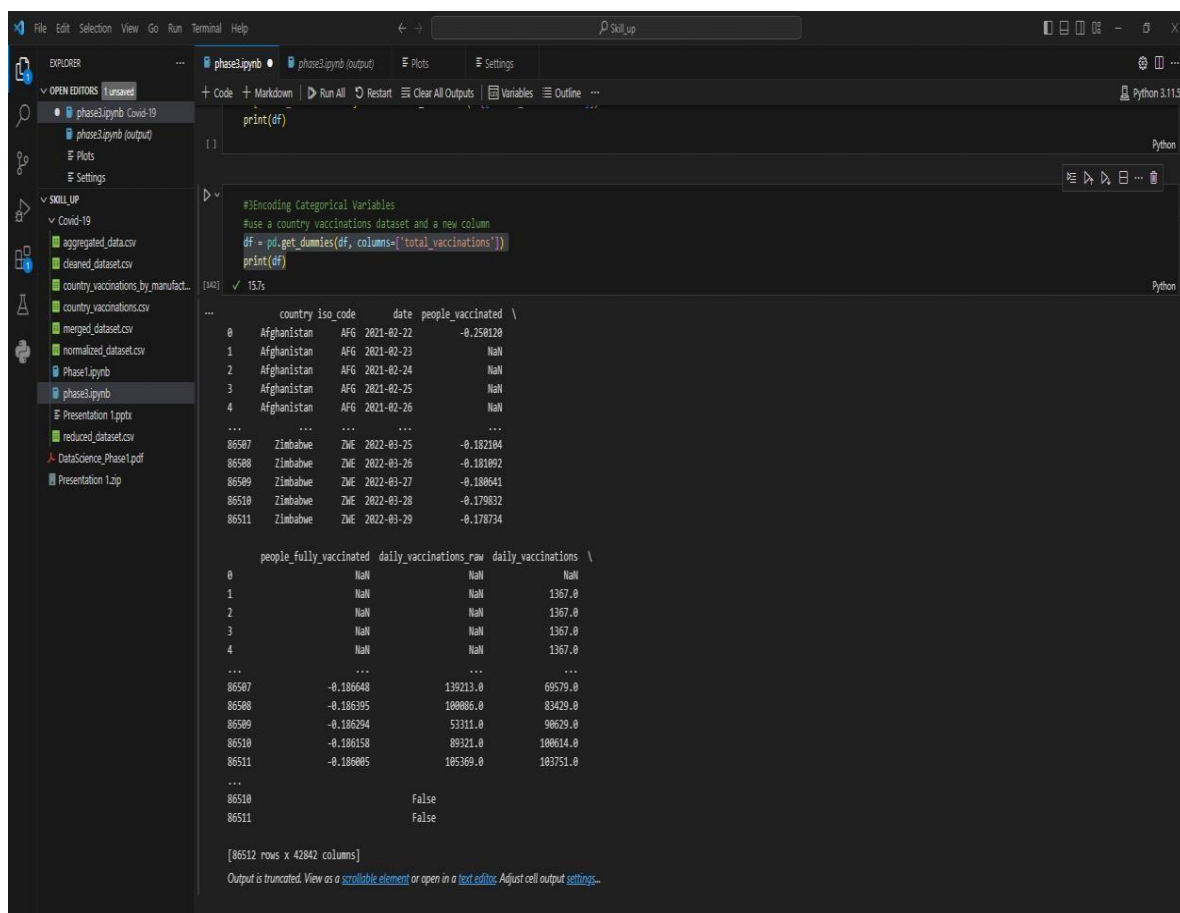
2.3.2 Encoding Categorical Variables:

Encoding categorical variables is a crucial step in data transformation, particularly when working with machine learning models or data analysis. Categorical variables represent categories or labels rather than numeric values. To make these variables compatible with most machine learning algorithms, you need to encode them into numerical form.

Program:

```
df = pd.get_dummies(df, columns=['total_vaccinations'])
print(df)
```

Output:



```
#Encoding Categorical Variables
#use a country vaccinations dataset and a new column
df = pd.get_dummies(df, columns=['total_vaccinations'])
print(df)
```

```
country iso_code date people_vaccinated \
0 Afghanistan AFG 2021-02-22 -0.250120
1 Afghanistan AFG 2021-02-23 NaN
2 Afghanistan AFG 2021-02-24 NaN
3 Afghanistan AFG 2021-02-25 NaN
4 Afghanistan AFG 2021-02-26 NaN
...
86507 Zimbabwe ZWE 2022-03-25 -0.182104
86508 Zimbabwe ZWE 2022-03-26 -0.181092
86509 Zimbabwe ZWE 2022-03-27 -0.180641
86510 Zimbabwe ZWE 2022-03-28 -0.179032
86511 Zimbabwe ZWE 2022-03-29 -0.178734

people_fully_vaccinated daily_vaccinations_raw daily_vaccinations \
0 NaN NaN NaN
1 NaN NaN 1367.0
2 NaN NaN 1367.0
3 NaN NaN 1367.0
4 NaN NaN 1367.0
...
86507 -0.186648 139213.0 69579.0
86508 -0.186395 100086.0 83429.0
86509 -0.186294 53311.0 90629.0
86510 -0.186158 89321.0 100614.0
86511 -0.186005 105369.0 103751.0
...
86510 False
86511 False

[86512 rows x 42842 columns]
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

2.3.3 Normalizing data in dataset:

Normalizing data is an essential data transformation technique used to scale numerical features within a dataset to a common range. The goal is to make different features or variables comparable, eliminate the influence of differing units or scales, and ensure that no single feature dominates the analysis.

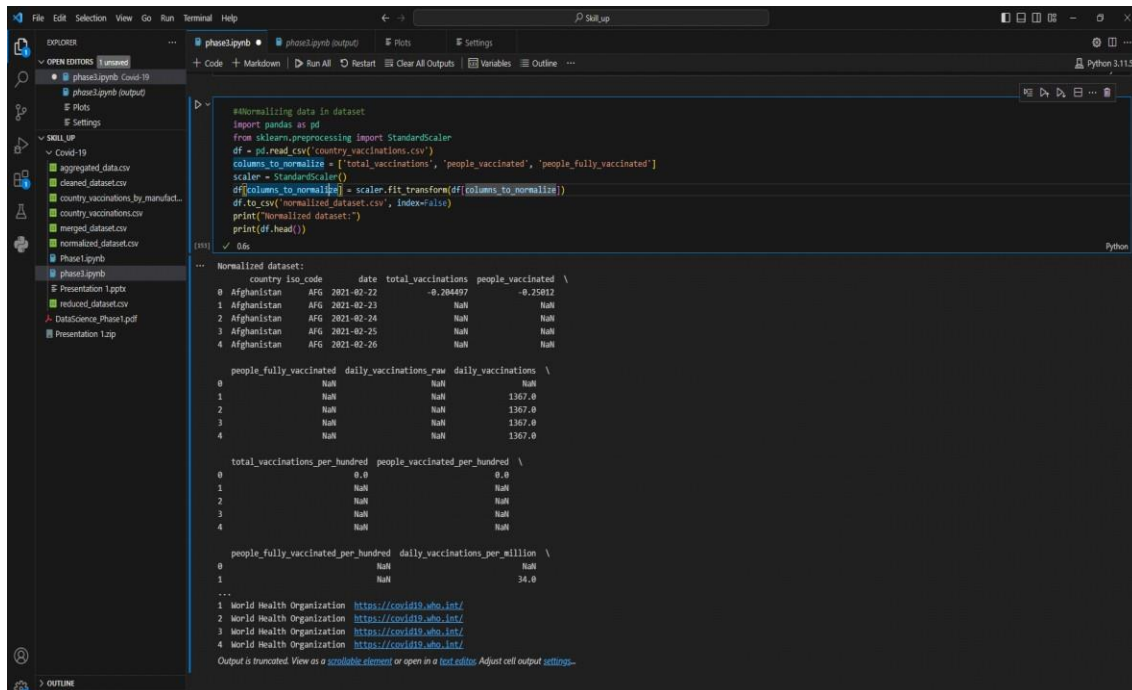
Normalization typically scales the data to a range between 0 and 1, but it can also involve other scales depending on the specific needs of the analysis.

We have normalized the dataset country and vaccinations, and its source code and output are given as follows:

Program:

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
df = pd.read_csv('country_vaccinations.csv')
columns_to_normalize=['total_vaccinations','people_vaccina
ted','people_fully_vaccinated']
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())
```


Output:



```
#normalizing data in dataset
import pandas as pd
from sklearn.preprocessing import StandardScaler
df = pd.read_csv('country_vaccinations.csv')
columns_to_normalize = ['total_vaccinations', 'people_vaccinated', 'people_fully_vaccinated']
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())
```

Normalized dataset:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated
0	Afghanistan	AFG	2021-02-22	0.284497	0.25812	0.25812
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN

2.3.4 Aggregating data:

Aggregating data in a dataset, as part of data transformation, involves the process of summarizing, grouping, or reducing data to obtain a more concise and understandable representation of the information. It is especially useful when dealing with large datasets or when you want to analyze data at a higher level of granularity. Aggregation often involves applying functions like sum, count, mean, median, or other statistical functions to groups of data points based on one or more grouping criteria.

Program:

```
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:

The screenshot shows a Jupyter Notebook with two cells. The first cell contains code for normalizing data using a scaler. The second cell contains code for aggregating data by date and saving it to a CSV file. The output of the second cell is displayed below the code.

```

# Aggregating data
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)

```

```

Aggregated data:
   date  total_vaccinations
0  2020-12-04      1.000000e+00
1  2020-12-07      1.000000e+00
2  2020-12-09      2.000000e+00
3  2020-12-15      3.000000e+00
4  2020-12-16      4.000000e+00
...
468 2022-03-26      3.583301e+07
469 2022-03-27      3.778547e+07
470 2022-03-28      3.457807e+07
471 2022-03-29      4.450356e+07
472 2022-03-30      1.670492e+07
[473 rows x 2 columns]

```

2.4 Data Reduction:

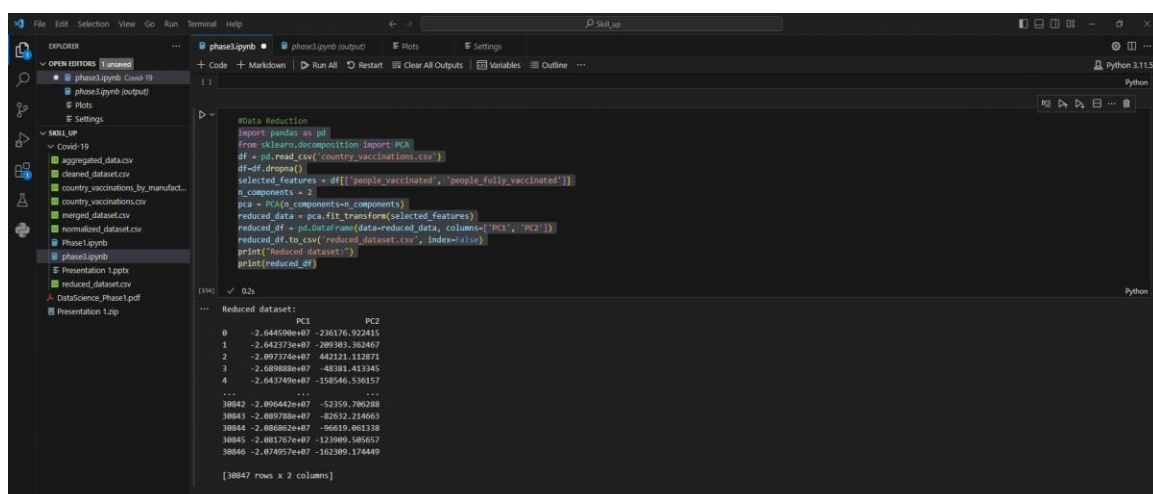
Data reduction, in the context of data preprocessing, refers to the process of reducing the volume but producing the same or similar analytical results. It involves techniques and methods to simplify and condense a dataset while retaining the essential information. Data reduction is often necessary for various reasons, including improving the efficiency of data processing, reducing storage requirements,

and mitigating the curse of dimensionality in machine learning. Data Reduction simply reduces the data by reducing their size.

Program:

```
import pandas as pd
from sklearn.decomposition import PCA
df = pd.read_csv('country_vaccinations.csv')
df=df.dropna()
selected_features = df[['people_vaccinated',
'people_fully_vaccinated']]
n_components = 2
pca = PCA(n_components=n_components)
reduced_data = pca.fit_transform(selected_features)
reduced_df = pd.DataFrame(data=reduced_data,
columns=['PC1', 'PC2'])
reduced_df.to_csv('reduced_dataset.csv', index=False)
print("Reduced dataset:")
print(reduced_df)
```

Output:



```
#Data Reduction
import pandas as pd
from sklearn.decomposition import PCA
df = pd.read_csv('country_vaccinations.csv')
df=df.dropna()
selected_features = df[['people_vaccinated', 'people_fully_vaccinated']]
n_components = 2
pca = PCA(n_components=n_components)
reduced_data = pca.fit_transform(selected_features)
reduced_df = pd.DataFrame(data=reduced_data, columns=['PC1', 'PC2'])
reduced_df.to_csv('reduced_dataset.csv', index=False)
print("Reduced dataset:")
print(reduced_df)
```

```
[104] ✓ 0.2s

... Reduced dataset:
   PC1      PC2
0 -2.64508e+07 -236176.922415
1 -2.642372e+07 -289383.362467
2 -2.097374e+07  462121.112871
3 -2.689888e+07 -48381.413345
4 -2.643749e+07 -158546.536157
...
38842 -2.096442e+07 -52359.786288
38843 -2.089788e+07 -82632.214663
38844 -2.086802e+07 -86219.861138
38845 -2.081267e+07 -123969.346567
38846 -2.074957e+07 -162389.174449

[38847 rows x 2 columns]
```

Visualizing The preprocessed datasets:

Visualizing preprocessed datasets is an essential step in exploratory data analysis (EDA) and data-driven decision-making.

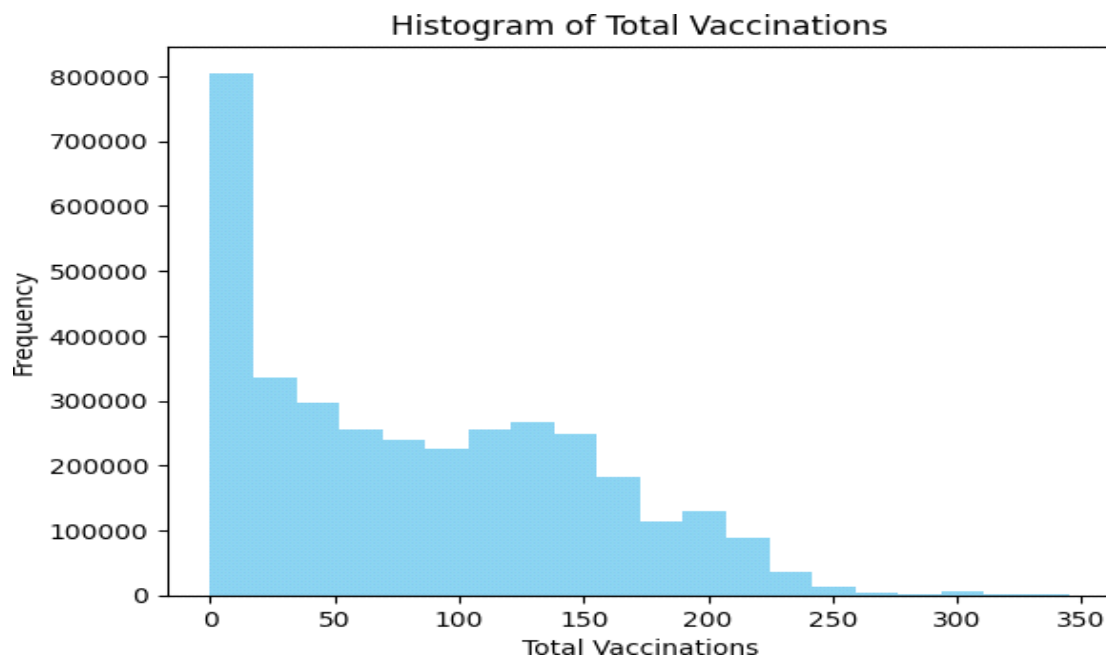
Once you've cleaned and transformed your data, creating visualizations can help you understand the data, identify patterns, and communicate insights effectively.

- So as a very first part of visualization we have visualized the merged dataset of both `country_vaccinations` and `country_vaccinations_by_manufacturer` by means of histogram.

Program:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('merged_dataset.csv')
plt.hist(df['total_vaccinations_per_hundred'], bins=20,
color='skyblue')
plt.title('Histogram of Total Vaccinations')
plt.xlabel('Total Vaccinations')
plt.ylabel('Frequency')
plt.show()
```

Output:

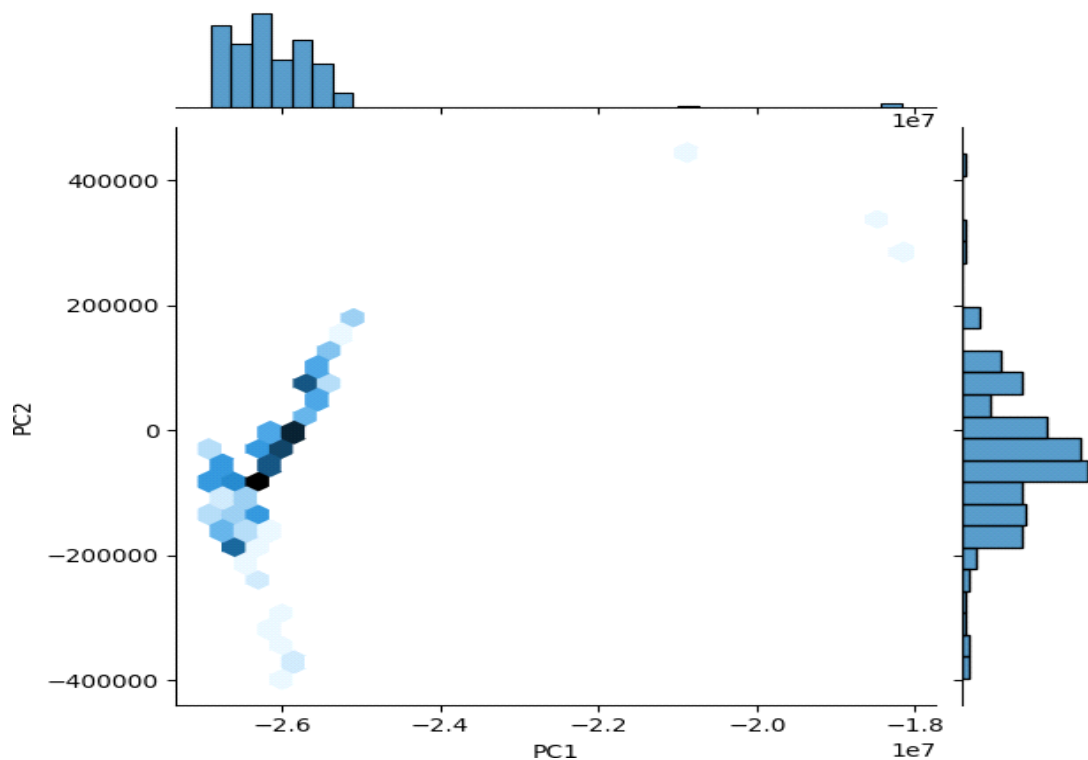


- Following this is the Joint plot representation of the reduced dataset that we have created during data reduction.
- We have made use of the reduced dataset that we have derived from data reduction
- By making use of this dataset we have vizualised it in joint plot.

Program:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('reduced_dataset.csv')
df=df.head(200)
print(sns.jointplot(df,x='PC1',y='PC2',kind='hex'))
```

Output:

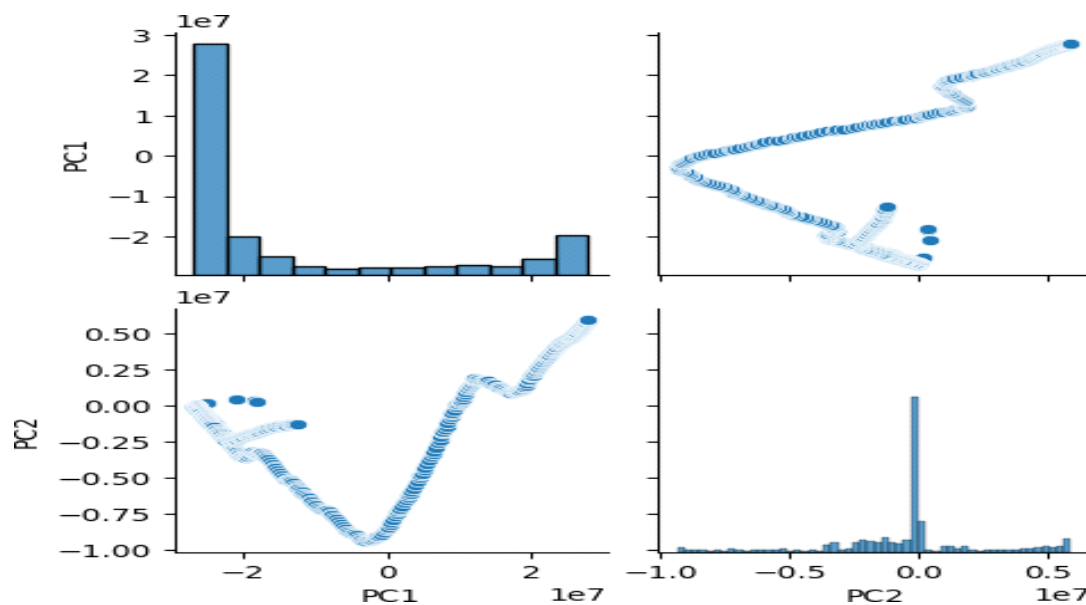


- Thirdly we have made use of the pair plot visualizing technique to visualize the reduced data sets.

Program:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('reduced_dataset.csv')
df=df.head(1000)
plt.figure(figsize=(12,8))
sns.pairplot(df)
```

Output:

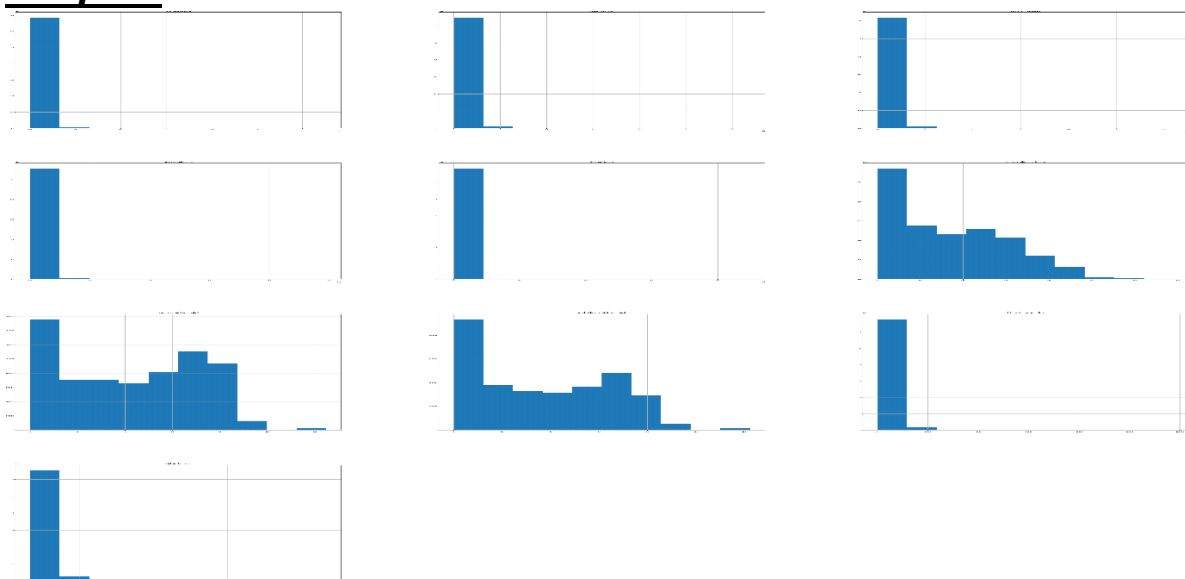


- Following these pair plots we have made use of the merged dataset to visualize it in histogram as it contains a combination of different columns and rows.

Program:

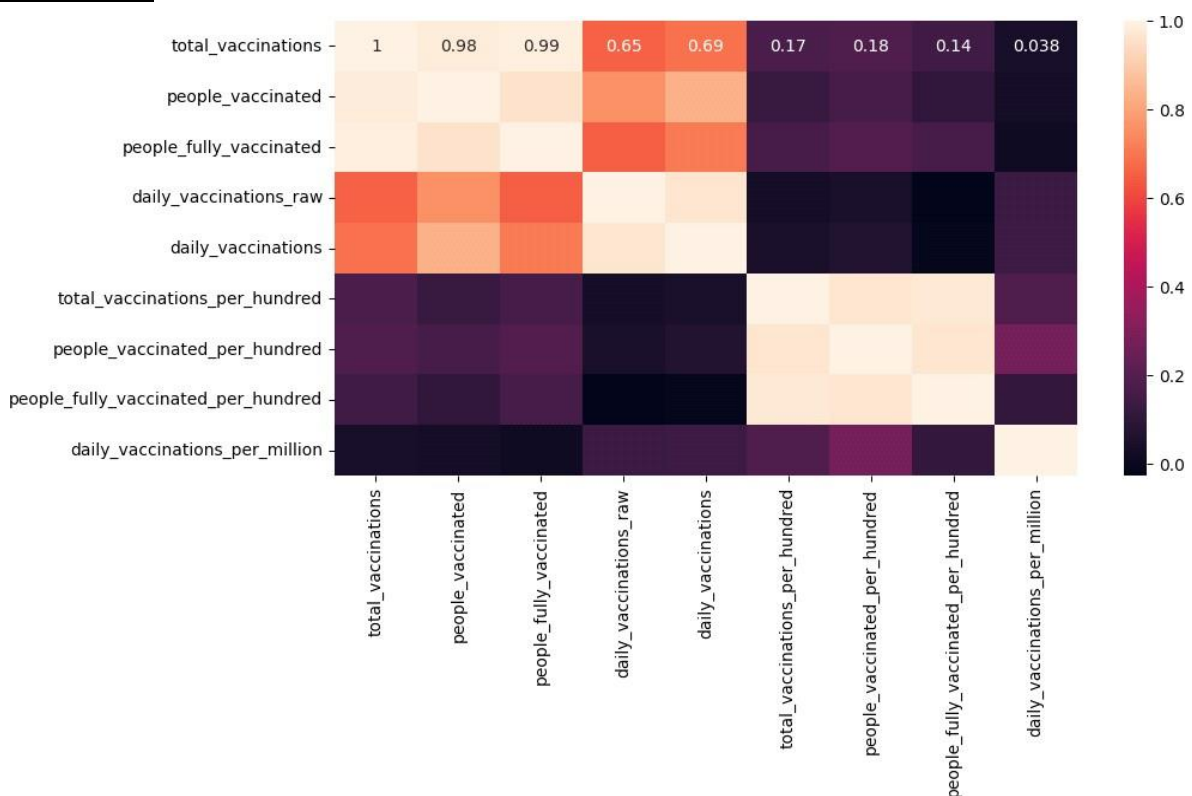
```
import numpy as np
import pandas as pd
df=pd.read_csv('merged_dataset.csv')
print(df.hist(figsize=(100,80)))
```

Output:



- Being last we have calculated the correlations for the `country_vaccinations` dataset and by making use of it we even have plotted a heatmap for it which is visualized below.

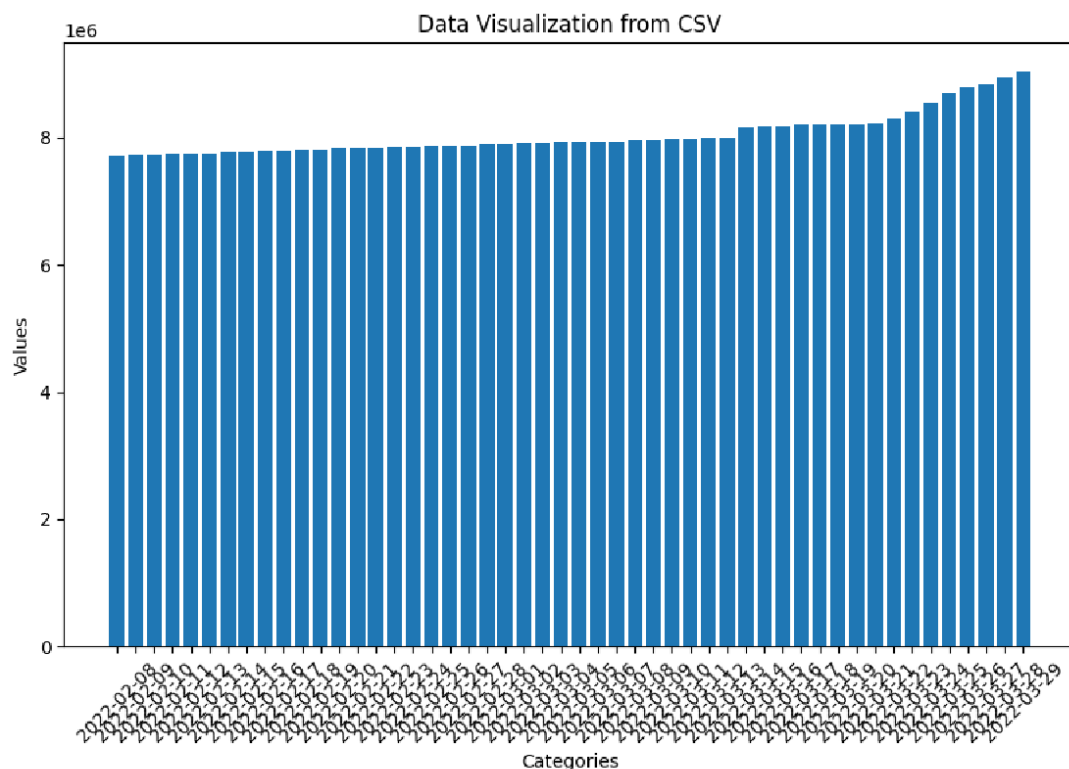
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
df=pd.read_csv('country_vaccinations.csv')
df.corr(numeric_only=True)
print(df)
plt.figure(figsize=(10,5))
print(sns.heatmap(df.corr(numeric_only=True),annot=True))
```



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:

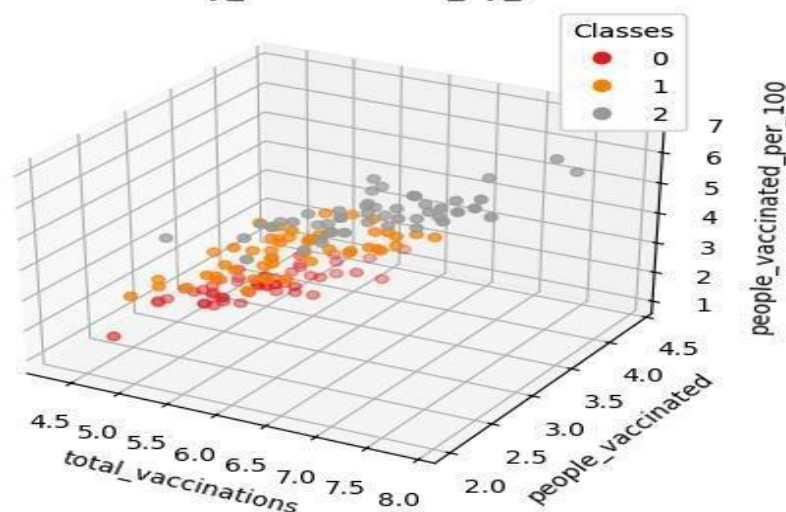


Program2:

```
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:

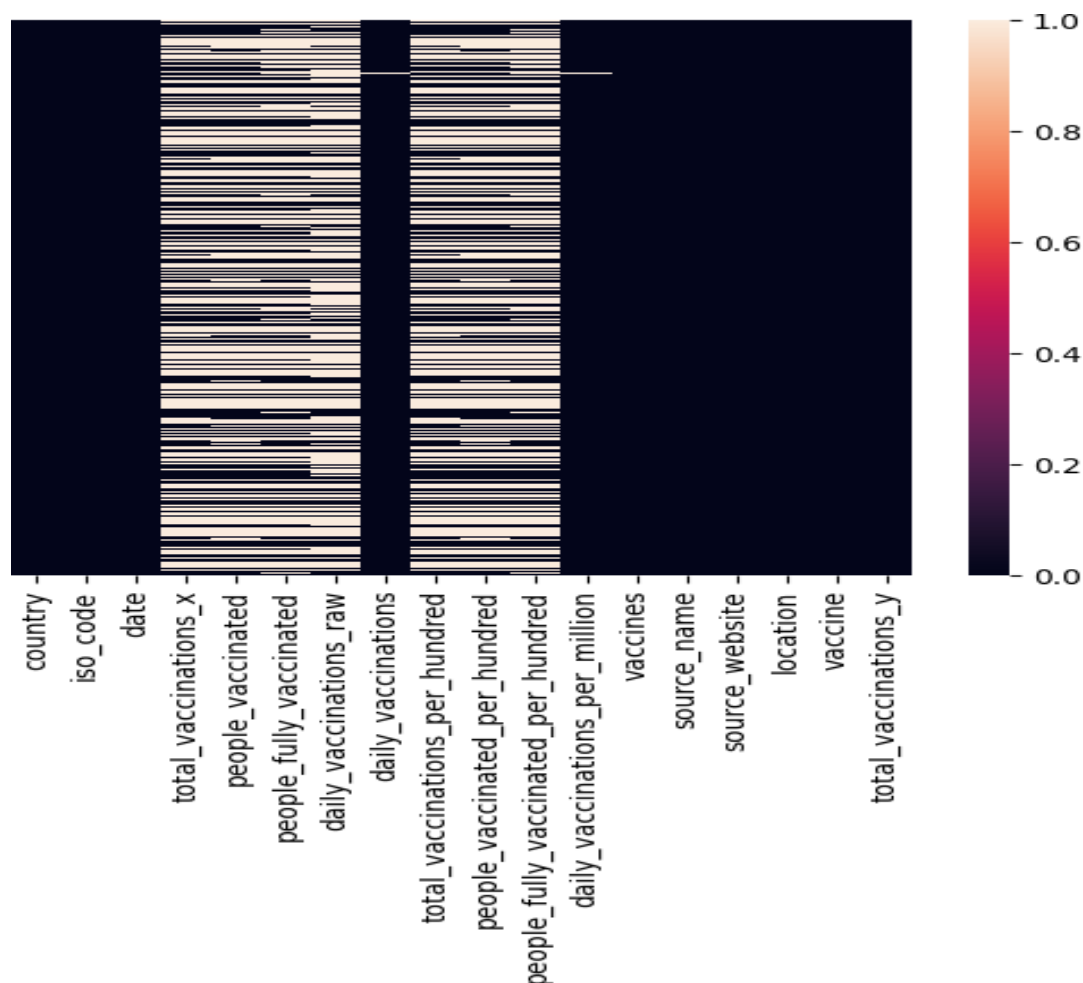
3D Scatter Plot of Country_vaccinations_by_manufacturerDataset



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_escape")
missing_values=["N/a","na",np.nan]
df=pd.read_csv("merged_dataset.csv",na_values=missing_values,encoding="unicode_escape")
print(sns.heatmap(df.isnull(),yticklabels=False))
```

Output:



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:

Univariate 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:

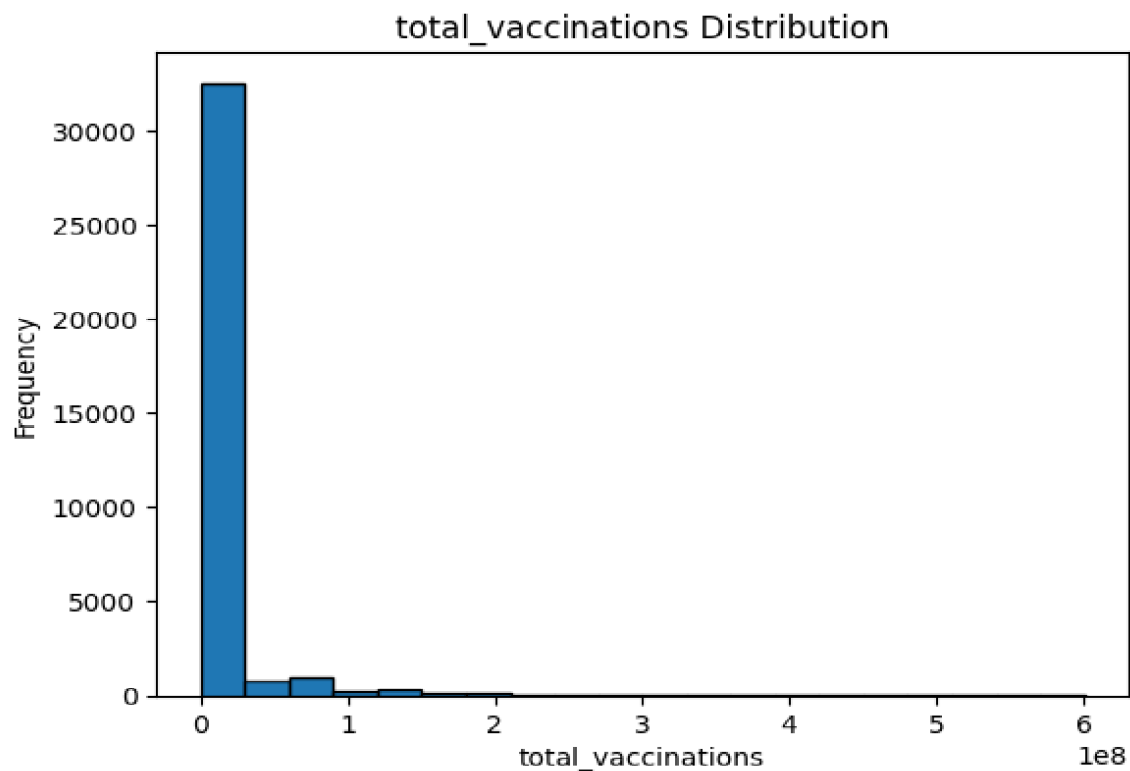
```
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv("country_vaccinations_by_manufacturer.csv")
column_name = "total_vaccinations"

summary = df[column_name].describe()
print(f"Summary statistics for
{column_name}:\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.

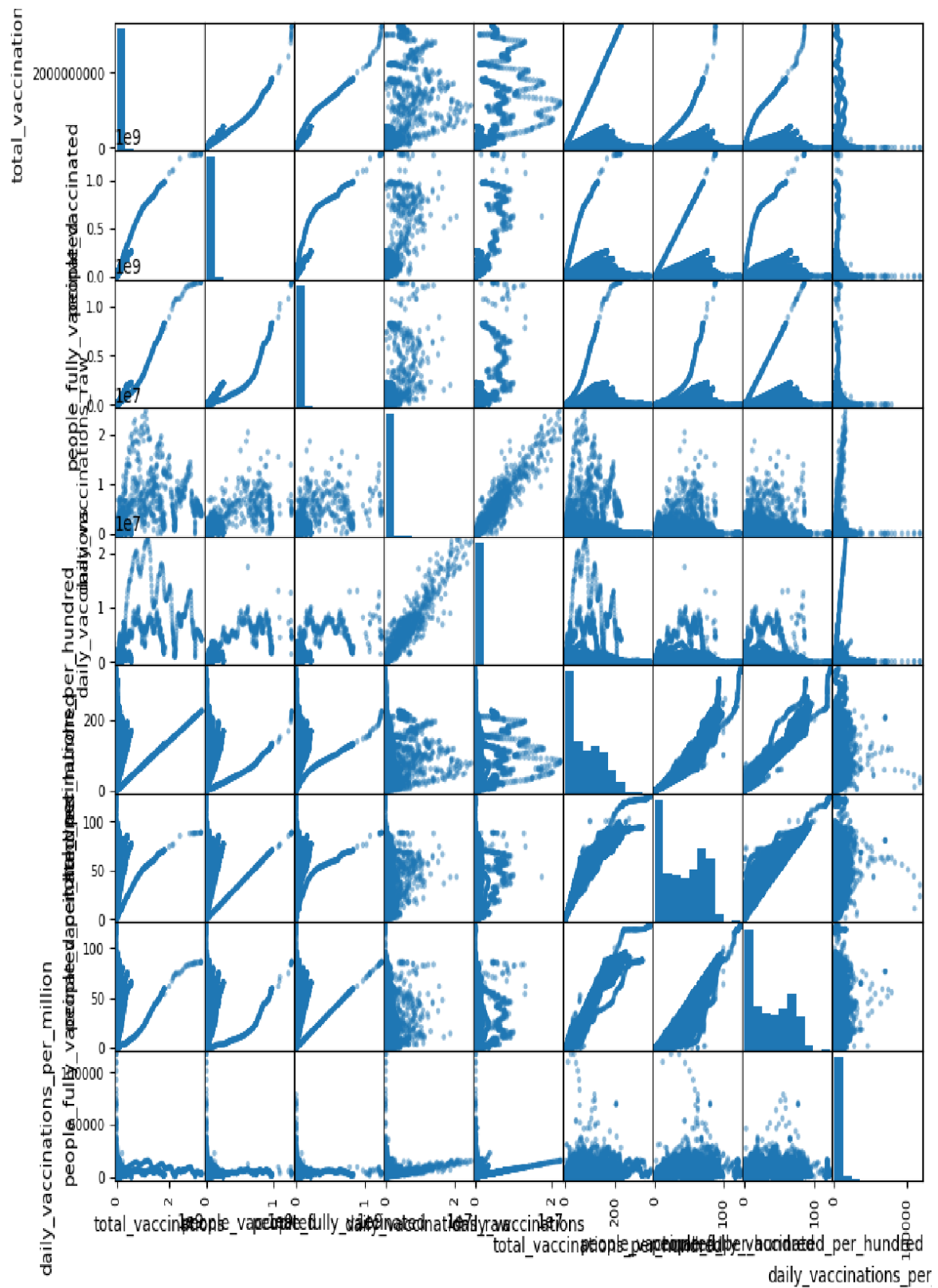
Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

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="unicode_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:



Statistical Analysis:

Statistical analysis is a systematic approach to understanding data through the application of mathematical and statistical techniques. It plays a crucial role in making sense of complex information, identifying patterns, and drawing meaningful insights. The process typically begins with data collection, followed by data cleaning and pre processing 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:

The screenshot shows a Jupyter Notebook in VS Code with the following output:

```

(41) ✓ 94s
... Descriptive Statistics:

```

	total_vaccinations_x	people_vaccinated	people_fully_vaccinated \
count	9.000000e+00	9.000000e+00	3.0
mean	6.834797e+05	6.667750e+05	14871.0
std	1.825217e+06	1.008100e+06	0.0
min	0.000000e+00	0.000000e+00	14871.0
25%	1.000000e+00	1.000000e+00	14871.0
50%	3.000000e+00	3.000000e+00	14871.0
75%	2.859435e+06	2.000321e+06	14871.0
max	2.859435e+06	2.000321e+06	14871.0

	daily_vaccinations_raw	daily_vaccinations \
count	6.000000	9.000000
mean	223844.166667	93609.444444
std	245280.633750	140413.916667
min	0.000000	0.000000
25%	0.750000	0.000000
50%	223844.500000	0.000000
75%	447688.000000	238828.000000
max	447688.000000	238828.000000

	total_vaccinations_per_hundred	people_vaccinated_per_hundred \
count	9.000000	9.0
mean	0.206667	0.2
std	0.310000	0.3
min	0.000000	0.0
25%	0.000000	0.0
50%	0.000000	0.0
75%	0.620000	0.6
max	0.620000	0.6

	people_fully_vaccinated_per_hundred	daily_vaccinations_per_million \
count	3.0	9.0
mean	0.0	282.0
std	0.0	423.0
min	0.0	0.0
25%	0.0	0.0
50%	0.0	0.0
75%	0.0	846.0
max	0.0	846.0

	total_vaccinations_y
count	18.000000
mean	3.200000
std	5.573748

Advantages:

The analysis of COVID-19 vaccines and their effects is crucial for public health and has several significant advantages. Here are some of the key advantages of COVID-19 vaccine analysis:

- **Efficacy Assessment:** Vaccine analysis helps determine the effectiveness of vaccines in preventing COVID-19 and its variants. This information is critical for public health officials and policymakers in making informed decisions about vaccination campaigns.
- **Safety Evaluation:** Continual analysis monitors the safety of COVID-19 vaccines, identifying any rare or unexpected adverse events. This helps in maintaining public confidence in vaccination programs and ensuring that the vaccines are as safe as possible.
- **Optimizing Vaccine Distribution:** Analysis of vaccine data can help governments and healthcare systems allocate vaccines more effectively. By understanding which populations are more or less responsive to specific vaccines, resources can be distributed where they are needed most.
- **Monitoring Vaccine Variants:** As new variants of the virus emerge, ongoing analysis can assess whether current vaccines remain effective against them. This information guides vaccine development and booster shot strategies.
- **Long-Term Immunity Assessment:** Vaccine analysis provides insights into the duration of vaccine-induced immunity. This information is valuable for determining when and if booster shots are needed.

Disadvantages:

While analyzing COVID-19 vaccines is essential for understanding their safety and effectiveness, there can be some disadvantages or challenges associated with this process. It's important to be aware of these potential issues:

- **Time-Consuming:** Rigorous analysis of vaccine data can be time-consuming, especially for large-scale clinical trials and real-world studies. Delays in data analysis may slow down the decision-making process during a pandemic.
- **Resource-Intensive:** Conducting thorough vaccine analysis requires significant resources, including funding, personnel, and laboratory facilities. This can strain healthcare systems and research organizations.
- **Data Availability and Quality:** Vaccine analysis depends on the availability and quality of data. In some regions, data collection and reporting may be inconsistent, leading to incomplete or unreliable information.
- **Privacy and Ethics:** Data used for vaccine analysis often contain personal and sensitive information. Ensuring data privacy and adhering to ethical standards can be challenging, especially in real-world data analysis.
- **Selection Bias:** In real-world studies, there can be selection bias, as individuals who choose to get vaccinated might differ from those who don't. This can affect the accuracy of vaccine effectiveness estimates.
- **Limited Data on Specific Groups:** Vaccine data might be limited for certain population groups, such as pregnant

women, children, or individuals with specific medical conditions. This can limit the generalizability of findings.

Conclusion:

In the intricate puzzle of the COVID-19 pandemic, analysis serves as the guiding light, illuminating the path to understanding, responding, and recovering. Through the lens of data and research, we inch closer to a world where the virus is but a memory. With each insight gained, we strengthen our resolve, adapt our strategies, and pave the way to a healthier, safer future. In the relentless pursuit of knowledge, we find hope, resilience, and the collective strength to overcome the challenges that COVID-19 has posed.