

COVID-19 VACCINE ANALYSIS

Phase 3: Development Part 1

Topic: Start building the Covid-19 Vaccine analysis project by loading and pre-processing the dataset.



Introduction:

- 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.
- Many received Emergency Use Authorization (EUA) initially, allowing for their rapid deployment, later followed by full regulatory approval. Furthermore, the ongoing evolution of the virus necessitates continuous research, development, and adaptation.
- Developing a Covid-19 Vaccine analysis project is a data-driven process that involves harnessing the power of machine learning to analyze the data from the collected data it involves exact analysis about the present, the past and the future. This Journey begins with the fundamental steps of data loading and preprocessing the given dataset.
- This Introduction will guide you through the initial steps of the process. We'll explore how to import essential libraries, load the Covid-19 dataset, and perform critical preprocessing steps. Data preprocessing is crucial as it helps clean, format, and prepare the data for further analysis. This Includes handling missing values, encoding categorical variables, and clustering that the data is appropriately scaled.

Dataset:

The dataset for the given Covid-19 Project can be downloaded from the link given below.

“ <https://www.kaggle.com/datasets/gpreda/covid-world-vaccination-progress> ”

The given dataset is picturized as below:

country	iso_code	date	total_vacc	people_va	people_ful	daily_vacc	source_na	source_website
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/

Loading and Preprocessing of dataset:

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.

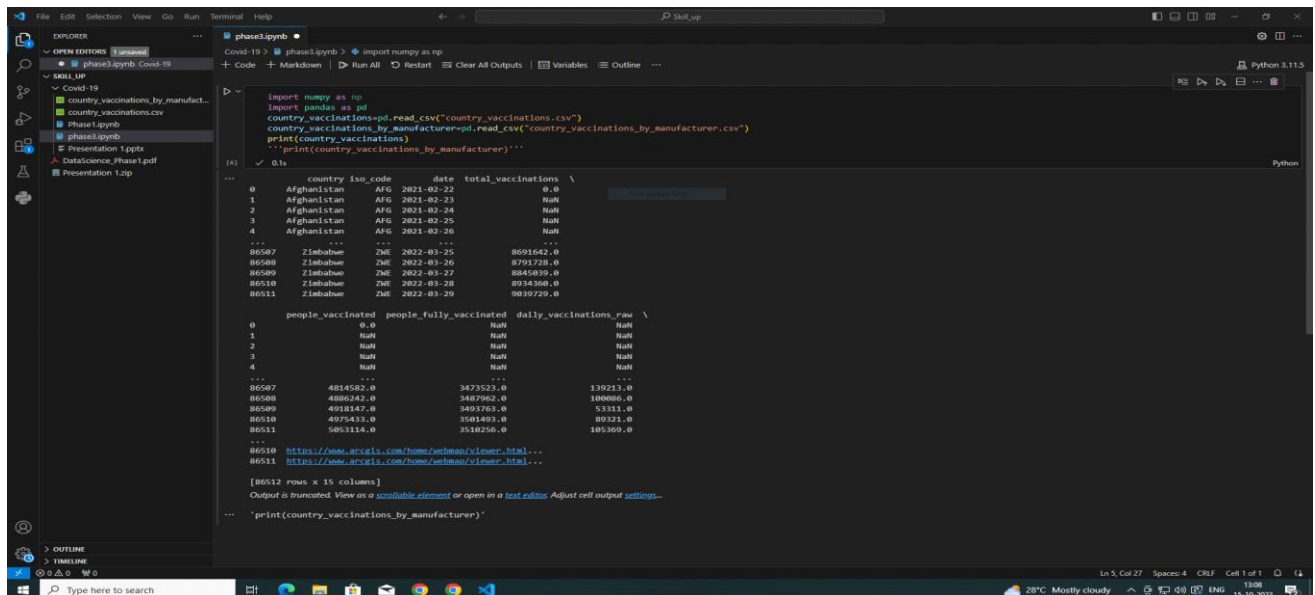
1.Dataset named country_vaccinations:

```
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:

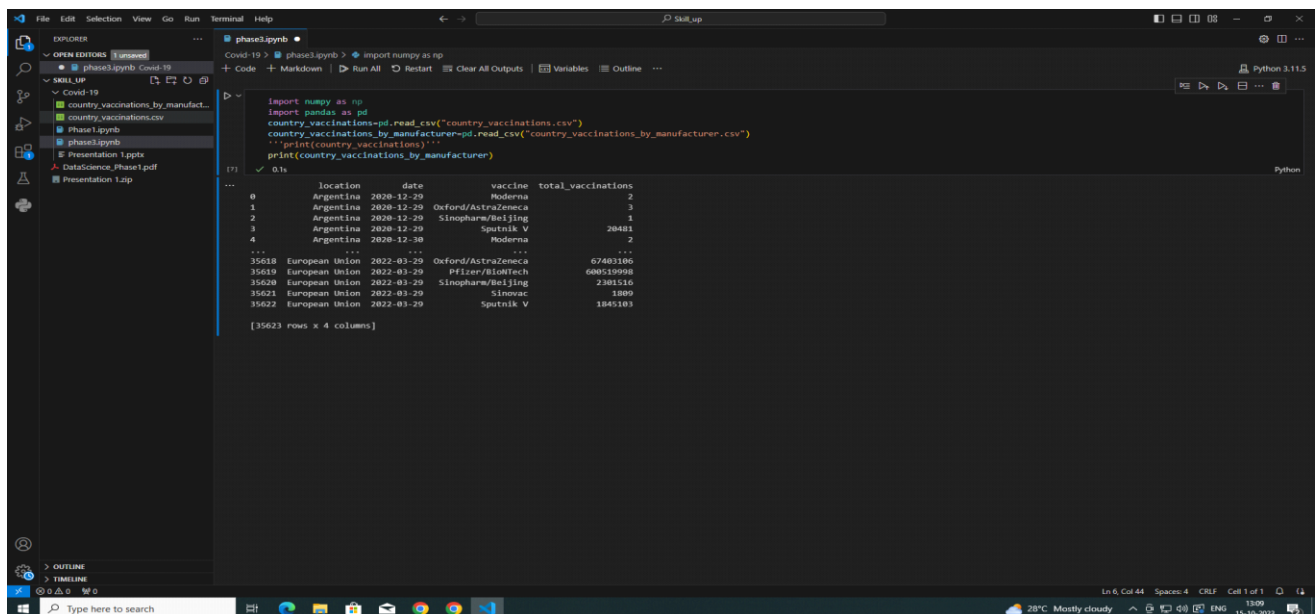
```
import numpy as np
import pandas as pd
country_vaccinations_by_manufacturer=pd.read_csv("country_vaccinations_by_manufacturer.csv")
print(country_vaccinations_by_manufacturer)
```

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
...
86507 Zimbabwe ZWE 2022-03-25 8691642.0
86508 Zimbabwe ZWE 2022-03-26 8791728.0
86509 Zimbabwe ZWE 2022-03-27 8645839.0
86510 Zimbabwe ZWE 2022-03-28 8934368.0
86511 Zimbabwe ZWE 2022-03-29 9039729.0
...
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
...
86507 4814582.0 3473523.0 139213.0
86508 4886242.0 3487962.0 100086.0
86509 4918147.0 3493763.0 53111.0
86510 4975433.0 3501463.0 89323.0
86511 5053114.0 3518256.0 105369.0
...
[86512 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 3
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 2381516
35621 European Union 2022-03-29 Sinovac 11809
35622 European Union 2022-03-29 Sputnik V 1845103
...
[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.

```
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:

```

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.head())

```

Output:

```

Missing values in the dataset:
country          0
iso_code         0
date            0
total_vaccinations    42985
people_vaccinated    45218
people_fully_vaccinated 47710
daily_vaccinations_raw 51150
daily_vaccinations    299
total_vaccinations_per_hundred    42985
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
Cleaned dataset:
   country iso_code  date  total_vaccinations  people_vaccinated \
94  Afghanistan  AFG  2021-05-27          503113.0          479574.0
101 Afghanistan  AFG  2021-06-03          630305.0          481800.0
339 Afghanistan  AFG  2022-01-27          5081064.0          4517380.0
433  Albania    ALB  2021-02-18           3849.0           2438.0
515  Albania    ALB  2021-05-11          622507.0          440921.0
...
101  https://covid19.who.int/
339  https://covid19.who.int/
433  https://shendetesia.gov.al/vaksinimi-anticovid...
515  https://shendetesia.gov.al/vaksinimi-anticovid...

```

2) Data cleansing for the dataset `country_vaccinations_by_manufacturer` 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 environment with a file explorer on the left and a code editor on the right. The file explorer lists several CSV files related to COVID-19 data, including 'aggregated_data.csv', 'cleaned_dataset.csv', 'country_vaccinations_by_manufacturer.csv', 'country_vaccinations.csv', 'merged_dataset.csv', 'normalized_dataset.csv', 'Phase1.ipynb', 'Presentation 1.pptx', 'reduced_dataset.csv', 'DataScience_Phase1.pdf', and 'Presentation 1.zip'. The code editor displays the following Python code:

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

Below the code, the output of the execution is shown, indicating that the data is clean and providing a preview of the dataset:

```
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  67403106
35619 European Union 2022-03-29  Pfizer/BioNTech    606519998
35620 European Union 2022-03-29  Sinopharm/Beijing  2381516
35621 European Union 2022-03-29  Sinovac           1809
35622 European Union 2022-03-29  Sputnik V        1845183
```

The output also indicates that the dataset has 35623 rows and 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

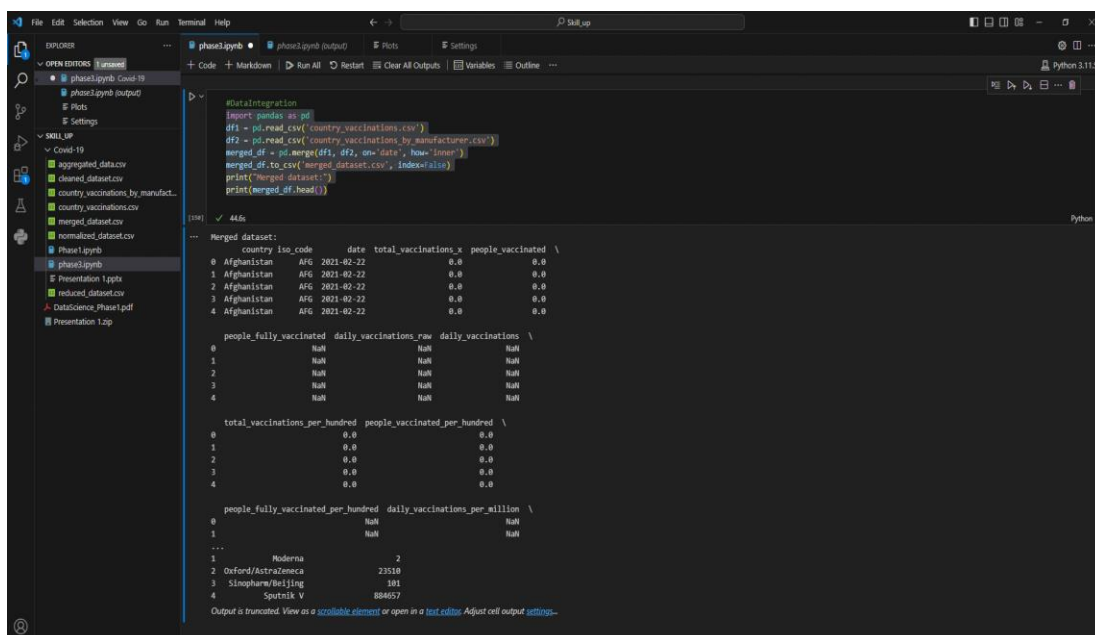
separately and stored as csv files.

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:



```
#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())
```

```
1591 ✓ 44s
--- 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_row  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              884057
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.

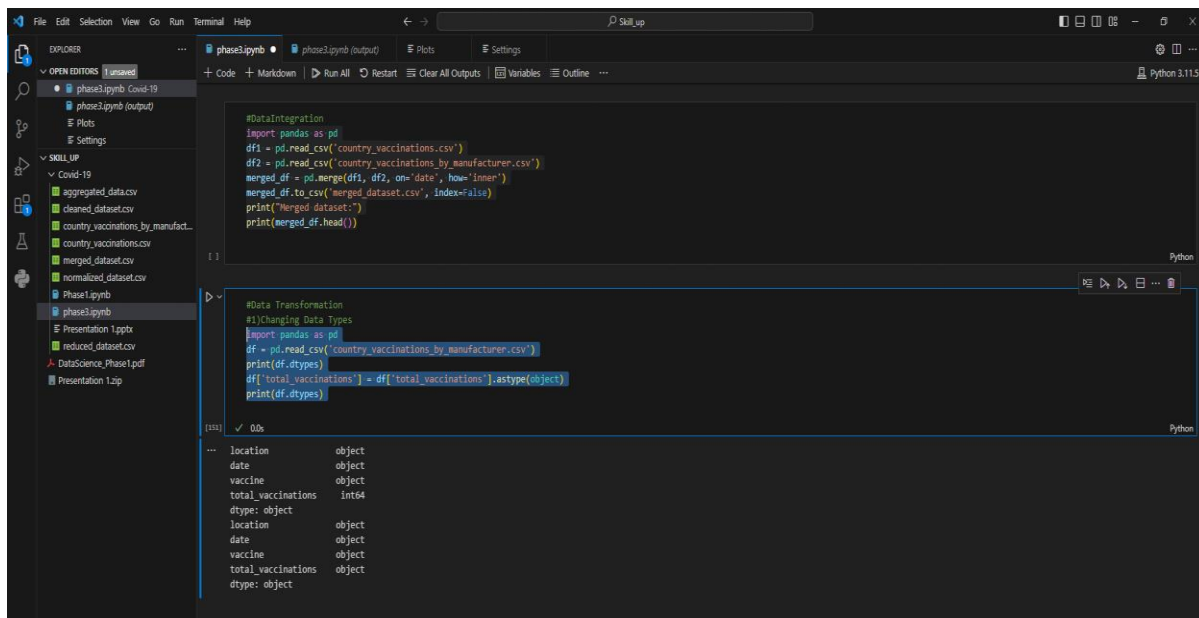
Here in the data set `country_vaccinations_by_manufacturer` we have changed the data type of the column `total_vaccination` into 'object' from 'int64' by making use of the method `astype()` suffixing the column name

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:



```
#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())

#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:

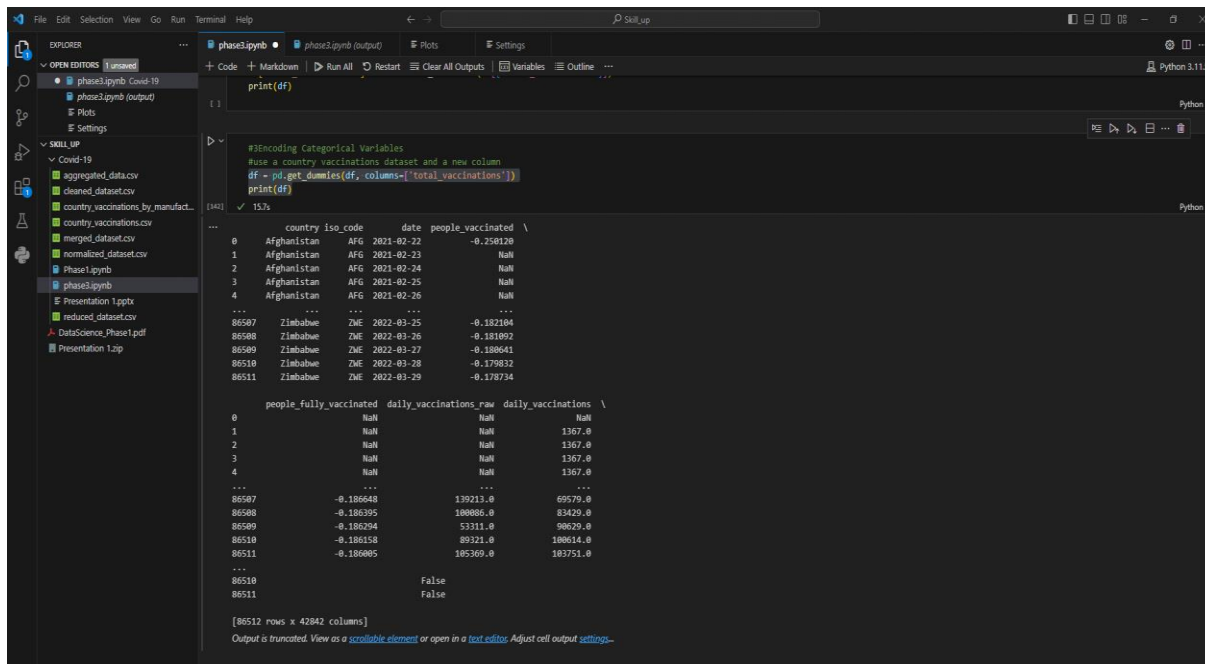
Encoding categorical variables is a crucial step in data transformation, particularly when working with machine learning models or data analysis. Categorical variables represent categories or labels rather than a numeric values. To make these variables compatible with most machine learning algorithms,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

numerical form.

Program:

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

Output:



```
142: ✓ 15.7s  
...  
country_iso_code  date  people_vaccinated \  
0      Afghanistan  AFG  2021-02-22      -0.258120  
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      -0.182184  
86508      Zimbabwe  ZWE  2022-03-26      -0.181992  
86509      Zimbabwe  ZWE  2022-03-27      -0.180641  
86510      Zimbabwe  ZWE  2022-03-28      -0.179832  
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      -0.186448      139213.0      69579.0  
86508      -0.186395      100886.0      83429.0  
86509      -0.186294      53311.0      90629.0  
86510      -0.186158      89321.0      100614.0  
86511      -0.186085      105369.0      183751.0  
...  
...  
86510      False  
86511      False  
[86512 rows x 6 columns]  
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

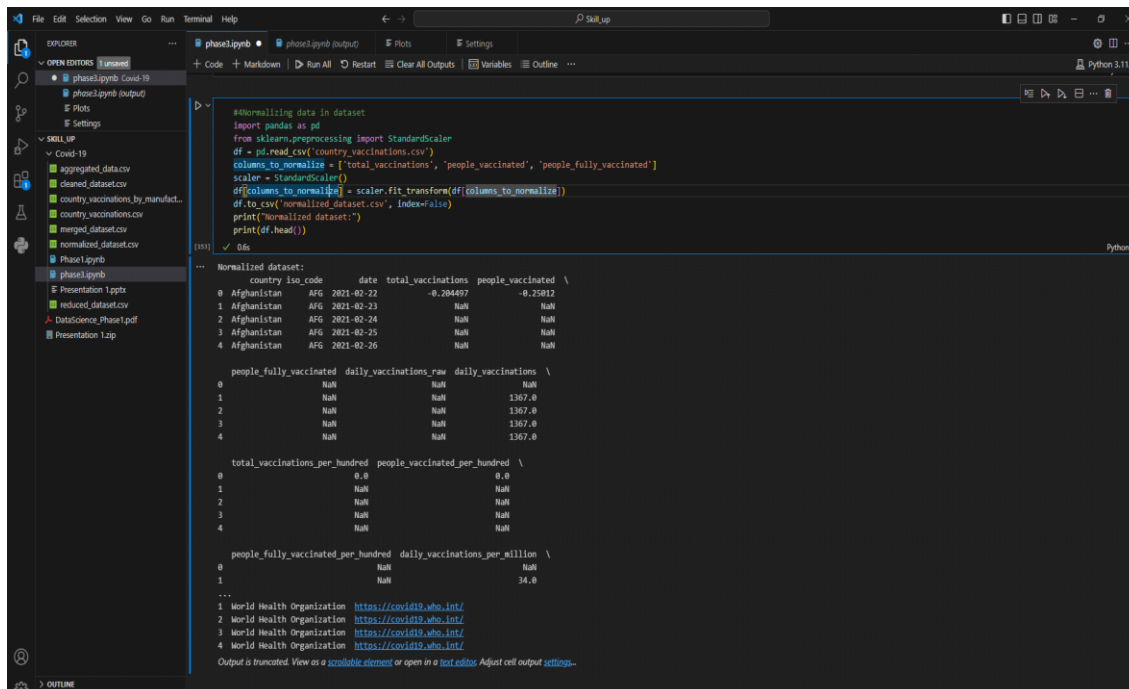
2.3.3 Normalizing data in dataset:

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_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())
```

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
Afghanistan	AFG	2021-02-22	-0.204497	-0.25012	NaN
Afghanistan	AFG	2021-02-23	NaN	NaN	NaN
Afghanistan	AFG	2021-02-24	NaN	NaN	NaN
Afghanistan	AFG	2021-02-25	NaN	NaN	NaN
Afghanistan	AFG	2021-02-26	NaN	NaN	NaN

people_fully_vaccinated daily_vaccinations_row daily_vaccinations

	people_fully_vaccinated	daily_vaccinations_row	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

total_vaccinations_per_hundred people_vaccinated_per_hundred

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

people_fully_vaccinated_per_hundred daily_vaccinations_per_million

	people_fully_vaccinated_per_hundred	daily_vaccinations_per_million
0	NaN	NaN
1	NaN	34.0

World Health Organization <https://covid19.who.int/>

World Health Organization <https://covid19.who.int/>

World Health Organization <https://covid19.who.int/>

World Health Organization <https://covid19.who.int/>

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings.

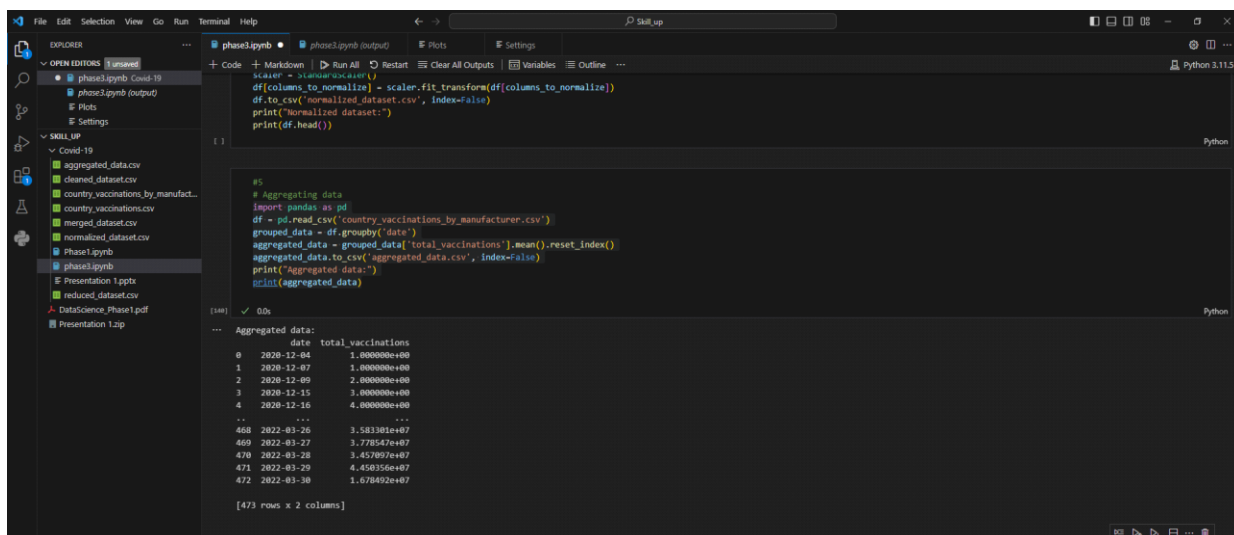
2.3.4 Aggregating data:

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 interface with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'Covid-19' containing several CSV files, including 'country_vaccinations_by_manufacturer.csv'. The code editor displays the following Python code:

```
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())

#5
# 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)
```

The output of the code is displayed below the code cell, showing the aggregated data as a table with two columns: 'date' and 'total_vaccinations'. The table contains 473 rows of data, with the first few rows showing dates from 2020-12-04 to 2020-12-16 and total vaccination counts ranging from 1.000000e+00 to 4.000000e+00. The output is summarized as [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:

The screenshot shows a Jupyter Notebook with a file explorer on the left containing various CSV files related to COVID-19 data. The main area displays a Python code cell for PCA reduction. The code imports pandas and PCA from sklearn, reads a CSV file, drops unnecessary columns, selects features for vaccination status, and performs PCA with 2 components. The output shows the first two principal components (PC1 and PC2) for 38847 rows of data.

```

#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)

```

```

(154) ✓ 0.2s
... Reduced dataset:
      PC1      PC2
0  -2.644590e+07  -236176.922415
1  -2.642373e+07  -209303.362467
2  -2.609737e+07  462121.112071
3  -2.609880e+07  -48381.413345
4  -2.643749e+07  -158546.536157
...
38842  -2.896442e+07  -52359.706288
38843  -2.889788e+07  -82632.214663
38844  -2.886862e+07  -96619.061338
38845  -2.881767e+07  -123969.505657
38846  -2.874957e+07  -162309.174449
[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:

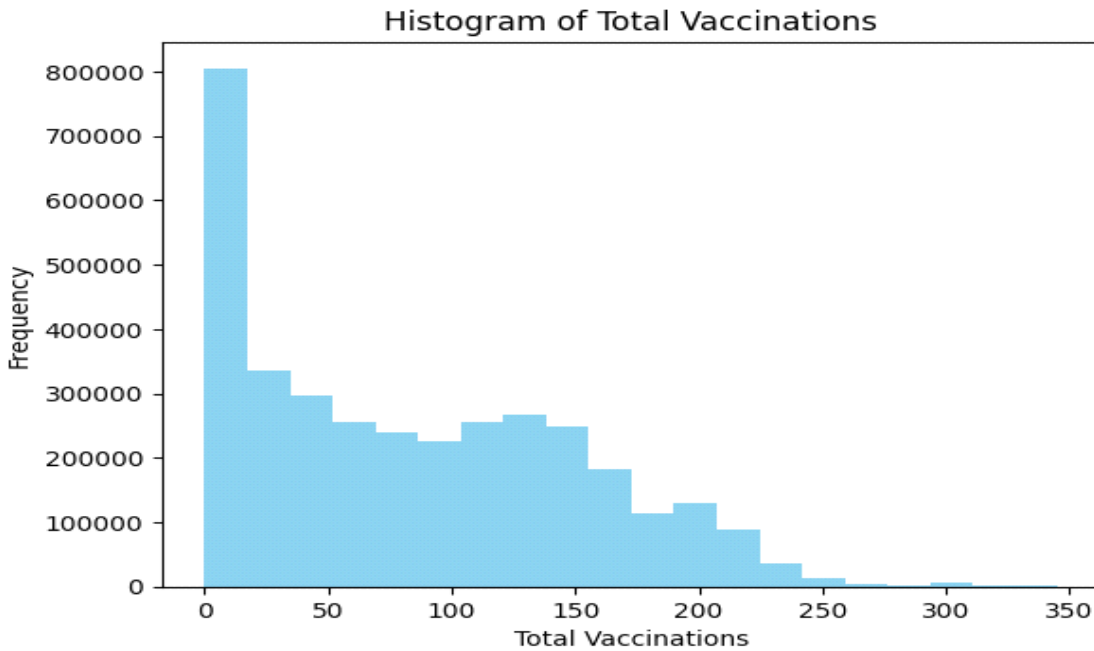
```

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:

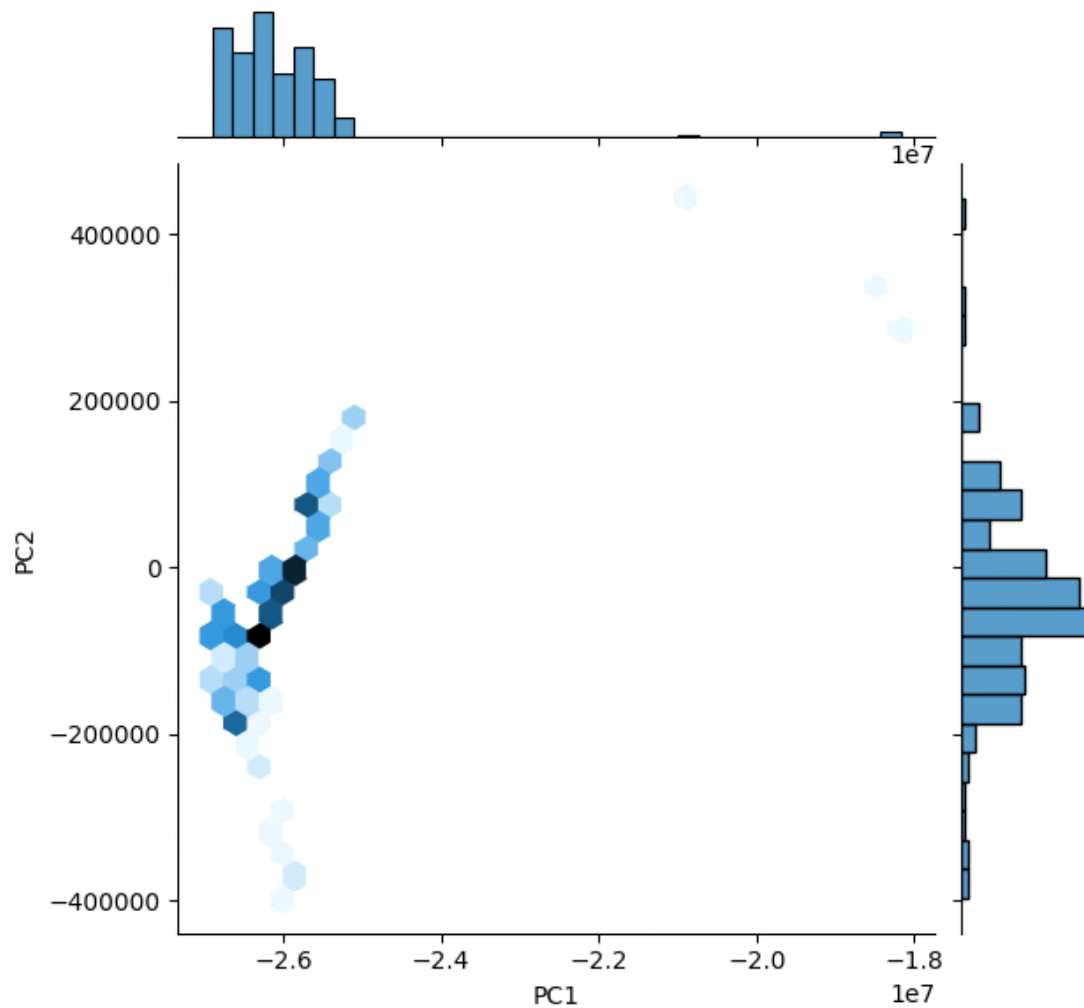


- Following this is the Joint plot representation of the reduced dataset that we have created during data reduction.

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:



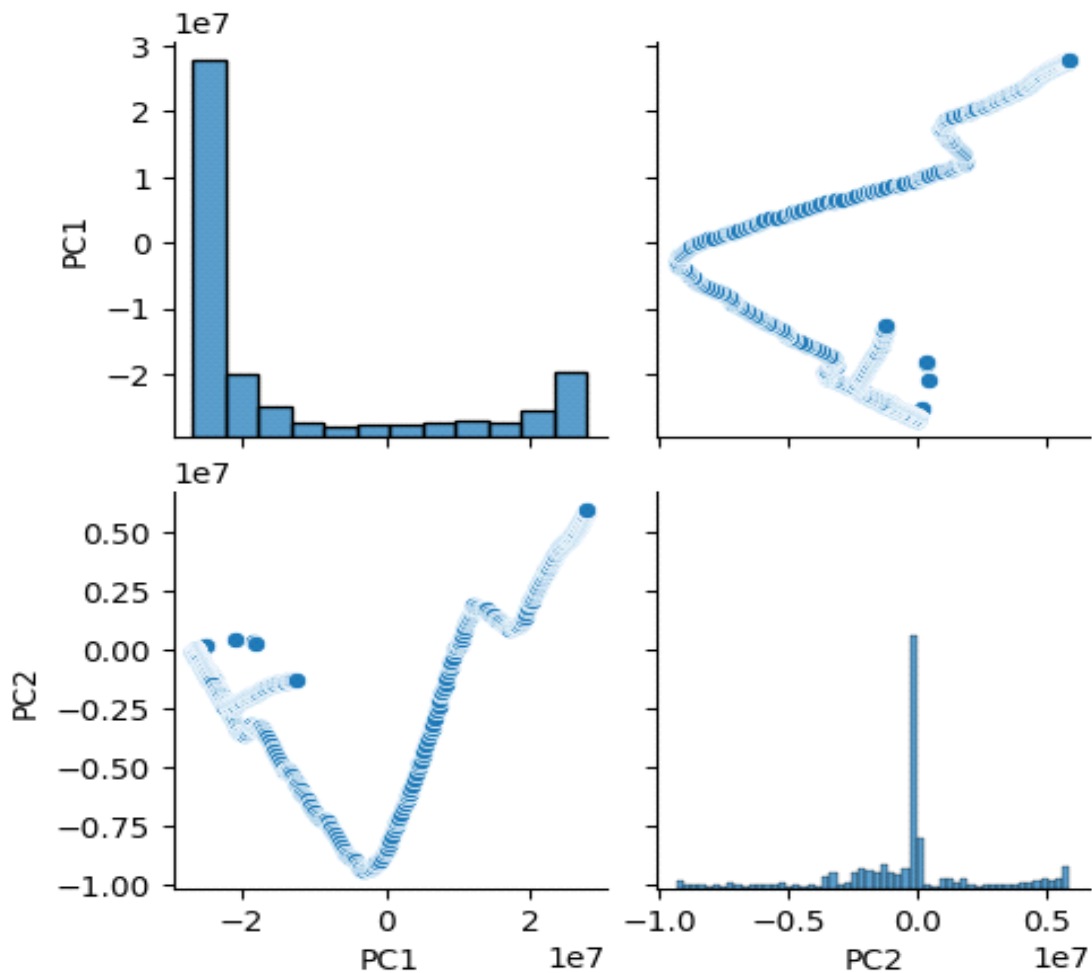
- Thirdly we have made use of the pair plot visualizing technique to visualize the reduced data sets.

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:

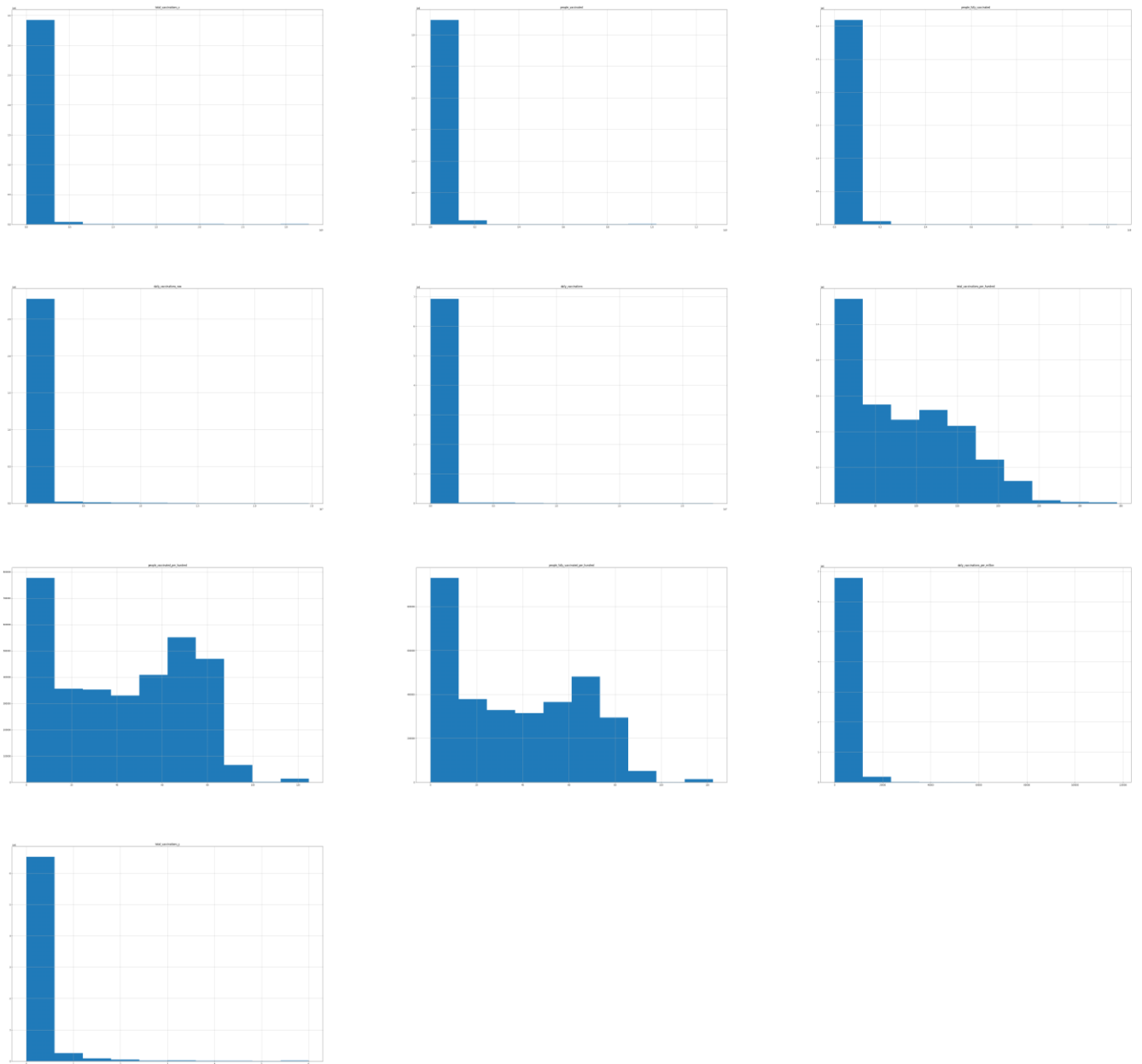


- Following these pair plots we have made use of the merged dataset to visualize it in histogram as it contains a combination of different columns and rows.

Program:

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

Output:

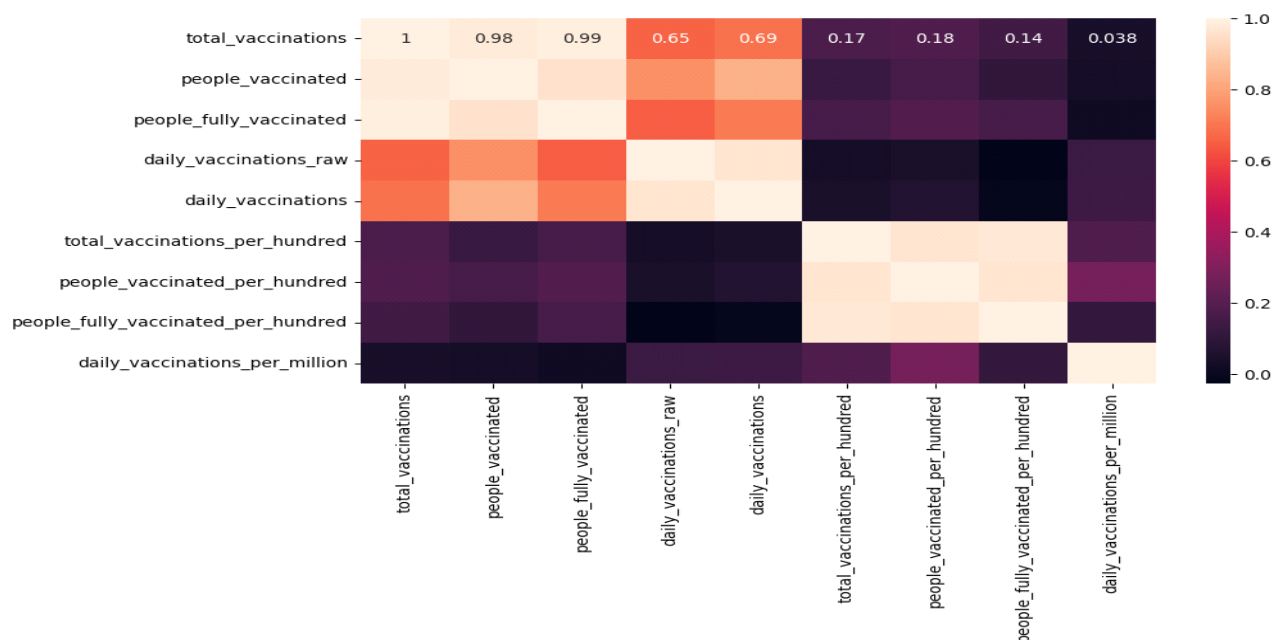


- Being last we have calculated the correlations for the country_vaccinations dataset and by making use of it we even have plotted a heatmap for it which is visualized below.

Program:

```
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))
```

Output:



Conclusion:

- In the quest of Covid-19 vaccine Analysis, we have embarked on a crucial journey that begins with loading and preprocessing the dataset. We have traversed through essential steps, starting with importing the necessary libraries to facilitate data manipulation and analysis.
- Understanding the data's structure, characteristics, and any potential issues exploratory data analysis (EDA) is essential for Informed decision-making.
- Data Preprocessing emerged as a pivotal aspect of this process. it involves cleaning, transforming, and refining the dataset to ensure that it aligns with the requirement of machine learning algorithms.
- With these foundational steps, completed, our dataset is now primed for the subsequent stages of building and training a Covid-19 vaccine analysis model.