

# ***PHASE-4***

## **PERFORMING EXPLORATORY DATA ANALYSIS**

*Exploratory Data Analysis (EDA) for COVID-19 vaccine analysis some steps to perform a exploratory data analysis to my dataset ,*

- 1. Data Collection:** Gather relevant datasets on COVID-19 vaccines, including vaccine types, administration dates, locations, and demographics.*
- 2. Data Cleaning:** Remove missing or inconsistent data, correct data types, and handle outliers.*
- 3. Data Visualization:** Create various plots and charts to visualize the data. Examples include bar charts for vaccine types, time series plots for vaccination rates, and demographic breakdowns.*
- 4. Descriptive Statistics:** Calculate summary statistics, such as mean, median, and standard deviation, to get an overall sense of the data.*
- 5. Geographic Analysis:** Use maps to visualize vaccination rates by region or country.*
- 6. Time-Series Analysis:** Explore how vaccination rates evolve over time and identify any trends or seasonality.*
- 7. Demographic Analysis:** Break down the data by age groups, gender, and other demographic factors to understand vaccination distribution.*
- 8. Correlation Analysis:** Examine relationships between variables, like vaccine distribution and COVID-19 case rates.*
- 9. Hypothesis Testing:** Perform statistical tests to check for significant differences or correlations.*
- 10. Anomaly Detection:** Identify unusual patterns or outliers in the data.*

**11. Interactive Dashboards:** Consider creating interactive dashboards using tools like Tableau or Power BI for a more user-friendly exploration.

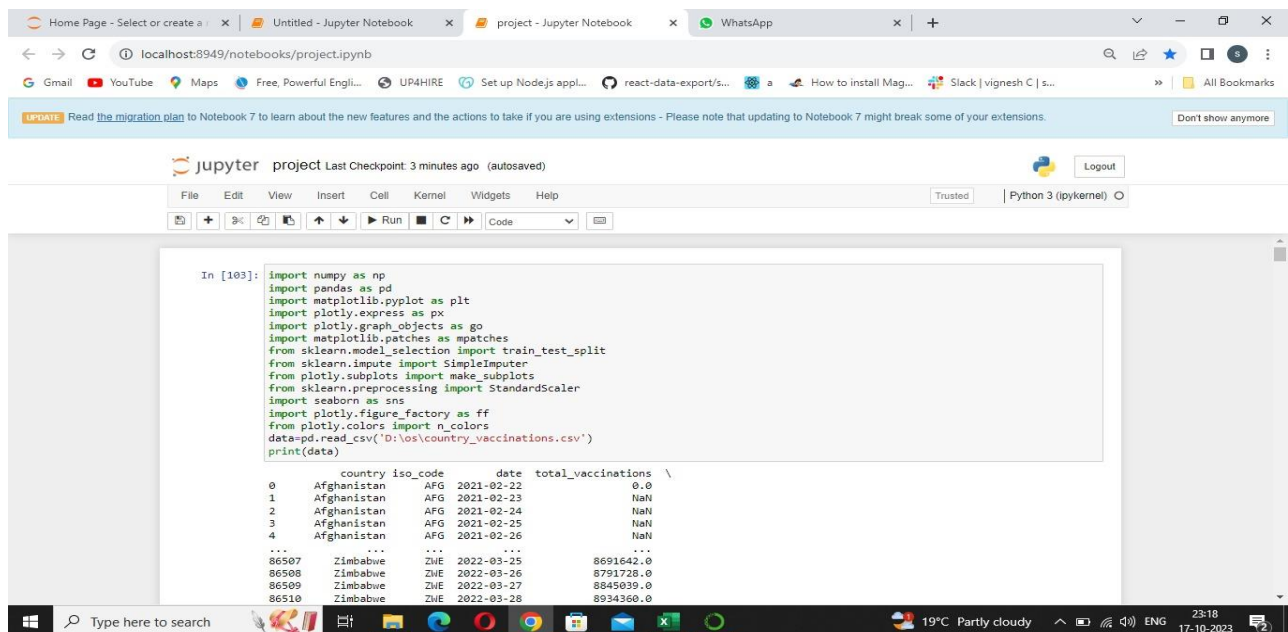
**12. Communication:** Present your findings through reports, visualizations, and clear insights.

## BEGIN BUILDING THE PROJECT BY LOAD THE DATASET

To import the required libraries and read a CSV file,

```
data=pd.read_csv('D:\os\country_vaccinations.csv')
```

*Print(data)*



The screenshot shows a Jupyter Notebook running in a web browser. The notebook has a menu bar with File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu bar is a toolbar with icons for running, saving, and other actions. The main area of the notebook displays a code cell with the following code:

```
In [103]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.patches as mpatches
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from plotly.subplots import make_subplots
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import plotly.figure_factory as ff
from plotly.colors import n_colors
data=pd.read_csv('D:\os\country_vaccinations.csv')
print(data)
```

The output of the code is a preview of the first few rows of the 'country\_vaccinations.csv' dataset. The data is displayed in a table format with columns: country, iso\_code, date, and total\_vaccinations. The first five rows show data for Afghanistan, and the next five rows show data for Zimbabwe.

	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	8845039.0
86510	Zimbabwe	ZWE	2022-03-28	8934360.0

## PREPROCESS DATASET

**Import The Required Libraries:**

```
import numpy as np
```

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```

import plotly.express as px

import plotly.graph_objects as go

import matplotlib.patches as mpatches

from sklearn.model_selection import train_test_split

from sklearn.impute import SimpleImputer

from plotly.subplots import make_subplots

from sklearn.preprocessing import StandardScaler

import seaborn as sns

import plotly.figure_factory as ff

from plotly.colors import n_colors

```

### Importing the Dataset:

- Read Dataset,

```

data=pd.read_csv('D:\os\country_vaccinations.csv')

print(data)

```

The screenshot shows a Jupyter Notebook window with the following code in a cell:

```

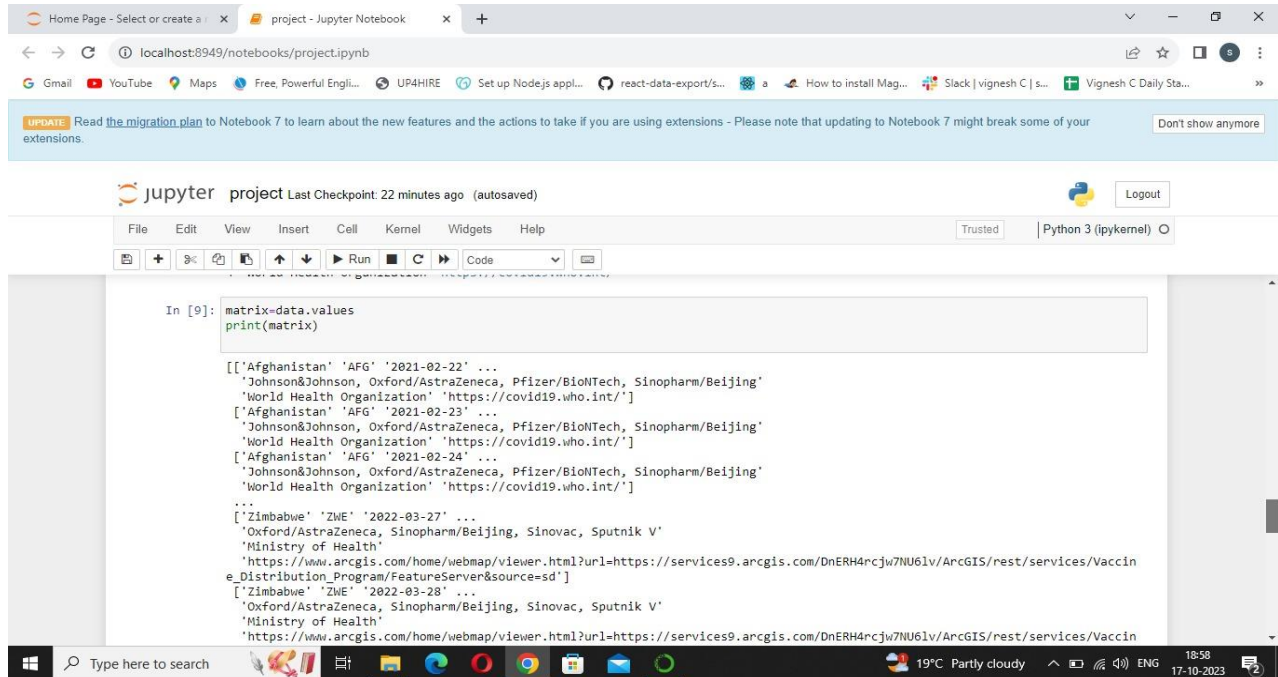
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.graph_objects as go
import matplotlib.patches as mpatches
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from plotly.subplots import make_subplots
from sklearn.preprocessing import StandardScaler
import seaborn as sns
import plotly.figure_factory as ff
from plotly.colors import n_colors
data=pd.read_csv('D:\os\country_vaccinations.csv')
print(data)

```

The output of the code is a table with the following columns: country, iso\_code, date, total\_vaccinations. The table shows data for Afghanistan and Zimbabwe.

	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	8691842.0
86508	Zimbabwe	ZWE	2022-03-26	8791728.0
86509	Zimbabwe	ZWE	2022-03-27	8845039.0
86510	Zimbabwe	ZWE	2022-03-28	8934360.0

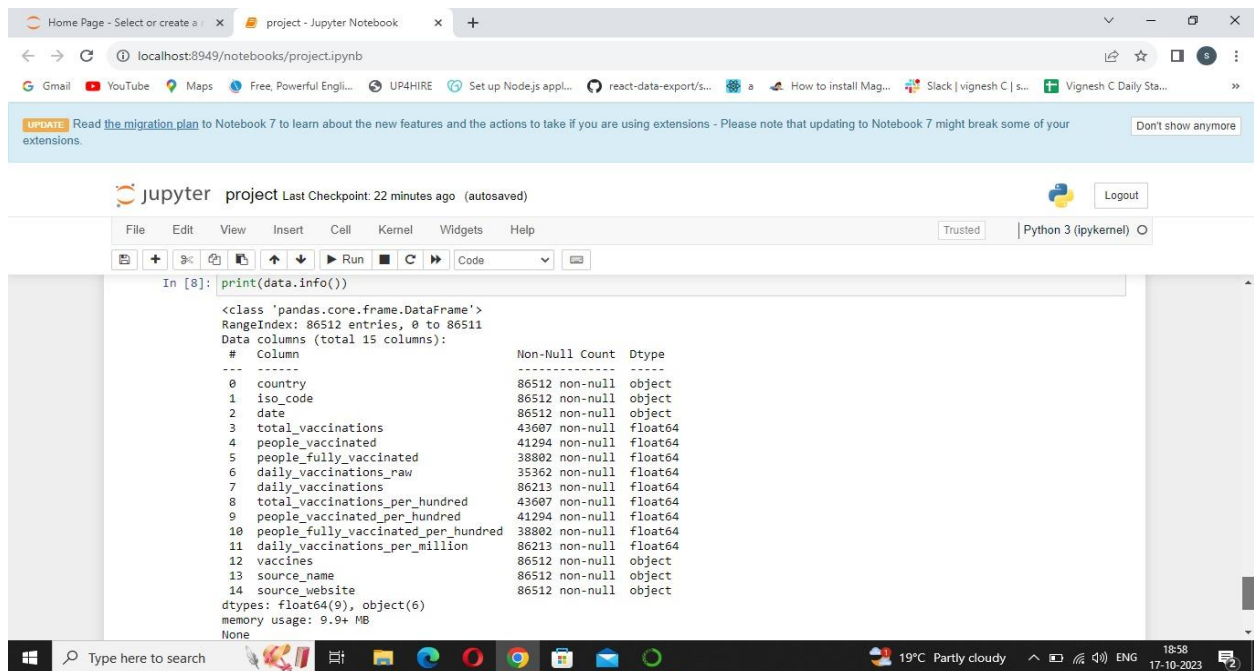
- *Create Matrix,*  
  
*matrix=data.values*  
  
*print(matrix)*



```
In [9]: matrix=data.values
print(matrix)

[['Afghanistan' 'AFG' '2021-02-22' ...
'Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing'
'World Health Organization' 'https://covid19.who.int/']
['Afghanistan' 'AFG' '2021-02-23' ...
'Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing'
'World Health Organization' 'https://covid19.who.int/']
['Afghanistan' 'AFG' '2021-02-24' ...
'Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing'
'World Health Organization' 'https://covid19.who.int/']
...
['Zimbabwe' 'ZWE' '2022-03-27' ...
'Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac, Sputnik V'
'Ministry of Health'
'https://www.arcgis.com/home/webmap/viewer.html?url=https://services9.arcgis.com/DnERH4rcjw7NU61v/ArcGIS/rest/services/Vaccin
e_Distribution_Program/FeatureServer&source=sd']
['Zimbabwe' 'ZWE' '2022-03-28' ...
'Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac, Sputnik V'
'Ministry of Health'
'https://www.arcgis.com/home/webmap/viewer.html?url=https://services9.arcgis.com/DnERH4rcjw7NU61v/ArcGIS/rest/services/Vaccin
```

- *Other Information about dataset,*

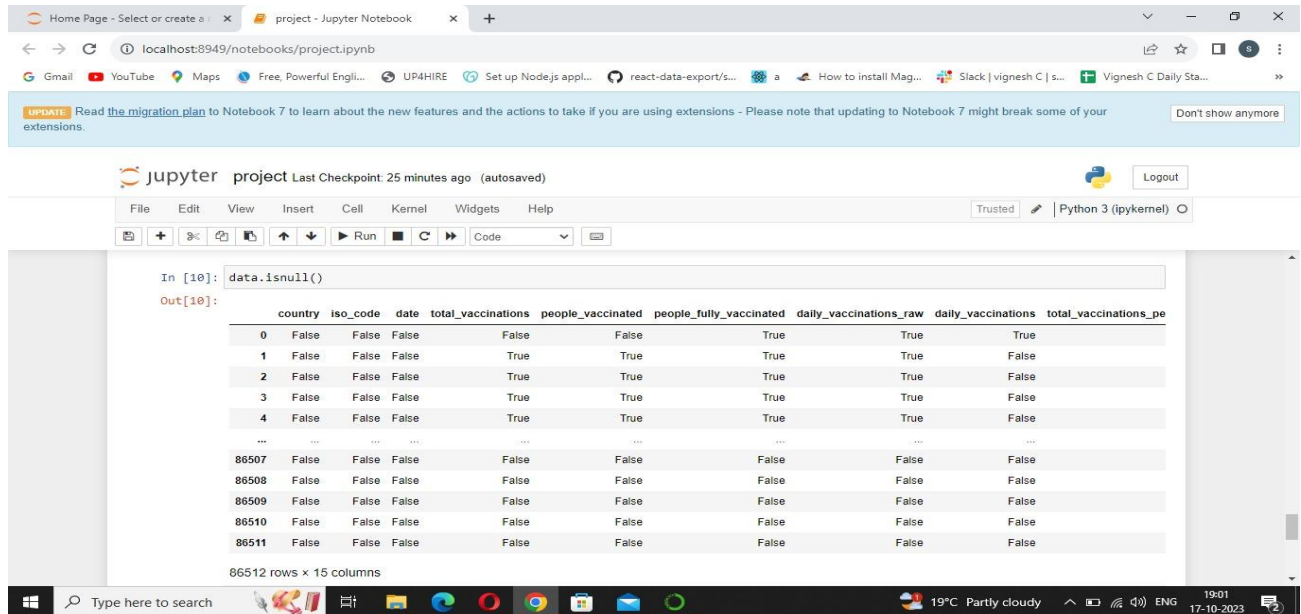


```
In [8]: print(data.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 86512 entries, 0 to 86511
Data columns (total 15 columns):
#   Column                                     Non-Null Count  Dtype
---  -
0   country                                   86512 non-null  object
1   iso_code                                  86512 non-null  object
2   date                                      86512 non-null  object
3   total_vaccinations                        43607 non-null  float64
4   people_vaccinated                        41294 non-null  float64
5   people_fully_vaccinated                  38802 non-null  float64
6   daily_vaccinations_raw                   35362 non-null  float64
7   daily_vaccinations                       86213 non-null  float64
8   total_vaccinations_per_hundred           43607 non-null  float64
9   people_vaccinated_per_hundred            41294 non-null  float64
10  people_fully_vaccinated_per_hundred       38802 non-null  float64
11  daily_vaccinations_per_million            86213 non-null  float64
12  vaccines                                  86512 non-null  object
13  source_name                              86512 non-null  object
14  source_website                           86512 non-null  object
dtypes: float64(9), object(6)
memory usage: 9.9+ MB
None
```

## Handling The Missing Data:

Before Handling the Missing data, we use `isnull()` to show the null values and using `isnull().sum` to get total number of null values.



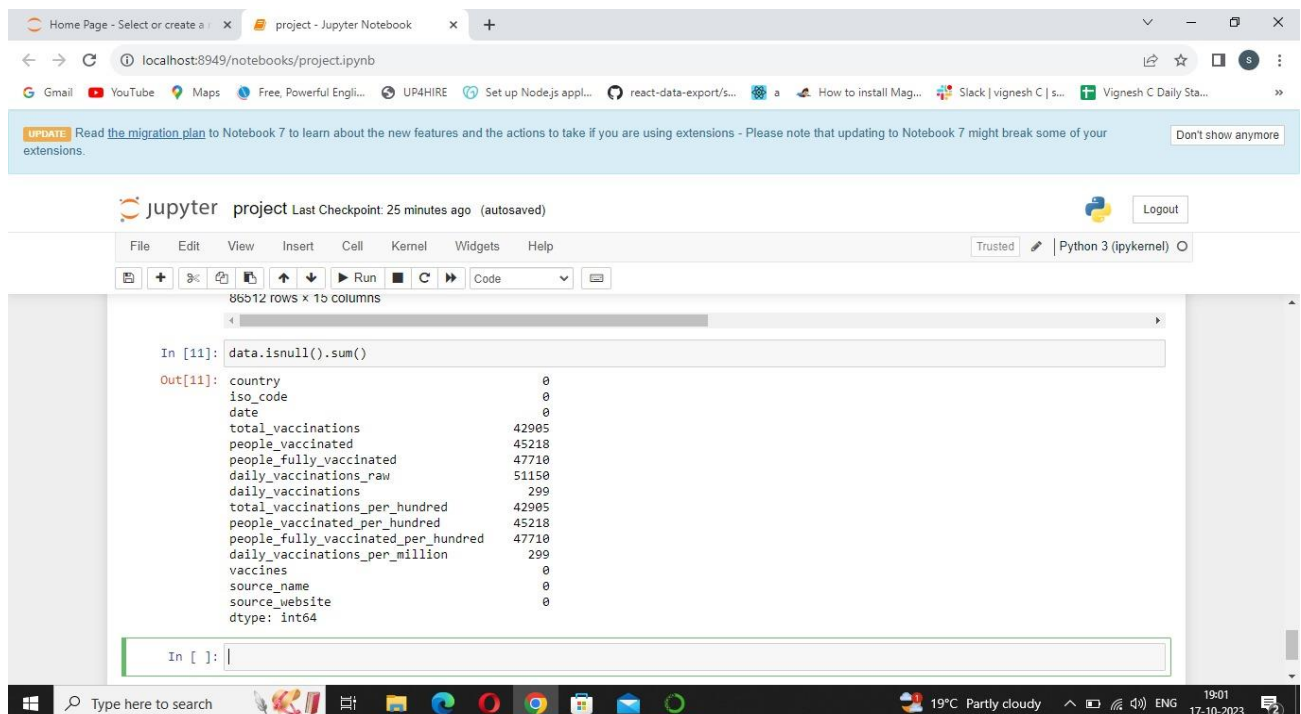
The screenshot shows a Jupyter Notebook interface with the following content:

```
In [10]: data.isnull()
```

The output is a table showing the presence of null values for each column across the first few rows of the dataset. The table has 10 columns: country, iso\_code, date, total\_vaccinations, people\_vaccinated, people\_fully\_vaccinated, daily\_vaccinations\_raw, daily\_vaccinations, and total\_vaccinations\_pe. The rows are indexed from 0 to 4, and then there is an ellipsis indicating more rows, followed by rows 86507 to 86511. The bottom of the output indicates 86512 rows x 15 columns.

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_pe
0	False	False	False	False	False	True	True	True	
1	False	False	False	True	True	True	True	True	False
2	False	False	False	True	True	True	True	True	False
3	False	False	False	True	True	True	True	True	False
4	False	False	False	True	True	True	True	True	False
...	...	...	...	...	...	...	...	...	...
86507	False	False	False	False	False	False	False	False	False
86508	False	False	False	False	False	False	False	False	False
86509	False	False	False	False	False	False	False	False	False
86510	False	False	False	False	False	False	False	False	False
86511	False	False	False	False	False	False	False	False	False

86512 rows x 15 columns



The screenshot shows a Jupyter Notebook interface with the following content:

```
In [11]: data.isnull().sum()
```

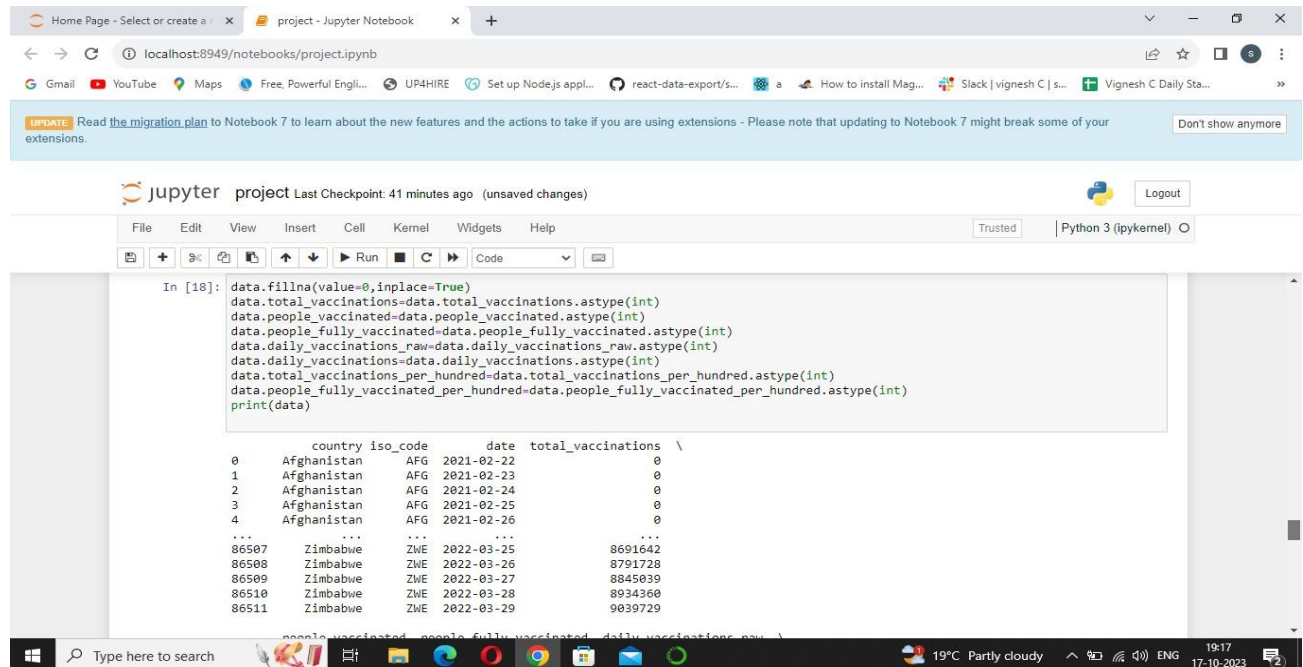
The output is a table showing the total number of null values for each column. The table has 15 columns: country, iso\_code, date, total\_vaccinations, people\_vaccinated, people\_fully\_vaccinated, daily\_vaccinations\_raw, daily\_vaccinations, total\_vaccinations\_per\_hundred, people\_vaccinated\_per\_hundred, people\_fully\_vaccinated\_per\_hundred, daily\_vaccinations\_per\_million, vaccines, source\_name, source\_website, and dtype. The values are as follows:

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

dtype: int64



The below code in the image for all the data cleaning,

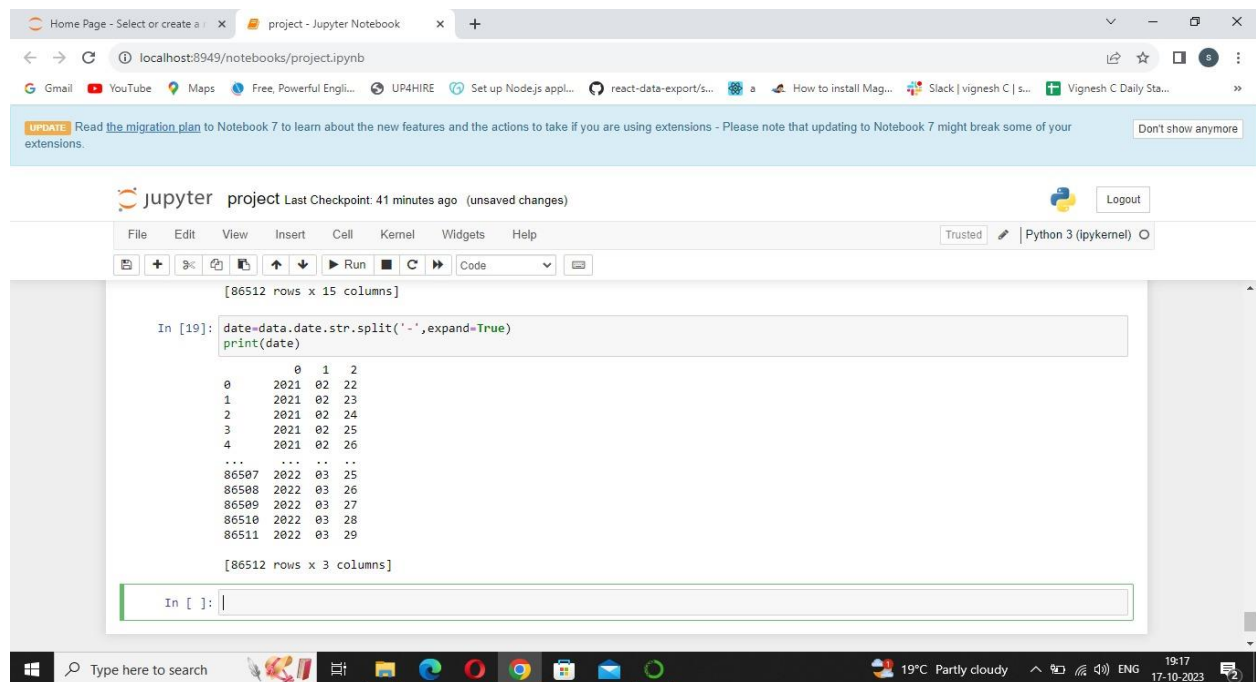


The screenshot shows a Jupyter Notebook interface with a browser window at localhost:8949/notebooks/project.ipynb. The notebook has a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The code cell contains the following Python code:

```
In [18]: data.fillna(value=0,inplace=True)
data.total_vaccinations=data.total_vaccinations.astype(int)
data.people_vaccinated=data.people_vaccinated.astype(int)
data.people_fully_vaccinated=data.people_fully_vaccinated.astype(int)
data.daily_vaccinations_raw=data.daily_vaccinations_raw.astype(int)
data.daily_vaccinations=data.daily_vaccinations.astype(int)
data.total_vaccinations_per_hundred=data.total_vaccinations_per_hundred.astype(int)
data.people_fully_vaccinated_per_hundred=data.people_fully_vaccinated_per_hundred.astype(int)
print(data)
```

The output of the code is a DataFrame with the following columns: country, iso\_code, date, total\_vaccinations, people\_vaccinated, people\_fully\_vaccinated, daily\_vaccinations\_raw, daily\_vaccinations, total\_vaccinations\_per\_hundred, and people\_fully\_vaccinated\_per\_hundred. The output shows data for Afghanistan and Zimbabwe.

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred	people_fully_vaccinated_per_hundred
0	Afghanistan	AFG	2021-02-22	0	0	0	0	0	0	0
1	Afghanistan	AFG	2021-02-23	0	0	0	0	0	0	0
2	Afghanistan	AFG	2021-02-24	0	0	0	0	0	0	0
3	Afghanistan	AFG	2021-02-25	0	0	0	0	0	0	0
4	Afghanistan	AFG	2021-02-26	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
86507	Zimbabwe	ZWE	2022-03-25	8691642	8691642	8691642	8691642	8691642	8691642	8691642
86508	Zimbabwe	ZWE	2022-03-26	8791728	8791728	8791728	8791728	8791728	8791728	8791728
86509	Zimbabwe	ZWE	2022-03-27	8845039	8845039	8845039	8845039	8845039	8845039	8845039
86510	Zimbabwe	ZWE	2022-03-28	8934360	8934360	8934360	8934360	8934360	8934360	8934360
86511	Zimbabwe	ZWE	2022-03-29	9039729	9039729	9039729	9039729	9039729	9039729	9039729



The screenshot shows a Jupyter Notebook interface with a browser window at localhost:8949/notebooks/project.ipynb. The notebook has a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar. The code cell contains the following Python code:

```
In [19]: date=date.date.str.split('-',expand=True)
print(date)
```

The output of the code is a DataFrame with the following columns: date, date\_0, date\_1, date\_2. The output shows data for Afghanistan and Zimbabwe.

	date	date_0	date_1	date_2
0	2021-02-22	2021	02	22
1	2021-02-23	2021	02	23
2	2021-02-24	2021	02	24
3	2021-02-25	2021	02	25
4	2021-02-26	2021	02	26
...	...	...	...	...
86507	2022-03-25	2022	03	25
86508	2022-03-26	2022	03	26
86509	2022-03-27	2022	03	27
86510	2022-03-28	2022	03	28
86511	2022-03-29	2022	03	29

Home Page - Select or create a x project - Jupyter Notebook x +

localhost:8949/notebooks/project.ipynb

UPDATE: Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

Jupyter project Last Checkpoint: an hour ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [22]:

```
data['year']=date[0]
data['month']=date[1]
data['day']=date[2]
data.year=pd.to_numeric(data.year)
data.month=pd.to_numeric(data.month)
data.day=pd.to_numeric(data.day)
data.date=pd.to_datetime(data.date)
data.head()
```

Out[22]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per
0	Afghanistan	AFG	2021-02-22	0	0	0	0	0	
1	Afghanistan	AFG	2021-02-23	0	0	0	0	1367	
2	Afghanistan	AFG	2021-02-24	0	0	0	0	1367	
3	Afghanistan	AFG	2021-02-25	0	0	0	0	1367	

19°C Partly cloudy 19:23 17-10-2023

Home Page - Select or create a x project - Jupyter Notebook x +

localhost:8949/notebooks/project.ipynb

UPDATE: Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

Jupyter project Last Checkpoint: an hour ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

In [23]:

```
data.date=pd.to_datetime(data.date)
data.head()
```

Out[23]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per
0	Afghanistan	AFG	2021-02-22	0	0	0	0	0	
1	Afghanistan	AFG	2021-02-23	0	0	0	0	1367	
2	Afghanistan	AFG	2021-02-24	0	0	0	0	1367	
3	Afghanistan	AFG	2021-02-25	0	0	0	0	1367	
4	Afghanistan	AFG	2021-02-26	0	0	0	0	1367	

19°C Partly cloudy 19:25 17-10-2023

Home Page - Select or create a... project - Jupyter Notebook WhatsApp

localhost:8949/notebooks/project.ipynb

Gmail YouTube Maps Free, Powerful Engli... UP4HIRE Set up Node.js appl... react-data-export/s... How to install Mag... Slack | vignesh C | s... Vignesh C Daily Sta...

**UPDATE:** Read [the migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. [Don't show anymore](#)

Jupyter project Last Checkpoint: an hour ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

	Country	Date					
3	Alghanistan	AFG 2021-02-25	0	0	0	0	1367
4	Alghanistan	AFG 2021-02-26	0	0	0	0	1367

```
In [26]: print('data point starts from',data.date.min())
print('data point ends at',data.date.max())
print('total number of countries in the data set',len(data.country.unique()))
print('total number of unique vaccines in the data set',len(data.vaccines.unique()))

data point starts from 2020-12-02 00:00:00
data point ends at 2022-03-29 00:00:00
total number of countries in the data set 223
total number of unique vaccines in the data set 84
```

In [ ]:

In [ ]:

In [ ]:

Type here to search 19°C Partly cloudy 19:32 17-10-2023

Home Page - Select or create a...project - Jupyter NotebookWhatsApp

localhost:8949/notebooks/project.ipynb

GmailYouTubeMapsFree, Powerful Engi...UP4HIRESet up Node.js appl...react-data-export/s...How to install Mag...Slack | vignesh C | s...Vignesh C Daily Sta...

UPDATERead the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.

Don't show anymore

jupyter project Last Checkpoint: an hour ago (autosaved)

Logout

FileEditViewInsertCellKernelWidgetsHelp

TrustedPython 3 (ipykernel)

total number of unique vaccines in the data set is

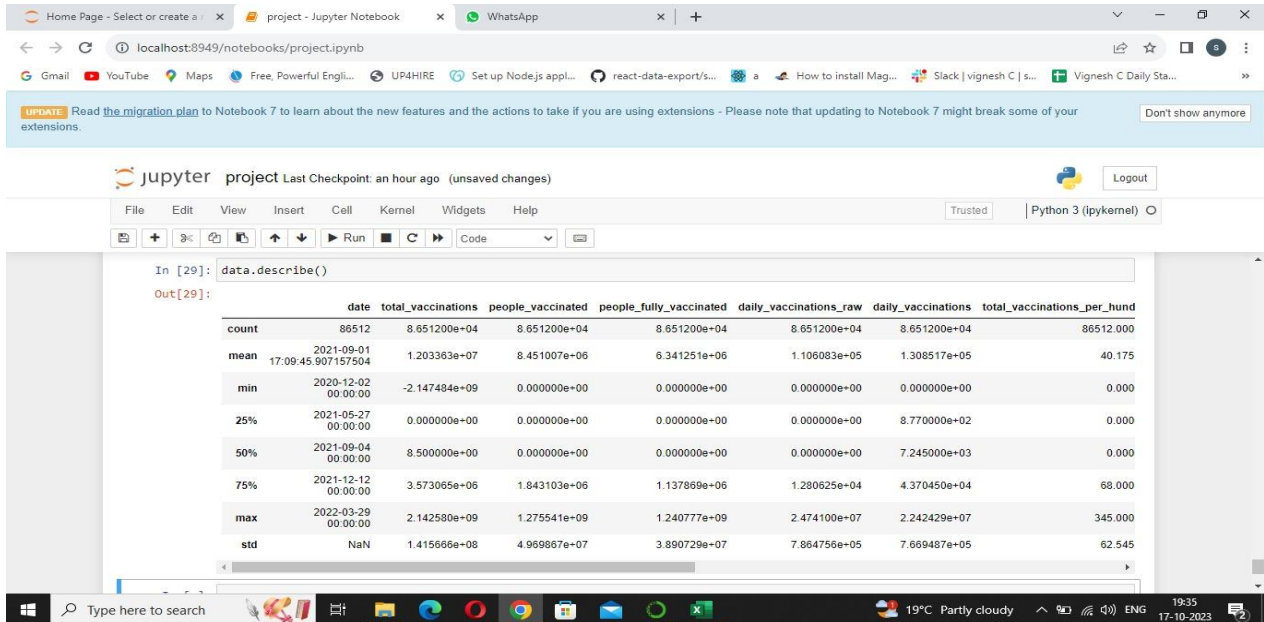
In [27]: data.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 86512 entries, 0 to 86511  
Data columns (total 18 columns):  
# ColumnNon-Null Count Dtype  
--- --  
0 country86512 non-null object  
1 iso\_code86512 non-null object  
2 date86512 non-null datetime64[ns]  
3 total\_vaccinations86512 non-null int32  
4 people\_vaccinated86512 non-null int32  
5 people\_fully\_vaccinated86512 non-null int32  
6 daily\_vaccinations\_raw86512 non-null int32  
7 daily\_vaccinations86512 non-null int32  
8 total\_vaccinations\_per\_hundred86512 non-null int32  
9 people\_vaccinated\_per\_hundred86512 non-null float64  
10 people\_fully\_vaccinated\_per\_hundred86512 non-null int32  
11 daily\_vaccinations\_per\_million86512 non-null float64  
12 vaccines86512 non-null object  
13 source\_name86512 non-null object  
14 source\_website86512 non-null object  
15 year86512 non-null int64



Using Data visualization we are going to draw some visuals to get insights from dataset,

Describe () function is used to get the statistics of each feature in dataset to get count, min, max, standard deviation, median, etc.,



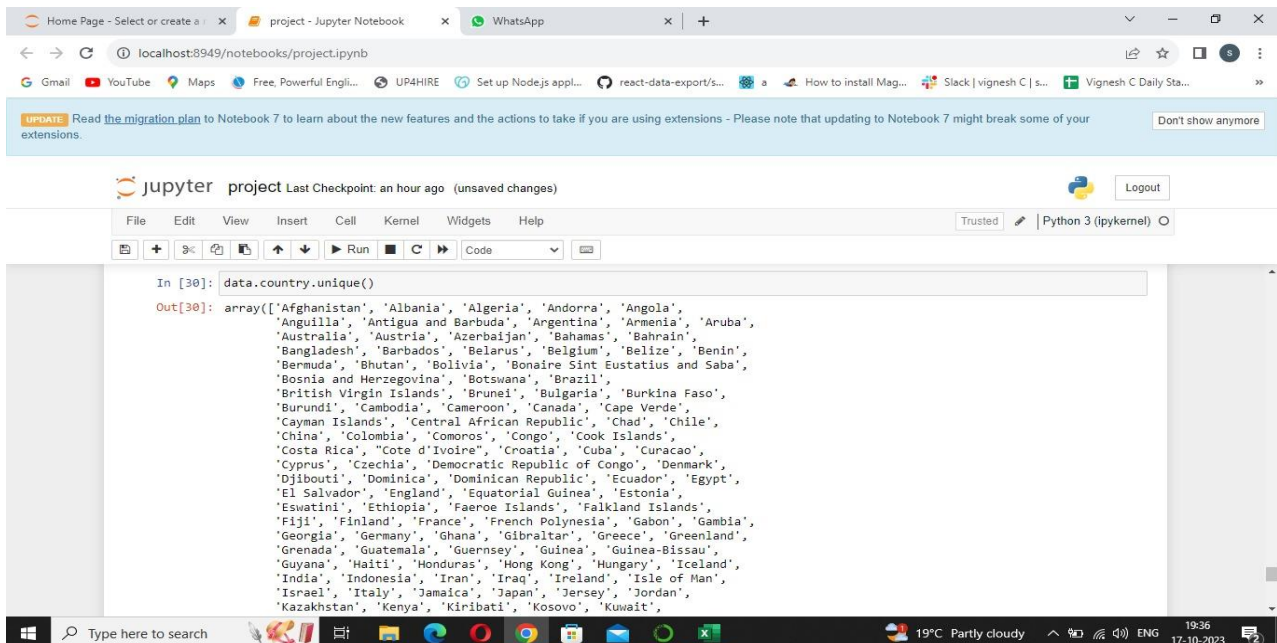
The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [29]: data.describe()
```

```
Out[29]:
```

	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hund
count	86512	8.651200e+04	8.651200e+04	8.651200e+04	8.651200e+04	8.651200e+04	86512.000
mean	2021-09-01 17:09:45.907157504	1.203363e+07	8.451007e+06	6.341251e+06	1.106083e+05	1.308517e+05	40.175
min	2020-12-02 00:00:00	-2.147484e+09	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000
25%	2021-05-27 00:00:00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	8.770000e+02	0.000
50%	2021-09-04 00:00:00	8.500000e+00	0.000000e+00	0.000000e+00	0.000000e+00	7.245000e+03	0.000
75%	2021-12-12 00:00:00	3.573065e+06	1.843103e+06	1.137869e+06	1.280625e+04	4.370450e+04	68.000
max	2022-03-29 00:00:00	2.142580e+09	1.275541e+09	1.240777e+09	2.474100e+07	2.242429e+07	345.000
std	NaN	1.415666e+08	4.969867e+07	3.890729e+07	7.864756e+05	7.669487e+05	62.545

Unique () function helps to get unique values,

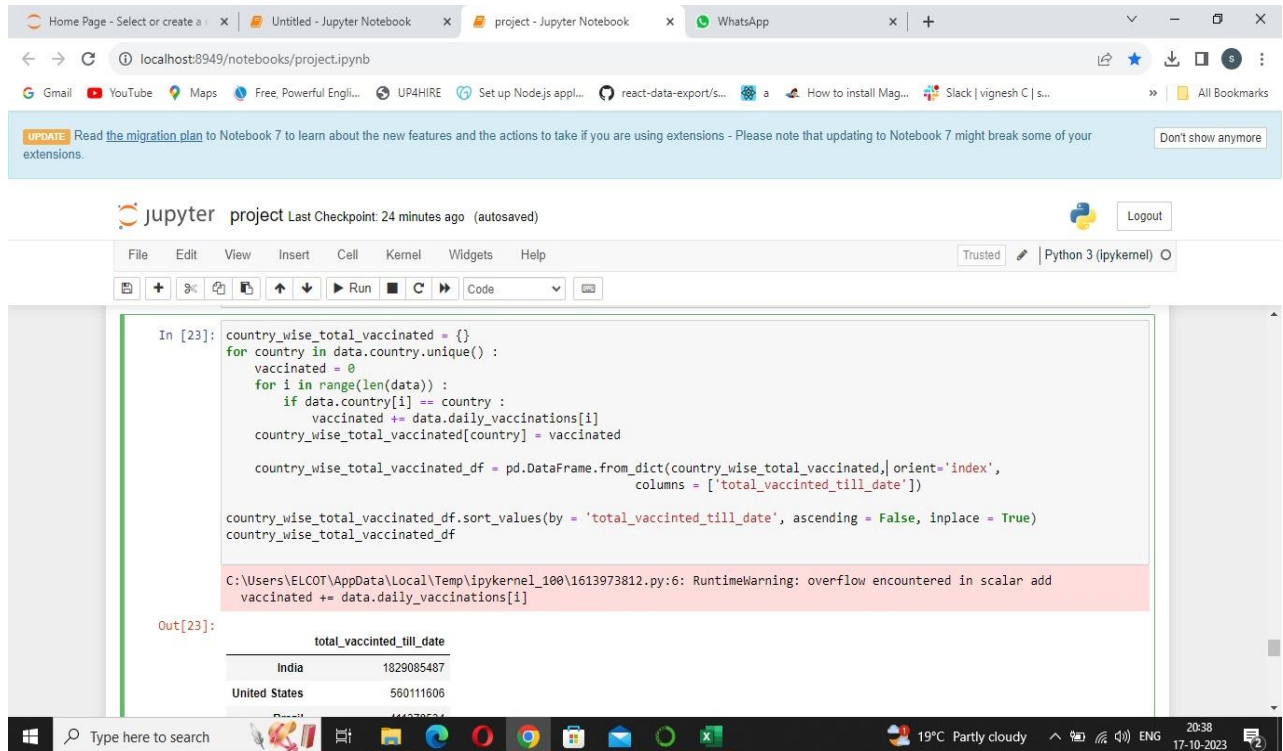


The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [30]: data.country.unique()
```

```
Out[30]: array(['Afghanistan', 'Albania', 'Algeria', 'Andorra', 'Angola', 'Anguilla', 'Antigua and Barbuda', 'Argentina', 'Armenia', 'Aruba', 'Australia', 'Austria', 'Azerbaijan', 'Bahamas', 'Bahrain', 'Bangladesh', 'Barbados', 'Belarus', 'Belgium', 'Belize', 'Benin', 'Bermuda', 'Bhutan', 'Bolivia', 'Bonaire Sint Eustatius and Saba', 'Bosnia and Herzegovina', 'Botswana', 'Brazil', 'British Virgin Islands', 'Brunei', 'Bulgaria', 'Burkina Faso', 'Burundi', 'Cambodia', 'Cameroon', 'Canada', 'Cape Verde', 'Cayman Islands', 'Central African Republic', 'Chad', 'Chile', 'China', 'Colombia', 'Comoros', 'Congo', 'Cook Islands', 'Costa Rica', 'Cote d'Ivoire', 'Croatia', 'Cuba', 'Curacao', 'Cyprus', 'Czechia', 'Democratic Republic of Congo', 'Denmark', 'Djibouti', 'Dominica', 'Dominican Republic', 'Ecuador', 'Egypt', 'El Salvador', 'England', 'Equatorial Guinea', 'Estonia', 'Eswatini', 'Ethiopia', 'Faeroe Islands', 'Falkland Islands', 'Fiji', 'Finland', 'France', 'French Polynesia', 'Gabon', 'Gambia', 'Georgia', 'Germany', 'Ghana', 'Gibraltar', 'Greece', 'Greenland', 'Grenada', 'Guatemala', 'Guernsey', 'Guinea', 'Guinea-Bissau', 'Guyana', 'Haiti', 'Honduras', 'Hong Kong', 'Hungary', 'Iceland', 'India', 'Indonesia', 'Iran', 'Iraq', 'Ireland', 'Isle of Man', 'Israel', 'Italy', 'Jamaica', 'Japan', 'Jersey', 'Jordan', 'Kazakhstan', 'Kenya', 'Kiribati', 'Kosovo', 'Kuwait',
```

*To see how many total vaccines have been used in each country using the code below,*



Home Page - Select or create a x | Untitled - Jupyter Notebook x | project - Jupyter Notebook x | WhatsApp x | +

localhost:8949/notebooks/project.ipynb

UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter project Last Checkpoint: 24 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [23]: country_wise_total_vaccinated = {}
for country in data.country.unique():
    vaccinated = 0
    for i in range(len(data)):
        if data.country[i] == country:
            vaccinated += data.daily_vaccinations[i]
    country_wise_total_vaccinated[country] = vaccinated

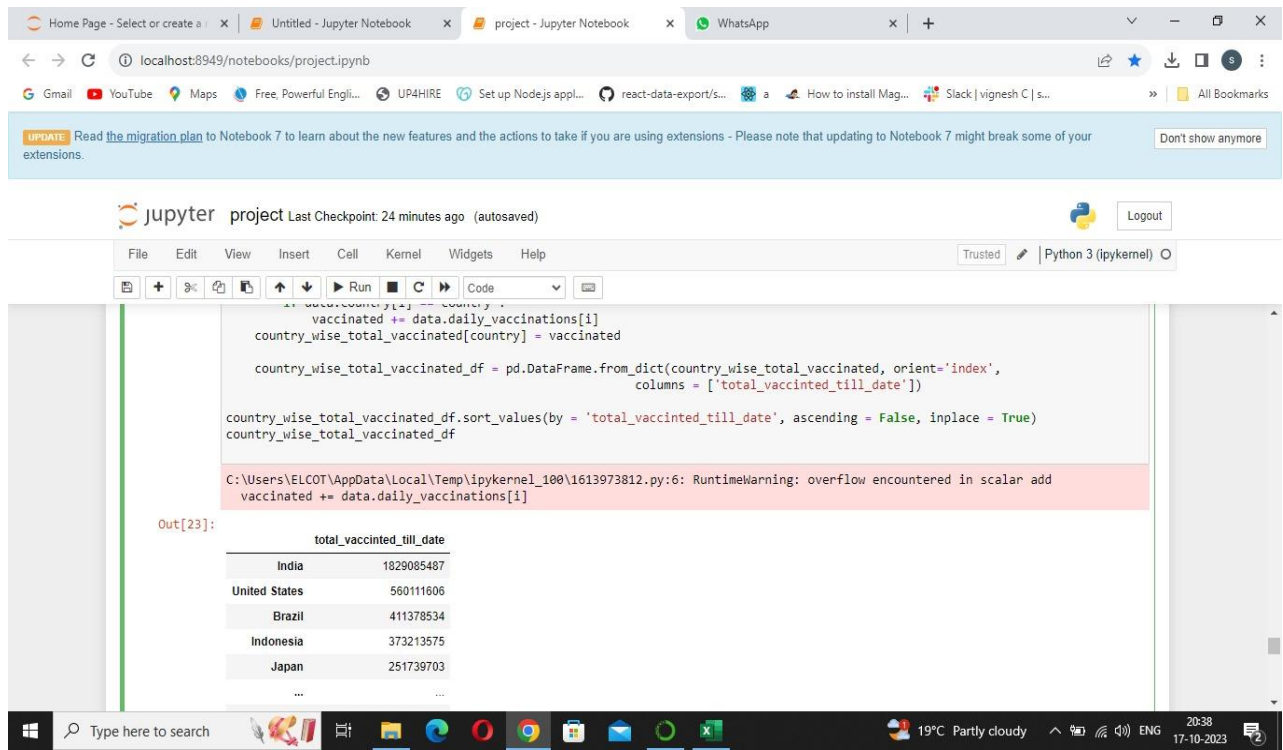
country_wise_total_vaccinated_df = pd.DataFrame.from_dict(country_wise_total_vaccinated, orient='index',
                                                         columns = ['total_vaccinted_till_date'])

country_wise_total_vaccinated_df.sort_values(by = 'total_vaccinted_till_date', ascending = False, inplace = True)
country_wise_total_vaccinated_df
```

C:\Users\ELCOT\AppData\Local\Temp\ipykernel\_100\1613973812.py:6: RuntimeWarning: overflow encountered in scalar add  
vaccinated += data.daily\_vaccinations[i]

Out[23]:

	total_vaccinted_till_date
India	1829085487
United States	560111606



Home Page - Select or create a x | Untitled - Jupyter Notebook x | project - Jupyter Notebook x | WhatsApp x | +

localhost:8949/notebooks/project.ipynb

UPDATE Read the [migration plan](#) to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions. Don't show anymore

jupyter project Last Checkpoint: 24 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

```
In [23]: country_wise_total_vaccinated = {}
for country in data.country.unique():
    vaccinated += data.daily_vaccinations[i]
    country_wise_total_vaccinated[country] = vaccinated

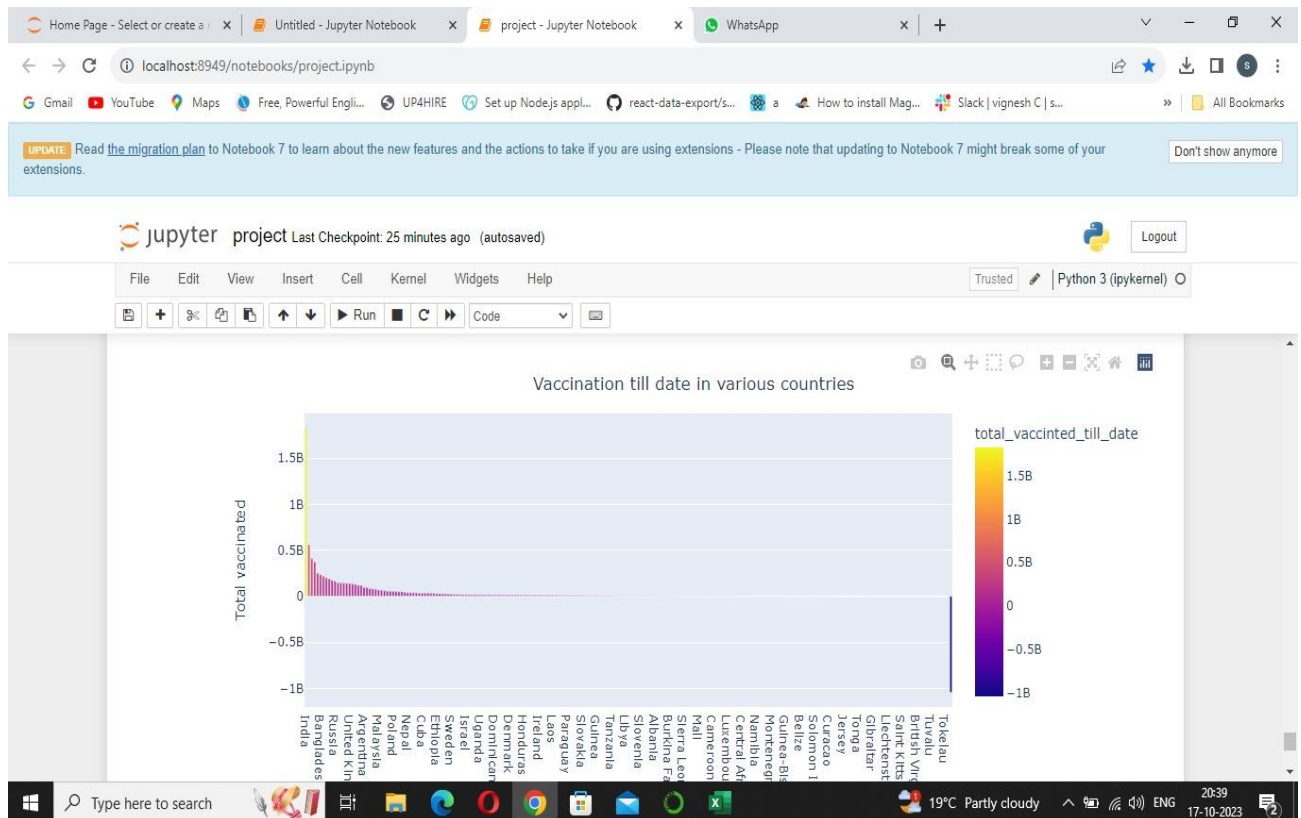
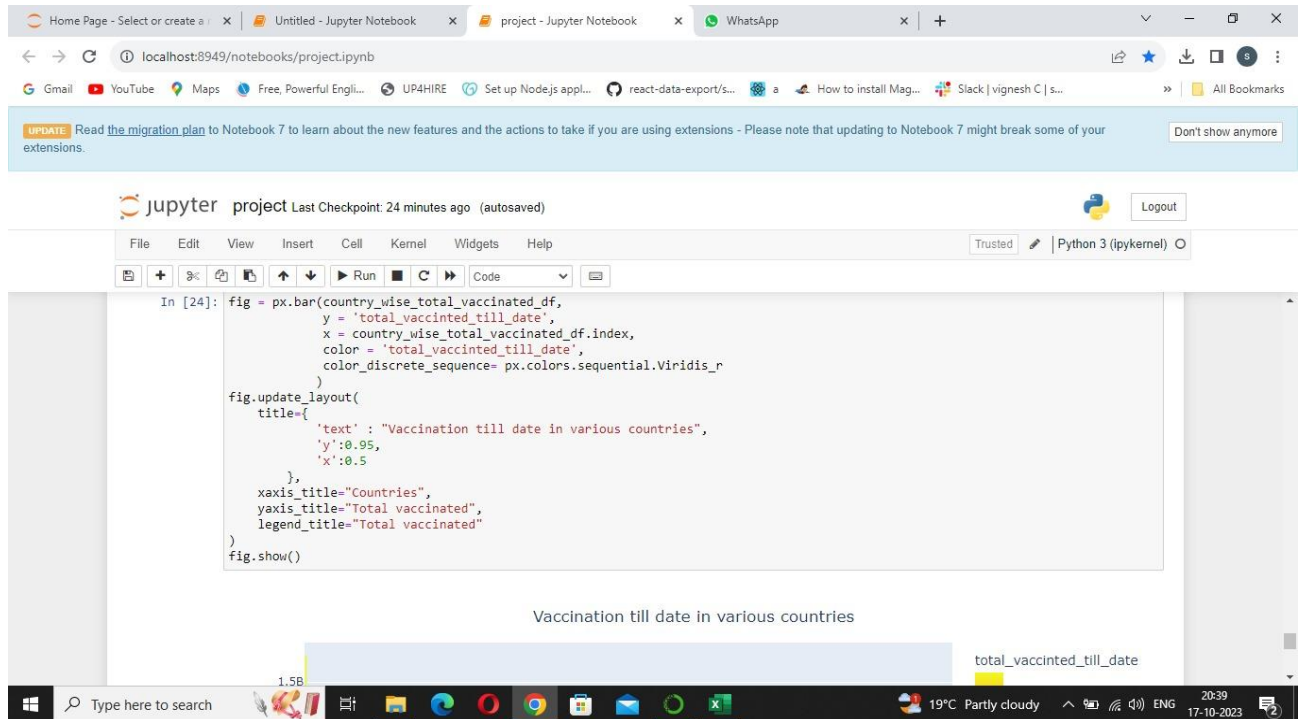
country_wise_total_vaccinated_df = pd.DataFrame.from_dict(country_wise_total_vaccinated, orient='index',
                                                         columns = ['total_vaccinted_till_date'])

country_wise_total_vaccinated_df.sort_values(by = 'total_vaccinted_till_date', ascending = False, inplace = True)
country_wise_total_vaccinated_df
```

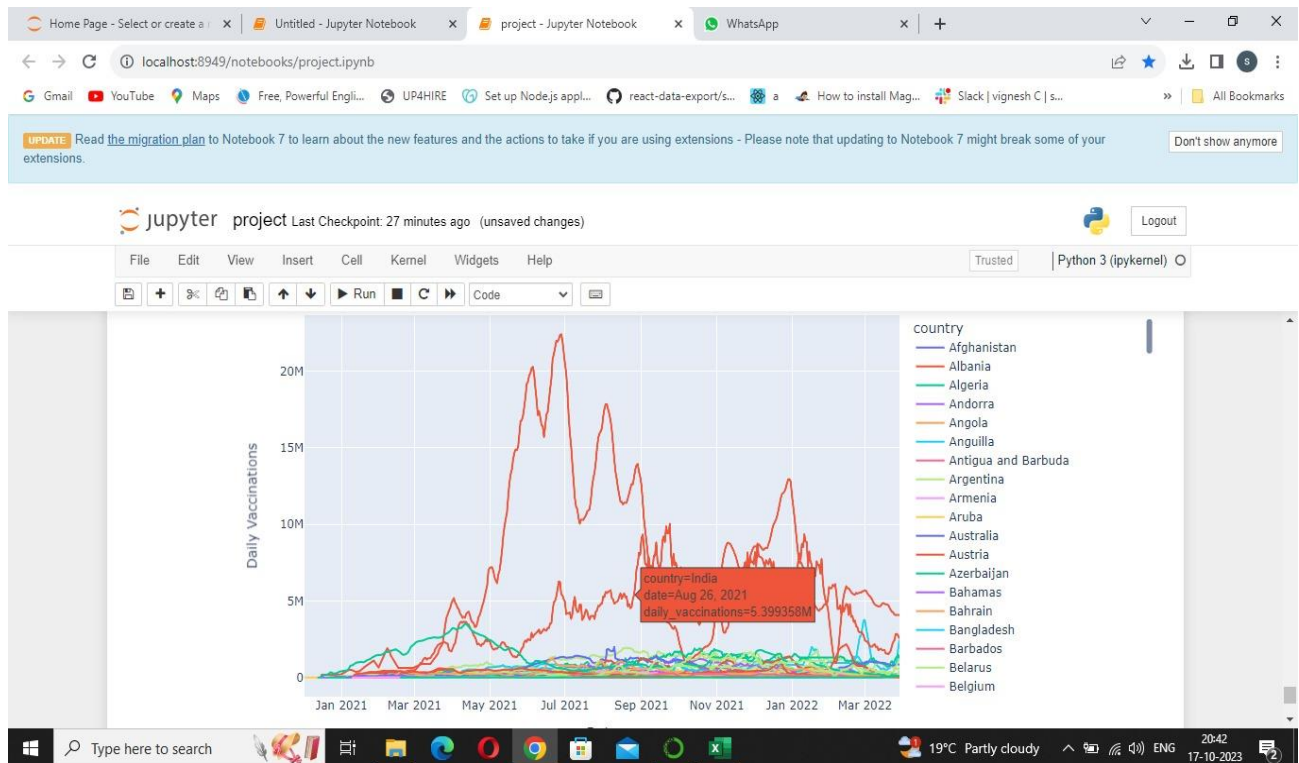
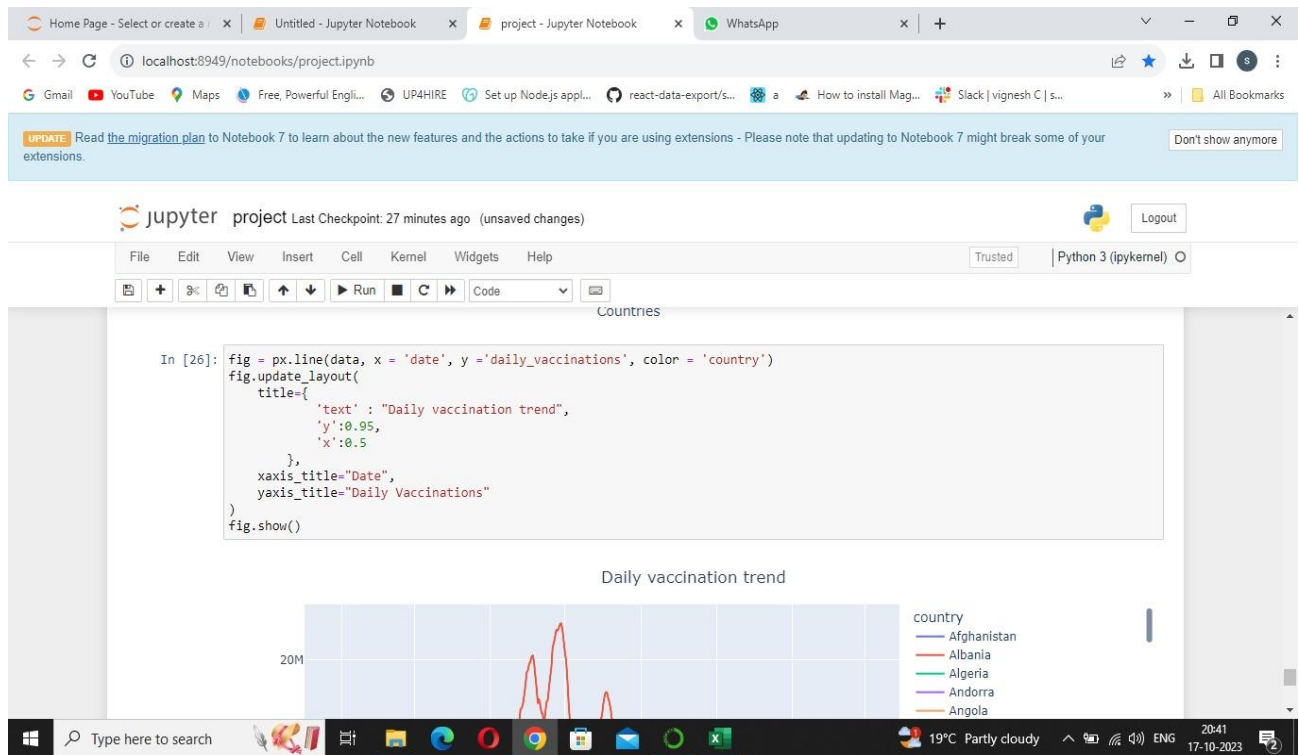
C:\Users\ELCOT\AppData\Local\Temp\ipykernel\_100\1613973812.py:6: RuntimeWarning: overflow encountered in scalar add  
vaccinated += data.daily\_vaccinations[i]

Out[23]:

	total_vaccinted_till_date
India	1829085487
United States	560111606
Brazil	411378534
Indonesia	373213575
Japan	251739703
...	...

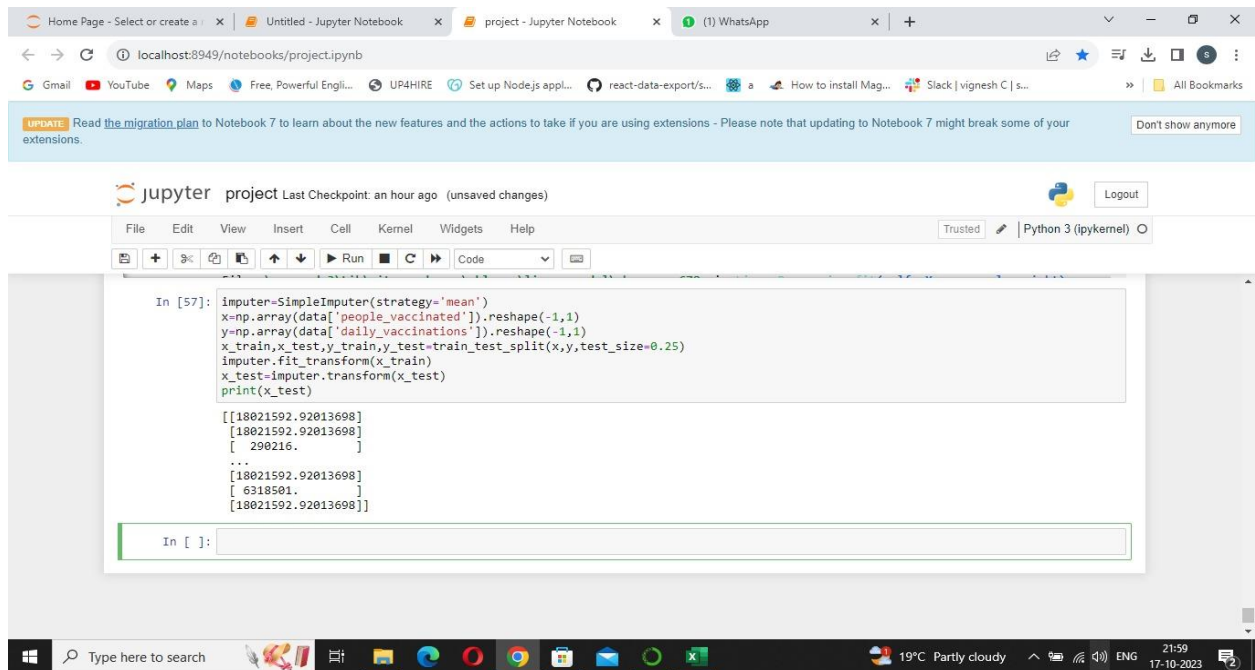


To draw a line plot where x-axis is Date and the y-axis is daily\_vaccination using the in the image,





Now, using the `sklearn.preprocessing` library contains class called `imputer`, helps in missing data by using the below:



The screenshot shows a Jupyter Notebook interface with a code cell containing the following Python code:

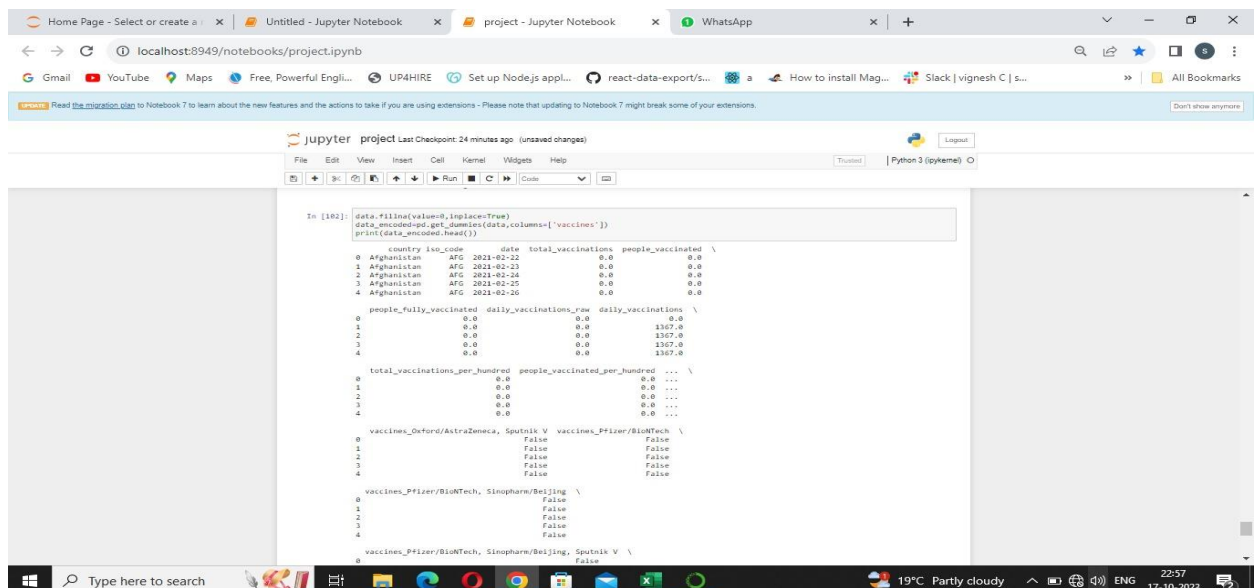
```
In [57]: imputer=SimpleImputer(strategy='mean')
x=np.array(data['people_vaccinated']).reshape(-1,1)
y=np.array(data['daily_vaccinations']).reshape(-1,1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
imputer.fit_transform(x_train)
x_test=imputer.transform(x_test)
print(x_test)
```

The output of the code is:

```
[[18021592.92013698]
 [18021592.92013698]
 [ 290216.         ]
 ...
 [18021592.92013698]
 [ 6318501.         ]
 [18021592.92013698]]
```

## Encoding categorical data(one-hot encoding)

One-hot encoding is a technique used to convert categorical data into a numerical format that machine learning algorithms can work with. Here's how you can perform one-hot encoding in Python, assuming you have a dataset with categorical variables:



The screenshot shows a Jupyter Notebook interface with a code cell containing the following Python code:

```
In [182]: data.fillna(value=0,inplace=True)
data_encoded=pd.get_dummies(data,columns=['vaccines'])
print(data_encoded.head())
```

The output of the code is a DataFrame showing the first five rows of the encoded data:

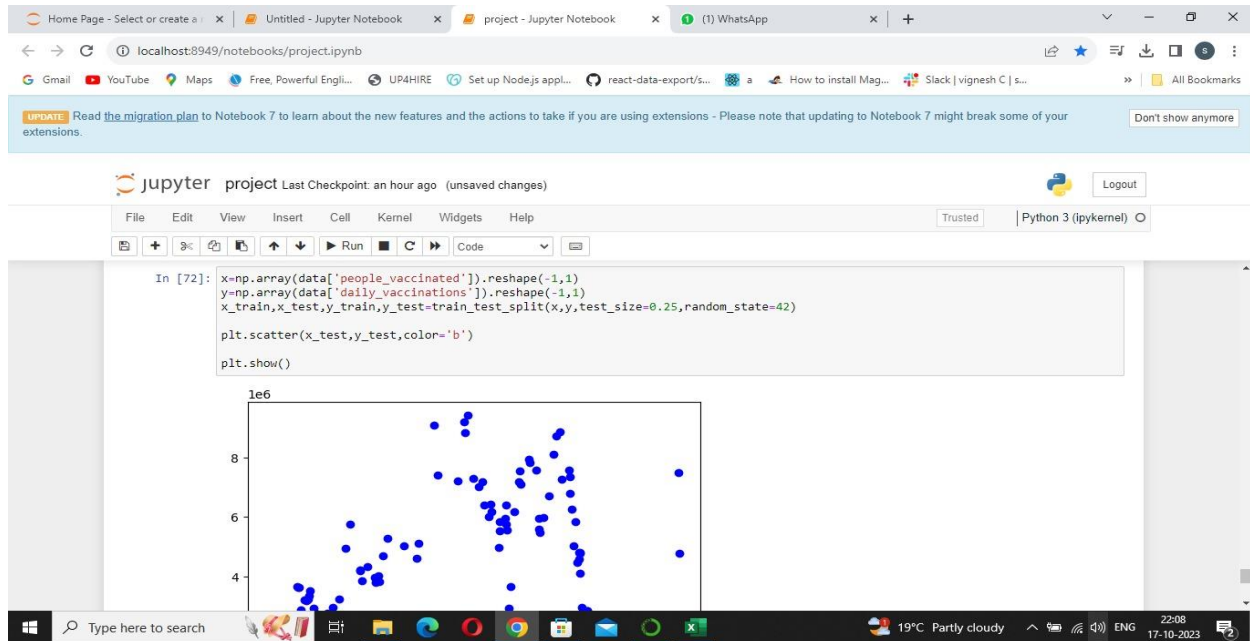
	country_iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred	people_vaccinated_per_hundred	...
0	Afghanistan	AFG	2021-02-22	0.0	0.0	0.0	0.0	0.0	0.0	...
1	Afghanistan	AFG	2021-02-23	0.0	0.0	1367.0	1367.0	0.0	0.0	...
2	Afghanistan	AFG	2021-02-24	0.0	0.0	1367.0	1367.0	0.0	0.0	...
3	Afghanistan	AFG	2021-02-25	0.0	0.0	1367.0	1367.0	0.0	0.0	...
4	Afghanistan	AFG	2021-02-26	0.0	0.0	1367.0	1367.0	0.0	0.0	...



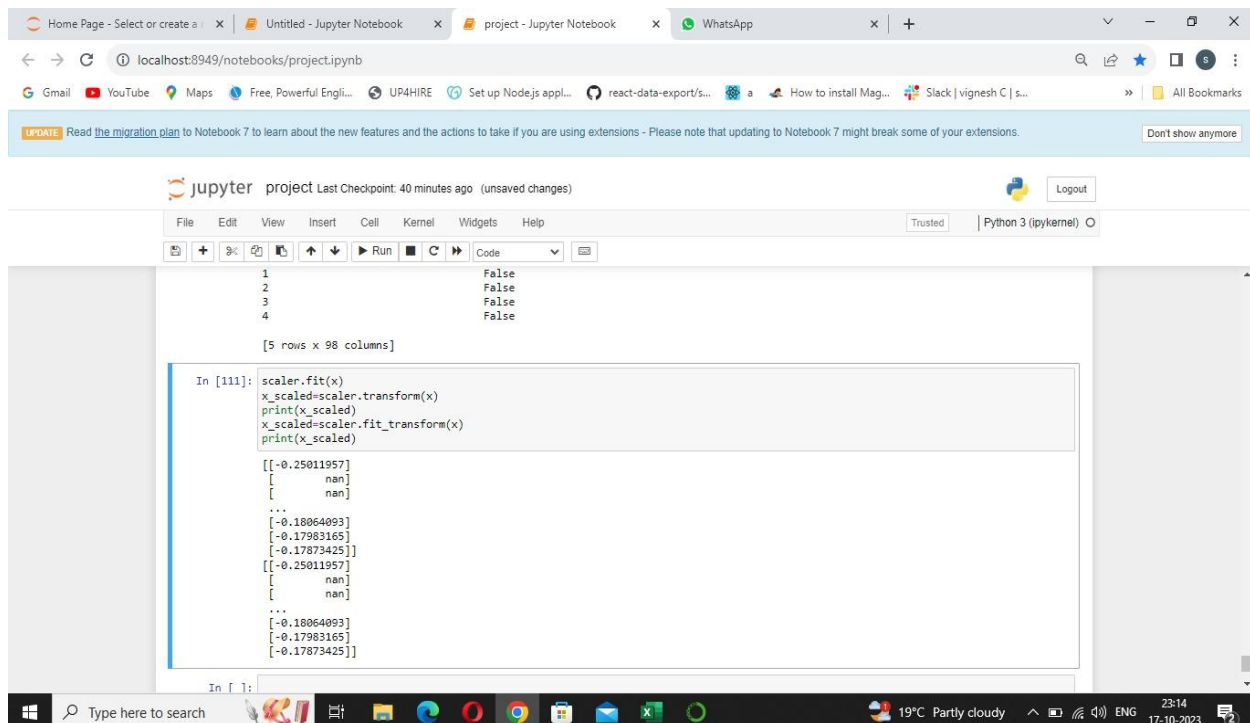
## Splitting the data set into test set and training set

By using,

*Import train\_test\_split*



## Feature Scaling



# STATISTICAL ANALYSIS

*Statistical analysis for COVID-19 vaccine analysis data can provide more in-depth insights and support decision-making. Here are some statistical techniques and analyses you can perform:*

- 1. Descriptive Statistics:** *Calculate summary statistics (mean, median, standard deviation) for key vaccine-related variables, such as vaccination rates, doses administered, and adverse events.*
- 2. Hypothesis Testing:** *Use statistical tests like t-tests or ANOVA to compare vaccination rates or outcomes between different groups (e.g., regions, age groups, vaccine types).*
- 3. Correlation Analysis:** *Determine the strength and direction of relationships between variables, such as vaccine coverage and COVID-19 case rates, using correlation coefficients (e.g., Pearson, Spearman).*
- 4. Regression Analysis:** *Perform regression analysis to model and predict vaccination rates or other outcomes based on various factors like time, demographics, and vaccine supply.*
- 5. Time Series Analysis:** *Utilize time series methods like ARIMA or Exponential Smoothing to forecast vaccination trends and assess their impact on COVID-19 cases.*
- 6. Spatial Analysis:** *Employ spatial statistics and geographic information systems (GIS) to analyze the spatial distribution of vaccination rates and identify hotspots.*
- 7. Survival Analysis:** *If relevant, conduct survival analysis to study the time to vaccination completion or the duration of vaccine effectiveness.*
- 8. Chi-Square Analysis:** *Use chi-square tests to analyze categorical data, such as adverse events or vaccine preference by age group.*
- 9. Cluster Analysis:** *Apply clustering algorithms to group areas or individuals with similar vaccination patterns.*

**10. Bayesian Analysis:** Employ Bayesian methods to model vaccine efficacy, taking into account prior information and updating as new data becomes available.

**11. Machine Learning:** Utilize machine learning techniques for classification or prediction tasks related to vaccine distribution, efficacy, or adverse events.

**12. A/B Testing:** If applicable, conduct A/B testing to compare the impact of different vaccine distribution strategies or communication approaches.

**13. Meta-Analysis:** If multiple studies are available, perform a meta-analysis to synthesize findings from different sources.

**14. Statistical Significance:** Ensure results are statistically significant by setting appropriate significance levels (e.g.,  $p < 0.05$ ) and adjusting for multiple comparisons if needed.

**15. Data Visualization:** Visualize statistical results using charts, plots, and graphs to make the findings more accessible.

## **DESCRIPTIVE STATISTICS**

*Descriptive statistics provide a summary of your data, including measures like mean, median, and standard deviation.*

- `mean_vaccination = data['total_vaccinations'].mean()`

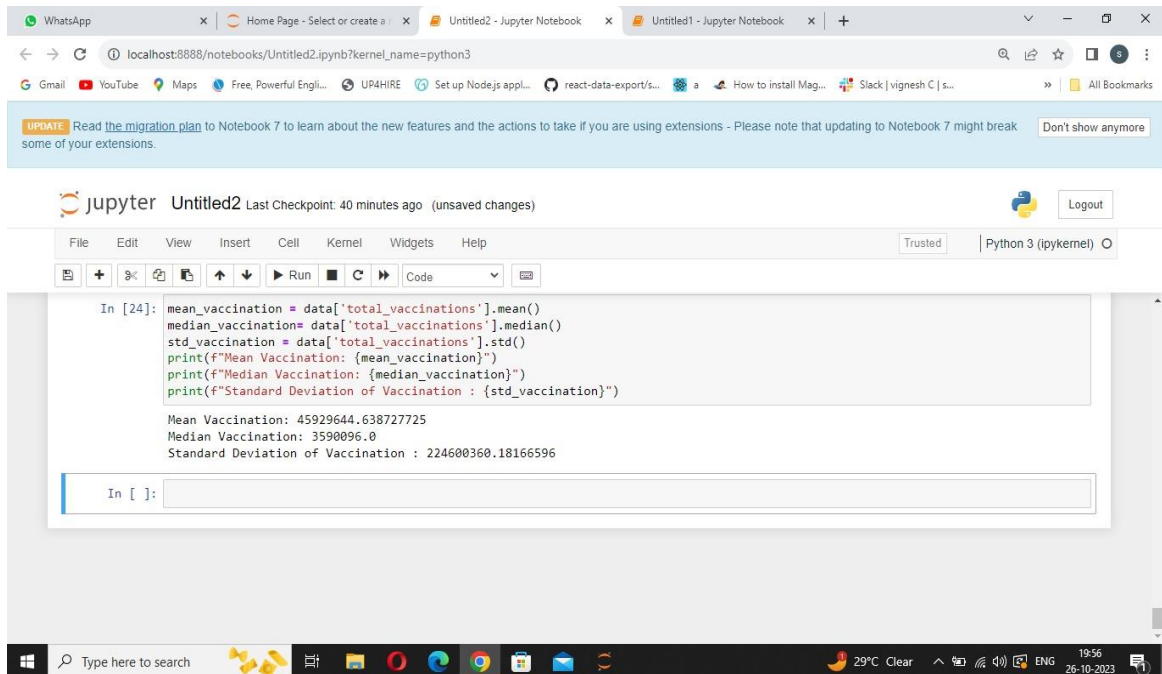
`median_vaccination = data['total_vaccinations'].median()`

`std_vaccination = data['total_vaccinations'].std()`

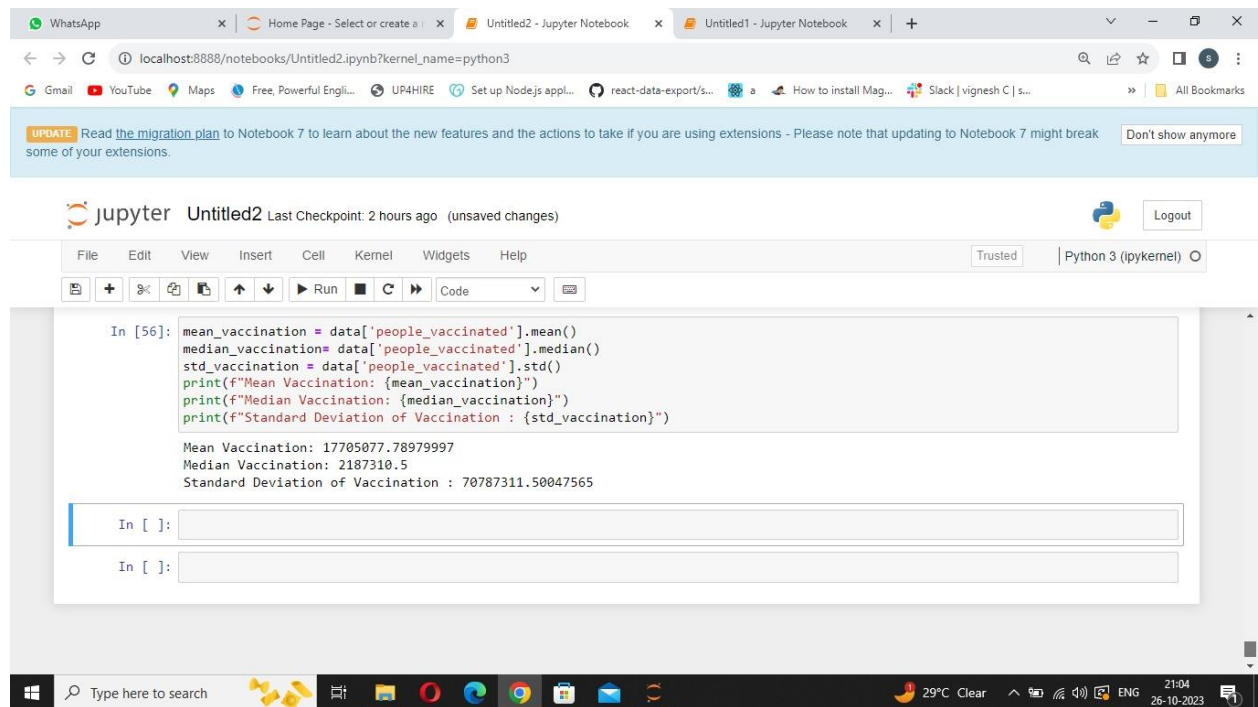
`print(f"Mean Vaccination: {mean_vaccination}")`

`print(f"Median Vaccination: {median_vaccination}")`

`print(f"Standard Deviation of Vaccination : {std_vaccination}")`

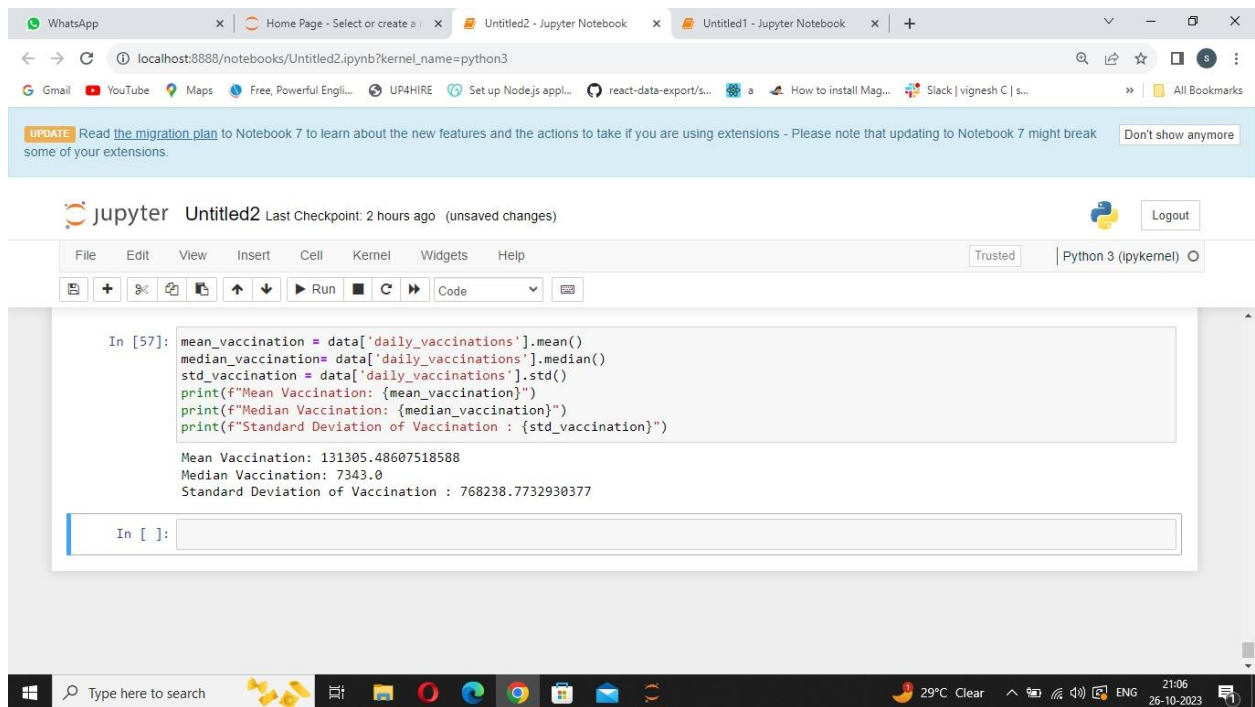


- *mean\_vaccination = data['people\_vaccinated'].mean()*
- median\_vaccination = data['people\_vaccinated'].median()*
- std\_vaccination = data['people\_vaccinated'].std()*
- print(f"Mean Vaccination: {mean\_vaccination}")*
- print(f"Median Vaccination: {median\_vaccination}")*
- print(f"Standard Deviation of Vaccination : {std\_vaccination}")*



- *mean\_vaccination = data['daily\_vaccinations'].mean()*  
*median\_vaccination= data['daily\_vaccinations'].median()*  
*std\_vaccination = data['daily\_vaccinations'].std()*  
*print(f"Mean Vaccination: {mean\_vaccination}")*  
*print(f"Median Vaccination: {median\_vaccination}")*  
*print(f"Standard Deviation of Vaccination : {std\_vaccination}")*





## ***HYPOTHESIS TESTING***

*from scipy import stats*

*country\_A = data[data['country'] == 'country\_A']['total\_vaccinations']*

*country\_B = data[data['country'] == 'country\_B']['total\_vaccinations']*

*t\_stat, p\_value = stats.ttest\_ind(country\_A, country\_B)*

*if p\_value < 0.05:*

*print("There is a significant difference between the two countries.")*

*else:*

*print("There is no significant difference between the two countries.")*

The screenshot shows a Jupyter Notebook titled 'Untitled2' running on a local server at localhost:8888. The notebook contains a code cell with the following Python code:

```
print('Standard Deviation of Vaccination : {std_vaccination}')
```

The output of the code is displayed below the cell:

```
Mean Vaccination: 45929644.638727725
Median Vaccination: 3590096.0
Standard Deviation of Vaccination : 224600360.18166596
```

Below the output, there is an interactive code cell labeled 'In [27]:' containing the following code:

```
from scipy import stats
country_A = data[data['country'] == 'country_A']['total_vaccinations']
country_B = data[data['country'] == 'country_B']['total_vaccinations']
t_stat, p_value = stats.ttest_ind(country_A, country_B)
if p_value < 0.05:
    print("There is a significant difference between the two countries.")
else:
    print("There is no significant difference between the two countries.")
```

The output of this code cell is:

```
There is no significant difference between the two countries.
```

The Jupyter Notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, code execution, and kernel management. The bottom status bar shows the system clock as 19:59 on 26-10-2023.

## VISUALISATION

*Visualizations play a crucial role in understanding and presenting COVID-19 vaccine analysis. Here are some types of visualizations you can use:*

- 1. Bar Charts:** *Show the distribution of different vaccine types administered in a region or over time.*
- 2. Line Charts:** *Display trends in vaccination rates over time, including first and second doses administered.*
- 3. Area Charts:** *Visualize the cumulative number of vaccines administered over time to track progress.*
- 4. Stacked Bar Charts:** *Illustrate the breakdown of vaccine distribution by age group or gender.*
- 5. Heatmaps:** *Depict vaccination rates across regions or countries using color gradients.*

**6. Choropleth Maps:** Show vaccination coverage by shading areas on a map, with darker colors indicating higher coverage.

**7. Scatter Plots:** Explore correlations between vaccination rates and variables like COVID-19 cases, GDP, or healthcare infrastructure.

**8. Histograms:** Examine the distribution of adverse events or vaccine side effects.

**9. Box Plots:** Display the distribution of vaccination rates or efficacy scores, showing median, quartiles, and outliers.

**10. Sankey Diagrams:** Visualize the flow of vaccines from manufacturers to distribution points.

**11. Network Diagrams:** Illustrate the connections between different stakeholders in the vaccine supply chain.

**12. Venn Diagrams:** Compare the overlap or differences between vaccinated populations by age, region, or other factors.

**13. Pareto Charts:** Identify the most significant contributors to vaccine distribution or adverse events.

**14. Pie Charts:** Show the proportion of the population that has received different vaccine types.

**15. Radar Charts:** Compare multiple attributes of different vaccines, such as efficacy, side effects, and cost.

**16. Dashboard:** Create an interactive dashboard with multiple visualizations for a comprehensive view of vaccine-related data.

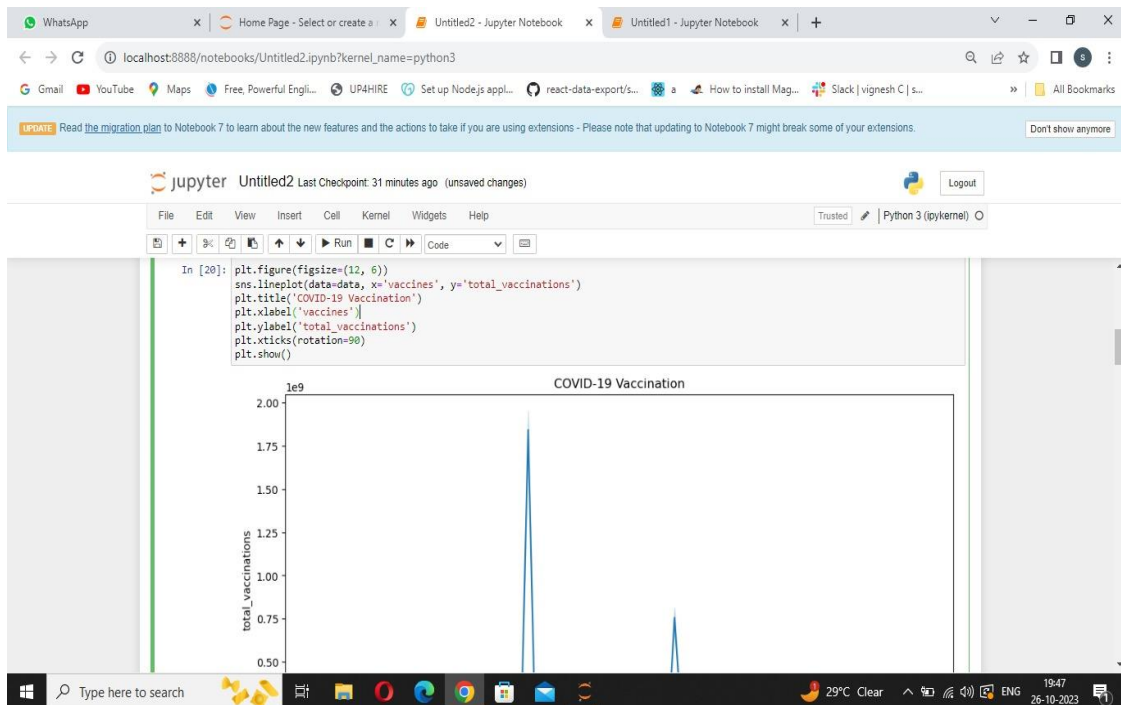
**17. Bubble Charts:** Represent data in a scatter plot format with bubbles of different sizes to show relationships between multiple variables.

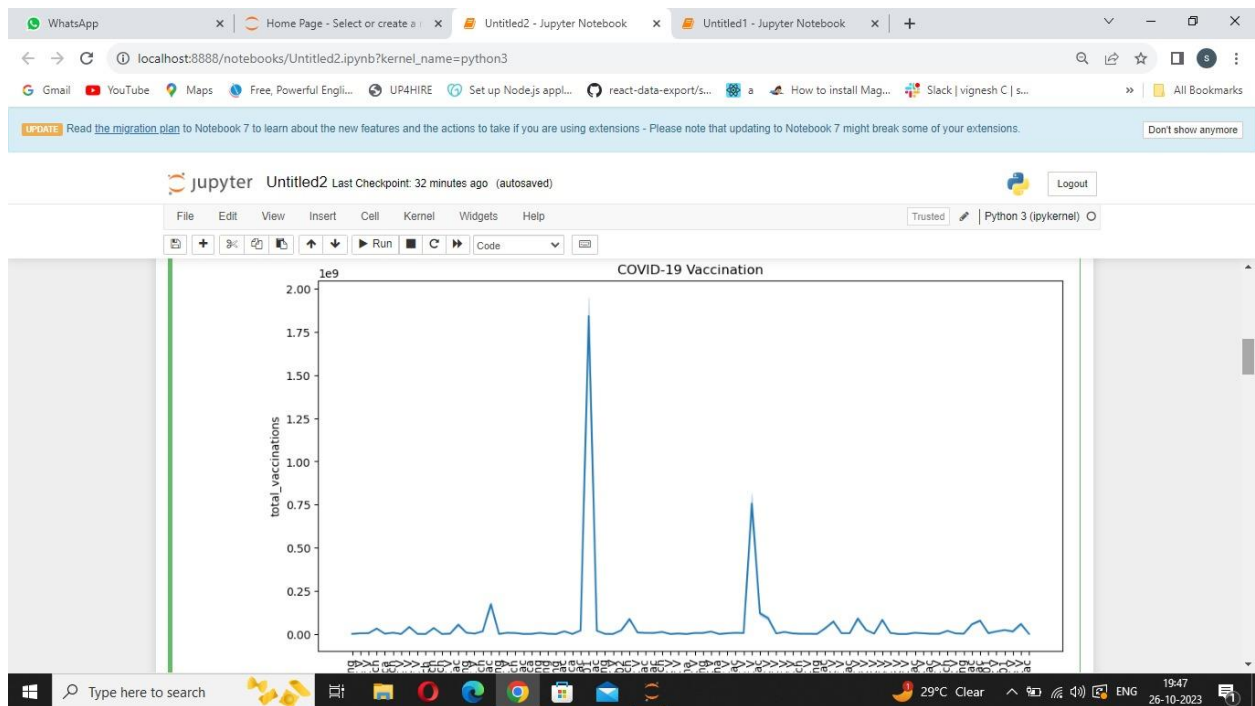
**18. Word Clouds:** Summarize public sentiment or discussions about vaccines on social media or news articles.

**19. Time Series Plots:** Analyze how vaccination rates change over time, including variations in different regions or age groups.

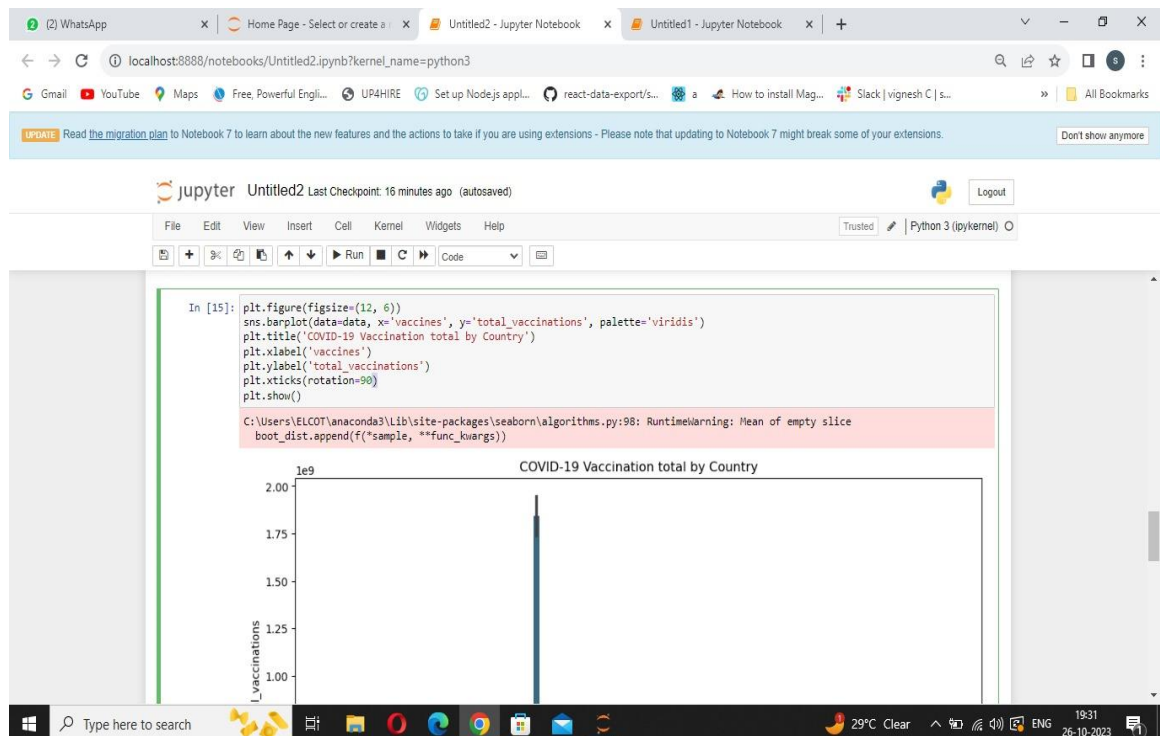
**20. Comparative Charts:** Compare vaccine distribution and coverage across different countries or regions.

- *create a line chart to utilize the progress of vaccination*

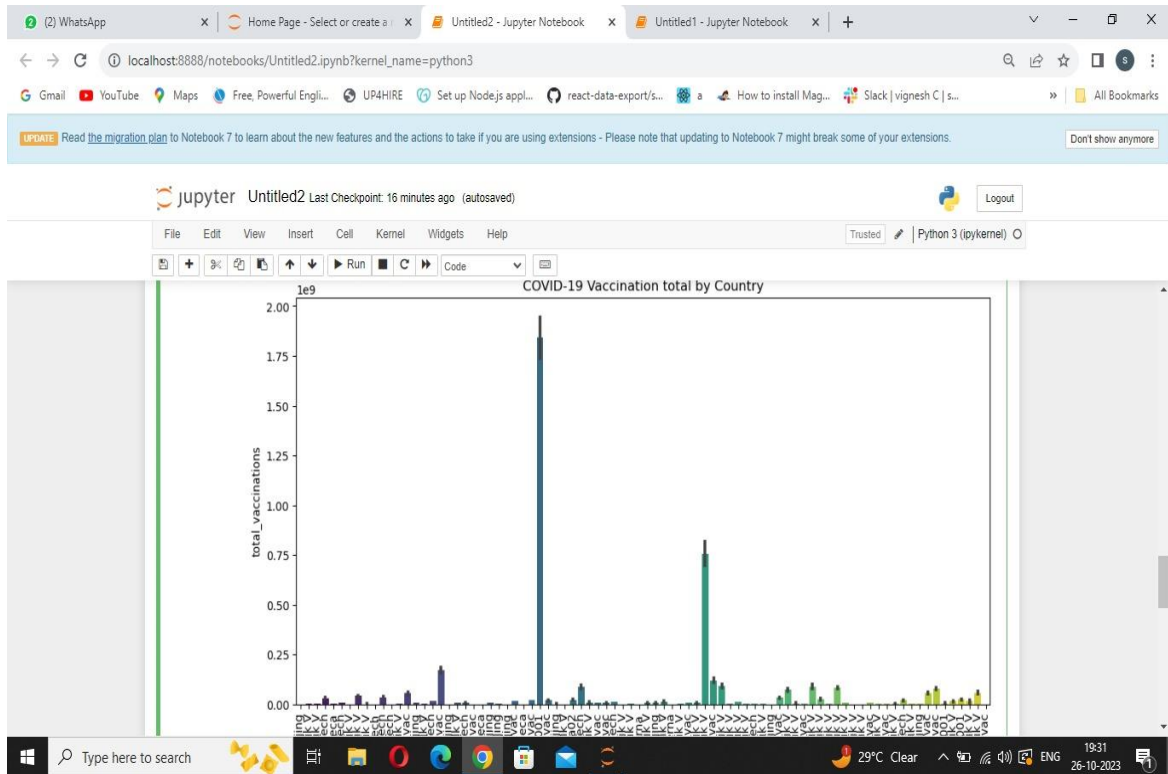




- *create a bar chart between total by country and total\_vaccination*







- **Create a pie chart to show the distribution of vaccine types**

