STEP 1: Import Libraries & Load the Dataset

```
import pandas as pdimport matplotlib.pyplot as pltimport seaborn as snsimport numpy as np
```

Explanation:

`pandas`: Used to load and manipulate structured data.

`matplotlib.pyplot`: A base Python plotting library (used for line, pie, and doughnut charts).

`seaborn`: A wrapper around matplotlib that provides prettier and more powerful charts.

`numpy`: Helps in numerical operations; used for calculations and creating shapes like a circle in the doughnut chart.

READ CSV

```
df = pd.read_csv('country_vaccinations.csv')
df.head()
```

- **EXPLANATION**
- `df`: A variable that holds your data in a **DataFrame** (a table).
- `pd.read_csv()`: Reads the CSV file and converts it into a DataFrame.
- `df.head()`: Shows the **first 5 rows** to help preview the structure.

STEP 2: Data Cleaning

1. Basic Data Info

```
df.info()
```

Gives a summary of:

Each column's **data type** (`float64`, `object`, etc.)

Non-null counts (helps detect missing values)

---

2. Check for Missing Values

```
df.isnull().sum()
```

- **EXPLANATION**
- Tells how many **null/missing values** are in each column.

---

3. Filling Missing Values

```
df['daily_vaccinations'] = df['daily_vaccinations'].fillna(0)
df['people_vaccinated'] = df['people_vaccinated'].fillna(0)
df['people_fully_vaccinated'] = df['people_fully_vaccinated'].fillna(0)
```

EXPLANATION

`fillna(0)` replaces missing values with 0

This is useful when missing means "no data for that day"

---

4. Convert Dates for Time Series

```
df['date'] = pd.to_datetime(df['date'])
```

EXPLANATION

Converts the `date` column from string to **datetime format**

This enables time-series plots

STEP 3: EDA + Visualizations

LINE CHART

```
india = df[df['country'] == 'India']
```

EXPLANATION

Filters rows only for India using **boolean indexing**.

```
plt.figure(figsize=(12,6))
plt.plot(india['date'], india['people_vaccinated'], label='People Vaccinated')
plt.plot(india['date'], india['people_fully_vaccinated'], label='Fully Vaccinated')
plt.title('Vaccination Progress in India Over Time')
plt.xlabel('Date')
plt.ylabel('Number of People')
plt.legend()
plt.grid(True)
```

Line Chart Details:

`plt.plot(x, y)`: Plots lines of vaccinated vs fully vaccinated people over time.

`figsize`: Sets the size of the chart.

`legend()`: Displays labels for each line.

`grid(True)`: Adds a light grid for better readability.

PIE CHART

```
latest = df[df['date'] == df['date'].max()]
top5 = latest.groupby('country')['total_vaccinations'].max().nlargest(5)
```

- **EXPLANATION**
- Filters data to include **only the latest date**
- Groups by `country` and finds the **top 5 countries** with the most vaccinations

```
plt.figure(figsize=(8,8))
plt.pie(top5, labels=top5.index, autopct='%1.1f%%', startangle=140)
plt.title('Top 5 Countries by Total Vaccinations')
```

Pie Chart Details:

`labels`: Country names

`autopct`: Shows % on the chart

`startangle`: Rotates the start angle for visual balance

DOUGHNUT CHART

```
vaccine_counts = df['vaccines'].value_counts().head(5)
```

- **EXPLANATION**

Counts how many times each **vaccine combination** is used.

```
plt.figure(figsize=(8,8))
wedges, texts, autotexts = plt.pie(vaccine_counts, labels=vaccine_counts.index,
autopct='%1.1f%%', startangle=140)
centre_circle = plt.Circle((0,0), 0.70, fc='white')
plt.gca().add_artist(centre_circle)
plt.title('Top 5 Vaccine Types Used (Doughnut Chart)')
```

Doughnut Chart Details:

Same as pie chart, but:

`plt.Circle((0,0), 0.70, fc='white')`: Draws a white circle in the middle

`add_artist()`: Adds the circle to the plot

BAR CHART - Last 30 Days

```
latest_india = india.sort_values('date').tail(30)
```

- **EXPLANATION**
- Sorts India data by date and keeps the last 30 days

```
plt.figure(figsize=(14,5))
sns.barplot(x='date', y='daily_vaccinations', data=latest_india, color='skyblue')
plt.xticks(rotation=45)
plt.title('Daily Vaccinations in India (Last 30 Days)')
```

Bar Chart Details:

- `barplot()`: Draws bars based on date and daily vaccinations
- `rotation=45`: Rotates x-axis labels for readability

HEATMAP - Correlation

```
df_num = df[['daily_vaccinations', 'people_vaccinated', 'people_fully_vaccinated']]
```

- **EXPLANATION**
- Selects only **numeric columns** for correlation

```
plt.figure(figsize=(8,6))
sns.heatmap(df_num.corr(), annot=True, cmap='YlGnBu')
plt.title('Correlation Between Vaccination Metrics')
```

Heatmap Details:

- `corr()`: Calculates correlation values (e.g. how strongly two variables are related)
- `annot=True`: Writes the values on the squares
- `cmap`: Changes the color scheme

Summary of Key Points

Feature	Code Used	Why It's Used
Date conversion	`pd.to_datetime()`	Enables time-based analysis
Filtering	`df[df['country'] == 'India']`	Analyze specific country
Pie chart	`plt.pie()`	Shows proportion data
Doughnut chart	`plt.Circle()`	Makes pie chart stylish
Line chart	`plt.plot()`	Track trends over time
Bar chart	`sns.barplot()`	Compare daily totals

Feature	Code Used	Why It's Used
Heatmap	sns.heatmap()	See variable relationships
Cleaning	fillna(), value_counts(), groupby()	Prepare for visuals