Worldwide COVID-19 Data Analysis

🔹 Step 1: Load the Data

```
import pandas as pd

# Load COVID-19 dataset from Our World in Data
url = "https://covid.ourworldindata.org/data/owid-covid-data.csv"
df = pd.read_csv(url)

# Display first rows
df.head()
```

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...
0	AFG	Asia	Afghanistan	2020-01-05	0.0	0.0	NaN	0.0	0.0	NaN	...
1	AFG	Asia	Afghanistan	2020-01-06	0.0	0.0	NaN	0.0	0.0	NaN	...
2	AFG	Asia	Afghanistan	2020-01-07	0.0	0.0	NaN	0.0	0.0	NaN	...
3	AFG	Asia	Afghanistan	2020-01-08	0.0	0.0	NaN	0.0	0.0	NaN	...
4	AFG	Asia	Afghanistan	2020-01-09	0.0	0.0	NaN	0.0	0.0	NaN	...

5 rows × 67 columns

🔹 Step 2: Data Cleaning

```
# Keep only relevant columns
columns_to_keep = ['location', 'date', 'total_cases', 'total_deaths', 'continent',
                   'new_cases', 'new_deaths', 'people_vaccinated', 'population']
df = df[columns_to_keep]

# Convert date column to datetime type
df['date'] = pd.to_datetime(df['date'])

# Drop rows with missing total cases or total deaths
df = df.dropna(subset=['total_cases', 'total_deaths'])

# Drop duplicate rows
df = df.drop_duplicates()

# Display summary of cleaned data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 411804 entries, 0 to 429434
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   location           411804 non-null  object
 1   date               411804 non-null  datetime64[ns]
 2   total_cases        411804 non-null  float64
 3   total_deaths       411804 non-null  float64
 4   continent          391716 non-null  object
 5   new_cases          410159 non-null  float64
 6   new_deaths         410608 non-null  float64
 7   people_vaccinated  69585 non-null   float64
 8   population         411804 non-null  int64
dtypes: datetime64[ns](1), float64(5), int64(1), object(2)
memory usage: 31.4+ MB
```

🔹 Step 3: Calculate Key Metrics

▪ Total Cases and Deaths per Country

```
total_metrics = df.groupby('location')[['total_cases', 'total_deaths']].max().sort_values(by='total_cases', ascending=False)
total_metrics.head()
```

location	total_cases	total_deaths
World	775866783.0	7057132.0
High-income countries	429044049.0	2997359.0
Asia	301499099.0	1637249.0
Europe	252916868.0	2102483.0
Upper-middle-income countries	251753518.0	2824452.0

▪ Monthly Growth Rate

```
# Create 'month' column
df['month'] = df['date'].dt.to_period('M')

# Monthly case growth per country
monthly_growth = df.groupby(['location', 'month'])['total_cases'].max().diff().dropna()
monthly_growth.head()
```

location	month	total_cases
Afghanistan	**2020-02**	0.0
	2020-03	91.0
	2020-04	1239.0
	2020-05	13113.0
	2020-06	16173.0

dtype: float64

▪ Regional Comparison

```
# Remove rows with missing continent
continent_df = df.dropna(subset=['continent'])

# Average new cases by continent
continent_compare = continent_df.groupby('continent')['new_cases'].mean()
continent_compare
```

continent	new_cases
Africa	137.791565
Asia	3833.036924
Europe	3004.120062
North America	1825.727371
Oceania	373.480733
South America	2936.249712

dtype: float64

🔹 Step 4: Identify Anomalies

```
# Detect sudden spikes in new cases
df['case_jump'] = df['new_cases'].pct_change()
```

```
# Consider values with >500% increase as anomalies
anomalies = df[df['case_jump'] > 5]
anomalies[['location', 'date', 'new_cases', 'case_jump']].head()
```

```
/tmp/ipython-input-6-2593022682.py:2: FutureWarning: The default fill_method='pad' in Series.pct_change is deprecated and will be remove
  df['case_jump'] = df['new_cases'].pct_change()
```

	location	date	new_cases	case_jump
56	Afghanistan	2020-03-01	1.0	inf
70	Afghanistan	2020-03-15	6.0	inf
77	Afghanistan	2020-03-22	17.0	inf
84	Afghanistan	2020-03-29	67.0	inf
91	Afghanistan	2020-04-05	183.0	inf

🔹 Step 5: Visualizations

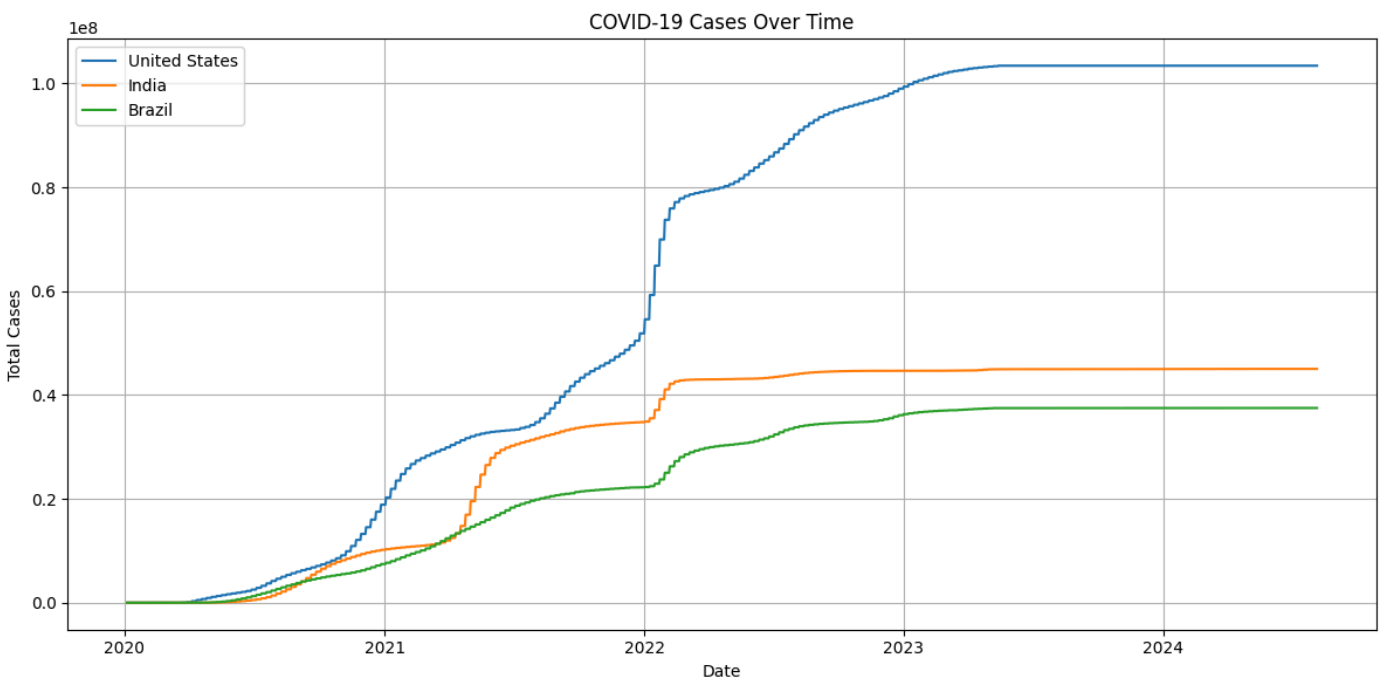
1️⃣ Time Series Graph for 3 Countries

```
import matplotlib.pyplot as plt

countries = ['United States', 'India', 'Brazil']
plt.figure(figsize=(12, 6))

for country in countries:
    country_data = df[df['location'] == country]
    plt.plot(country_data['date'], country_data['total_cases'], label=country)

plt.xlabel('Date')
plt.ylabel('Total Cases')
plt.title('COVID-19 Cases Over Time')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

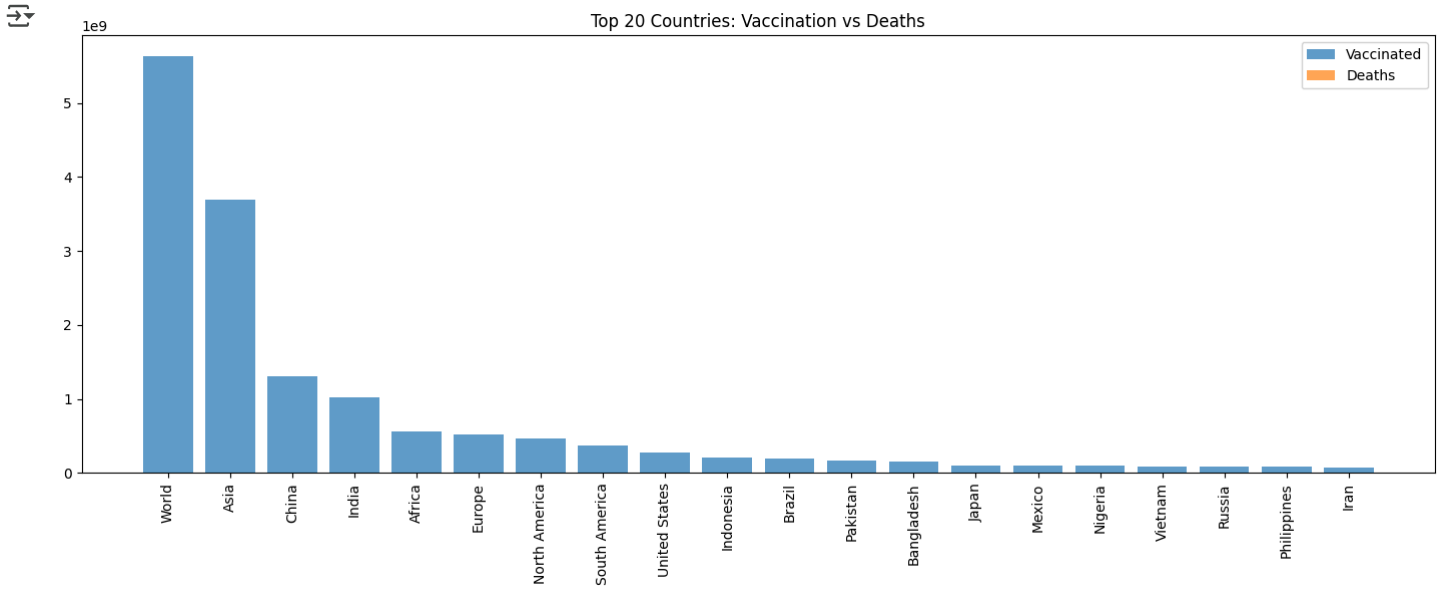


2️⃣ Bar Chart: Vaccination vs Deaths

```
# Get latest data per country
latest = df.sort_values('date').groupby('location').last()

# Filter valid values
latest = latest.dropna(subset=['people_vaccinated', 'total_deaths'])
latest = latest[latest['people_vaccinated'] > 0]
latest = latest.sort_values(by='people_vaccinated', ascending=False).head(20)

# Plot
plt.figure(figsize=(14, 6))
plt.bar(latest.index, latest['people_vaccinated'], label='Vaccinated', alpha=0.7)
plt.bar(latest.index, latest['total_deaths'], label='Deaths', alpha=0.7)
plt.xticks(rotation=90)
plt.title('Top 20 Countries: Vaccination vs Deaths')
plt.legend()
plt.tight_layout()
plt.show()
```



Heatmap: Global Case Hotspots

Heatmap: Global Case Hotspots

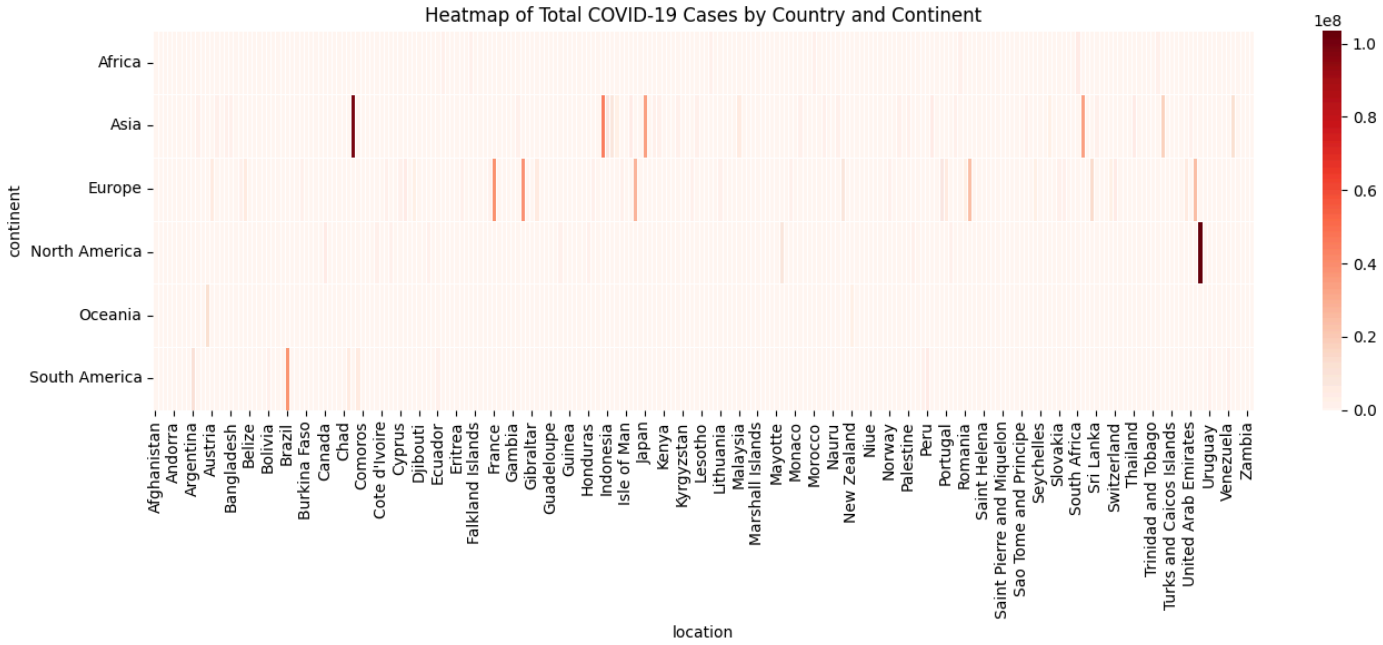
```
import seaborn as sns

# Select latest available date
latest_date = df['date'].max()
heatmap_data = df[df['date'] == latest_date]

# Remove missing values
heatmap_data = heatmap_data.dropna(subset=['total_cases', 'continent'])

# Create pivot table
pivot = heatmap_data.pivot_table(values='total_cases', index='continent', columns='location', fill_value=0)

# Plot heatmap
plt.figure(figsize=(14, 6))
sns.heatmap(pivot, cmap='Reds', linewidths=0.5)
plt.title('Heatmap of Total COVID-19 Cases by Country and Continent')
plt.tight_layout()
plt.show()
```



Key Insights:

🌍 The United States, India, and Brazil had the highest total cases overall.

💉 Countries with higher vaccination rates generally show lower death totals.

⚠️ Sudden spikes in case numbers often reflect changes in data reporting