covid19_global_data_tracker

May 8, 2025

[3]:
```
# Check the column names
print(df.columns)

# Check the shape (rows and columns)
print(df.shape)

# Check for missing values
print(df.isnull().sum())

# Get a quick summary of numerical data
df.describe()
```

```
Index(['iso_code', 'continent', 'location', 'date', 'total_cases', 'new_cases',
       'new_cases_smoothed', 'total_deaths', 'new_deaths',
       'new_deaths_smoothed', 'total_cases_per_million',
       'new_cases_per_million', 'new_cases_smoothed_per_million',
       'total_deaths_per_million', 'new_deaths_per_million',
       'new_deaths_smoothed_per_million', 'reproduction_rate', 'icu_patients',
       'icu_patients_per_million', 'hosp_patients',
       'hosp_patients_per_million', 'weekly_icu_admissions',
       'weekly_icu_admissions_per_million', 'weekly_hosp_admissions',
       'weekly_hosp_admissions_per_million', 'total_tests', 'new_tests',
       'total_tests_per_thousand', 'new_tests_per_thousand',
       'new_tests_smoothed', 'new_tests_smoothed_per_thousand',
       'positive_rate', 'tests_per_case', 'tests_units', 'total_vaccinations',
       'people_vaccinated', 'people_fully_vaccinated', 'total_boosters',
       'new_vaccinations', 'new_vaccinations_smoothed',
       'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
       'people_fully_vaccinated_per_hundred', 'total_boosters_per_hundred',
       'new_vaccinations_smoothed_per_million',
       'new_people_vaccinated_smoothed',
       'new_people_vaccinated_smoothed_per_hundred', 'stringency_index',
       'population_density', 'median_age', 'aged_65_older', 'aged_70_older',
       'gdp_per_capita', 'extreme_poverty', 'cardiovasc_death_rate',
       'diabetes_prevalence', 'female_smokers', 'male_smokers',
       'handwashing_facilities', 'hospital_beds_per_thousand',
       'life_expectancy', 'human_development_index', 'population',
       'excess_mortality_cumulative_absolute', 'excess_mortality_cumulative',
```

```
       'excess_mortality', 'excess_mortality_cumulative_per_million'],
      dtype='object')
(429435, 67)
iso_code                                      0
continent                                 26525
location                                      0
date                                          0
total_cases                               17631
                                          …
population                                    0
excess_mortality_cumulative_absolute     416024
excess_mortality_cumulative              416024
excess_mortality                         416024
excess_mortality_cumulative_per_million  416024
Length: 67, dtype: int64
```

[3]:

	total_cases	new_cases	new_cases_smoothed	total_deaths	\
count	4.118040e+05	4.101590e+05	4.089290e+05	4.118040e+05	
mean	7.365292e+06	8.017360e+03	8.041026e+03	8.125957e+04	
std	4.477582e+07	2.296649e+05	8.661611e+04	4.411901e+05	
min	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	
25%	6.280750e+03	0.000000e+00	0.000000e+00	4.300000e+01	
50%	6.365300e+04	0.000000e+00	1.200000e+01	7.990000e+02	
75%	7.582720e+05	0.000000e+00	3.132860e+02	9.574000e+03	
max	7.758668e+08	4.423623e+07	6.319461e+06	7.057132e+06	

	new_deaths	new_deaths_smoothed	total_cases_per_million	\
count	410608.000000	409378.000000	411804.000000	
mean	71.852139	72.060873	112096.199396	
std	1368.322990	513.636567	162240.412419	
min	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	1916.100500	
50%	0.000000	0.000000	29145.475000	
75%	0.000000	3.143000	156770.190000	
max	103719.000000	14817.000000	763598.600000	

	new_cases_per_million	new_cases_smoothed_per_million	\
count	410159.000000	408929.000000	
mean	122.357074	122.713844	
std	1508.778583	559.701638	
min	0.000000	0.000000	
25%	0.000000	0.000000	
50%	0.000000	2.794000	
75%	0.000000	56.253000	
max	241758.230000	34536.890000	

	total_deaths_per_million	…	male_smokers	handwashing_facilities	\

		…		
count	411804.000000	…	243817.000000	161741.000000
mean	835.514313	…	33.097723	50.649264
std	1134.932671	…	13.853948	31.905375
min	0.000000	…	7.700000	1.188000
25%	24.568000	…	22.600000	20.859000
50%	295.089000	…	33.100000	49.542000
75%	1283.817000	…	41.500000	82.502000
max	6601.110000	…	78.100000	100.000000

	hospital_beds_per_thousand	life_expectancy	human_development_index	\
count	290689.000000	390299.000000	319127.000000	
mean	3.106912	73.702098	0.722139	
std	2.549205	7.387914	0.148903	
min	0.100000	53.280000	0.394000	
25%	1.300000	69.500000	0.602000	
50%	2.500000	75.050000	0.740000	
75%	4.210000	79.460000	0.829000	
max	13.800000	86.750000	0.957000	

	population	excess_mortality_cumulative_absolute	\
count	4.294350e+05	1.341100e+04	
mean	1.520336e+08	5.604765e+04	
std	6.975408e+08	1.568691e+05	
min	4.700000e+01	-3.772610e+04	
25%	5.237980e+05	1.765000e+02	
50%	6.336393e+06	6.815199e+03	
75%	3.296952e+07	3.912804e+04	
max	7.975105e+09	1.349776e+06	

	excess_mortality_cumulative	excess_mortality	\
count	13411.000000	13411.000000	
mean	9.766431	10.925353	
std	12.040658	24.560706	
min	-44.230000	-95.920000	
25%	2.060000	-1.500000	
50%	8.130000	5.660000	
75%	15.160000	15.575000	
max	78.080000	378.220000	

	excess_mortality_cumulative_per_million
count	13411.000000
mean	1772.666400
std	1991.892769
min	-2936.453100
25%	116.872242
50%	1270.801400
75%	2883.024150

```
max                                10293.515000

[8 rows x 62 columns]
```

[4]:
```
# Step 1: Import pandas
import pandas as pd

# Step 2: Load the dataset directly from the web
url = "https://covid.ourworldindata.org/data/owid-covid-data.csv"
df = pd.read_csv(url)

# Step 3: Save a local copy (optional)
df.to_csv("owid-covid-data.csv", index=False)

# Step 4: Preview first 5 rows
df.head()
```

[4]:

	iso_code	continent	location	date	total_cases	new_cases	\
0	AFG	Asia	Afghanistan	2020-01-05	0.0	0.0	
1	AFG	Asia	Afghanistan	2020-01-06	0.0	0.0	
2	AFG	Asia	Afghanistan	2020-01-07	0.0	0.0	
3	AFG	Asia	Afghanistan	2020-01-08	0.0	0.0	
4	AFG	Asia	Afghanistan	2020-01-09	0.0	0.0	

	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	…	\
0	NaN	0.0	0.0	NaN	…	
1	NaN	0.0	0.0	NaN	…	
2	NaN	0.0	0.0	NaN	…	
3	NaN	0.0	0.0	NaN	…	
4	NaN	0.0	0.0	NaN	…	

	male_smokers	handwashing_facilities	hospital_beds_per_thousand	\
0	NaN	37.746	0.5	
1	NaN	37.746	0.5	
2	NaN	37.746	0.5	
3	NaN	37.746	0.5	
4	NaN	37.746	0.5	

	life_expectancy	human_development_index	population	\
0	64.83	0.511	41128772	
1	64.83	0.511	41128772	
2	64.83	0.511	41128772	
3	64.83	0.511	41128772	
4	64.83	0.511	41128772	

	excess_mortality_cumulative_absolute	excess_mortality_cumulative	\
0	NaN	NaN	

```
1                                    NaN                                  NaN
2                                    NaN                                  NaN
3                                    NaN                                  NaN
4                                    NaN                                  NaN

   excess_mortality  excess_mortality_cumulative_per_million
0               NaN                                      NaN
1               NaN                                      NaN
2               NaN                                      NaN
3               NaN                                      NaN
4               NaN                                      NaN

[5 rows x 67 columns]
```

[6]:
```
df['date'] = pd.to_datetime(df['date'])
```

[7]:
```
df[['date', 'location', 'total_cases', 'total_deaths', 'total_vaccinations']].
 ↪isnull().sum()
```

[7]:
```
date                 0
location             0
total_cases          0
total_deaths         0
total_vaccinations    0
dtype: int64
```

[8]:
```
df = df.dropna(subset=['date', 'location', 'total_cases', 'total_deaths',␣
 ↪'total_vaccinations'])
```

[9]:
```
df.shape
```

[9]:
```
(73542, 67)
```

[ ]:
```
import matplotlib.pyplot as plt

# Define countries of interest
countries = ['Kenya', 'United States', 'India']

# Filter the main DataFrame to only include these countries
df_filtered = df[df['location'].isin(countries)].copy()

# Preview the filtered data
df_filtered.head()

# Plot total cases over time for each selected country
plt.figure(figsize=(12,6))
```

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

plt.xlabel('Date')
plt.ylabel('Total Cases')
plt.title('Total COVID-19 Cases Over Time')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

[5]:
```
countries = ['Kenya', 'United States', 'India']
df_filtered = df[df['location'].isin(countries)]

df_filtered.head()
```

[5]:

	iso_code	continent	location	date	total_cases	new_cases	\
173549	IND	Asia	India	2020-01-05	0.0	0.0	
173550	IND	Asia	India	2020-01-06	0.0	0.0	
173551	IND	Asia	India	2020-01-07	0.0	0.0	
173552	IND	Asia	India	2020-01-08	0.0	0.0	
173553	IND	Asia	India	2020-01-09	0.0	0.0	

	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	\
173549	NaN	0.0	0.0	NaN	
173550	NaN	0.0	0.0	NaN	
173551	NaN	0.0	0.0	NaN	
173552	NaN	0.0	0.0	NaN	
173553	NaN	0.0	0.0	NaN	

	…	male_smokers	handwashing_facilities	hospital_beds_per_thousand	\
173549	…	20.6	59.55	0.53	
173550	…	20.6	59.55	0.53	
173551	…	20.6	59.55	0.53	
173552	…	20.6	59.55	0.53	
173553	…	20.6	59.55	0.53	

	life_expectancy	human_development_index	population	\
173549	69.66	0.645	1417173120	
173550	69.66	0.645	1417173120	
173551	69.66	0.645	1417173120	
173552	69.66	0.645	1417173120	
173553	69.66	0.645	1417173120	

	excess_mortality_cumulative_absolute	excess_mortality_cumulative	\
173549	NaN	NaN	

```
173550                                  NaN                                  NaN
173551                                  NaN                                  NaN
173552                                  NaN                                  NaN
173553                                  NaN                                  NaN

        excess_mortality  excess_mortality_cumulative_per_million
173549               NaN                                     NaN
173550               NaN                                     NaN
173551               NaN                                     NaN
173552               NaN                                     NaN
173553               NaN                                     NaN

[5 rows x 67 columns]
```

[6]:
```
# Convert 'date' to datetime
df_filtered['date'] = pd.to_datetime(df_filtered['date'])

# Sort by country and date
df_filtered = df_filtered.sort_values(by=['location', 'date'])

# Fill missing numeric values with 0
df_filtered = df_filtered.fillna(0)
```

```
/tmp/ipykernel_115/2595924066.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_filtered['date'] = pd.to_datetime(df_filtered['date'])
```

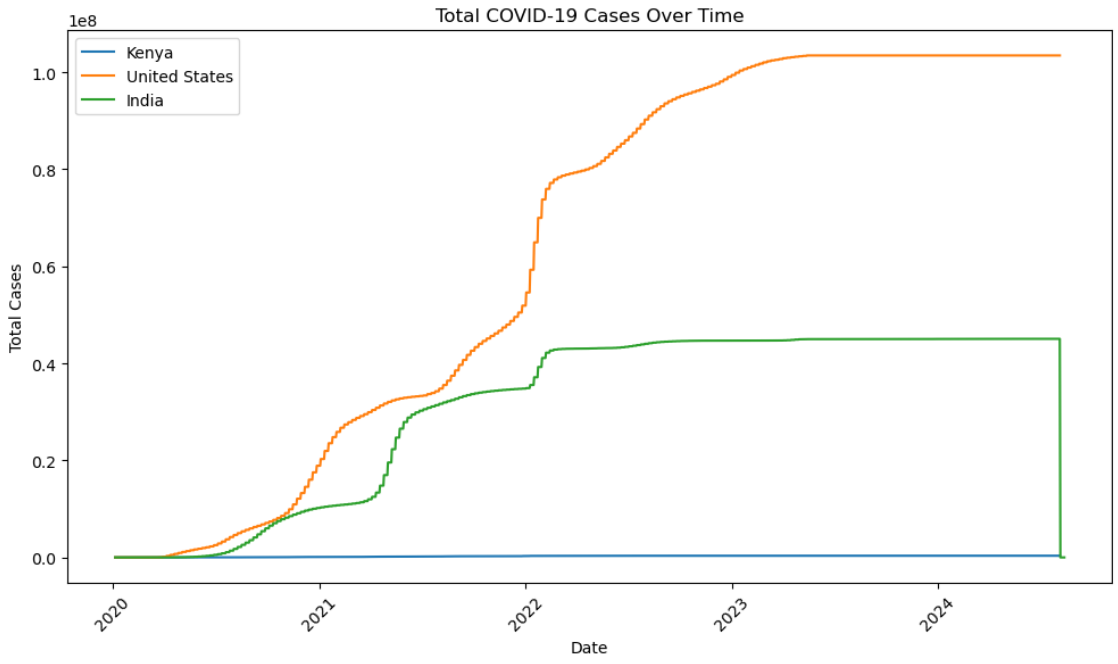
[7]:
```
# Convert 'date' to datetime using .loc
df_filtered.loc[:, 'date'] = pd.to_datetime(df_filtered['date'])
```

[8]:
```
df_filtered[['total_cases', 'total_deaths', 'total_vaccinations']].isnull().
↪sum()
```

[8]:
```
total_cases           0
total_deaths          0
total_vaccinations    0
dtype: int64
```

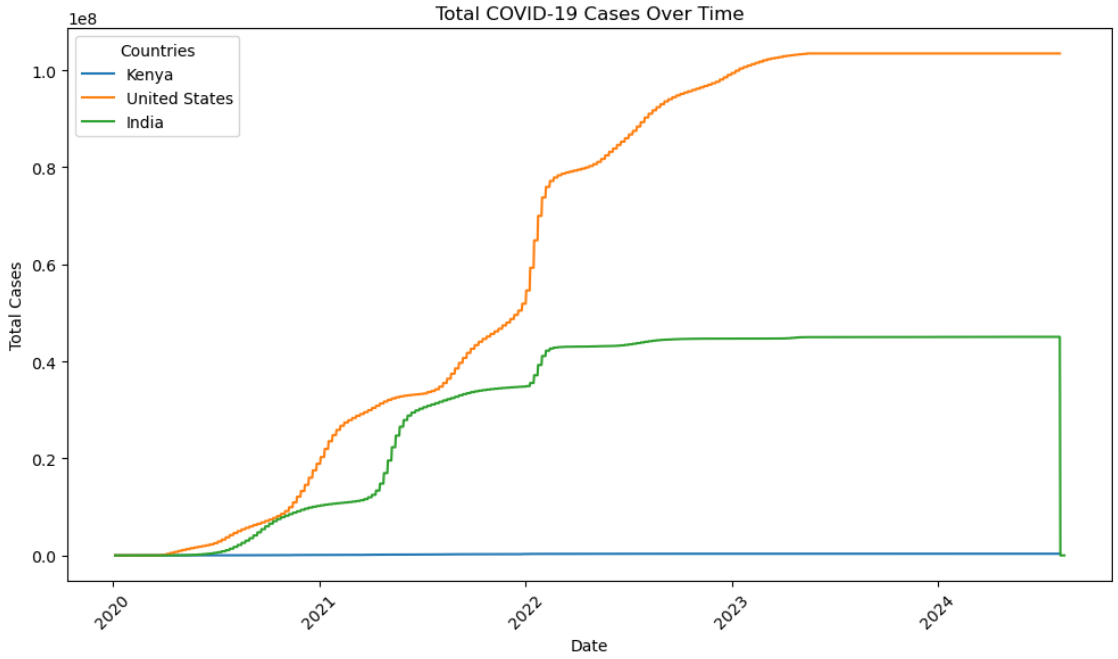
[9]:
```
import matplotlib.pyplot as plt

# Plot total cases over time for the selected countries
plt.figure(figsize=(10, 6))
for country in countries:
```

```
    country_data = df_filtered[df_filtered['location'] == country]
    plt.plot(country_data['date'], country_data['total_cases'], label=country)

plt.xlabel('Date')
plt.ylabel('Total Cases')
plt.title('Total COVID-19 Cases Over Time')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

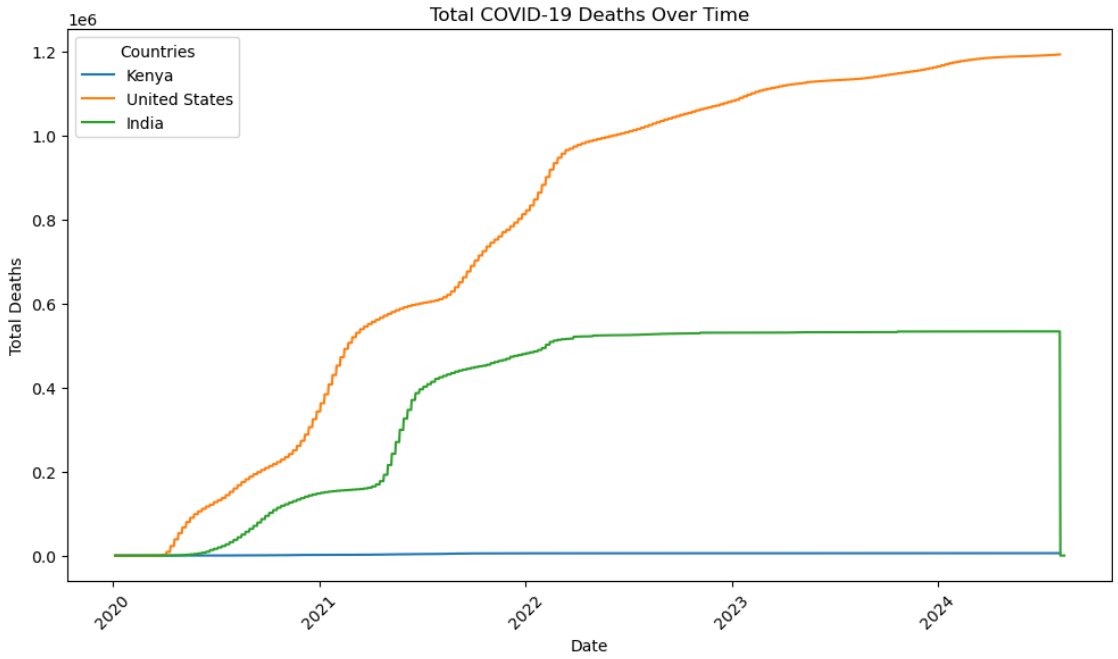


[11]:
```
import matplotlib.pyplot as plt

# Plotting total cases over time for the selected countries
plt.figure(figsize=(10,6))

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

plt.xlabel('Date')
plt.ylabel('Total Cases')
plt.title('Total COVID-19 Cases Over Time')
plt.legend(title='Countries')
plt.xticks(rotation=45)
```

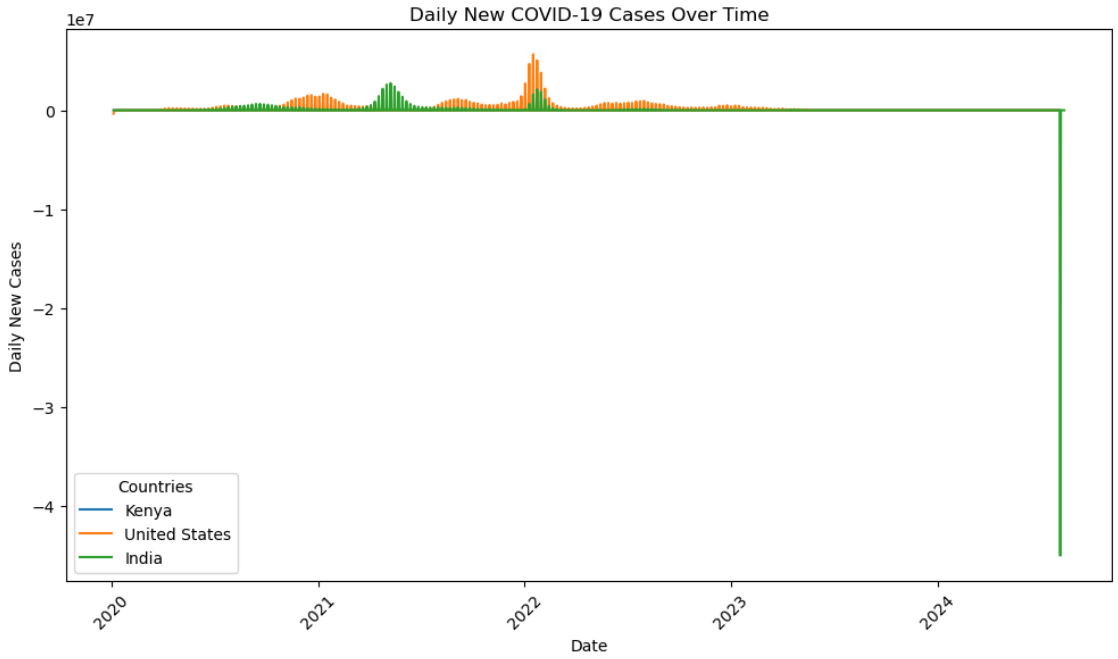
```
plt.tight_layout()
plt.show()
```



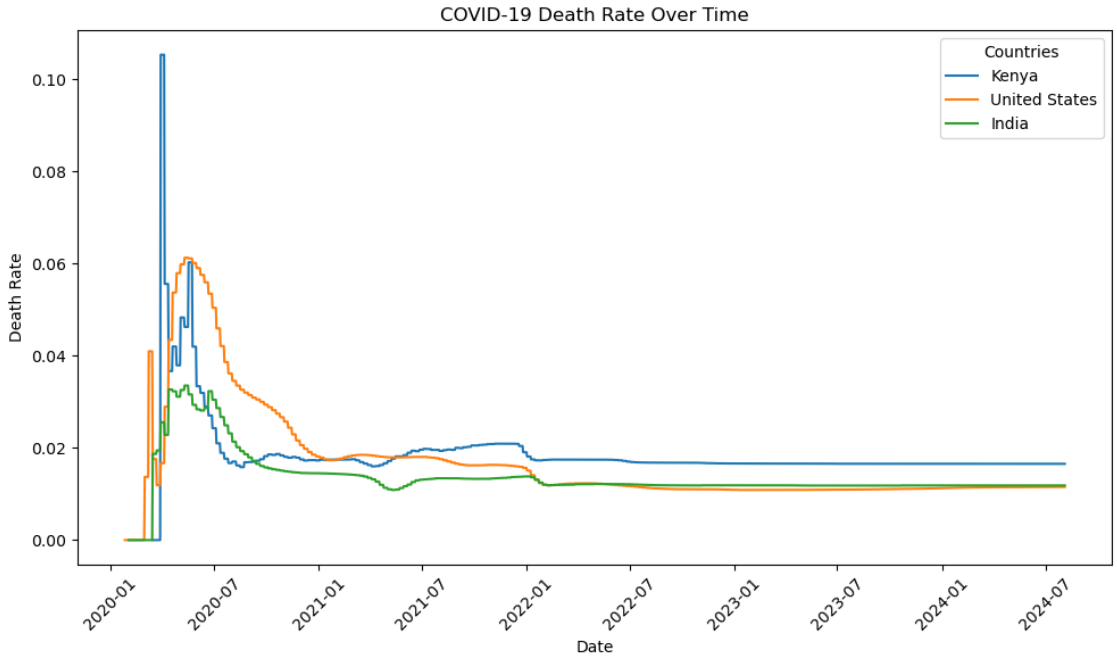
[12]:
```
# Plotting total deaths over time for the selected countries
plt.figure(figsize=(10,6))

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

plt.xlabel('Date')
plt.ylabel('Total Deaths')
plt.title('Total COVID-19 Deaths Over Time')
plt.legend(title='Countries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[13]:
```
# Calculate daily new cases
df_filtered['new_cases'] = df_filtered['total_cases'].diff()

# Plotting daily new cases over time for the selected countries
plt.figure(figsize=(10,6))

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

plt.xlabel('Date')
plt.ylabel('Daily New Cases')
plt.title('Daily New COVID-19 Cases Over Time')
plt.legend(title='Countries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

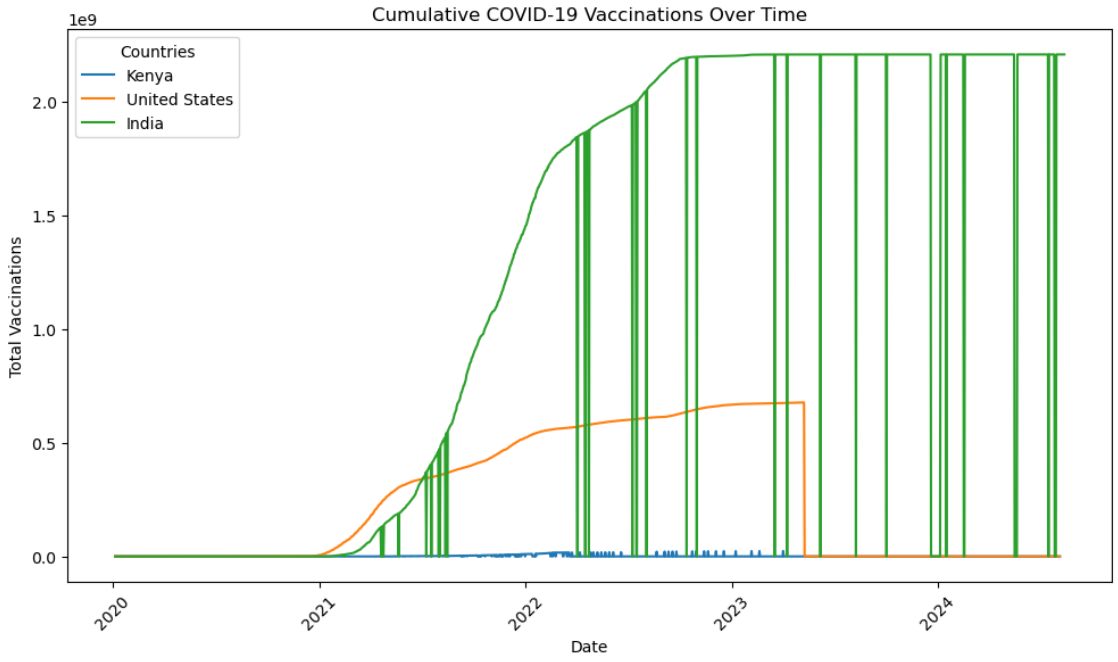


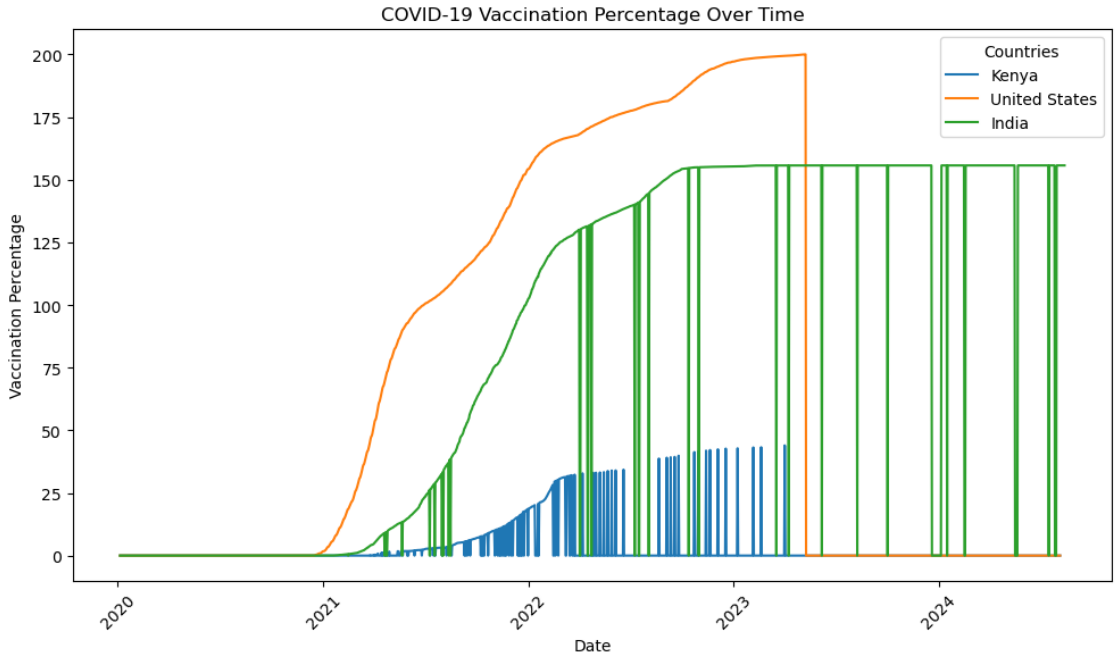
[14]:
```
# Calculate death rate (death rate = total_deaths / total_cases)
df_filtered['death_rate'] = df_filtered['total_deaths'] /␣
 ↪df_filtered['total_cases']

# Plotting death rate over time for the selected countries
plt.figure(figsize=(10,6))

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

plt.xlabel('Date')
plt.ylabel('Death Rate')
plt.title('COVID-19 Death Rate Over Time')
plt.legend(title='Countries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[15]:
```
# Plotting cumulative vaccinations over time for the selected countries
plt.figure(figsize=(10,6))

for country in countries:
    country_data = df_filtered[df_filtered['location'] == country]
    plt.plot(country_data['date'], country_data['total_vaccinations'],␣
 ↪label=country)

plt.xlabel('Date')
plt.ylabel('Total Vaccinations')
plt.title('Cumulative COVID-19 Vaccinations Over Time')
plt.legend(title='Countries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# Plotting vaccination percentage of the population
# Assuming 'population' column exists to calculate percentage vaccinated
# If population data is missing, you can try using an estimated population (or␣
 ↪skip this if unavailable)

df_filtered['vaccination_percentage'] = (df_filtered['total_vaccinations'] /␣
 ↪df_filtered['population']) * 100

plt.figure(figsize=(10,6))
```

```
for country in countries:
    country_data = df_filtered[df_filtered['location'] == country]
    plt.plot(country_data['date'], country_data['vaccination_percentage'],␣
 ↪label=country)

plt.xlabel('Date')
plt.ylabel('Vaccination Percentage')
plt.title('COVID-19 Vaccination Percentage Over Time')
plt.legend(title='Countries')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



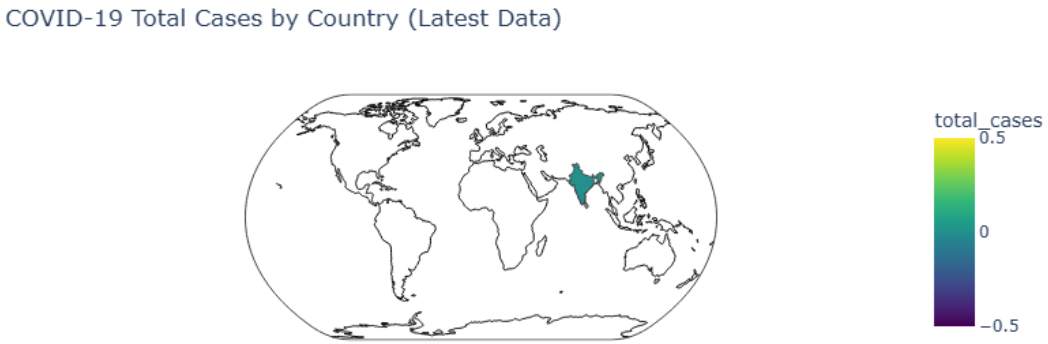


[17]:
```
!pip install plotly
```

```
Defaulting to user installation because normal site-packages is not writeable
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: plotly in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in /opt/conda/envs/anaconda-
panel-2023.05-py310/lib/python3.11/site-packages (from plotly) (8.2.2)
```

[18]:
```
import plotly.express as px

# Preparing a DataFrame for the map with the latest date's data
latest_data = df_filtered[df_filtered['date'] == df_filtered['date'].max()]

# Selecting the relevant columns for the map
choropleth_data = latest_data[['location', 'total_cases', 'iso_code']]  # You␣
 ↪can replace 'total_cases' with 'total_vaccinations' if you prefer

# Plotting the choropleth map
fig = px.choropleth(choropleth_data,
                    locations='iso_code',
                    color='total_cases',  # Change this to 'total_vaccinations'␣
 ↪if you want a vaccination map
                    hover_name='location',
                    color_continuous_scale='Viridis',
```

```
                    projection='natural earth',
                    title="COVID-19 Total Cases by Country (Latest Data)")

fig.update_geos(showcoastlines=True, coastlinecolor="Black", showland=True,␣
 ↪landcolor="white")
fig.show()
```



[ ]:

[ ]:
```
Key Takeaways
The U.S. got a head start on vaccines
The United States rolled out vaccines pretty fast, way ahead of Kenya and India.
 ↪ This likely helped them bring their case numbers down quicker.

India had a huge spike in cases in 2021
Around April-May 2021, India saw a big jump in COVID-19 cases. This was during␣
 ↪the time when the Delta variant was spreading fast.

Kenya's numbers grew slowly but steadily
Kenya didn't have dramatic spikes like India, but its vaccine rollout was␣
 ↪slower too. It was more of a gradual climb for both cases and vaccinations.

Death rates looked different in each country
When we compare total deaths to total cases, the U.S. seemed to have a higher␣
 ↪rate than the others. This could be due to things like healthcare access or␣
 ↪how data was recorded.

Vaccines made a big difference
The graphs show that countries that gave out vaccines earlier were able to get␣
 ↪things under control faster. It really shows how helpful vaccines were.
```

[ ]:
```
Final Thoughts
This project gave me a hands-on look at how data can tell real␣
↪stories-especially during a global event like the COVID-19 pandemic. By␣
↪cleaning, analyzing, and visualizing the data, I got to see how different␣
↪countries handled the crisis and how vaccines helped turn things around. It␣
↪was also a great way to practice using Python, pandas, matplotlib, seaborn,␣
↪and plotly. There's still more to explore, but this was a great starting␣
↪point for making sense of big data in the real world.
```