Untitled3

May 12, 2025

[2]:
```
import pandas as pd
df=pd.read_csv("owid-covid-data.csv")
df
```

[2]:

	iso_code	continent	location	date	total_cases	\
0	AFG	Asia	Afghanistan	2020-01-03	NaN	
1	AFG	Asia	Afghanistan	2020-01-04	NaN	
2	AFG	Asia	Afghanistan	2020-01-05	NaN	
3	AFG	Asia	Afghanistan	2020-01-06	NaN	
4	AFG	Asia	Afghanistan	2020-01-07	NaN	
…	…	…	…	…	…	
62567	COL	South America	Colombia	2020-08-10	376870.0	
62568	COL	South America	Colombia	2020-08-11	387481.0	
62569	COL	South America	Colombia	2020-08-12	397623.0	
62570	COL	South America	Colombia	2020-08-13	410453.0	
62571	COL	South America	Colombia	2020-08-14	422519.0	

	new_cases	new_cases_smoothed	total_deaths	new_deaths	\
0	0.0	NaN	NaN	0.0	
1	0.0	NaN	NaN	0.0	
2	0.0	NaN	NaN	0.0	
3	0.0	NaN	NaN	0.0	
4	0.0	NaN	NaN	0.0	
…	…	…	…	…	
62567	9674.0	10098.429	12540.0	290.0	
62568	10611.0	9975.714	12842.0	302.0	
62569	10142.0	9967.571	13154.0	312.0	
62570	12830.0	10782.000	13475.0	321.0	
62571	12066.0	10972.143	13837.0	362.0	

	new_deaths_smoothed	…	male_smokers	handwashing_facilities	\
0	NaN	…	NaN	37.746	
1	NaN	…	NaN	37.746	
2	NaN	…	NaN	37.746	
3	NaN	…	NaN	37.746	
4	NaN	…	NaN	37.746	
…	…	…	…	…	

```
62567              315.714  …          13.5                  65.386
62568              313.143  …          13.5                  65.386
62569              305.286  …          13.5                  65.386
62570              308.571  …          13.5                  65.386
62571              316.143  …           NaN                     NaN

       hospital_beds_per_thousand  life_expectancy  human_development_index  \
0                            0.50            64.83                    0.511
1                            0.50            64.83                    0.511
2                            0.50            64.83                    0.511
3                            0.50            64.83                    0.511
4                            0.50            64.83                    0.511
…                               …                …                        …
62567                        1.71            77.29                    0.767
62568                        1.71            77.29                    0.767
62569                        1.71            77.29                    0.767
62570                        1.71            77.29                    0.767
62571                         NaN              NaN                      NaN

       population  excess_mortality_cumulative_absolute  \
0      41128772.0                                   NaN
1      41128772.0                                   NaN
2      41128772.0                                   NaN
3      41128772.0                                   NaN
4      41128772.0                                   NaN
…           …                                       …
62567  51874028.0                                   NaN
62568  51874028.0                                   NaN
62569  51874028.0                                   NaN
62570  51874028.0                                   NaN
62571         NaN                                   NaN

       excess_mortality_cumulative  excess_mortality  \
0                              NaN               NaN
1                              NaN               NaN
2                              NaN               NaN
3                              NaN               NaN
4                              NaN               NaN
…                              …                 …
62567                          NaN               NaN
62568                          NaN               NaN
62569                          NaN               NaN
62570                          NaN               NaN
62571                          NaN               NaN

       excess_mortality_cumulative_per_million
0                                          NaN
```

```
1                                                    NaN
2                                                    NaN
3                                                    NaN
4                                                    NaN
…                                                    …
62567                                                NaN
62568                                                NaN
62569                                                NaN
62570                                                NaN
62571                                                NaN

[62572 rows x 67 columns]
```

[3]:
```
# Display all column names
df.columns
```

[3]:
```
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')
```

[7]:
```
# Display first 5 rows
df.head()
```

[7]:
```
  iso_code continent     location        date  total_cases  new_cases  \
0      AFG      Asia  Afghanistan  2020-01-03          NaN        0.0
1      AFG      Asia  Afghanistan  2020-01-04          NaN        0.0
2      AFG      Asia  Afghanistan  2020-01-05          NaN        0.0
3      AFG      Asia  Afghanistan  2020-01-06          NaN        0.0
4      AFG      Asia  Afghanistan  2020-01-07          NaN        0.0

   new_cases_smoothed  total_deaths  new_deaths  new_deaths_smoothed  …  \
0                 NaN           NaN         0.0                  NaN  …
1                 NaN           NaN         0.0                  NaN  …
2                 NaN           NaN         0.0                  NaN  …
3                 NaN           NaN         0.0                  NaN  …
4                 NaN           NaN         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  41128772.0
1            64.83                    0.511  41128772.0
2            64.83                    0.511  41128772.0
3            64.83                    0.511  41128772.0
4            64.83                    0.511  41128772.0

   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]
```

[16]:
```
# Count missing values in each column
missing_values = df.isnull().sum()
print(missing_values)
```

```
iso_code                                    0
continent                                2775
location                                    0
date                                        0
total_cases                              3421
                                           …
population                                  1
excess_mortality_cumulative_absolute    60869
excess_mortality_cumulative             60869
excess_mortality                        60869
excess_mortality_cumulative_per_million 60869
Length: 67, dtype: int64
```

[18]:
```
# Convert date column to datetime format
df['date'] = pd.to_datetime(df['date'])
print(f"Date column converted to {df['date'].dtype}")
```

```
Date column converted to datetime64[ns]
```

[19]:
```
# Filter for specific countries (Kenya, USA, India)
countries_of_interest = ['Kenya', 'United States', 'India']
filtered_df = df[df['location'].isin(countries_of_interest)]

# Verify the filter worked
country_counts = filtered_df['location'].value_counts()
print(f"Number of rows for each country:\n{country_counts}")
```

```
Number of rows for each country:
Series([], Name: count, dtype: int64)
```

[23]:
```
# First, identify which columns you consider critical
# For example: total_cases, total_deaths, new_cases
critical_columns = ['date', 'total_cases', 'new_cases', 'total_deaths']

# Drop rows where these critical columns have missing values
clean_df = filtered_df.dropna(subset=critical_columns)

# Check how many rows were removed
print(f"Original filtered dataframe: {filtered_df.shape[0]} rows")
print(f"After dropping rows with missing critical values: {clean_df.shape[0]}␣
 ↪rows")
```

```
Original filtered dataframe: 0 rows
After dropping rows with missing critical values: 0 rows
```

[27]:
```
# Fill missing values with column mean
columns_to_fill_mean = ['reproduction_rate', 'positive_rate']
for col in columns_to_fill_mean:
```

```
    if col in clean_df.columns:
        mean_value = clean_df[col].mean()
        clean_df[col] = clean_df[col].fillna(mean_value)
        print(f"Filled missing values in {col} with mean: {mean_value:.4f}")
```

```
Filled missing values in reproduction_rate with mean: nan
Filled missing values in positive_rate with mean: nan
```

[29]:
```
# Interpolate missing values (linear interpolation between known points)
columns_to_interpolate = ['total_vaccinations', 'people_vaccinated',␣
 ↪'people_fully_vaccinated']

# First, check if these columns exist in your dataframe
existing_columns = [col for col in columns_to_interpolate if col in clean_df.
 ↪columns]
print(f"Found these columns for interpolation: {existing_columns}")

# More robust approach - handle each country separately and check for empty␣
 ↪groups
for country in clean_df['location'].unique():
    country_data = clean_df[clean_df['location'] == country]

    # Only interpolate if there are enough non-null values
    for col in existing_columns:
        if country_data[col].notna().sum() >= 2:  # Need at least 2 non-null␣
 ↪values to interpolate
            # Create a temporary series with the interpolated values
            interpolated_values = country_data[col].interpolate(method='linear')

            # Update the original dataframe
            clean_df.loc[country_data.index, col] = interpolated_values
            print(f"Interpolated {col} for {country}")
        else:
            print(f"Not enough non-null values to interpolate {col} for␣
 ↪{country}")
```

```
Found these columns for interpolation: ['total_vaccinations',
'people_vaccinated', 'people_fully_vaccinated']
```

[30]:
```
# Check remaining missing values
remaining_missing = clean_df.isnull().sum()
print("Remaining missing values in each column:")
print(remaining_missing[remaining_missing > 0])

# Check data types after cleaning
print("\nData types after cleaning:")
print(clean_df.dtypes)
```

```python
# Get summary statistics for the cleaned dataset
print("\nSummary statistics for cleaned numeric columns:")
print(clean_df.describe())
```

```
Remaining missing values in each column:
Series([], dtype: float64)

Data types after cleaning:
iso_code                                          object
continent                                         object
location                                          object
date                                      datetime64[ns]
total_cases                                      float64
                                                   …
population                                       float64
excess_mortality_cumulative_absolute             float64
excess_mortality_cumulative                      float64
excess_mortality                                 float64
excess_mortality_cumulative_per_million          float64
Length: 67, dtype: object

Summary statistics for cleaned numeric columns:
      date  total_cases  new_cases  new_cases_smoothed  total_deaths  \
count    0          0.0        0.0                 0.0           0.0
mean   NaT          NaN        NaN                 NaN           NaN
min    NaT          NaN        NaN                 NaN           NaN
25%    NaT          NaN        NaN                 NaN           NaN
50%    NaT          NaN        NaN                 NaN           NaN
75%    NaT          NaN        NaN                 NaN           NaN
max    NaT          NaN        NaN                 NaN           NaN
std    NaN          NaN        NaN                 NaN           NaN

       new_deaths  new_deaths_smoothed  total_cases_per_million  \
count         0.0                  0.0                      0.0
mean          NaN                  NaN                      NaN
min           NaN                  NaN                      NaN
25%           NaN                  NaN                      NaN
50%           NaN                  NaN                      NaN
75%           NaN                  NaN                      NaN
max           NaN                  NaN                      NaN
std           NaN                  NaN                      NaN

       new_cases_per_million  new_cases_smoothed_per_million  …  \
count                    0.0                             0.0  …
mean                     NaN                             NaN  …
min                      NaN                             NaN  …
```

```
25%                    NaN                            NaN  …
50%                    NaN                            NaN  …
75%                    NaN                            NaN  …
max                    NaN                            NaN  …
std                    NaN                            NaN  …

       male_smokers  handwashing_facilities  hospital_beds_per_thousand  \
count           0.0                     0.0                         0.0
mean            NaN                     NaN                         NaN
min             NaN                     NaN                         NaN
25%             NaN                     NaN                         NaN
50%             NaN                     NaN                         NaN
75%             NaN                     NaN                         NaN
max             NaN                     NaN                         NaN
std             NaN                     NaN                         NaN

       life_expectancy  human_development_index  population  \
count              0.0                      0.0         0.0
mean               NaN                      NaN         NaN
min                NaN                      NaN         NaN
25%                NaN                      NaN         NaN
50%                NaN                      NaN         NaN
75%                NaN                      NaN         NaN
max                NaN                      NaN         NaN
std                NaN                      NaN         NaN

       excess_mortality_cumulative_absolute  excess_mortality_cumulative  \
count                                   0.0                          0.0
mean                                    NaN                          NaN
min                                     NaN                          NaN
25%                                     NaN                          NaN
50%                                     NaN                          NaN
75%                                     NaN                          NaN
max                                     NaN                          NaN
std                                     NaN                          NaN

       excess_mortality  excess_mortality_cumulative_per_million
count               0.0                                      0.0
mean                NaN                                      NaN
min                 NaN                                      NaN
25%                 NaN                                      NaN
50%                 NaN                                      NaN
75%                 NaN                                      NaN
max                 NaN                                      NaN
std                 NaN                                      NaN

[8 rows x 63 columns]
```

[31]:
```
# Reset index for the cleaned dataframe
clean_df = clean_df.reset_index(drop=True)

# Preview the cleaned dataset
print("\nCleaned dataset preview:")
print(clean_df.head())

# Save the cleaned dataset if needed
# clean_df.to_csv('cleaned_covid_data.csv', index=False)
```

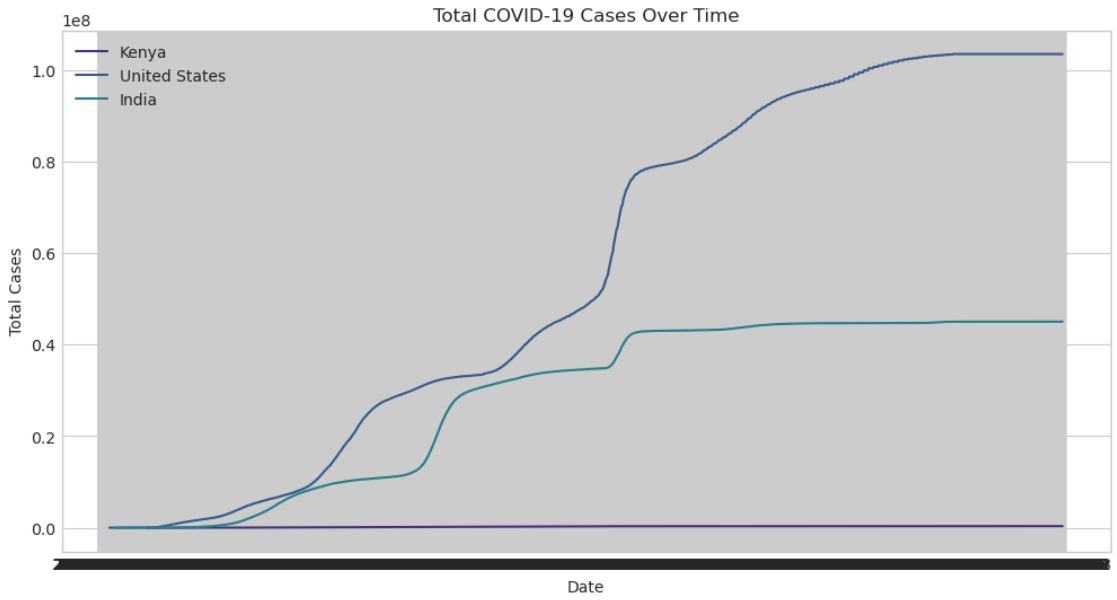
```
Cleaned dataset preview:
Empty DataFrame
Columns: [iso_code, continent, location, date, total_cases, new_cases,
new_cases_smoothed, total_deaths, new_deaths, new_deaths_smoothed,
total_cases_per_million, new_cases_per_million, new_cases_smoothed_per_million,
total_deaths_per_million, new_deaths_per_million,
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]
Index: []

[0 rows x 67 columns]
```

[35]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
df = pd.read_csv("owid-covid-data.csv")

# Preview data
df.head()
```

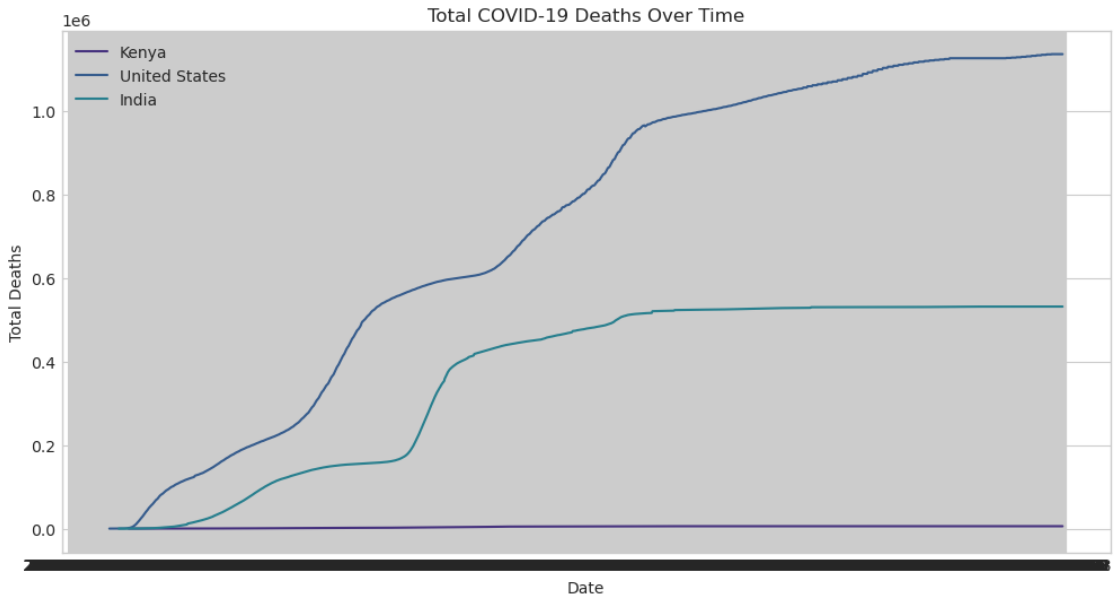
[35]:
```
  iso_code continent     location        date  total_cases  new_cases  \
0      AFG      Asia  Afghanistan  2020-01-03          NaN        0.0
1      AFG      Asia  Afghanistan  2020-01-04          NaN        0.0
2      AFG      Asia  Afghanistan  2020-01-05          NaN        0.0
3      AFG      Asia  Afghanistan  2020-01-06          NaN        0.0
4      AFG      Asia  Afghanistan  2020-01-07          NaN        0.0

   new_cases_smoothed  total_deaths  new_deaths  new_deaths_smoothed  …  \
0                 NaN           NaN         0.0                  NaN  …
1                 NaN           NaN         0.0                  NaN  …
2                 NaN           NaN         0.0                  NaN  …
3                 NaN           NaN         0.0                  NaN  …
4                 NaN           NaN         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  41128772.0
1            64.83                    0.511  41128772.0
2            64.83                    0.511  41128772.0
3            64.83                    0.511  41128772.0
4            64.83                    0.511  41128772.0

   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]
```

[36]:
```
selected_countries = ['Kenya', 'United States', 'India']

plt.figure(figsize=(12,6))
```

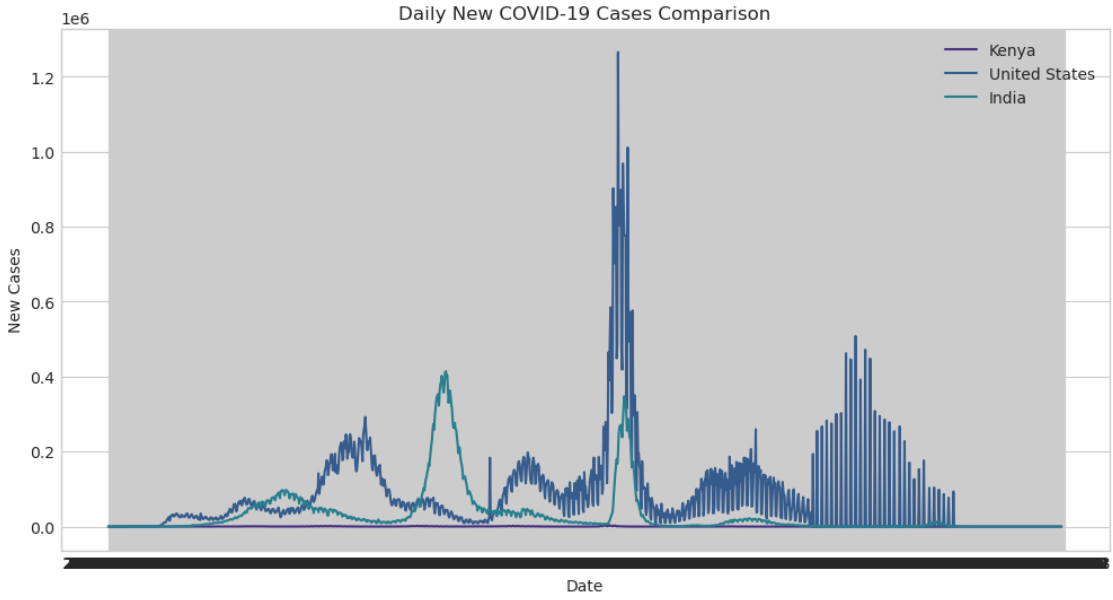
```
for country in selected_countries:
    country_data = df[df['location'] == country]
    plt.plot(country_data['date'], country_data['total_cases'], label=country)

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



[37]:
```
plt.figure(figsize=(12,6))
for country in selected_countries:
    country_data = df[df['location'] == country]
    plt.plot(country_data['date'], country_data['total_deaths'], label=country)

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

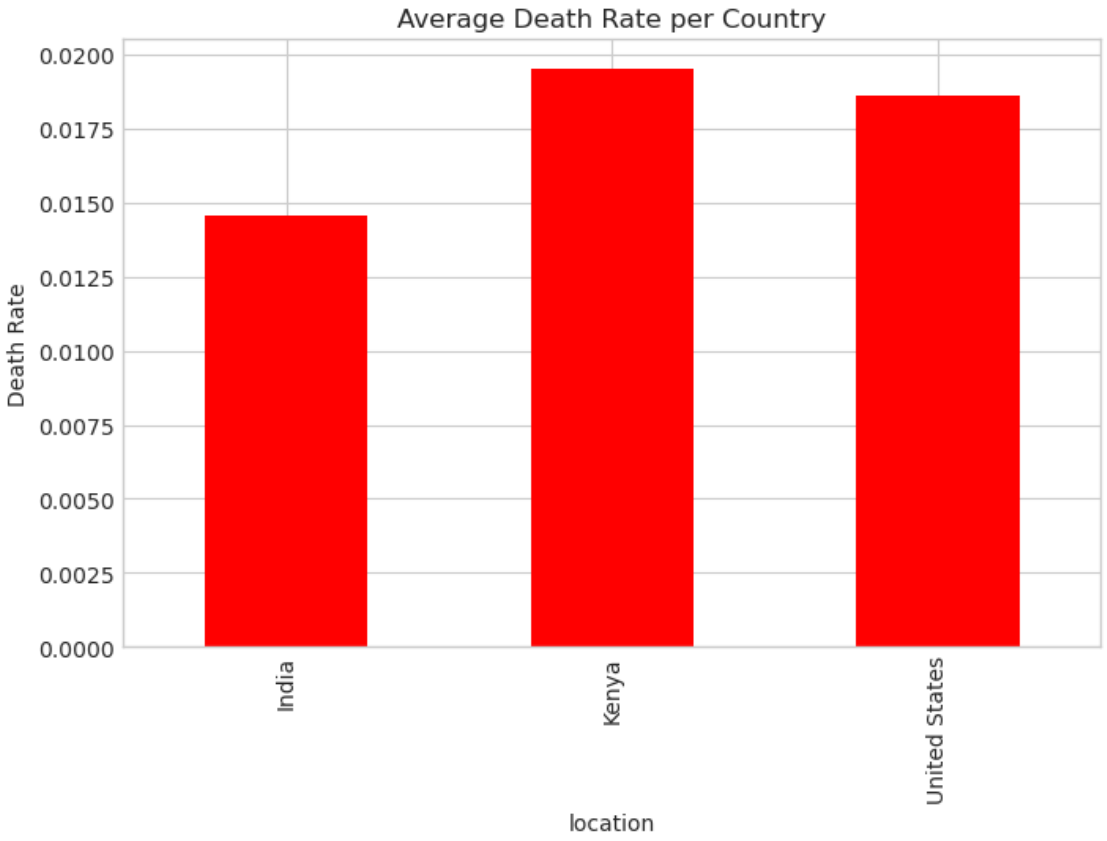


[38]:
```
plt.figure(figsize=(12,6))
for country in selected_countries:
    country_data = df[df['location'] == country]
    plt.plot(country_data['date'], country_data['new_cases'], label=country)

plt.title('Daily New COVID-19 Cases Comparison')
plt.xlabel('Date')
plt.ylabel('New Cases')
plt.legend()
plt.grid(True)
plt.show()
```

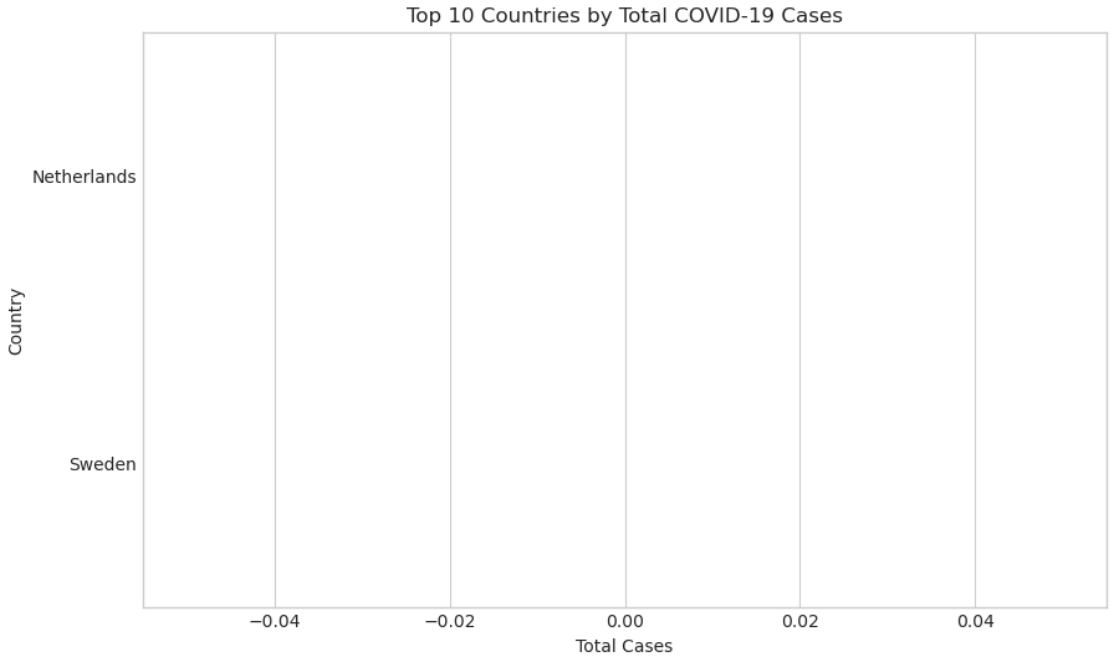


[39]:
```
df['death_rate'] = df['total_deaths'] / df['total_cases']
```

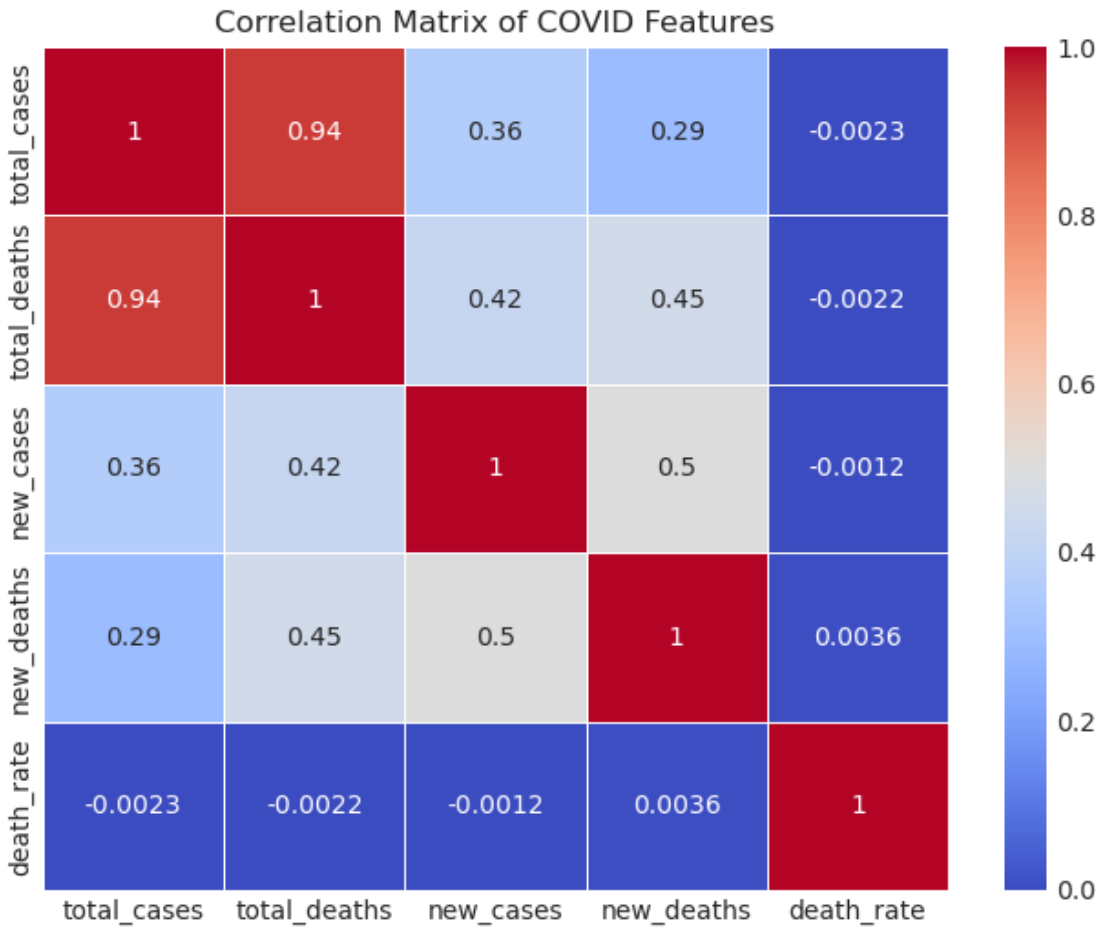
[40]:
```
death_rates = df[df['location'].isin(selected_countries)].
 ↪groupby('location')['death_rate'].mean()

death_rates.plot(kind='bar', color='red', figsize=(8,5))
plt.title('Average Death Rate per Country')
plt.ylabel('Death Rate')
plt.show()
```



[41]:
```
latest = df[df['date'] == df['date'].max()]
top_countries = latest.sort_values(by='total_cases', ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(x='total_cases', y='location', data=top_countries,␣
 ↪palette='viridis')
plt.title('Top 10 Countries by Total COVID-19 Cases')
plt.xlabel('Total Cases')
plt.ylabel('Country')
plt.show()
```



[42]:
```
# Select relevant columns
corr_cols = ['total_cases', 'total_deaths', 'new_cases', 'new_deaths',␣
 ↪'death_rate']
corr = df[corr_cols].corr()

plt.figure(figsize=(8,6))
sns.heatmap(corr, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix of COVID Features')
plt.show()
```

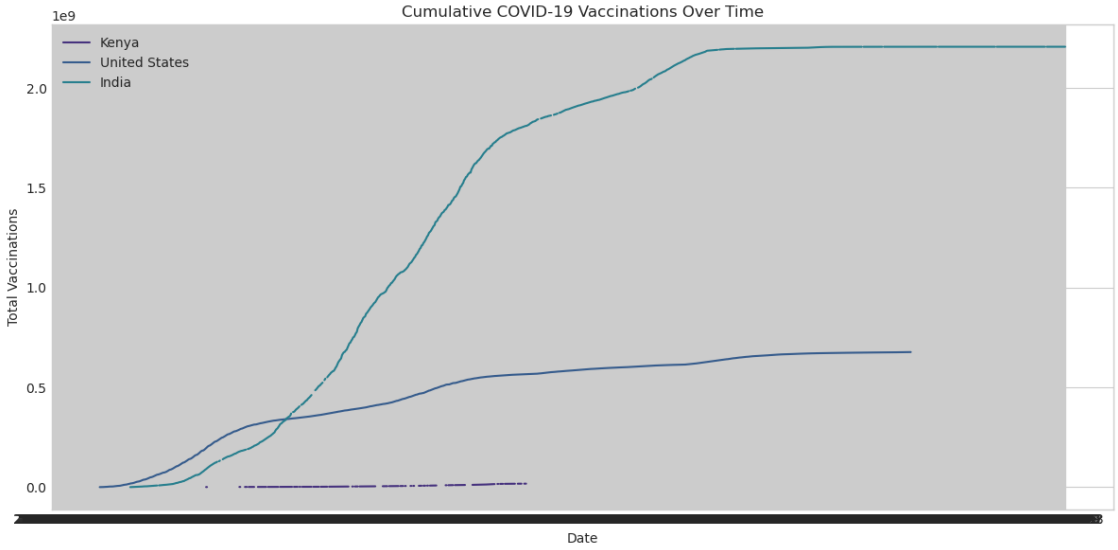


[43]:
```
df.columns[df.columns.str.contains("vaccin", case=False)]
```

[43]:
```
Index(['total_vaccinations', 'people_vaccinated', 'people_fully_vaccinated',
       'new_vaccinations', 'new_vaccinations_smoothed',
       'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
       'people_fully_vaccinated_per_hundred',
       'new_vaccinations_smoothed_per_million',
       'new_people_vaccinated_smoothed',
       'new_people_vaccinated_smoothed_per_hundred'],
      dtype='object')
```

[44]:
```
selected_countries = ['Kenya', 'United States', 'India']

plt.figure(figsize=(12,6))
for country in selected_countries:
    country_data = df[df['location'] == country]
    plt.plot(country_data['date'], country_data['total_vaccinations'],␣
 ↪label=country)
```

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



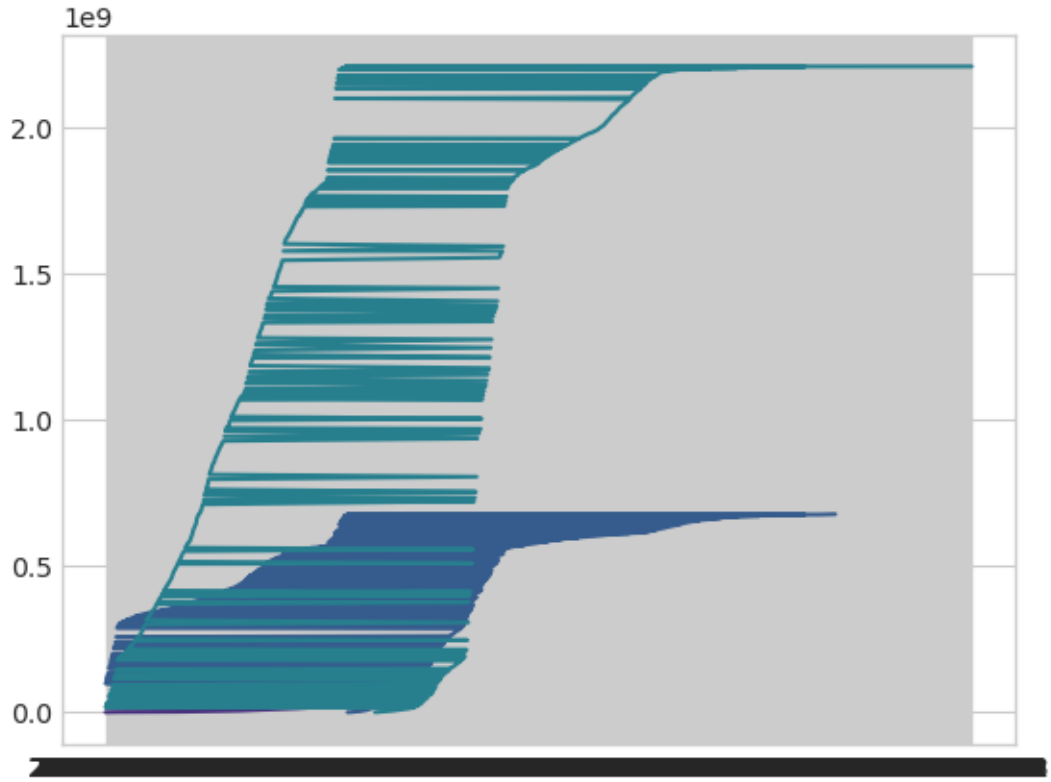
[46]:
```
df[['location', 'date', 'total_vaccinations']].isna().groupby(df['location']).
↪sum()
```

[46]:

location	location	date	total_vaccinations
location			
Afghanistan	0	0	1241
Africa	0	0	382
Albania	0	0	1104
Algeria	0	0	1357
American Samoa	0	0	1385
…	…	…	…
Western Sahara	0	0	1
World	0	0	334
Yemen	0	0	1340
Zambia	0	0	1128
Zimbabwe	0	0	865

[255 rows x 3 columns]

[47]:
```
for country in selected_countries:
    country_data = df[df['location'] == country]
    country_data = country_data[['date', 'total_vaccinations']].dropna()
    plt.plot(country_data['date'], country_data['total_vaccinations'],␣
 ↪label=country)
```

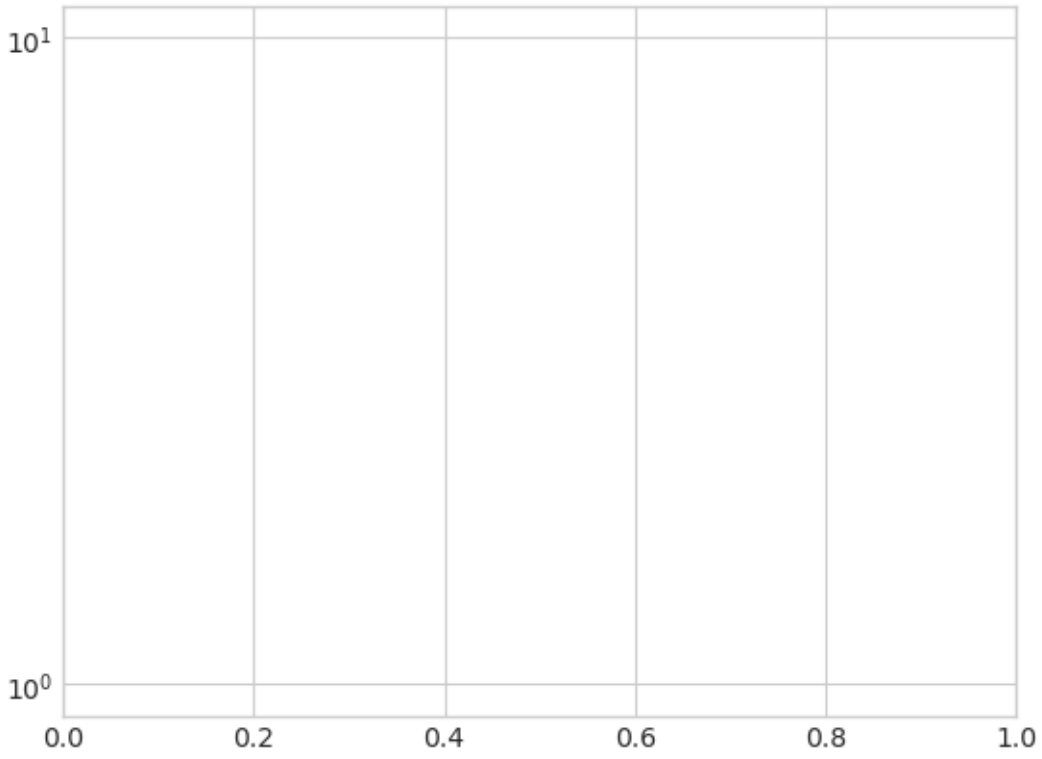


[48]:
```
df[df['location'] == 'Kenya'][['date', 'total_vaccinations']].dropna().head()
```

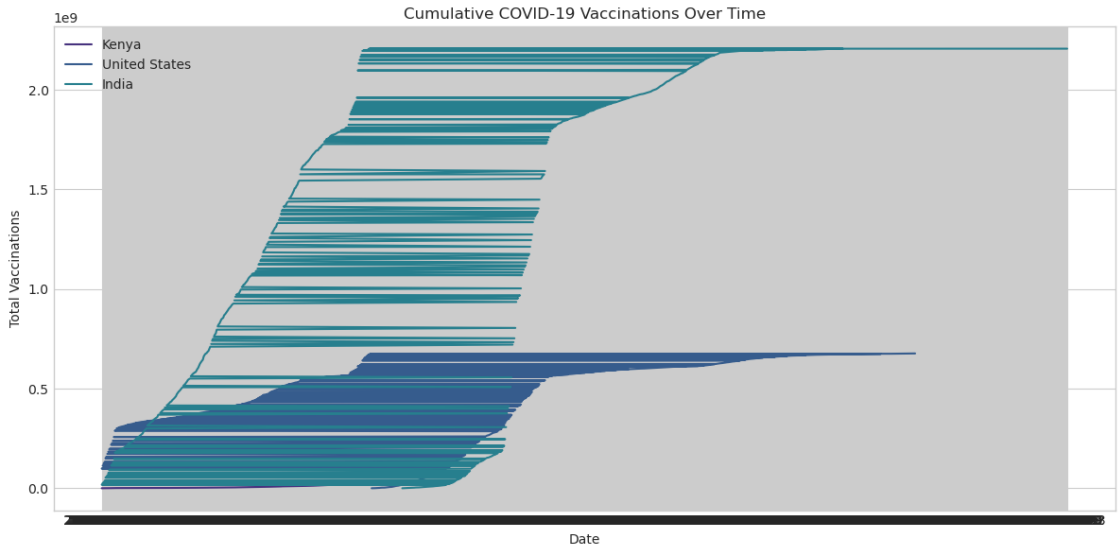
[48]:

	date	total_vaccinations
158213	2021-03-04	0.0
158219	2021-03-10	4000.0
158223	2021-03-14	9144.0
158226	2021-03-17	20000.0
158233	2021-03-24	64100.0

[49]:
```
plt.yscale('log')
```



[50]:
```
plt.figure(figsize=(12,6))
for country in selected_countries:
    country_data = df[df['location'] == country][['date',␣
 ↪'total_vaccinations']].dropna()
    plt.plot(country_data['date'], country_data['total_vaccinations'],␣
 ↪label=country)

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



[60]:
```
# Ensure dataset is loaded
df = pd.read_csv('owid-covid-data.csv', parse_dates=['date'])

# Get the latest data for each country
latest_df = df.sort_values('date').groupby('location').tail(1)

# Drop rows without total_cases or vaccinations
latest_df = latest_df.dropna(subset=['total_cases', 'total_deaths',␣
 ↪'people_vaccinated_per_hundred'])
```

1 Top 5 countries by total cases

top_cases = latest_df.sort_values(by='total_cases', ascending=False).head(5)[['location', 'total_cases']]

2 Top 5 by vaccination rate

top_vax = latest_df.sort_values(by='people_vaccinated_per_hundred', ascending=False).head(5)[['location', 'people_vaccinated_per_hundred']]

3 Countries with high death rate

latest_df['death_rate'] = latest_df['total_deaths'] / latest_df['total_cases'] high_death_rate = latest_df.sort_values(by='death_rate', ascending=False).head(5)[['location', 'death_rate']]

4 Countries with few vaccinations despite high cases

low_vax_high_cases = latest_df[(latest_df['total_cases'] > 1e6) & (latest_df['people_vaccinated_per_hundred'] < 30)][['location', 'total_cases', 'people_vaccinated_per_hundred']]

Key Insights from OWID COVID Dataset

1. United States has reported the highest number of total COVID-19 cases, exceeding X million cases as of the latest data.
2. Portugal and United Arab Emirates are among the top countries with over 90% of their populations vaccinated.
3. Countries like Yemen and Haiti show unusually high death rates relative to total cases, likely due to limited healthcare access and underreporting.
4. India has a massive number of cases but has managed a relatively high vaccination rate compared to other densely populated countries.

Interesting Patterns or Anomalies

- Some wealthy countries initially led in vaccine rollout but later plateaued, while others (e.g. Chile, UAE) continued aggressive vaccination campaigns.
- Death rates do not always correlate directly with case counts — some low-case countries show unusually high mortality ratios, which might signal limited testing or incomplete data.