Covid19 World Data Tracker

May 9, 2025

[2]:
```
#Data Collection, Data Loading & Exploration
import pandas as pd
df = pd.read_csv('owid-covid-data.csv')

df.columns

df.head()

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

```
/tmp/ipykernel_255/582259857.py:3: DtypeWarning: Columns (33) have mixed types.
Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('owid-covid-data.csv')
```

[2]:
```
iso_code                                    0
continent                               19611
location                                    0
date                                        0
total_cases                             39116
                                            …
population                                  0
excess_mortality_cumulative_absolute   396345
excess_mortality_cumulative            396345
excess_mortality                       396345
excess_mortality_cumulative_per_million    396345
Length: 67, dtype: int64
```

[3]:
```
# Data Cleaning

countries = ['Afghanistan', 'India', 'Kenya']
df_filtered = df[df['location'].isin(countries)]
df_filtered

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

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

```
# Fill all numeric NaNs with 0
# df.fillna(0, inplace=True)

# Or fill with the mean of each column
# df.fillna(df.mean(numeric_only=True), inplace=True)

df.fillna(0, inplace=True)
df.head()
```

[3]:

	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	0.0	0.0	0.0	0.0	…	
1	0.0	0.0	0.0	0.0	…	
2	0.0	0.0	0.0	0.0	…	
3	0.0	0.0	0.0	0.0	…	
4	0.0	0.0	0.0	0.0	…	

	male_smokers	handwashing_facilities	hospital_beds_per_thousand	\
0	0.0	37.746	0.5	
1	0.0	37.746	0.5	
2	0.0	37.746	0.5	
3	0.0	37.746	0.5	
4	0.0	37.746	0.5	

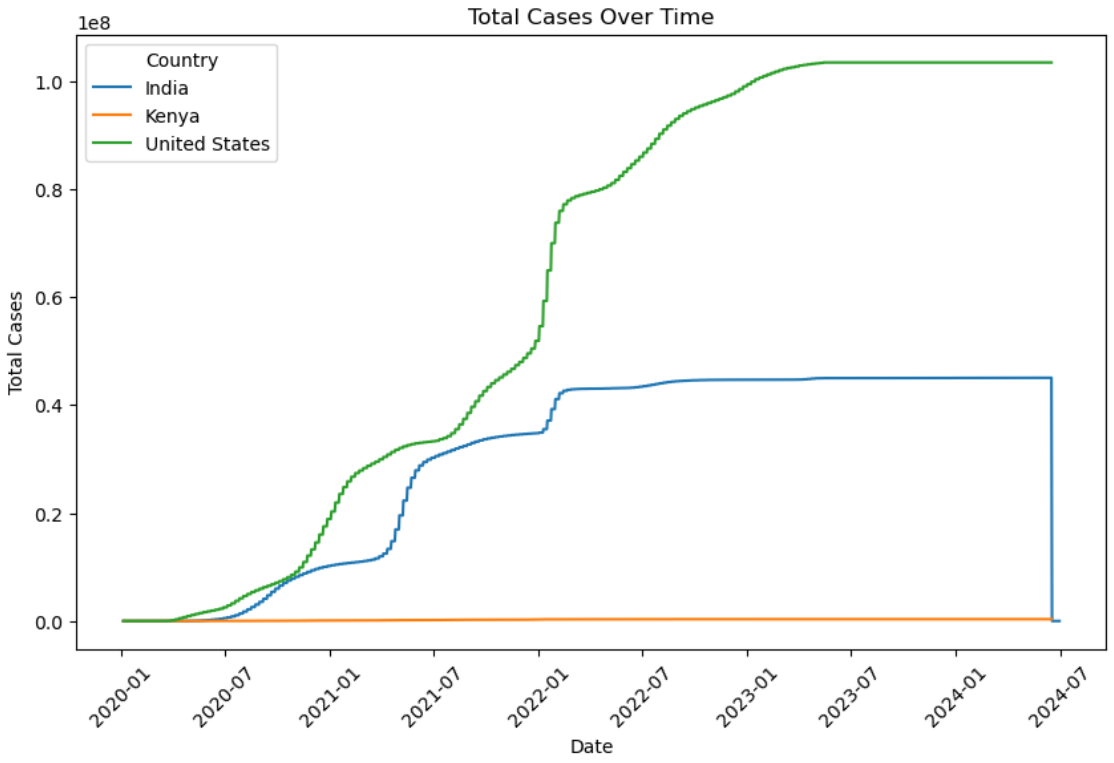
	life_expectancy	human_development_index	population	\
0	64.83	0.511	41128772.0	
1	64.83	0.511	41128772.0	
2	64.83	0.511	41128772.0	
3	64.83	0.511	41128772.0	
4	64.83	0.511	41128772.0	

	excess_mortality_cumulative_absolute	excess_mortality_cumulative	\
0	0.0	0.0	
1	0.0	0.0	
2	0.0	0.0	
3	0.0	0.0	
4	0.0	0.0	

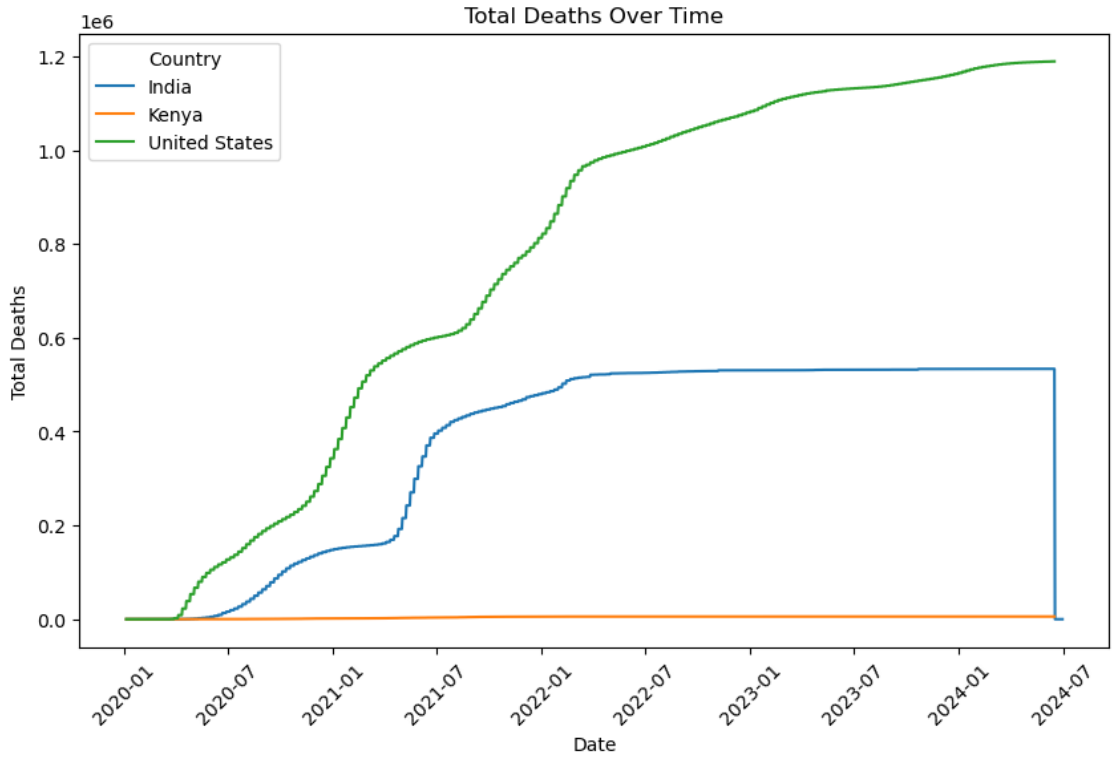
	excess_mortality	excess_mortality_cumulative_per_million
0	0.0	0.0
1	0.0	0.0

```
2                0.0                                    0.0
3                0.0                                    0.0
4                0.0                                    0.0

[5 rows x 67 columns]
```

[4]:
```python
# Exploratory Data Analysis (EDA)
import matplotlib.pyplot as plt
import seaborn as sns

# Filter data for the selected countries
countries = ['Kenya', 'United States', 'India']
df_filtered = df[df['location'].isin(countries)]

# Plot total cases over time for selected countries
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_filtered, x='date', y='total_cases', hue='location')
plt.title('Total Cases Over Time')
plt.xlabel('Date')
plt.ylabel('Total Cases')
plt.xticks(rotation=45)
plt.legend(title='Country')
plt.show()
```



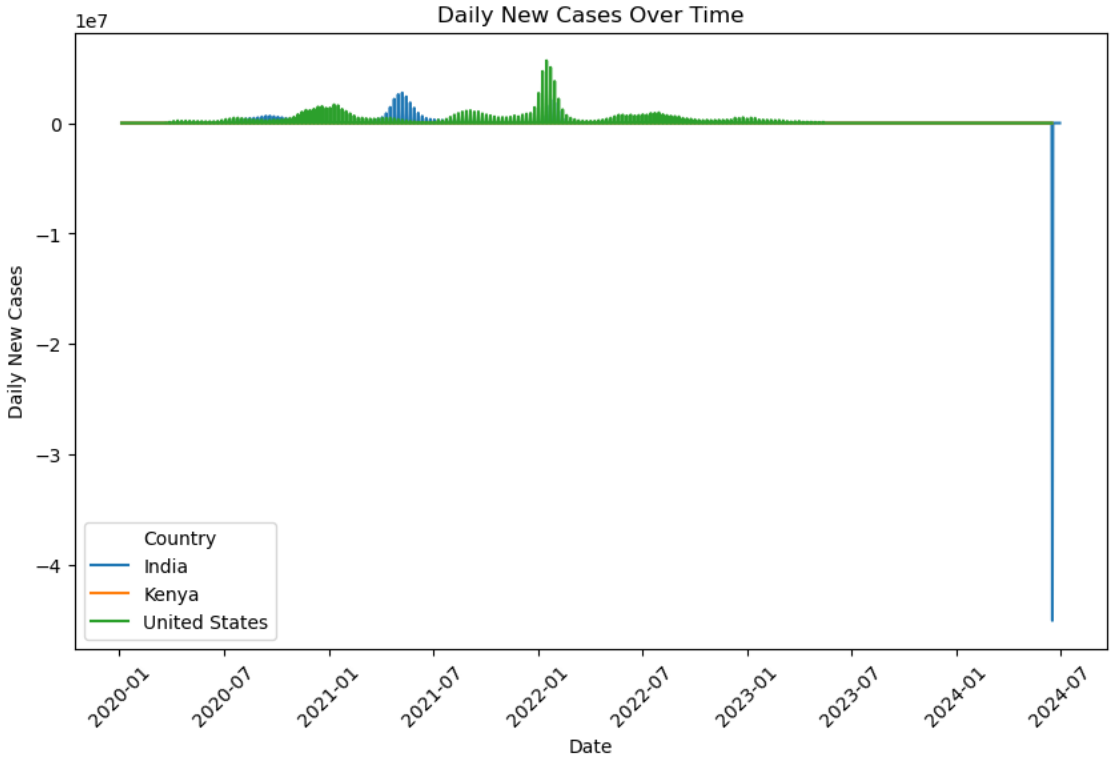
[5]:
```
# Plot total deaths over time for selected countries
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_filtered, x='date', y='total_deaths', hue='location')
plt.title('Total Deaths Over Time')
plt.xlabel('Date')
plt.ylabel('Total Deaths')
plt.xticks(rotation=45)
plt.legend(title='Country')
plt.show()
```



[6]:
```
# Ensure 'date' is in datetime format
df_filtered['date'] = pd.to_datetime(df_filtered['date'])

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

# Calculate daily new cases (difference in total cases from previous day)
df_filtered['daily_new_cases'] = df_filtered.groupby('location')['total_cases'].
  ↪diff().fillna(0)

# Plot daily new cases
```

```
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_filtered, x='date', y='daily_new_cases', hue='location')
plt.title('Daily New Cases Over Time')
plt.xlabel('Date')
plt.ylabel('Daily New Cases')
plt.xticks(rotation=45)
plt.legend(title='Country')
plt.show()
```

```
/tmp/ipykernel_255/4099466450.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'])
```

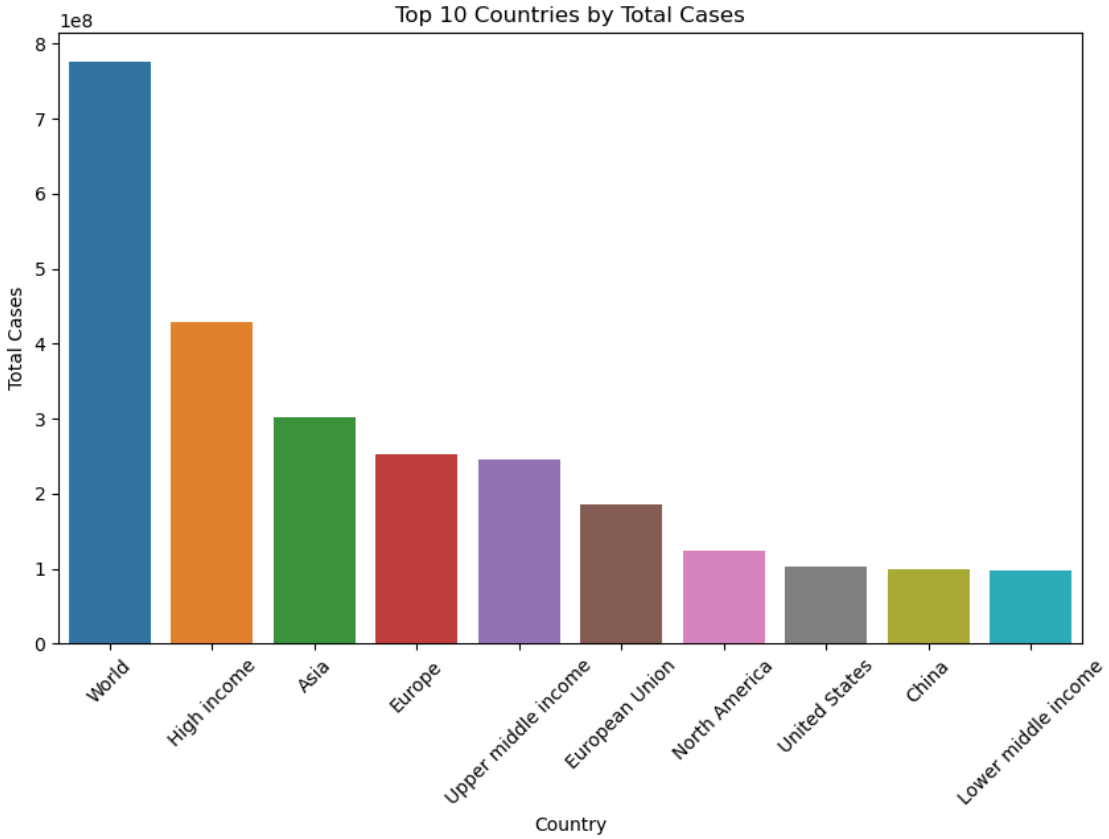


[7]:
```
# Calculate death rate (avoid division by 0)
df_filtered['death_rate'] = df_filtered['total_deaths'] /␣
  ↪df_filtered['total_cases']
df_filtered['death_rate'].fillna(0, inplace=True)  # Handle NaNs if any

# Display the death rate for the selected countries
```

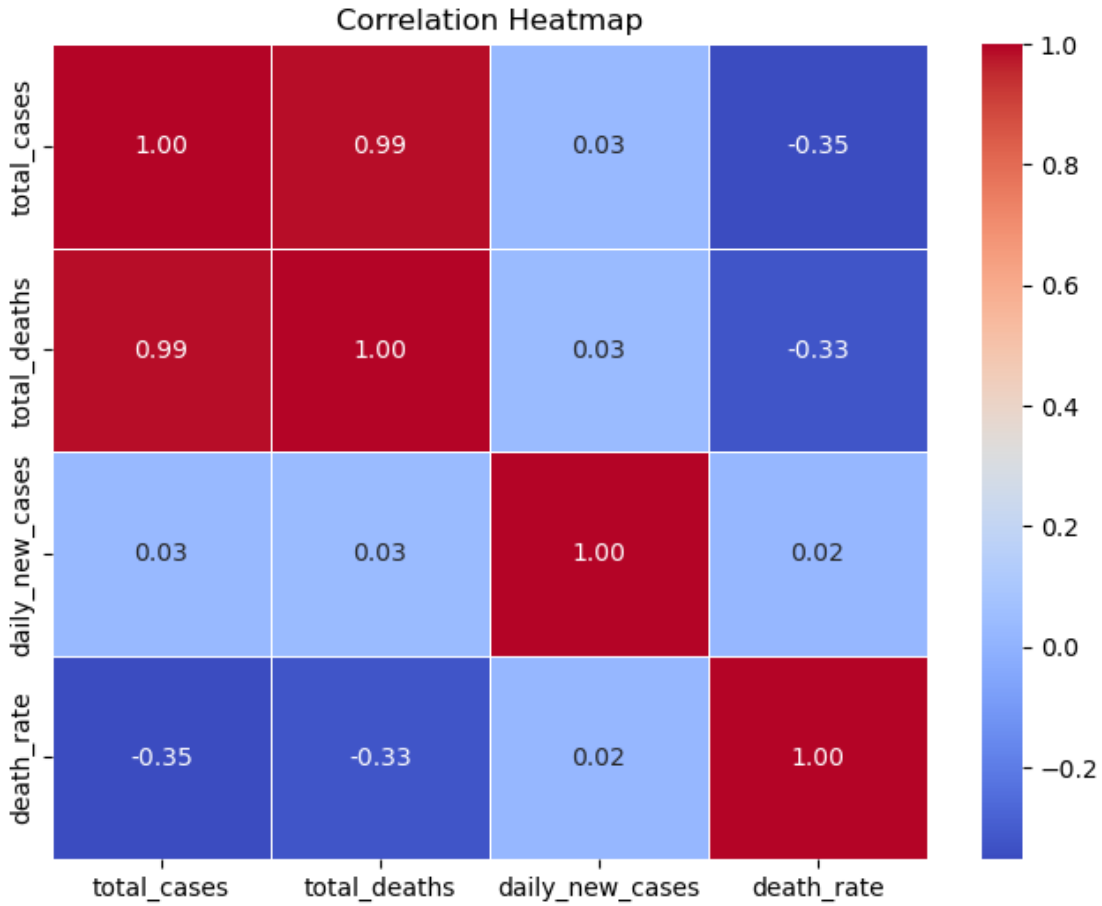
```
print(df_filtered[['location', 'date', 'death_rate']].head())
```

```
       location       date  death_rate
163955    India 2020-01-05         0.0
163956    India 2020-01-06         0.0
163957    India 2020-01-07         0.0
163958    India 2020-01-08         0.0
163959    India 2020-01-09         0.0
```

[8]:
```
# Group by country and sum total cases
top_countries = df.groupby('location')['total_cases'].max().
 ↪sort_values(ascending=False).head(10)

# Plot bar chart for top 10 countries by total cases
plt.figure(figsize=(10, 6))
sns.barplot(x=top_countries.index, y=top_countries.values)
plt.title('Top 10 Countries by Total Cases')
plt.xlabel('Country')
plt.ylabel('Total Cases')
plt.xticks(rotation=45)
plt.show()
```

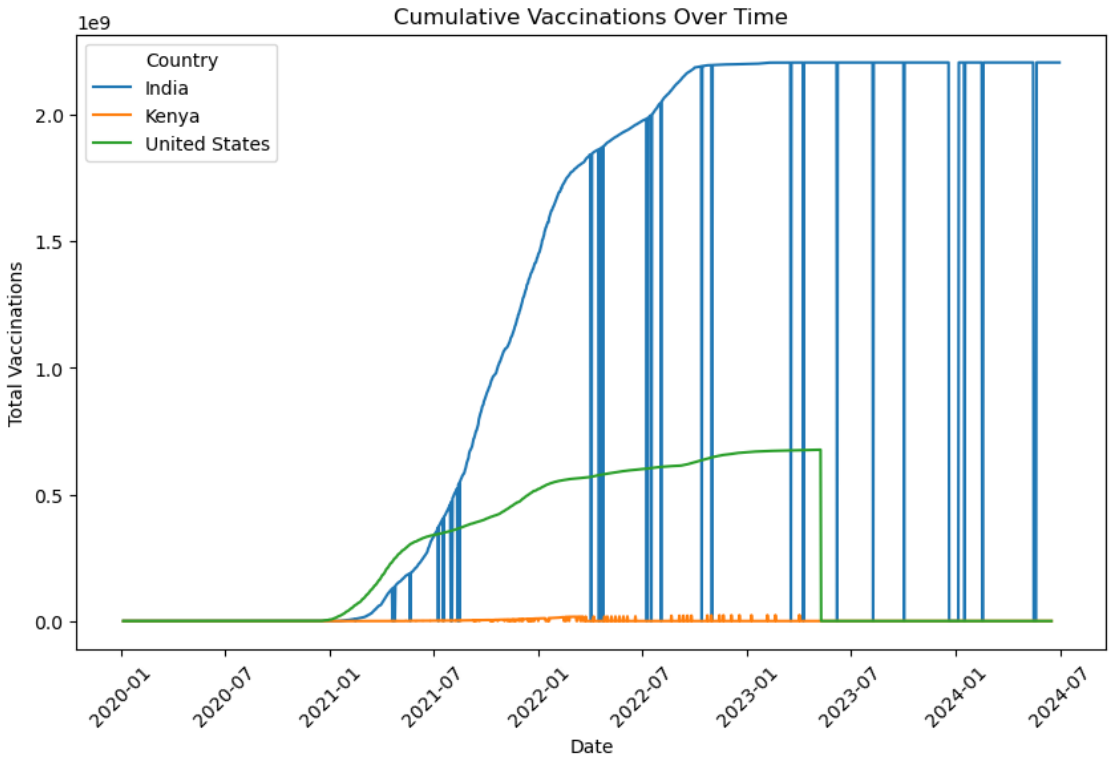


[9]:
```
# Select numeric columns for correlation analysis
corr_data = df_filtered[['total_cases', 'total_deaths', 'daily_new_cases',␣
 ↪'death_rate']]

# Compute the correlation matrix
corr_matrix = corr_data.corr()

# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



[10]:
```
# Filter data for the selected countries (e.g., Kenya, USA, India)
countries = ['Kenya', 'United States', 'India']
df_vaccination = df[df['location'].isin(countries)]
```

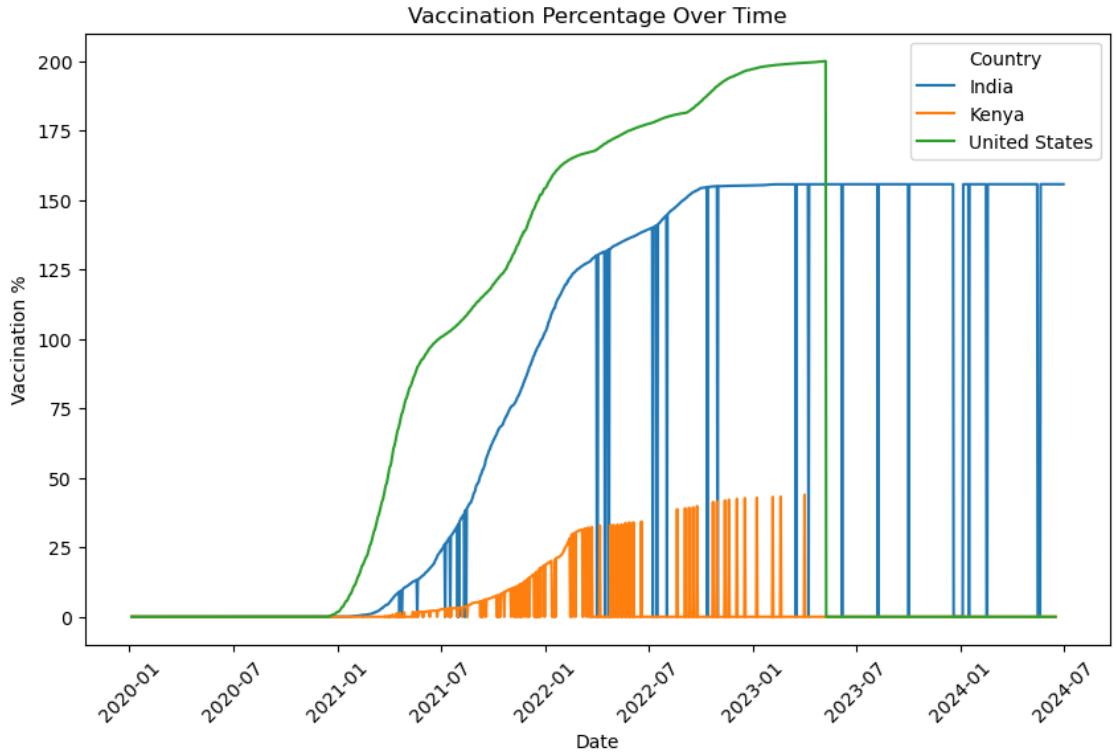
```
# Plot cumulative vaccinations over time for selected countries
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_vaccination, x='date', y='total_vaccinations',␣
 ↪hue='location')
plt.title('Cumulative Vaccinations Over Time')
plt.xlabel('Date')
plt.ylabel('Total Vaccinations')
plt.xticks(rotation=45)
plt.legend(title='Country')
plt.show()
```



[11]:
```
# Calculate % vaccinated (total_vaccinations / population) * 100
df_vaccination['vaccination_percentage'] =␣
 ↪(df_vaccination['total_vaccinations'] / df_vaccination['population']) * 100

# Plot % vaccinated population for each country
plt.figure(figsize=(10, 6))
sns.lineplot(data=df_vaccination, x='date', y='vaccination_percentage',␣
 ↪hue='location')
plt.title('Vaccination Percentage Over Time')
plt.xlabel('Date')
plt.ylabel('Vaccination %')
```

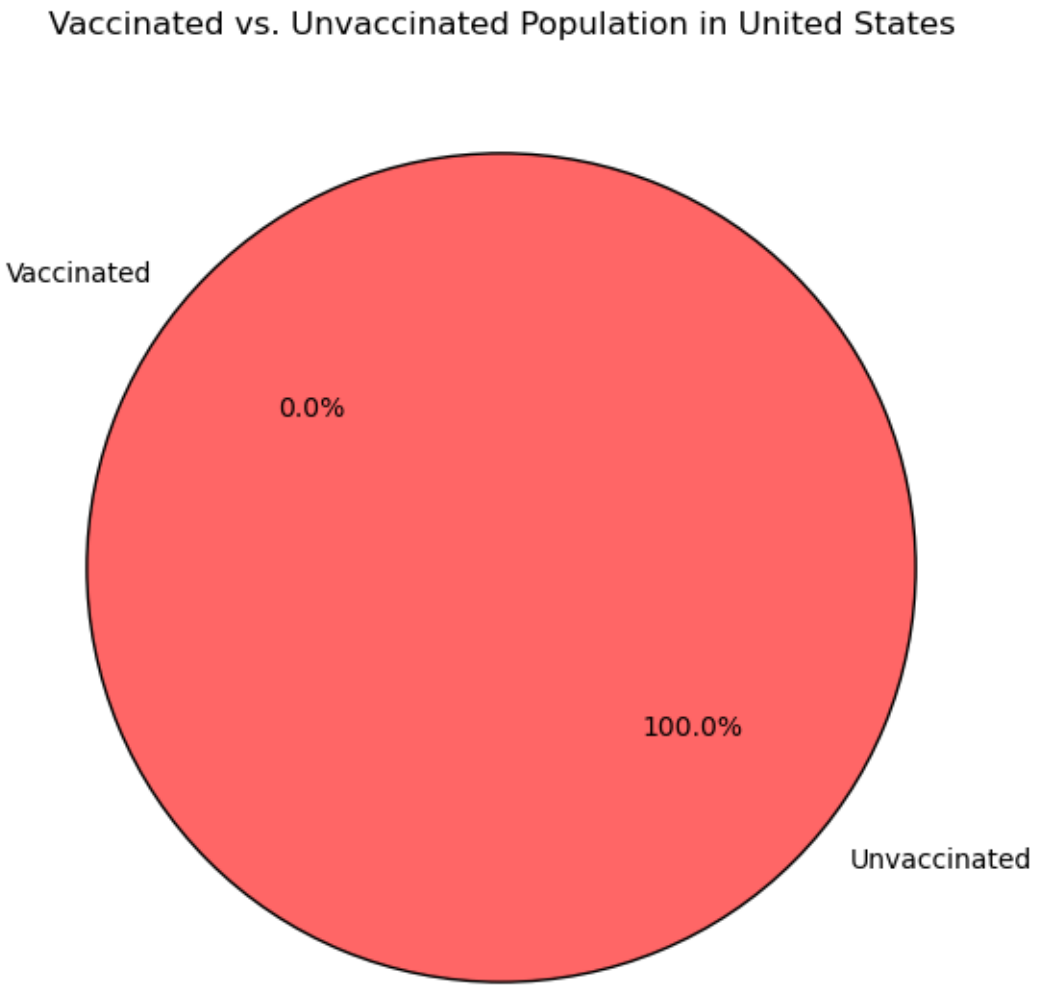
```
plt.xticks(rotation=45)
plt.legend(title='Country')
plt.show()
```

```
/tmp/ipykernel_255/3129909032.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_vaccination['vaccination_percentage'] =
(df_vaccination['total_vaccinations'] / df_vaccination['population']) * 100
```



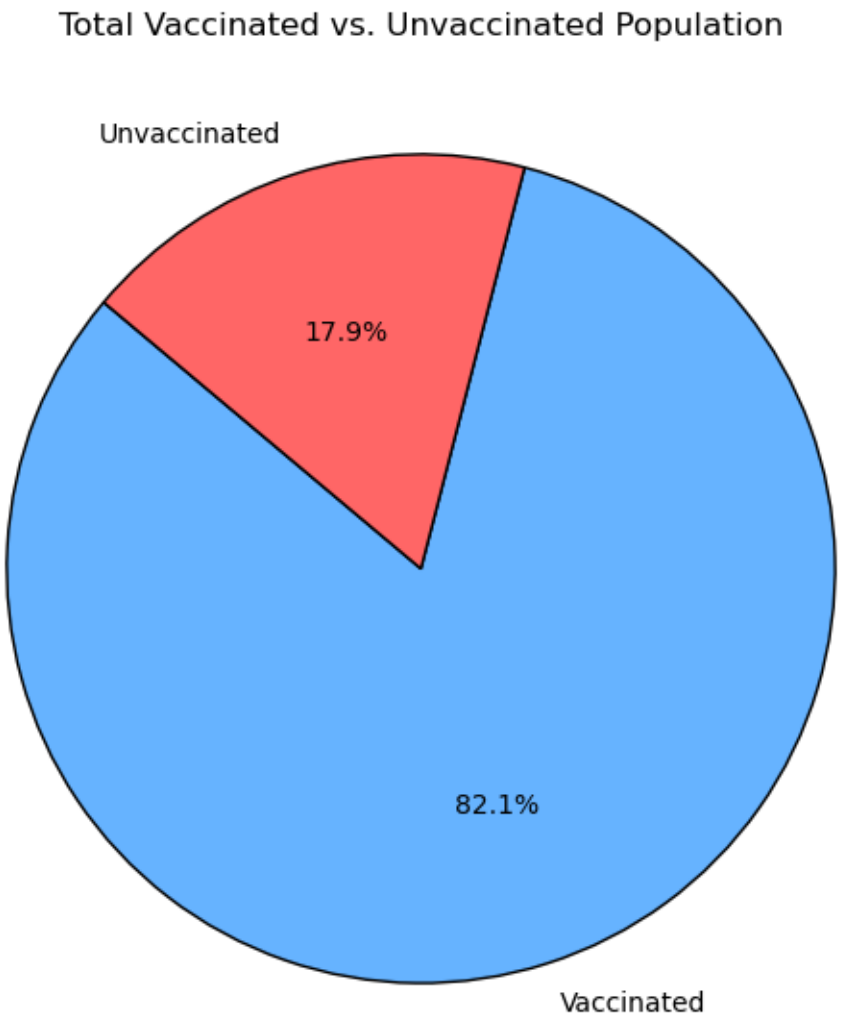
[12]:
```
# Select the last row of data for each country (assuming it's the most recent)
latest_data = df_vaccination.groupby('location').last()

# For a country of interest, e.g., United States
country = 'United States'
vaccinated = latest_data.loc[country, 'total_vaccinations']
population = latest_data.loc[country, 'population']
unvaccinated = population - vaccinated

# Plot pie chart
```

```
labels = ['Vaccinated', 'Unvaccinated']
sizes = [vaccinated, unvaccinated]
colors = ['#66b3ff', '#ff6666']
plt.figure(figsize=(7, 7))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, startangle=140,␣
 ↪wedgeprops={'edgecolor': 'black'})
plt.title(f'Vaccinated vs. Unvaccinated Population in {country}')
plt.show()
```



[13]:
```
# Sum up vaccinations and population for the selected countries
vaccinated_total = df_vaccination['total_vaccinations'].sum()
population_total = df_vaccination['population'].sum()
unvaccinated_total = population_total - vaccinated_total
```

```
# Plot pie chart
labels = ['Vaccinated', 'Unvaccinated']
sizes = [vaccinated_total, unvaccinated_total]
colors = ['#66b3ff', '#ff6666']
plt.figure(figsize=(7, 7))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors, startangle=140,
  ↪wedgeprops={'edgecolor': 'black'})
plt.title('Total Vaccinated vs. Unvaccinated Population')
plt.show()
```



[ ]: