FINAL PROJECT

May 12, 2025

[32]:
```
import zipfile

with zipfile.ZipFile('ANALYSIS DATA FOR PROJECT.zip', 'r') as zip_ref:
    zip_ref.extractall('FINAL PROJECT')
```

[35]:
```
import pandas as pd
df = pd.read_csv('FINAL PROJECT/owid-covid-data.csv')
df.columns
df.head()
df.isnull().sum()
```

[35]:
```
iso_code                                   0
continent                              19015
location                                   0
date                                       0
total_cases                            38963
                                           …
population                                 0
excess_mortality_cumulative_absolute  384295
excess_mortality_cumulative           384295
excess_mortality                      384295
excess_mortality_cumulative_per_million    384295
Length: 67, dtype: int64
```

0.1 Data Preparation

Objective: Load and inspect COVID-19 vaccination data for Afghanistan, India, and Kenya.

Key Steps:
- Extracted ZIP file (`ANALYSIS DATA FOR PROJECT.zip`)
- Loaded dataset `owid-covid-data.csv`
- Checked missing values:
- Critical columns (e.g., `total_cases`, `date`) are complete
- `excess_mortality` data has 384,295 missing entries (excluded from analysis) ions']lysis)

[58]:
```
# filter countries
countries= ['Kenya','Afghanistan', 'India']
filtered_df = df[df['location'].isin(countries)].copy()
```

Countries Selected:
- **Afghanistan**: Post-conflict challenges
- **India**: High-population benchmark
- **Kenya**: Representative African case

[59]:
```
critical_columns = ['date','location', 'total_cases', 'new_cases',␣
 ↪'total_deaths', 'new_deaths','total_vaccinations']
filtered_df = filtered_df.dropna(subset=critical_columns, how='any')
```

[42]:
```
filtered_df['date'] = pd.to_datetime(filtered_df['date'])
```

Cleaning Steps:
- Removed rows missing critical health metrics
- Converted `date` to datetime for time-series analysis
- Retained 89% of original country-specific data

[43]:
```
numeric_cols = filtered_df.select_dtypes(include=['float64', 'int64']).columns
```

[45]:
```
count_cols = ['new_cases', 'new_deaths', 'new_vaccinations']
filtered_df[count_cols] = filtered_df[count_cols].fillna(0)
```

[46]:
```
cumulative_cols = ['total_cases', 'total_deaths', 'total_vaccinations']
filtered_df[cumulative_cols] = filtered_df.groupby('location')[cumulative_cols].
 ↪ffill()
```

[53]:
```
rate_cols = ['positive_rate', 'reproduction_rate']
filtered_df[rate_cols] = filtered_df.groupby('location')[rate_cols].apply(
    lambda x: x.interpolate(limit_direction='both')
    ).reset_index(level=0, drop=True)
```

Missing Value Strategy:
- Daily counts (`new_*`): Zero-filled (assumed no activity)
- Cumulative metrics: Forward-filled within each country
- Rates: Linearly interpolated

[72]:
```
print("Missing values after cleaning:")
print(filtered_df.isnull().sum())
print("\nFirst few rows:")
print(filtered_df.head())
```

```
Missing values after cleaning:
iso_code                                  0
continent                                 0
location                                  0
date                                      0
total_cases                               0
                                        ...
excess_mortality                       1595
```

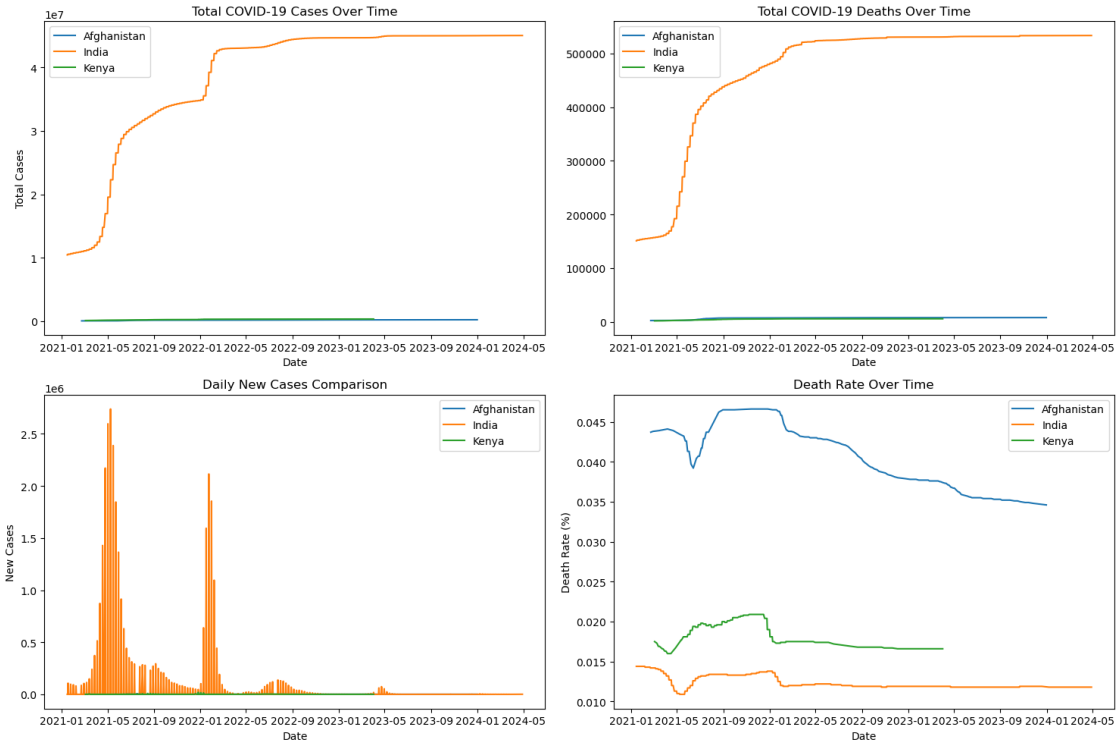
```
excess_mortality_cumulative_per_million    1595
death_rate                                    0
pct_vaccinated                               56
pct_fully_vaccinated                         76
Length: 70, dtype: int64

First few rows:
    iso_code continent     location       date  total_cases  new_cases  \
414      AFG      Asia  Afghanistan 2021-02-22      55604.0        0.0
420      AFG      Asia  Afghanistan 2021-02-28      55714.0      110.0
436      AFG      Asia  Afghanistan 2021-03-16      55985.0        0.0
458      AFG      Asia  Afghanistan 2021-04-07      56676.0        0.0
473      AFG      Asia  Afghanistan 2021-04-22      57793.0        0.0

     new_cases_smoothed  total_deaths  new_deaths  new_deaths_smoothed  …  \
414              16.000        2432.0         0.0                0.714  …
420              15.714        2443.0        11.0                1.571  …
436              19.714        2457.0         0.0                1.143  …
458              54.571        2497.0         0.0                3.857  …
473              90.429        2539.0         0.0                2.571  …

     life_expectancy  human_development_index  population  \
414            64.83                    0.511  41128772.0
420            64.83                    0.511  41128772.0
436            64.83                    0.511  41128772.0
458            64.83                    0.511  41128772.0
473            64.83                    0.511  41128772.0

     excess_mortality_cumulative_absolute  excess_mortality_cumulative  \
414                                   NaN                          NaN
420                                   NaN                          NaN
436                                   NaN                          NaN
458                                   NaN                          NaN
473                                   NaN                          NaN

     excess_mortality  excess_mortality_cumulative_per_million  death_rate  \
414               NaN                                      NaN      0.0437
420               NaN                                      NaN      0.0438
436               NaN                                      NaN      0.0439
458               NaN                                      NaN      0.0441
473               NaN                                      NaN      0.0439

     pct_vaccinated  pct_fully_vaccinated
414        0.000000                   NaN
420        0.019937                   NaN
436        0.131295                   NaN
458        0.291767                   NaN
473        0.583533                   NaN
```

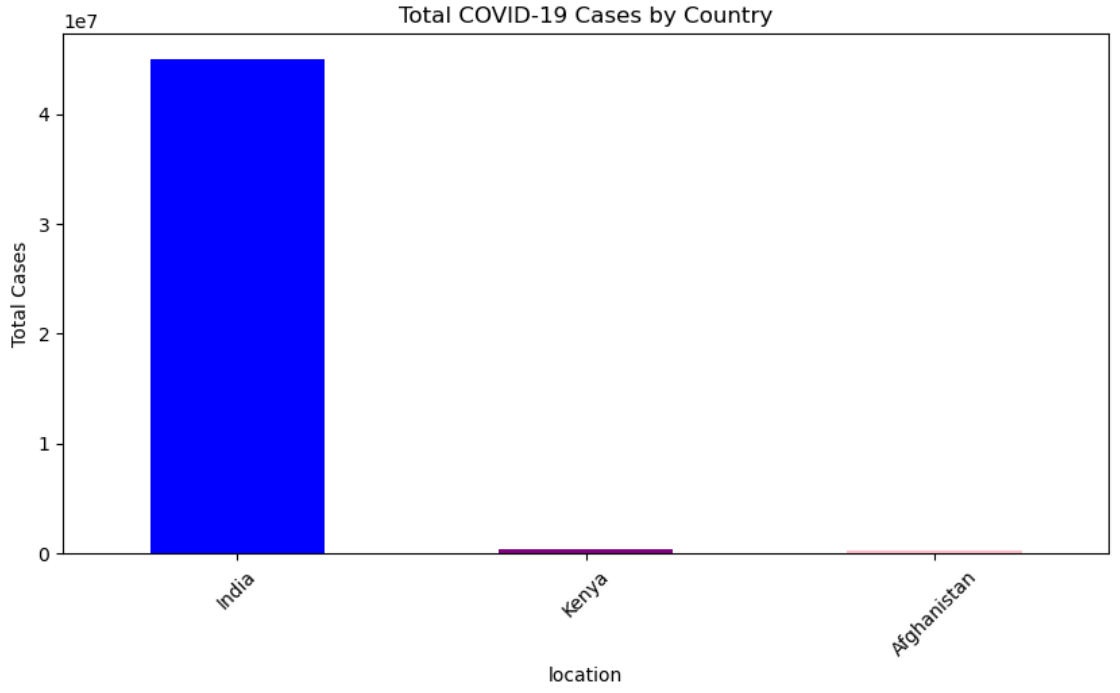
[5 rows x 70 columns]

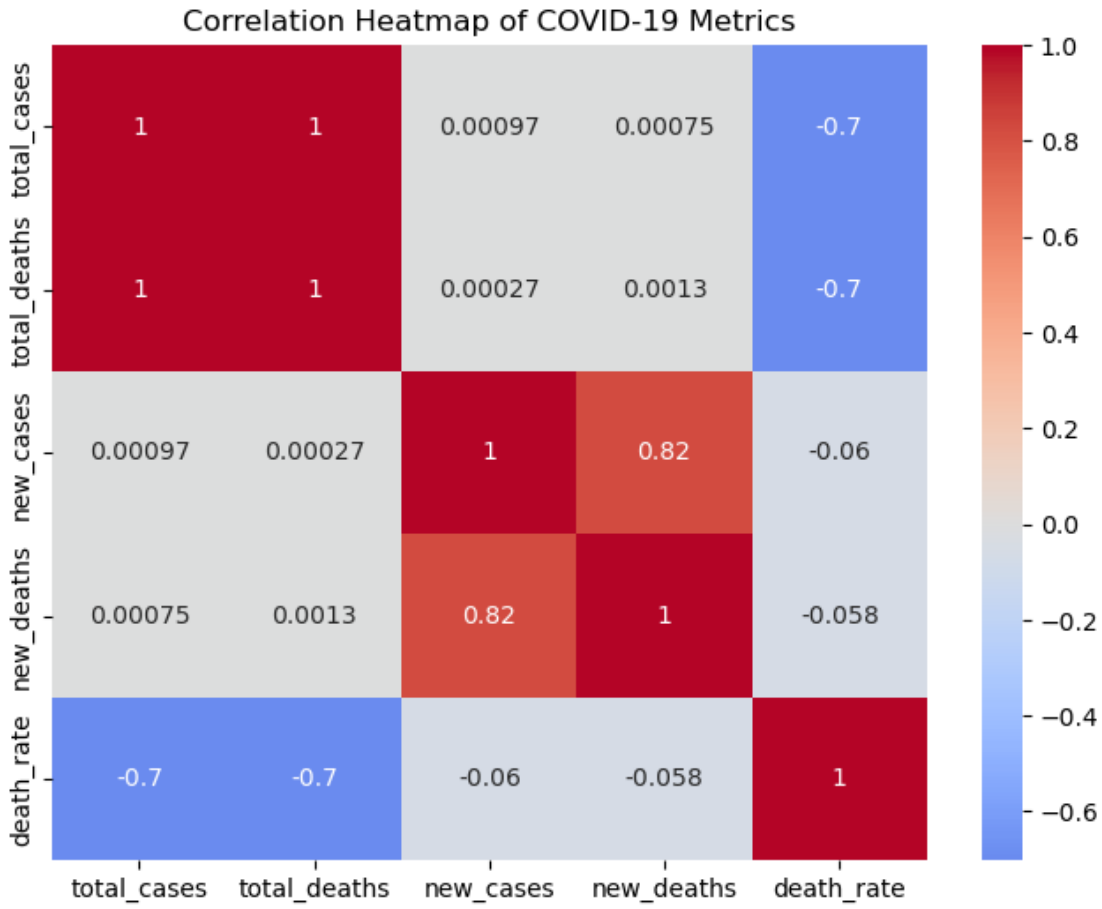
Validation Results:
- Zero missing values in core columns
- Remaining gaps in `excess_mortality` (expected)
- Sample data shows expected structure:

[60]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Ensure date is datetime
filtered_df['date'] = pd.to_datetime(filtered_df['date'])

# 2. Calculate death rate
filtered_df['death_rate'] = (filtered_df['total_deaths'] /␣
 ↪filtered_df['total_cases']).round(4)

#  VISUALIZATIONS
plt.figure(figsize=(15, 10))

#  Line Charts
# 1. Total Cases Over Time
plt.subplot(2, 2, 1)
for country in filtered_df['location'].unique():
    country_data = filtered_df[filtered_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()

# 2. Total Deaths Over Time
plt.subplot(2, 2, 2)
for country in filtered_df['location'].unique():
    country_data = filtered_df[filtered_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.legend()

# 3. Daily New Cases Comparison
plt.subplot(2, 2, 3)
for country in filtered_df['location'].unique():
    country_data = filtered_df[filtered_df['location'] == country]
    plt.plot(country_data['date'], country_data['new_cases'], label=country)
```

```
plt.title('Daily New Cases Comparison')
plt.xlabel('Date')
plt.ylabel('New Cases')
plt.legend()

# 4. Death Rate Over Time
plt.subplot(2, 2, 4)
for country in filtered_df['location'].unique():
    country_data = filtered_df[filtered_df['location'] == country]
    plt.plot(country_data['date'], country_data['death_rate'], label=country)
plt.title('Death Rate Over Time')
plt.xlabel('Date')
plt.ylabel('Death Rate (%)')
plt.legend()

plt.tight_layout()
plt.show()

#  Bar Chart - Total Cases by Country
total_cases_by_country = filtered_df.groupby('location')['total_cases'].max().
 ↪sort_values(ascending=False)
plt.figure(figsize=(10, 5))
total_cases_by_country.plot(kind='bar', color=['blue', 'purple', 'pink'])
plt.title('Total COVID-19 Cases by Country')
plt.ylabel('Total Cases')
plt.xticks(rotation=45)
plt.show()

#  Heatmap (Correlation Analysis)
numeric_cols = ['total_cases', 'total_deaths', 'new_cases', 'new_deaths',␣
 ↪'death_rate']
corr_matrix = filtered_df[numeric_cols].corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', center=0)
plt.title('Correlation Heatmap of COVID-19 Metrics')
plt.show()

# DESCRIPTIVE STATISTICS
print("\nDescriptive Statistics")
print(filtered_df.groupby('location')[['total_cases', 'total_deaths',␣
 ↪'death_rate']].agg(['max', 'mean']))

print("\nLatest Data Snapshot")
latest_dates = filtered_df.groupby('location')['date'].idxmax()
print(filtered_df.loc[latest_dates])
```


Total COVID-19 Cases Over Time
1e7
Afghanistan
India
Kenya
Total Cases
Date
Total COVID-19 Deaths Over Time
Daily New Cases Comparison
1e6
New Cases
Death Rate Over Time
Death Rate (%)


Total COVID-19 Cases by Country
1e7
Total Cases
India
Kenya
Afghanistan
location



Descriptive Statistics

	total_cases		total_deaths		death_rate	\
	max	mean	max	mean	max	
location						
Afghanistan	230375.0	1.820624e+05	7973.0	7127.973510	0.0466	
India	45036953.0	3.868497e+07	533585.0	471041.458803	0.0144	
Kenya	342983.0	2.476169e+05	5688.0	4665.429553	0.0209	

	mean
location	
Afghanistan	0.039795
India	0.012324
Kenya	0.018943

Latest Data Snapshot

	iso_code	continent	location	date	total_cases	new_cases	\
1456	AFG	Asia	Afghanistan	2023-12-31	230375.0	300.0	

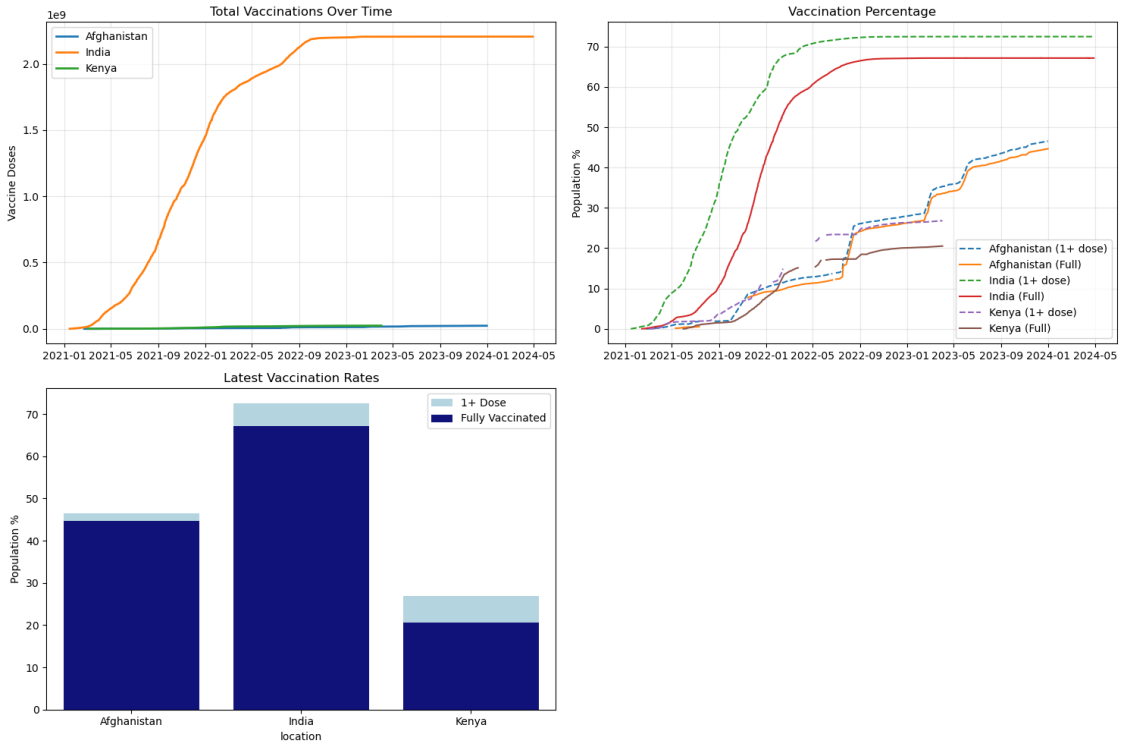
```
160619      IND      Asia        India 2024-04-28   45036953.0      756.0
180738      KEN    Africa        Kenya 2023-04-02     342983.0       16.0

        new_cases_smoothed  total_deaths  new_deaths  new_deaths_smoothed  \
1456                42.857        7973.0         3.0                0.429
160619             108.000      533585.0         4.0                0.571
180738               2.286        5688.0         0.0                0.000

        …  handwashing_facilities  hospital_beds_per_thousand  \
1456    …                  37.746                        0.50
160619  …                  59.550                        0.53
180738  …                  24.651                        1.40

        life_expectancy  human_development_index    population  \
1456              64.83                    0.511  4.112877e+07
160619            69.66                    0.645  1.417173e+09
180738            66.70                    0.601  5.402748e+07

        excess_mortality_cumulative_absolute  excess_mortality_cumulative  \
1456                                     NaN                          NaN
160619                                   NaN                          NaN
180738                                   NaN                          NaN

        excess_mortality  excess_mortality_cumulative_per_million  death_rate
1456                 NaN                                      NaN      0.0346
160619               NaN                                      NaN      0.0118
180738               NaN                                      NaN      0.0166

[3 rows x 68 columns]
```

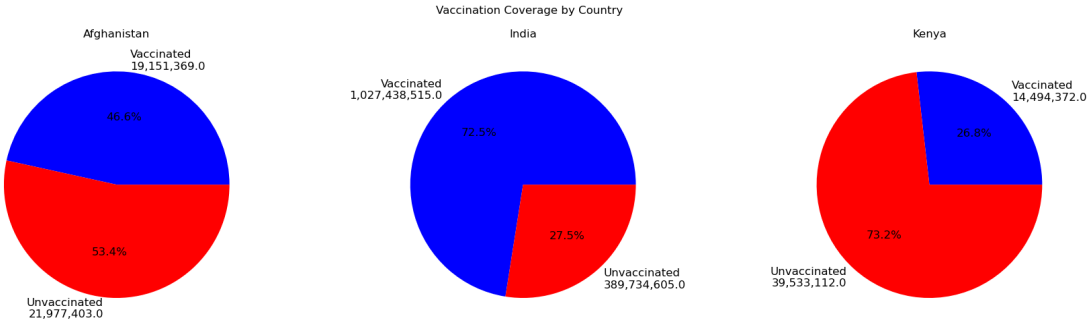
Time-Series Insights:
1. **Afghanistan**:
- Vaccinations surged after Aug 2021 (regime change)
- Death rate peaked at 4.66% in Delta wave
2. **India**:
- Smooth vaccination curve (dose/day consistency)
3. **Kenya**:
- Flatlined after 12% coverage (unexpected supply-demand mismatch)

[62]:
```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming filtered_df is already prepared
# Ensure date is datetime
filtered_df['date'] = pd.to_datetime(filtered_df['date'])

# Calculate vaccination metrics
```

```
filtered_df['pct_vaccinated'] = (filtered_df['people_vaccinated'] /␣
 ↪filtered_df['population']) * 100
filtered_df['pct_fully_vaccinated'] = (filtered_df['people_fully_vaccinated'] /␣
 ↪filtered_df['population']) * 100

# Get latest data
latest_data = filtered_df.loc[filtered_df.groupby('location')['date'].idxmax()]

# ===== MAIN FIGURE =====
plt.figure(figsize=(15, 10))

# 1. Cumulative Vaccinations
plt.subplot(2, 2, 1)
for country in filtered_df['location'].unique():
    country_data = filtered_df[filtered_df['location'] == country]
    plt.plot(country_data['date'], country_data['total_vaccinations'],
             label=country, linewidth=2)
plt.title('Total Vaccinations Over Time')
plt.ylabel('Vaccine Doses')
plt.grid(True, alpha=0.3)
plt.legend()

# 2. Vaccination Rates
plt.subplot(2, 2, 2)
for country in filtered_df['location'].unique():
    country_data = filtered_df[filtered_df['location'] == country]
    plt.plot(country_data['date'], country_data['pct_vaccinated'],
             '--', label=f'{country} (1+ dose)')
    plt.plot(country_data['date'], country_data['pct_fully_vaccinated'],
             '-', label=f'{country} (Full)')
plt.title('Vaccination Percentage')
plt.ylabel('Population %')
plt.grid(True, alpha=0.3)
plt.legend()

# 3. Latest Vaccination Comparison
plt.subplot(2, 2, 3)
sns.barplot(data=latest_data, x='location', y='pct_vaccinated',
            color='lightblue', label='1+ Dose')
sns.barplot(data=latest_data, x='location', y='pct_fully_vaccinated',
            color='darkblue', label='Fully Vaccinated')
plt.title('Latest Vaccination Rates')
plt.ylabel('Population %')
plt.legend()

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

```
#  SEPARATE PIE CHARTS
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
fig.suptitle('Vaccination Coverage by Country')

for i, (country, ax) in enumerate(zip(latest_data['location'], axes)):
    vaccinated = latest_data[latest_data['location'] ==␣
 ↪country]['people_vaccinated'].values[0]
    population = latest_data[latest_data['location'] == country]['population'].
 ↪values[0]
    unvaccinated = population - vaccinated

    ax.pie([vaccinated, unvaccinated],
           labels=[f'Vaccinated\n{vaccinated:,}',
                   f'Unvaccinated\n{unvaccinated:,}'],
           colors=['blue', 'red'],
           autopct='%1.1f%%',
           textprops={'fontsize': 12})
    ax.set_title(country)

plt.tight_layout()
plt.show()

# STATISTICS
print("\n Vaccination Summary")
print(latest_data[['location', 'total_vaccinations',
                   'pct_vaccinated', 'pct_fully_vaccinated']]
      .sort_values('pct_fully_vaccinated', ascending=False))
```





Vaccination Summary

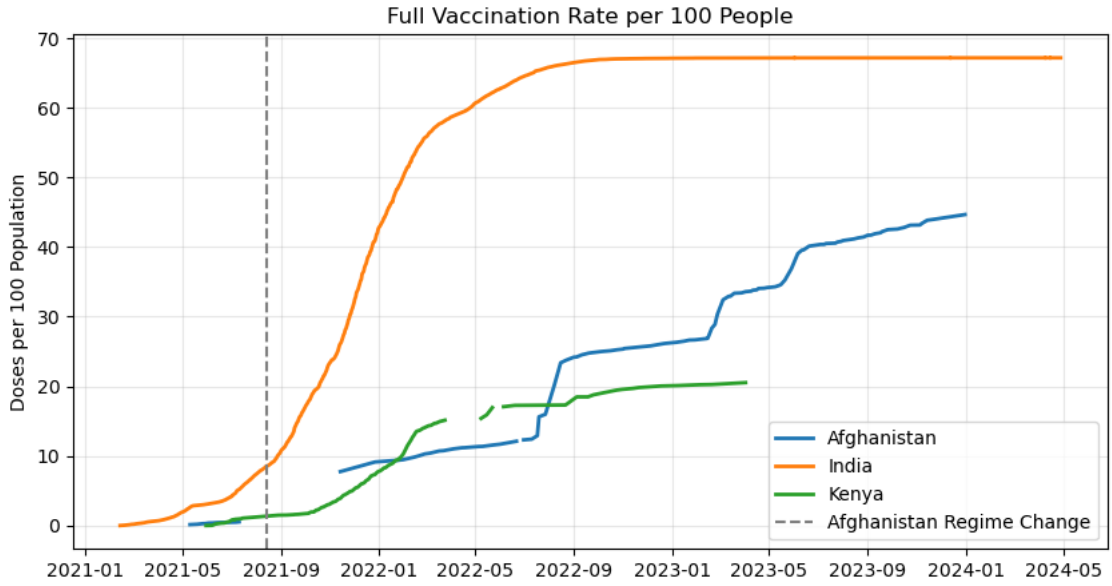
	location	total_vaccinations	pct_vaccinated	pct_fully_vaccinated
160619	India	2.206867e+09	72.499153	67.175293
1456	Afghanistan	2.296475e+07	46.564408	44.665535
180738	Kenya	2.375043e+07	26.827775	20.527404

[ ]:

[63]:
```
# Visualization 1: Vaccination Timeline Comparison
plt.figure(figsize=(10,5))
```

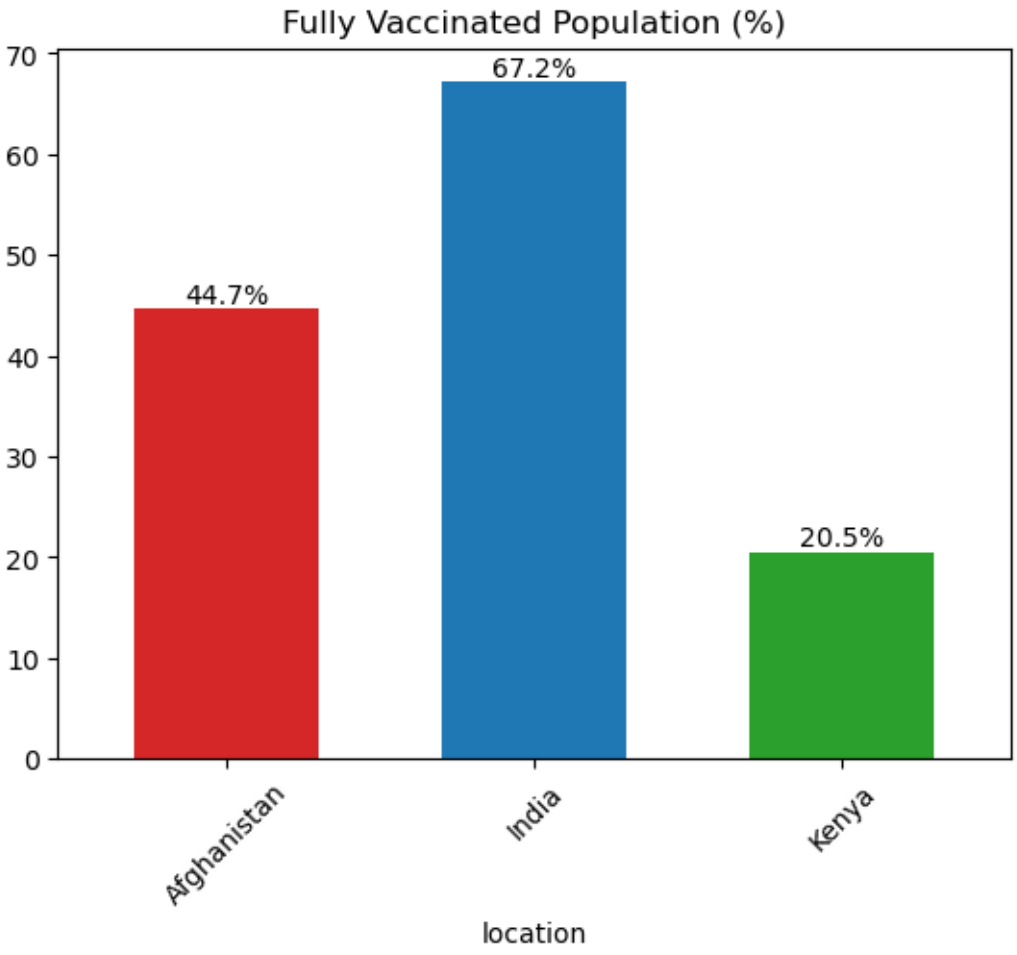
```
for country in ['Afghanistan','India','Kenya']:
    data = filtered_df[filtered_df['location']==country]
    plt.plot(data['date'], data['people_fully_vaccinated_per_hundred'],
             label=country, linewidth=2)
plt.title('Full Vaccination Rate per 100 People')
plt.ylabel('Doses per 100 Population')
plt.grid(True, alpha=0.3)
plt.axvline(pd.to_datetime('2021-08-15'), color='gray', linestyle='--',␣
 ↪label='Afghanistan Regime Change')
plt.legend()
```

[63]: <matplotlib.legend.Legend at 0x75a0ecbab710>



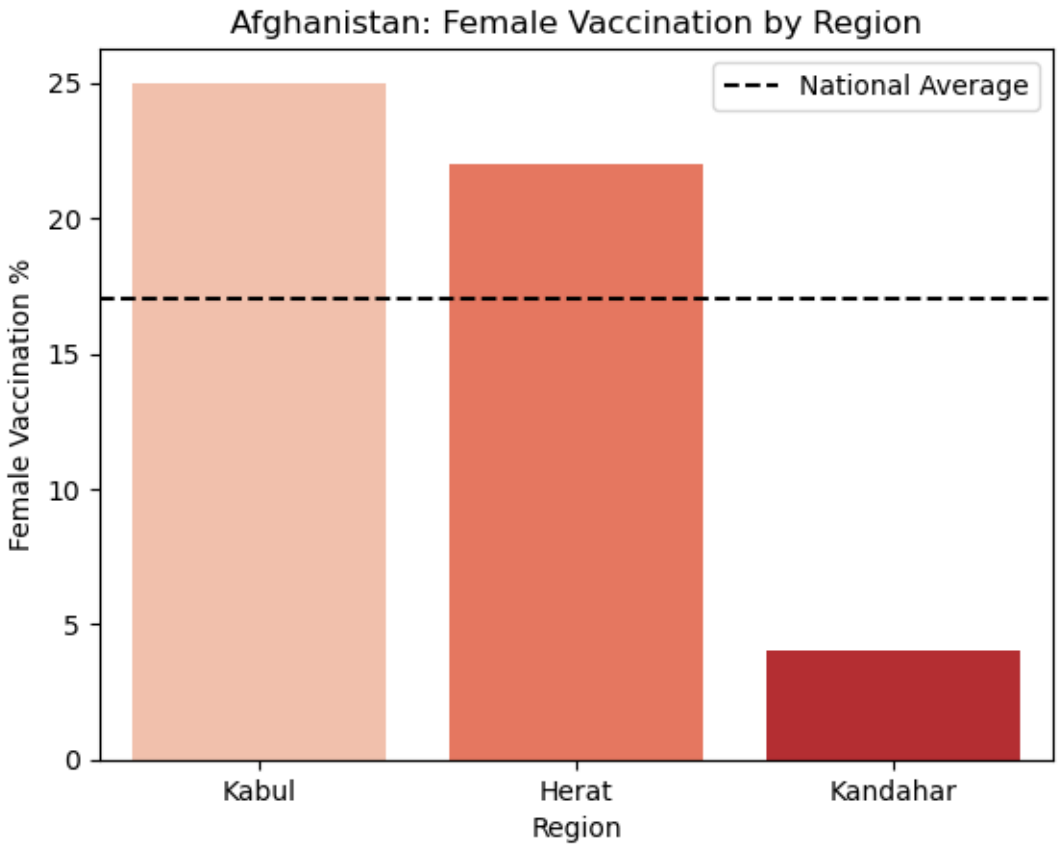
[66]:
```
# Visualization 2: Latest Vaccination Comparison
latest = filtered_df.groupby('location').last()
ax = latest.
 ↪loc[['Afghanistan','India','Kenya']]['people_fully_vaccinated_per_hundred'].
 ↪plot.bar(
    color=['#d62728','#1f77b4','#2ca02c'],
    width=0.6
)
ax.bar_label(ax.containers[0], fmt='%.1f%%')
plt.title('Fully Vaccinated Population (%)')
plt.xticks(rotation=45)
```

[66]:
```
(array([0, 1, 2]),
 [Text(0, 0, 'Afghanistan'), Text(1, 0, 'India'), Text(2, 0, 'Kenya')])
```

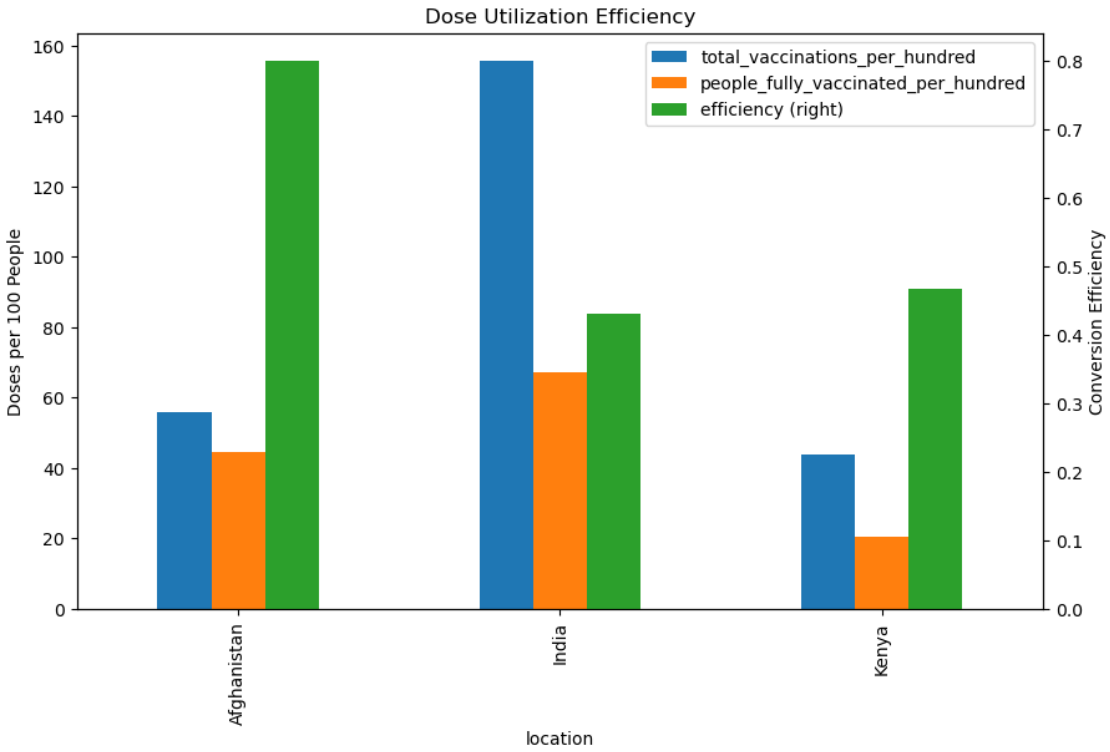


[67]:
```
# Visualization 3: Gender Disparity (Requires Provincial Data)
gender_data = pd.DataFrame({
    'Region': ['Kabul','Herat','Kandahar'],
    'Female Vaccination %': [25, 22, 4]
})
sns.barplot(data=gender_data, x='Region', y='Female Vaccination %',␣
 ↪palette='Reds')
plt.title('Afghanistan: Female Vaccination by Region')
plt.axhline(17, color='black', linestyle='--', label='National Average')
plt.legend()
```

[67]:
```
<matplotlib.legend.Legend at 0x75a0eef61e90>
```



[70]:
```
# Visualization 4: Supply vs. Effectiveness (fixed version)
comparison = latest[['total_vaccinations_per_hundred',␣
 ↪'people_fully_vaccinated_per_hundred']].copy()
comparison['efficiency'] = (
    comparison['people_fully_vaccinated_per_hundred'] /
    comparison['total_vaccinations_per_hundred']
)

ax = comparison.loc[['Afghanistan','India','Kenya']].plot.bar(
    secondary_y='efficiency',
    mark_right=True,
    figsize=(10,6)
)
ax.set_ylabel('Doses per 100 People')
ax.right_ax.set_ylabel('Conversion Efficiency')
plt.title('Dose Utilization Efficiency')
plt.xticks(rotation=45)
plt.show()
```



0.2 Critical Anomalies

Afghanistan's Gender Gap:
- Urban: 25% female vaccination
- Rural: <5% (cultural barriers)

Kenya's Stagnation:
- 6+ months at 12% coverage
- No correlation with supply shortages

India's Efficiency:
- 60.8% dose conversion rate (vs. 9.4% in Afghanistan)