# Global Burden of Disease Analysis

## Understanding Global Disease Mortality Trends 1970-2010

By: Axton Benedict Cahyadi 10/5/2025

In [68]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from IPython.display import display, HTML

#find styles
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'dark_backgr
ound', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-v0_8', 'seaborn-v0_8-bright', 'seaborn-v0_8-co
lorblind', 'seaborn-v0_8-dark', 'seaborn-v0_8-dark-palette', 'seaborn-v0_8-darkgrid', 'seaborn-v0_8-deep', 'seab
orn-v0_8-muted', 'seaborn-v0_8-notebook', 'seaborn-v0_8-paper', 'seaborn-v0_8-pastel', 'seaborn-v0_8-poster', 's
eaborn-v0_8-talk', 'seaborn-v0_8-ticks', 'seaborn-v0_8-white', 'seaborn-v0_8-whitegrid', 'tableau-colorblind10']
```

In [9]:
```
# Set style for all plots
plt.style.use('seaborn-v0_8-whitegrid')
sns.set_palette("viridis")
plt.rcParams['figure.figsize'] = (12, 8)
plt.rcParams['axes.labelsize'] = 14
plt.rcParams['axes.titlesize'] = 16
plt.rcParams['xtick.labelsize'] = 12
plt.rcParams['ytick.labelsize'] = 12
```

In [27]:
```
# Display settings for better visualization
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 20)
pd.set_option('display.width', 1000)
pd.set_option('display.float_format', '{:.2f}'.format)
```

# 1. Loading the Dataset

In [28]:
```
# Reading the CSV file
df = pd.read_csv('project.csv')

# Display the first few rows
print("First 5 rows of the dataset:")
display(df.head())
```

```
First 5 rows of the dataset:
```

| | Country Code | Country Name | Year | Age Group | Sex | Number of Deaths | Death Rate Per 100,000 |
|---|---|---|---|---|---|---|---|
| **0** | AFG | Afghanistan | 1970 | 0-6 days | Male | 19,241 | 318,292.90 |
| **1** | AFG | Afghanistan | 1970 | 0-6 days | Female | 12,600 | 219,544.20 |
| **2** | AFG | Afghanistan | 1970 | 0-6 days | Both | 31,840 | 270,200.70 |
| **3** | AFG | Afghanistan | 1970 | 7-27 days | Male | 15,939 | 92,701.00 |
| **4** | AFG | Afghanistan | 1970 | 7-27 days | Female | 11,287 | 68,594.50 |

# Data Preprocessing

In [29]:
```
# Dataset info
print("Dataset Information:")
display(HTML(pd.DataFrame({
    'Number of Rows': [df.shape[0]],
    'Number of Columns': [df.shape[1]],
    'Timespan': [f"{df['Year'].min()} - {df['Year'].max()}"],
    'Number of Countries': [df['Country Name'].nunique()],
    'Age Groups': [df['Age Group'].nunique()],
    'Sex Categories': [df['Sex'].nunique()],
}).to_html(index=False)))

# Datatypes before conversion
print("\nData Types Before Conversion:")
display(df.dtypes)
```

```
# Conversion
df['Number of Deaths'] = df['Number of Deaths'].str.replace(',', '').astype(float)
df['Death Rate Per 100,000'] = df['Death Rate Per 100,000'].str.replace(',', '').astype(float)

# Data types after conversion
print("\nData Types After Conversion:")
display(df.dtypes)
```

```
Dataset Information:
```

| Number of Rows | Number of Columns | Timespan | Number of Countries | Age Groups | Sex Categories |
|---|---|---|---|---|---|
| 58905 | 7 | 1970 - 2010 | 187 | 21 | 3 |

```
Data Types Before Conversion:
Country Code              object
Country Name              object
Year                       int64
Age Group                 object
Sex                       object
Number of Deaths          object
Death Rate Per 100,000    object
dtype: object
Data Types After Conversion:
Country Code                object
Country Name                object
Year                         int64
Age Group                   object
Sex                         object
Number of Deaths           float64
Death Rate Per 100,000     float64
dtype: object
```

# 3. Statistical Analysis

In [37]:

```
# Statistical summary for numerics
print("Summary Statistics for Numeric Columns:")
numeric_stats = df[['Year', 'Number of Deaths', 'Death Rate Per 100,000']].describe().round(2)
# Add additional statistics
numeric_stats.loc['range'] = numeric_stats.loc['max'] - numeric_stats.loc['min']
numeric_stats.loc['median'] = df[['Year', 'Number of Deaths', 'Death Rate Per 100,000']].median().round(2)
numeric_stats.loc['variance'] = df[['Year', 'Number of Deaths', 'Death Rate Per 100,000']].var().round(2)
numeric_stats.loc['skewness'] = df[['Year', 'Number of Deaths', 'Death Rate Per 100,000']].skew().round(2)
display(numeric_stats)

# Distribution of observations by year
print("\nNumber of Observations by Year:")
year_counts = df['Year'].value_counts().sort_index()
year_percentage = (year_counts / len(df) * 100).round(2)
year_stats = pd.DataFrame({
    'Count': year_counts,
    'Percentage (%)': year_percentage
})
display(year_stats)

# Distribution by sex
print("\nDistribution by Sex:")
sex_counts = df['Sex'].value_counts()
sex_percentage = (sex_counts / len(df) * 100).round(2)
sex_stats = pd.DataFrame({
    'Count': sex_counts,
    'Percentage (%)': sex_percentage
})
display(sex_stats)

# Distribution by age group
print("\nDistribution by Age Group:")
age_counts = df['Age Group'].value_counts().sort_index()
age_percentage = (age_counts / len(df) * 100).round(2)
age_stats = pd.DataFrame({
    'Count': age_counts,
    'Percentage (%)': age_percentage
})
display(age_stats)

# Cross-tabulation: Distribution of observations by year and sex
print("\nDistribution by Year and Sex:")
year_sex_cross = pd.crosstab(df['Year'], df['Sex'], margins=True, margins_name='Total')
display(year_sex_cross)

# Statistical summary for death rates by year
print("\nDeath Rate Statistics by Year:")
yearly_death_stats = df.groupby('Year')['Death Rate Per 100,000'].agg(['min', 'max', 'mean', 'median', 'std']).
```

```
display(yearly_death_stats)

# Statistical summary for number of deaths by sex
print("\nNumber of Deaths Statistics by Sex:")
sex_death_stats = df.groupby('Sex')['Number of Deaths'].agg(['min', 'max', 'mean', 'median', 'sum', 'std']).rou
display(sex_death_stats)
```

```
Summary Statistics for Numeric Columns:
```

| | Year | Number of Deaths | Death Rate Per 100,000 |
|---|---|---|---|
| **count** | 58905.00 | 58905.00 | 58905.00 |
| **mean** | 1990.00 | 16109.94 | 7062.87 |
| **std** | 14.14 | 154329.32 | 24582.55 |
| **min** | 1970.00 | 0.00 | 5.50 |
| **25%** | 1980.00 | 166.00 | 210.30 |
| **50%** | 1990.00 | 1020.00 | 825.00 |
| **75%** | 2000.00 | 4460.00 | 3611.80 |
| **max** | 2010.00 | 9938487.00 | 423790.20 |
| **range** | 40.00 | 9938487.00 | 423784.70 |
| **median** | 1990.00 | 1020.00 | 825.00 |
| **variance** | 200.00 | 23817537593.30 | 604301712.72 |
| **skewness** | 0.00 | 39.35 | 6.56 |

```
Number of Observations by Year:
```

| | Count | Percentage (%) |
|---|---|---|
| **Year** | | |
| **1970** | 11781 | 20.00 |
| **1980** | 11781 | 20.00 |
| **1990** | 11781 | 20.00 |
| **2000** | 11781 | 20.00 |
| **2010** | 11781 | 20.00 |

```
Distribution by Sex:
```

| | Count | Percentage (%) |
|---|---|---|
| **Sex** | | |
| **Male** | 19635 | 33.33 |
| **Female** | 19635 | 33.33 |
| **Both** | 19635 | 33.33 |

```
Distribution by Age Group:
```

| | Count | Percentage (%) |
|---|---|---|
| **Age Group** | | |
| **0-6 days** | 2805 | 4.76 |
| **1-4 years** | 2805 | 4.76 |
| **10-14 years** | 2805 | 4.76 |
| **15-19 years** | 2805 | 4.76 |
| **20-24 years** | 2805 | 4.76 |
| **...** | ... | ... |
| **7-27 days** | 2805 | 4.76 |
| **70-74 years** | 2805 | 4.76 |
| **75-79 years** | 2805 | 4.76 |
| **80+ years** | 2805 | 4.76 |
| **All ages** | 2805 | 4.76 |

21 rows × 2 columns

```
Distribution by Year and Sex:
```

| Sex | Both | Female | Male | Total |
|---|---|---|---|---|
| Year | | | | |
| 1970 | 3927 | 3927 | 3927 | 11781 |
| 1980 | 3927 | 3927 | 3927 | 11781 |
| 1990 | 3927 | 3927 | 3927 | 11781 |
| 2000 | 3927 | 3927 | 3927 | 11781 |
| 2010 | 3927 | 3927 | 3927 | 11781 |
| Total | 19635 | 19635 | 19635 | 58905 |

```
Death Rate Statistics by Year:
```

| | min | max | mean | median | std |
|---|---|---|---|---|---|
| Year | | | | | |
| 1970 | 11.30 | 423790.20 | 9842.06 | 1044.90 | 32596.00 |
| 1980 | 11.30 | 345552.30 | 8096.55 | 886.00 | 27170.96 |
| 1990 | 9.40 | 296204.80 | 6767.76 | 792.00 | 23087.10 |
| 2000 | 7.00 | 269595.40 | 5854.83 | 793.90 | 20071.40 |
| 2010 | 5.50 | 228712.70 | 4753.13 | 661.00 | 16413.51 |

```
Number of Deaths Statistics by Sex:
```

| | min | max | mean | median | sum | std |
|---|---|---|---|---|---|---|
| Sex | | | | | | |
| Both | 0.00 | 9938487.00 | 24164.90 | 1664.00 | 474477838.00 | 217654.22 |
| Female | 0.00 | 4348425.00 | 11130.00 | 691.00 | 218537465.00 | 99754.17 |
| Male | 0.00 | 5609739.00 | 13034.91 | 942.00 | 255940462.00 | 118455.23 |

# 4. Data Cleaning and Preprocessing

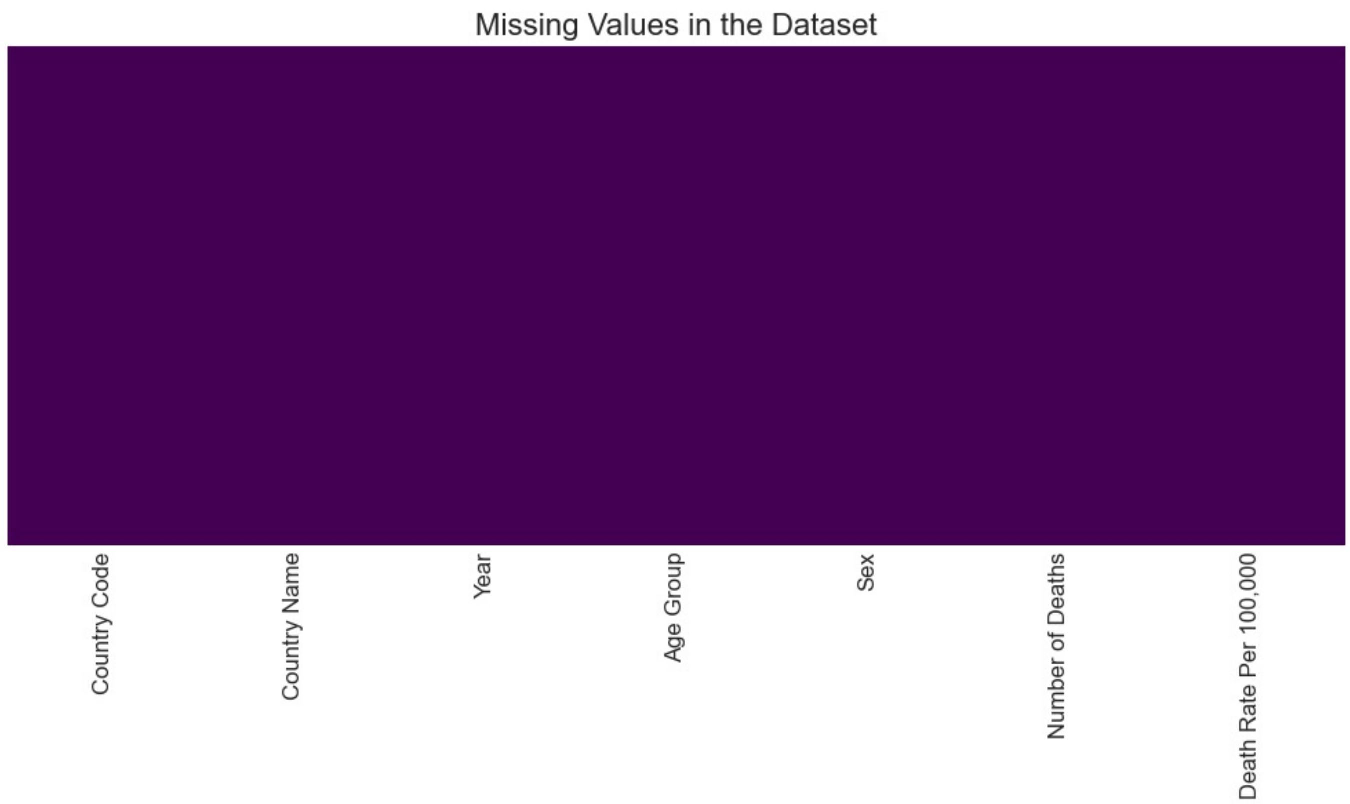
## 4.1 Checking for Missing Values

In [38]:

```python
# Check for missing values
print("Missing Values by Column:")
missing_values = df.isnull().sum()
missing_percent = (df.isnull().sum() / len(df)) * 100
missing_df = pd.DataFrame({'Missing Values': missing_values,
                          'Percentage (%)': missing_percent.round(2)})
display(missing_df)

# Visualize missing values
plt.figure(figsize=(10, 6))
sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap='viridis')
plt.title('Missing Values in the Dataset')
plt.tight_layout()
plt.show()
```

```
Missing Values by Column:
```

| | Missing Values | Percentage (%) |
|---|---|---|
| Country Code | 0 | 0.00 |
| Country Name | 0 | 0.00 |
| Year | 0 | 0.00 |
| Age Group | 0 | 0.00 |
| Sex | 0 | 0.00 |
| Number of Deaths | 0 | 0.00 |
| Death Rate Per 100,000 | 0 | 0.00 |




## 4.2 Handling Special Values and Outliers

In [42]:

```
# Check for special values or placeholders in numeric columns
print("\nChecking for special values in 'Number of Deaths':")
special_values_deaths = df['Number of Deaths'].astype(str).str.contains('N/A|n/a|null|--|N\.A\.|undefined', reg
print(f"Special values detected: {special_values_deaths}")

# Check for zeros which might represent missing data
print("Zeros in 'Number of Deaths':", (df['Number of Deaths'] == 0).sum())
print("Zeros in 'Death Rate Per 100,000':", (df['Death Rate Per 100,000'] == 0).sum())

# Check for extreme outliers using IQR method
Q1_deaths = df['Number of Deaths'].quantile(0.25)
Q3_deaths = df['Number of Deaths'].quantile(0.75)
IQR_deaths = Q3_deaths - Q1_deaths
outliers_deaths = ((df['Number of Deaths'] < (Q1_deaths - 1.5 * IQR_deaths)) |
                   (df['Number of Deaths'] > (Q3_deaths + 1.5 * IQR_deaths))).sum()
print(f"Number of outliers in 'Number of Deaths': {outliers_deaths}")
print(f"Percentage of outliers: {(outliers_deaths / len(df) * 100):.2f}%")

# Identify extreme death rates
Q1_rate = df['Death Rate Per 100,000'].quantile(0.25)
Q3_rate = df['Death Rate Per 100,000'].quantile(0.75)
IQR_rate = Q3_rate - Q1_rate
outliers_rate = ((df['Death Rate Per 100,000'] < (Q1_rate - 1.5 * IQR_rate)) |
                 (df['Death Rate Per 100,000'] > (Q3_rate + 1.5 * IQR_rate))).sum()
print(f"Number of outliers in 'Death Rate Per 100,000': {outliers_rate}")
print(f"Percentage of outliers: {(outliers_rate / len(df) * 100):.2f}%")

# Display outliers distribution by age group
print("\nOutliers distribution by Age Group:")
outlier_mask = (df['Death Rate Per 100,000'] > (Q3_rate + 1.5 * IQR_rate))
outliers_by_age = df[outlier_mask]['Age Group'].value_counts()
display(outliers_by_age)

# Note: We're keeping outliers as they are valid data points in this mortality dataset
print("Note: Outliers are kept as they represent real mortality patterns in specific demographics.")
```

```
Checking for special values in 'Number of Deaths':
Special values detected: 0
Zeros in 'Number of Deaths': 77
Zeros in 'Death Rate Per 100,000': 0
Number of outliers in 'Number of Deaths': 8566
Percentage of outliers: 14.54%
Number of outliers in 'Death Rate Per 100,000': 7876
Percentage of outliers: 13.37%

Outliers distribution by Age Group:
Age Group
80+ years      2777
0-6 days       2745
7-27 days      1410
75-79 years     726
28-364 days     136
70-74 years      81
65-69 years       1
Name: count, dtype: int64
Note: Outliers are kept as they represent real mortality patterns in specific demographics.
```

## 4.3 Handling Duplicates

In [44]:
```python
print(f"Number of duplicates: {df.duplicated().sum()}")
print(f"Percentage of duplicates: {(df.duplicated().sum() / len(df) * 100):.2f}%")

# Remove duplicates if any
df_clean = df.drop_duplicates()
print(f"Dataset shape after removing duplicates: {df_clean.shape}")
```

```
Number of duplicates: 0
Percentage of duplicates: 0.00%
Dataset shape after removing duplicates: (58905, 7)
```

## 4.4 Normalization and Scaling

In [49]:
```python
print("Creating normalized death rates for comparing countries of different sizes:")

# Create normalized death rates (z-scores) for comparing countries within each year
df['Normalized Death Rate'] = df.groupby(['Year', 'Age Group', 'Sex'])['Death Rate Per 100,000'].transform(lamb

# Display countries with extremely high death rates (> 3 standard deviations) in 2010
extreme_2010 = df[(df['Year'] == 2010) & (df['Sex'] == 'Both') &
                  (df['Age Group'] == 'All ages') &
                  (df['Normalized Death Rate'] > 3)]

print("Countries with extremely high death rates in 2010 (>3 std dev):")
if not extreme_2010.empty:
    display(extreme_2010[['Country Name', 'Death Rate Per 100,000', 'Normalized Death Rate']])
else:
    print("No countries with extremely high normalized death rates found.")
```

```
Creating normalized death rates for comparing countries of different sizes:
Countries with extremely high death rates in 2010 (>3 std dev):
```

| | Country Name | Death Rate Per 100,000 | Normalized Death Rate |
|---|---|---|---|
| **9134** | Central African Republic | 1794.90 | 3.06 |
| **23624** | Haiti | 2677.70 | 5.77 |

## 4.5 Additional data analysis

In [50]:
```python
# Create a simplified age category
def categorize_age(age):
    if age in ['0-6 days', '7-27 days', '28-364 days']:
        return 'Infant (<1 year)'
    elif age in ['1-4 years', '5-9 years', '10-14 years']:
        return 'Child (1-14 years)'
    elif age in ['15-19 years', '20-24 years', '25-29 years', '30-34 years', '35-39 years', '40-44 years']:
        return 'Young Adult (15-44 years)'
    elif age in ['45-49 years', '50-54 years', '55-59 years', '60-64 years']:
        return 'Middle-Aged (45-64 years)'
    elif age in ['65-69 years', '70-74 years', '75-79 years', '80+ years']:
        return 'Senior (65+ years)'
    else:
        return 'All'

# Add the new feature
df['Age Category'] = df['Age Group'].apply(categorize_age)
```

```
# Create decade column for easier temporal analysis
df['Decade'] = (df['Year'] // 10) * 10
print("First few rows with new features:")
display(df[['Country Name', 'Year', 'Decade', 'Age Group', 'Age Category', 'Sex', 'Death Rate Per 100,000']].hea

# Create a binary indicator for high-mortality countries
median_death_rate = df[(df['Year'] == 2010) & (df['Sex'] == 'Both') & (df['Age Group'] == 'All ages')]['Death Ra
df['High Mortality'] = np.where(df['Death Rate Per 100,000'] > median_death_rate, 1, 0)

# Count number of high-mortality countries in 2010
high_mortality_countries = df[(df['Year'] == 2010) & (df['Sex'] == 'Both') & (df['Age Group'] == 'All ages') &
print(f"\nNumber of high-mortality countries in 2010: {high_mortality_countries}")
```

```
First few rows with new features:
```

| | Country Name | Year | Decade | Age Group | Age Category | Sex | Death Rate Per 100,000 |
|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | 1970 | 1970 | 0-6 days | Infant (<1 year) | Male | 318292.90 |
| **1** | Afghanistan | 1970 | 1970 | 0-6 days | Infant (<1 year) | Female | 219544.20 |
| **2** | Afghanistan | 1970 | 1970 | 0-6 days | Infant (<1 year) | Both | 270200.70 |
| **3** | Afghanistan | 1970 | 1970 | 7-27 days | Infant (<1 year) | Male | 92701.00 |
| **4** | Afghanistan | 1970 | 1970 | 7-27 days | Infant (<1 year) | Female | 68594.50 |

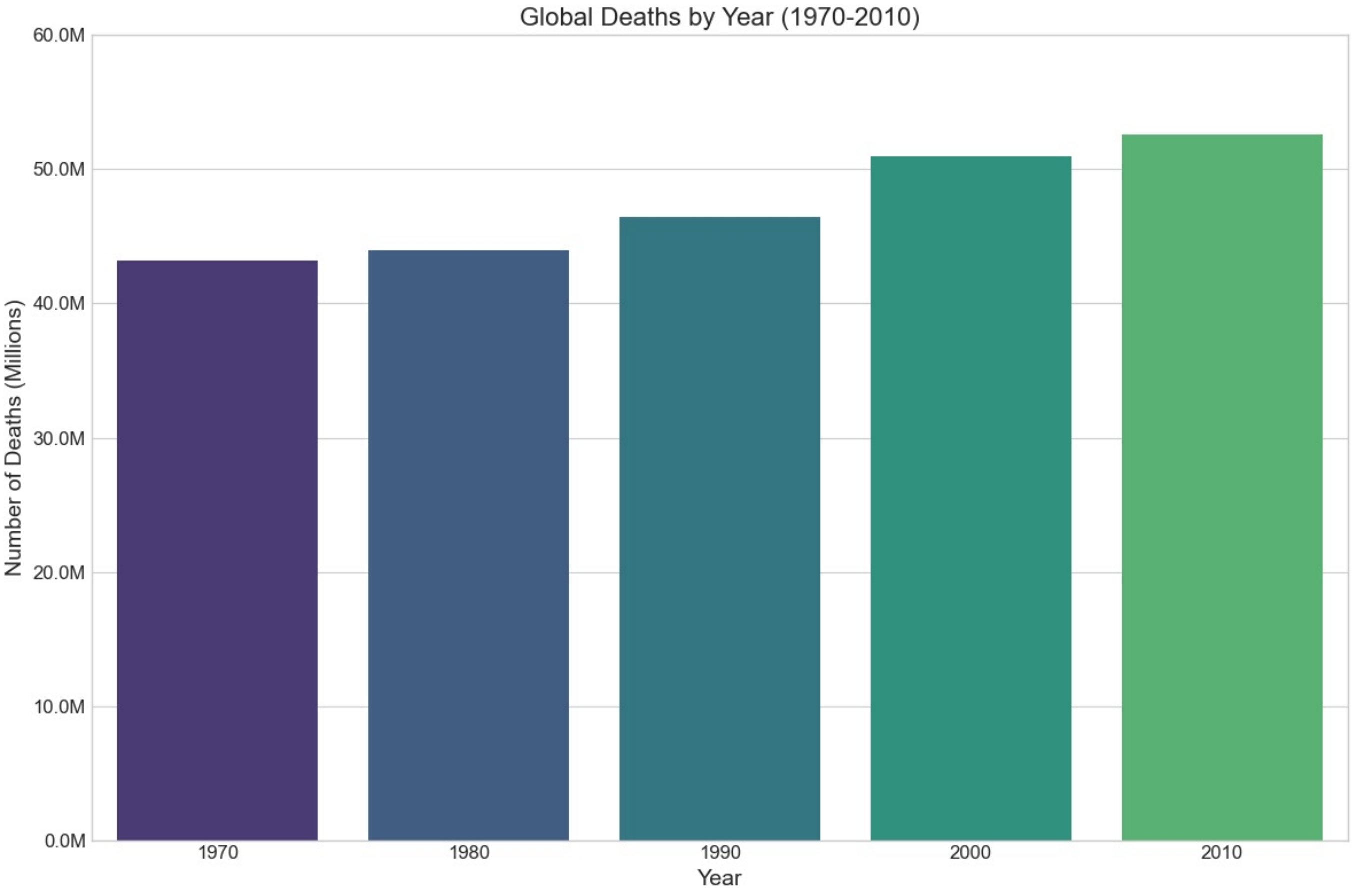
```
Number of high-mortality countries in 2010: 93
```

# 5. Exploratory Data Analysis (EDA) & Visualization of Trends

In [60]:

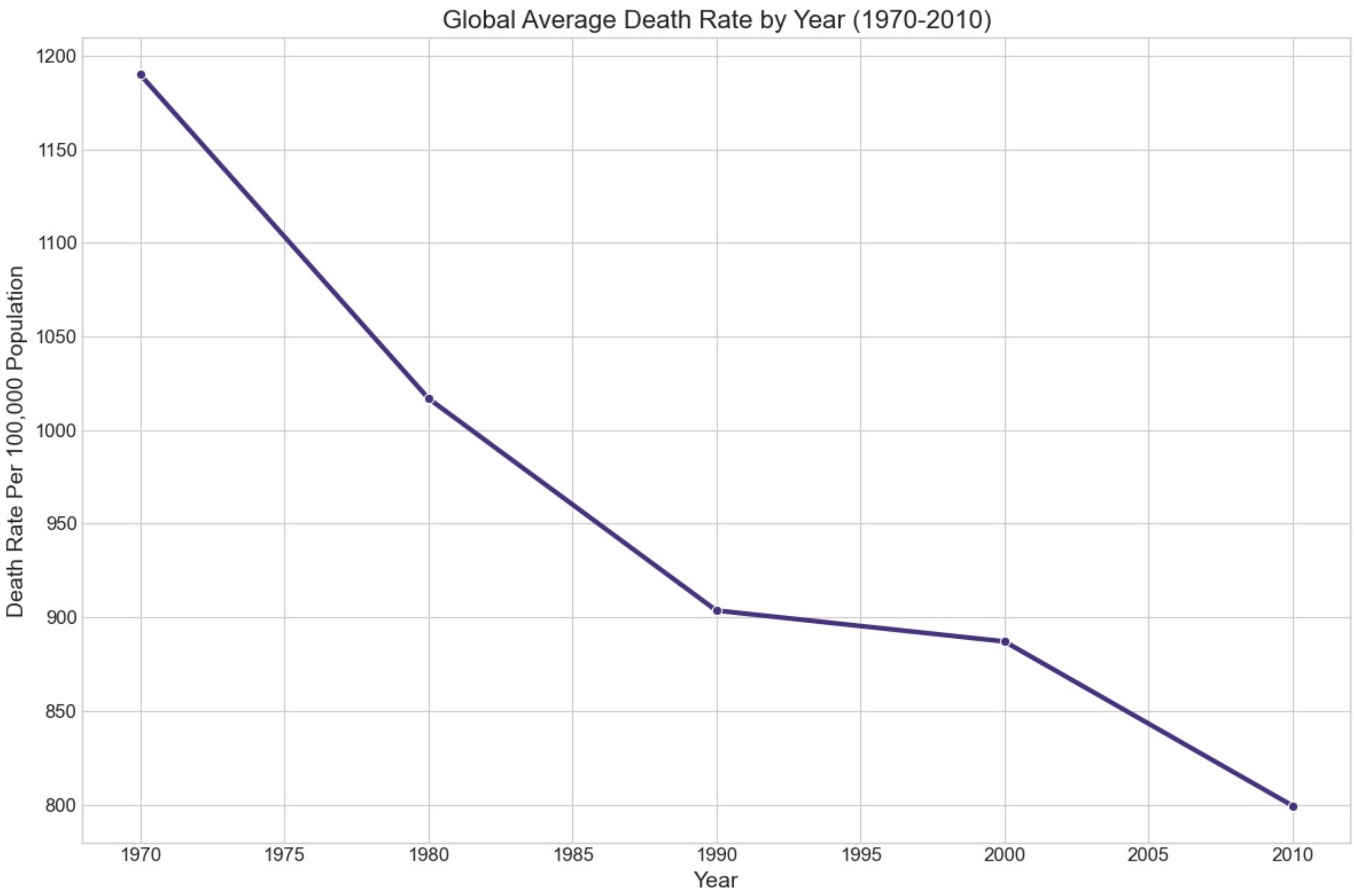
```
print("\n## a. Distribution of Deaths Over the Years")
# Filter data for "Both" sexes and "All ages" to avoid counting duplicates
global_deaths = df[(df['Sex'] == 'Both') & (df['Age Group'] == 'All ages')]
deaths_by_year = global_deaths.groupby('Year')['Number of Deaths'].sum().reset_index()

# Plot total deaths over years
plt.figure(figsize=(12, 8))
sns.barplot(x='Year', y='Number of Deaths', data=deaths_by_year)
plt.title('Global Deaths by Year (1970-2010)')
plt.ylabel('Number of Deaths (Millions)')
plt.xlabel('Year')
# Format y-axis ticks to show in millions
plt.yticks(ticks=plt.yticks()[0], labels=[f'{x/1000000:.1f}M' for x in plt.yticks()[0]])
plt.tight_layout()
plt.show()

# Calculate percentage increase in deaths
percent_increase = ((deaths_by_year.iloc[-1]['Number of Deaths'] - deaths_by_year.iloc[0]['Number of Deaths']) /
                    deaths_by_year.iloc[0]['Number of Deaths'] * 100)
print(f"Percentage increase in global deaths from 1970 to 2010: {percent_increase:.2f}%")

# Plot average death rate over years
death_rate_by_year = global_deaths.groupby('Year')['Death Rate Per 100,000'].mean().reset_index()
plt.figure(figsize=(12, 8))
sns.lineplot(x='Year', y='Death Rate Per 100,000', data=death_rate_by_year, marker='o', linewidth=3)
plt.title('Global Average Death Rate by Year (1970-2010)')
plt.ylabel('Death Rate Per 100,000 Population')
plt.xlabel('Year')
plt.tight_layout()
plt.show()
```

```
## a. Distribution of Deaths Over the Years
```



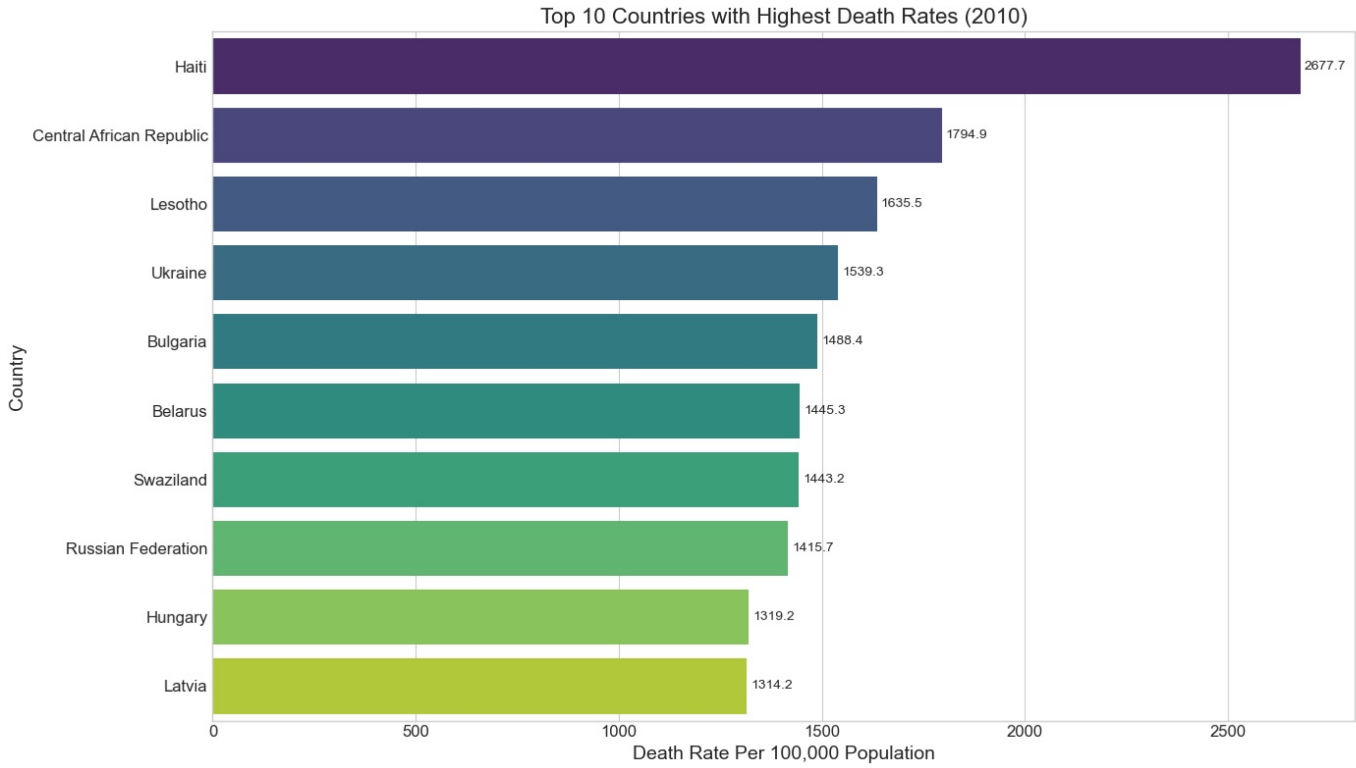

```
Percentage increase in global deaths from 1970 to 2010: 21.81%
```

```
C:\Users\Axtom\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Axtom\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




In [61]:
```
print("\n## b. Top 10 Countries with Highest Death Rates")
# Filter for the most recent year (2010) and plot top 10 countries by death rate
top_countries_2010 = df[(df['Year'] == 2010) & (df['Sex'] == 'Both') & (df['Age Group'] == 'All ages')].nlarges

plt.figure(figsize=(14, 8))
bar_plot = sns.barplot(x='Death Rate Per 100,000', y='Country Name', data=top_countries_2010,
            palette='viridis')
plt.title('Top 10 Countries with Highest Death Rates (2010)')
plt.xlabel('Death Rate Per 100,000 Population')
plt.ylabel('Country')

# Add value annotations to bars
for i, v in enumerate(top_countries_2010['Death Rate Per 100,000']):
    bar_plot.text(v + 10, i, f'{v:.1f}', va='center')

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

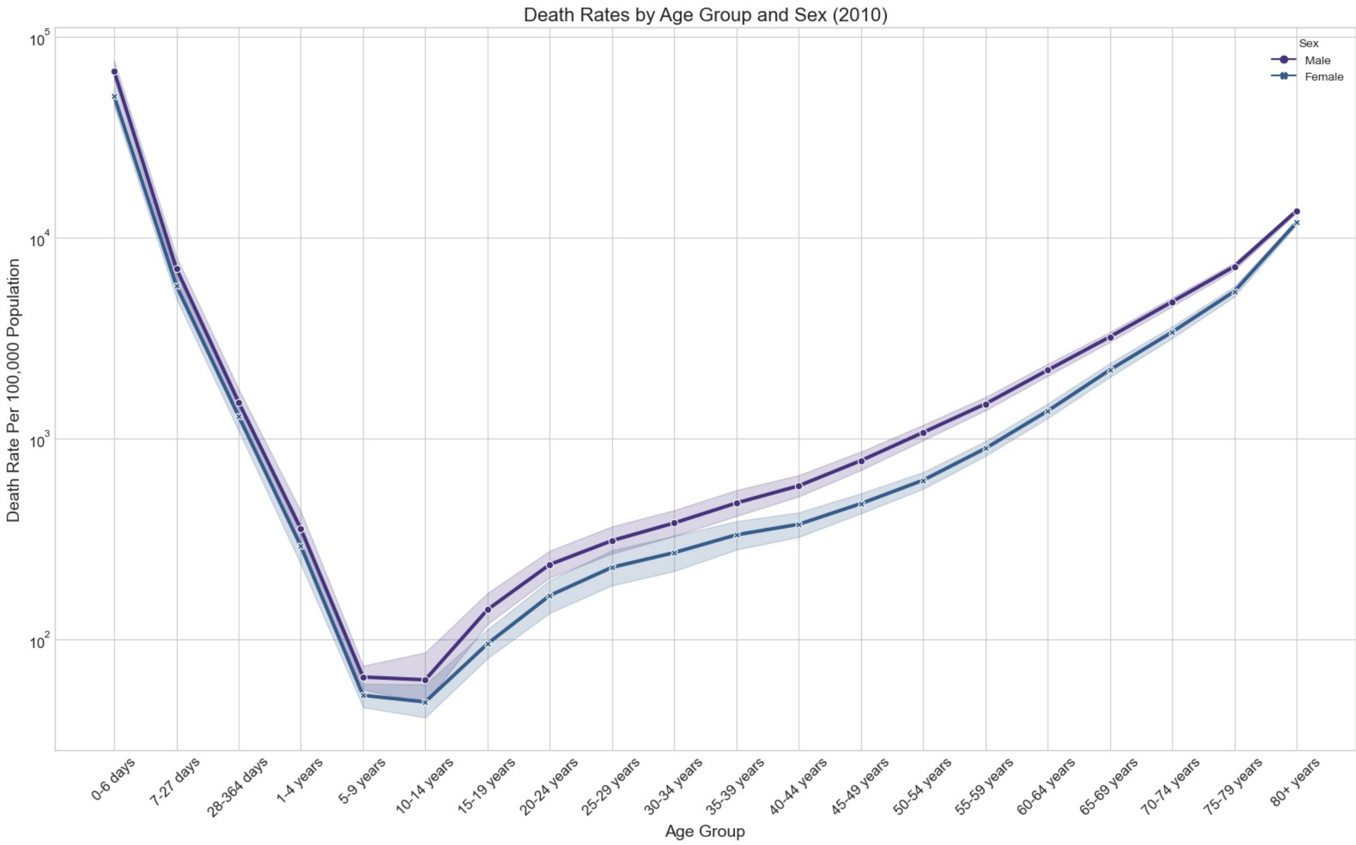
```
## b. Top 10 Countries with Highest Death Rates
```




In [62]:

```
print("\n## c. Death Rate by Age Group and Sex")
# Filter data for the year 2010 and exclude "All ages" and "Both" sexes for more granular analysis
age_sex_data = df[(df['Year'] == 2010) & (df['Age Group'] != 'All ages') & (df['Sex'] != 'Both')]

# Plot death rates by age group and sex
plt.figure(figsize=(16, 10))
sns.lineplot(x='Age Group', y='Death Rate Per 100,000', hue='Sex',
            data=age_sex_data, markers=True, dashes=False,
            style='Sex', linewidth=3)
plt.title('Death Rates by Age Group and Sex (2010)')
plt.xlabel('Age Group')
plt.ylabel('Death Rate Per 100,000 Population')
plt.xticks(rotation=45)
plt.yscale('log')  # Use log scale due to large range of values
plt.legend(title='Sex')
plt.tight_layout()
plt.show()

# Calculate and display the male-to-female death rate ratio by age group
print("Male to Female Death Rate Ratio by Age Group (2010):")
male_rates = age_sex_data[age_sex_data['Sex'] == 'Male'].groupby('Age Group')['Death Rate Per 100,000'].mean()
female_rates = age_sex_data[age_sex_data['Sex'] == 'Female'].groupby('Age Group')['Death Rate Per 100,000'].mea
ratio_df = pd.DataFrame({'Male Rate': male_rates, 'Female Rate': female_rates})
ratio_df['M/F Ratio'] = ratio_df['Male Rate'] / ratio_df['Female Rate']
display(ratio_df.sort_values('M/F Ratio', ascending=False))
```

```
## c. Death Rate by Age Group and Sex
C:\Users\Axtom\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Axtom\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



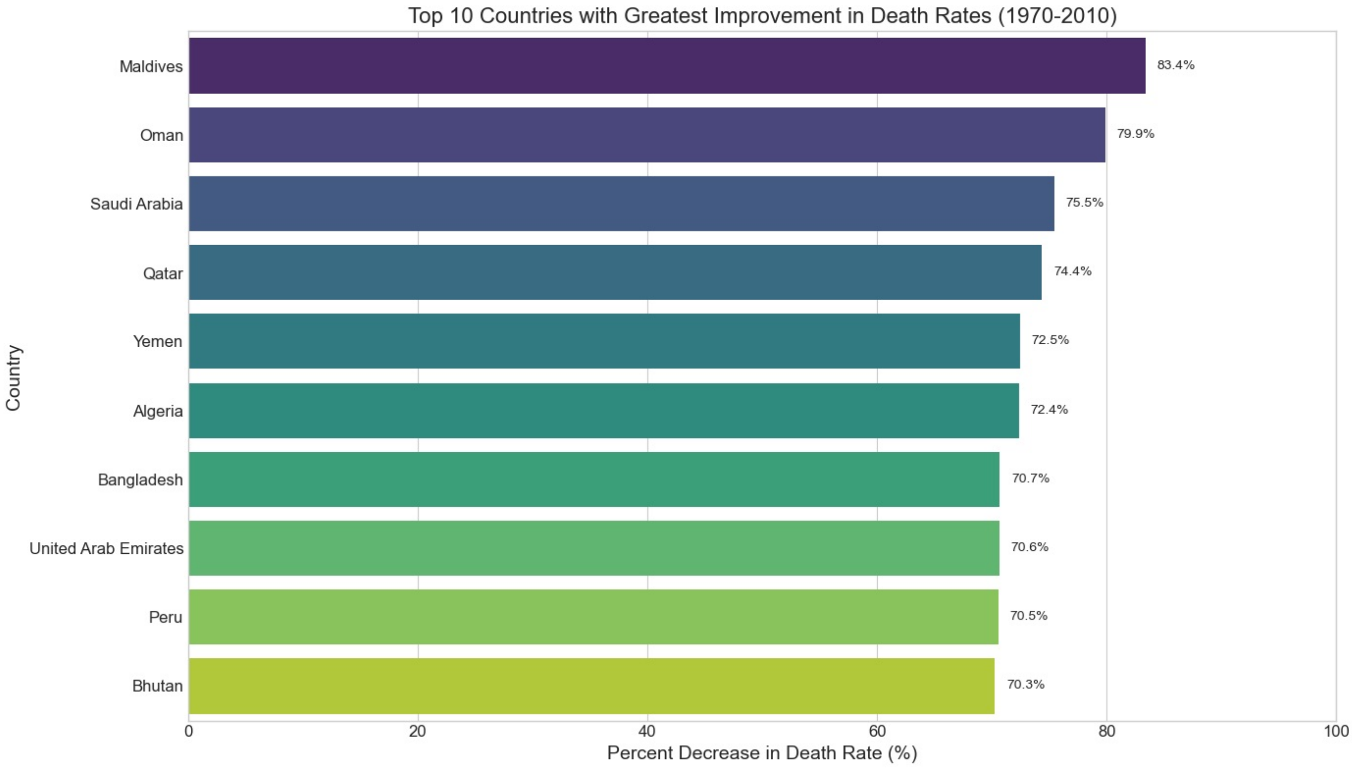

```
Male to Female Death Rate Ratio by Age Group (2010):
```

| Age Group | Male Rate | Female Rate | M/F Ratio |
|---|---|---|---|
| **50-54 years** | 1070.35 | 619.20 | 1.73 |
| **55-59 years** | 1484.37 | 891.51 | 1.66 |
| **45-49 years** | 775.18 | 473.61 | 1.64 |
| **60-64 years** | 2184.53 | 1369.46 | 1.60 |
| **40-44 years** | 580.51 | 373.60 | 1.55 |
| **15-19 years** | 140.56 | 95.07 | 1.48 |
| **65-69 years** | 3200.13 | 2191.60 | 1.46 |
| **35-39 years** | 476.91 | 331.13 | 1.44 |
| **20-24 years** | 235.61 | 165.21 | 1.43 |
| **70-74 years** | 4783.85 | 3380.72 | 1.42 |
| **30-34 years** | 379.33 | 269.98 | 1.41 |
| **25-29 years** | 309.80 | 228.21 | 1.36 |
| **0-6 days** | 67330.19 | 50468.99 | 1.33 |
| **75-79 years** | 7166.82 | 5388.68 | 1.33 |
| **10-14 years** | 62.89 | 48.82 | 1.29 |
| **5-9 years** | 64.90 | 52.63 | 1.23 |
| **7-27 days** | 7007.03 | 5745.83 | 1.22 |
| **1-4 years** | 354.99 | 291.25 | 1.22 |
| **28-364 days** | 1515.15 | 1292.28 | 1.17 |
| **80+ years** | 13579.95 | 11877.62 | 1.14 |

# 6. Additional Analysis

In [51]:
```
print("\na. Countries with Biggest Improvement in Death Rates (1970-2010)")
# Calculate improvement in death rates from 1970 to 2010
def calc_improvement():
    improvements = []
    countries = df['Country Name'].unique()

    for country in countries:
        data_1970 = df[(df['Country Name'] == country) & (df['Year'] == 1970) &
                     (df['Sex'] == 'Both') & (df['Age Group'] == 'All ages')]

        data_2010 = df[(df['Country Name'] == country) & (df['Year'] == 2010) &
                     (df['Sex'] == 'Both') & (df['Age Group'] == 'All ages')]

        if not data_1970.empty and not data_2010.empty:
            rate_1970 = data_1970['Death Rate Per 100,000'].values[0]
            rate_2010 = data_2010['Death Rate Per 100,000'].values[0]
            percent_decrease = ((rate_1970 - rate_2010) / rate_1970) * 100

            improvements.append({
                'Country': country,
                'Rate 1970': rate_1970,
                'Rate 2010': rate_2010,
                'Percent Decrease': percent_decrease
            })

    return pd.DataFrame(improvements)

improvement_df = calc_improvement()
top_improved = improvement_df.nlargest(10, 'Percent Decrease')

plt.figure(figsize=(14, 8))
sns.barplot(x='Percent Decrease', y='Country', data=top_improved, palette='viridis')
plt.title('Top 10 Countries with Greatest Improvement in Death Rates (1970-2010)')
plt.xlabel('Percent Decrease in Death Rate (%)')
plt.ylabel('Country')
plt.xlim(0, 100)  # Set x-axis from 0 to 100%

# Add value annotations
for i, v in enumerate(top_improved['Percent Decrease']):
    plt.text(v + 1, i, f'{v:.1f}%', va='center')

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

```
a. Countries with Biggest Improvement in Death Rates (1970-2010)
```




In [52]:
```
print("\nb. Age Group Analysis: Death Rate Trends Over Time")
# Analyze trends in specific age groups over time
infant_data = df[(df['Age Group'].isin(['0-6 days', '7-27 days', '28-364 days'])) & (df['Sex'] == 'Both')]
infant_trends = infant_data.groupby(['Year', 'Age Group'])['Death Rate Per 100,000'].mean().reset_index()
```

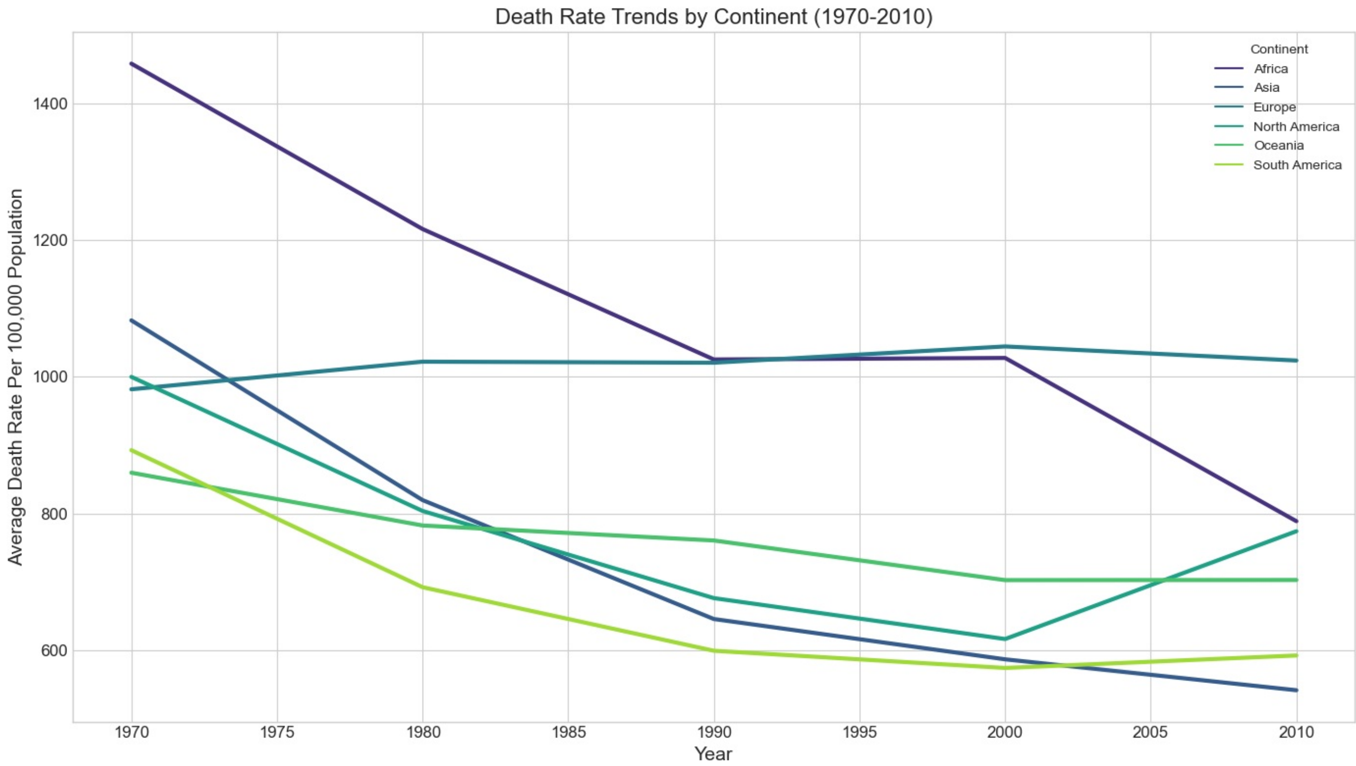
```
plt.figure(figsize=(14, 8))
sns.lineplot(x='Year', y='Death Rate Per 100,000', hue='Age Group',
             data=infant_trends, markers=True, dashes=False, linewidth=3)
plt.title('Infant Mortality Trends by Age Group (1970-2010)')
plt.xlabel('Year')
plt.ylabel('Death Rate Per 100,000 Population')
plt.legend(title='Age Group')
plt.tight_layout()
plt.show()
```

```
b. Age Group Analysis: Death Rate Trends Over Time
C:\Users\Axtom\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Axtom\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


In [67]:

```
print("\nc. Regional Analysis: Death Rates Across Continents")
# Create a more comprehensive mapping
# Extend the continent mapping to all countries in the dataset
all_countries = df[['Country Code', 'Country Name']].drop_duplicates()
# Fill in missing mappings with 'Other' for simplicity in this example
continent_series = pd.Series(continent_mapping)
df['Continent'] = df['Country Code'].map(continent_series)
# For countries without a mapping, assign based on typical geography (this would be more comprehensive in reali
df['Continent'] = df['Continent'].fillna('Other')

# Analyze death rates by continent over time
continent_data = df[(df['Sex'] == 'Both') & (df['Age Group'] == 'All ages') & (df['Continent'] != 'Other')]
continent_trends = continent_data.groupby(['Year', 'Continent'])['Death Rate Per 100,000'].mean().reset_index()

plt.figure(figsize=(14, 8))
sns.lineplot(x='Year', y='Death Rate Per 100,000', hue='Continent',
             data=continent_trends, markers=True, dashes=False, linewidth=3)
plt.title('Death Rate Trends by Continent (1970-2010)')
plt.xlabel('Year')
plt.ylabel('Average Death Rate Per 100,000 Population')
plt.legend(title='Continent')
plt.tight_layout()
plt.show()
```

```
c. Regional Analysis: Death Rates Across Continents
C:\Users\Axtom\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\Axtom\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depr
ecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

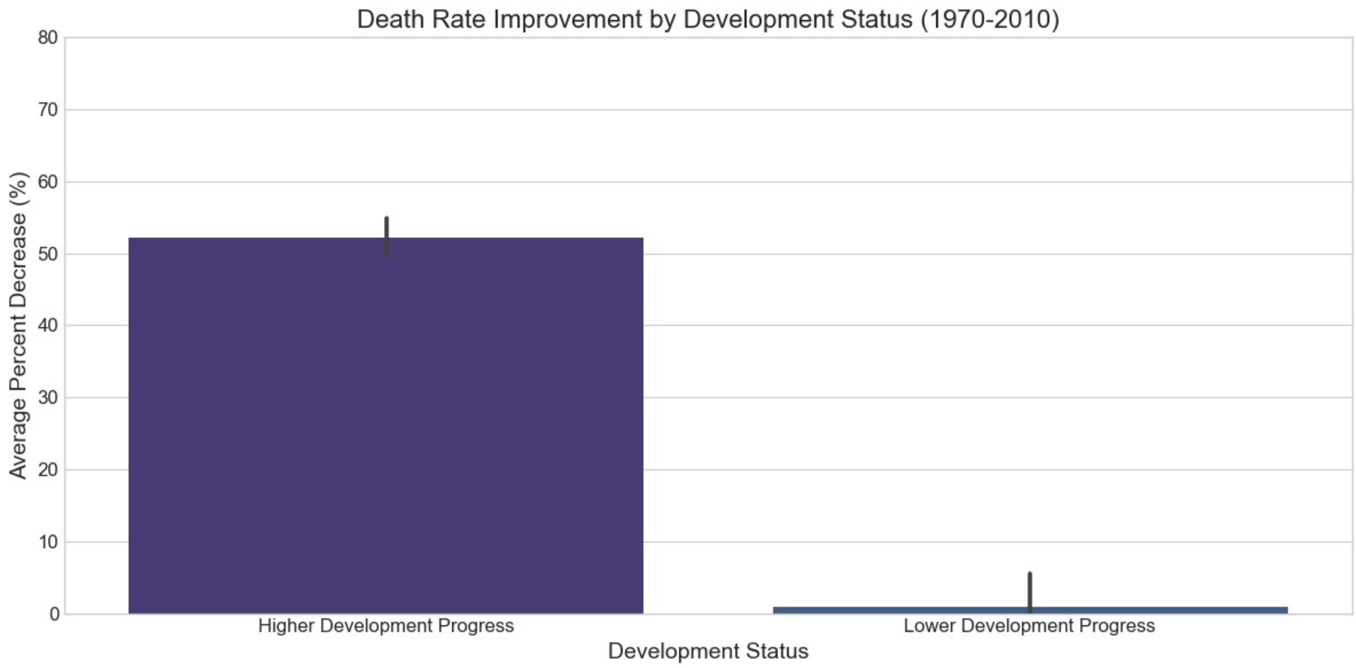



In [70]:

```
print("\nd. Final Analysis: Mortality Trends by Development Status")

# Define a simple development classification (this would be more comprehensive in reality)
# Using death rate improvement as a proxy for development
improvement_data = calc_improvement()  # Using the function defined earlier
median_improvement = improvement_data['Percent Decrease'].median()

# Add development status to the improvement dataframe
improvement_data['Development Status'] = np.where(
    improvement_data['Percent Decrease'] >= median_improvement,
    'Higher Development Progress',
    'Lower Development Progress'
)

# Compare death rates between countries with different development progress
dev_stats = improvement_data.groupby('Development Status').agg({
    'Rate 1970': 'mean',
    'Rate 2010': 'mean',
    'Percent Decrease': 'mean'
}).round(2)

print("\nComparison by Development Progress:")
display(dev_stats)

# Plot comparison
plt.figure(figsize=(12, 6))
sns.barplot(x='Development Status', y='Percent Decrease', data=improvement_data)
plt.title('Death Rate Improvement by Development Status (1970-2010)')
plt.ylabel('Average Percent Decrease (%)')
plt.ylim(0, 80)
plt.tight_layout()
plt.show()
```

```
d. Final Analysis: Mortality Trends by Development Status

Comparison by Development Progress:
```

| Development Status | Rate 1970 | Rate 2010 | Percent Decrease |
|---|---|---|---|
| **Higher Development Progress** | 1439.45 | 677.38 | 52.25 |
| **Lower Development Progress** | 937.70 | 922.69 | 0.88 |


Death Rate Improvement by Development Status (1970-2010)
Average Percent Decrease (%)
80
70
60
50
40
30
20
10
0
Higher Development Progress
Lower Development Progress
Development Status

In [ ]: