```
In [2]: import pandas as pd
df = pd.read_csv(r'C:\Users\divaa\Downloads\IHME_GBD_2010_MORTALITY_AGE_SPECIFIC_BY_COUNTRY_1970_2010.csv')
print(df.head())
print(df.info())
print(df.describe())
print(df.isnull().sum())
```

```
  Country Code Country Name  Year  Age Group     Sex Number of Deaths  \
0          AFG  Afghanistan  1970   0-6 days    Male           19,241
1          AFG  Afghanistan  1970   0-6 days  Female           12,600
2          AFG  Afghanistan  1970   0-6 days    Both           31,840
3          AFG  Afghanistan  1970  7-27 days    Male           15,939
4          AFG  Afghanistan  1970  7-27 days  Female           11,287

  Death Rate Per 100,000 Unnamed: 7  Unnamed: 8  Unnamed: 9  Unnamed: 10
0            3,18,292.90        NaN         NaN         NaN          NaN
1            2,19,544.20        NaN         NaN         NaN          NaN
2            2,70,200.70                    NaN         NaN          NaN
3              92,701.00        NaN         NaN         NaN          NaN
4              68,594.50        NaN         NaN         NaN          NaN
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 58905 entries, 0 to 58904
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Country Code            58905 non-null  object
 1   Country Name            58905 non-null  object
 2   Year                    58905 non-null  int64
 3   Age Group               58905 non-null  object
 4   Sex                     58905 non-null  object
 5   Number of Deaths        58905 non-null  object
 6   Death Rate Per 100,000  58905 non-null  object
 7   Unnamed: 7              1 non-null      object
 8   Unnamed: 8              0 non-null      float64
 9   Unnamed: 9              0 non-null      float64
 10  Unnamed: 10             0 non-null      float64
dtypes: float64(3), int64(1), object(7)
memory usage: 4.9+ MB
None
               Year  Unnamed: 8  Unnamed: 9  Unnamed: 10
count  58905.000000         0.0         0.0          0.0
mean    1990.000000         NaN         NaN          NaN
std       14.142256         NaN         NaN          NaN
min     1970.000000         NaN         NaN          NaN
25%     1980.000000         NaN         NaN          NaN
50%     1990.000000         NaN         NaN          NaN
75%     2000.000000         NaN         NaN          NaN
max     2010.000000         NaN         NaN          NaN
Country Code                  0
Country Name                  0
Year                          0
Age Group                     0
Sex                           0
Number of Deaths              0
Death Rate Per 100,000        0
Unnamed: 7                58904
Unnamed: 8                58905
Unnamed: 9                58905
Unnamed: 10               58905
dtype: int64
```

```
In [3]: df['Year'] = df['Year'].astype(int)
df
```

Out[3]:

	Country Code	Country Name	Year	Age Group	Sex	Number of Deaths	Death Rate Per 100,000	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnamed: 10
0	AFG	Afghanistan	1970	0-6 days	Male	19,241	3,18,292.90	NaN	NaN	NaN	NaN
1	AFG	Afghanistan	1970	0-6 days	Female	12,600	2,19,544.20	NaN	NaN	NaN	NaN
2	AFG	Afghanistan	1970	0-6 days	Both	31,840	2,70,200.70		NaN	NaN	NaN
3	AFG	Afghanistan	1970	7-27 days	Male	15,939	92,701.00	NaN	NaN	NaN	NaN
4	AFG	Afghanistan	1970	7-27 days	Female	11,287	68,594.50	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
58900	ZWE	Zimbabwe	2010	80+ years	Female	7,606	16,179.40	NaN	NaN	NaN	NaN
58901	ZWE	Zimbabwe	2010	80+ years	Both	13,305	16,633.80	NaN	NaN	NaN	NaN
58902	ZWE	Zimbabwe	2010	All ages	Male	82,010	1,320.10	NaN	NaN	NaN	NaN
58903	ZWE	Zimbabwe	2010	All ages	Female	77,420	1,211.20	NaN	NaN	NaN	NaN
58904	ZWE	Zimbabwe	2010	All ages	Both	1,59,430	1,264.90	NaN	NaN	NaN	NaN

58905 rows × 11 columns

```
In [4]: unique_age_groups = df['Age Group'].unique()

# distinct age groups
print("Distinct Age Groups in the dataset:")
for x in unique_age_groups:
    print(x)
```

```
Distinct Age Groups in the dataset:
0-6 days
7-27 days
28-364 days
1-4 years
5-9 years
10-14 years
15-19 years
20-24 years
25-29 years
30-34 years
35-39 years
40-44 years
45-49 years
50-54 years
55-59 years
60-64 years
65-69 years
70-74 years
75-79 years
80+ years
All ages
```

verifying does all year have same category of age in df

```
In [5]: age_groups_by_year = df.groupby('Year')['Age Group'].unique().reset_index()
age_groups_by_year_df = pd.DataFrame({
    year: age_groups_by_year[age_groups_by_year['Year'] == year]['Age Group'].values[0]
    for year in age_groups_by_year['Year']
})

age_groups_by_year_df = age_groups_by_year_df.T

age_groups_by_year_df.index.name = 'Year'

age_groups_by_year_df
```

Out[5]:

Year	0	1	2	3	4	5	6	7	8	9	...	11	12	13	14	15	16	17	18	19	20
1970	0-6 days	7-27 days	28-364 days	1-4 years	5-9 years	10-14 years	15-19 years	20-24 years	25-29 years	30-34 years	...	40-44 years	45-49 years	50-54 years	55-59 years	60-64 years	65-69 years	70-74 years	75-79 years	80+ years	All ages
1980	0-6 days	7-27 days	28-364 days	1-4 years	5-9 years	10-14 years	15-19 years	20-24 years	25-29 years	30-34 years	...	40-44 years	45-49 years	50-54 years	55-59 years	60-64 years	65-69 years	70-74 years	75-79 years	80+ years	All ages
1990	0-6 days	7-27 days	28-364 days	1-4 years	5-9 years	10-14 years	15-19 years	20-24 years	25-29 years	30-34 years	...	40-44 years	45-49 years	50-54 years	55-59 years	60-64 years	65-69 years	70-74 years	75-79 years	80+ years	All ages
2000	0-6 days	7-27 days	28-364 days	1-4 years	5-9 years	10-14 years	15-19 years	20-24 years	25-29 years	30-34 years	...	40-44 years	45-49 years	50-54 years	55-59 years	60-64 years	65-69 years	70-74 years	75-79 years	80+ years	All ages
2010	0-6 days	7-27 days	28-364 days	1-4 years	5-9 years	10-14 years	15-19 years	20-24 years	25-29 years	30-34 years	...	40-44 years	45-49 years	50-54 years	55-59 years	60-64 years	65-69 years	70-74 years	75-79 years	80+ years	All ages

5 rows × 21 columns

```
In [6]: pd.set_option('display.max_rows', None)    # Show all rows
pd.set_option('display.max_columns', None) # Show all columns
pd.set_option('display.width', None)       # Disable line wrapping for wide DataFrames
pd.set_option('display.max_colwidth', None) # Show full column width
```

```
In [31]: df['Death Rate Per 100,000'] = df['Death Rate Per 100,000'].replace({',': ''}, regex=True)
df['Death Rate Per 100,000'] = pd.to_numeric(df['Death Rate Per 100,000'], errors='coerce')

avg_death_rate = df.groupby(['Country Name', 'Year'])['Death Rate Per 100,000'].mean().reset_index()

avg_death_rate.head(10)
```

Out[31]:

	Country Name	Year	Death Rate Per 100,000
0	Afghanistan	1970	20698.087302
1	Afghanistan	1980	16202.161905
2	Afghanistan	1990	12241.011111
3	Afghanistan	2000	12085.296825
4	Afghanistan	2010	9090.528571
5	Albania	1970	3703.938095
6	Albania	1980	3419.473016
7	Albania	1990	3067.988889
8	Albania	2000	2826.514286
9	Albania	2010	2257.311111

```
In [8]: print(df['Number of Deaths'].dtype)
```

```
object
```

```
In [9]: df['Number of Deaths'] = df['Number of Deaths'].replace({',': ''}, regex=True)
df['Number of Deaths'] = pd.to_numeric(df['Number of Deaths'], errors='coerce')


total_deaths = df.groupby(['Age Group', 'Sex'])['Number of Deaths'].sum().reset_index()

total_deaths['Number of Deaths'] = total_deaths['Number of Deaths'].astype(int)

print("Total Deaths by Age Group and Sex:")
print(total_deaths)
```

```
Total Deaths by Age Group and Sex:
      Age Group     Sex  Number of Deaths
0      0-6 days    Both          14939574
1      0-6 days  Female           6170816
2      0-6 days    Male           8768740
3     1-4 years    Both          18341384
4     1-4 years  Female           8541885
5     1-4 years    Male           9799497
6   10-14 years    Both           2257405
7   10-14 years  Female           1034706
8   10-14 years    Male           1222721
9   15-19 years    Both           3584053
10  15-19 years  Female           1574548
11  15-19 years    Male           2009490
12  20-24 years    Both           4686591
13  20-24 years  Female           1986733
14  20-24 years    Male           2699881
15  25-29 years    Both           4939511
16  25-29 years  Female           2086776
17  25-29 years    Male           2852741
18  28-364 days    Both          17858504
19  28-364 days  Female           8143367
20  28-364 days    Male           9715165
21  30-34 years    Both           5150454
22  30-34 years  Female           2115593
23  30-34 years    Male           3034879
24  35-39 years    Both           5859297
25  35-39 years  Female           2391097
26  35-39 years    Male           3468215
27  40-44 years    Both           6660430
28  40-44 years  Female           2569468
29  40-44 years    Male           4090982
30  45-49 years    Both           8032689
31  45-49 years  Female           3046823
32  45-49 years    Male           4985852
33     5-9 years    Both           3429188
34     5-9 years  Female           1560274
35     5-9 years    Male           1868907
36  50-54 years    Both          10081483
37  50-54 years  Female           3815622
38  50-54 years    Male           6265872
39  55-59 years    Both          12401146
40  55-59 years  Female           4794538
41  55-59 years    Male           7606616
42  60-64 years    Both          15607135
43  60-64 years  Female           6310579
44  60-64 years    Male           9296550
45  65-69 years    Both          18101171
46  65-69 years  Female           7774154
47  65-69 years    Male          10327020
48     7-27 days    Both           6395830
49     7-27 days  Female           2773008
50     7-27 days    Male           3622818
51  70-74 years    Both          20544791
52  70-74 years  Female           9502466
53  70-74 years    Male          11042341
54  75-79 years    Both          20623983
55  75-79 years  Female          10384302
56  75-79 years    Male          10239679
57    80+ years    Both          37744291
58    80+ years  Female          22692004
59    80+ years    Male          15052289
60     All ages    Both         237238928
61     All ages  Female         109268706
62     All ages    Male         127970207
```

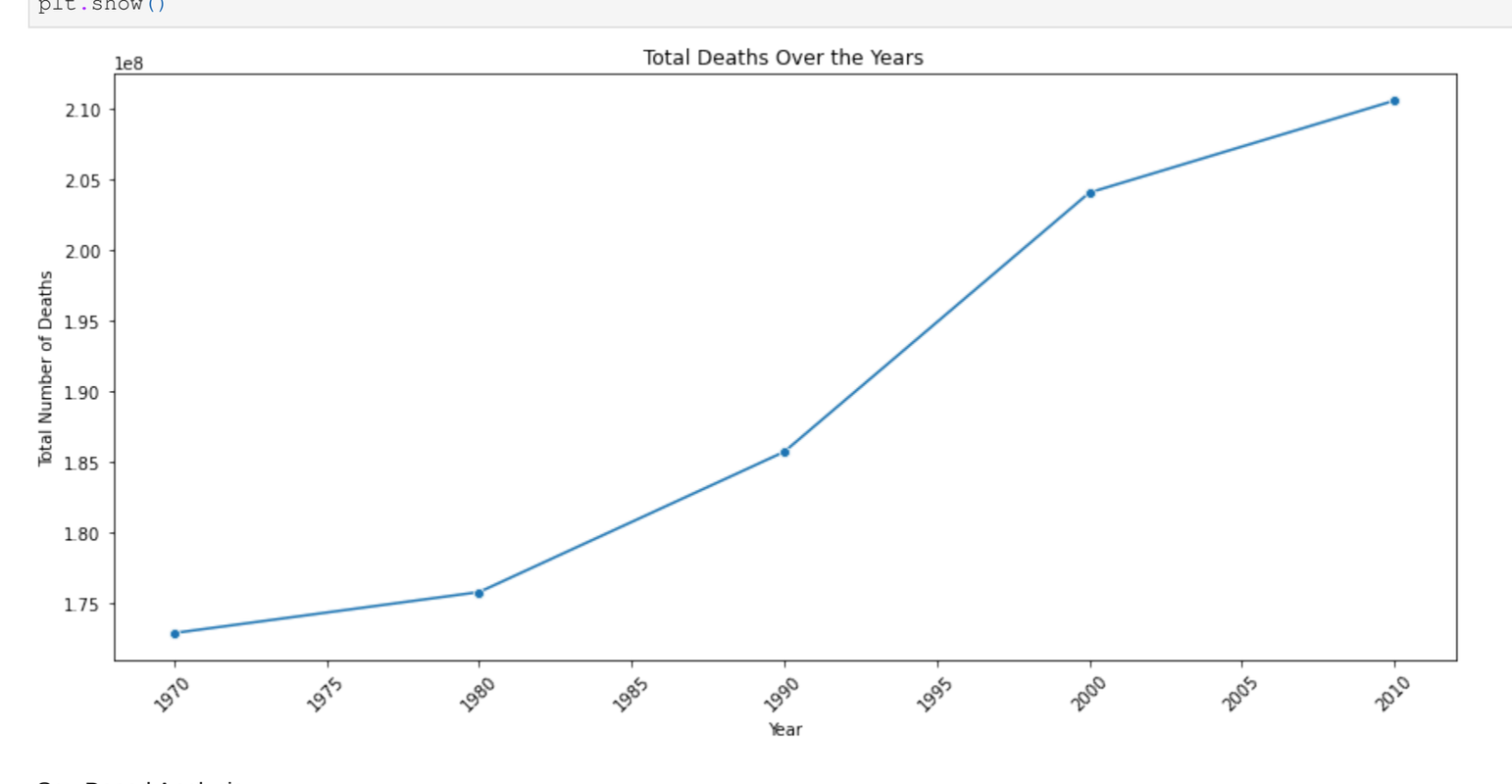
```
In [10]: import matplotlib.pyplot as plt
import seaborn as sns

yearly_deaths = df.groupby('Year')['Number of Deaths'].sum().reset_index()
yearly_deaths
```

Out[10]:

	Year	Number of Deaths
0	1970	172866152
1	1980	175767051
2	1990	185702142
3	2000	204052611
4	2010	210567809

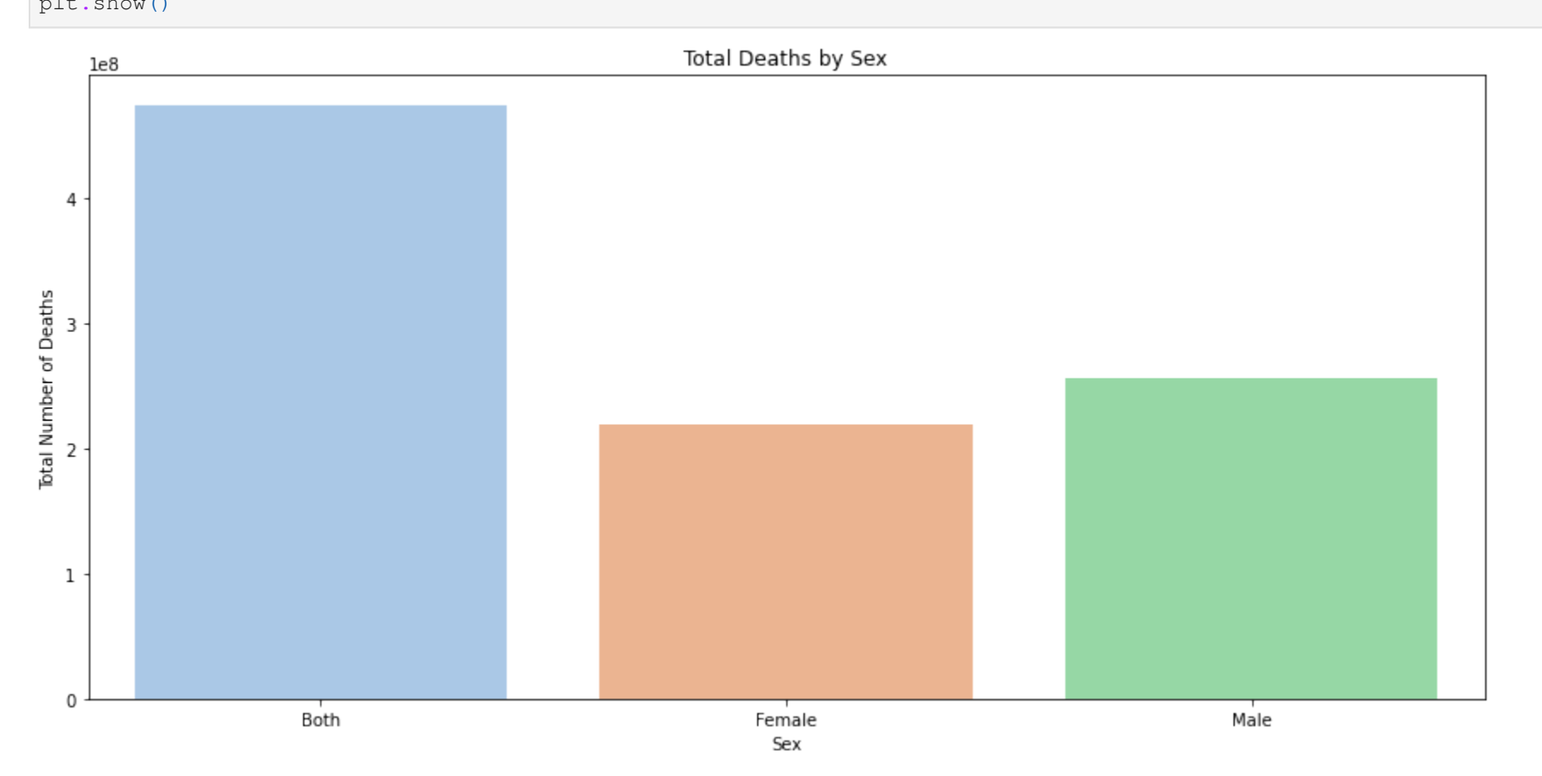
Trend Analysis Over Years

```
In [11]: # Time Series Analysis
plt.figure(figsize=(12, 6))
sns.lineplot(data=yearly_deaths, x='Year', y='Number of Deaths', marker='o')
plt.title('Total Deaths Over the Years')
plt.xlabel('Year')
plt.ylabel('Total Number of Deaths')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



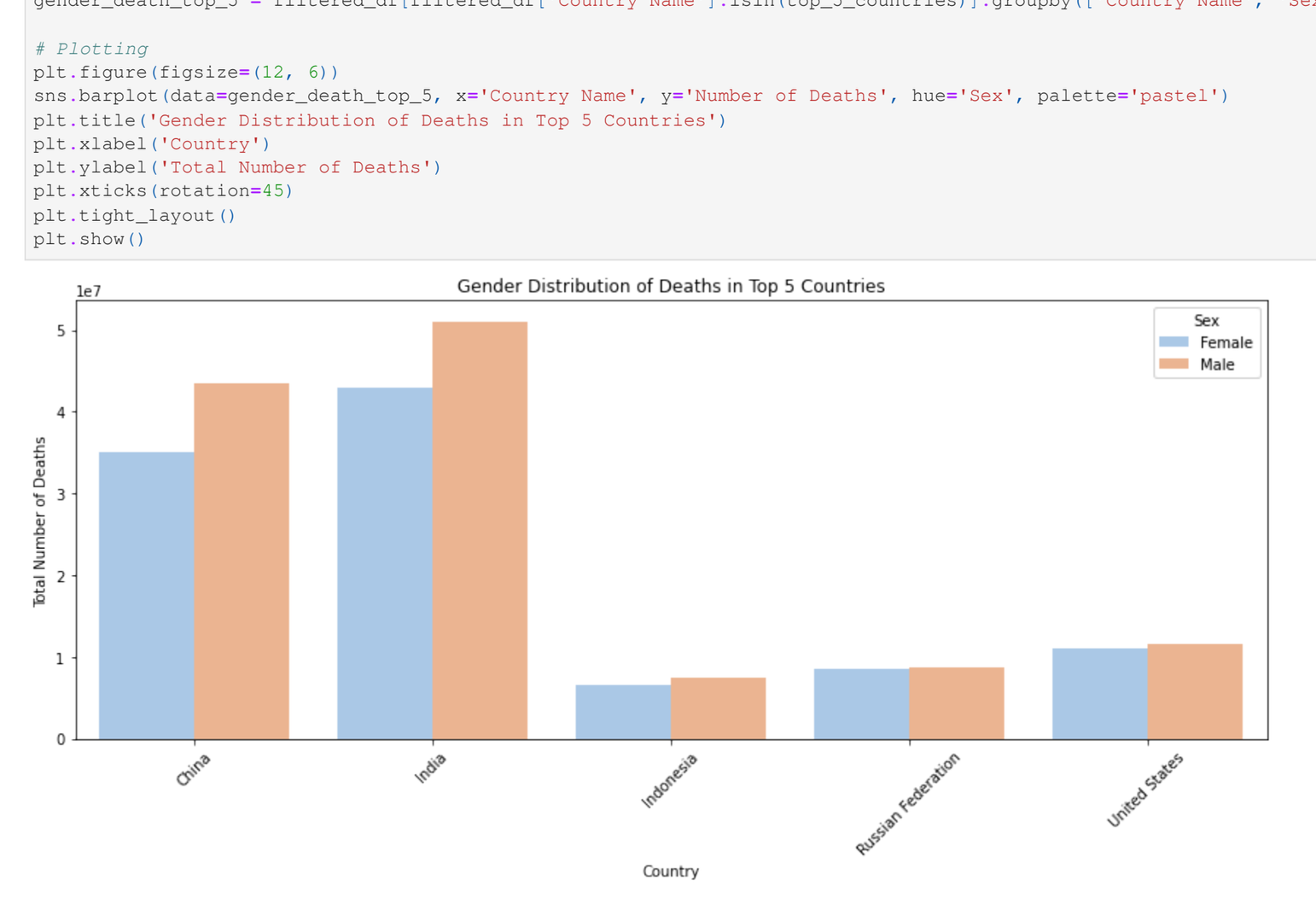
Sex-Based Analysis

```
In [12]: sex_deaths = df.groupby('Sex')['Number of Deaths'].sum().reset_index()

plt.figure(figsize=(12, 6))
sns.barplot(data=sex_deaths, x='Sex', y='Number of Deaths', palette='pastel')
plt.title('Total Deaths by Sex')
plt.xlabel('Sex')
plt.ylabel('Total Number of Deaths')
plt.tight_layout()
plt.show()
```



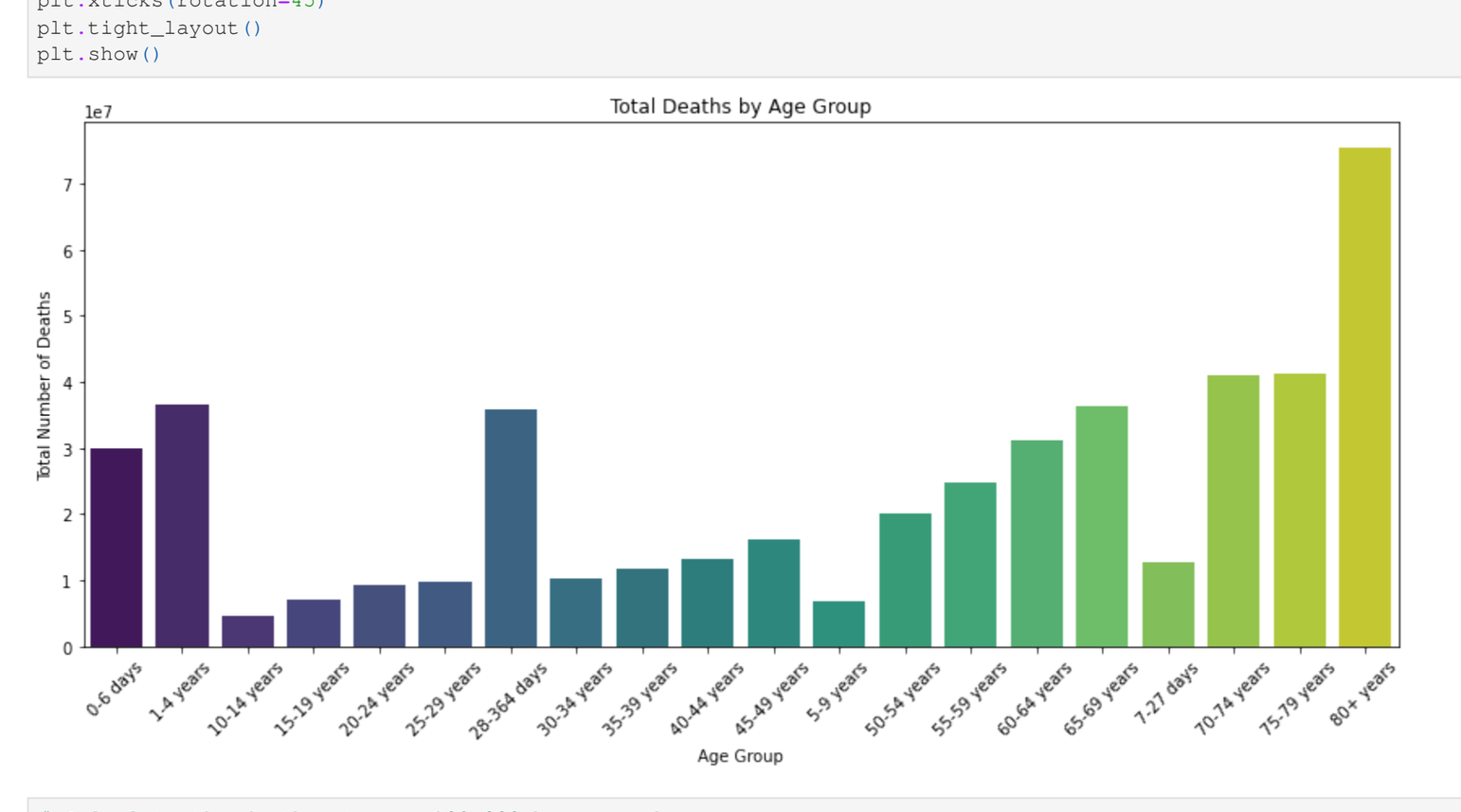
Gender Distribution of Deaths in Top 5 Countries

```
In [13]: #top 5 countries by total deaths
top_5_countries = df.groupby('Country Name')['Number of Deaths'].sum().nlargest(5).index

# Filter out 'Both' from the 'Sex' column
filtered_df = df[df['Sex'] != 'Both']

# total deaths by sex in the top 5 countries
gender_death_top_5 = filtered_df[filtered_df['Country Name'].isin(top_5_countries)].groupby(['Country Name', 'Sex'])['Number of Deaths'].sum().reset_index()

# Plotting
plt.figure(figsize=(12, 6))
sns.barplot(data=gender_death_top_5, x='Country Name', y='Number of Deaths', hue='Sex', palette='pastel')
plt.title('Gender Distribution of Deaths in Top 5 Countries')
plt.xlabel('Country')
plt.ylabel('Total Number of Deaths')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

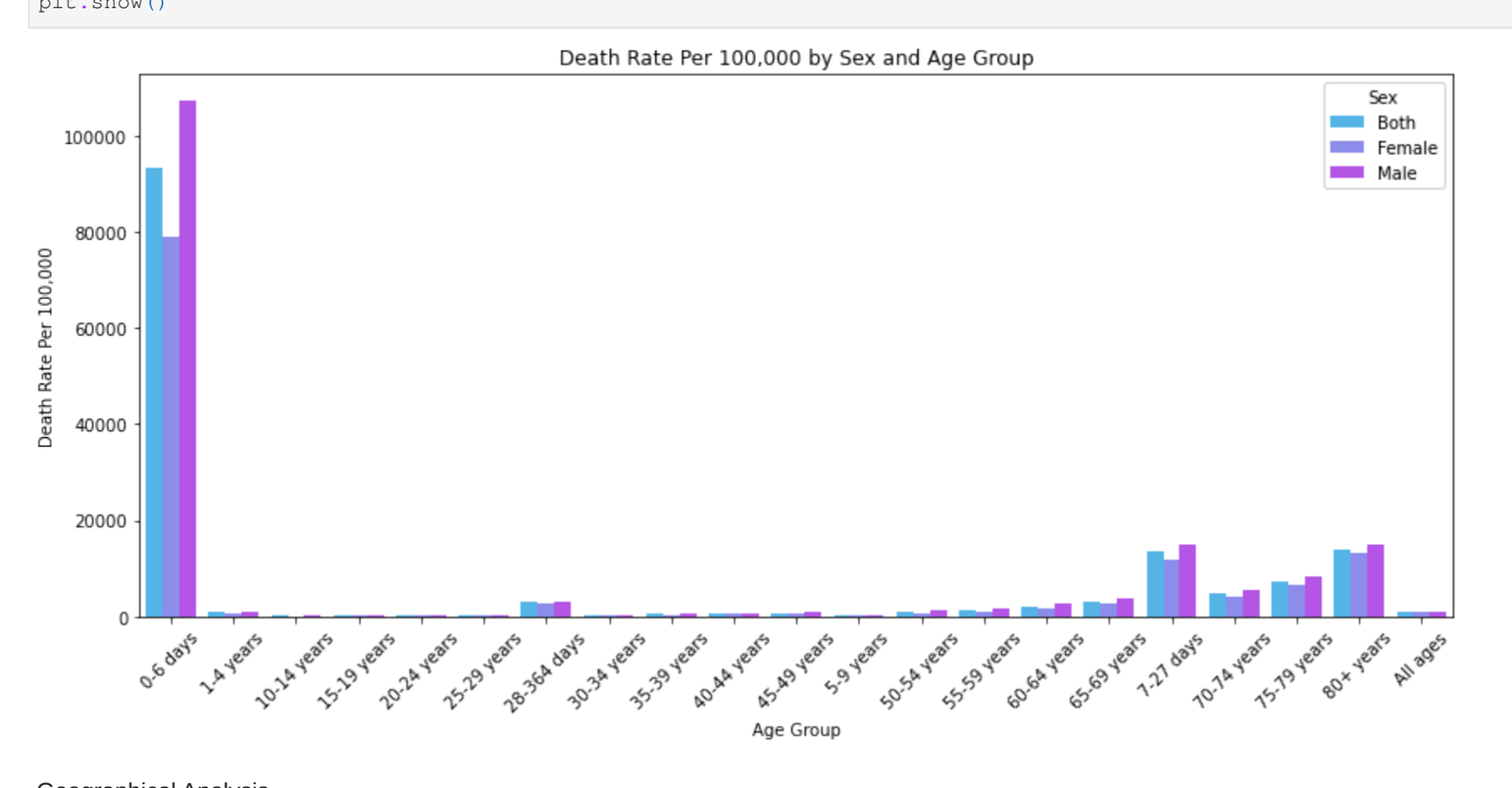


Age Group Comparison

```
In [14]: # data by Age Group and summing Number of Deaths
age_group_deaths = df.groupby('Age Group')['Number of Deaths'].sum().reset_index()
age_group_deaths = age_group_deaths[age_group_deaths['Age Group'] != 'All ages']


plt.figure(figsize=(12, 6))
sns.barplot(data=age_group_deaths, x='Age Group', y='Number of Deaths', palette='viridis')
plt.title('Total Deaths by Age Group')
plt.xlabel('Age Group')
plt.ylabel('Total Number of Deaths')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



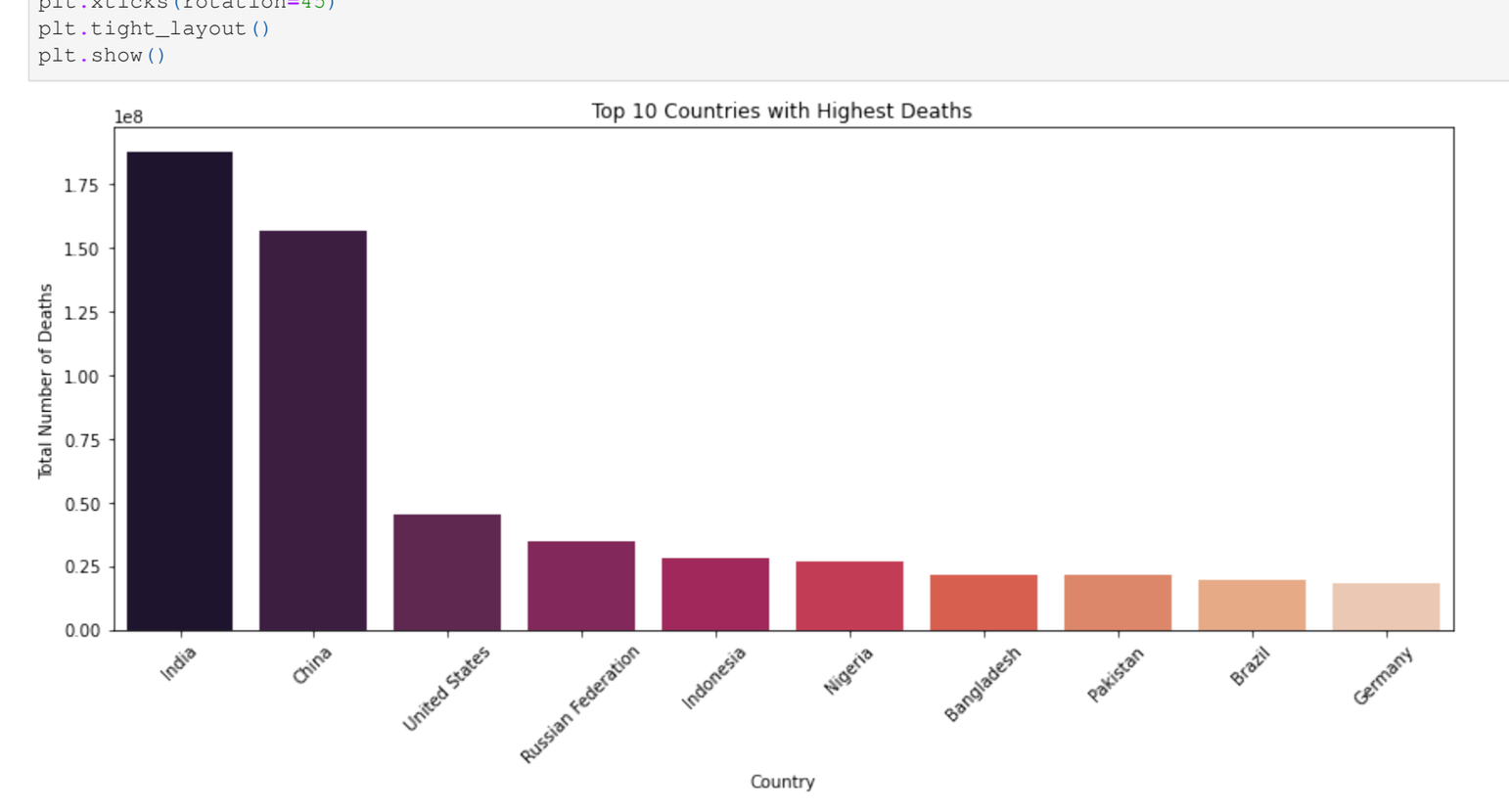

```
In [15]: # Calculate the death rate per 100,000 by sex and age group
sex_age_death_rate = df.groupby(['Sex', 'Age Group'])['Death Rate Per 100,000'].mean().reset_index()

# Plotting
plt.figure(figsize=(12, 6))
sns.barplot(data=sex_age_death_rate, x='Age Group', y='Death Rate Per 100,000', hue='Sex', palette='cool')
plt.title('Death Rate Per 100,000 by Sex and Age Group')
plt.xlabel('Age Group')
plt.ylabel('Death Rate Per 100,000')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



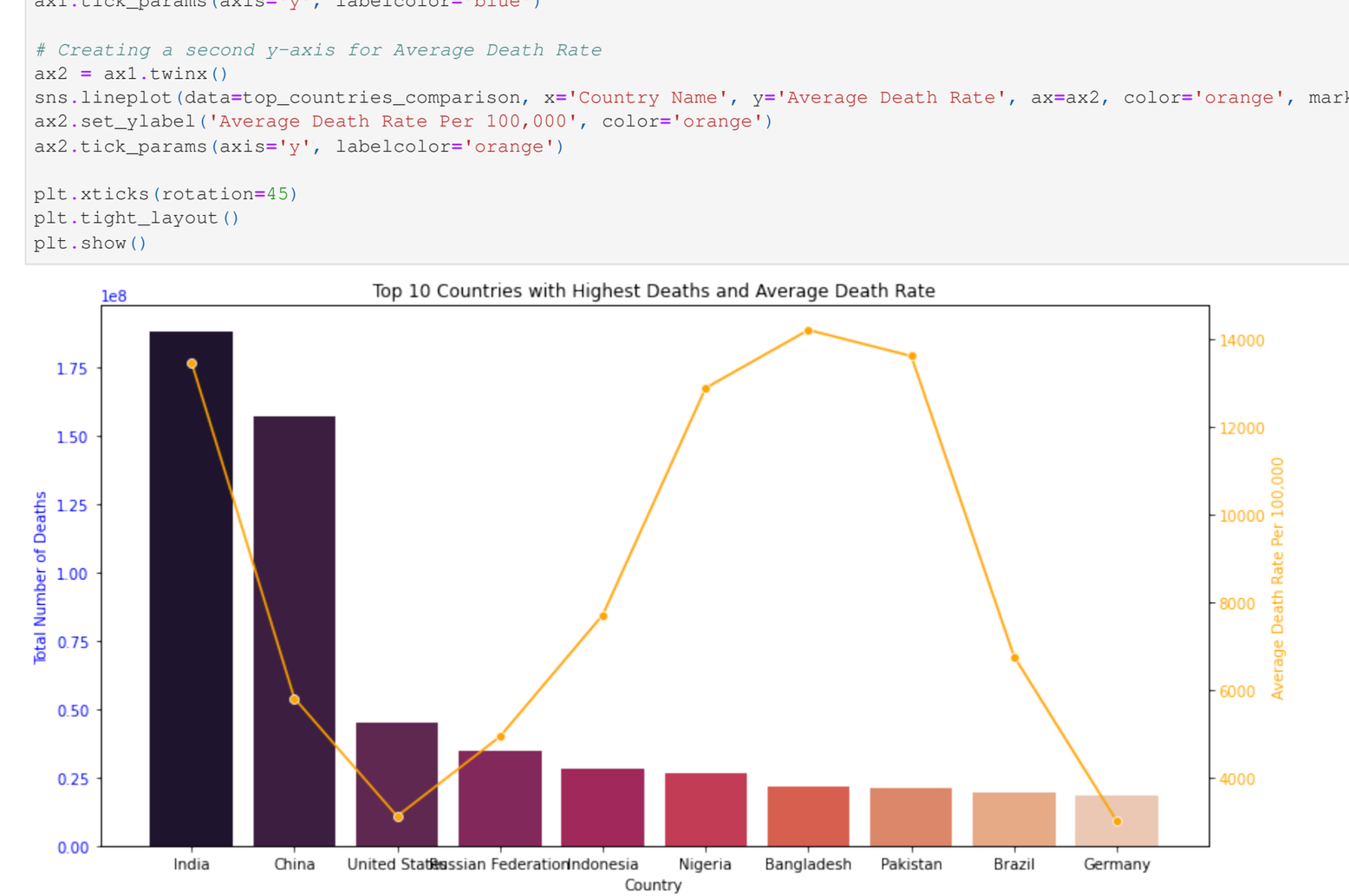
Geographical Analysis

Country-Wise Comparison

```
In [16]: country_deaths = df.groupby('Country Name')['Number of Deaths'].sum().reset_index()

# Top 10 countries with the highest deaths
top_countries = country_deaths.nlargest(10, 'Number of Deaths')

# Plotting
plt.figure(figsize=(12, 6))
sns.barplot(data=top_countries, x='Country Name', y='Number of Deaths', palette='rocket')
plt.title('Top 10 Countries with Highest Deaths')
plt.xlabel('Country')
plt.ylabel('Total Number of Deaths')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

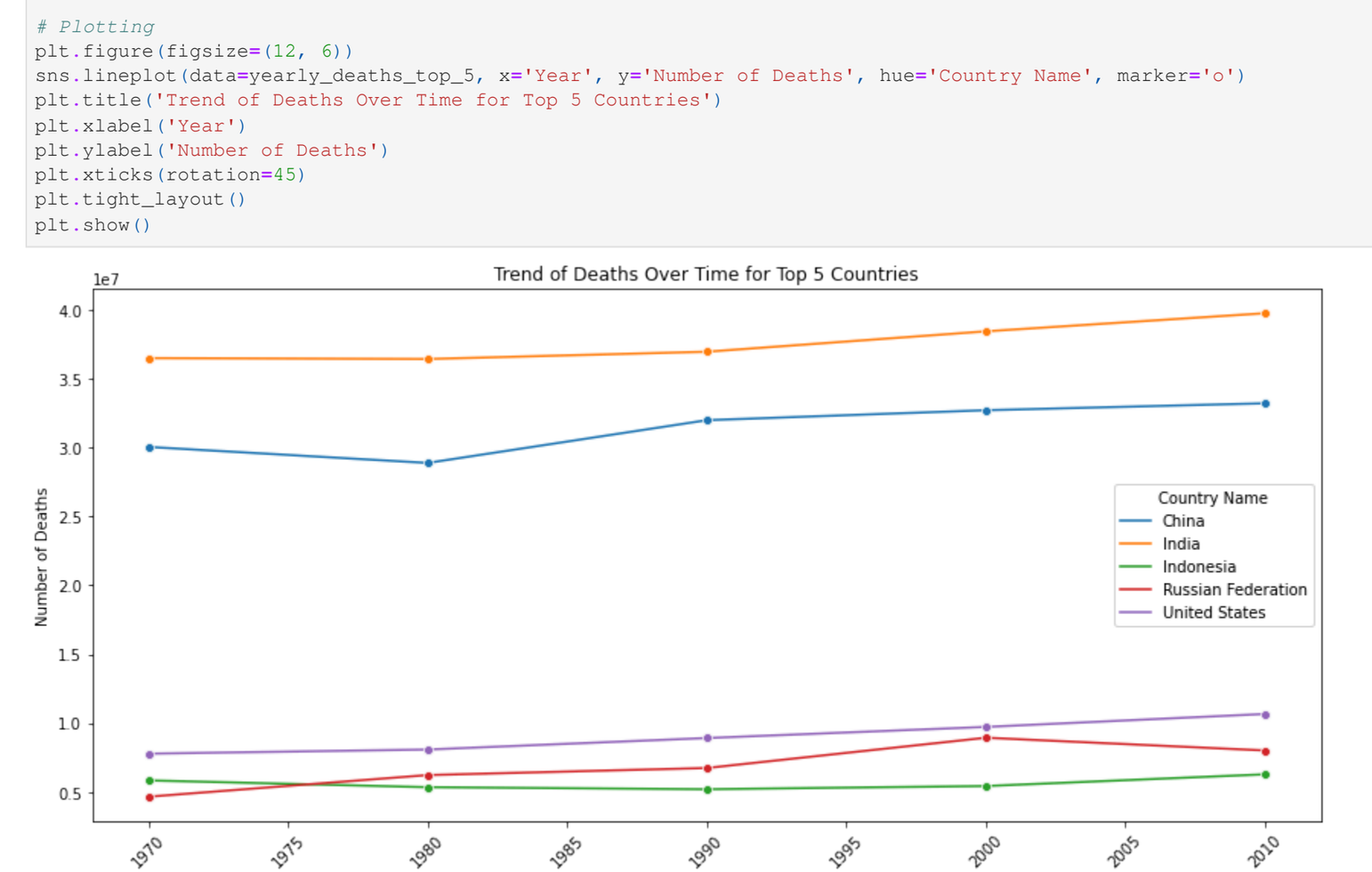



```
In [17]: # Country to get total deaths and average death rate
country_deaths = df.groupby('Country Name')['Number of Deaths'].sum().reset_index()
country_death_rate = df.groupby('Country Name')['Death Rate Per 100,000'].mean().reset_index()

country_comparison = pd.merge(country_deaths, country_death_rate, on='Country Name')
country_comparison.columns = ['Country Name', 'Total Deaths', 'Average Death Rate']

top_countries_comparison = country_comparison.nlargest(10, 'Total Deaths')

fig, ax1 = plt.subplots(figsize=(12, 6))

# Bar plot for Total Deaths
sns.barplot(data=top_countries_comparison, x='Country Name', y='Total Deaths', ax=ax1, palette='rocket')
ax1.set_title('Top 10 Countries with Highest Deaths and Average Death Rate')
ax1.set_xlabel('Country')
ax1.set_ylabel('Total Number of Deaths', color='blue')
ax1.tick_params(axis='y', labelcolor='blue')

# Creating a second y-axis for Average Death Rate
ax2 = ax1.twinx()
sns.lineplot(data=top_countries_comparison, x='Country Name', y='Average Death Rate', ax=ax2, color='orange', marker='o')
ax2.set_ylabel('Average Death Rate Per 100,000', color='orange')
ax2.tick_params(axis='y', labelcolor='orange')

plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Trend of Deaths for Top 5 Countries Over Time

```
In [18]: # Calculate total deaths by country
top_5_countries = df.groupby('Country Name')['Number of Deaths'].sum().nlargest(5).index

top_5_countries_data = df[df['Country Name'].isin(top_5_countries)]


yearly_deaths_top_5 = top_5_countries_data.groupby(['Year', 'Country Name'])['Number of Deaths'].sum().reset_index()

# Plotting
plt.figure(figsize=(12, 6))
sns.lineplot(data=yearly_deaths_top_5, x='Year', y='Number of Deaths', hue='Country Name', marker='o')
plt.title('Trend of Deaths Over Time for Top 5 Countries')
plt.xlabel('Year')
plt.ylabel('Number of Deaths')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
In [19]: from scipy import stats

male_deaths = df[df['Sex'] == 'Male']['Number of Deaths']
female_deaths = df[df['Sex'] == 'Female']['Number of Deaths']

t_stat, p_value = stats.ttest_ind(male_deaths.dropna(), female_deaths.dropna())
print(f'T-statistic: {t_stat}, P-value: {p_value}')
```

```
T-statistic: 1.7236274282573487, P-value: 0.08478299512285009
```

T-statistic shows there is a difference in the number of deaths between males and females.

P-value tells us that the difference is likely just random and not meaningful.

```
In [20]: if 'Year' in df.columns and 'Number of Deaths' in df.columns:
    # Group by 'Year' and sum 'Number of Deaths'
    yearly_deaths = df.groupby('Year')['Number of Deaths'].sum().reset_index()
else:
    print("Columns 'Year' and 'Number of Deaths' not found in DataFrame")
```

```
In [21]: print(df[['Year', 'Number of Deaths']].head())  # Inspect the original data
```

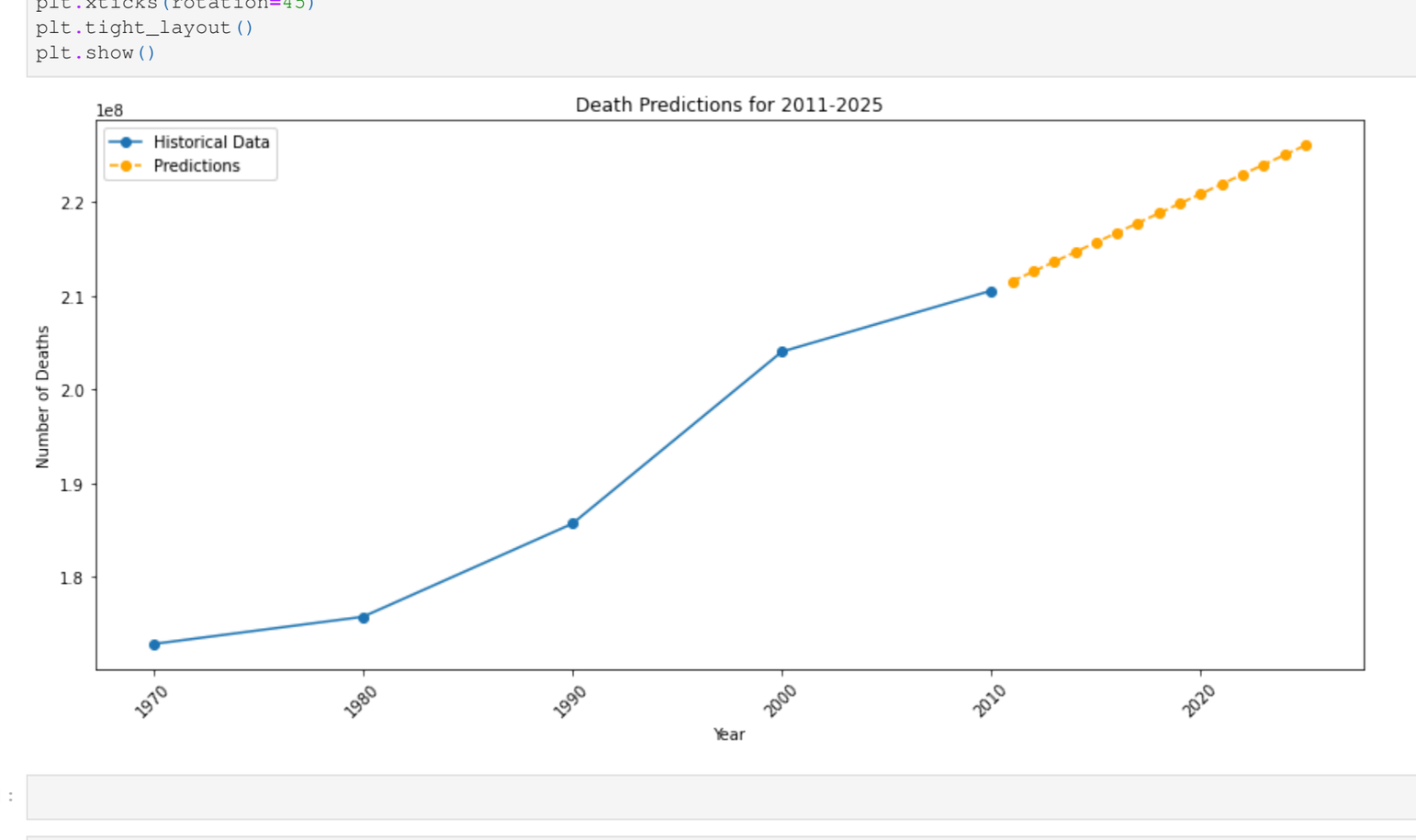
```
   Year  Number of Deaths
0  1970             19241
1  1970             12600
2  1970             31840
3  1970             15939
4  1970             11287
```

```
In [22]: df = df.dropna(subset=['Year', 'Number of Deaths'])

df['Year'] = pd.to_numeric(df['Year'], errors='coerce')

yearly_deaths = df.groupby('Year')['Number of Deaths'].sum().reset_index()
```

```
In [23]: print(yearly_deaths.head())
print(yearly_deaths.shape)
```

```
   Year  Number of Deaths
0  1970         172866152
1  1980         175767051
2  1990         185702142
3  2000         204052611
4  2010         210567809
(5, 2)
```

```
In [24]: from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt

# Prepare data
X = yearly_deaths['Year'].values.reshape(-1, 1)
y = yearly_deaths['Number of Deaths'].values

# Ensure that X and y contain data
if X.size == 0 or y.size == 0:
    raise ValueError("X or y contains no data. Please check the dataset.")

# Fit
model = LinearRegression()
model.fit(X, y)

# Predict
future_years = np.array(range(2011, 2026)).reshape(-1, 1)
predictions = model.predict(future_years)

# Plotting
plt.figure(figsize=(12, 6))
plt.plot(yearly_deaths['Year'], yearly_deaths['Number of Deaths'], marker='o', label='Historical Data')
plt.plot(future_years, predictions, marker='o', linestyle='--', color='orange', label='Predictions')
plt.title('Death Predictions for 2011-2025')
plt.xlabel('Year')
plt.ylabel('Number of Deaths')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```




```
In [ ]:
```

```
In [25]: import pandas as pd

deaths_per_year = df.groupby('Year')['Number of Deaths'].sum().reset_index()

# Convert 'Total Deaths' to millions
deaths_per_year['Total Deaths (in millions)'] = deaths_per_year['Number of Deaths'] / 1_000_000

# Drop the original 'Number of Deaths' column if you don't need it
deaths_per_year.drop(columns=['Number of Deaths'], inplace=True)

# Display the result
print(deaths_per_year)
```

```
   Year  Total Deaths (in millions)
0  1970                  172.866152
1  1980                  175.767051
2  1990                  185.702142
3  2000                  204.052611
4  2010                  210.567809
```

```
In [26]: import pandas as pd

# Assuming your data is in a DataFrame called df
# Group by 'Year' and calculate the average of 'Death Rate Per 100,000'
death_rate_per_year = df.groupby('Year')['Death Rate Per 100,000'].mean().reset_index()

# Convert 'Death Rate Per 100,000' to 'Death Rate Per Million' by dividing by 10
death_rate_per_year['Death Rate Per Million'] = death_rate_per_year['Death Rate Per 100,000'] / 10

# Drop the original 'Death Rate Per 100,000' column if you don't need it
death_rate_per_year.drop(columns=['Death Rate Per 100,000'], inplace=True)

# Display the result
print(death_rate_per_year)
```

```
   Year  Death Rate Per Million
0  1970              984.206221
1  1980              809.655411
2  1990              676.776020
3  2000              585.482619
4  2010              475.312958
```

```
In [27]: # Group by Country Name and sum the Number of Deaths
country_deaths = df.groupby('Country Name')['Number of Deaths'].sum().reset_index()

# Top 10 countries with the highest deaths
top_countries = country_deaths.nlargest(10, 'Number of Deaths')

# Display the data in table format
print("Top 10 Countries with Highest Deaths:")
print(top_countries)
```

```
Top 10 Countries with Highest Deaths:
           Country Name  Number of Deaths
75                India         188079240
35                China         156849037
178       United States          45128930
138  Russian Federation          34541811
76            Indonesia          28061221
124             Nigeria          26624172
13           Bangladesh          21570855
128            Pakistan          21344898
23               Brazil          19489395
63              Germany          18111967
```

```
In [28]: # Clean and convert 'Number of Deaths' to numeric (remove commas, if any)
df['Number of Deaths'] = df['Number of Deaths'].replace({',': ''}, regex=True)
df['Number of Deaths'] = pd.to_numeric(df['Number of Deaths'], errors='coerce')

# Group by Country Name and sum the Number of Deaths
country_deaths = df.groupby('Country Name')['Number of Deaths'].sum().reset_index()

# Top 10 countries with the highest deaths
top_countries = country_deaths.nlargest(10, 'Number of Deaths')

# Display the data in table format
print("Top 10 Countries with Highest Deaths:")
print(top_countries)
```

```
Top 10 Countries with Highest Deaths:
           Country Name  Number of Deaths
75                India         188079240
35                China         156849037
178       United States          45128930
138  Russian Federation          34541811
76            Indonesia          28061221
124             Nigeria          26624172
13           Bangladesh          21570855
128            Pakistan          21344898
23               Brazil          19489395
63              Germany          18111967
```

```
In [29]: # Clean and convert 'Number of Deaths' to numeric (remove commas, if any)
df['Number of Deaths'] = df['Number of Deaths'].replace({',': ''}, regex=True)
df['Number of Deaths'] = pd.to_numeric(df['Number of Deaths'], errors='coerce')

# Group by Country Name and sum the Number of Deaths
country_deaths = df.groupby('Country Name')['Number of Deaths'].sum().reset_index()

# Convert the Number of Deaths to millions
country_deaths['Number of Deaths'] = country_deaths['Number of Deaths'] / 1_000_000

# Top 10 countries with the highest deaths
top_countries = country_deaths.nlargest(10, 'Number of Deaths')

# Display the data in table format
print("Top 10 Countries with Highest Deaths (in millions):")
print(top_countries)
```

```
Top 10 Countries with Highest Deaths (in millions):
           Country Name  Number of Deaths
75                India        188.079240
35                China        156.849037
178       United States         45.128930
138  Russian Federation         34.541811
76            Indonesia         28.061221
124             Nigeria         26.624172
13           Bangladesh         21.570855
128            Pakistan         21.344898
23               Brazil         19.489395
63              Germany         18.111967
```

```
In [ ]:
```