In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

In [7]:
```
df = pd.read_csv(r"C:\Users\God\Downloads\covid_data.csv")
df
```

Out[7]:

	Country	Other names	ISO 3166-1 alpha-3 CODE	Population	Continent	Total Cases	Total Deaths	To
0	Afghanistan	Afghanistan	AFG	40462186	Asia	177827	7671	
1	Albania	Albania	ALB	2872296	Europe	273870	3492	
2	Algeria	Algeria	DZA	45236699	Africa	265691	6874	
3	Andorra	Andorra	AND	77481	Europe	40024	153	
4	Angola	Angola	AGO	34654212	Africa	99194	1900	
...	...	...	...	...	...	...	...	...
220	Wallis and Futuna	Wallis and Futuna Islands	WLF	10894	Oceania	454	7	
221	Western Sahara	Western Sahara	ESHÂ	623031	Africa	10	1	
222	Yemen	Yemen	YEM	30975258	Asia	11806	2143	
223	Zambia	Zambia	ZMB	19284482	Africa	317076	3967	
224	Zimbabwe	Zimbabwe	ZWE	15241601	Africa	246525	5446	

225 rows × 10 columns

In [5]:
```
df.head(10)
```

Out[5]:

	Country	Other names	ISO 3166-1 alpha-3 CODE	Population	Continent	Total Cases	Total Deaths	Tot
0	Afghanistan	Afghanistan	AFG	40462186	Asia	177827	7671	
1	Albania	Albania	ALB	2872296	Europe	273870	3492	
2	Algeria	Algeria	DZA	45236699	Africa	265691	6874	
3	Andorra	Andorra	AND	77481	Europe	40024	153	
4	Angola	Angola	AGO	34654212	Africa	99194	1900	
5	Anguilla	Anguilla	AIA	15237	Latin America and the Caribbean	2700	9	
6	Antigua and Barbuda	Antigua and Barbuda	ATG	99348	Latin America and the Caribbean	7493	135	
7	Argentina	Argentina	ARG	45921761	Latin America and the Caribbean	9041124	128065	
8	Armenia	Armenia	ARM	2972939	Asia	422574	8617	
9	Aruba	Aruba	ABW	107560	Latin America and the Caribbean	34051	212	

In [9]:
```
df.tail(5)
```

Out[9]:

	Country	Other names	ISO 3166-1 alpha-3 CODE	Population	Continent	Total Cases	Total Deaths	Tot C
220	Wallis and Futuna	Wallis and Futuna Islands	WLF	10894	Oceania	454	7	
221	Western Sahara	Western Sahara	ESHÂ	623031	Africa	10	1	
222	Yemen	Yemen	YEM	30975258	Asia	11806	2143	
223	Zambia	Zambia	ZMB	19284482	Africa	317076	3967	
224	Zimbabwe	Zimbabwe	ZWE	15241601	Africa	246525	5446	

In [10]:
```
df.shape
```

Out[10]: (225, 10)

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 225 entries, 0 to 224
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Country                 225 non-null    object 
 1   Other names             224 non-null    object 
 2   ISO 3166-1 alpha-3 CODE  225 non-null    object 
 3   Population              225 non-null    int64  
 4   Continent               225 non-null    object 
 5   Total Cases             225 non-null    int64  
 6   Total Deaths            225 non-null    int64  
 7   Tot Cases//1M pop       225 non-null    int64  
 8   Tot Deaths/1M pop       225 non-null    int64  
 9   Death percentage        225 non-null    float64
dtypes: float64(1), int64(5), object(4)
memory usage: 17.7+ KB
```

In [12]:
```
df.describe()
```

Out[12]:

	Population	Total Cases	Total Deaths	Tot Cases//1M pop	Tot Deaths/ 1M pop	p
count	2.250000e+02	2.250000e+02	2.250000e+02	225.000000	225.000000	
mean	3.507321e+07	2.184781e+06	2.744813e+04	136900.373333	1096.715556	
std	1.392418e+08	7.275938e+06	9.689177e+04	145060.340289	1195.715543	
min	8.050000e+02	1.000000e+00	0.000000e+00	9.000000	0.000000	
25%	5.665570e+05	2.407100e+04	1.890000e+02	11384.000000	123.000000	
50%	5.827911e+06	1.639360e+05	1.965000e+03	88987.000000	708.000000	
75%	2.190585e+07	1.092547e+06	1.366000e+04	223335.000000	1795.000000	
max	1.439324e+09	8.183905e+07	1.008222e+06	696044.000000	6286.000000	

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
Country                    0
Other names                1
ISO 3166-1 alpha-3 CODE    0
Population                 0
Continent                  0
Total Cases                0
Total Deaths               0
Tot Cases//1M pop          0
Tot Deaths/1M pop          0
Death percentage           0
dtype: int64
```

In [16]:
```
df[df.isnull().any(axis=1)]
```

Out[16]:

	Country	Other names	ISO 3166-1 alpha-3 CODE	Population	Continent	Total Cases	Total Deaths	Tot Ca
135	Montenegro	NaN	MNE	628205	Europe	233326	2705	

In [17]:
```
df['Other names'].fillna(df['Country'], inplace=True)

df['ISO 3166-1 alpha-3 CODE'].fillna('MISSING', inplace=True)

print("Null values have been filled.")
```

```
Null values have been filled.
```

```
C:\Users\God\AppData\Local\Temp\ipykernel_9044\1110956703.py:1: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df['Other names'].fillna(df['Country'], inplace=True)
C:\Users\God\AppData\Local\Temp\ipykernel_9044\1110956703.py:3: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df['ISO 3166-1 alpha-3 CODE'].fillna('MISSING', inplace=True)
```

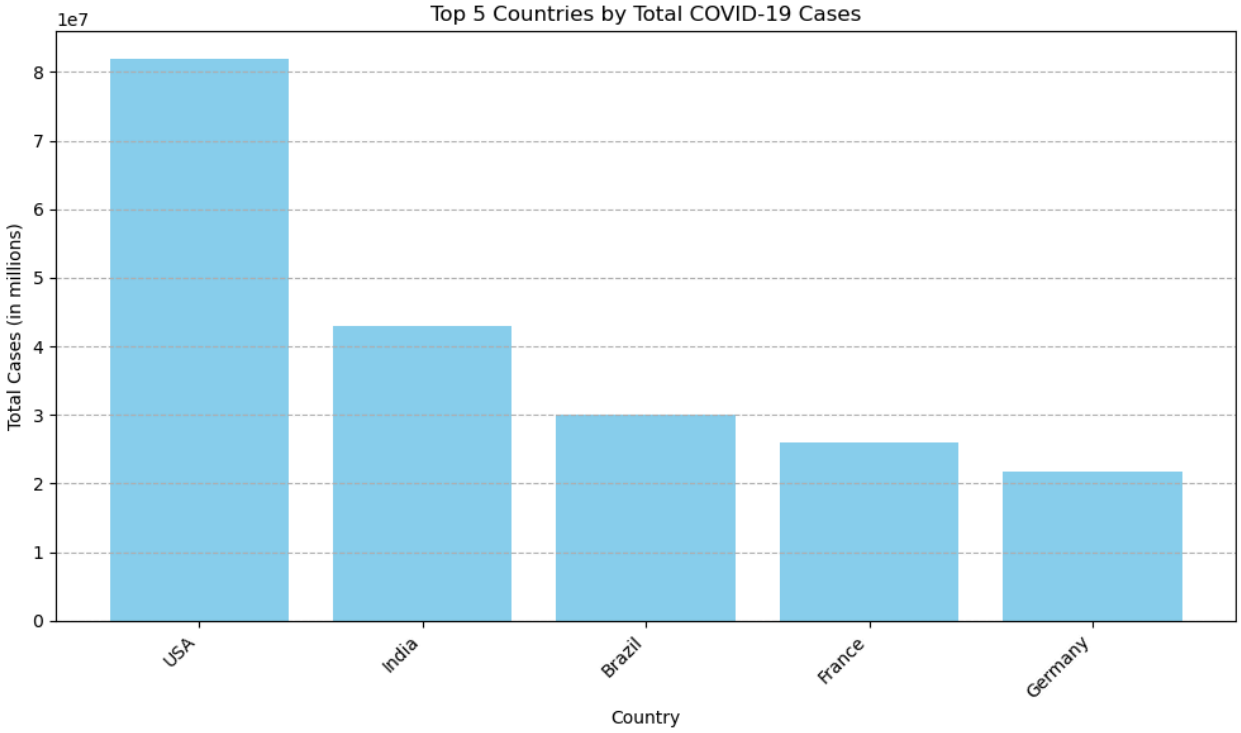
In [18]:
```
df.isnull().sum()
```

Out[18]:
```
Country                   0
Other names               0
ISO 3166-1 alpha-3 CODE   0
Population                0
Continent                 0
Total Cases               0
Total Deaths              0
Tot Cases//1M pop         0
Tot Deaths/1M pop         0
Death percentage          0
dtype: int64
```

In [19]:
```
top_5_cases = df.nlargest(5, 'Total Cases')

print("Top 5 Countries by Total Cases:")
print(top_5_cases[['Country', 'Total Cases']])
```

```
Top 5 Countries by Total Cases:
     Country  Total Cases
214      USA     81839052
92     India     43029044
26    Brazil     29999816
70    France     25997852
76   Germany     21646375
```

In [20]:
```
plt.figure(figsize=(10, 6))
plt.bar(top_5_cases['Country'], top_5_cases['Total Cases'], color='skyblue')
plt.title('Top 5 Countries by Total COVID-19 Cases')
plt.xlabel('Country')
plt.ylabel('Total Cases (in millions)')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```

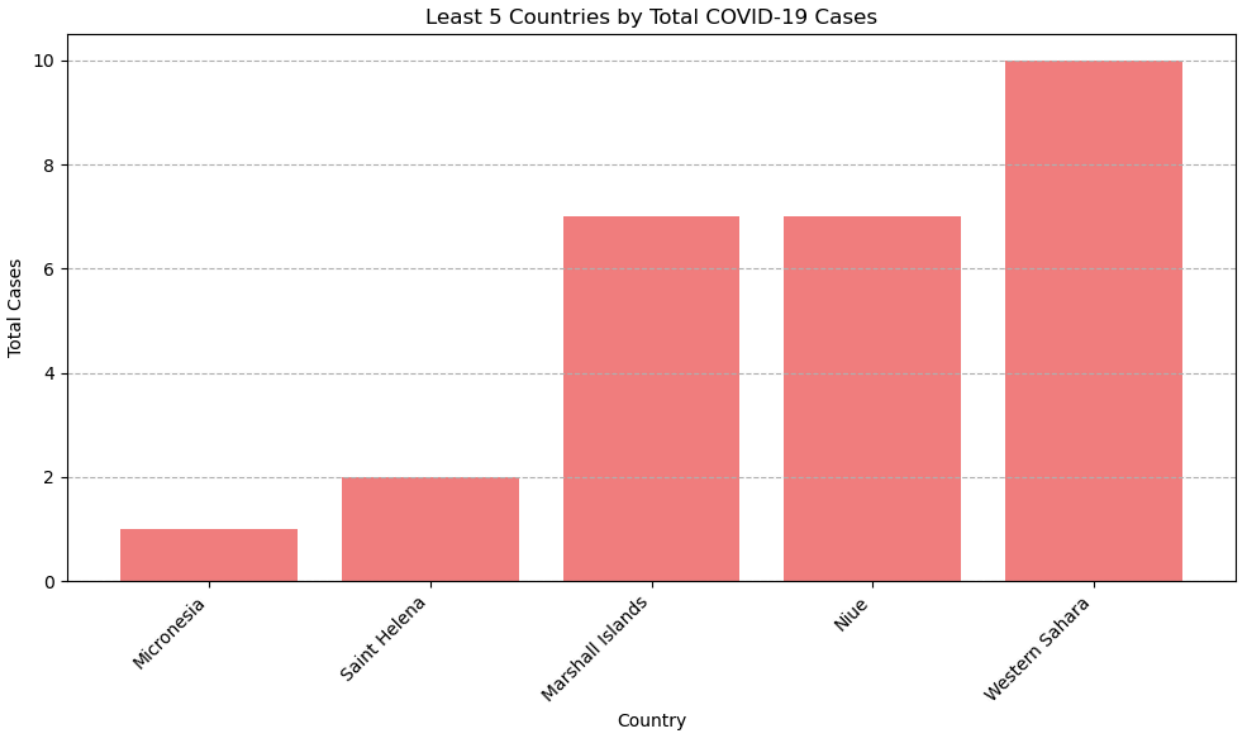


In [21]:
```
least_5_cases = df.nsmallest(5, 'Total Cases')

print("\nLeast 5 Countries by Total Cases:")
print(least_5_cases[['Country', 'Total Cases']])
```

```
Least 5 Countries by Total Cases:
             Country  Total Cases
131        Micronesia            1
168      Saint Helena            2
125  Marshall Islands            7
148              Niue            7
221    Western Sahara           10
```

In [22]:
```
plt.figure(figsize=(10, 6))
plt.bar(least_5_cases['Country'], least_5_cases['Total Cases'], color='lightcc
plt.title('Least 5 Countries by Total COVID-19 Cases')
plt.xlabel('Country')
plt.ylabel('Total Cases')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
```

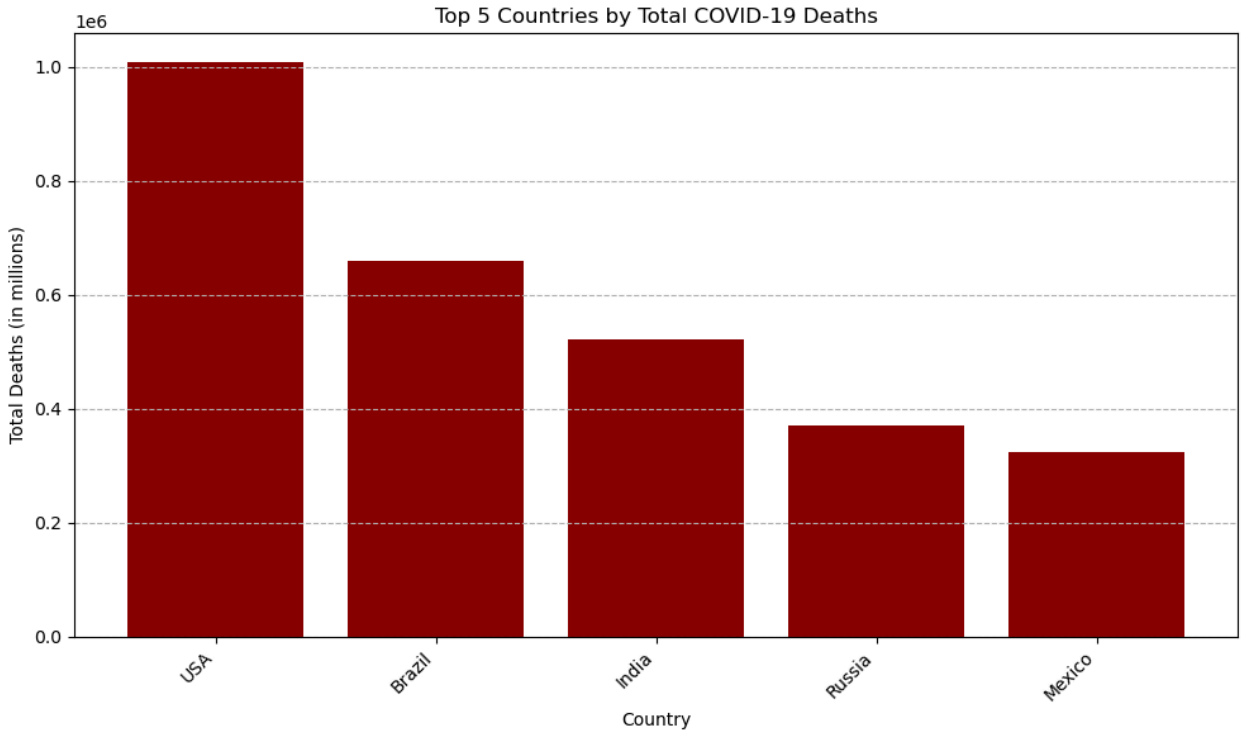
```
plt.show()
```



In [25]:
```
top_5_deaths = df.nlargest(5, 'Total Deaths')

print("\nTop 5 Countries by Total Deaths:")
print(top_5_deaths[['Country', 'Total Deaths']])
```

```
Top 5 Countries by Total Deaths:
    Country  Total Deaths
214     USA       1008222
26   Brazil        660269
92    India        521388
165  Russia        369708
130  Mexico        323212
```

In [26]:
```
plt.figure(figsize=(10, 6))
plt.bar(top_5_deaths['Country'], top_5_deaths['Total Deaths'], color='darkred'
plt.title('Top 5 Countries by Total COVID-19 Deaths')
plt.xlabel('Country')
plt.ylabel('Total Deaths (in millions)')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```



In [ ]:
```
plt.figure(figsize=(10, 6))
plt.bar(least_5_deaths['Country'], least_5_deaths['Total Deaths'], color='ligh
plt.title('Least 5 Countries by Total COVID-19 Deaths')
plt.xlabel('Country')
plt.ylabel('Total Deaths')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```

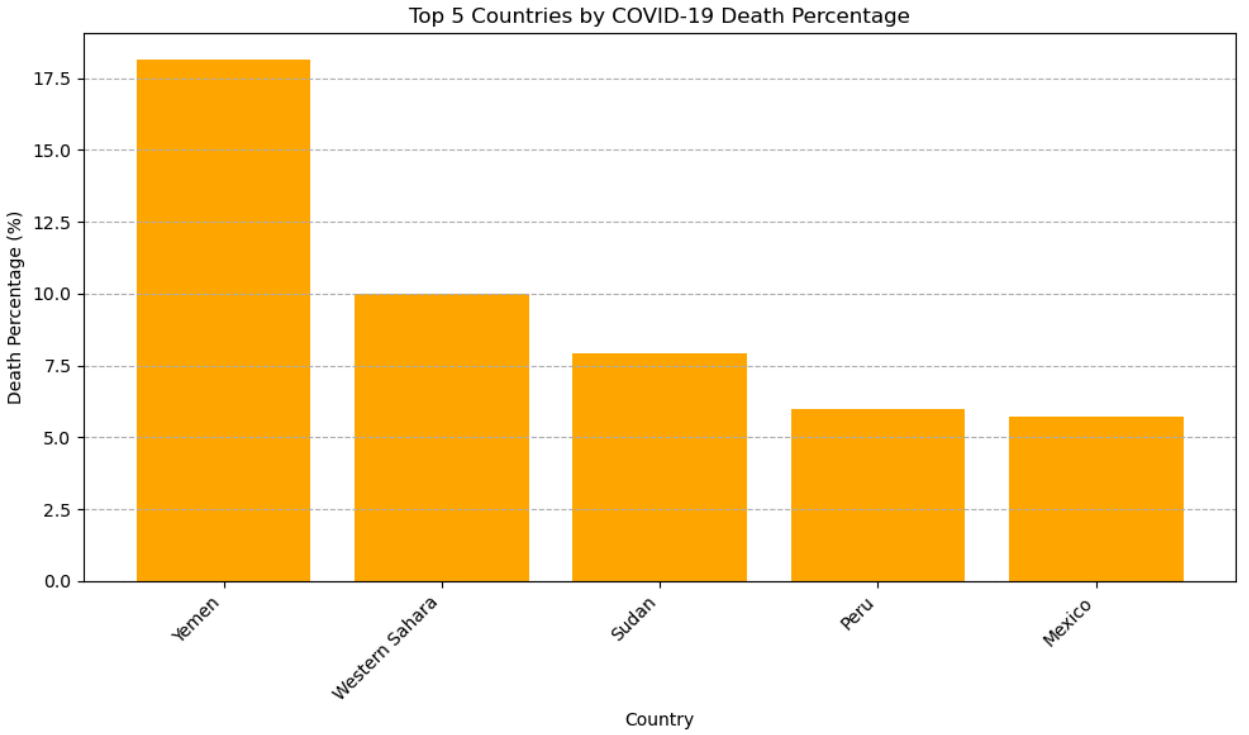
In [27]:
```
least_5_deaths = df.nsmallest(5, 'Total Deaths')

print("\nLeast 5 Countries by Total Deaths:")
print(least_5_deaths[['Country', 'Total Deaths']])
```

```
Least 5 Countries by Total Deaths:
              Country  Total Deaths
46       Cook Islands             0
67   Falkland Islands             0
118             Macao             0
125  Marshall Islands             0
131        Micronesia             0
```

In [23]:
```
top_5_death_percentage = df.nlargest(5, 'Death percentage')

print("\nTop 5 Countries by COVID-19 Death Percentage:")
print(top_5_death_percentage[['Country', 'Death percentage']])
```

```
Top 5 Countries by COVID-19 Death Percentage:
            Country  Death percentage
222           Yemen         18.151787
221  Western Sahara         10.000000
193           Sudan          7.920265
158            Peru          5.983499
130          Mexico          5.705041
```

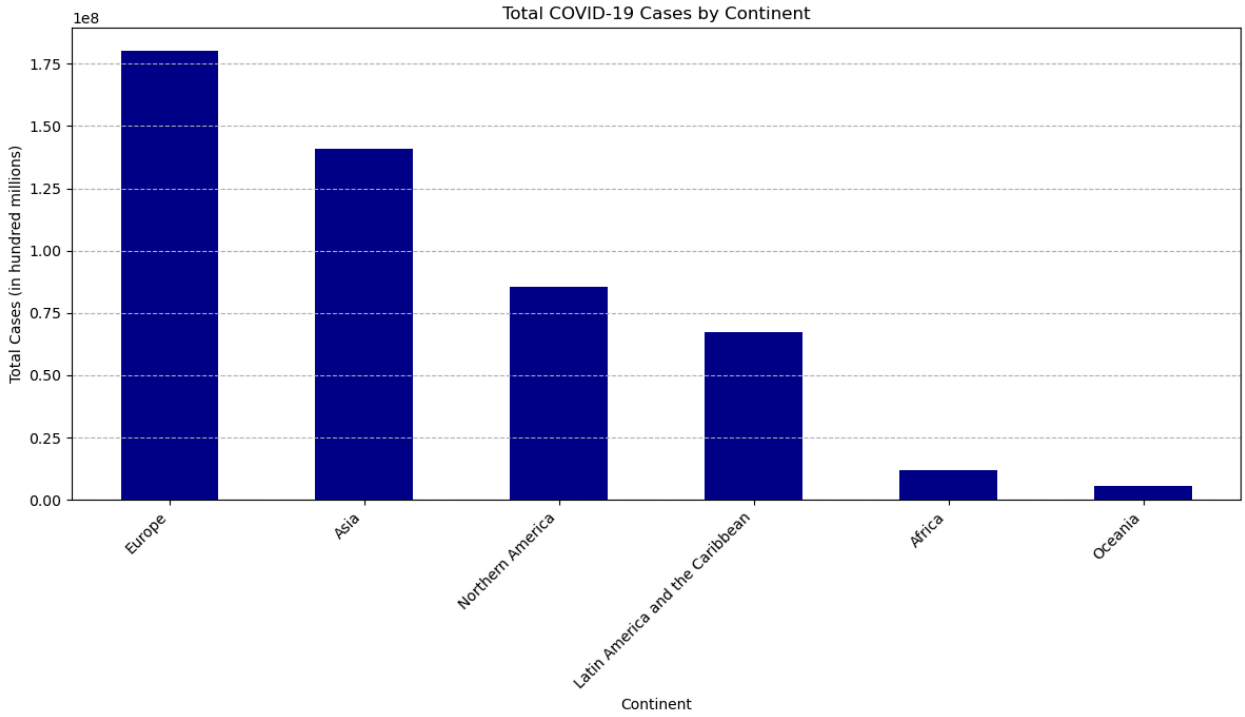
In [24]:
```
plt.figure(figsize=(10, 6))
plt.bar(top_5_death_percentage['Country'], top_5_death_percentage['Death perce
plt.title('Top 5 Countries by COVID-19 Death Percentage')
plt.xlabel('Country')
plt.ylabel('Death Percentage (%)')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```



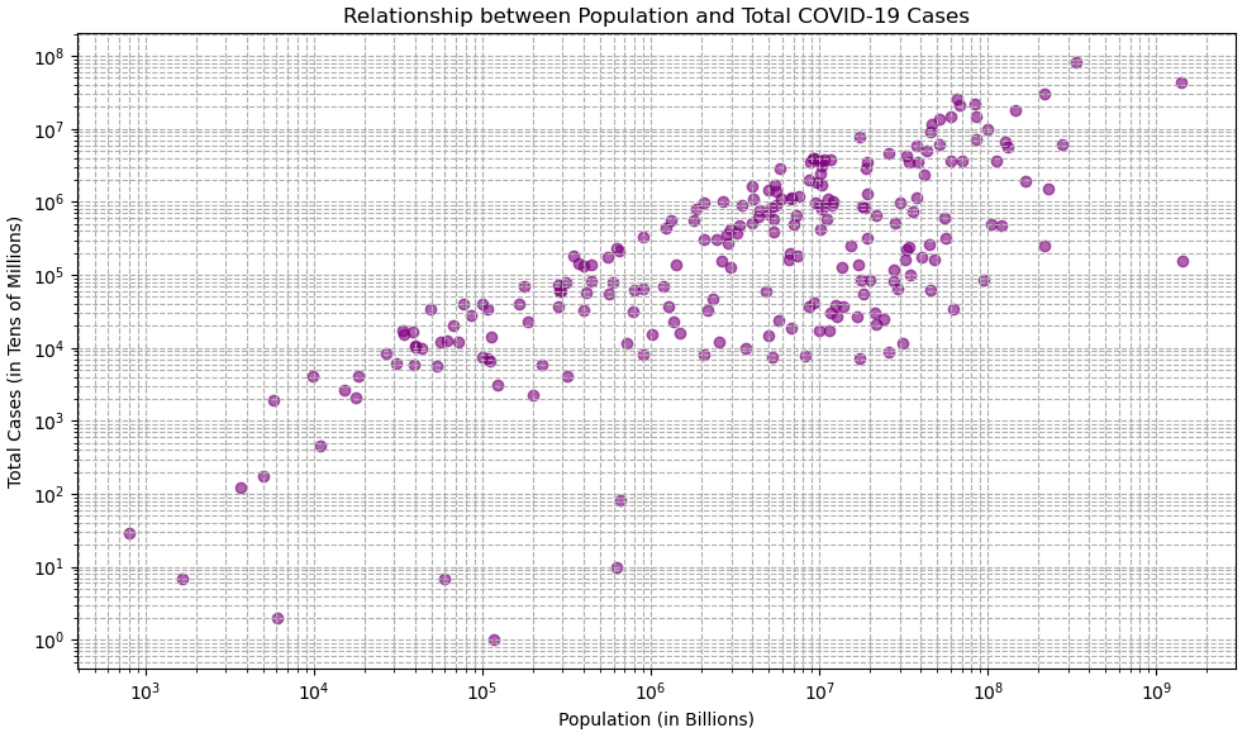
In [28]:
```
continent_cases = df.groupby('Continent')['Total Cases'].sum().sort_values(asc

print("\nContinent-wise Total COVID-19 Cases:")
print(continent_cases)
```

```
Continent-wise Total COVID-19 Cases:
Continent
Europe                                180332483
Asia                                  140957179
Northern America                       85364770
Latin America and the Caribbean        67509231
Africa                                 11764207
Oceania                                 5647957
Name: Total Cases, dtype: int64
```

In [29]:
```
plt.figure(figsize=(12, 7))
continent_cases.plot(kind='bar', color='darkblue')
plt.title('Total COVID-19 Cases by Continent')
plt.xlabel('Continent')
plt.ylabel('Total Cases (in hundred millions)')
plt.xticks(rotation=45, ha='right')
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```



In [30]:
```
plt.figure(figsize=(10, 6))
plt.scatter(df['Population'], df['Total Cases'], color='purple', alpha=0.6)
plt.title('Relationship between Population and Total COVID-19 Cases')
plt.xlabel('Population (in Billions)')
plt.ylabel('Total Cases (in Tens of Millions)')
plt.xscale('log') # Use log scale for population for better visualization of c
plt.yscale('log') # Use log scale for total cases
plt.grid(True, which="both", ls="--")
plt.tight_layout()
plt.show()
```



In [39]:
```
# First, let's check if the original column exists
if 'Tot Deaths/1M pop' in df.columns:
    # If it exists, rename it
    df.rename(columns={'Tot Deaths/1M pop': 'Total Deaths/1M pop'}, inplace=Tr
else:
    # If the column name is different, you might need to check your DataFrame
    print("Column names in DataFrame:", df.columns.tolist())
    # You may need to identify the correct column name from the list above

# After ensuring the column exists, calculate the correlation coefficient
if 'Total Deaths/1M pop' in df.columns:
    correlation_value = df['Total Cases'].corr(df['Total Deaths/1M pop'])
    print(f"Correlation between Total Cases and Total Deaths/1M pop: {correlat

    # Create a correlation matrix for the heatmap
    corr_matrix = df[['Total Cases', 'Total Deaths/1M pop', 'Population', 'Dea
else:
    print("Column 'Total Deaths/1M pop' not found. Please check column names."
```

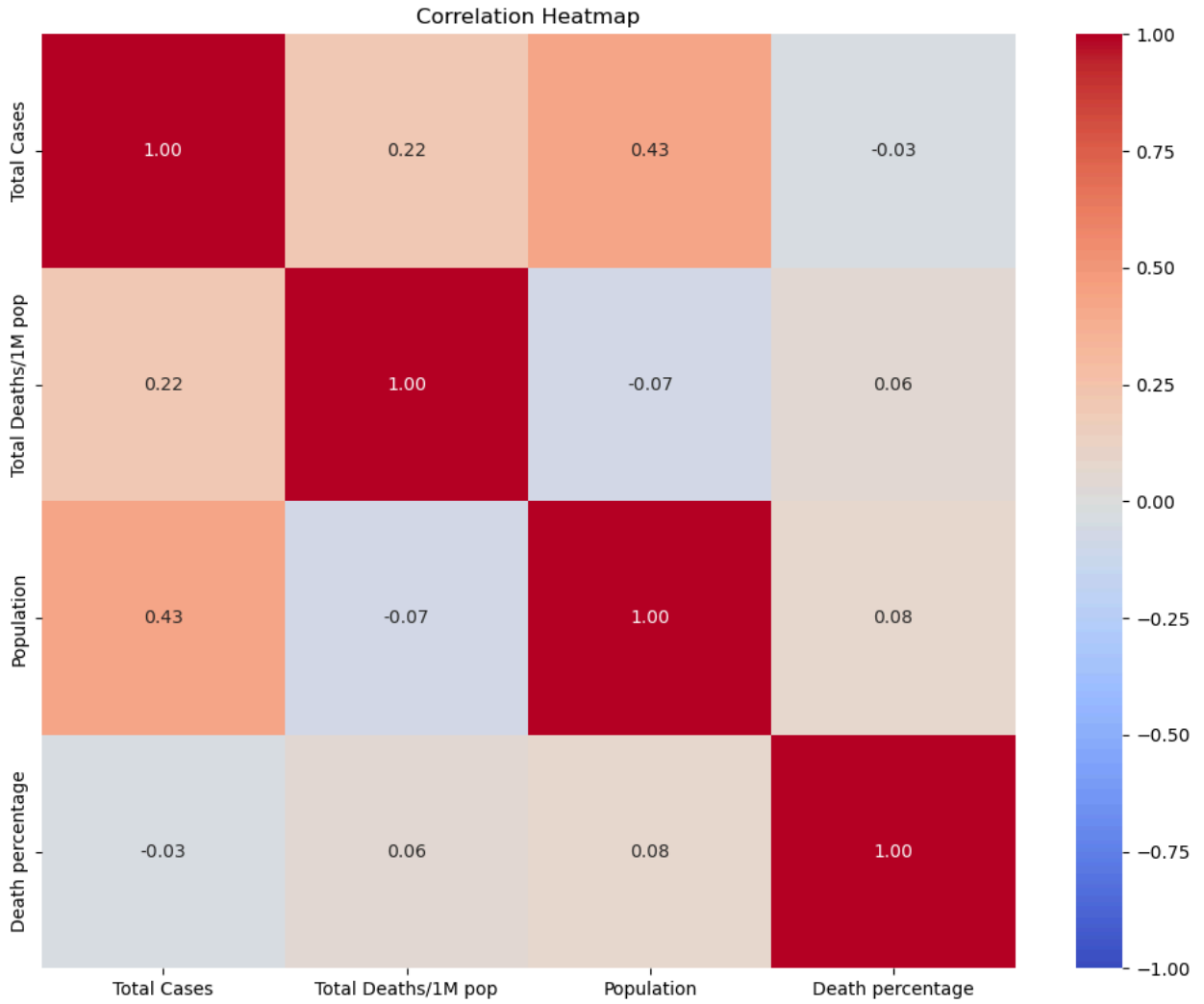
```
Column names in DataFrame: ['Country', 'Other names', 'ISO 3166-1 alpha-3 COD
E', 'Population', 'Continent', 'Total Cases', 'Total Deaths', 'Tot\xa0Cases//1M
pop', 'Tot\xa0Deaths/1M pop', 'Death percentage']
Column 'Total Deaths/1M pop' not found. Please check column names.
```

In [41]:
```
# Rename the column with the correct original name (with non-breaking space)
df.rename(columns={'Tot\xa0Deaths/1M pop': 'Total Deaths/1M pop'}, inplace=Tru

# Calculate the correlation coefficient
correlation_value = df['Total Cases'].corr(df['Total Deaths/1M pop'])

print(f"Correlation between Total Cases and Total Deaths/1M pop: {correlation_
```

```
# Create a correlation matrix for the heatmap
corr_matrix = df[['Total Cases', 'Total Deaths/1M pop', 'Population', 'Death p
```

```
Correlation between Total Cases and Total Deaths/1M pop: 0.22
```

In [42]:
```
# Create a heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, fmt='.2
plt.title('Correlation Heatmap')
plt.tight_layout()
plt.show()
```



Overview of Dataset The final dataset consists of 225 rows and 10 columns, providing a snapshot of COVID-19 statistics globally. Initial data cleaning was necessary to resolve character encoding issues in column names (like Tot Cases/1M pop) and handle missing categorical data. Specifically, null values in Other names were imputed with the corresponding Country name, and missing ISO 3166-1 alpha-3 CODE values were replaced with 'MISSING'. All columns now have zero null values, ensuring reliable calculations. The statistical summary highlights a large

variation in population and case counts, suggesting the presence of major global outliers.

Key Observations and Insights from Visualizations The data clearly identifies the USA and India as having the largest absolute burdens, leading both the Total Cases (≈81.8 million and ≈43.0 million, respectively) and Total Deaths lists. However, the Fatality Rate (Death percentage) tells a different story: Yemen (18.15%) and Western Sahara (10.00%) exhibit the highest percentage of deaths relative to confirmed cases. This likely reflects severe healthcare challenges, limited testing leading to underreporting of mild cases, or a combination of both in those regions. Geographically, Europe and Asia dominate the global case count, underscoring their vast populations and the widespread impact of the pandemic across large continents.

Conclusion The analysis reveals that the impact of the COVID-19 pandemic cannot be summarized by a single metric. While large, wealthy nations like the USA have the largest absolute counts due to high case numbers, smaller, often developing nations like Yemen faced the most severe outcomes relative to their confirmed cases. The weak positive correlation (0.22) between Total Cases and Total Deaths/ 1M pop confirms this duality, indicating that a country's high total case count is not a strong predictor of its population-adjusted mortality rate. The true severity of the pandemic appears to be more closely tied to local healthcare infrastructure, demographics, and public health response quality, as demonstrated by the contrasting outlier positions of the USA and China in the population-vs-cases scatter plot.

In [ ]: