- Created By: Blessy louis
- Created On: 04.03.2024
- Purpose: Prodigy Infotech Data Science internship Task_1

Aim:

To create a bar chart or histogram to visualize the distribution of a categorical or continuous variable

Import necessary libraries

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

Load the dataset

In [3]:
```
df = pd.read_csv('task1.csv')
df
```

Out[3]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...	2013	
0	Aruba	ABW	Population, total	SP.POP.TOTL	54608.0	55811.0	56682.0	57475.0	58178.0	58782.0	...	102880.0	1
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	130692579.0	134169237.0	137835590.0	141630546.0	145605995.0	149742351.0	...	567892149.0	5836
2	Afghanistan	AFG	Population, total	SP.POP.TOTL	8622466.0	8790140.0	8969047.0	9157465.0	9355514.0	9565147.0	...	31541209.0	327
3	Africa Western and Central	AFW	Population, total	SP.POP.TOTL	97256290.0	99314028.0	101445032.0	103667517.0	105959979.0	108336203.0	...	387204553.0	3978
4	Angola	AGO	Population, total	SP.POP.TOTL	5357195.0	5441333.0	5521400.0	5599827.0	5673199.0	5736582.0	...	26147002.0	271
...	...	...	...	...	...	...	...	...	...	...	...	...	
261	Kosovo	XKX	Population, total	SP.POP.TOTL	947000.0	966000.0	994000.0	1022000.0	1050000.0	1078000.0	...	1818117.0	18
262	Yemen, Rep.	YEM	Population, total	SP.POP.TOTL	5542459.0	5646668.0	5753386.0	5860197.0	5973803.0	6097298.0	...	26984002.0	277
263	South Africa	ZAF	Population, total	SP.POP.TOTL	16520441.0	16989464.0	17503133.0	18042215.0	18603097.0	19187194.0	...	53873616.0	547
264	Zambia	ZMB	Population, total	SP.POP.TOTL	3119430.0	3219451.0	3323427.0	3431381.0	3542764.0	3658024.0	...	15234976.0	157
265	Zimbabwe	ZWE	Population, total	SP.POP.TOTL	3806310.0	3925952.0	4049778.0	4177931.0	4310332.0	4447149.0	...	13555422.0	138

266 rows × 67 columns

Exploratory data Analysis

In [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266 entries, 0 to 265
Data columns (total 67 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Country Name    266 non-null    object 
 1   Country Code    266 non-null    object 
 2   Indicator Name  266 non-null    object 
 3   Indicator Code  266 non-null    object 
 4   1960            264 non-null    float64
 5   1961            264 non-null    float64
 6   1962            264 non-null    float64
 7   1963            264 non-null    float64
 8   1964            264 non-null    float64
 9   1965            264 non-null    float64
 10  1966            264 non-null    float64
 11  1967            264 non-null    float64
 12  1968            264 non-null    float64
 13  1969            264 non-null    float64
 14  1970            264 non-null    float64
 15  1971            264 non-null    float64
 16  1972            264 non-null    float64
 17  1973            264 non-null    float64
 18  1974            264 non-null    float64
 19  1975            264 non-null    float64
 20  1976            264 non-null    float64
 21  1977            264 non-null    float64
 22  1978            264 non-null    float64
 23  1979            264 non-null    float64
 24  1980            264 non-null    float64
 25  1981            264 non-null    float64
 26  1982            264 non-null    float64
 27  1983            264 non-null    float64
 28  1984            264 non-null    float64
 29  1985            264 non-null    float64
 30  1986            264 non-null    float64
 31  1987            264 non-null    float64
 32  1988            264 non-null    float64
 33  1989            264 non-null    float64
 34  1990            265 non-null    float64
 35  1991            265 non-null    float64
 36  1992            265 non-null    float64
 37  1993            265 non-null    float64
 38  1994            265 non-null    float64
```

```
 39  1995            265 non-null    float64
 40  1996            265 non-null    float64
 41  1997            265 non-null    float64
 42  1998            265 non-null    float64
 43  1999            265 non-null    float64
 44  2000            265 non-null    float64
 45  2001            265 non-null    float64
 46  2002            265 non-null    float64
 47  2003            265 non-null    float64
 48  2004            265 non-null    float64
 49  2005            265 non-null    float64
 50  2006            265 non-null    float64
 51  2007            265 non-null    float64
 52  2008            265 non-null    float64
 53  2009            265 non-null    float64
 54  2010            265 non-null    float64
 55  2011            265 non-null    float64
 56  2012            265 non-null    float64
 57  2013            265 non-null    float64
 58  2014            265 non-null    float64
 59  2015            265 non-null    float64
 60  2016            265 non-null    float64
 61  2017            265 non-null    float64
 62  2018            265 non-null    float64
 63  2019            265 non-null    float64
 64  2020            265 non-null    float64
 65  2021            265 non-null    float64
 66  2022            265 non-null    float64
dtypes: float64(63), object(4)
memory usage: 139.4+ KB
```

In [5]:
```
df.describe()
```

Out[5]:

	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969
count	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02
mean	1.172712e+08	1.188807e+08	1.210511e+08	1.237333e+08	1.264378e+08	1.291813e+08	1.320404e+08	1.348980e+08	1.378358e+08	1.408789e+08
std	3.695439e+08	3.740897e+08	3.808061e+08	3.895039e+08	3.982439e+08	4.071153e+08	4.164504e+08	4.257424e+08	4.353218e+08	4.452927e+08
min	2.646000e+03	2.888000e+03	3.171000e+03	3.481000e+03	3.811000e+03	4.161000e+03	4.531000e+03	4.930000e+03	5.354000e+03	5.646000e+03
25%	5.132212e+05	5.231345e+05	5.337595e+05	5.449288e+05	5.566630e+05	5.651150e+05	5.691470e+05	5.773872e+05	5.832700e+05	5.875942e+05
50%	3.757486e+06	3.887144e+06	4.023896e+06	4.139356e+06	4.224612e+06	4.277636e+06	4.331825e+06	4.385700e+06	4.450934e+06	4.530800e+06
75%	2.670606e+07	2.748694e+07	2.830289e+07	2.914708e+07	3.001684e+07	3.084892e+07	3.163010e+07	3.209247e+07	3.249927e+07	3.277149e+07
max	3.031474e+09	3.072422e+09	3.126850e+09	3.193429e+09	3.260442e+09	3.328209e+09	3.398480e+09	3.468371e+09	3.540164e+09	3.614573e+09

8 rows × 63 columns

In [6]:
```
df.shape
```

Out[6]:
```
(266, 67)
```

In [7]:
```
df['Indicator Name'].unique()
```

Out[7]:
```
array(['Population, total'], dtype=object)
```

In [8]:
```
df['Indicator Code'].unique()
```

Out[8]:
```
array(['SP.POP.TOTL'], dtype=object)
```

In [9]:
```
df['Country Code'].unique()
```

Out[9]:
```
array(['ABW', 'AFE', 'AFG', 'AFW', 'AGO', 'ALB', 'AND', 'ARB', 'ARE',
       'ARG', 'ARM', 'ASM', 'ATG', 'AUS', 'AUT', 'AZE', 'BDI', 'BEL',
       'BEN', 'BFA', 'BGD', 'BGR', 'BHR', 'BHS', 'BIH', 'BLR', 'BLZ',
       'BMU', 'BOL', 'BRA', 'BRB', 'BRN', 'BTN', 'BWA', 'CAF', 'CAN',
       'CEB', 'CHE', 'CHI', 'CHL', 'CHN', 'CIV', 'CMR', 'COD', 'COG',
       'COL', 'COM', 'CPV', 'CRI', 'CSS', 'CUB', 'CUW', 'CYM', 'CYP',
       'CZE', 'DEU', 'DJI', 'DMA', 'DNK', 'DOM', 'DZA', 'EAP', 'EAR',
       'EAS', 'ECA', 'ECS', 'ECU', 'EGY', 'EMU', 'ERI', 'ESP', 'EST',
       'ETH', 'EUU', 'FCS', 'FIN', 'FJI', 'FRA', 'FRO', 'FSM', 'GAB',
       'GBR', 'GEO', 'GHA', 'GIB', 'GIN', 'GMB', 'GNB', 'GNQ', 'GRC',
       'GRD', 'GRL', 'GTM', 'GUM', 'GUY', 'HIC', 'HKG', 'HND', 'HPC',
       'HRV', 'HTI', 'HUN', 'IBD', 'IBT', 'IDA', 'IDB', 'IDN', 'IDX',
       'IMN', 'IND', 'INX', 'IRL', 'IRN', 'IRQ', 'ISL', 'ISR', 'ITA',
       'JAM', 'JOR', 'JPN', 'KAZ', 'KEN', 'KGZ', 'KHM', 'KIR', 'KNA',
       'KOR', 'KWT', 'LAC', 'LAO', 'LBN', 'LBR', 'LBY', 'LCA', 'LCN',
       'LDC', 'LIC', 'LIE', 'LKA', 'LMC', 'LMY', 'LSO', 'LTE', 'LTU',
       'LUX', 'LVA', 'MAC', 'MAF', 'MAR', 'MCO', 'MDA', 'MDG', 'MDV',
       'MEA', 'MEX', 'MHL', 'MIC', 'MKD', 'MLI', 'MLT', 'MMR', 'MNA',
       'MNE', 'MNG', 'MNP', 'MOZ', 'MRT', 'MUS', 'MWI', 'MYS', 'NAC',
       'NAM', 'NCL', 'NER', 'NGA', 'NIC', 'NLD', 'NOR', 'NPL', 'NRU',
       'NZL', 'OED', 'OMN', 'OSS', 'PAK', 'PAN', 'PER', 'PHL', 'PLW',
       'PNG', 'POL', 'PRE', 'PRI', 'PRK', 'PRT', 'PRY', 'PSE', 'PSS',
       'PST', 'PYF', 'QAT', 'ROU', 'RUS', 'RWA', 'SAS', 'SAU', 'SDN',
       'SEN', 'SGP', 'SLB', 'SLE', 'SLV', 'SMR', 'SOM', 'SRB', 'SSA',
       'SSD', 'SSF', 'SST', 'STP', 'SUR', 'SVK', 'SVN', 'SWE', 'SWZ',
       'SXM', 'SYC', 'SYR', 'TCA', 'TCD', 'TEA', 'TEC', 'TGO', 'THA',
       'TJK', 'TKM', 'TLA', 'TLS', 'TMN', 'TON', 'TSA', 'TSS', 'TTO',
       'TUN', 'TUR', 'TUV', 'TZA', 'UGA', 'UKR', 'UMC', 'URY', 'USA',
       'UZB', 'VCT', 'VEN', 'VGB', 'VIR', 'VNM', 'VUT', 'WLD', 'WSM',
       'XKX', 'YEM', 'ZAF', 'ZMB', 'ZWE'], dtype=object)
```

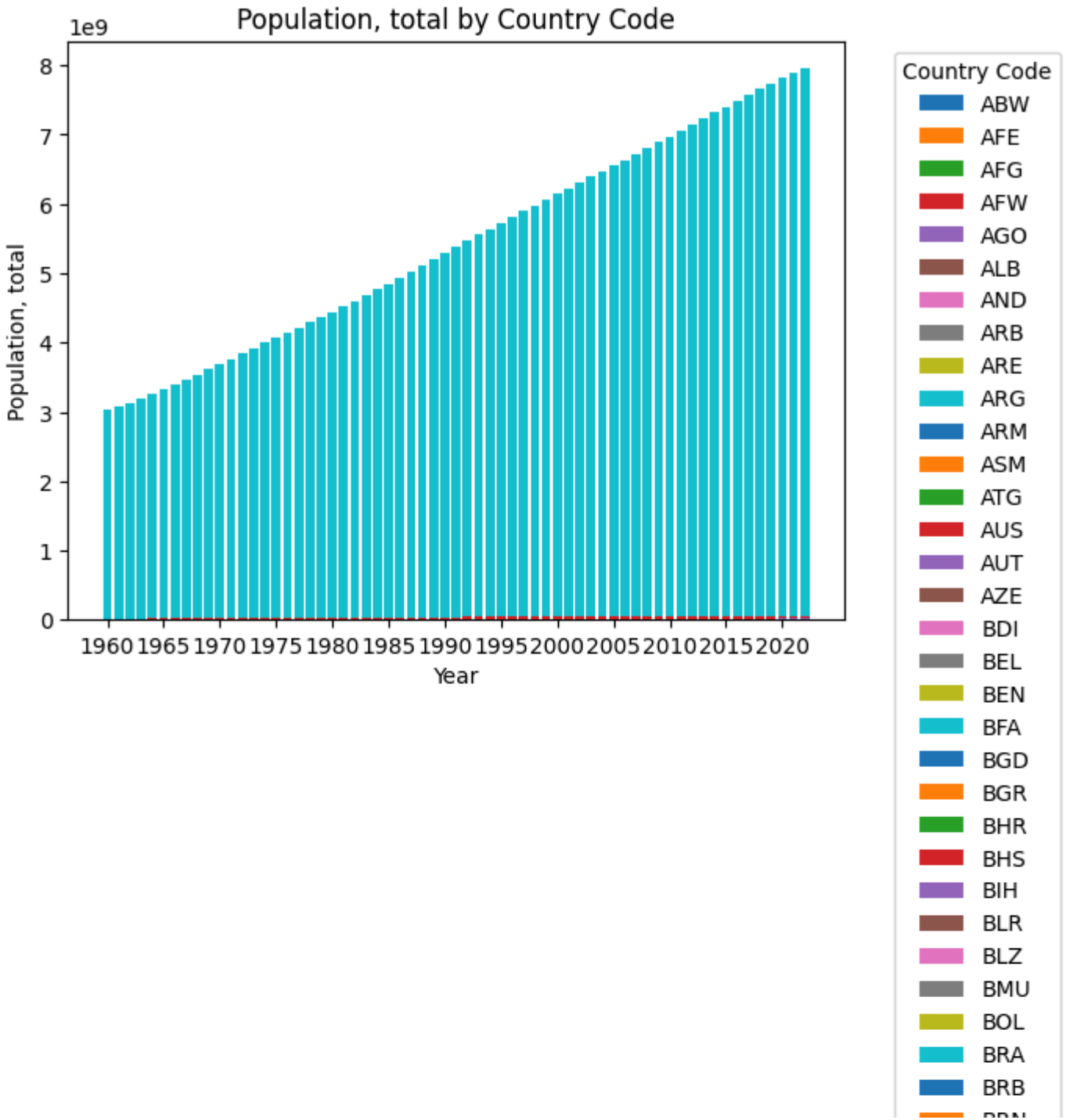
data Visulization

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

Out[10]:
```
Index(['Country Name', 'Country Code', 'Indicator Name', 'Indicator Code',
       '1960', '1961', '1962', '1963', '1964', '1965', '1966', '1967', '1968',
       '1969', '1970', '1971', '1972', '1973', '1974', '1975', '1976', '1977',
       '1978', '1979', '1980', '1981', '1982', '1983', '1984', '1985', '1986',
       '1987', '1988', '1989', '1990', '1991', '1992', '1993', '1994', '1995',
       '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
       '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013',
       '2014', '2015', '2016', '2017', '2018', '2019', '2020', '2021', '2022'],
      dtype='object')
```

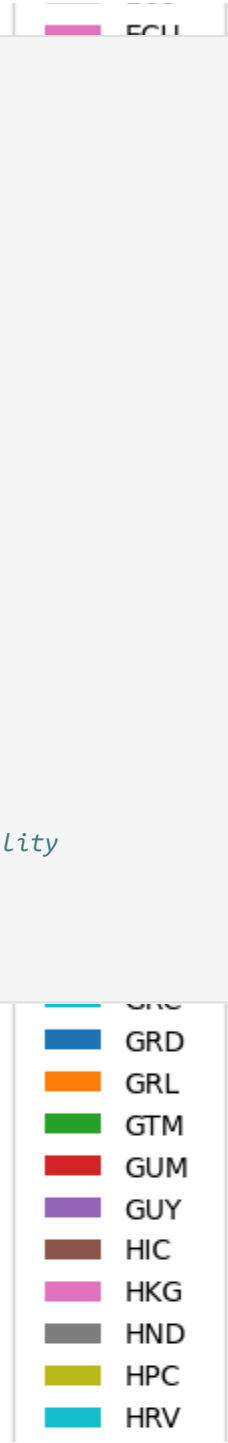
In [11]:
```
# Choose the indicator and years you want to visualize
indicator_name = 'Population, total'  # Population indicator
years = range(1960, 2023)  # Range of years in the dataset

# Loop through each country code
for index, row in df.iterrows():
    # Extract the country code and indicator values
    country_code = row['Country Code']
    values = row[4:]  # Extract population values starting from the 5th column

    # Plot a bar chart for the current country
    plt.bar(years, values, label=country_code)

# Set labels and title
plt.xlabel('Year')
plt.ylabel(indicator_name)
plt.title(f'{indicator_name} by Country Code')
plt.legend(title='Country Code', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.xticks(range(1960, 2023, 5))  # Show ticks every 5 years for better readability
plt.tight_layout()

# Show the plot
plt.show()
```

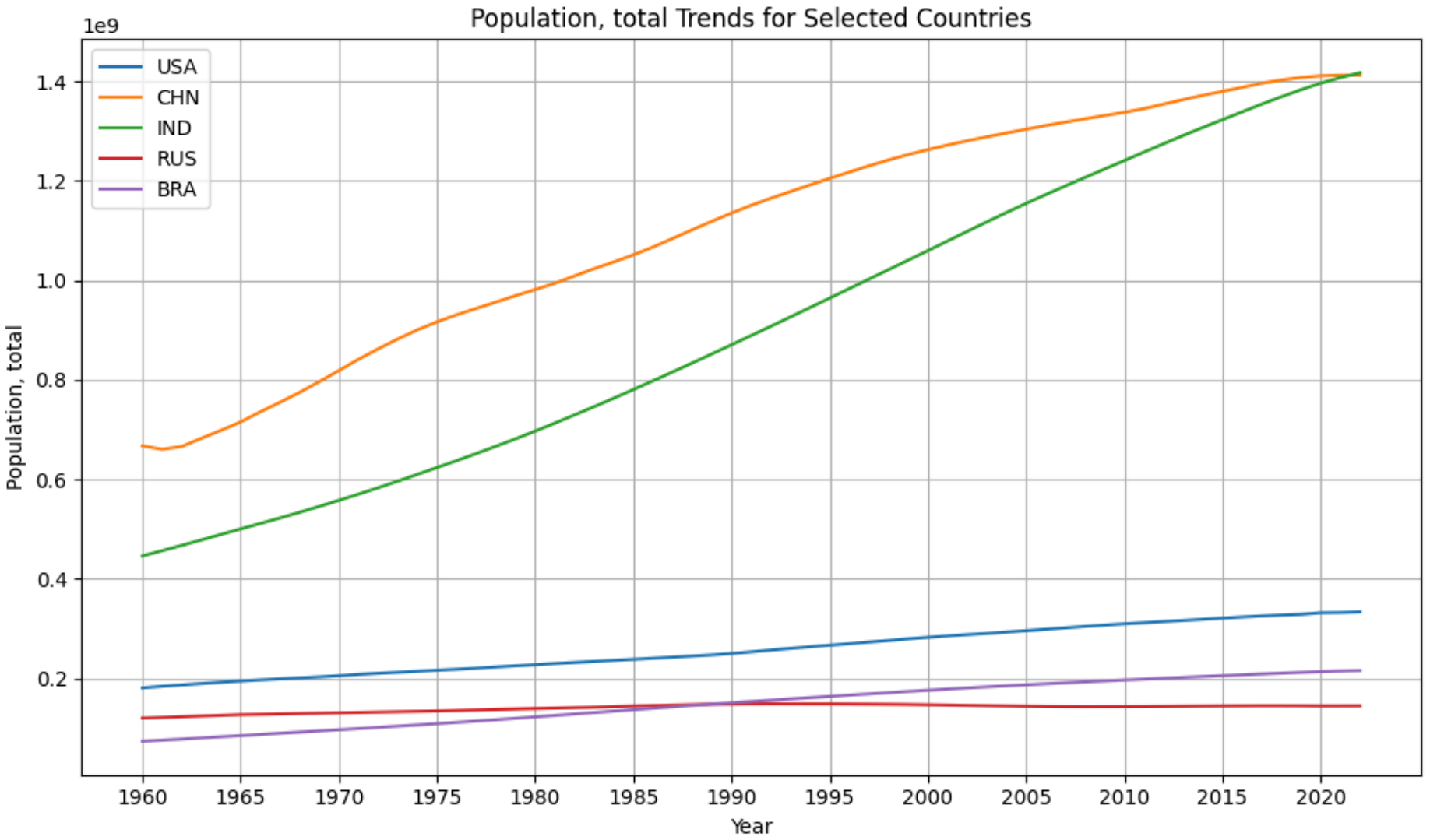
```
<ipython-input-11-4b92786a96c8>:20: UserWarning: Tight layout not applied. The bottom and top margins cannot be made large enoug
h to accommodate all axes decorations.
  plt.tight_layout()
```


Population, total by Country Code
1e9
Population, total
Year
0
1
2
3
4
5
6
7
8
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
Country Code
ABW
AFE
AFG
AFW
AGO
ALB
AND
ARB
ARE
ARG
ARM
ASM
ATG
AUS
AUT
AZE
BDI
BEL
BEN
BFA
BGD
BGR
BHR
BHS
BIH
BLR
BLZ
BMU
BOL
BRA
BRB


BRN
BTN
BWA
CAF
CAN
CEB
CHE
CHI
CHL
CHN
CIV
CMR
COD
COG
COL
COM
CPV
CRI
CSS
CUB
CUW
CYM
CYP
CZE
DEU
DJI
DMA
DNK
DOM
DZA
EAP
EAR
EAS
ECA
ECS

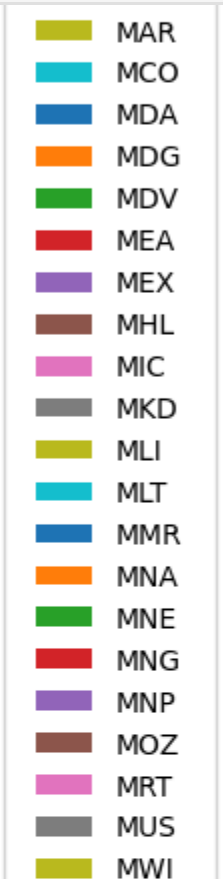
In [12]:
```
# Choose the countries you want to visualize
countries = ['USA', 'CHN', 'IND', 'RUS', 'BRA']  # Example countries

# Choose the indicator
indicator_name = 'Population, total'  # Population indicator

# Extract years and corresponding population values for the selected countries
years = range(1960, 2023)
country_data = df[df['Indicator Name'] == indicator_name]
country_data = country_data[country_data['Country Code'].isin(countries)]
country_data = country_data.set_index('Country Code')
country_data = country_data.loc[countries, '1960':]

# Plot the population trends for the selected countries
plt.figure(figsize=(10, 6))
for country in countries:
    plt.plot(years, country_data.loc[country], label=country)

# Set labels and title
plt.xlabel('Year')
plt.ylabel(indicator_name)
plt.title(f'{indicator_name} Trends for Selected Countries')
plt.legend()
plt.grid(True)
plt.xticks(range(1960, 2023, 5))  # Show ticks every 5 years for better readability
plt.tight_layout()

# Show the plot
plt.show()
```

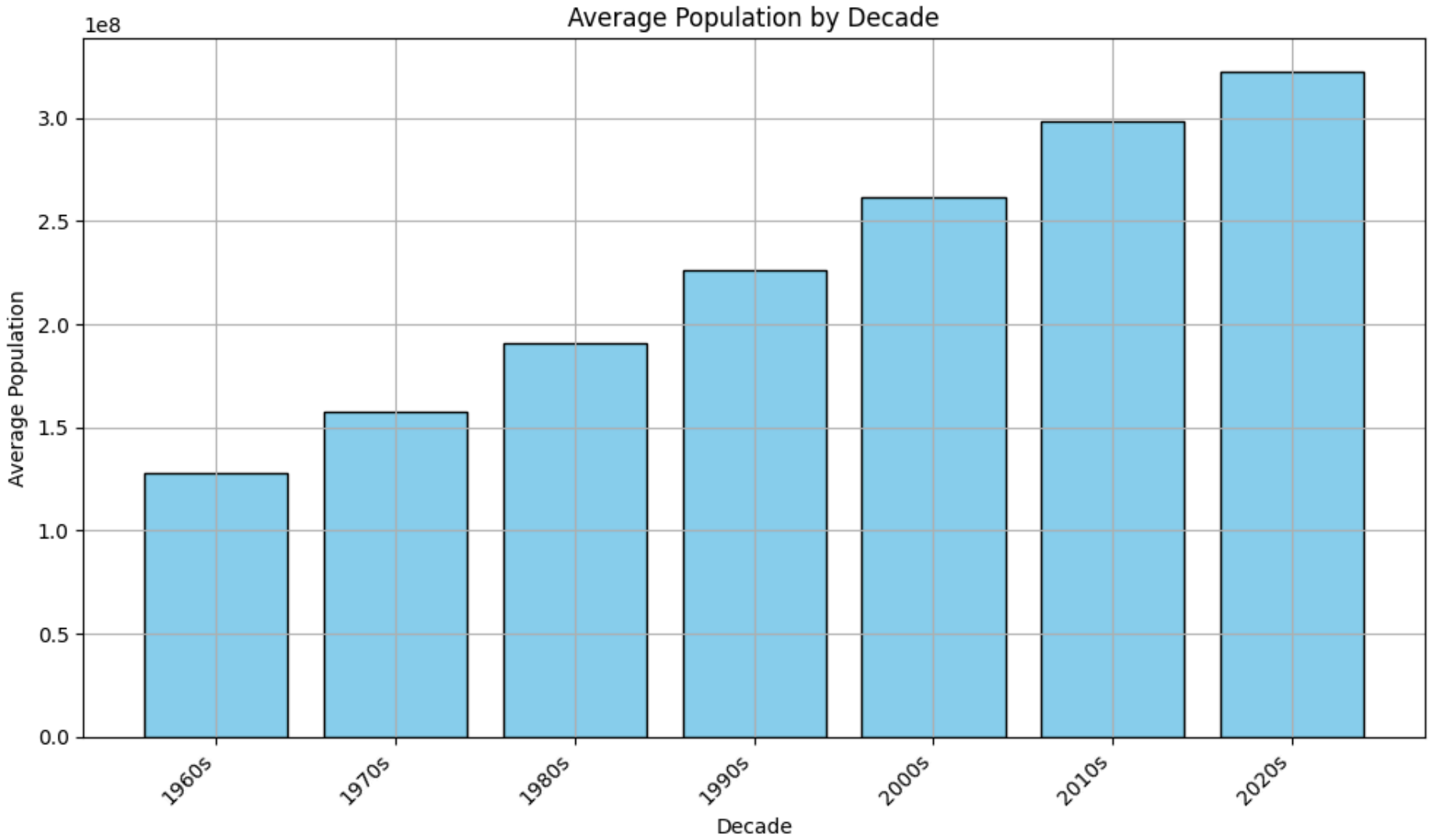




In [13]:
```
# Extract the population data columns (years)
population_columns = df.columns[4:]  # Assuming population data starts from the 5th column

# Group the population data by decade and calculate the average population for each decade
decades_population = {}
for year in population_columns:
    decade = year[:3] + '0s'
    if decade not in decades_population:
        decades_population[decade] = []
```

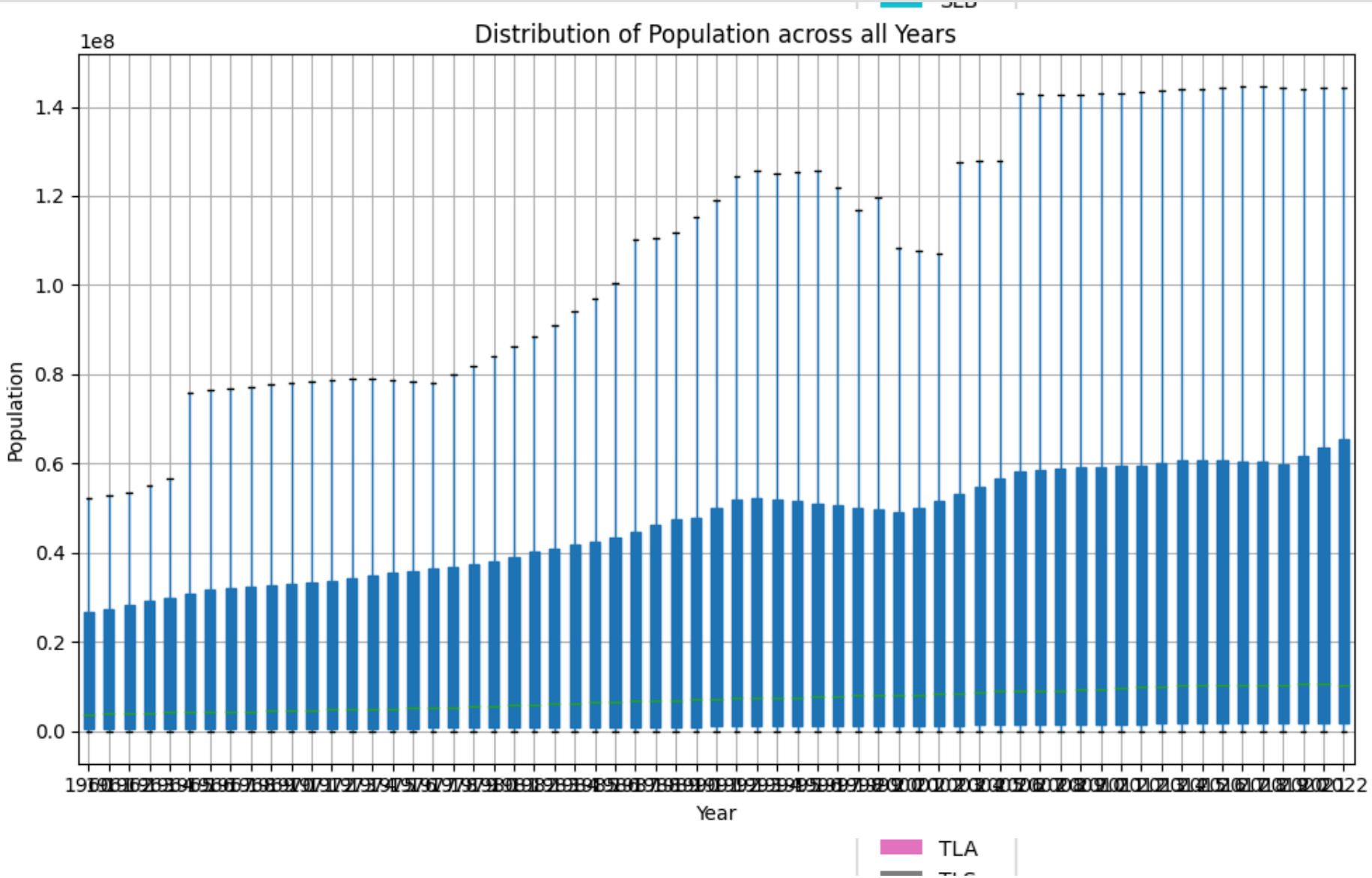
```
    decades_population[decade].extend(df[year].dropna())

# Calculate the average population for each decade
average_population = {decade: sum(populations) / len(populations) for decade, populations in decades_population.items()}

# Create a bar plot for average population by decade
plt.figure(figsize=(10, 6))
plt.bar(average_population.keys(), average_population.values(), color='skyblue', edgecolor='black')
plt.xlabel('Decade')
plt.ylabel('Average Population')
plt.title('Average Population by Decade')
plt.xticks(rotation=45, ha='right')
plt.grid(True)
plt.tight_layout()

# Show the plot
plt.show()
```



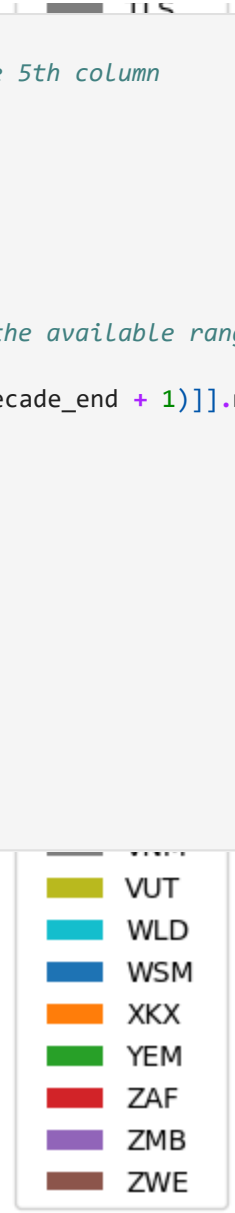


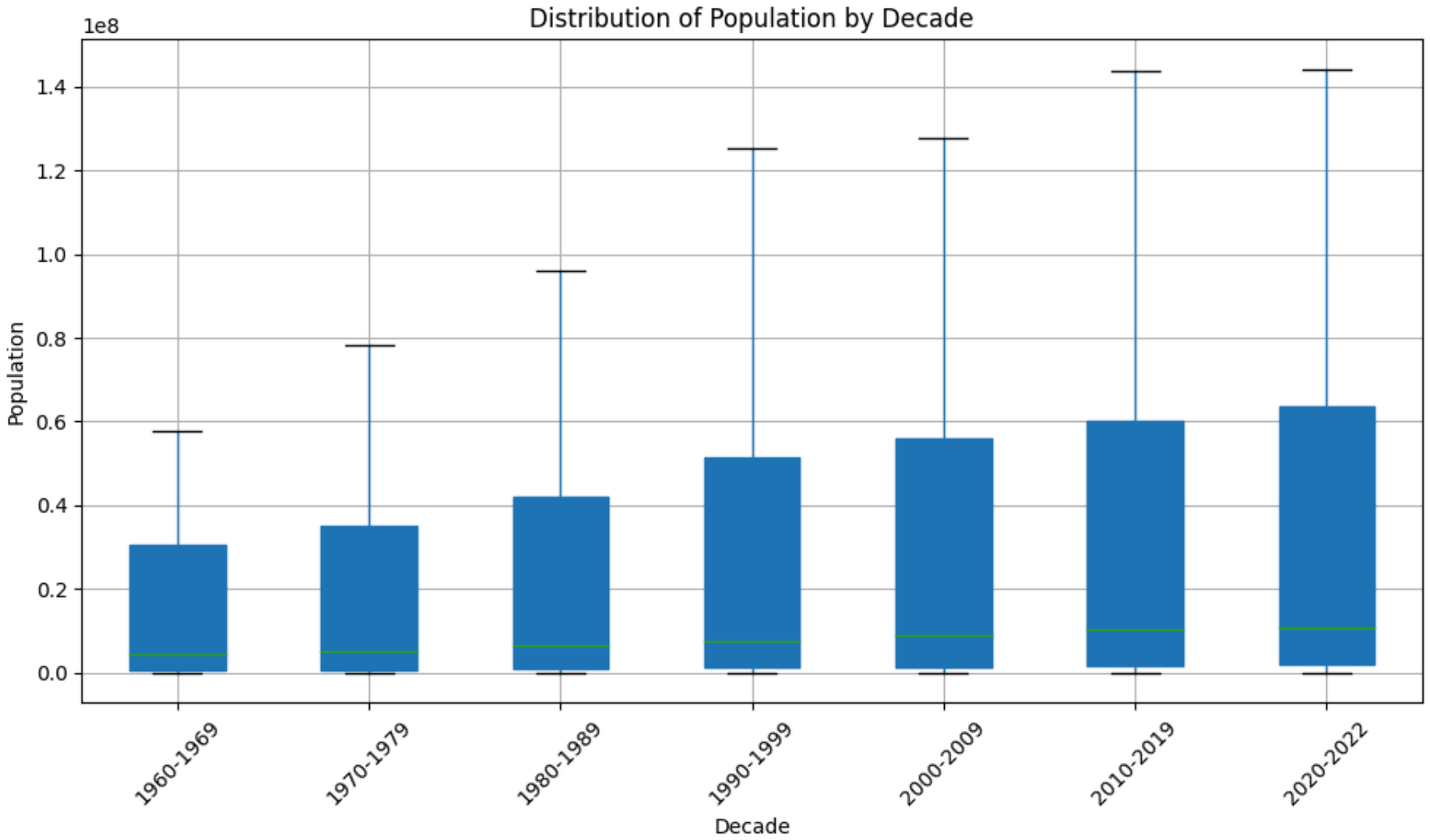
In [14]:
```
# Extract the population data columns (years)
population_columns = df.columns[4:]  # Assuming population data starts from the 5th column

# Extract population data
population_data = df[population_columns]

# Plot the box plot
plt.figure(figsize=(10, 6))
population_data.boxplot(patch_artist=True, showfliers=False)
```

```
plt.xlabel('Year')
plt.ylabel('Population')
plt.title('Distribution of Population across all Years')
plt.grid(True)
plt.tight_layout()

# Show the plot
plt.show()
```



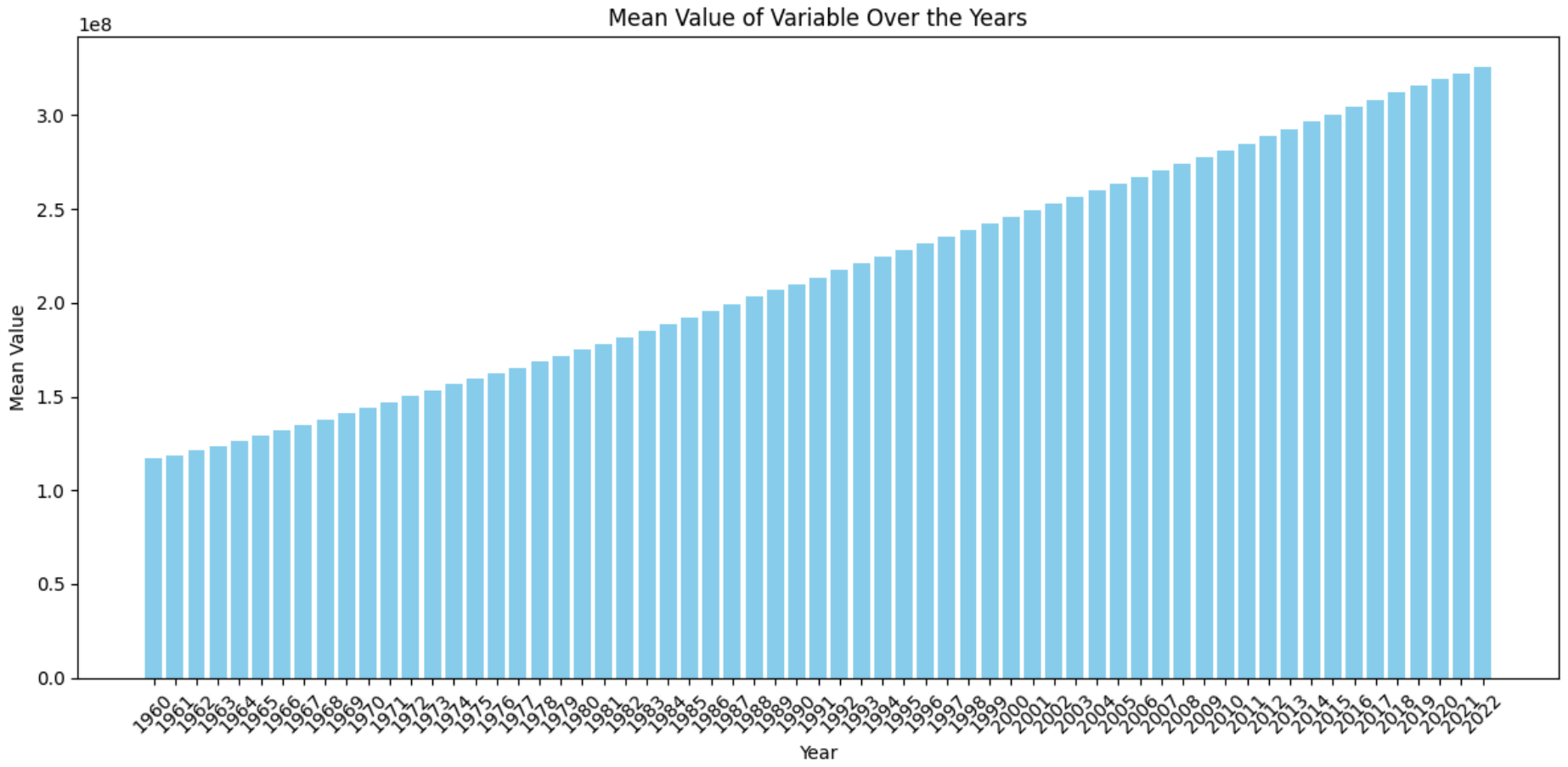
In [15]:
```
# Extract the population data columns (years)
population_columns = df.columns[4:]  # Assuming population data starts from the 5th column

# Group years into decades
decade_columns = [str(decade) + 's' for decade in range(1960, 2021, 10)]

# Create a DataFrame with population data grouped by decades
decade_data = pd.DataFrame()
for decade_start in range(1960, 2021, 10):
    decade_end = min(decade_start + 9, 2022)  # Ensure the end year is within the available range
    decade_name = str(decade_start) + '-' + str(decade_end)
    decade_data[decade_name] = df[[str(year) for year in range(decade_start, decade_end + 1)]].mean(axis=1)

# Plot the box plot for each decade
plt.figure(figsize=(10, 6))
decade_data.boxplot(patch_artist=True, showfliers=False)
plt.xlabel('Decade')
plt.ylabel('Population')
plt.title('Distribution of Population by Decade')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()

# Show the plot
plt.show()
```





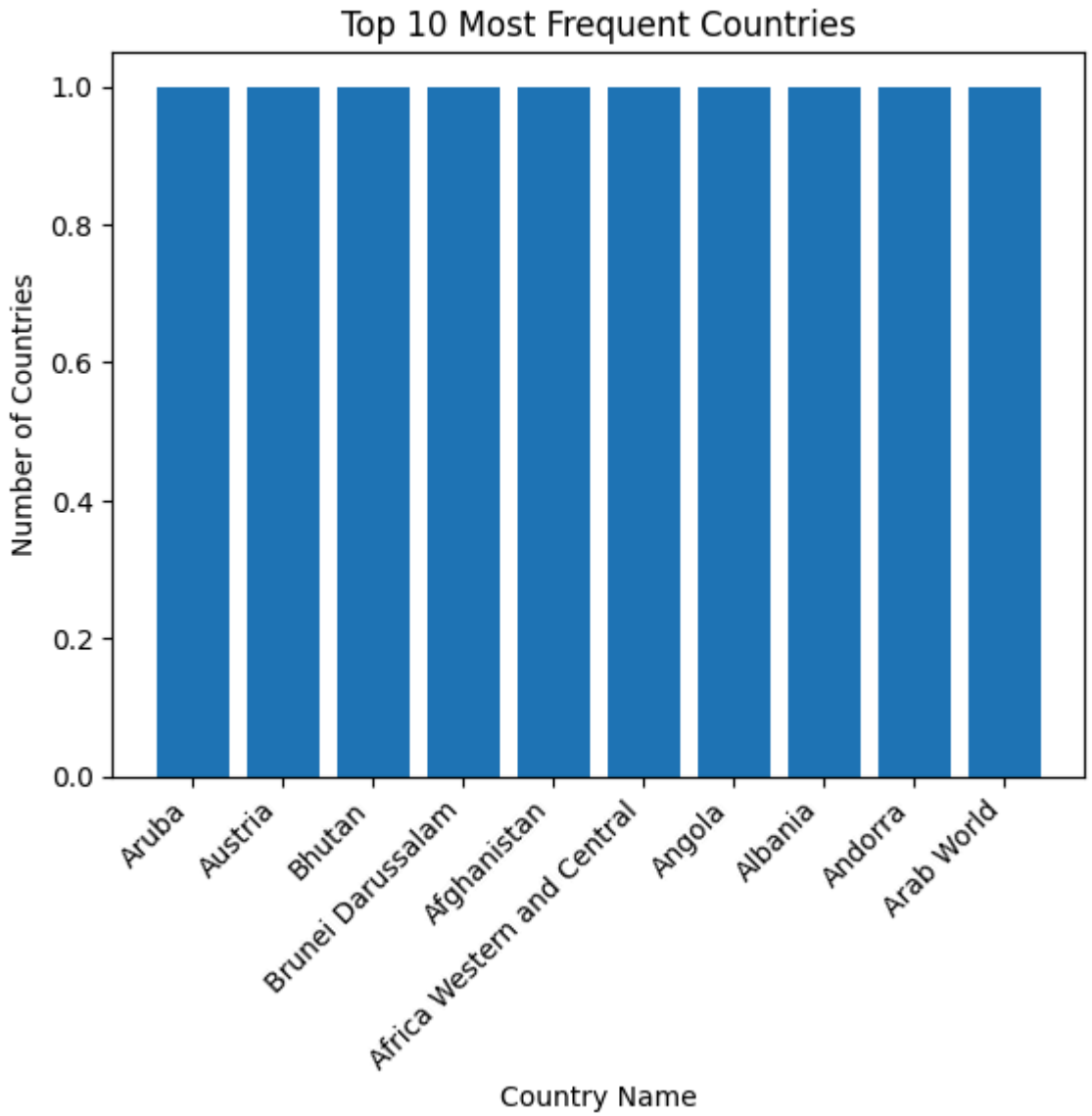
In [17]:
```
variable_column_name = 'Indicator Name'

# Extract the column of interest
variable_data = df.loc[:, '1960':'2022']  # Assuming the column name follows the year pattern

# You can compute some statistics if needed
# For example, you might want to compute the mean of each year
yearly_mean = variable_data.mean()
```

```
# Plotting
plt.figure(figsize=(12, 6))
plt.bar(yearly_mean.index, yearly_mean.values, color='skyblue')
plt.xlabel('Year')
plt.ylabel('Mean Value')
plt.title('Mean Value of Variable Over the Years')
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



In [19]:
```
# Sort countries by count in descending order
country_counts_sorted = country_counts.sort_values(ascending=False)

# Choose the top N countries to display (adjust N as needed)
top_n_countries = country_counts_sorted.index[:10]  # Display the top 10
```

```
# Filter data and counts for selected countries
filtered_data = df[df['Country Name'].isin(top_n_countries)]
filtered_counts = country_counts_sorted[top_n_countries]

# Create the bar chart
plt.bar(filtered_counts.index, filtered_counts.values)
plt.xlabel('Country Name')
plt.ylabel('Number of Countries')
plt.title('Top 10 Most Frequent Countries')
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for better readability
plt.show()
```


Top 10 Most Frequent Countries
Number of Countries
1.0
0.8
0.6
0.4
0.2
0.0
Aruba
Austria
Bhutan
Brunei Darussalam
Afghanistan
Africa Western and Central
Angola
Albania
Andorra
Arab World
Country Name