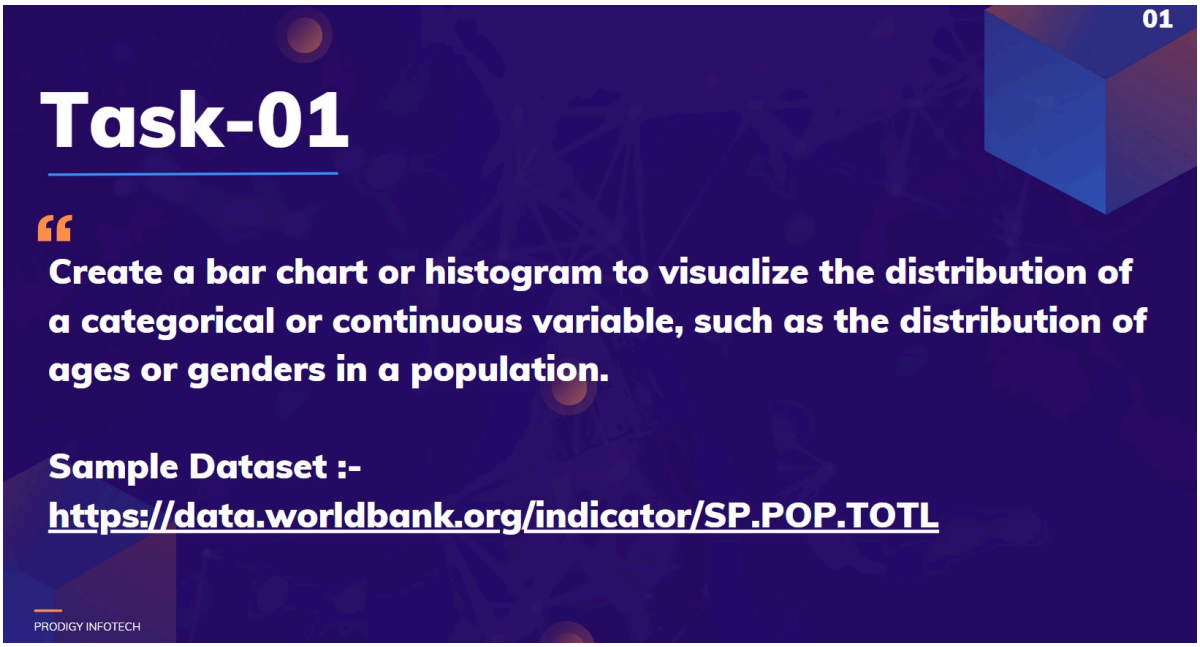
Project Name: Exploring Global Population Trends: A Data Visualization Project

By Mehul Chafekar

Project Introduction

- Understanding global population trends is crucial for governments, policymakers, researchers, and businesses.
- Population data offers insights into demographic changes, economic growth, and social dynamics.
- This project aims to visualize world population data from 1960 to 2023, leveraging Python and various data visualization libraries such as Matplotlib and Seaborn.
- By analyzing historical population data, we can uncover patterns, trends, and anomalies that can inform future planning and decision-making.



Project Summary

- This project utilizes a dataset containing population data for various countries over the years 1960 to 2023.
- The dataset includes columns such as Country Name, Country Code, Indicator Name, Indicator Code, and yearly population values.
- The main objectives of the project include:

  1. Data Cleaning and Preparation

  2. Descriptive Statistics

  3. Data Visualization

  4. Analysis and Insights

Business Objective

- The primary business objective of this project is to provide actionable insights into global population trends that can inform strategic planning and decision-making.
- By understanding historical population data and identifying key patterns, stakeholders can make data-driven decisions in areas such as:

1. **Urban Planning and Development**:
   - Plan for infrastructure development and resource allocation based on population growth trends.
2. **Healthcare Services**:
   - Anticipate healthcare needs and services for regions with rapidly growing populations.
3. **Market Research and Expansion**:
   - Identify potential markets for business expansion by analyzing population demographics.
4. **Policy Formulation**:
   - Develop policies that address the needs of diverse populations and promote sustainable growth.
5. **Educational Planning**:
   - Forecast educational needs and allocate resources to regions with high population growth.

Import Libraries

In [42]:
```
# Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the Dataset

In [43]:
```
# Load Dataset
df = pd.read_csv('population_Data.csv')
df
```

Out[43]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	
0	Aruba	ABW	Population, total	SP.POP.TOTL	54922	55578	56320	57002	
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	130072080	133534923	137171659	140945536	144
2	Afghanistan	AFG	Population, total	SP.POP.TOTL	9035043	9214083	9404406	9604487	9
3	Africa Western and Central	AFW	Population, total	SP.POP.TOTL	97630925	99706674	101854756	104089175	106
4	Angola	AGO	Population, total	SP.POP.TOTL	5231654	5301583	5354310	5408320	5
...	...	...	...	...	...	...	...	...	
259	Kosovo	XKX	Population, total	SP.POP.TOTL	984846	1011421	1036950	1062737	1
260	Yemen, Rep.	YEM	Population, total	SP.POP.TOTL	5532301	5655232	5782221	5911135	6
261	South Africa	ZAF	Population, total	SP.POP.TOTL	16440172	16908035	17418522	17954564	18
262	Zambia	ZMB	Population, total	SP.POP.TOTL	3153729	3254086	3358099	3465907	3
263	Zimbabwe	ZWE	Population, total	SP.POP.TOTL	3809389	3930401	4055959	4185877	4

264 rows × 68 columns

Dataset first View

In [44]:
```
df.head(10)
```

Out[44]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	
0	Aruba	ABW	Population, total	SP.POP.TOTL	54922	55578	56320	57002	5
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	130072080	133534923	137171659	140945536	14490
2	Afghanistan	AFG	Population, total	SP.POP.TOTL	9035043	9214083	9404406	9604487	981
3	Africa Western and Central	AFW	Population, total	SP.POP.TOTL	97630925	99706674	101854756	104089175	10638
4	Angola	AGO	Population, total	SP.POP.TOTL	5231654	5301583	5354310	5408320	546
5	Albania	ALB	Population, total	SP.POP.TOTL	1608800	1659800	1711319	1762621	181
6	Andorra	AND	Population, total	SP.POP.TOTL	9510	10283	11086	11915	1
7	Arab World	ARB	Population, total	SP.POP.TOTL	91540853	93931683	96428599	99038509	10172
8	United Arab Emirates	ARE	Population, total	SP.POP.TOTL	131334	137989	144946	152211	15
9	Argentina	ARG	Population, total	SP.POP.TOTL	20386045	20726276	21072538	21421705	2176

10 rows × 68 columns

Dataset Last Rows

In [46]:
```
df.tail()
```

Out[46]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964
259	Kosovo	XKX	Population, total	SP.POP.TOTL	984846	1011421	1036950	1062737	1090270
260	Yemen, Rep.	YEM	Population, total	SP.POP.TOTL	5532301	5655232	5782221	5911135	6048006
261	South Africa	ZAF	Population, total	SP.POP.TOTL	16440172	16908035	17418522	17954564	18511361
262	Zambia	ZMB	Population, total	SP.POP.TOTL	3153729	3254086	3358099	3465907	3577017
263	Zimbabwe	ZWE	Population, total	SP.POP.TOTL	3809389	3930401	4055959	4185877	4320006

5 rows × 68 columns

Dataset Rows & Columns count

In [47]:
```
# Dataset Rows & Columns count
df.shape
```

Out[47]: (264, 68)

Dataset Information

In [48]:
```
# Information about the dataset
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 264 entries, 0 to 263
Data columns (total 68 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Country Name    264 non-null    object
 1   Country Code    264 non-null    object
 2   Indicator Name  264 non-null    object
 3   Indicator Code  264 non-null    object
 4   1960            264 non-null    int64
 5   1961            264 non-null    int64
 6   1962            264 non-null    int64
 7   1963            264 non-null    int64
 8   1964            264 non-null    int64
 9   1965            264 non-null    int64
 10  1966            264 non-null    int64
 11  1967            264 non-null    int64
 12  1968            264 non-null    int64
 13  1969            264 non-null    int64
 14  1970            264 non-null    int64
 15  1971            264 non-null    int64
 16  1972            264 non-null    int64
 17  1973            264 non-null    int64
 18  1974            264 non-null    int64
 19  1975            264 non-null    int64
 20  1976            264 non-null    int64
 21  1977            264 non-null    int64
 22  1978            264 non-null    int64
 23  1979            264 non-null    int64
 24  1980            264 non-null    int64
 25  1981            264 non-null    int64
 26  1982            264 non-null    int64
 27  1983            264 non-null    int64
 28  1984            264 non-null    int64
 29  1985            264 non-null    int64
 30  1986            264 non-null    int64
 31  1987            264 non-null    int64
 32  1988            264 non-null    int64
 33  1989            264 non-null    int64
 34  1990            264 non-null    int64
 35  1991            264 non-null    int64
 36  1992            264 non-null    float64
 37  1993            264 non-null    int64
 38  1994            264 non-null    float64
 39  1995            264 non-null    int64
 40  1996            264 non-null    float64
 41  1997            264 non-null    float64
 42  1998            264 non-null    float64
 43  1999            264 non-null    int64
 44  2000            264 non-null    float64
 45  2001            264 non-null    float64
 46  2002            264 non-null    int64
 47  2003            264 non-null    int64
 48  2004            264 non-null    float64
 49  2005            264 non-null    float64
 50  2006            264 non-null    int64
 51  2007            264 non-null    float64
 52  2008            264 non-null    int64
 53  2009            264 non-null    int64
 54  2010            264 non-null    int64
 55  2011            264 non-null    float64
 56  2012            264 non-null    int64
 57  2013            264 non-null    int64
 58  2014            264 non-null    float64
```

```
 59  2015            264 non-null    float64
 60  2016            264 non-null    float64
 61  2017            264 non-null    float64
 62  2018            264 non-null    float64
 63  2019            264 non-null    int64
 64  2020            264 non-null    float64
 65  2021            264 non-null    int64
 66  2022            264 non-null    float64
 67  2023            264 non-null    int64
dtypes: float64(18), int64(46), object(4)
memory usage: 140.4+ KB
None
```

In [49]:
```
df.describe()
```

Out[49]:

	1960	1961	1962	1963	1964	1965	1
count	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e+02	2.640000e
mean	1.154482e+08	1.170540e+08	1.192163e+08	1.218881e+08	1.245838e+08	1.273114e+08	1.301584e
std	3.626524e+08	3.671661e+08	3.738304e+08	3.824609e+08	3.911398e+08	3.999257e+08	4.091871e
min	2.715000e+03	2.970000e+03	3.264000e+03	3.584000e+03	3.922000e+03	4.282000e+03	4.664000e
25%	5.152028e+05	5.255230e+05	5.363018e+05	5.475875e+05	5.593638e+05	5.675750e+05	5.711695e
50%	3.659633e+06	3.747132e+06	3.831900e+06	3.919710e+06	4.010150e+06	4.102976e+06	4.198738e
75%	2.686293e+07	2.761326e+07	2.837302e+07	2.915448e+07	2.995223e+07	3.075921e+07	3.147516e
max	3.021529e+09	3.062769e+09	3.117373e+09	3.184063e+09	3.251253e+09	3.318998e+09	3.389087e

8 rows × 64 columns

In [50]:
```
df.columns
```

Out[50]:
```
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',
       '2023'],
      dtype='object')
```

Data Cleaning and Preprocessing

Checking any duplicate values

In [51]:
```
## Checking any Duplicate Values
print(df.duplicated().sum())
```

```
0
```

In [52]:
```
Duplicate_values = df.duplicated().value_counts()
print(Duplicate_values)
```

```
False    264
dtype: int64
```

Checking any null values

In [53]:
```
print(df.isnull().sum())
```

```
Country Name      0
Country Code      0
Indicator Name    0
Indicator Code    0
1960              0
                 ..
2019              0
2020              0
2021              0
2022              0
2023              0
Length: 68, dtype: int64
```

Checking Unique value for each columns

In [54]:
```
# Check Unique Values for each variable.
unique_value = df.nunique()
unique_value
```

Out[54]:
```
Country Name      264
Country Code      264
Indicator Name      1
Indicator Code      1
1960              260
                 ...
2019              262
2020              262
2021              262
2022              262
2023              262
Length: 68, dtype: int64
```

In [ ]:

In [55]:
```
#Let's look unique values for every columns
df.apply(lambda col: col.unique())
```

Out[55]:
```
Country Name      [Aruba, Africa Eastern and Southern, Afghanist...
Country Code      [ABW, AFE, AFG, AFW, AGO, ALB, AND, ARB, ARE, ...
Indicator Name                                  [Population, total]
Indicator Code                                        [SP.POP.TOTL]
1960              [54922, 130072080, 9035043, 97630925, 5231654,...
                                              ...
2019              [109203, 675950189, 37856121, 463365429, 32375...
2020              [108587.0, 694446100.0, 39068979.0, 474569351....
2021              [107700, 713090928, 40000412, 485920997, 34532...
2022              [107310.0, 731821393.0, 40578842.0, 497387180....
2023              [107359, 750503764, 41454761, 509398589, 36749...
Length: 68, dtype: object
```

Checking Unique value for country name column

In [56]:
```
print(df['Country Name'].unique())
print('\n')
print(f"Total number of unique countries are : {df['Country Name'].nunique()}")
```

```
['Aruba' 'Africa Eastern and Southern' 'Afghanistan'
 'Africa Western and Central' 'Angola' 'Albania' 'Andorra' 'Arab World'
 'United Arab Emirates' 'Argentina' 'Armenia' 'American Samoa'
 'Antigua and Barbuda' 'Australia' 'Austria' 'Azerbaijan' 'Burundi'
 'Belgium' 'Benin' 'Burkina Faso' 'Bangladesh' 'Bulgaria' 'Bahrain'
 'Bahamas, The' 'Bosnia and Herzegovina' 'Belarus' 'Belize' 'Bermuda'
 'Bolivia' 'Brazil' 'Barbados' 'Brunei Darussalam' 'Bhutan' 'Botswana'
 'Central African Republic' 'Canada' 'Central Europe and the Baltics'
 'Switzerland' 'Channel Islands' 'Chile' 'China' "Cote d'Ivoire"
 'Cameroon' 'Congo, Dem. Rep.' 'Congo, Rep.' 'Colombia' 'Comoros'
 'Cabo Verde' 'Costa Rica' 'Caribbean small states' 'Cuba' 'Curacao'
 'Cayman Islands' 'Cyprus' 'Czechia' 'Germany' 'Djibouti' 'Dominica'
 'Denmark' 'Dominican Republic' 'Algeria'
 'East Asia & Pacific (excluding high income)'
 'Early-demographic dividend' 'East Asia & Pacific'
 'Europe & Central Asia (excluding high income)' 'Europe & Central Asia'
 'Ecuador' 'Egypt, Arab Rep.' 'Euro area' 'Eritrea' 'Spain' 'Estonia'
 'Ethiopia' 'European Union' 'Fragile and conflict affected situations'
 'Finland' 'Fiji' 'France' 'Faroe Islands' 'Micronesia, Fed. Sts.' 'Gabon'
 'United Kingdom' 'Georgia' 'Ghana' 'Gibraltar' 'Guinea' 'Gambia, The'
 'Guinea-Bissau' 'Equatorial Guinea' 'Greece' 'Grenada' 'Greenland'
 'Guatemala' 'Guam' 'Guyana' 'High income' 'Hong Kong SAR, China'
 'Honduras' 'Heavily indebted poor countries (HIPC)' 'Croatia' 'Haiti'
 'Hungary' 'IBRD only' 'IDA & IBRD total' 'IDA total' 'IDA blend'
 'Indonesia' 'IDA only' 'Isle of Man' 'India' 'Ireland'
 'Iran, Islamic Rep.' 'Iraq' 'Iceland' 'Israel' 'Italy' 'Jamaica' 'Jordan'
 'Japan' 'Kazakhstan' 'Kenya' 'Kyrgyz Republic' 'Cambodia' 'Kiribati'
 'St. Kitts and Nevis' 'Korea, Rep.' 'Kuwait'
 'Latin America & Caribbean (excluding high income)' 'Lao PDR' 'Lebanon'
 'Liberia' 'Libya' 'St. Lucia' 'Latin America & Caribbean'
 'Least developed countries: UN classification' 'Low income'
 'Liechtenstein' 'Sri Lanka' 'Lower middle income' 'Low & middle income'
 'Lesotho' 'Late-demographic dividend' 'Lithuania' 'Luxembourg' 'Latvia'
 'Macao SAR, China' 'St. Martin (French part)' 'Morocco' 'Monaco'
 'Moldova' 'Madagascar' 'Maldives' 'Middle East & North Africa' 'Mexico'
 'Marshall Islands' 'Middle income' 'North Macedonia' 'Mali' 'Malta'
 'Myanmar' 'Middle East & North Africa (excluding high income)'
 'Montenegro' 'Mongolia' 'Northern Mariana Islands' 'Mozambique'
 'Mauritania' 'Mauritius' 'Malawi' 'Malaysia' 'North America' 'Namibia'
 'New Caledonia' 'Niger' 'Nigeria' 'Nicaragua' 'Netherlands' 'Norway'
 'Nepal' 'Nauru' 'New Zealand' 'OECD members' 'Oman' 'Other small states'
 'Pakistan' 'Panama' 'Peru' 'Philippines' 'Palau' 'Papua New Guinea'
 'Poland' 'Pre-demographic dividend' 'Puerto Rico'
 "Korea, Dem. People's Rep." 'Portugal' 'Paraguay'
 'Pacific island small states' 'Post-demographic dividend'
 'French Polynesia' 'Qatar' 'Romania' 'Russian Federation' 'Rwanda'
 'South Asia' 'Saudi Arabia' 'Sudan' 'Senegal' 'Singapore'
 'Solomon Islands' 'Sierra Leone' 'El Salvador' 'San Marino' 'Somalia'
 'Serbia' 'Sub-Saharan Africa (excluding high income)' 'South Sudan'
 'Sub-Saharan Africa' 'Small states' 'Sao Tome and Principe' 'Suriname'
 'Slovak Republic' 'Slovenia' 'Sweden' 'Eswatini'
 'Sint Maarten (Dutch part)' 'Seychelles' 'Syrian Arab Republic'
 'Turks and Caicos Islands' 'Chad'
 'East Asia & Pacific (IDA & IBRD countries)'
 'Europe & Central Asia (IDA & IBRD countries)' 'Togo' 'Thailand'
 'Tajikistan' 'Turkmenistan'
 'Latin America & the Caribbean (IDA & IBRD countries)' 'Timor-Leste'
 'Middle East & North Africa (IDA & IBRD countries)' 'Tonga'
 'South Asia (IDA & IBRD)' 'Sub-Saharan Africa (IDA & IBRD countries)'
 'Trinidad and Tobago' 'Tunisia' 'Turkiye' 'Tuvalu' 'Tanzania' 'Uganda'
 'Ukraine' 'Upper middle income' 'Uruguay' 'United States' 'Uzbekistan'
 'St. Vincent and the Grenadines' 'Venezuela, RB' 'British Virgin Islands'
 'Virgin Islands (U.S.)' 'Viet Nam' 'Vanuatu' 'World' 'Samoa' 'Kosovo'
 'Yemen, Rep.' 'South Africa' 'Zambia' 'Zimbabwe']
```

```
Total number of unique countries are : 264
```

In [ ]:

In [57]:
```
print(df['Country Code'].unique())
print('\n')
print(f"Total number of unique countries code are : {df['Country Code'].nunique()}"
```

```
['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' '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' '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']


Total number of unique countries code are : 264
```

Column Selection:

- Dropping unnecessary columns like Proposed_Remedy,Risky_Behaviors_Engaged because it's not essential for analysis.

In [58]:
```
# Drop Indicator Name column
df = df.drop(['Indicator Name'], axis=1)
```

In [59]:
```
df.shape
```

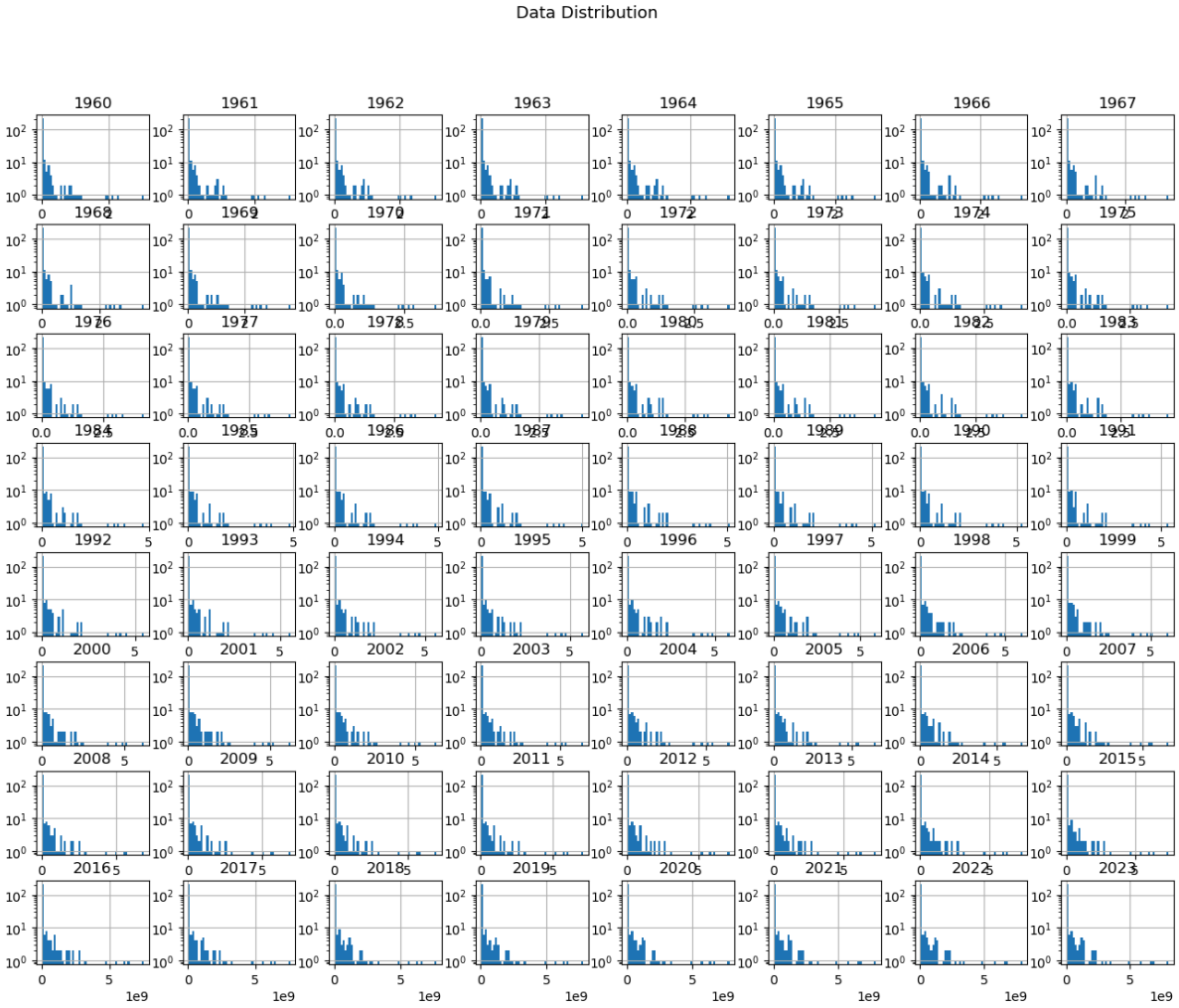
Out[59]:
```
(264, 67)
```

Exploratory Data Analysis (EDA)

Check Data Distribution

- by Histograms: To Visualize Skewness of Each Column.

In [60]:
```
# check data distribution
df.hist(figsize=(16, 12), bins=50, log=True)
plt.suptitle('Data Distribution', fontsize=13)
plt.show()
```



skewness calculation only on the numeric columns, avoiding the inclusion of non-numeric columns.

In [61]:
```
# Select only numeric columns
numeric_df = df.select_dtypes(include=['number'])

# Calculate skewness for numeric columns
skewness = numeric_df.skew()

# Print the skewness
print(skewness)
```

```
1960    4.890496
1961    4.897392
1962    4.901252
1963    4.902471
1964    4.903891
          ...
2019    4.966489
2020    4.965437
2021    4.963988
2022    4.962397
2023    4.960944
Length: 64, dtype: float64
```

In [ ]:

Extracting Top 10 Most Populated Countries in 2023

In [62]:
```
# filter data for total population
total_population_data = df[df['Indicator Code']=='SP.POP.TOTL']
print(total_population_data)
```

```
                    Country Name Country Code Indicator Code       1960  \
0                          Aruba          ABW    SP.POP.TOTL      54922
1    Africa Eastern and Southern          AFE    SP.POP.TOTL  130072080
2                    Afghanistan          AFG    SP.POP.TOTL    9035043
3     Africa Western and Central          AFW    SP.POP.TOTL   97630925
4                         Angola          AGO    SP.POP.TOTL    5231654
..                           ...          ...            ...        ...
259                       Kosovo          XKX    SP.POP.TOTL     984846
260                  Yemen, Rep.          YEM    SP.POP.TOTL    5532301
261                 South Africa          ZAF    SP.POP.TOTL   16440172
262                       Zambia          ZMB    SP.POP.TOTL    3153729
263                     Zimbabwe          ZWE    SP.POP.TOTL    3809389

          1961       1962       1963       1964       1965       1966  ...  \
0        55578      56320      57002      57619      58190      58694  ...
1    133534923  137171659  140945536  144904094  149033472  153281203  ...
2      9214083    9404406    9604487    9814318   10036008   10266395  ...
3     99706674  101854756  104089175  106388440  108772632  111246953  ...
4      5301583    5354310    5408320    5464187    5521981    5581386  ...
..         ...        ...        ...        ...        ...        ...  ...
259    1011421    1036950    1062737    1090270    1120168    1152586  ...
260    5655232    5782221    5911135    6048006    6195593    6351494  ...
261   16908035   17418522   17954564   18511361   19089380   19690087  ...
262    3254086    3358099    3465907    3577017    3692086    3812003  ...
263    3930401    4055959    4185877    4320006    4458462    4601217  ...

            2014         2015         2016         2017         2018  \
0       106807.0     107906.0     108727.0     108735.0     108908.0
1    590968990.0  607123269.0  623369401.0  640058741.0  657801085.0
2     32792523.0   33831764.0   34700612.0   35688935.0   36743039.0
3    406992047.0  418127845.0  429454743.0  440882906.0  452195915.0
4     27160769.0   28157798.0   29183070.0   30234839.0   31297155.0
..           ...          ...          ...          ...          ...
259    1812788.0    1788274.0    1777568.0    1791019.0    1797086.0
260   30226309.0   31159379.0   32109010.0   33090921.0   34085182.0
261   55594838.0   56723537.0   57259551.0   57635162.0   58613001.0
262   15895315.0   16399089.0   16914423.0   17441320.0   17973569.0
263   14207359.0   14399013.0   14600294.0   14812482.0   15034452.0

          2019         2020       2021         2022       2023
0       109203     108587.0     107700     107310.0     107359
1    675950189  694446100.0  713090928  731821393.0  750503764
2     37856121   39068979.0   40000412   40578842.0   41454761
3    463365429  474569351.0  485920997  497387180.0  509398589
4     32375632   33451132.0   34532429   35635029.0   36749906
..         ...          ...        ...          ...        ...
259    1788891    1790152.0    1786080    1768096.0    1756366
260   35111408   36134863.0   37140230   38222876.0   39390799
261   59587885   60562381.0   61502603   62378410.0   63212384
262   18513839   19059395.0   19603607   20152938.0   20723965
263   15271368   15526888.0   15797210   16069056.0   16340822

[264 rows x 67 columns]
```

In [63]:
```
# Sort the DataFrame by the '2023' column in descending order (to get the highest p

total_population_data_sorted = total_population_data.sort_values(by='2023', ascendi

# Display the sorted DataFrame
print(total_population_data_sorted)
```

```
                Country Name Country Code Indicator Code        1960  \
257                    World          WLD    SP.POP.TOTL  3021529236
103          IDA & IBRD total          IBT    SP.POP.TOTL  2289188361
139       Low & middle income          LMY    SP.POP.TOTL  2106419056
155             Middle income          MIC    SP.POP.TOTL  1970293325
102                 IBRD only          IBD    SP.POP.TOTL  1893700976
..                        ...          ...           ...         ...
210                San Marino          SMR    SP.POP.TOTL       15428
146  St. Martin (French part)          MAF    SP.POP.TOTL        4250
187                    Palau          PLW    SP.POP.TOTL        9328
178                    Nauru          NRU    SP.POP.TOTL        4607
243                   Tuvalu          TUV    SP.POP.TOTL        5598

           1961        1962        1963        1964        1965        1966  \
257  3062769479  3117373096  3184063049  3251253200  3318997522  3389087189
103  2321079191  2366117598  2423348992  2481068241  2539564344  2600930009
139  2136240778  2179211852  2234339068  2289931597  2346426203  2406483811
155  1996885454  2036499219  2088133586  2140061515  2192727365  2248823830
102  1915613080  1950274734  1996753344  2043309631  2090209567  2139608511
..          ...         ...         ...         ...         ...         ...
210       15799       16183       16580       16977       17305       17523
146        4386        4527        4673        4827        4996        5213
187        9555        9793       10037       10292       10547       10805
178        4774        4979        5226        5515        5832        6043
243        5648        5699        5751        5752        5715        5686

     ...          2014          2015          2016          2017  \
257  ...  7.353911e+09  7.441472e+09  7.528523e+09  7.614114e+09
103  ...  6.201872e+09  6.281303e+09  6.360571e+09  6.440497e+09
139  ...  5.971198e+09  6.050055e+09  6.128880e+09  6.208764e+09
155  ...  5.393612e+09  5.457879e+09  5.521013e+09  5.584207e+09
102  ...  4.630872e+09  4.677114e+09  4.721984e+09  4.766267e+09
..   ...           ...           ...           ...           ...
210  ...  3.265500e+04  3.289700e+04  3.310100e+04  3.382500e+04
146  ...  3.745000e+04  3.736900e+04  3.717500e+04  3.683700e+04
187  ...  1.771500e+04  1.777000e+04  1.779700e+04  1.781200e+04
178  ...  1.074200e+04  1.095400e+04  1.115000e+04  1.132400e+04
243  ...  1.098400e+04  1.096300e+04  1.093000e+04  1.086900e+04

           2018        2019          2020        2021          2022  \
257  7.696495e+09  7776892015  7.856139e+09  7921184346  7.989982e+09
103  6.518627e+09  6594857152  6.667541e+09  6733337540  6.795222e+09
139  6.287535e+09  6364581440  6.438170e+09  6505838578  6.568732e+09
155  5.645349e+09  5703863074  5.758609e+09  5807623372  5.851866e+09
102  4.807550e+09  4845715724  4.879367e+09  4905899020  4.928694e+09
..           ...         ...           ...         ...           ...
210  3.452200e+04       34663  3.477000e+04       34252  3.375500e+04
146  3.601200e+04       34267  3.178600e+04       29961  2.887000e+04
187  1.781400e+04       17798  1.779200e+04       17783  1.775900e+04
178  1.147700e+04       11587  1.164300e+04       11709  1.180100e+04
243  1.075100e+04       10581  1.039900e+04       10194  9.992000e+03

          2023
257  8061876001
103  6858957145
139  6633109634
155  5896643239
102  4952574126
..          ...
210       33860
146       27515
187       17727
178       11875
243        9816
```

```
[264 rows x 67 columns]
```

In [64]:
```
top_10_country =total_population_data_sorted.head(10)
top_10_country
```

Out[64]:

	Country Name	Country Code	Indicator Code	1960	1961	1962	1963	196
257	World	WLD	SP.POP.TOTL	3021529236	3062769479	3117373096	3184063049	325125320
103	IDA & IBRD total	IBT	SP.POP.TOTL	2289188361	2321079191	2366117598	2423348992	248106824
139	Low & middle income	LMY	SP.POP.TOTL	2106419056	2136240778	2179211852	2234339068	228993159
155	Middle income	MIC	SP.POP.TOTL	1970293325	1996885454	2036499219	2088133586	214006151
102	IBRD only	IBD	SP.POP.TOTL	1893700976	1915613080	1950274734	1996753344	204330963
62	Early-demographic dividend	EAR	SP.POP.TOTL	968909356	994176841	1020130725	1046697479	107402128
138	Lower middle income	LMC	SP.POP.TOTL	819047154	839649083	860740314	882371971	90452878
247	Upper middle income	UMC	SP.POP.TOTL	1151246171	1157236371	1175758905	1205761615	123553272
63	East Asia & Pacific	EAS	SP.POP.TOTL	1042838451	1044749888	1059216594	1085095424	111065117
141	Late-demographic dividend	LTE	SP.POP.TOTL	1094880069	1096959453	1111293141	1137056579	116240529

10 rows × 67 columns

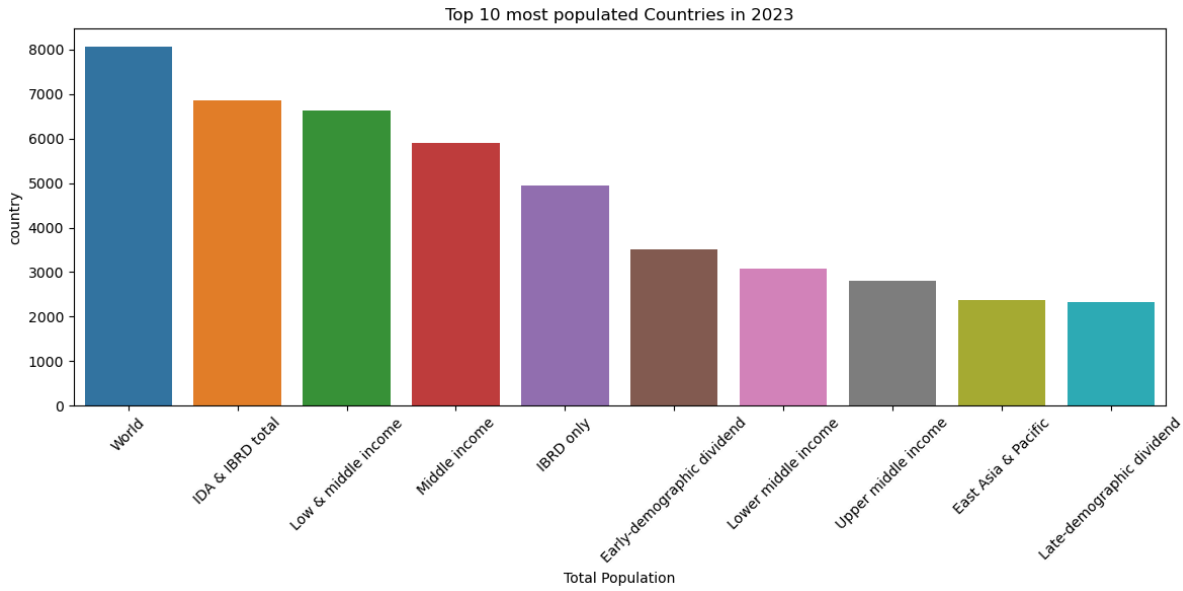
In [65]:
```
print("Top 10 countries in total population")
print(top_10_country[['Country Code','Country Name','2023']])
```

```
Top 10 countries in total population
    Country Code                Country Name        2023
257          WLD                       World  8061876001
103          IBT           IDA & IBRD total  6858957145
139          LMY        Low & middle income  6633109634
155          MIC              Middle income  5896643239
102          IBD                  IBRD only  4952574126
62           EAR  Early-demographic dividend  3511076811
138          LMC        Lower middle income  3079778780
247          UMC        Upper middle income  2816864459
63           EAS        East Asia & Pacific  2384463611
141          LTE  Late-demographic dividend  2326658047
```

Visualizations

Bar Chart: Top 10 Most Populated Countries in 2023

In [76]:
```
# Create a Bar chart
plt.figure(figsize=(12,6))
sns.barplot(x= top_10_country['Country Name'], y= top_10_country['2023']/ 1e6)
plt.title('Top 10 most populated Countries in 2023')
plt.xlabel('Total Population')
plt.ylabel('country')
plt.xticks(rotation=45)

# Adjust layout to prevent label cut-off
plt.tight_layout()
plt.show()
```



Extracting bottom 10 countries based on population

In [67]:
```
# Sort the DataFrame to get the bottom 10 contries  populations first)

total_population_data_sorted_bottom = total_population_data.sort_values(by='2023',

# Display the sorted DataFrame
print(total_population_data_sorted_bottom)
```

```
                Country Name Country Code Indicator Code        1960  \
243                   Tuvalu          TUV    SP.POP.TOTL        5598
178                    Nauru          NRU    SP.POP.TOTL        4607
187                    Palau          PLW    SP.POP.TOTL        9328
146  St. Martin (French part)          MAF    SP.POP.TOTL        4250
210                San Marino          SMR    SP.POP.TOTL       15428
..                        ...          ...            ...         ...
102                 IBRD only          IBD    SP.POP.TOTL  1893700976
155             Middle income          MIC    SP.POP.TOTL  1970293325
139       Low & middle income          LMY    SP.POP.TOTL  2106419056
103          IDA & IBRD total          IBT    SP.POP.TOTL  2289188361
257                     World          WLD    SP.POP.TOTL  3021529236

           1961        1962        1963        1964        1965        1966  \
243        5648        5699        5751        5752        5715        5686
178        4774        4979        5226        5515        5832        6043
187        9555        9793       10037       10292       10547       10805
146        4386        4527        4673        4827        4996        5213
210       15799       16183       16580       16977       17305       17523
..          ...         ...         ...         ...         ...         ...
102  1915613080  1950274734  1996753344  2043309631  2090209567  2139608511
155  1996885454  2036499219  2088133586  2140061515  2192727365  2248823830
139  2136240778  2179211852  2234339068  2289931597  2346426203  2406483811
103  2321079191  2366117598  2423348992  2481068241  2539564344  2600930009
257  3062769479  3117373096  3184063049  3251253200  3318997522  3389087189

     ...          2014          2015          2016          2017  \
243  ...  1.098400e+04  1.096300e+04  1.093000e+04  1.086900e+04
178  ...  1.074200e+04  1.095400e+04  1.115000e+04  1.132400e+04
187  ...  1.771500e+04  1.777000e+04  1.779700e+04  1.781200e+04
146  ...  3.745000e+04  3.736900e+04  3.717500e+04  3.683700e+04
210  ...  3.265500e+04  3.289700e+04  3.310100e+04  3.382500e+04
..   ...           ...           ...           ...           ...
102  ...  4.630872e+09  4.677114e+09  4.721984e+09  4.766267e+09
155  ...  5.393612e+09  5.457879e+09  5.521013e+09  5.584207e+09
139  ...  5.971198e+09  6.050055e+09  6.128880e+09  6.208764e+09
103  ...  6.201872e+09  6.281303e+09  6.360571e+09  6.440497e+09
257  ...  7.353911e+09  7.441472e+09  7.528523e+09  7.614114e+09

              2018        2019          2020        2021          2022  \
243  1.075100e+04       10581  1.039900e+04       10194  9.992000e+03
178  1.147700e+04       11587  1.164300e+04       11709  1.180100e+04
187  1.781400e+04       17798  1.779200e+04       17783  1.775900e+04
146  3.601200e+04       34267  3.178600e+04       29961  2.887000e+04
210  3.452200e+04       34663  3.477000e+04       34252  3.375500e+04
..            ...         ...           ...         ...           ...
102  4.807550e+09  4845715724  4.879367e+09  4905899020  4.928694e+09
155  5.645349e+09  5703863074  5.758609e+09  5807623372  5.851866e+09
139  6.287535e+09  6364581440  6.438170e+09  6505838578  6.568732e+09
103  6.518627e+09  6594857152  6.667541e+09  6733337540  6.795222e+09
257  7.696495e+09  7776892015  7.856139e+09  7921184346  7.989982e+09

           2023
243        9816
178       11875
187       17727
146       27515
210       33860
..          ...
102  4952574126
155  5896643239
139  6633109634
103  6858957145
257  8061876001
```

[264 rows x 67 columns]

In [68]:
```
bottom_10_country =total_population_data_sorted_bottom.head(10)
bottom_10_country
```

Out[68]:

	Country Name	Country Code	Indicator Code	1960	1961	1962	1963	1964	1965	1966	...	201
243	Tuvalu	TUV	SP.POP.TOTL	5598	5648	5699	5751	5752	5715	5686	...	10984
178	Nauru	NRU	SP.POP.TOTL	4607	4774	4979	5226	5515	5832	6043	...	10742
187	Palau	PLW	SP.POP.TOTL	9328	9555	9793	10037	10292	10547	10805	...	17715
146	St. Martin (French part)	MAF	SP.POP.TOTL	4250	4386	4527	4673	4827	4996	5213	...	37450
210	San Marino	SMR	SP.POP.TOTL	15428	15799	16183	16580	16977	17305	17523	...	32655
84	Gibraltar	GIB	SP.POP.TOTL	21839	21872	22197	22753	23315	23878	24444	...	32813
154	Marshall Islands	MHL	SP.POP.TOTL	14703	15219	15755	16302	16852	17402	17927	...	49796
148	Monaco	MCO	SP.POP.TOTL	21808	21901	22078	22385	22686	22941	23131	...	36128
253	British Virgin Islands	VGB	SP.POP.TOTL	7950	8034	8070	8114	8168	8257	8413	...	32962
136	Liechtenstein	LIE	SP.POP.TOTL	16451	16877	17443	18093	18755	19195	19610	...	37248

10 rows × 67 columns

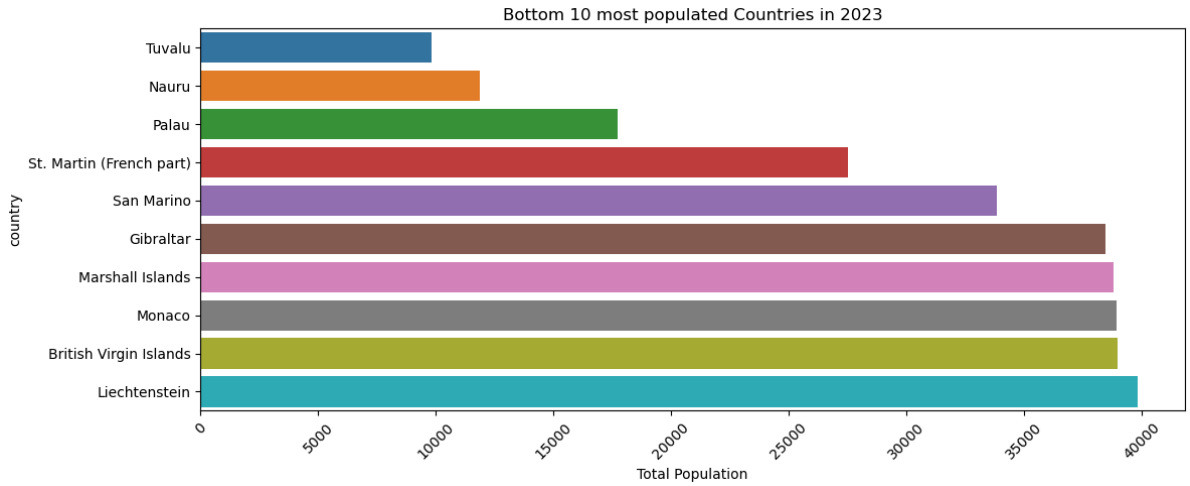
In [69]:
```
print("Bottom 10 countries in total population")
print(bottom_10_country[['Country Code','Country Name','2023']])
```

```
Bottom 10 countries in total population
    Country Code              Country Name   2023
243          TUV                    Tuvalu   9816
178          NRU                     Nauru  11875
187          PLW                     Palau  17727
146          MAF  St. Martin (French part)  27515
210          SMR                San Marino  33860
84           GIB                 Gibraltar  38471
154          MHL          Marshall Islands  38827
148          MCO                    Monaco  38956
253          VGB    British Virgin Islands  38985
136          LIE             Liechtenstein  39850
```

Visualizations

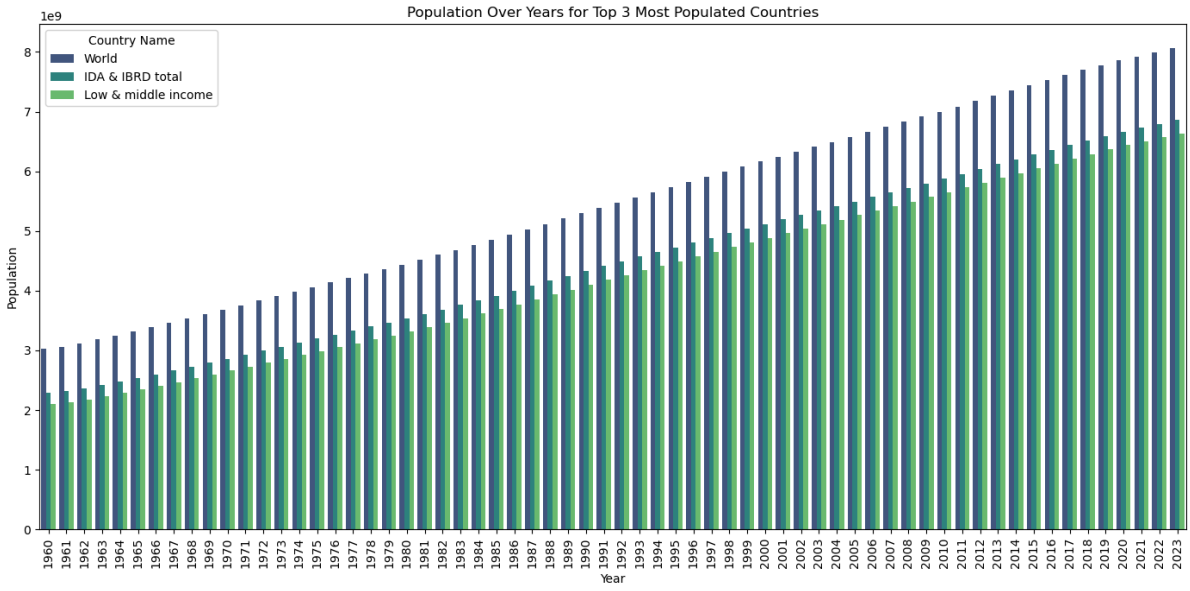
Bar Plot Bottom 10 most populated Countries in 2023

In [70]:
```
plt.figure(figsize=(12,5))
sns.barplot(x= bottom_10_country['2023'], y= bottom_10_country['Country Name'])
plt.title('Bottom 10 most populated Countries in 2023')
plt.xlabel('Total Population')
plt.ylabel('country')
plt.xticks(rotation=45)

# Adjust layout to prevent label cut-off
plt.tight_layout()
plt.show()
```



Grouped Bar Chart: Population Over Years for Top 3 Countries

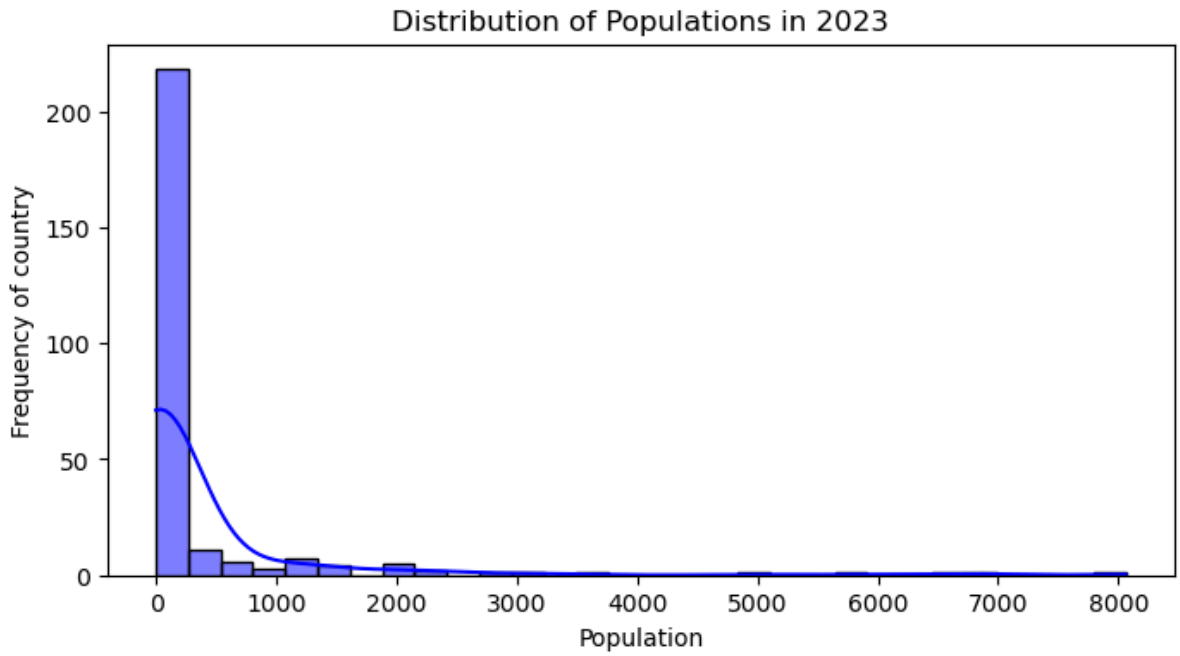
In [71]:
```
# Get the top 3 most populated countries
top_3_countries = total_population_data.sort_values(by='2023', ascending=False).hea

# Melt the DataFrame to have years as a variable
melted_df = pd.melt(top_3_countries, id_vars=['Country Name'], value_vars=[str(year

# Convert year to integer
melted_df['Year'] = melted_df['Year'].astype(int)

# Create the grouped bar chart
plt.figure(figsize=(14, 7))
sns.barplot(x='Year', y='Population', hue='Country Name', data=melted_df, palette='
plt.title('Population Over Years for Top 3 Most Populated Countries')
plt.xlabel('Year')
plt.ylabel('Population')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



Histogram: Distribution of Populations in 2023

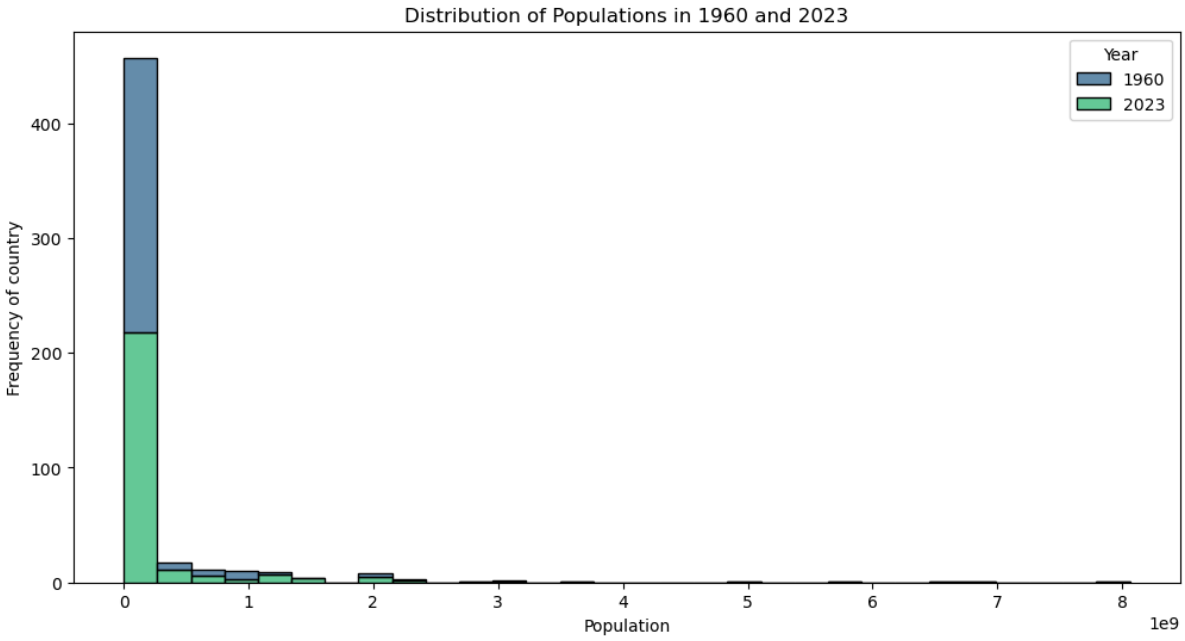
- A histogram can be used to show the distribution of populations for a specific year.

In [72]:
```
# Filter for total population in 2023
population_2023 = (total_population_data['2023']/ 1e6)

# Create the histogram
plt.figure(figsize=(8, 4))
sns.histplot(population_2023, bins=30, kde=True, color='b')
plt.title('Distribution of Populations in 2023')
plt.xlabel('Population')
plt.ylabel('Frequency of country')
plt.show()
```



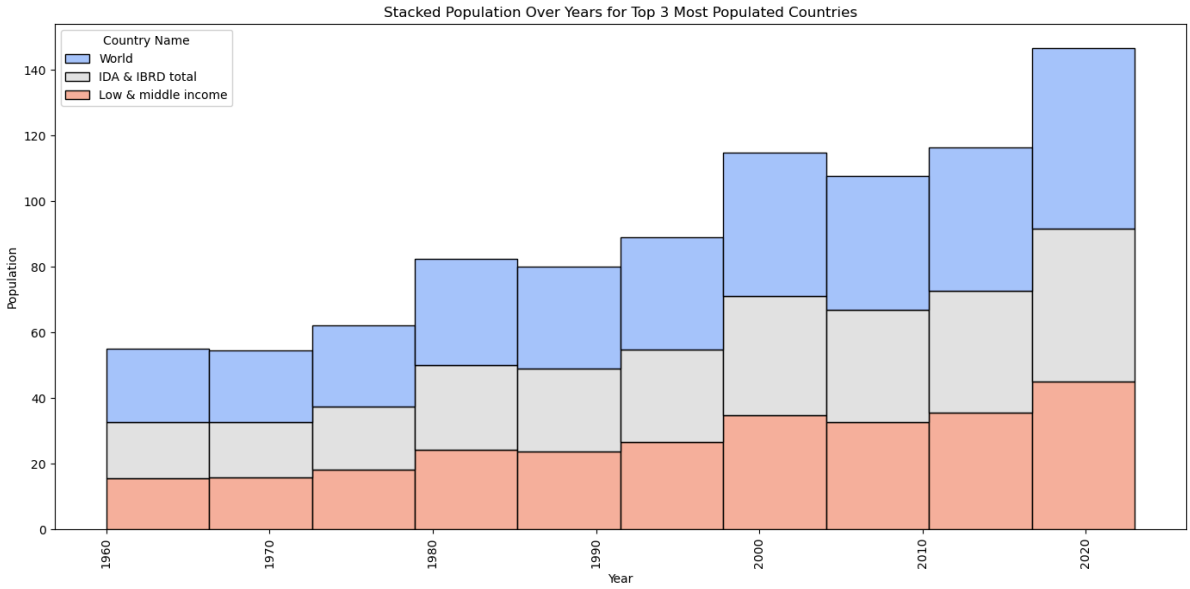
Histogram: Comparison of Population Distributions in 1960 and 2023

In [73]:
```
# Filter for total populations in 1960 and 2023
population_1960 = total_population_data['1960'].dropna()
population_2023 = total_population_data['2023'].dropna()

# Combine into a DataFrame
population_dist = pd.DataFrame({'1960': population_1960, '2023': population_2023})

# Melt the DataFrame for plotting
melted_dist = pd.melt(population_dist, value_vars=['1960', '2023'], var_name='Year'

# Create the grouped histogram
plt.figure(figsize=(12, 6))
sns.histplot(data=melted_dist, x='Population', hue='Year', multiple='stack', bins=3
plt.title('Distribution of Populations in 1960 and 2023')
plt.xlabel('Population')
plt.ylabel('Frequency of country')
plt.show()
```



Stacked Bar Chart: Population Over Years for Top 3 Countries

In [74]:
```
# Create a stacked bar chart
plt.figure(figsize=(14, 7))
sns.histplot(data=melted_df,bins= 10, x=melted_df['Year'], weights=melted_df['Popul
plt.title('Stacked Population Over Years for Top 3 Most Populated Countries')
plt.xlabel('Year')
plt.ylabel('Population')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



Summary Table

In [77]:
```
# Select relevant years
years = ['1960', '1980', '2000', '2023']
summary_data = total_population_data[['Country Name'] + years]

# Calculate summary statistics
summary_table = summary_data.describe(include='all').transpose()

# Add custom statistics
summary_table['Total Population'] = summary_data[years].sum()
summary_table['Mean Population'] = summary_data[years].mean()
summary_table['Min Population'] = summary_data[years].min()
summary_table['Max Population'] = summary_data[years].max()

# Display the summary table
print(summary_table)
```

```
              count unique    top freq              mean                std  \
Country Name    264    264  Aruba    1               NaN                NaN
1960          264.0    NaN    NaN  NaN  115448231.041667  362652351.853549
1980          264.0    NaN    NaN  NaN  173230062.314394  547550195.969665
2000          264.0    NaN    NaN  NaN  245889871.051136  775677300.984536
2023          264.0    NaN    NaN  NaN  330015290.492424 1020328994.859521

                 min         25%         50%          75%           max  \
Country Name     NaN         NaN         NaN          NaN           NaN
1960          2715.0   515202.75   3659633.0   26862930.0  3021529236.0
1980          7366.0    815569.0   5743968.5  38553412.25  4437602155.0
2000          9544.0  1265364.25   8258541.0   52116108.5  6161884826.0
2023          9816.0   1809953.5  10754446.5  67035076.25  8061876001.0

              Total Population  Mean Population  Min Population  \
Country Name               NaN             NaN             NaN
1960              3.047833e+10    1.154482e+08          2715.0
1980              4.573274e+10    1.732301e+08          7366.0
2000              6.491493e+10    2.458899e+08          9544.0
2023              8.712404e+10    3.300153e+08          9816.0

              Max Population
Country Name             NaN
1960            3.021529e+09
1980            4.437602e+09
2000            6.161885e+09
2023            8.061876e+09
```

Conclusion

The Global Population Trends Data Visualization project has provided a comprehensive analysis of population data from 1960 to 2023, offering valuable insights into demographic changes across different countries. Through data cleaning, descriptive statistics, and a variety of visualizations, we have uncovered key patterns and trends in the global population landscape.

Key takeaways from this project include:

1. **Identification of Demographic Patterns**:
    - We identified the most and least populated countries and observed significant growth trends in specific regions.

- The data revealed how population sizes have evolved over the decades, highlighting periods of rapid growth and stability.

2. **Insightful Visualizations**:
   - Bar charts and line plots effectively showcased the population sizes and trends for top countries.
   - Histograms provided a clear view of population distributions, emphasizing shifts and changes over time.
   - Grouped and stacked bar charts offered a comparative perspective, making it easier to analyze and understand complex data.
3. **Business and Policy Implications**:
   - The insights gained can assist urban planners, healthcare providers, market researchers, and policymakers in making informed decisions.
   - By understanding population dynamics, stakeholders can develop strategies that address the needs of growing populations and promote sustainable development.
4. **Data-Driven Decision Making**:
   - The project emphasized the importance of using data-driven approaches to analyze and interpret population data.
   - Visualizations not only made the data more accessible but also highlighted trends that might have been overlooked in raw data.

Prepared by [Mehul Chafekar]