eeyfhn6l7

December 1, 2023

TASK-01

- Create a bar chart or histogram to visualize the distribution of a categorical or continuous variable, such as the distribution of ages or genders in a population.
- Dataset :- https://data.worldbank.org/indicator/SP.POP.TOTL

1 Import necessary libraries

[109]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

[110]:
```
# Load the dataset
df=pd.read_csv('Population.csv')
```

[111]:
```
df
```

[111]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963
0	Aruba	ABW	Population, total	SP.POP.TOTL	54608.0	55811.0	56682.0	57475.0
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	130692579.0	134169237.0	137835590.0	141630546.0
2	Afghanistan	AFG	Population, total	SP.POP.TOTL	8622466.0	8790140.0	8969047.0	9157465.0
3	Africa Western and Central	AFW	Population, total	SP.POP.TOTL	97256290.0	99314028.0	101445032.0	103667517.0
4	Angola	AGO	Population, total	SP.POP.TOTL	5357195.0	5441333.0	5521400.0	5599827.0
..	…	…	…	…	…	…	…	…
261	Kosovo	XKX	Population, total					
262	Yemen, Rep.	YEM	Population, total					
263	South Africa	ZAF	Population, total					
264	Zambia	ZMB	Population, total					
265	Zimbabwe	ZWE	Population, total					

```
261    SP.POP.TOTL     947000.0     966000.0     994000.0    1022000.0
262    SP.POP.TOTL    5542459.0    5646668.0    5753386.0    5860197.0
263    SP.POP.TOTL   16520441.0   16989464.0   17503133.0   18042215.0
264    SP.POP.TOTL    3119430.0    3219451.0    3323427.0    3431381.0
265    SP.POP.TOTL    3806310.0    3925952.0    4049778.0    4177931.0

            1964         1965  …         2013         2014         2015  \
0        58178.0      58782.0  …     102880.0     103594.0     104257.0
1    145605995.0  149742351.0  …  567892149.0  583651101.0  600008424.0
2      9355514.0    9565147.0  …   31541209.0   32716210.0   33753499.0
3    105959979.0  108336203.0  …  387204553.0  397855507.0  408690375.0
4      5673199.0    5736582.0  …   26147002.0   27128337.0   28127721.0
..           …            …  …          …            …            …
261    1050000.0    1078000.0  …    1818117.0    1812771.0    1788196.0
262    5973803.0    6097298.0  …   26984002.0   27753304.0   28516545.0
263   18603097.0   19187194.0  …   53873616.0   54729551.0   55876504.0
264    3542764.0    3658024.0  …   15234976.0   15737793.0   16248230.0
265    4310332.0    4447149.0  …   13555422.0   13855753.0   14154937.0

            2016         2017         2018         2019         2020  \
0       104874.0     105439.0     105962.0     106442.0     106585.0
1    616377605.0  632746570.0  649757148.0  667242986.0  685112979.0
2     34636207.0   35643418.0   36686784.0   37769499.0   38972230.0
3    419778384.0  431138704.0  442646825.0  454306063.0  466189102.0
4     29154746.0   30208628.0   31273533.0   32353588.0   33428486.0
..           …            …            …            …            …
261    1777557.0    1791003.0    1797085.0    1788878.0    1790133.0
262   29274002.0   30034389.0   30790513.0   31546691.0   32284046.0
263   56422274.0   56641209.0   57339635.0   58087055.0   58801927.0
264   16767761.0   17298054.0   17835893.0   18380477.0   18927715.0
265   14452704.0   14751101.0   15052184.0   15354608.0   15669666.0

            2021         2022
0       106537.0     106445.0
1    702977106.0  720839314.0
2     40099462.0   41128771.0
3    478185907.0  490330870.0
4     34503774.0   35588987.0
..           …            …
261    1786038.0    1761985.0
262   32981641.0   33696614.0
263   59392255.0   59893885.0
264   19473125.0   20017675.0
265   15993524.0   16320537.0

[266 rows x 67 columns]
```

#Exploratory data analysis(EDA)

[112]:
```
# Method returns a copy of the DataFrame
df2=df.copy()
```

[113]:
```
#shape of a DataFrame.
df.shape
```

[113]:
```
(266, 67)
```

[114]:
```
# Display all columns
df.columns
```

[114]:
```
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')
```

[115]:
```
# displays the top rows of a DataFrame
df.head()
```

[115]:

	Country Name	Country Code	Indicator Name	Indicator Code	\
0	Aruba	ABW	Population, total	SP.POP.TOTL	
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	
2	Afghanistan	AFG	Population, total	SP.POP.TOTL	
3	Africa Western and Central	AFW	Population, total	SP.POP.TOTL	
4	Angola	AGO	Population, total	SP.POP.TOTL	

	1960	1961	1962	1963	1964	\
0	54608.0	55811.0	56682.0	57475.0	58178.0	
1	130692579.0	134169237.0	137835590.0	141630546.0	145605995.0	
2	8622466.0	8790140.0	8969047.0	9157465.0	9355514.0	
3	97256290.0	99314028.0	101445032.0	103667517.0	105959979.0	
4	5357195.0	5441333.0	5521400.0	5599827.0	5673199.0	

	1965	…	2013	2014	2015	2016	\
0	58782.0	…	102880.0	103594.0	104257.0	104874.0	
1	149742351.0	…	567892149.0	583651101.0	600008424.0	616377605.0	
2	9565147.0	…	31541209.0	32716210.0	33753499.0	34636207.0	
3	108336203.0	…	387204553.0	397855507.0	408690375.0	419778384.0	
4	5736582.0	…	26147002.0	27128337.0	28127721.0	29154746.0	

```
          2017         2018         2019         2020         2021  \
0     105439.0     105962.0     106442.0     106585.0     106537.0
1  632746570.0  649757148.0  667242986.0  685112979.0  702977106.0
2   35643418.0   36686784.0   37769499.0   38972230.0   40099462.0
3  431138704.0  442646825.0  454306063.0  466189102.0  478185907.0
4   30208628.0   31273533.0   32353588.0   33428486.0   34503774.0

          2022
0     106445.0
1  720839314.0
2   41128771.0
3  490330870.0
4   35588987.0

[5 rows x 67 columns]
```

[116]:
```
#shows the bottom rows
df.tail()
```

[116]:
```
     Country Name Country Code     Indicator Name Indicator Code        1960  \
261        Kosovo          XKX  Population, total    SP.POP.TOTL    947000.0
262   Yemen, Rep.          YEM  Population, total    SP.POP.TOTL   5542459.0
263  South Africa          ZAF  Population, total    SP.POP.TOTL  16520441.0
264        Zambia          ZMB  Population, total    SP.POP.TOTL   3119430.0
265      Zimbabwe          ZWE  Population, total    SP.POP.TOTL   3806310.0

           1961        1962        1963        1964        1965  …  \
261    966000.0    994000.0   1022000.0   1050000.0   1078000.0  …
262   5646668.0   5753386.0   5860197.0   5973803.0   6097298.0  …
263  16989464.0  17503133.0  18042215.0  18603097.0  19187194.0  …
264   3219451.0   3323427.0   3431381.0   3542764.0   3658024.0  …
265   3925952.0   4049778.0   4177931.0   4310332.0   4447149.0  …

           2013        2014        2015        2016        2017        2018  \
261   1818117.0   1812771.0   1788196.0   1777557.0   1791003.0   1797085.0
262  26984002.0  27753304.0  28516545.0  29274002.0  30034389.0  30790513.0
263  53873616.0  54729551.0  55876504.0  56422274.0  56641209.0  57339635.0
264  15234976.0  15737793.0  16248230.0  16767761.0  17298054.0  17835893.0
265  13555422.0  13855753.0  14154937.0  14452704.0  14751101.0  15052184.0

           2019        2020        2021        2022
261   1788878.0   1790133.0   1786038.0   1761985.0
262  31546691.0  32284046.0  32981641.0  33696614.0
263  58087055.0  58801927.0  59392255.0  59893885.0
264  18380477.0  18927715.0  19473125.0  20017675.0
265  15354608.0  15669666.0  15993524.0  16320537.0
```

[5 rows x 67 columns]

[117]:
```
# specific rows of a DataFrame ( "integer location" Method)
df.iloc[100:200]
```

[117]:

```
                    Country Name Country Code     Indicator Name  \
100                        Haiti          HTI  Population, total
101                      Hungary          HUN  Population, total
102                    IBRD only          IBD  Population, total
103            IDA & IBRD total          IBT  Population, total
104                    IDA total          IDA  Population, total
..                           …          …                …
195                     Paraguay          PRY  Population, total
196           West Bank and Gaza          PSE  Population, total
197  Pacific island small states          PSS  Population, total
198    Post-demographic dividend          PST  Population, total
199              French Polynesia          PYF  Population, total

    Indicator Code          1960          1961          1962          1963  \
100    SP.POP.TOTL  3.901139e+06  3.974934e+06  4.049504e+06  4.122260e+06
101    SP.POP.TOTL  9.983967e+06  1.002932e+07  1.006173e+07  1.008795e+07
102    SP.POP.TOTL  1.904347e+09  1.926043e+09  1.960606e+09  2.007061e+09
103    SP.POP.TOTL  2.297972e+09  2.329504e+09  2.374276e+09  2.431314e+09
104    SP.POP.TOTL  3.936256e+08  4.034613e+08  4.136700e+08  4.242533e+08
..             …             …             …             …             …
195    SP.POP.TOTL  1.894829e+06  1.941208e+06  1.989376e+06  2.039390e+06
196    SP.POP.TOTL           NaN           NaN           NaN           NaN
197    SP.POP.TOTL  9.055370e+05  9.325200e+05  9.602580e+05  9.887670e+05
198    SP.POP.TOTL  7.555751e+08  7.646789e+08  7.739717e+08  7.830631e+08
199    SP.POP.TOTL  8.485100e+04  8.692100e+04  8.920800e+04  9.196300e+04

             1964          1965  …          2013          2014  \
100  4.196349e+06  4.274348e+06  …  1.026121e+07  1.041274e+07
101  1.011984e+07  1.014794e+07  …  9.893082e+06  9.866468e+06
102  2.053555e+09  2.100537e+09  …  4.568406e+09  4.617515e+09
103  2.488809e+09  2.547220e+09  …  6.079930e+09  6.161220e+09
104  4.352541e+08  4.466833e+08  …  1.511524e+09  1.543705e+09
..            …             …  …             …             …
195  2.090840e+06  2.143153e+06  …  6.005652e+06  6.090721e+06
196           NaN           NaN  …  4.076708e+06  4.173398e+06
197  1.017629e+06  1.046929e+06  …  2.379069e+06  2.405308e+06
198  7.920609e+08  8.008346e+08  …  1.087231e+09  1.092180e+09
199  9.519200e+04  9.867400e+04  …  2.880320e+05  2.898730e+05

             2015          2016          2017          2018          2019  \
100  1.056376e+07  1.071385e+07  1.086354e+07  1.101242e+07  1.116044e+07
101  9.843028e+06  9.814023e+06  9.787966e+06  9.775564e+06  9.771141e+06
```

```
102  4.665081e+09  4.710746e+09  4.755029e+09  4.795958e+09  4.833831e+09
103  6.241659e+09  6.321547e+09  6.401430e+09  6.479098e+09  6.554881e+09
104  1.576578e+09  1.610801e+09  1.646401e+09  1.683140e+09  1.721049e+09
..            …             …             …             …             …
195  6.177950e+06  6.266615e+06  6.355404e+06  6.443328e+06  6.530026e+06
196  4.270092e+06  4.367088e+06  4.454805e+06  4.569087e+06  4.685306e+06
197  2.431426e+06  2.457814e+06  2.484263e+06  2.510226e+06  2.536070e+06
198  1.097061e+09  1.102020e+09  1.106215e+09  1.110127e+09  1.113311e+09
199  2.917870e+05  2.935410e+05  2.954500e+05  2.976060e+05  2.997170e+05

             2020          2021          2022
100  1.130680e+07  1.144757e+07  1.158500e+07
101  9.750149e+06  9.709891e+06  9.683505e+06
102  4.867842e+09  4.895307e+09  4.914355e+09
103  6.628068e+09  6.695340e+09  6.754327e+09
104  1.760226e+09  1.800033e+09  1.839971e+09
..            …             …             …
195  6.618695e+06  6.703799e+06  6.780744e+06
196  4.803269e+06  4.922749e+06  5.043612e+06
197  2.566819e+06  2.602173e+06  2.639019e+06
198  1.117424e+09  1.116545e+09  1.113419e+09
199  3.019200e+05  3.040320e+05  3.062790e+05

[100 rows x 67 columns]
```

[118]:
```
# prints information about the DataFrame.
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
```

[119]:
```
# Display summary statistics for numerical columns in  DataFrame.
df.describe().T
```

[119]:

	count	mean	std	min	25%	50%	\
1960	264.0	1.172712e+08	3.695439e+08	2646.0	513221.25	3757485.5	
1961	264.0	1.188807e+08	3.740897e+08	2888.0	523134.50	3887144.0	
1962	264.0	1.210511e+08	3.808061e+08	3171.0	533759.50	4023895.5	
1963	264.0	1.237333e+08	3.895039e+08	3481.0	544928.75	4139356.5	
1964	264.0	1.264378e+08	3.982439e+08	3811.0	556663.00	4224612.5	
…	…	…	…	…	…	…	
2018	265.0	3.120276e+08	9.746880e+08	10865.0	1797085.00	10395329.0	
2019	265.0	3.157110e+08	9.851690e+08	10956.0	1788878.00	10447666.0	
2020	265.0	3.192936e+08	9.952294e+08	11069.0	1790133.00	10606227.0	
2021	265.0	3.225180e+08	1.004211e+09	11204.0	1786038.00	10505772.0	
2022	265.0	3.254839e+08	1.012174e+09	11312.0	1761985.00	10526073.0	

	75%	max
1960	26706062.75	3.031474e+09
1961	27486939.00	3.072422e+09
1962	28302886.00	3.126850e+09
1963	29147077.00	3.193429e+09
1964	30016841.75	3.260442e+09
…	…	…
2018	60421760.00	7.661777e+09
2019	59872579.00	7.742682e+09
2020	61704518.00	7.820964e+09
2021	63588334.00	7.888161e+09
2022	65497748.00	7.951150e+09

[63 rows x 8 columns]

[120]:
```
# Dispaly (string) columns in the summary statistics.
df.describe(include=object)
```

[120]:

	Country Name	Country Code	Indicator Name	Indicator Code
count	266	266	266	266
unique	266	266	1	1
top	Aruba	ABW	Population, total	SP.POP.TOTL
freq	1	1	266	266

2 Data cleaning

[121]:
```
pd.set_option('display.max_rows', None)
```

[122]:
```
# To check for duplicate values in a  DataFrame
df.duplicated().sum()
```

[122]: 0

[123]:
```
df.isnull().sum().sort_values(ascending=False)
```

[123]:
```
1989    2
1973    2
1988    2
1987    2
1986    2
1985    2
1984    2
1983    2
1982    2
1981    2
1979    2
1978    2
1977    2
1976    2
1975    2
1974    2
1980    2
1972    2
1965    2
1971    2
1960    2
1961    2
1962    2
1964    2
1963    2
1966    2
1967    2
1968    2
1969    2
1970    2
2006    1
2013    1
2007    1
2008    1
2009    1
2010    1
```

```
2011              1
2012              1
2021              1
2014              1
2015              1
2016              1
2017              1
2018              1
2019              1
2020              1
2004              1
2005              1
2022              1
2003              1
1995              1
1990              1
1991              1
1992              1
1993              1
2002              1
1994              1
1996              1
1997              1
1998              1
1999              1
2000              1
2001              1
Country Code      0
Indicator Code    0
Indicator Name    0
Country Name      0
dtype: int64
```

[124]:
```
# Missing value percentage calculator
df.isnull().sum()/df.shape[0]*100
```

[124]:
```
Country Name      0.00000
Country Code      0.00000
Indicator Name    0.00000
Indicator Code    0.00000
1960              0.75188
1961              0.75188
1962              0.75188
1963              0.75188
1964              0.75188
1965              0.75188
1966              0.75188
```

1967	0.75188
1968	0.75188
1969	0.75188
1970	0.75188
1971	0.75188
1972	0.75188
1973	0.75188
1974	0.75188
1975	0.75188
1976	0.75188
1977	0.75188
1978	0.75188
1979	0.75188
1980	0.75188
1981	0.75188
1982	0.75188
1983	0.75188
1984	0.75188
1985	0.75188
1986	0.75188
1987	0.75188
1988	0.75188
1989	0.75188
1990	0.37594
1991	0.37594
1992	0.37594
1993	0.37594
1994	0.37594
1995	0.37594
1996	0.37594
1997	0.37594
1998	0.37594
1999	0.37594
2000	0.37594
2001	0.37594
2002	0.37594
2003	0.37594
2004	0.37594
2005	0.37594
2006	0.37594
2007	0.37594
2008	0.37594
2009	0.37594
2010	0.37594
2011	0.37594
2012	0.37594
2013	0.37594

```
2014              0.37594
2015              0.37594
2016              0.37594
2017              0.37594
2018              0.37594
2019              0.37594
2020              0.37594
2021              0.37594
2022              0.37594
dtype: float64
```

my domain dataset country population so missing values are very important so I am not using Dropna function[df=df.dropna()]. missing values handle filling method

[125]:
```
df.fillna(df.median(), inplace=True)
```

```
<ipython-input-125-e2cd313b306c>:1: FutureWarning: The default value of
numeric_only in DataFrame.median is deprecated. In a future version, it will
default to False. In addition, specifying 'numeric_only=None' is deprecated.
Select only valid columns or specify the value of numeric_only to silence this
warning.
  df.fillna(df.median(), inplace=True)
```

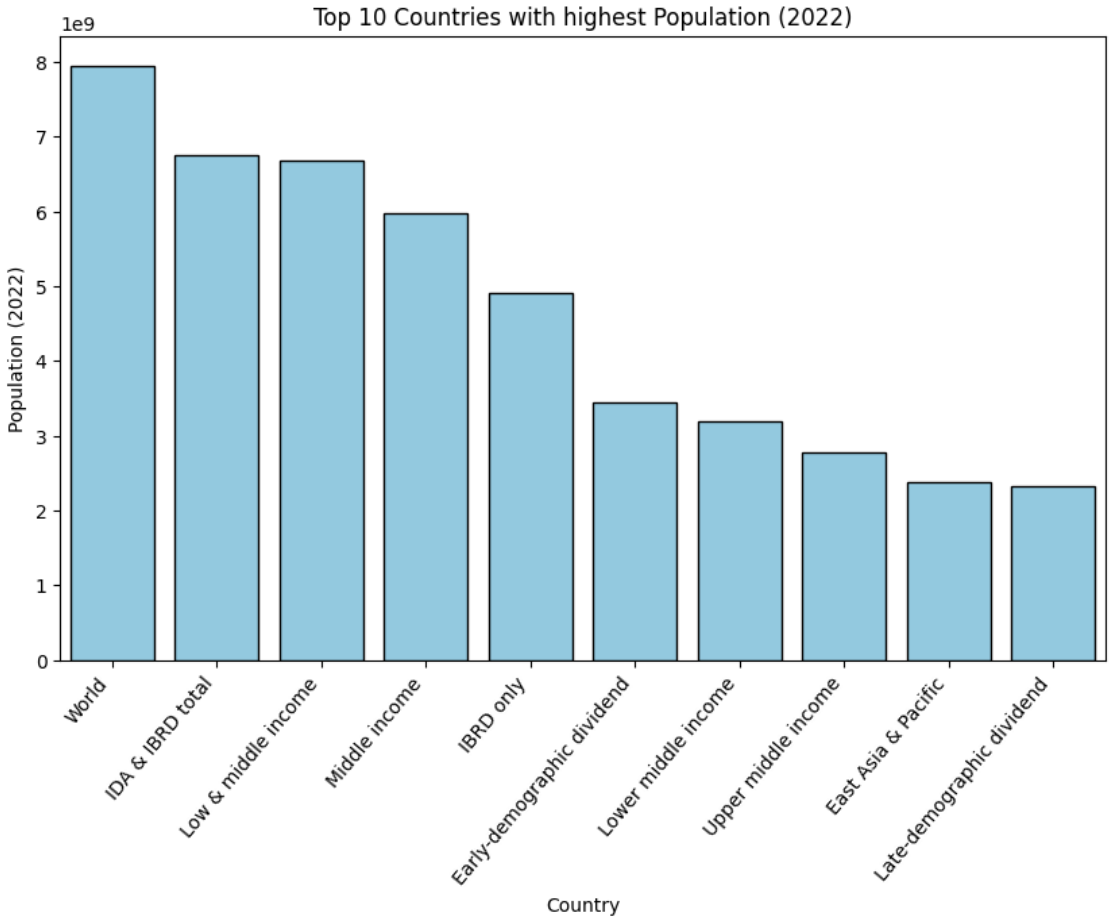
[126]:
```
df.isnull().sum().any()
```

[126]:
```
False
```

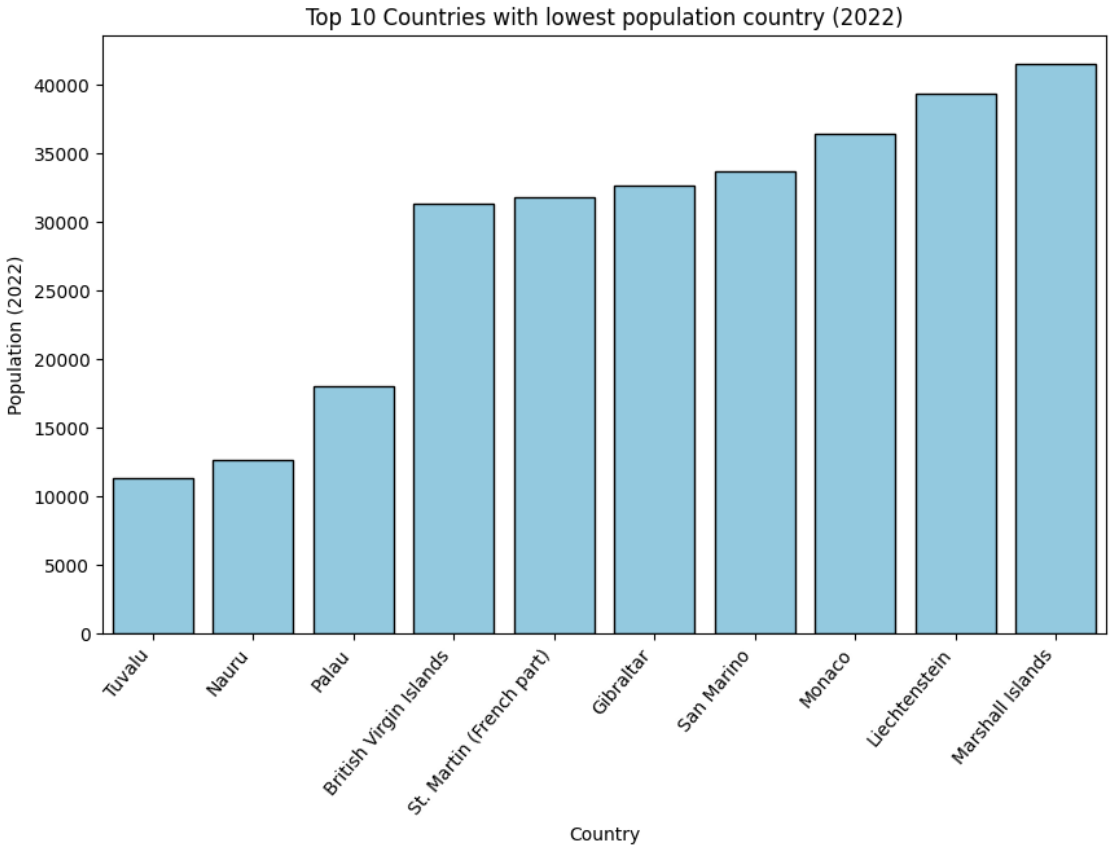
3 visualization - BOXPOLT

[127]:
```
# Visualization - Function to plot top/bottom countries
def plot_top_countries(df, column, ascending=True,color='Skyblue',title=''):
    df_sort = df.sort_values(column, ascending=ascending).head(10)
    plt.figure(figsize=(10, 6))
    sns.barplot(x=df_sort['Country Name'],y=df_sort[column], color=color,␣
 ↪edgecolor='black')
    plt.xlabel('Country')
    plt.ylabel(f'Population ({column})')
    plt.title(title)
    plt.xticks(rotation=50, ha='right')
```

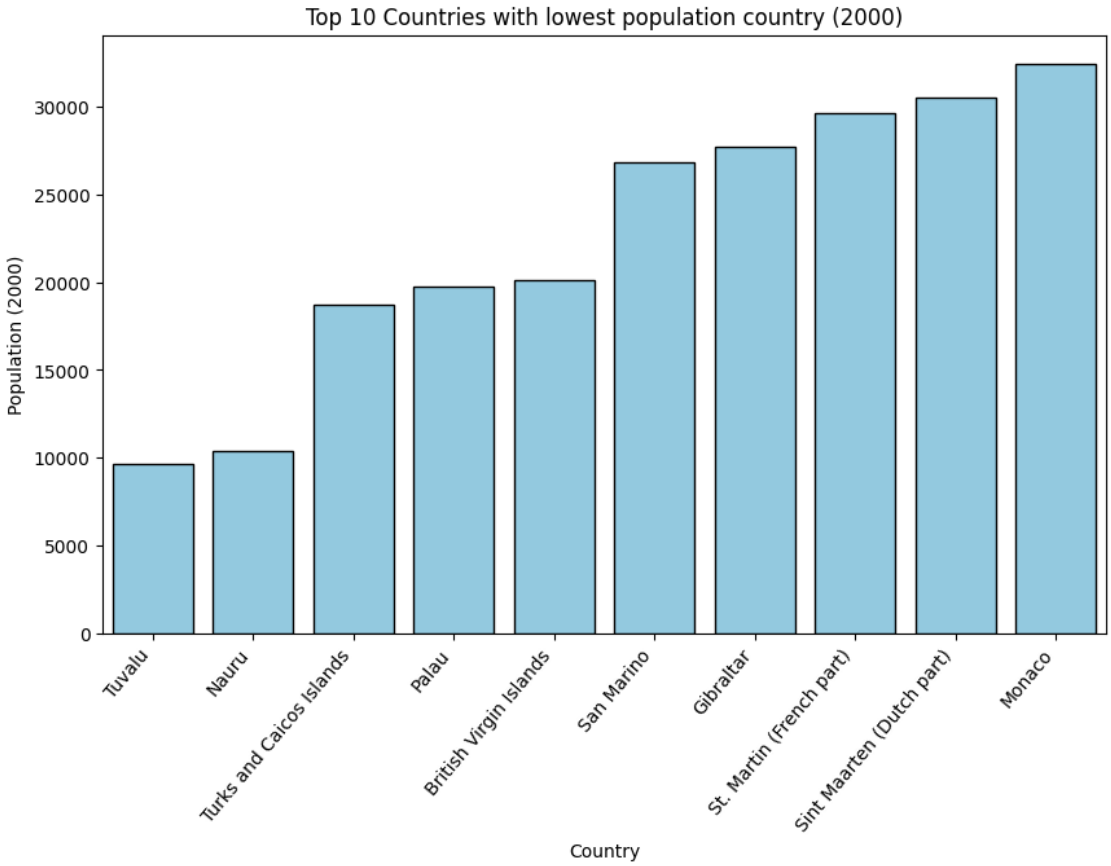
[128]:
```
# Top 10 Countries with highest Population (2022)
plot_top_countries(df,'2022',ascending=False,title='Top 10 Countries with␣
 ↪highest Population (2022)')
```



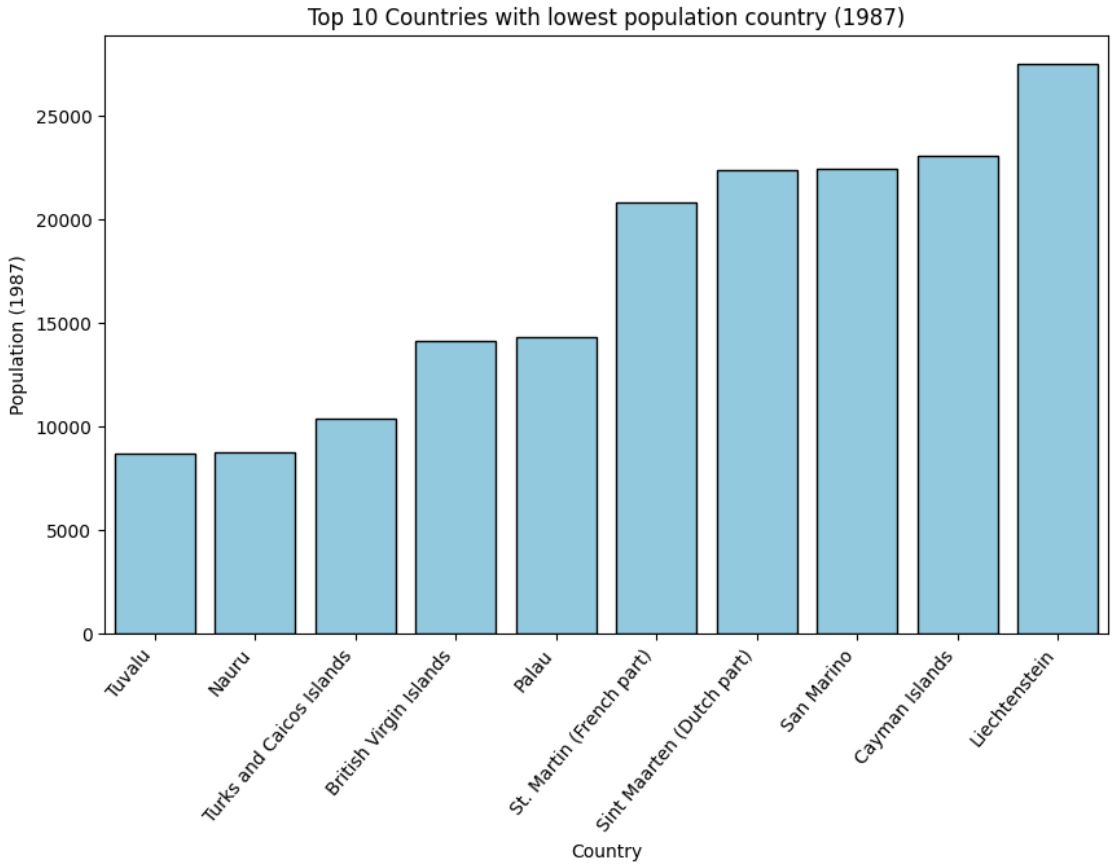
[129]:
```
# Top 10 Countries with lowest population country (2022)'
plot_top_countries(df,'2022',ascending=True,title='Top 10 Countries with lowest␣
↪population country (2022)')
```



[130]:
```
# Top 10 Countries with lowest population country (2000)
plot_top_countries(df,'2000',ascending=True,title='Top 10 Countries with lowest␣
↪population country (2000)')
```



[131]:
```
# Top 10 Countries with lowest population country (1987)
plot_top_countries(df,'1987',ascending=True,title='Top 10 Countries with lowest␣
↪population country (1987)')
```



Calculate the mean only for numerical columns along each row in a DataFrame, you can use the mean method

[132]:
```
row_avg =np.log(df.mean(axis=1))
country=df['Country Name'].to_list()
data={'Country Name':country,
      'mean (average)':row_avg}
df_avg=pd.DataFrame(data)
```

```
<ipython-input-132-92a00e2d13f2>:1: FutureWarning: Dropping of nuisance columns
in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future
version this will raise TypeError.  Select only valid columns before calling the
reduction.
  row_avg =np.log(df.mean(axis=1))
```

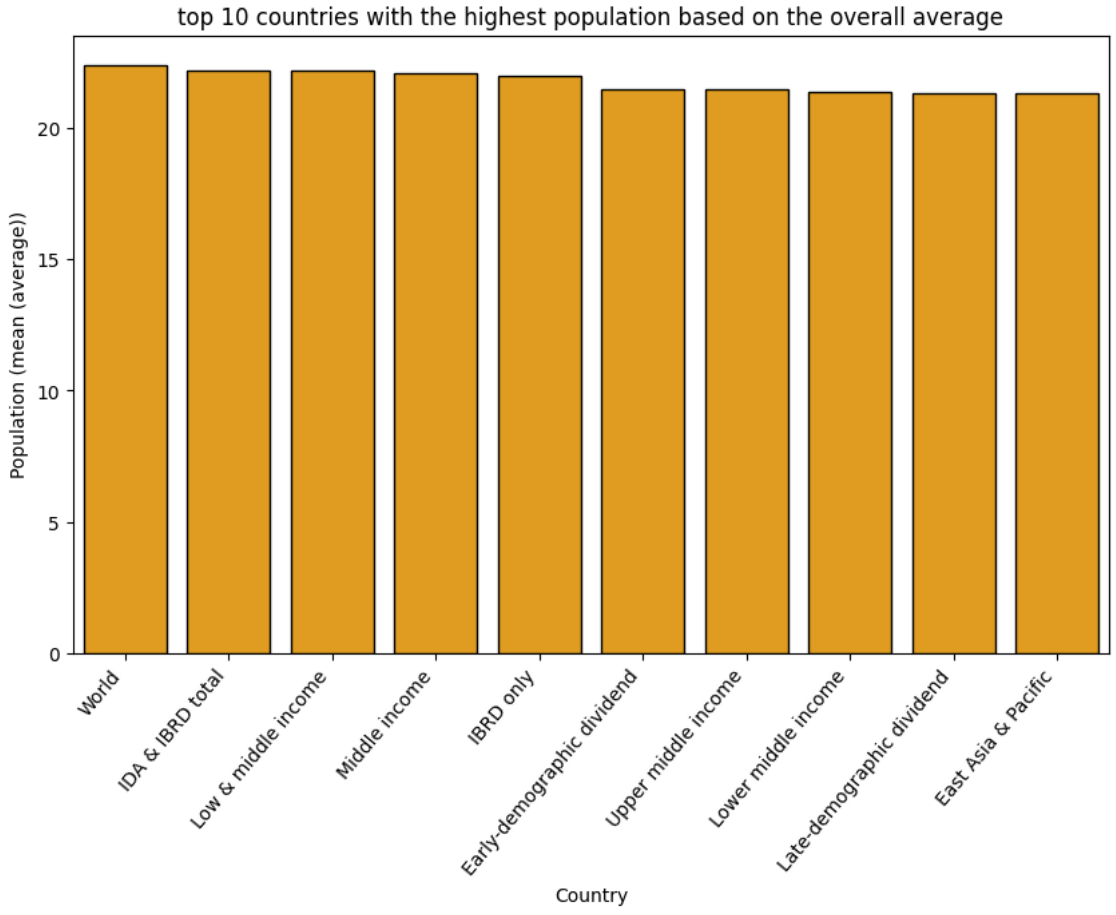
[133]:
```
pd.set_option('display.max_rows',30)
df_avg
```

[133]:

	Country Name	mean (average)
0	Aruba	11.257468

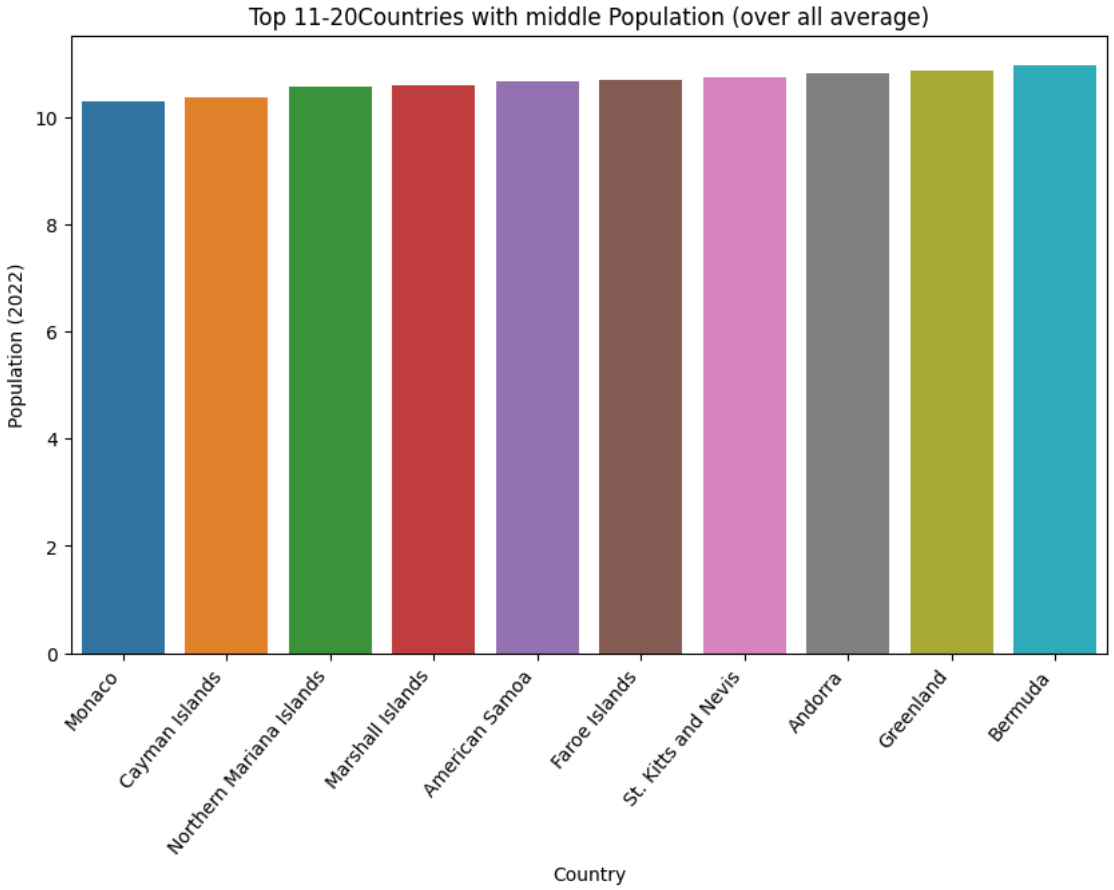
```
1     Africa Eastern and Southern       19.678914
2                     Afghanistan       16.728410
3      Africa Western and Central       19.295134
4                          Angola       16.513360
..                            …             …
261                       Kosovo       14.286503
262                  Yemen, Rep.       16.577622
263                 South Africa       17.468907
264                       Zambia       16.018880
265                     Zimbabwe       16.084263

[266 rows x 2 columns]
```

[134]:
```
# top 10 countries with the highest population based on the overall average
plot_top_countries(df_avg,'mean (average)',ascending=False␣
 ↪,color='Orange',title='top 10 countries with the highest population based on␣
 ↪the overall average')
```



[135]:
```
# top 11-20 countries with the middle population based on the overall average
df_avg1=df_avg.sort_values('mean (average)',ascending=True).iloc[10:20]
plt.figure(figsize=(10, 6))
sns.barplot(x=df_avg1['Country Name'],y=df_avg1['mean (average)'])
plt.xlabel('Country')
plt.ylabel('Population (2022)')
plt.title('Top 11-20Countries with middle Population (over all average)')
plt.xticks(rotation=50, ha='right')
```

[135]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
 [Text(0, 0, 'Monaco'),
  Text(1, 0, 'Cayman Islands'),
  Text(2, 0, 'Northern Mariana Islands'),
  Text(3, 0, 'Marshall Islands'),
  Text(4, 0, 'American Samoa'),
  Text(5, 0, 'Faroe Islands'),
  Text(6, 0, 'St. Kitts and Nevis'),
  Text(7, 0, 'Andorra'),
  Text(8, 0, 'Greenland'),
  Text(9, 0, 'Bermuda')])
```



[136]:
```
df.iloc[6, df.columns.get_loc("Country Name")]
```

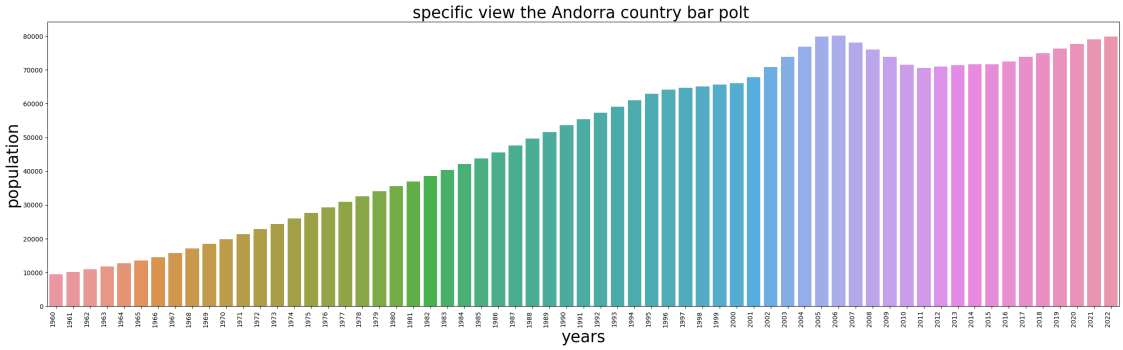
[136]:
```
'Andorra'
```

[137]:
```
df.iloc[6]
```

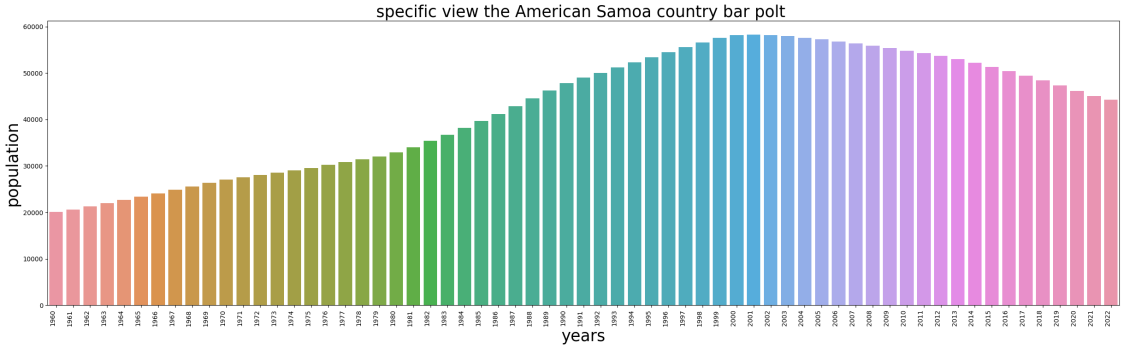
[137]:
```
Country Name                 Andorra
Country Code                     AND
Indicator Name    Population, total
Indicator Code          SP.POP.TOTL
1960                         9443.0
                               ...
2018                        75013.0
2019                        76343.0
2020                        77700.0
2021                        79034.0
2022                        79824.0
Name: 6, Length: 67, dtype: object
```

[138]:
```
# Visualization - barplot for a specific country
def specific_country_polt(df,country_index_number):
  row_to_plot = df.iloc[country_index_number]
  plt.figure(figsize=(30,8))
  sns.barplot(x=row_to_plot.index[4:],y=row_to_plot.values[4:])
  plt.xlabel('years',fontsize=25)
  plt.ylabel('population',fontsize=25)
  plt.xticks(rotation=90, ha='right')
  plt.title(f'specific view the {row_to_plot[0]} country bar polt ',fontsize=25)
```

[139]:
```
specific_country_polt(df,6)
```



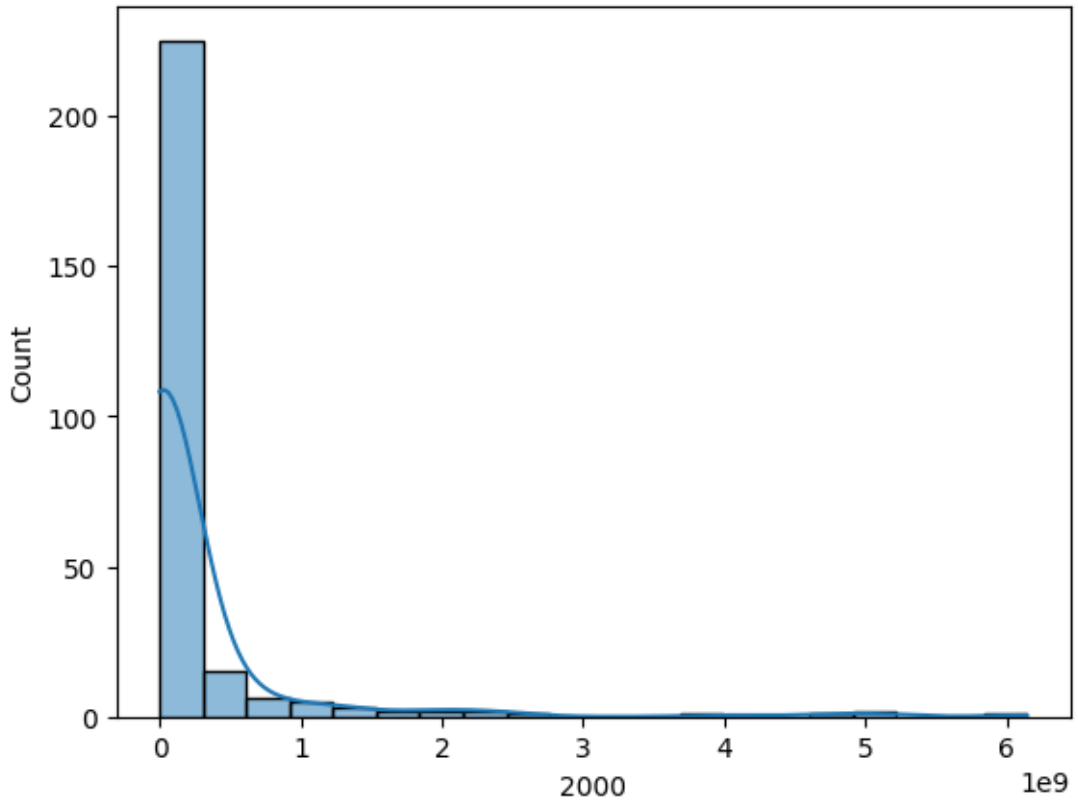
[140]:
```
specific_country_polt(df,11)
```



4 visualization - Histograms

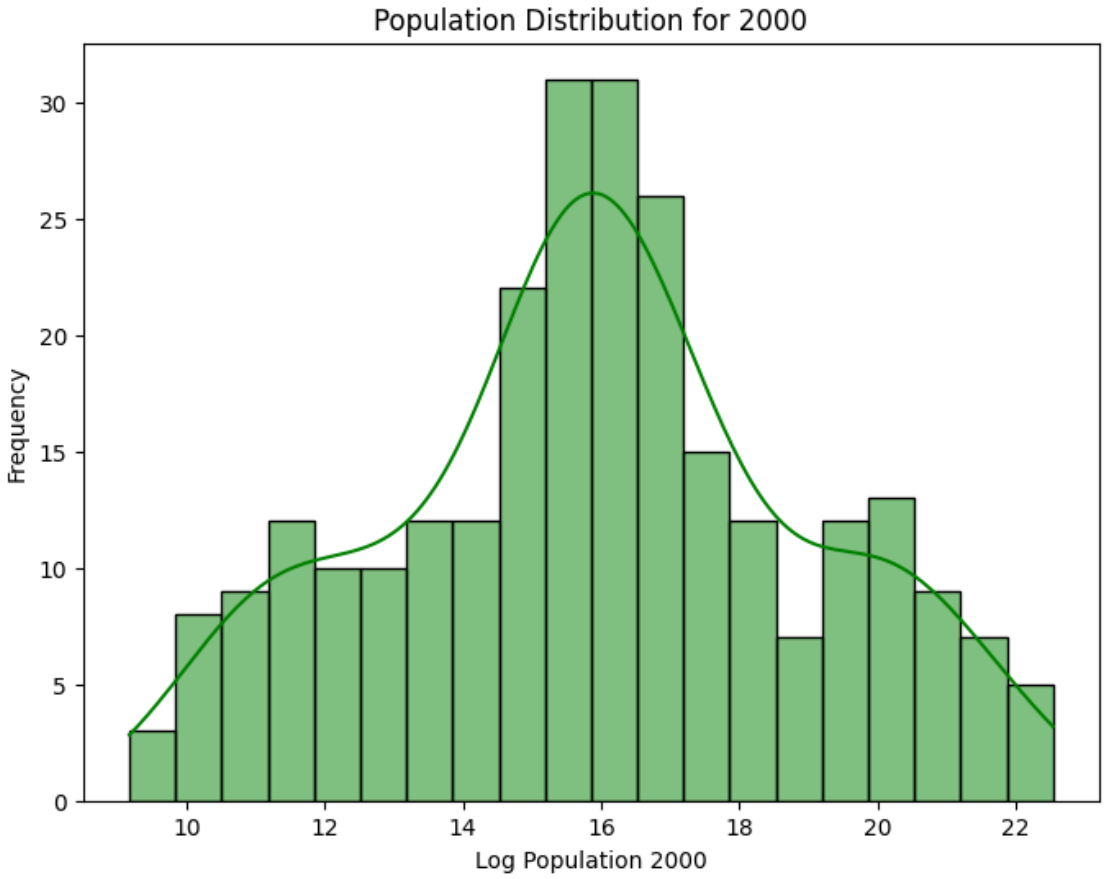
[141]:
```
sns.histplot(x=df['2000'],bins=20,kde=True)
```

[141]:
```
<Axes: xlabel='2000', ylabel='Count'>
```

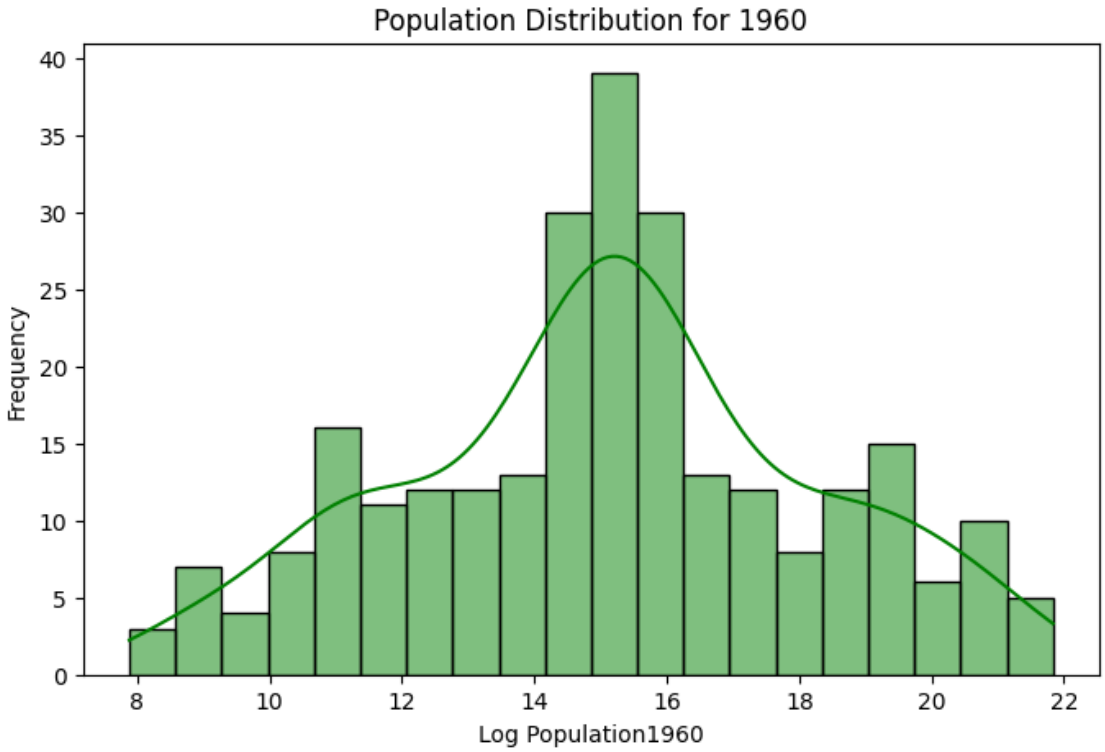


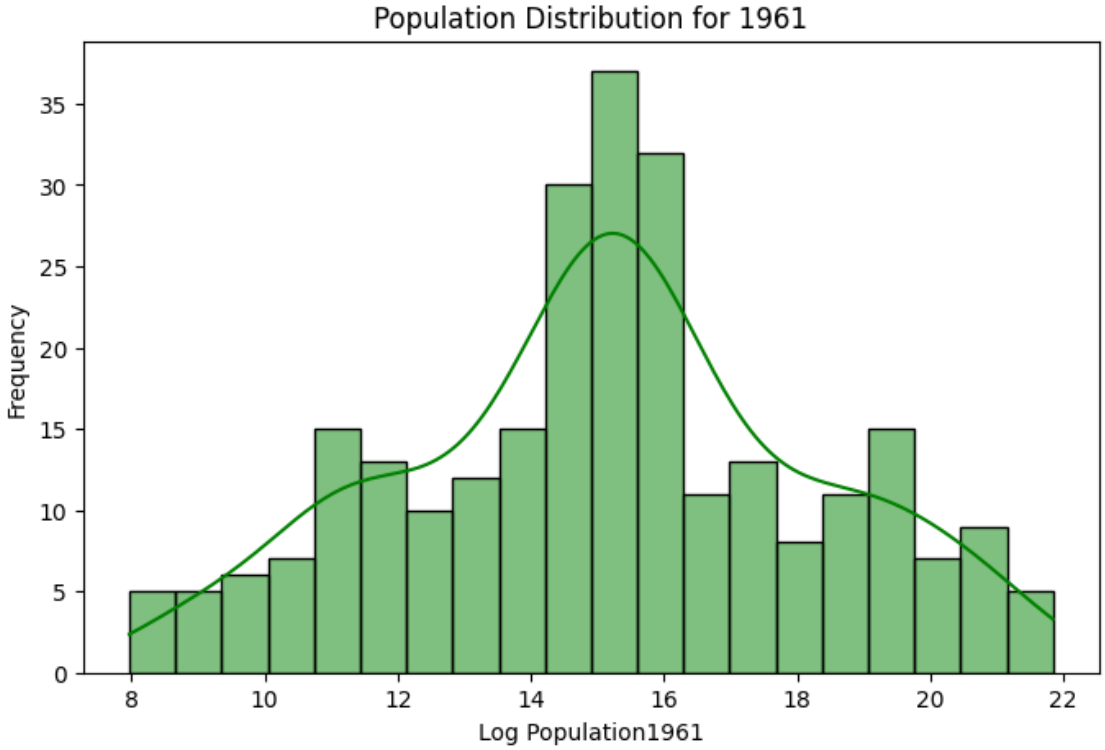
Let's apply log transformation for better understanding of the data distribution

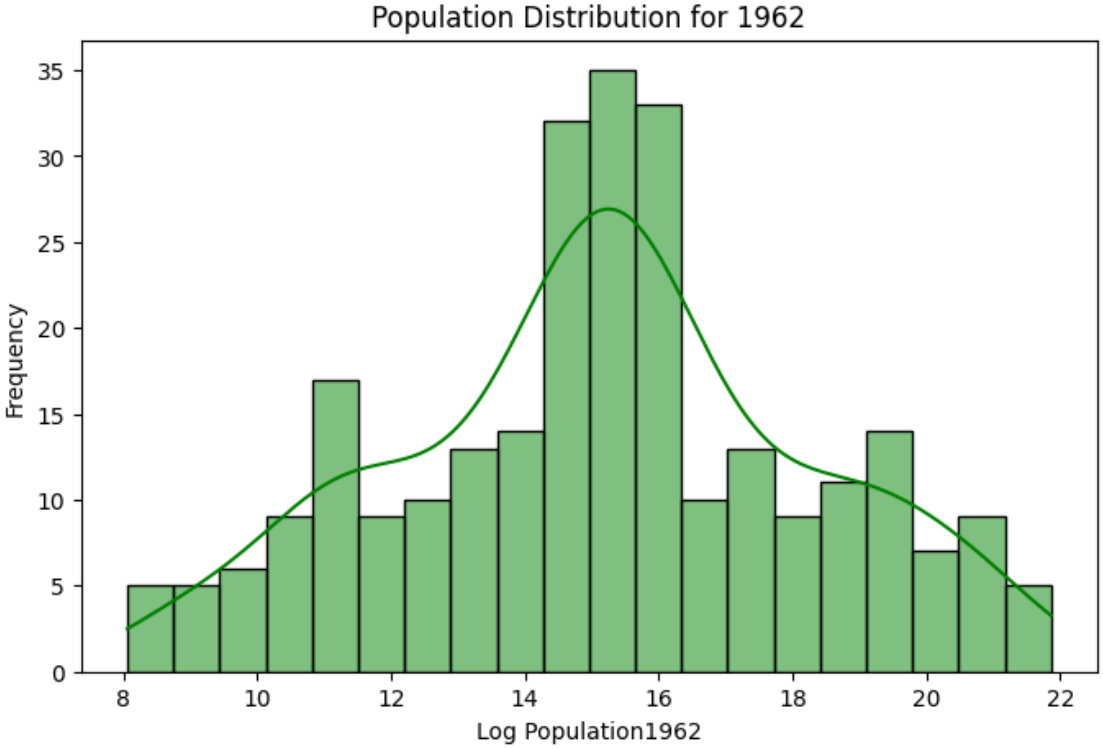
[142]:
```
plt.figure(figsize=(8,6))
sns.histplot(np.log(df['2000']),bins=20,kde=True,color='Green')
plt.title(f'Population Distribution for 2000')
plt.xlabel(f'Log Population 2000')
plt.ylabel('Frequency')
plt.show()
```

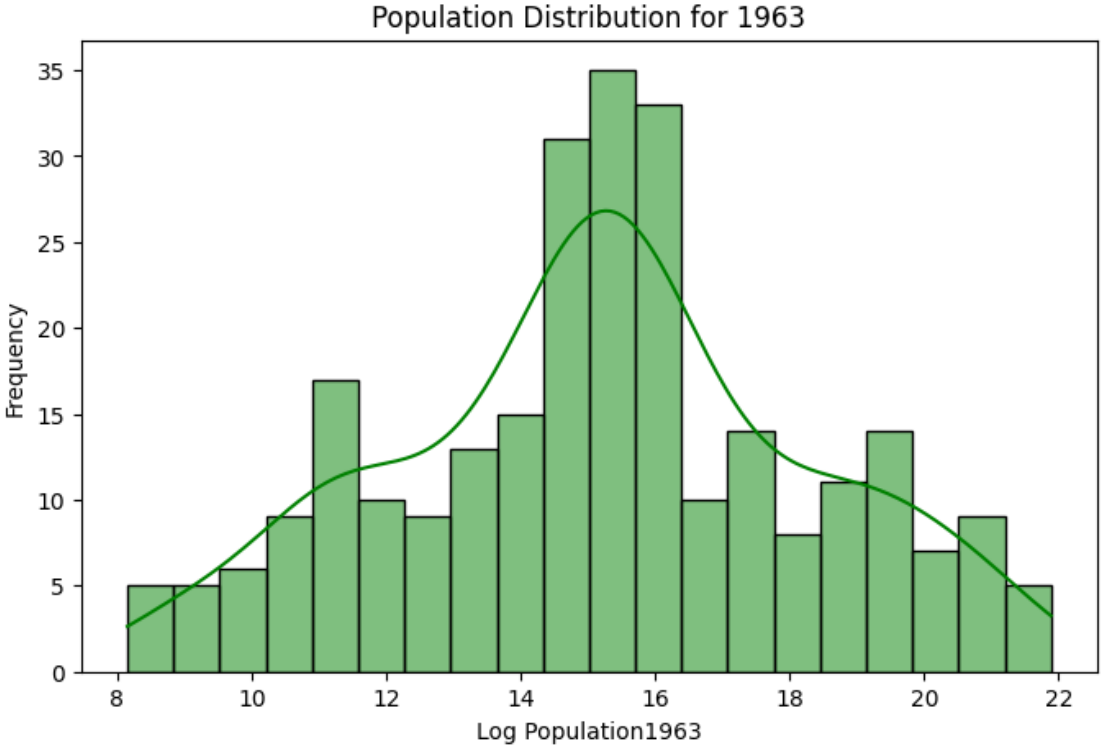


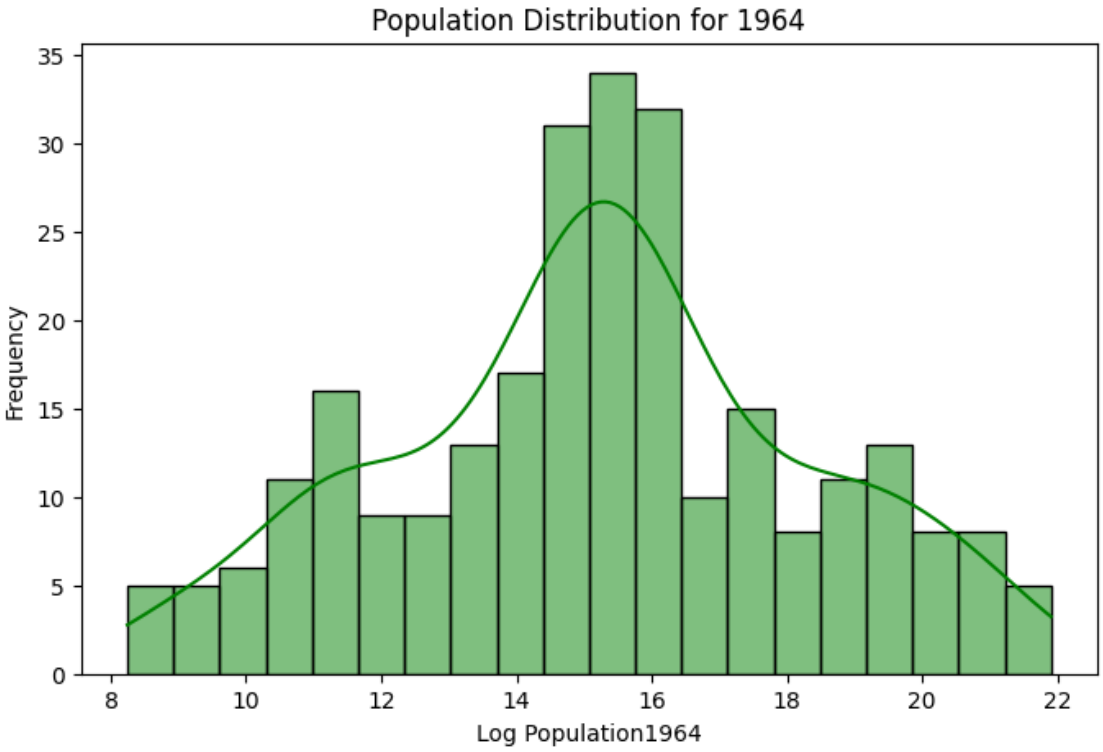
[143]:
```
for i in df.columns[4:]:
  plt.figure(figsize=(8,5))
  sns.histplot(np.log(df[i]),bins=20,kde=True,color='Green')
  plt.title(f'Population Distribution for {i}')
  plt.xlabel(f'Log Population{i}')
  plt.ylabel('Frequency')
  plt.show()
```

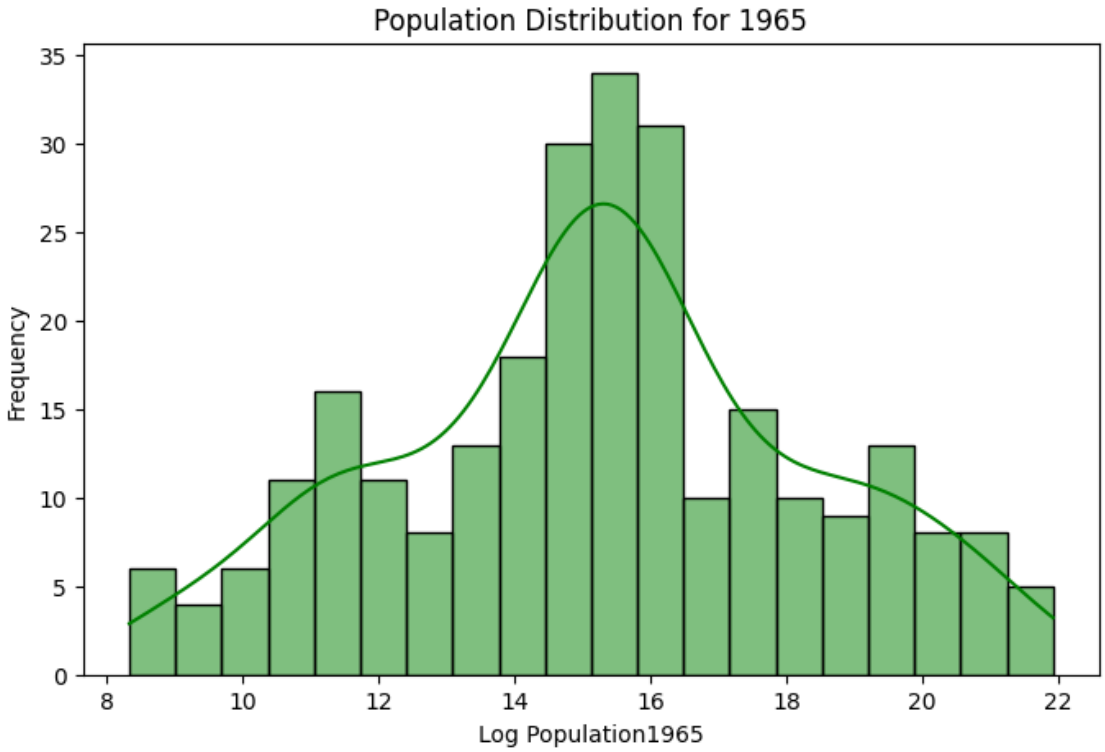

Population Distribution for 1960
Frequency
Log Population1960
0
5
10
15
20
25
30
35
40
8
10
12
14
16
18
20
22

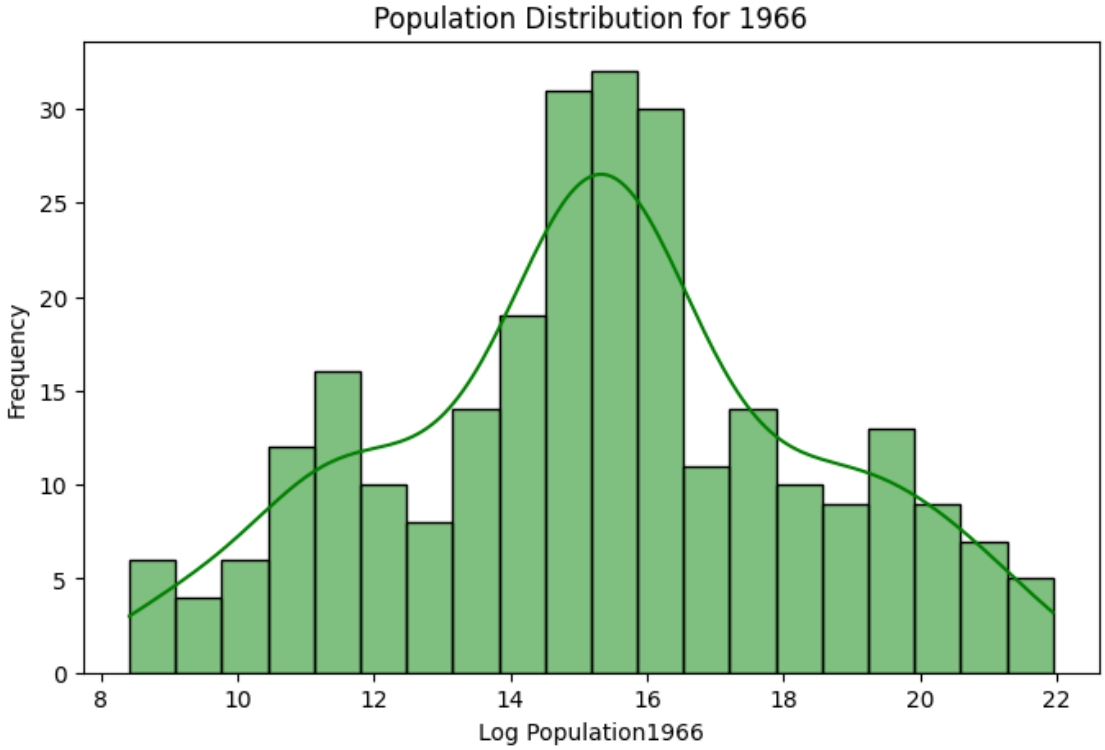

Population Distribution for 1961
Frequency
Log Population1961
0
5
10
15
20
25
30
35
8
10
12
14
16
18
20
22

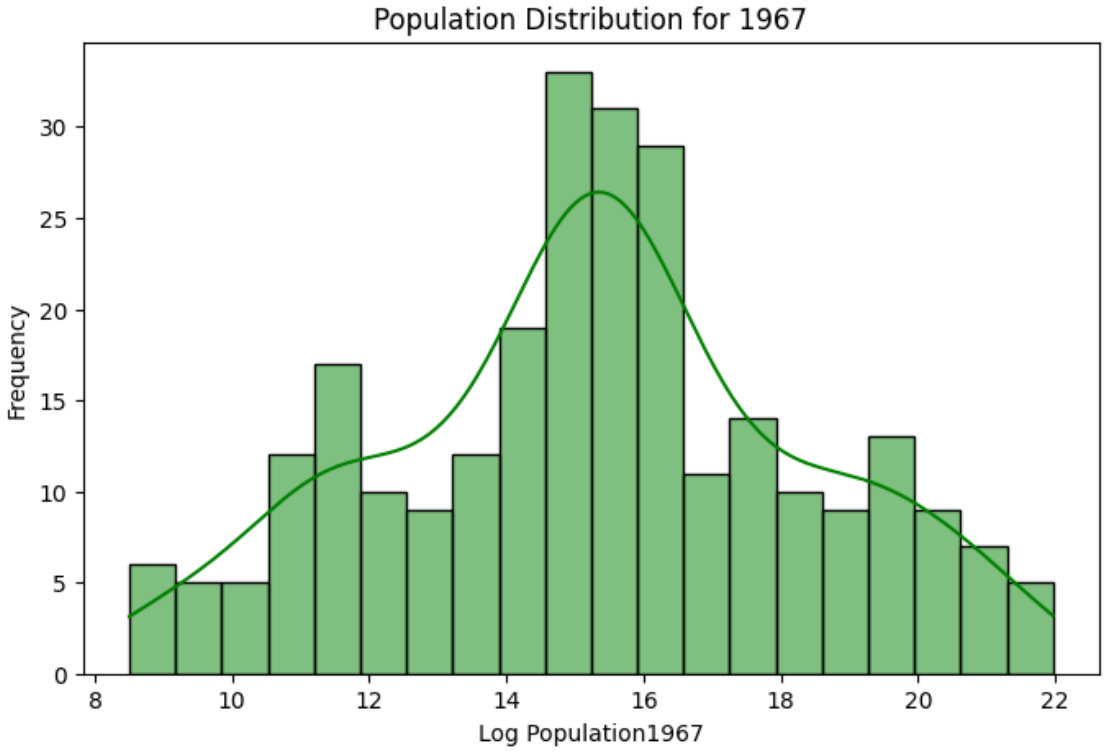

Population Distribution for 1962
Frequency
Log Population1962
0
5
10
15
20
25
30
35
8
10
12
14
16
18
20
22

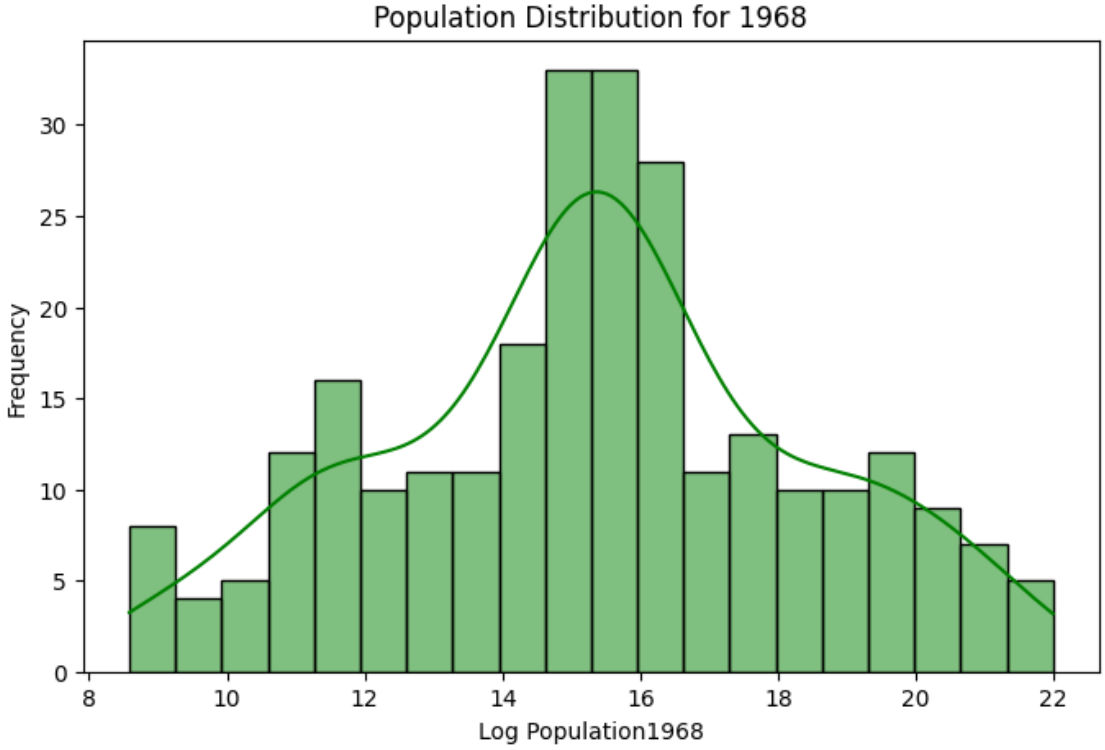

Population Distribution for 1963
Frequency
Log Population1963
0
5
10
15
20
25
30
35
8
10
12
14
16
18
20
22

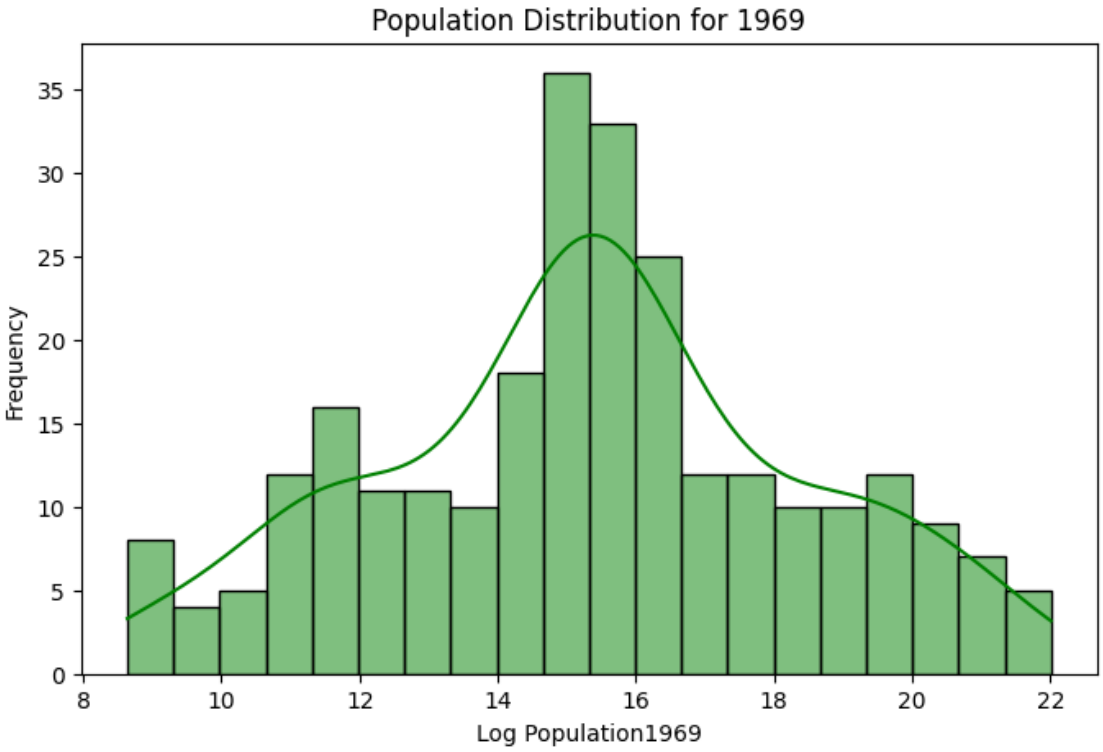

Population Distribution for 1964
Frequency
Log Population1964
0
5
10
15
20
25
30
35
8
10
12
14
16
18
20
22

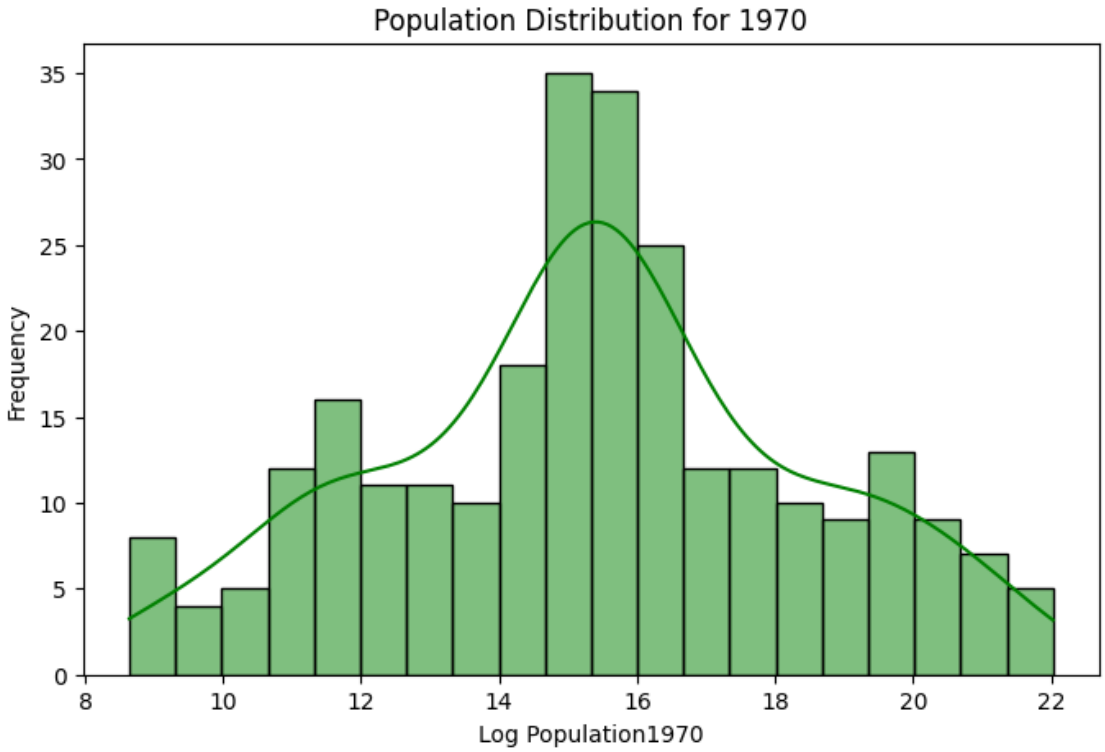

Population Distribution for 1965
Frequency
Log Population1965
0
5
10
15
20
25
30
35
8
10
12
14
16
18
20
22

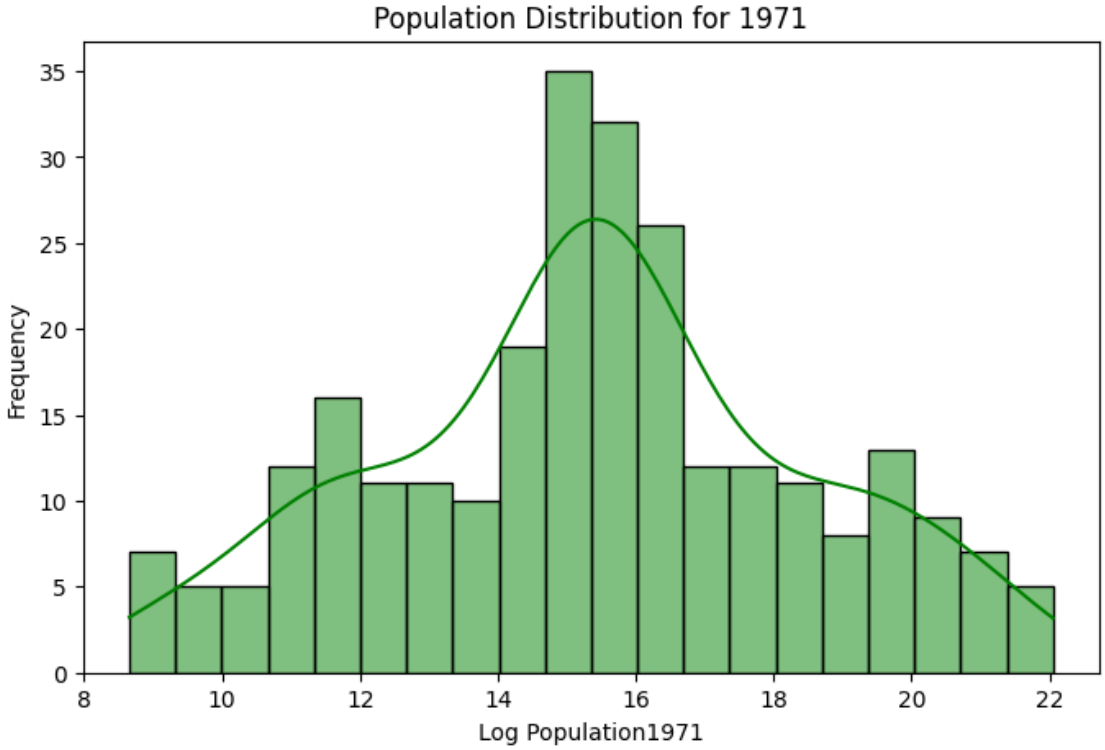

Population Distribution for 1966
Frequency
Log Population1966
0
5
10
15
20
25
30
8
10
12
14
16
18
20
22

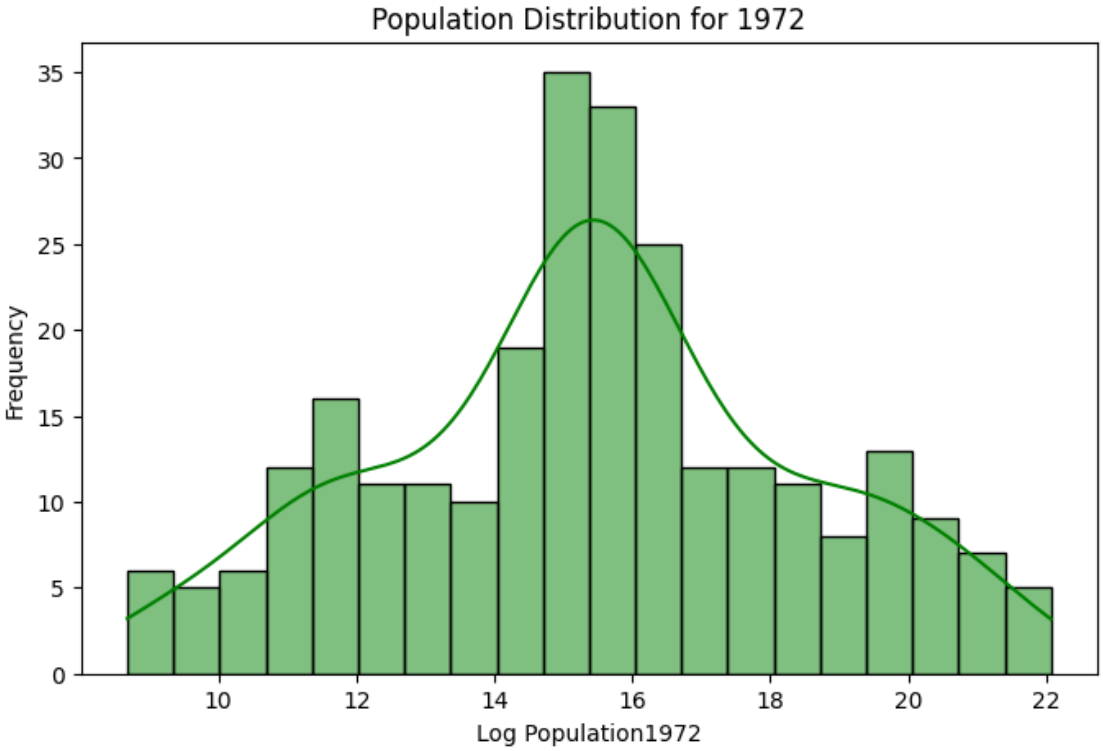

Population Distribution for 1967
Frequency
Log Population1967
0
5
10
15
20
25
30
8
10
12
14
16
18
20
22

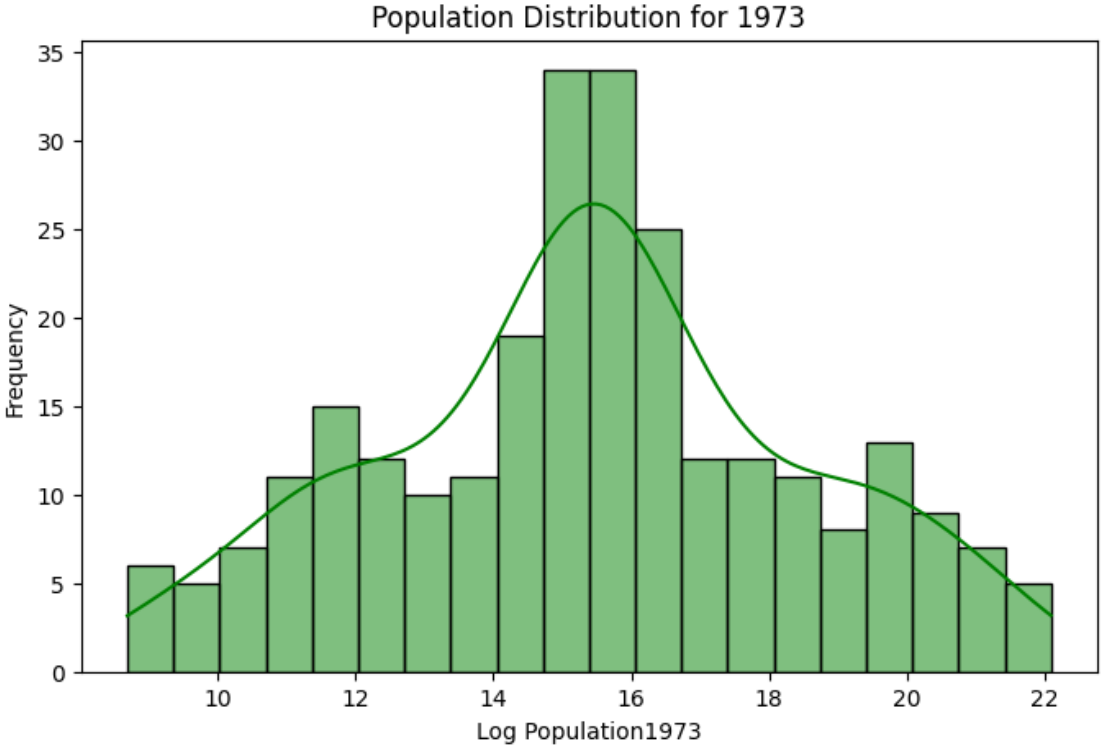

Population Distribution for 1968
Frequency
Log Population1968
0
5
10
15
20
25
30
8
10
12
14
16
18
20
22

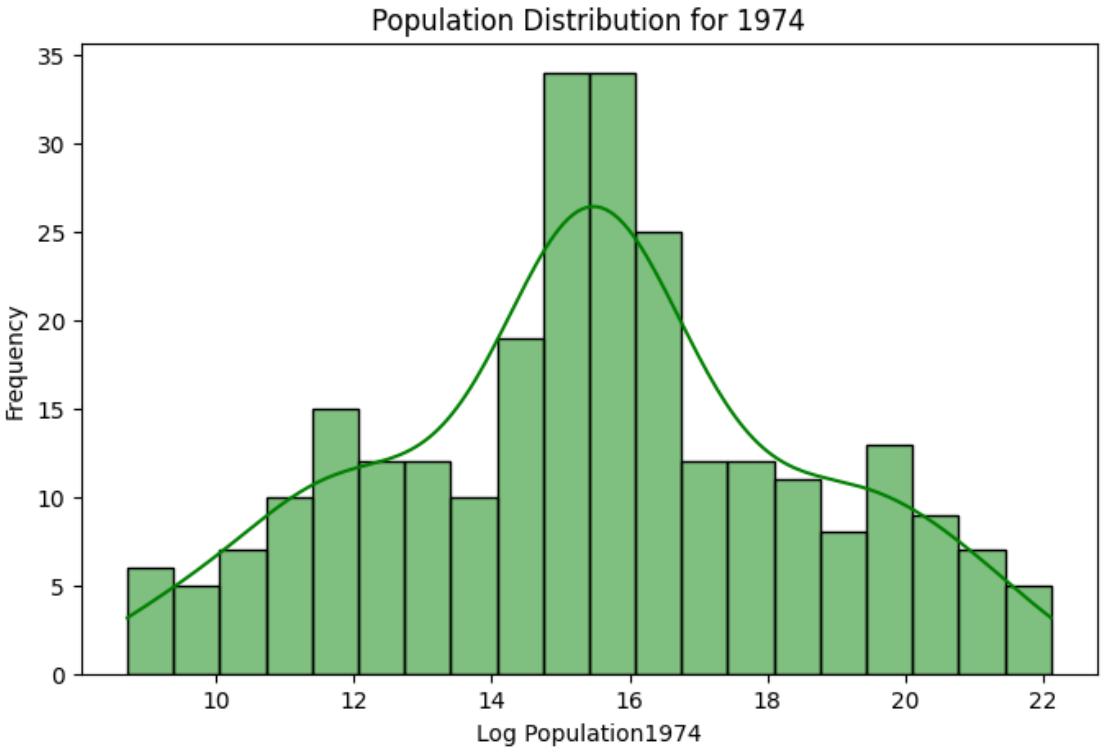

Population Distribution for 1969
Frequency
Log Population1969
0
5
10
15
20
25
30
35
8
10
12
14
16
18
20
22

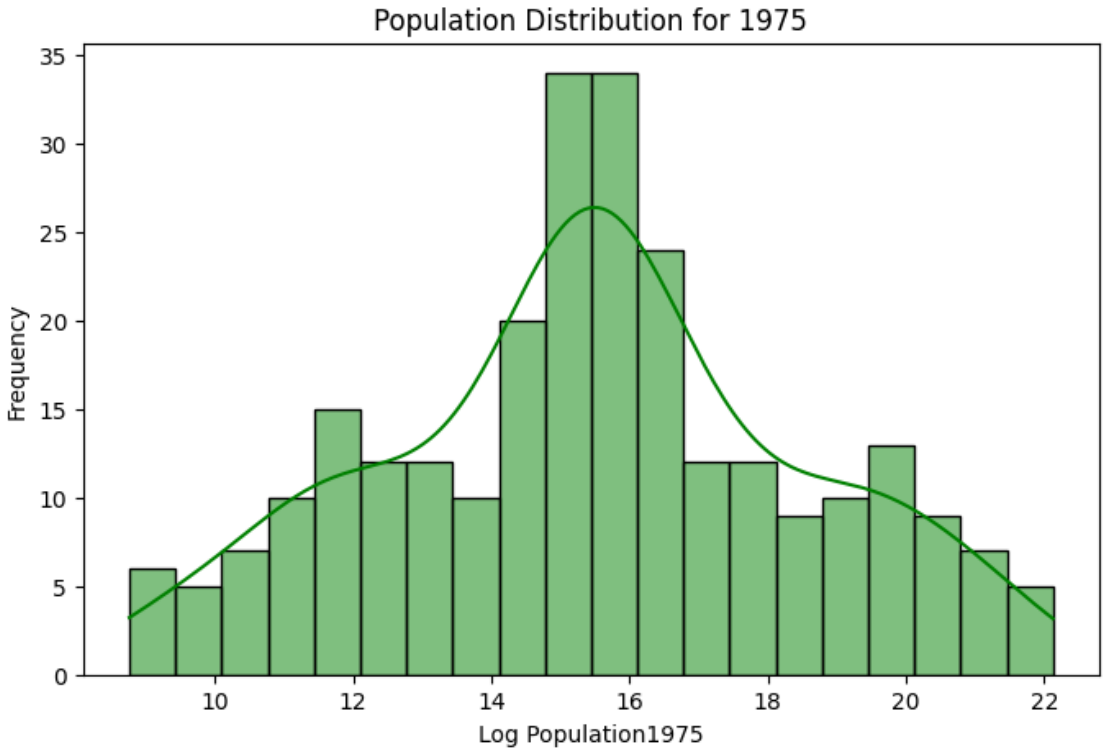

Population Distribution for 1970
Frequency
Log Population1970
0
5
10
15
20
25
30
35
8
10
12
14
16
18
20
22

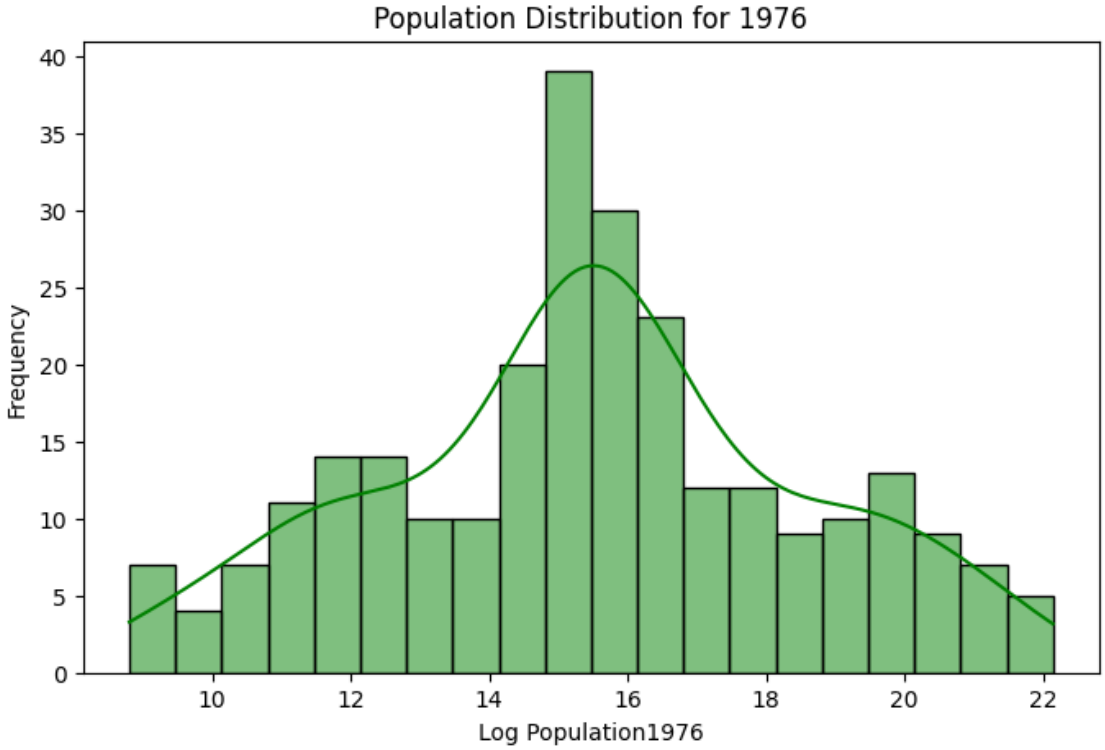

Population Distribution for 1971
Frequency
Log Population1971
0
5
10
15
20
25
30
35
8
10
12
14
16
18
20
22

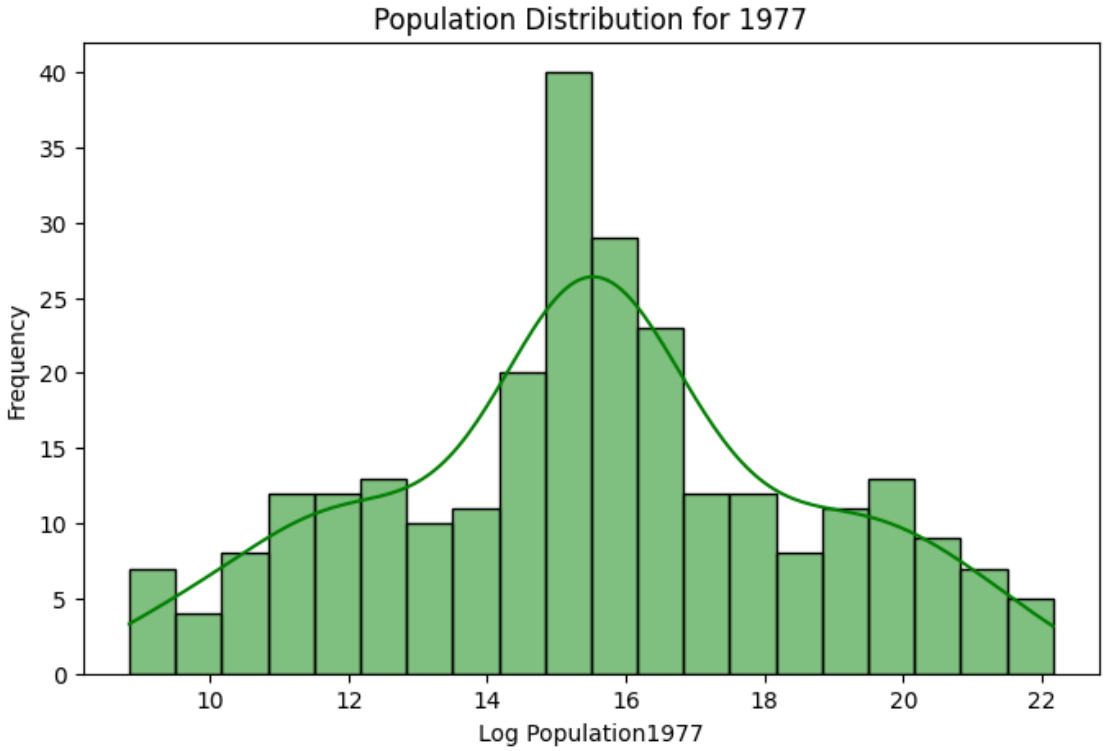

Population Distribution for 1972
Frequency
Log Population1972
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

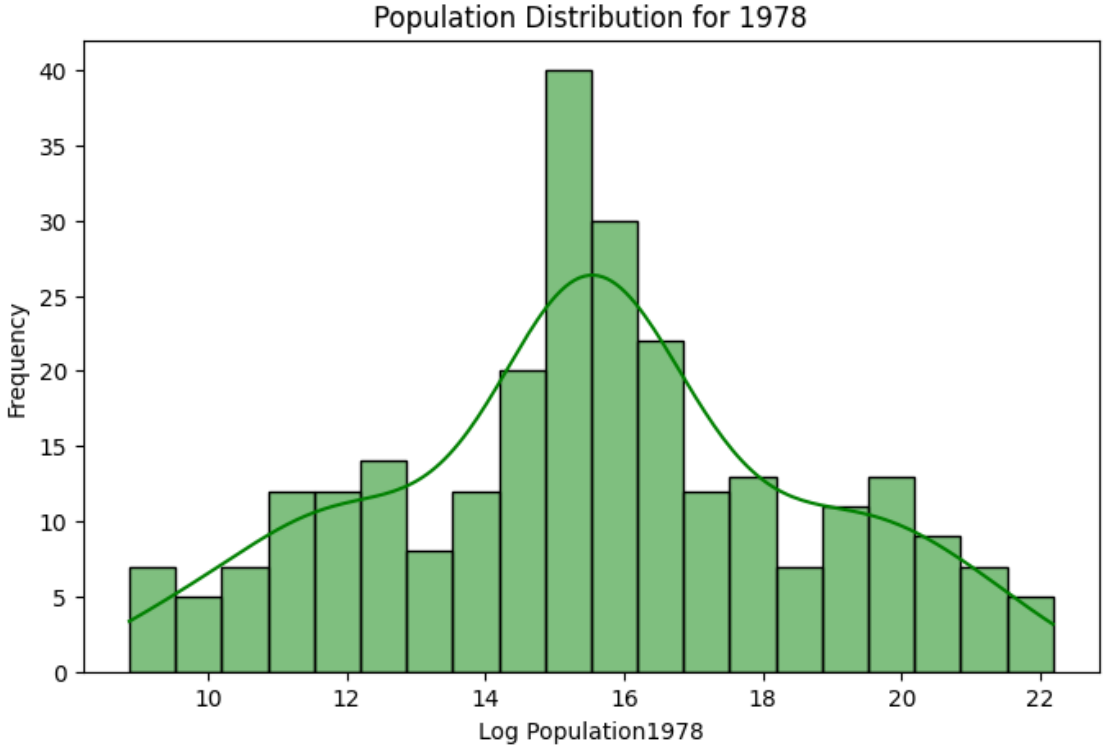

Population Distribution for 1973
Frequency
Log Population1973
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

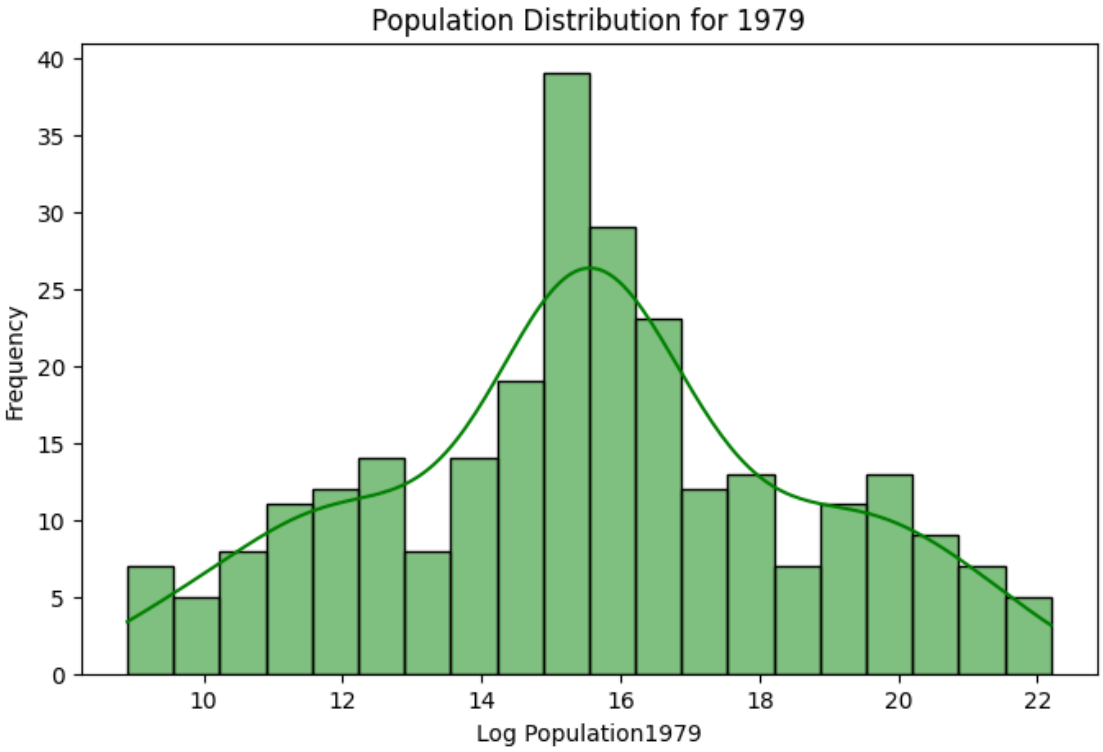

Population Distribution for 1974
Frequency
Log Population1974
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

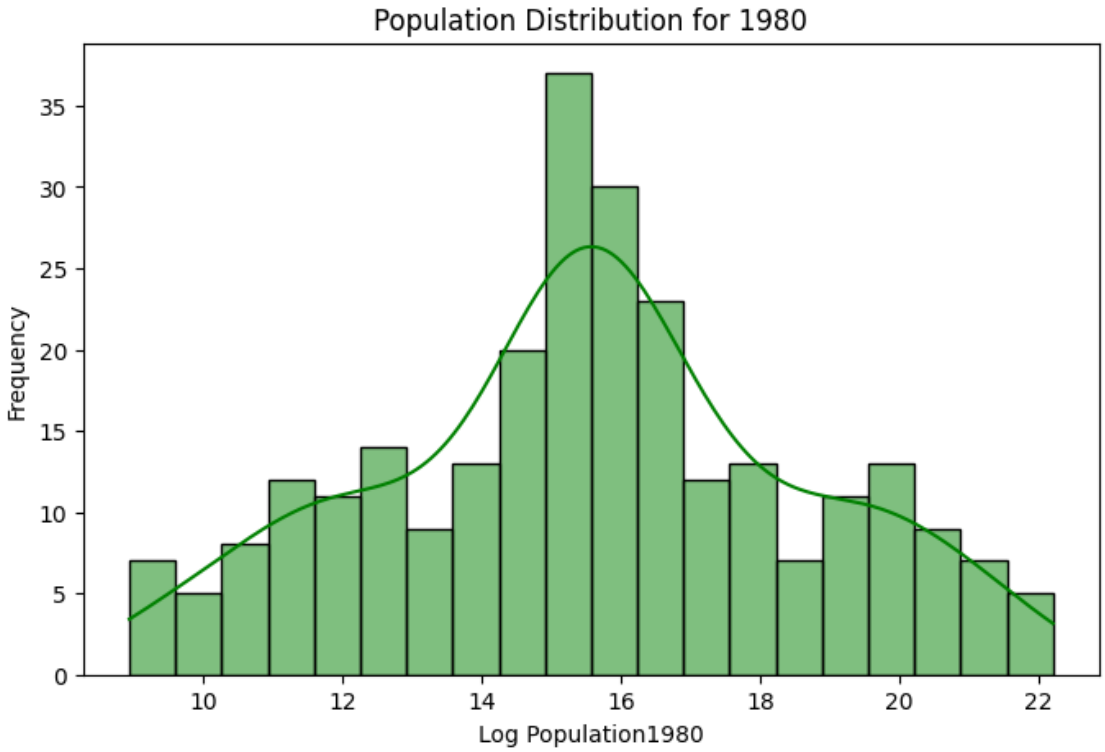

Population Distribution for 1975
Frequency
Log Population1975
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

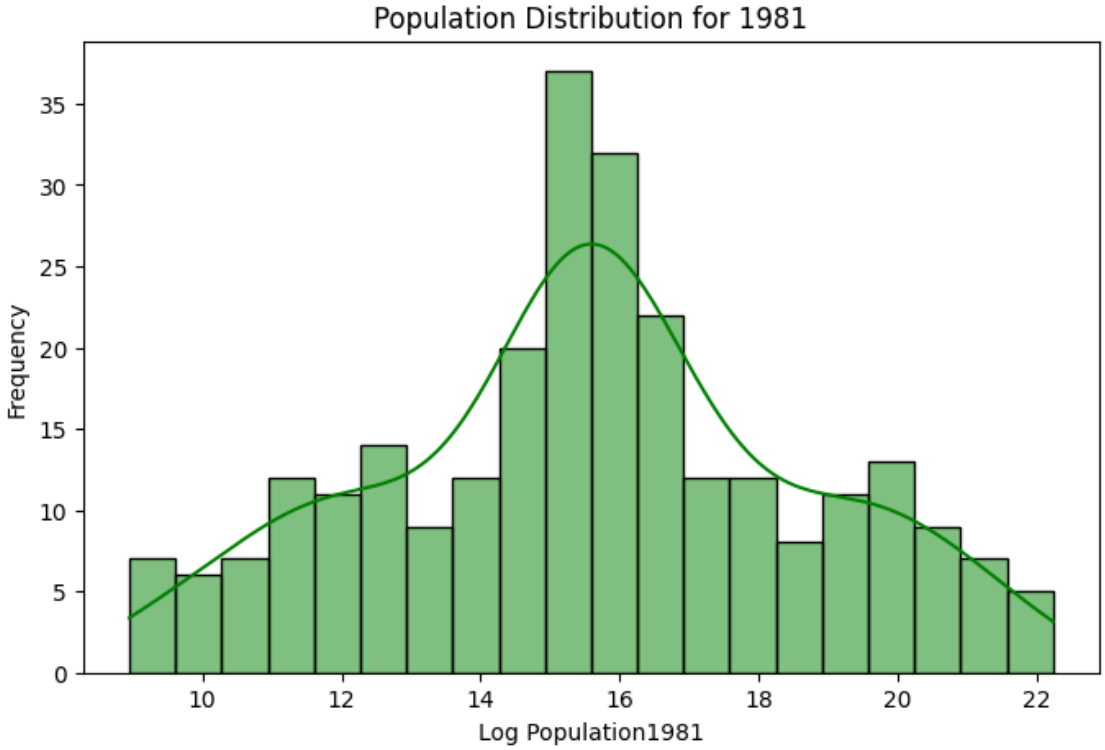

Population Distribution for 1976
Frequency
Log Population1976
0
5
10
15
20
25
30
35
40
10
12
14
16
18
20
22

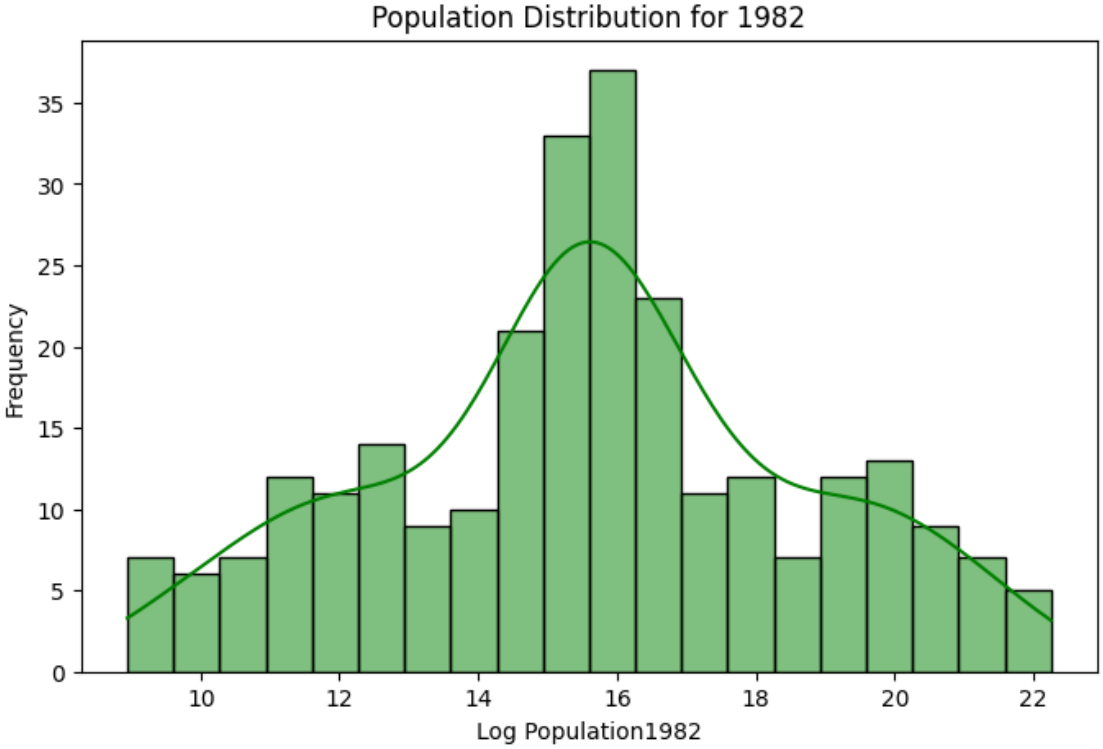

Population Distribution for 1977
Frequency
Log Population1977
0
5
10
15
20
25
30
35
40
10
12
14
16
18
20
22

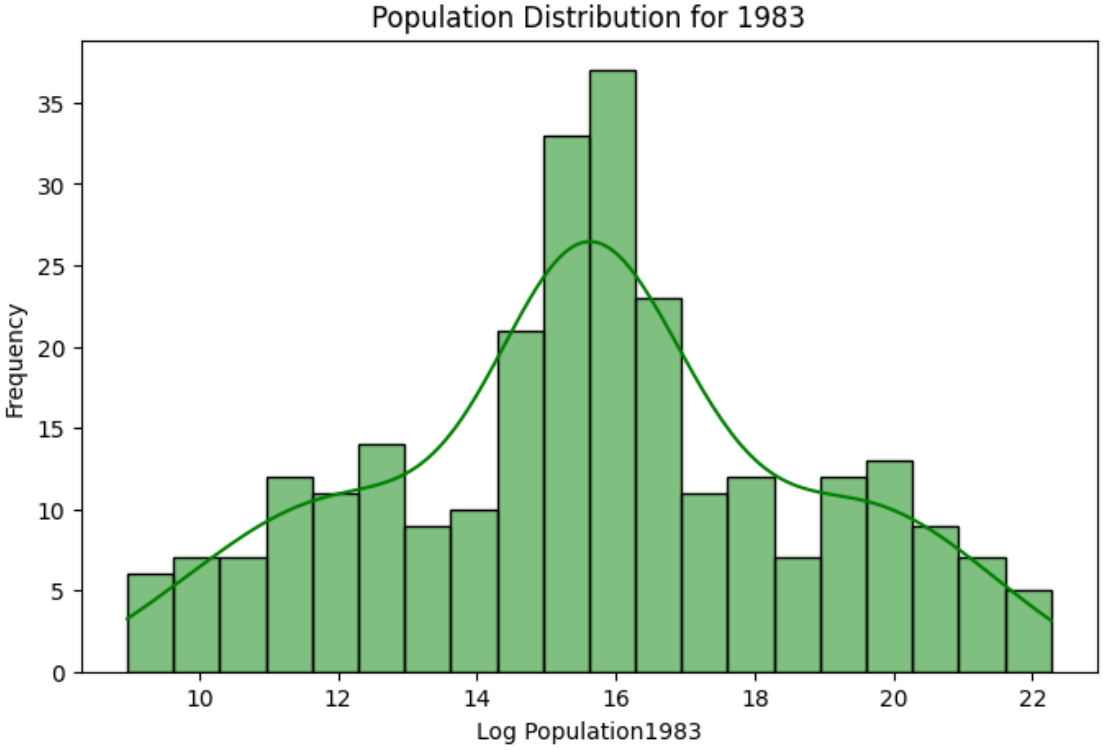

Population Distribution for 1978
Frequency
Log Population1978
0
5
10
15
20
25
30
35
40
10
12
14
16
18
20
22

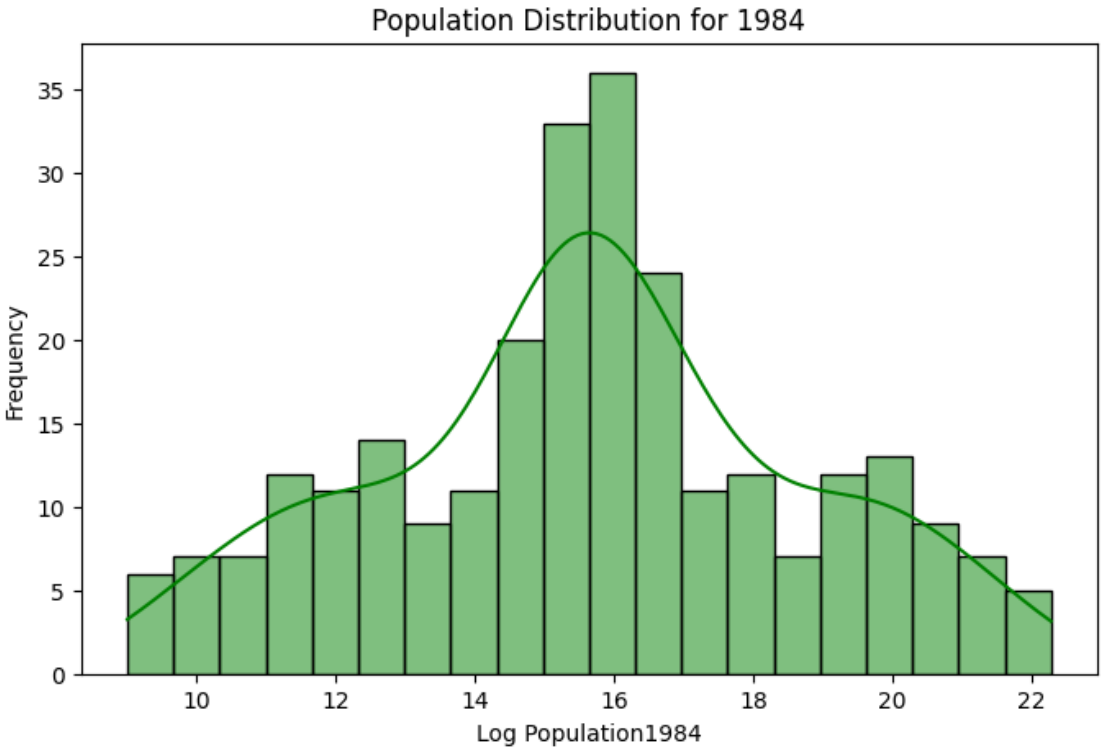

Population Distribution for 1979
Frequency
Log Population1979
0
5
10
15
20
25
30
35
40
10
12
14
16
18
20
22

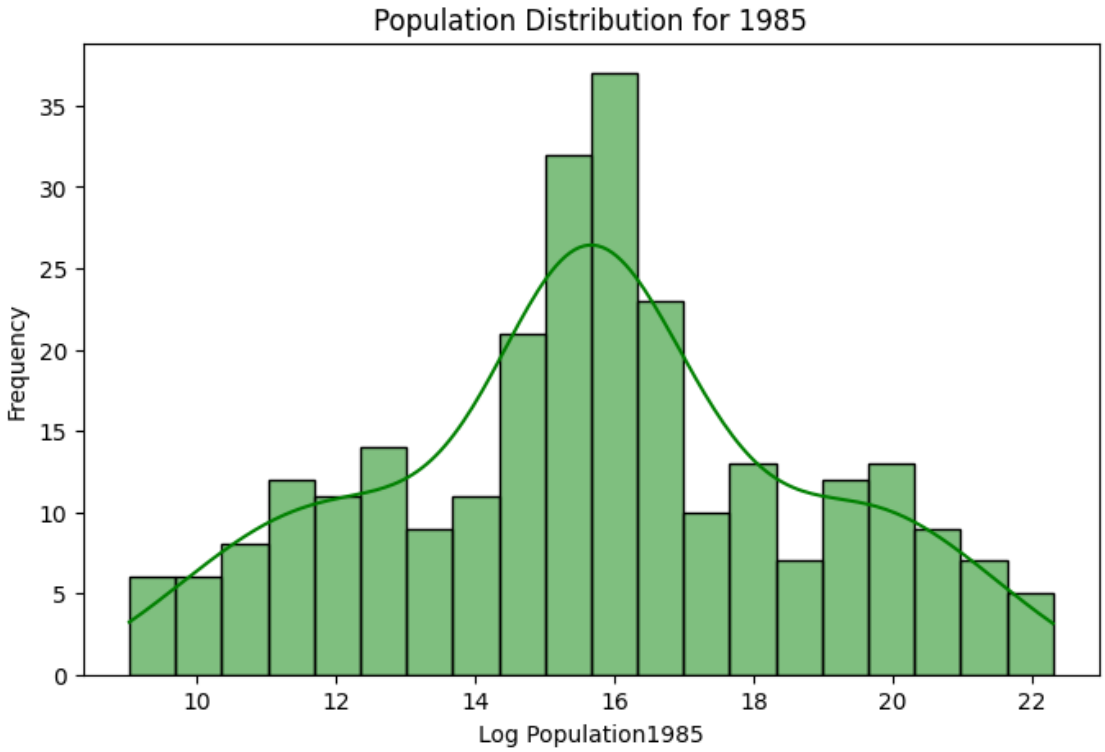

Population Distribution for 1980
Frequency
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22
Log Population1980

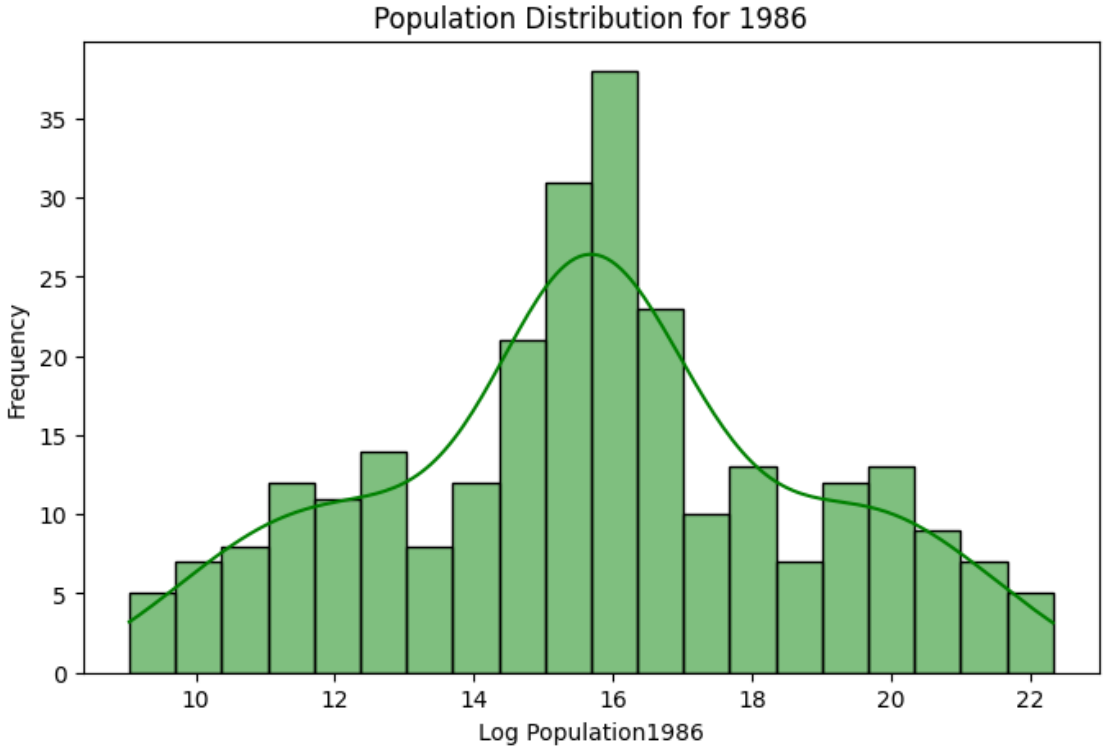

Population Distribution for 1981
Frequency
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22
Log Population1981

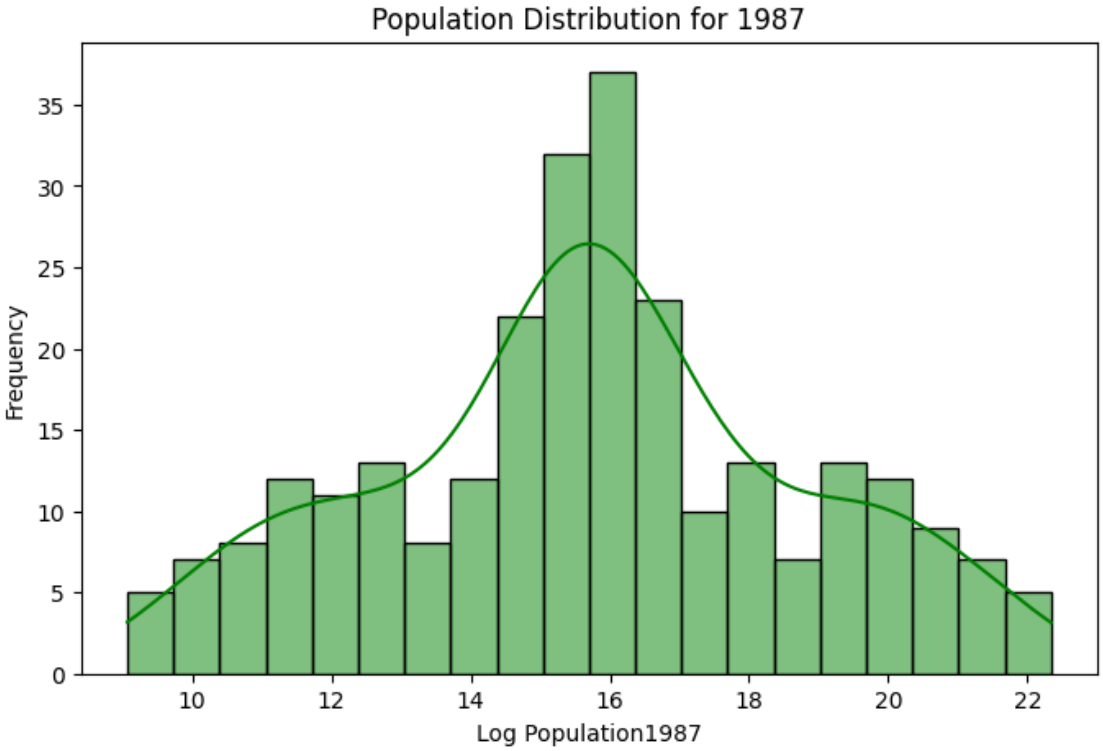

Population Distribution for 1982
Frequency
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22
Log Population1982

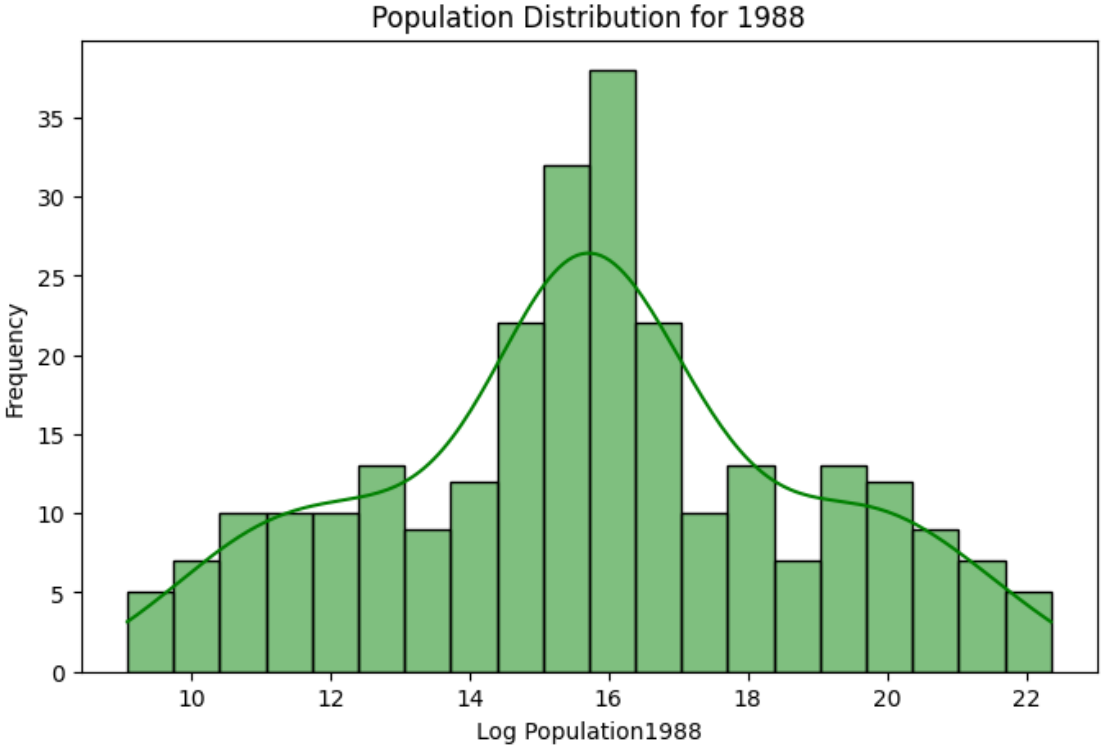

Population Distribution for 1983
Frequency
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22
Log Population1983

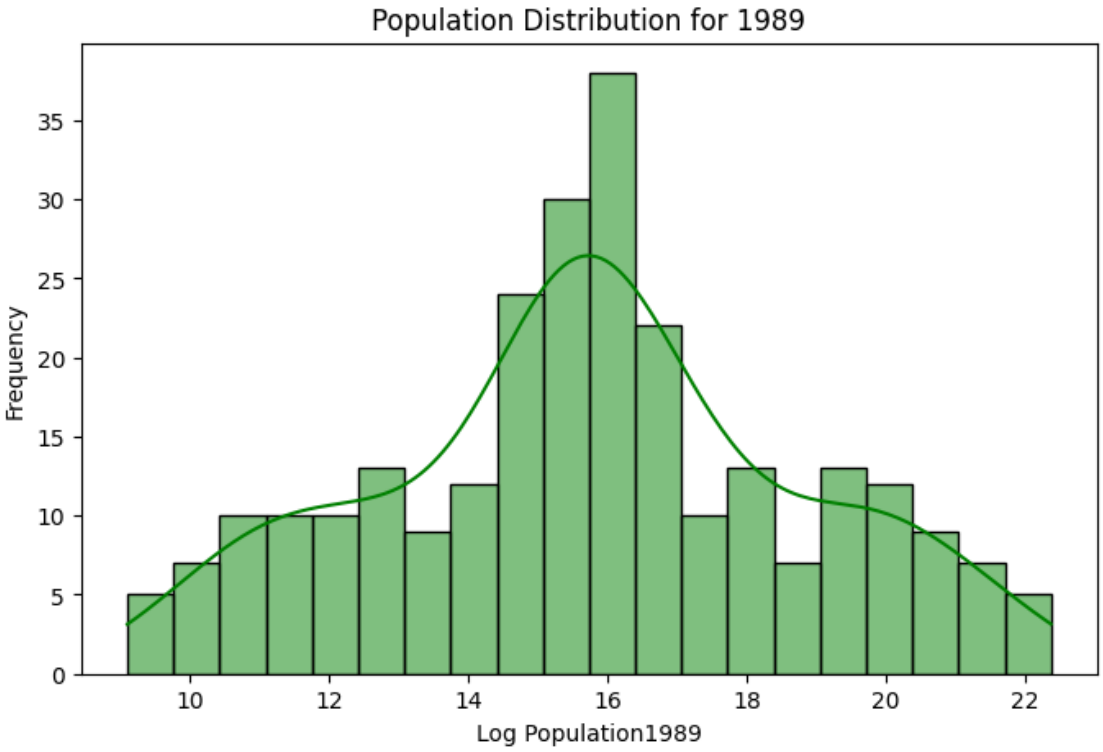

Population Distribution for 1984
Frequency
Log Population1984
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

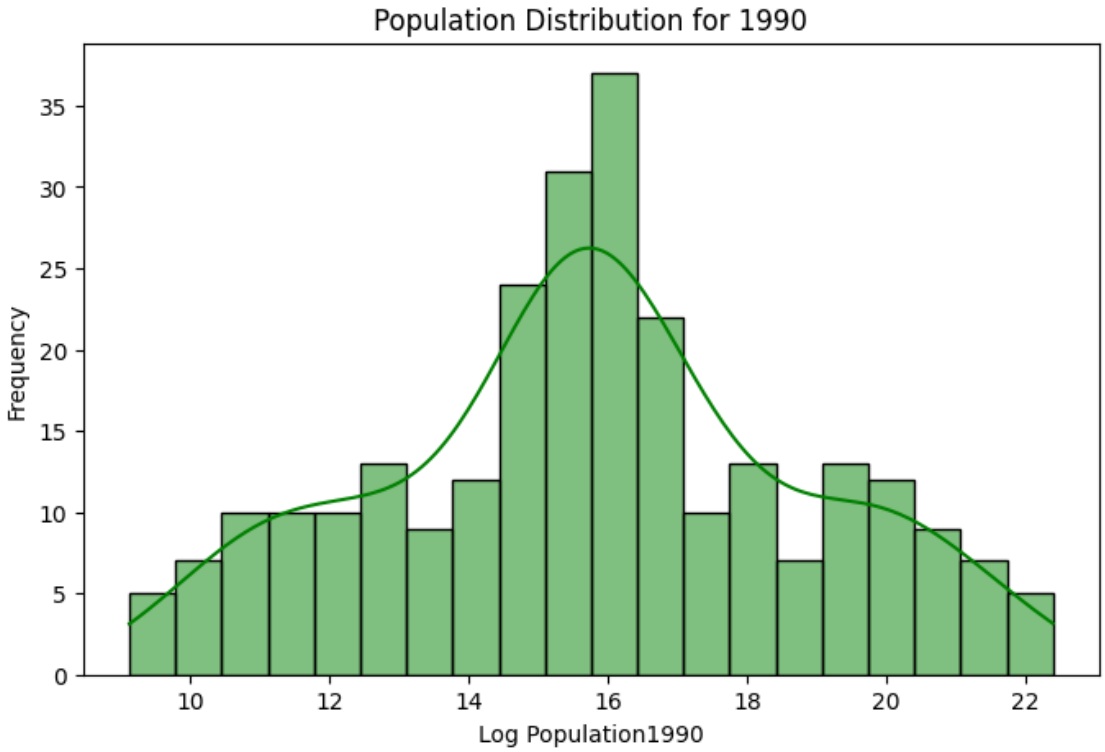

Population Distribution for 1985
Frequency
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22
Log Population1985

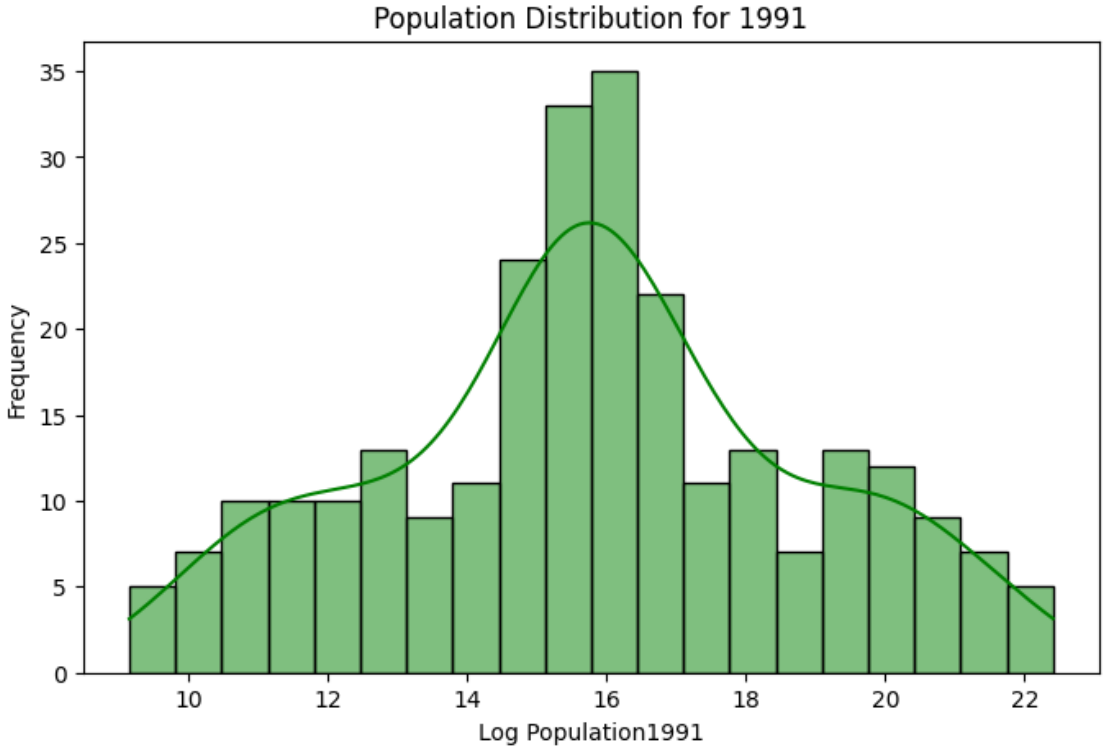

Population Distribution for 1986
Frequency
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22
Log Population1986

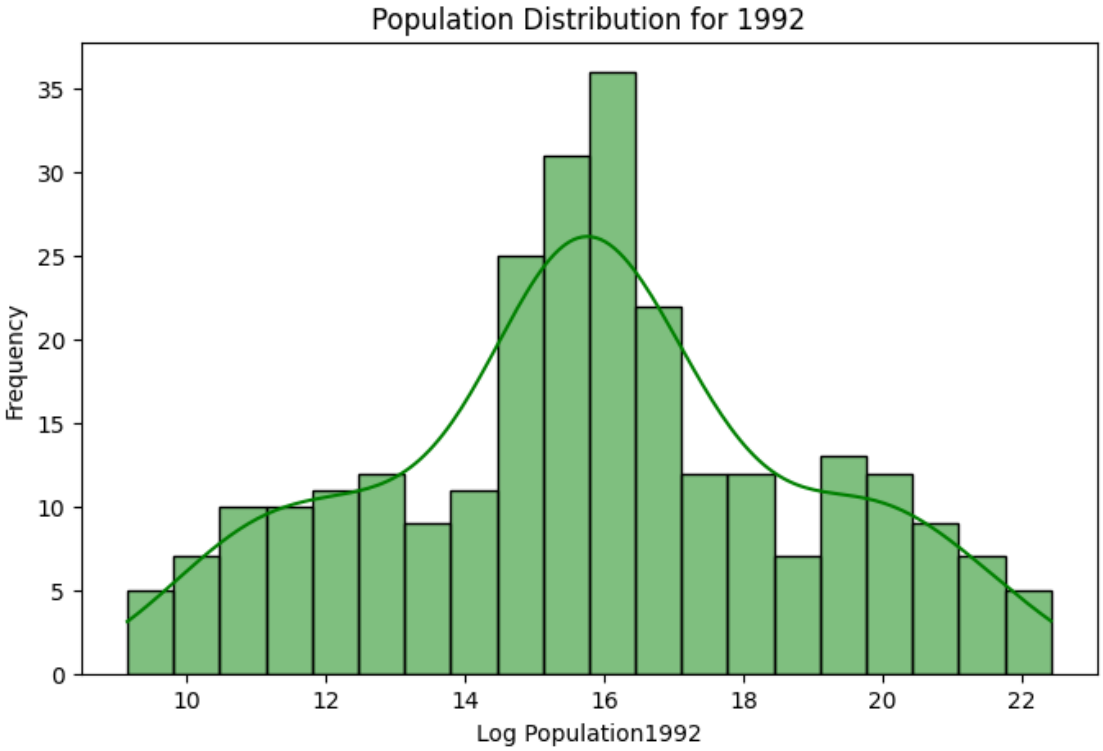

Population Distribution for 1987
Frequency
Log Population1987
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

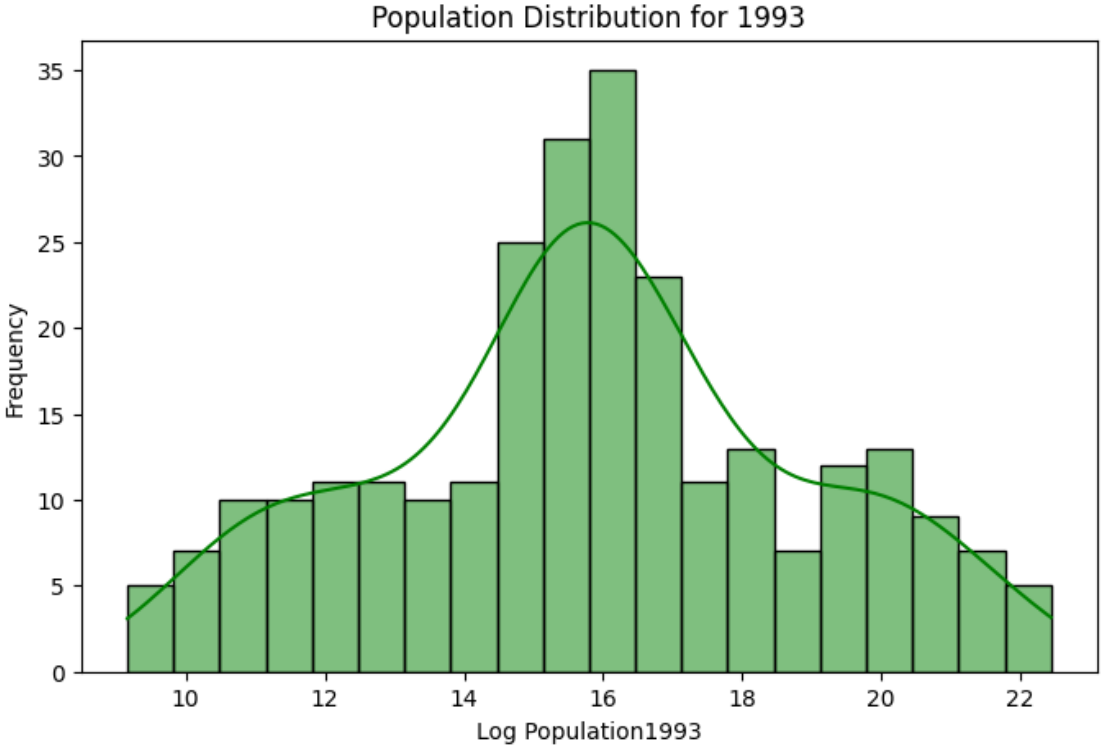

Population Distribution for 1988
Frequency
Log Population1988
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

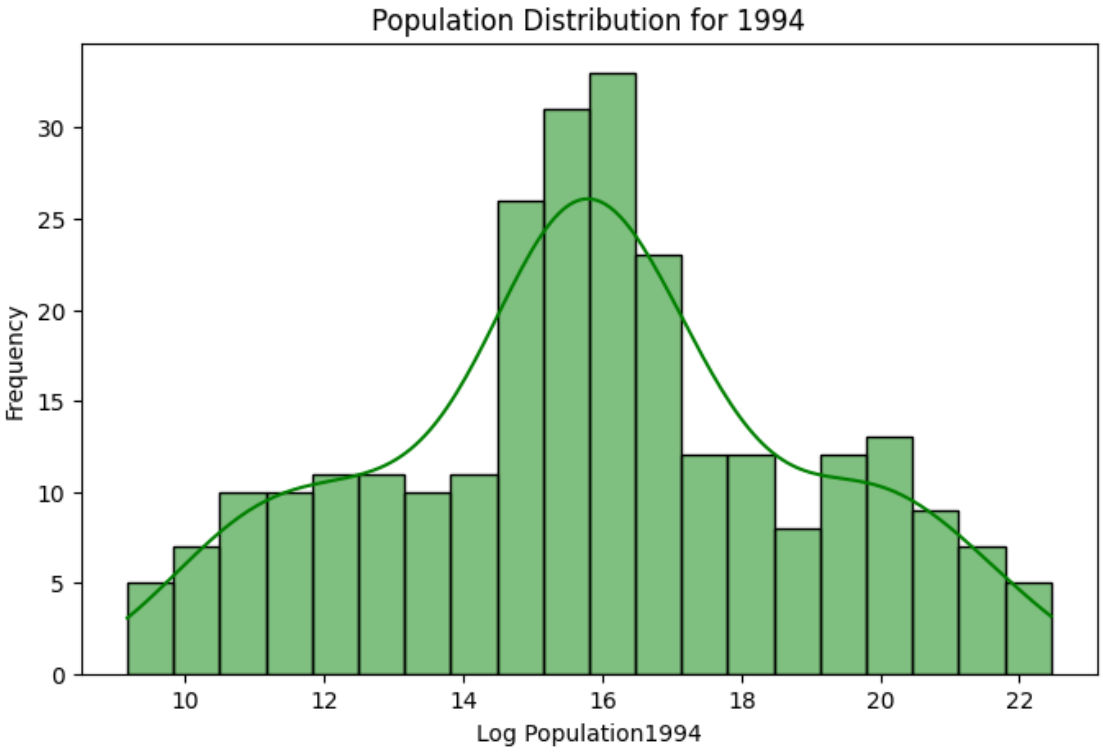

Population Distribution for 1989
Frequency
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22
Log Population1989

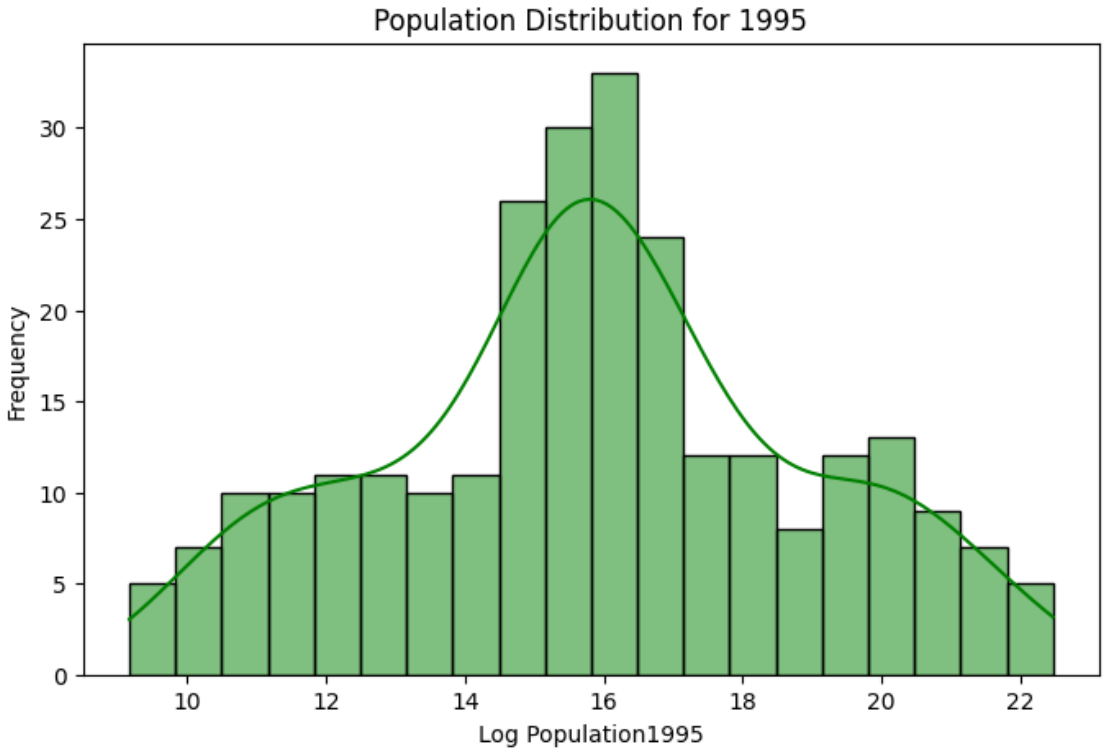

Population Distribution for 1990
Frequency
Log Population1990
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

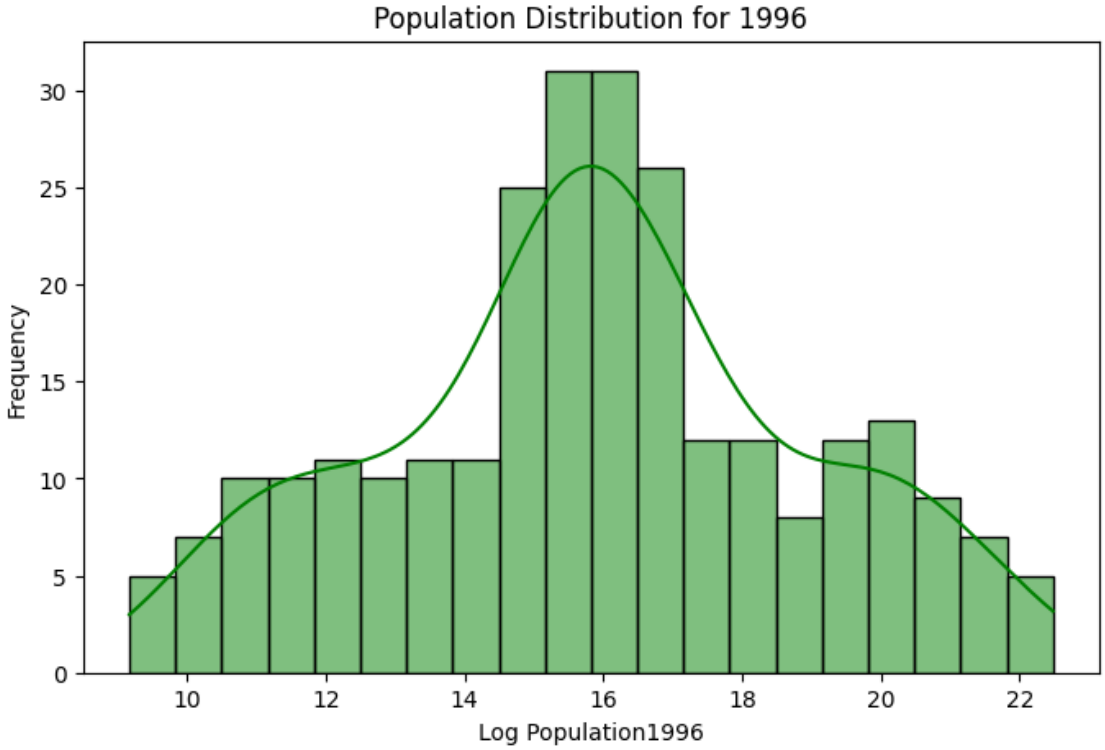

Population Distribution for 1991
Frequency
Log Population1991
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

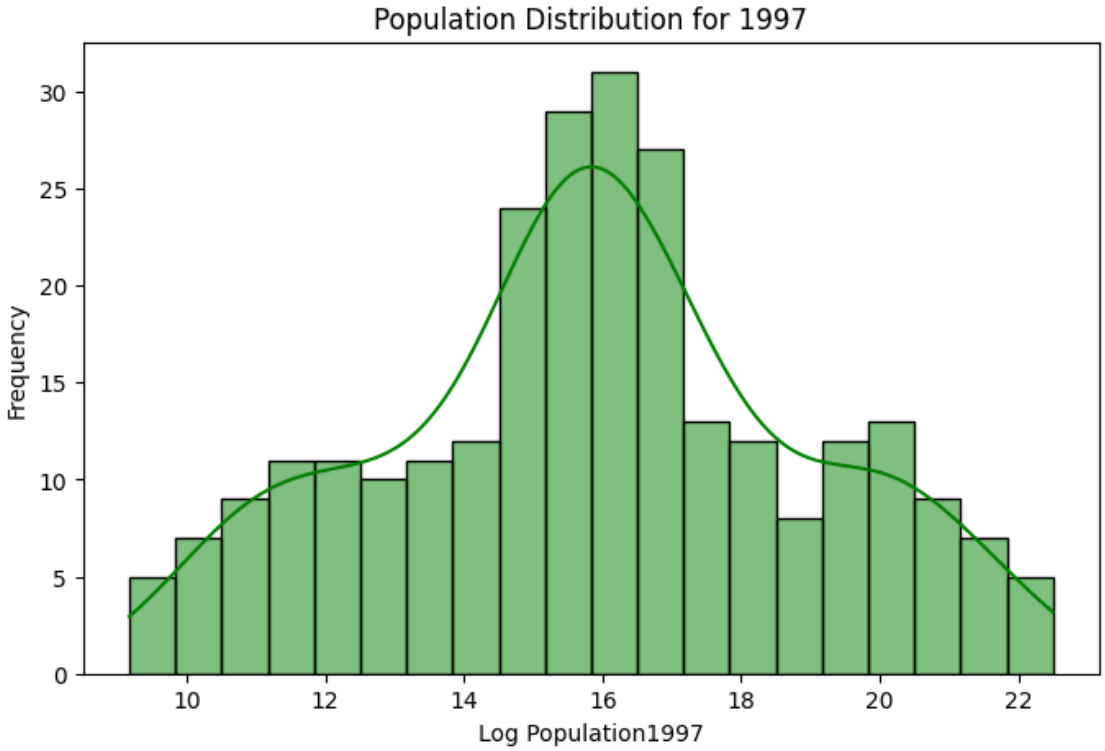

Population Distribution for 1992
Frequency
Log Population1992
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

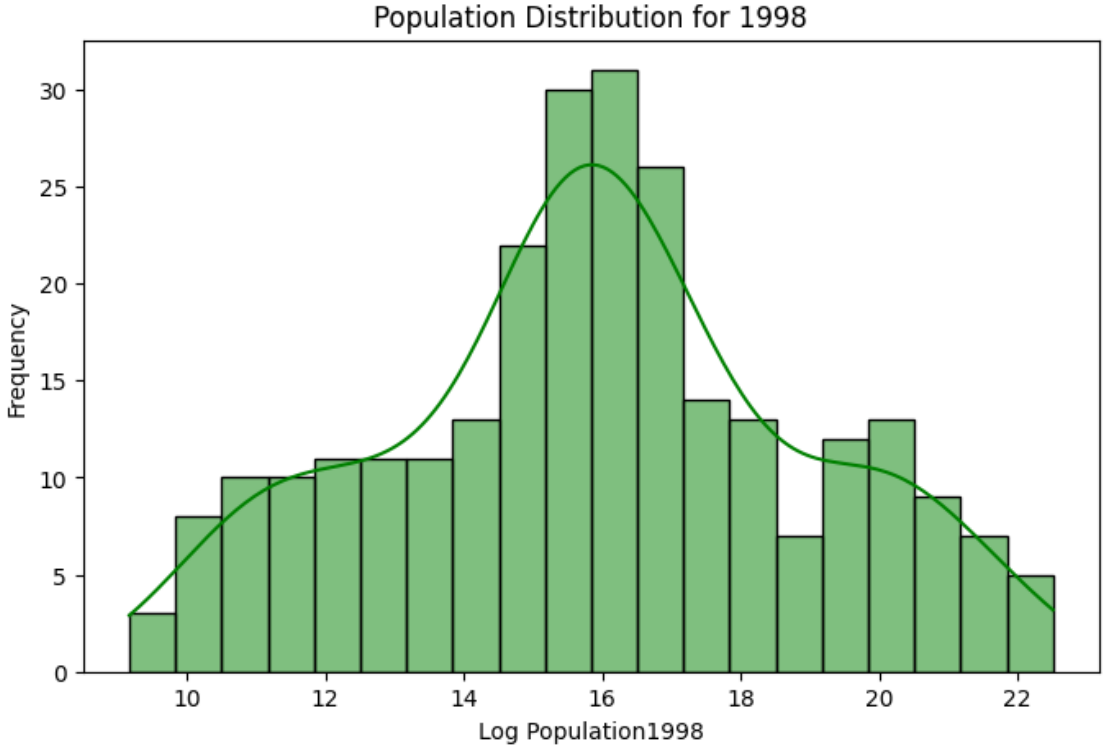

Population Distribution for 1993
Frequency
Log Population1993
0
5
10
15
20
25
30
35
10
12
14
16
18
20
22

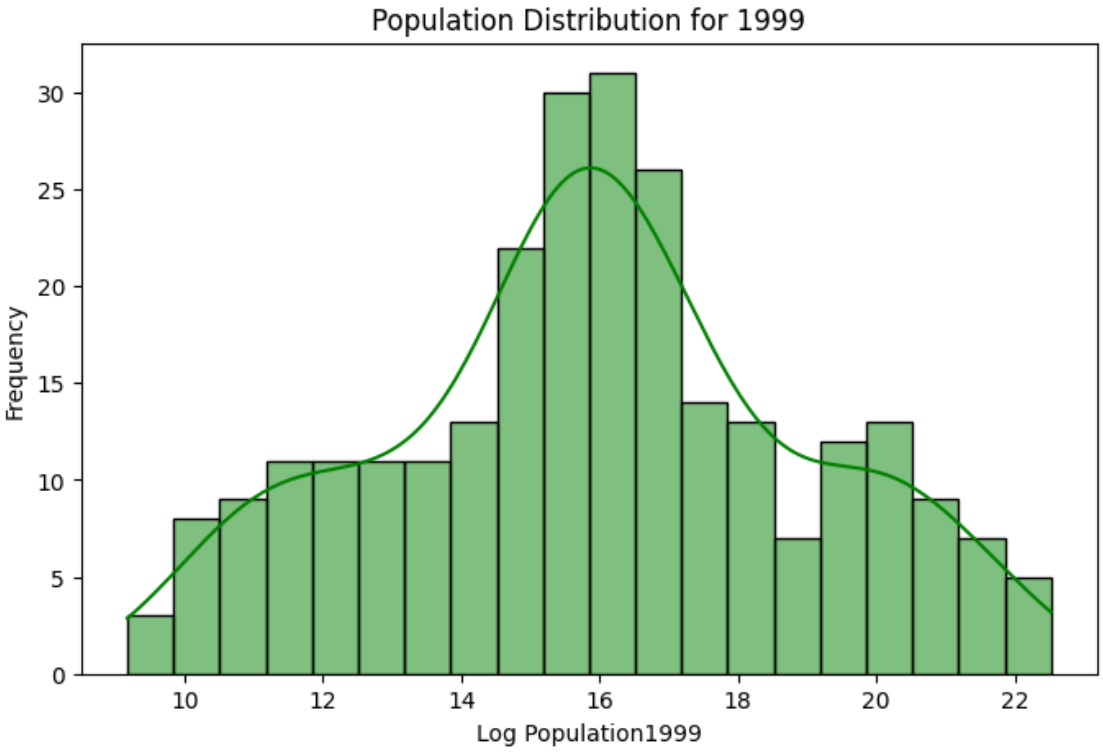

Population Distribution for 1994
Frequency
0
5
10
15
20
25
30
10
12
14
16
18
20
22
Log Population1994

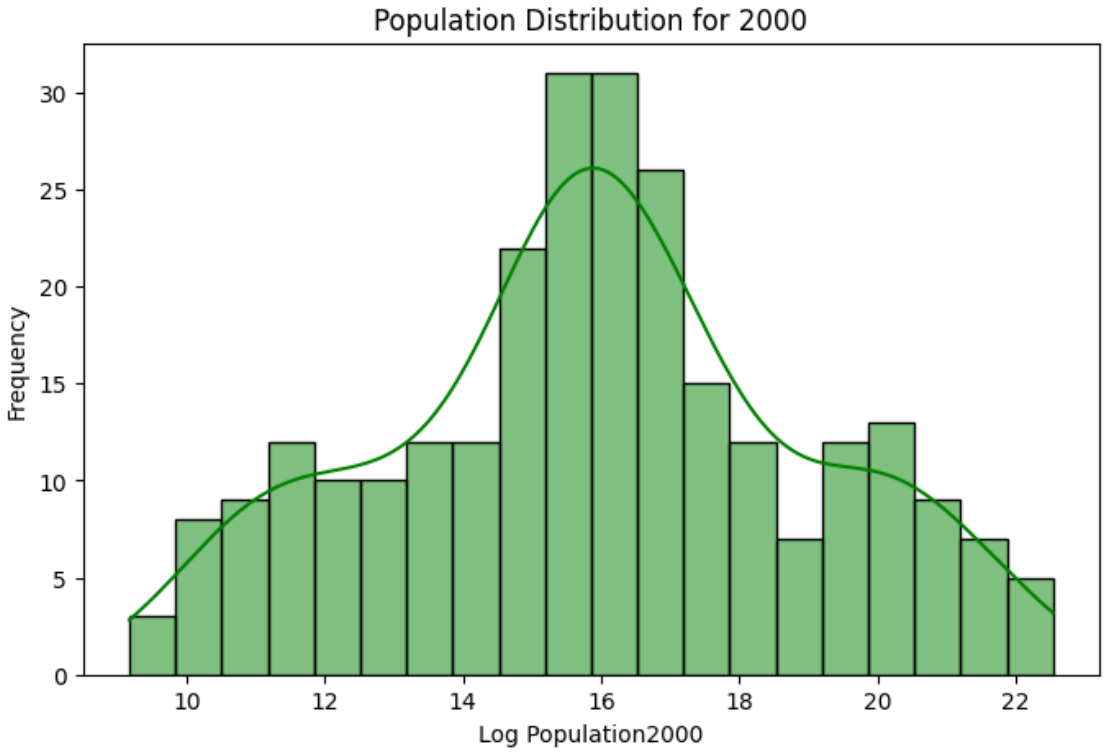

Population Distribution for 1995
Frequency
Log Population1995
0
5
10
15
20
25
30
10
12
14
16
18
20
22

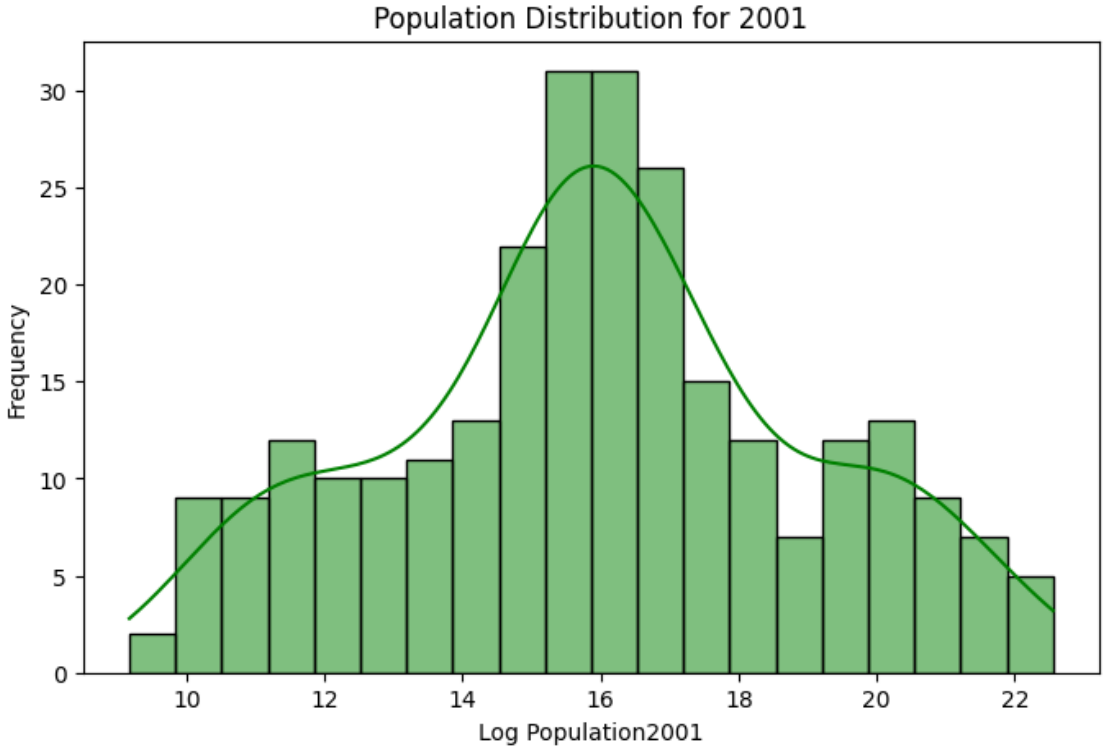

Population Distribution for 1996
Frequency
Log Population1996
0
5
10
15
20
25
30
10
12
14
16
18
20
22

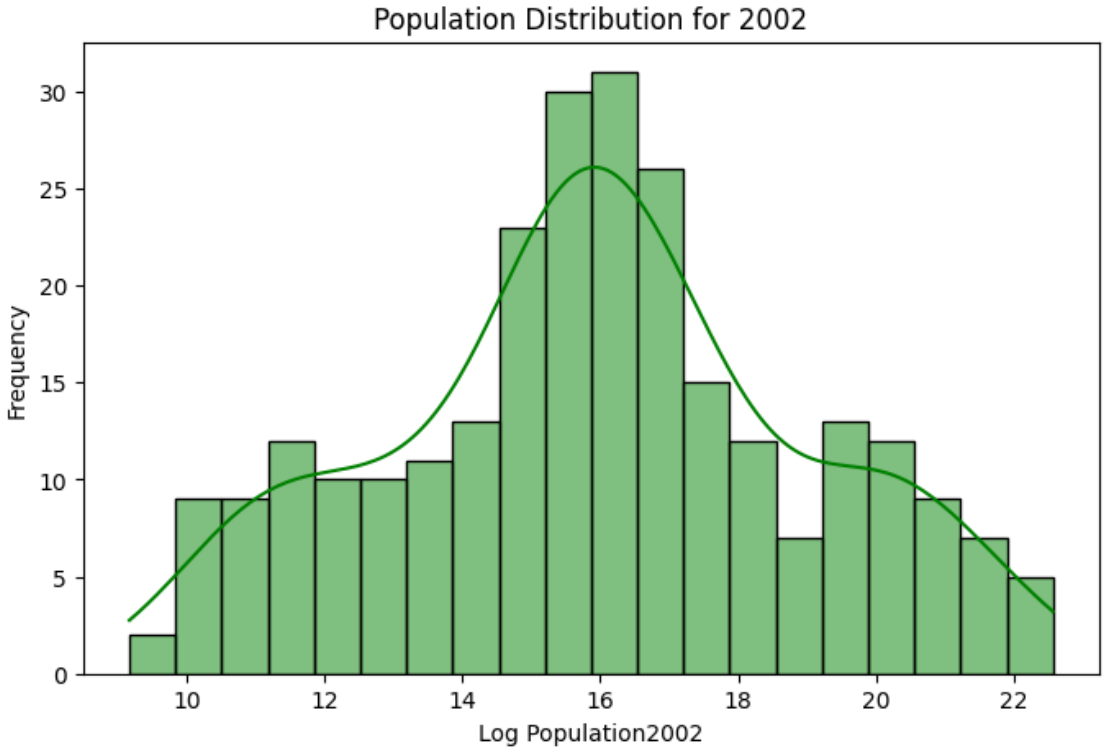

Population Distribution for 1997
Frequency
Log Population1997
0
5
10
15
20
25
30
10
12
14
16
18
20
22

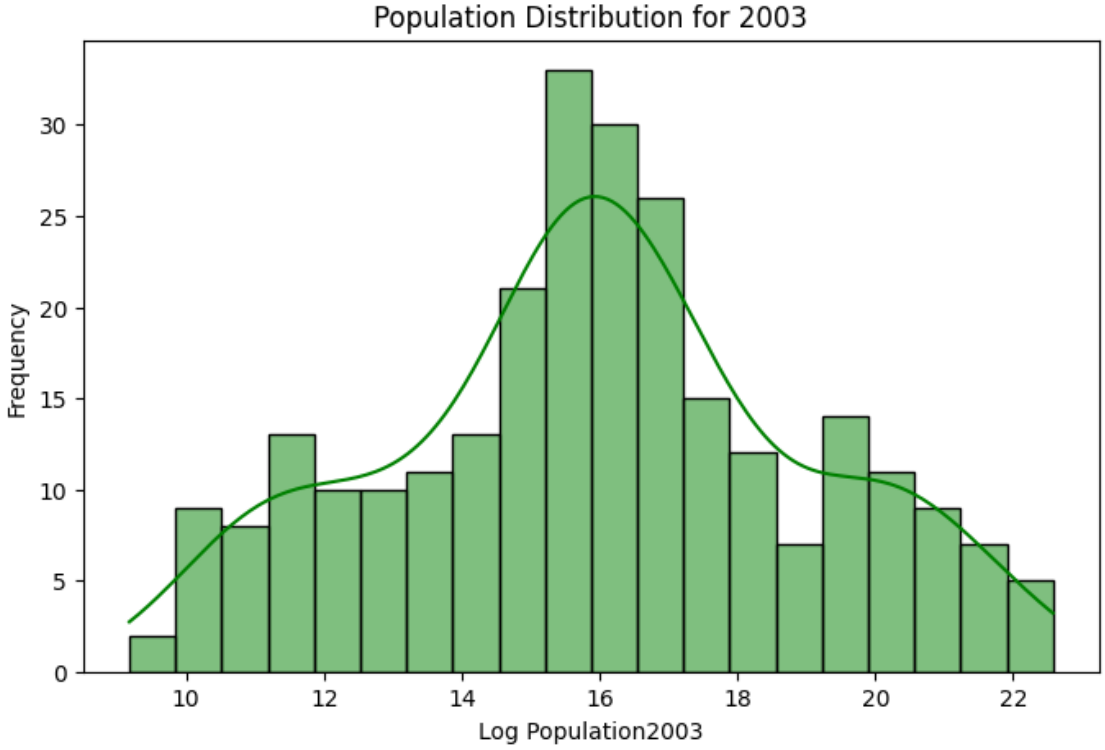

Population Distribution for 1998
Frequency
Log Population1998
0
5
10
15
20
25
30
10
12
14
16
18
20
22

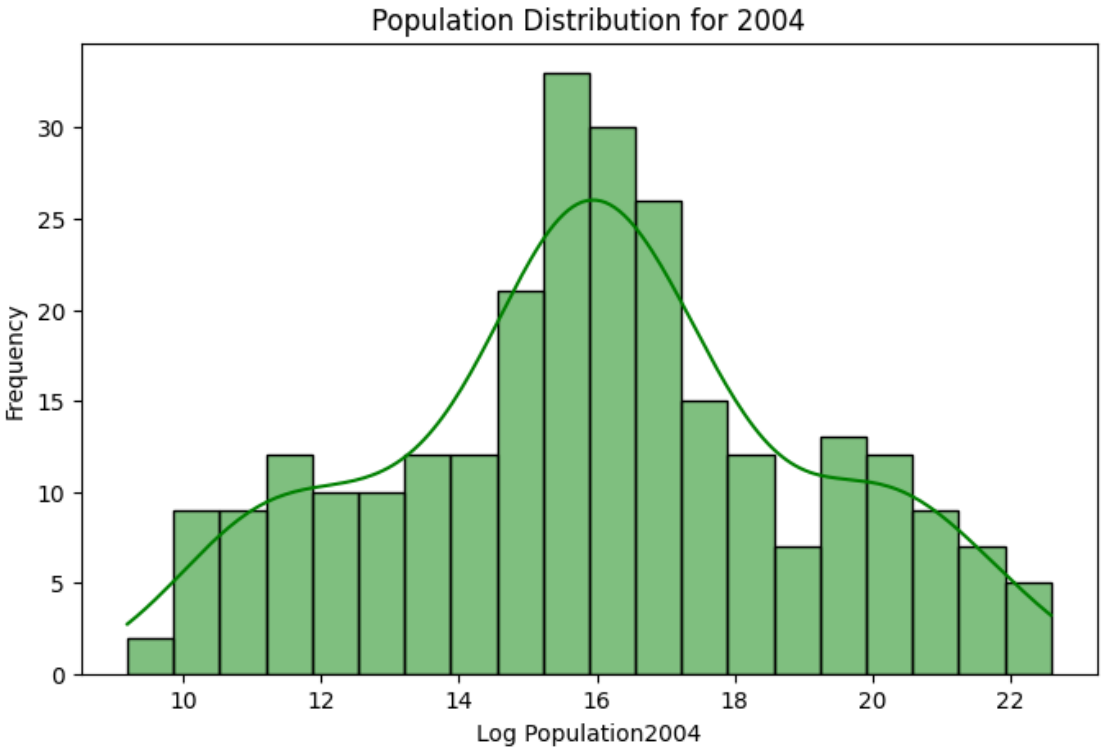

Population Distribution for 1999
Frequency
0
5
10
15
20
25
30
10
12
14
16
18
20
22
Log Population1999

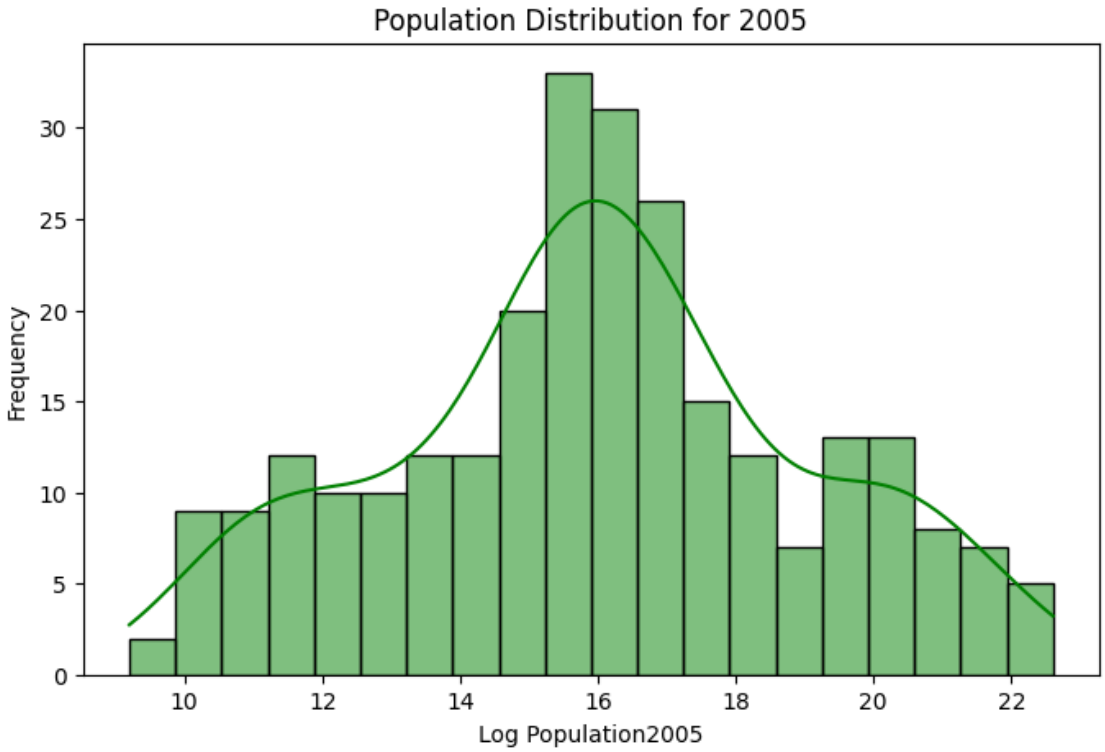

Population Distribution for 2000
Frequency
Log Population2000
0
5
10
15
20
25
30
10
12
14
16
18
20
22

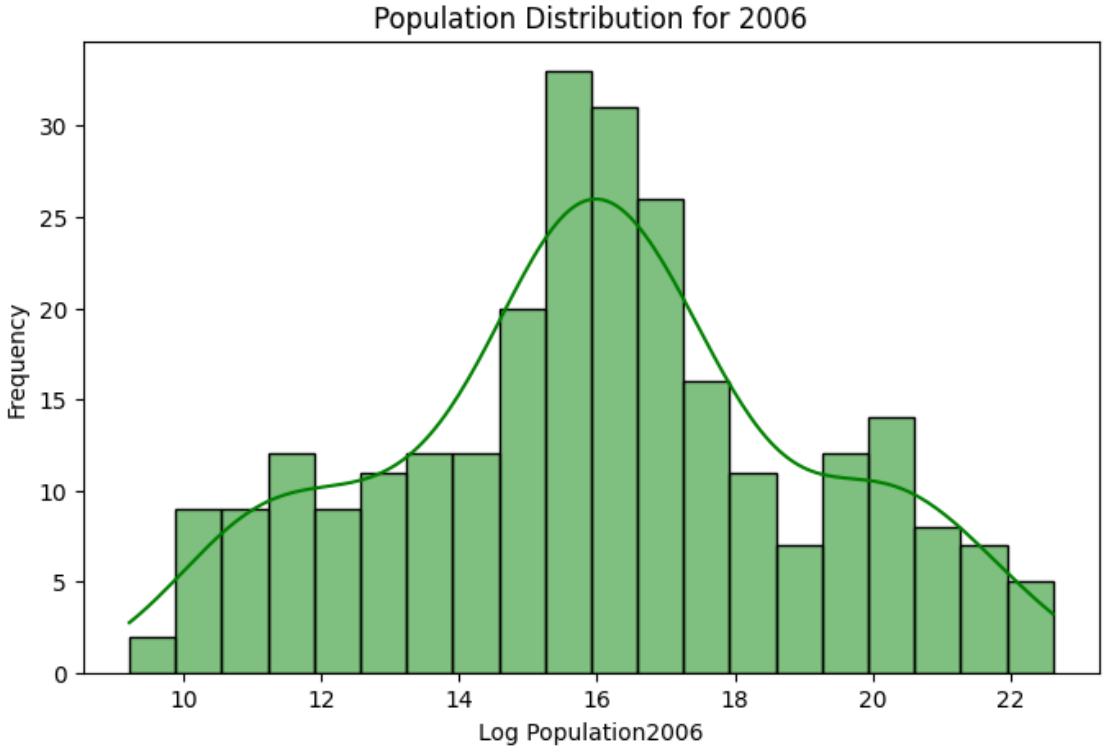

Population Distribution for 2001
Frequency
Log Population2001
0
5
10
15
20
25
30
10
12
14
16
18
20
22

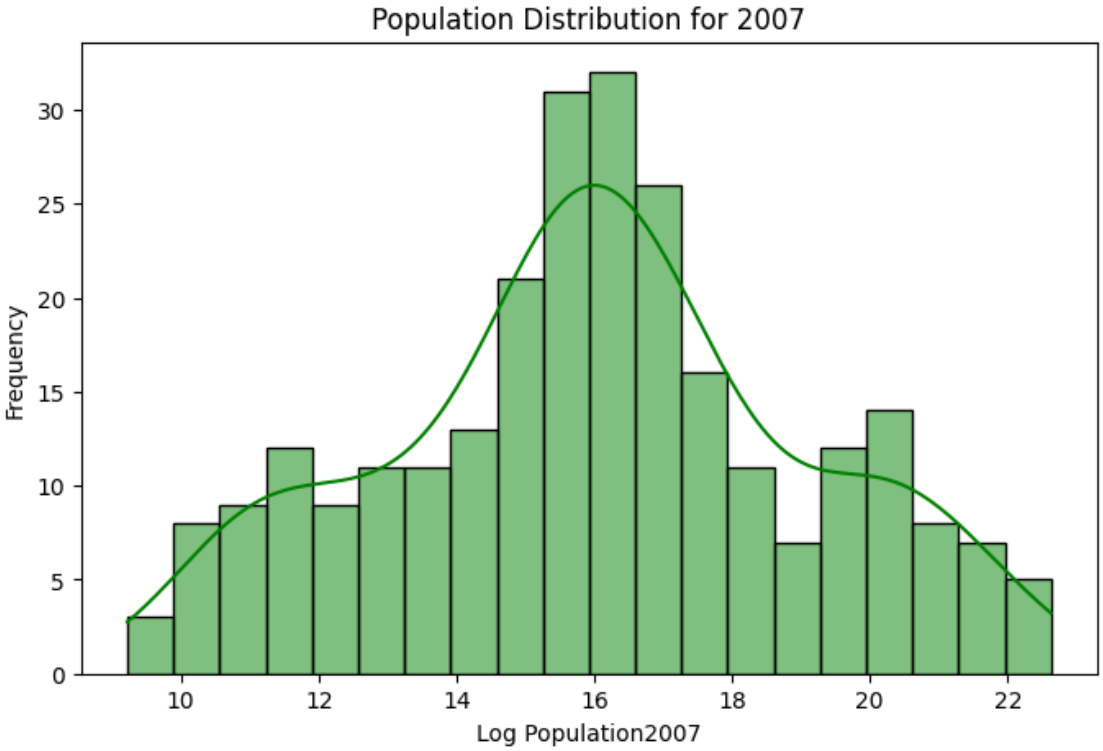

Population Distribution for 2002
Frequency
Log Population2002
0
5
10
15
20
25
30
10
12
14
16
18
20
22

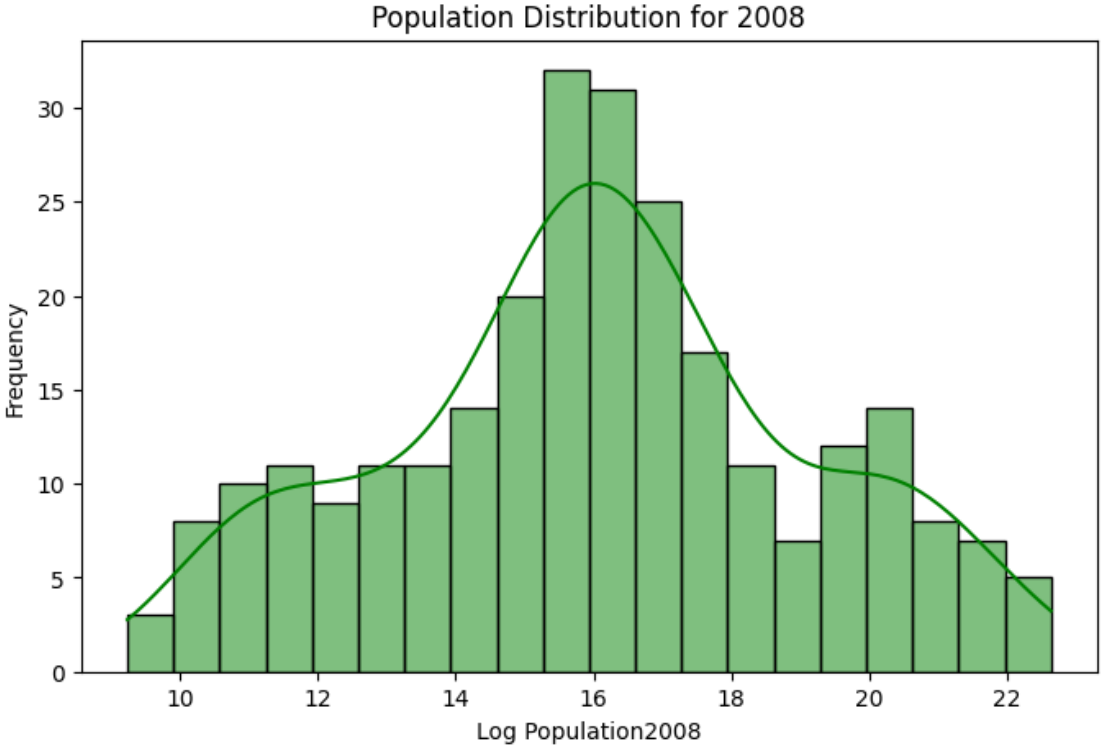

Population Distribution for 2003
Frequency
Log Population2003
0
5
10
15
20
25
30
10
12
14
16
18
20
22

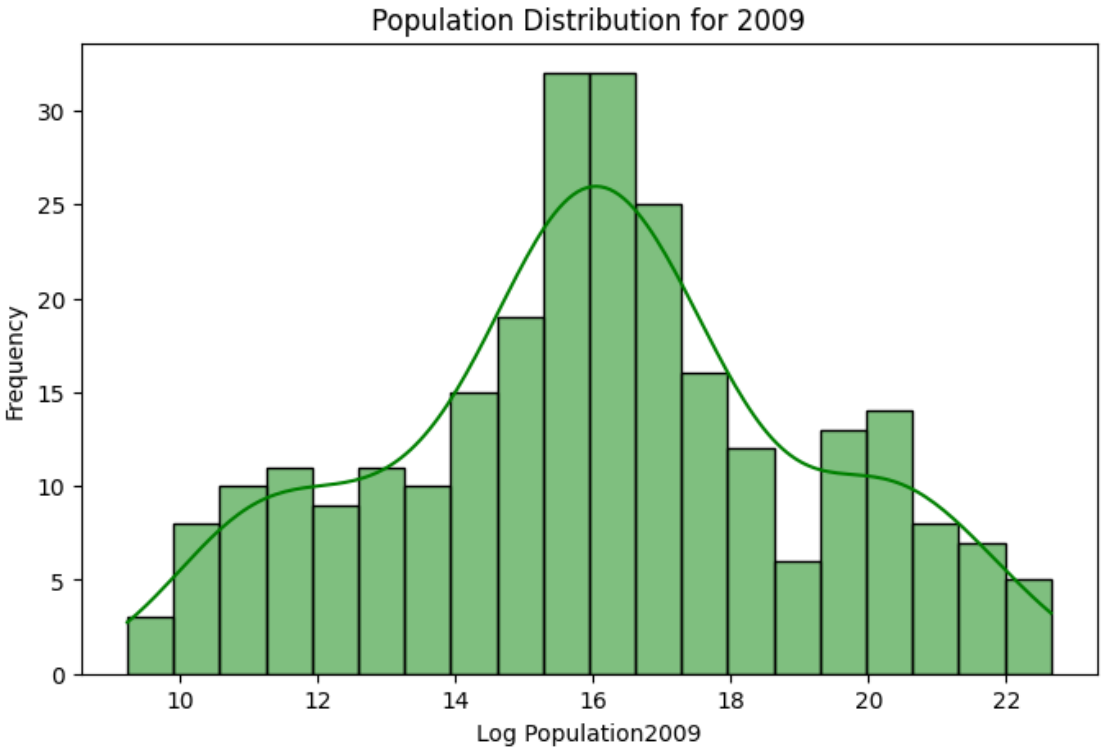

Population Distribution for 2004
Frequency
0
5
10
15
20
25
30
10
12
14
16
18
20
22
Log Population2004

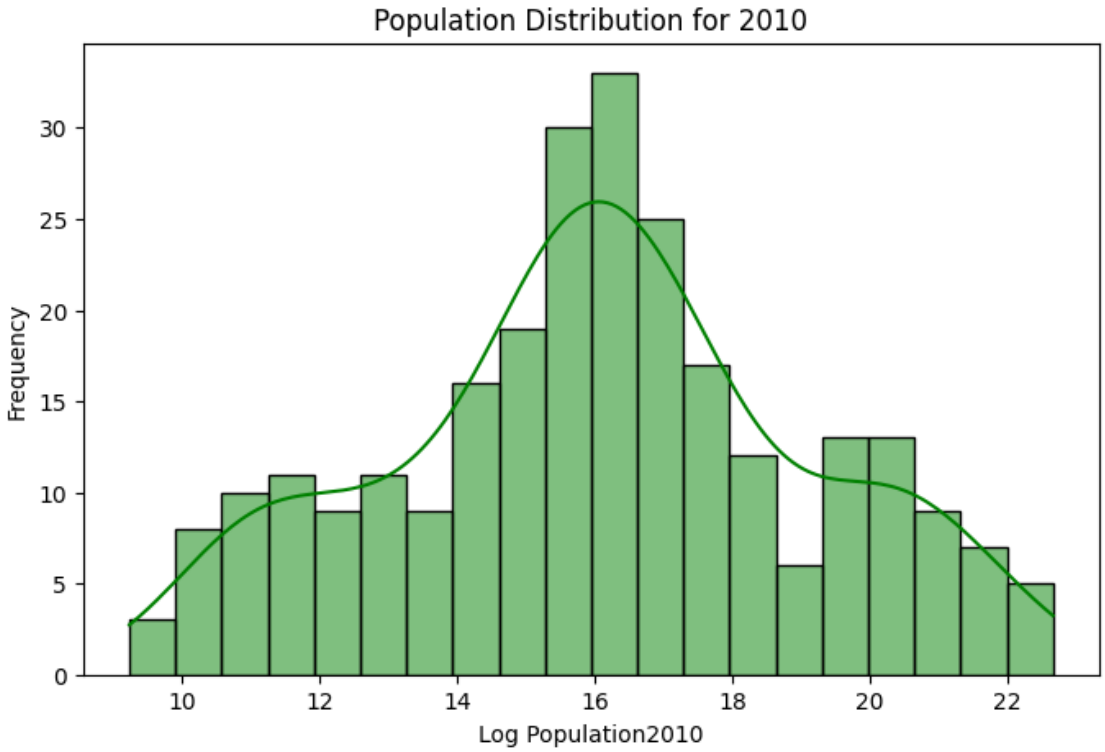

Population Distribution for 2005
Frequency
0
5
10
15
20
25
30
10
12
14
16
18
20
22
Log Population2005

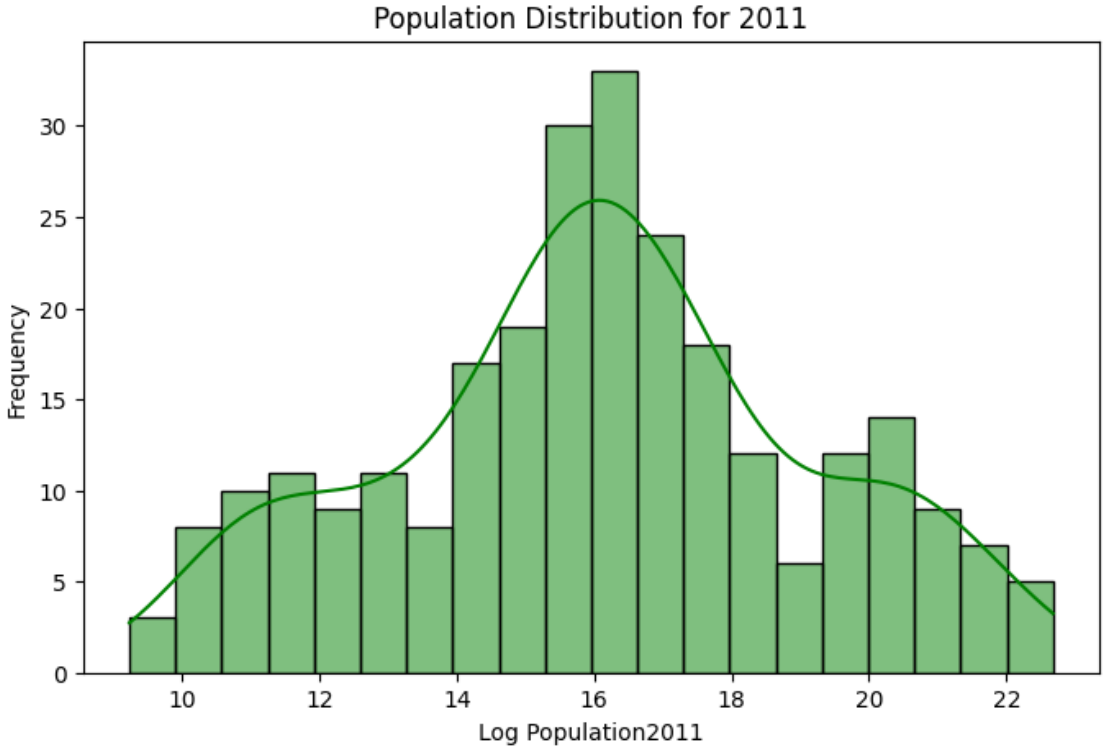

Population Distribution for 2006
Frequency
0
5
10
15
20
25
30
10
12
14
16
18
20
22
Log Population2006

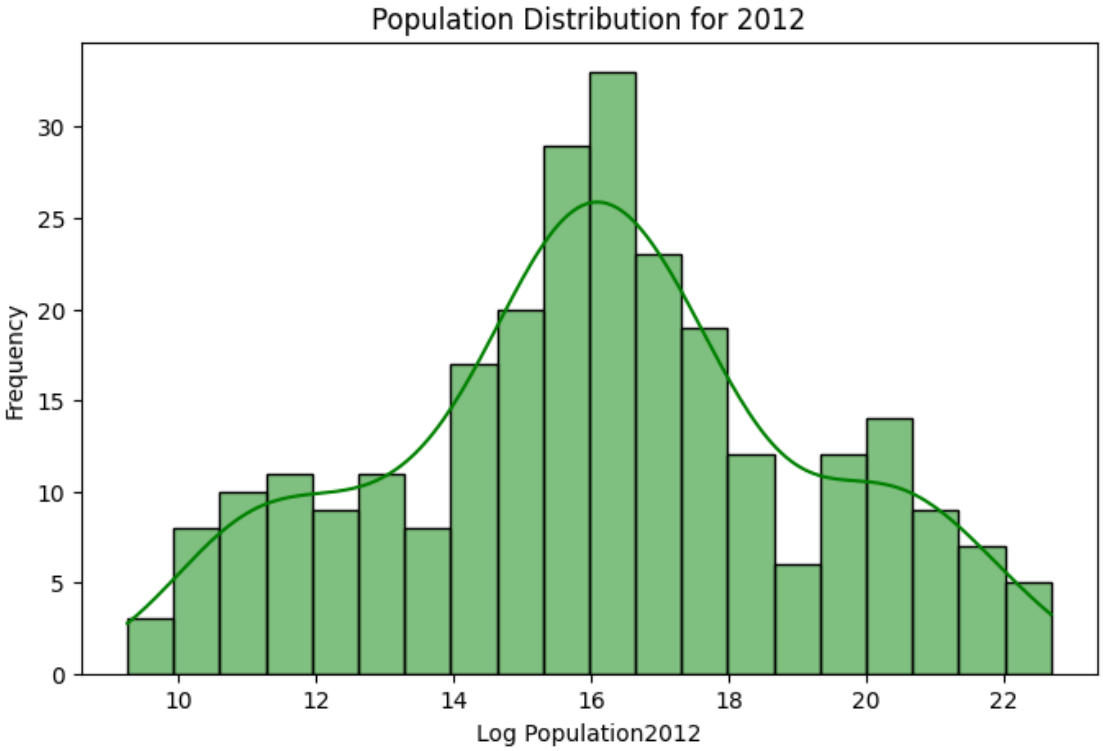

Population Distribution for 2007
Frequency
Log Population2007
0
5
10
15
20
25
30
10
12
14
16
18
20
22

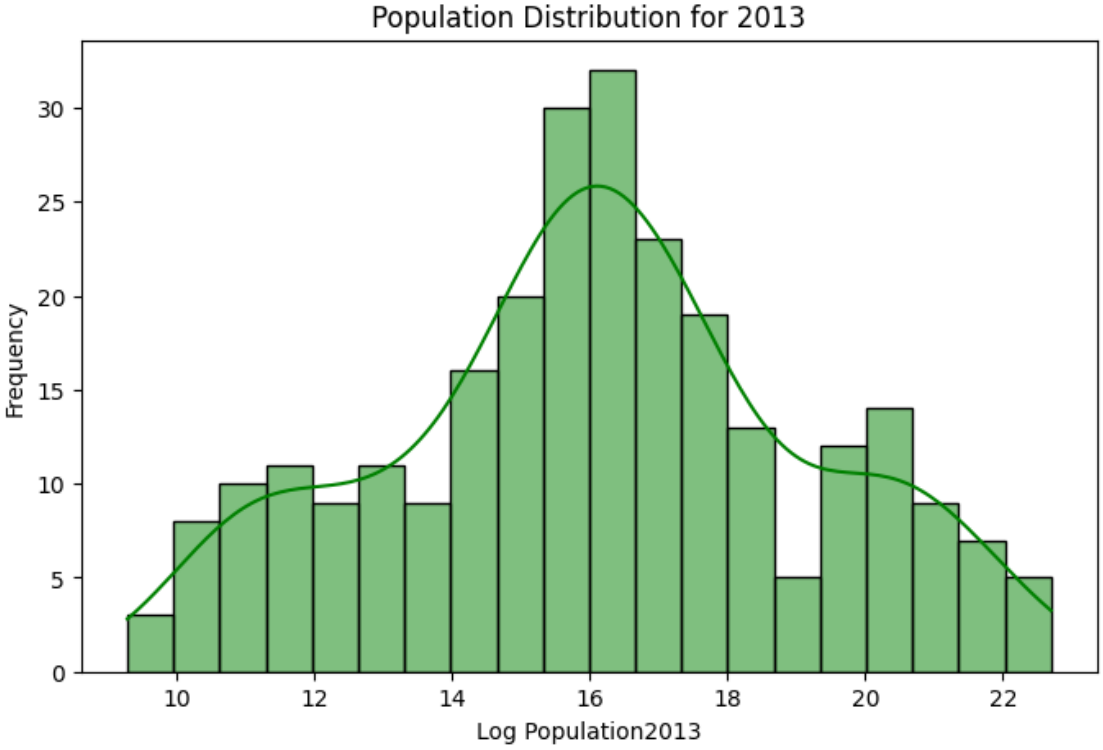

Population Distribution for 2008
Frequency
Log Population2008
0
5
10
15
20
25
30
10
12
14
16
18
20
22

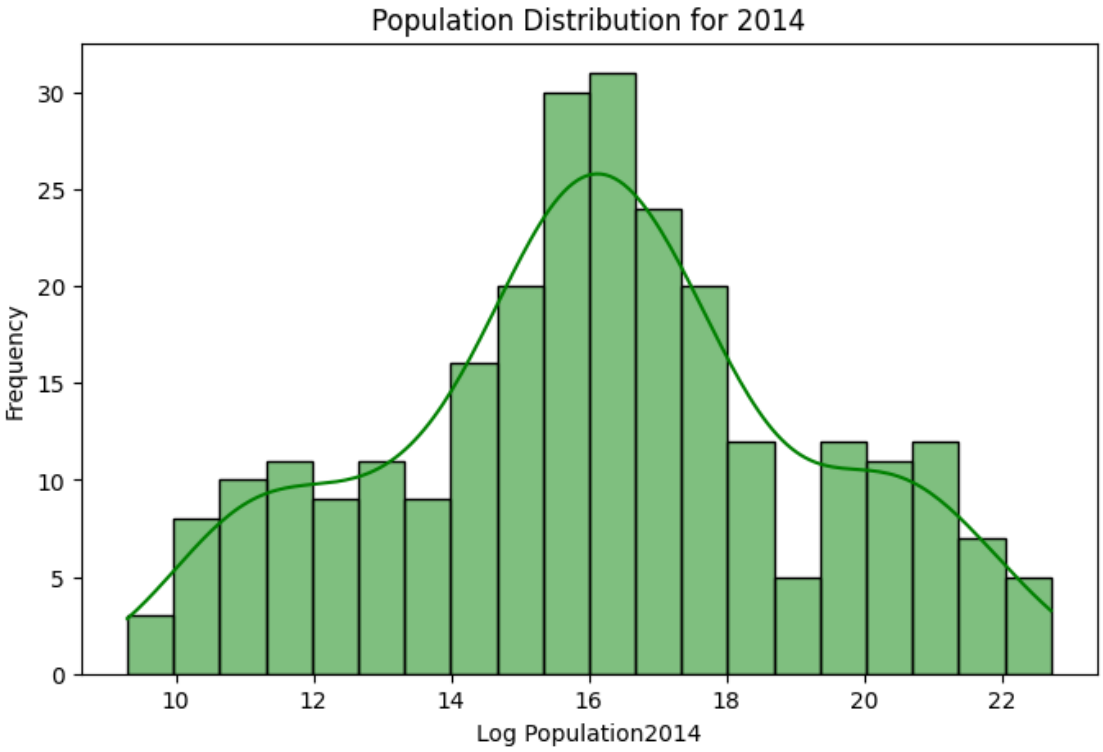

Population Distribution for 2009
Frequency
Log Population2009
0
5
10
15
20
25
30
10
12
14
16
18
20
22

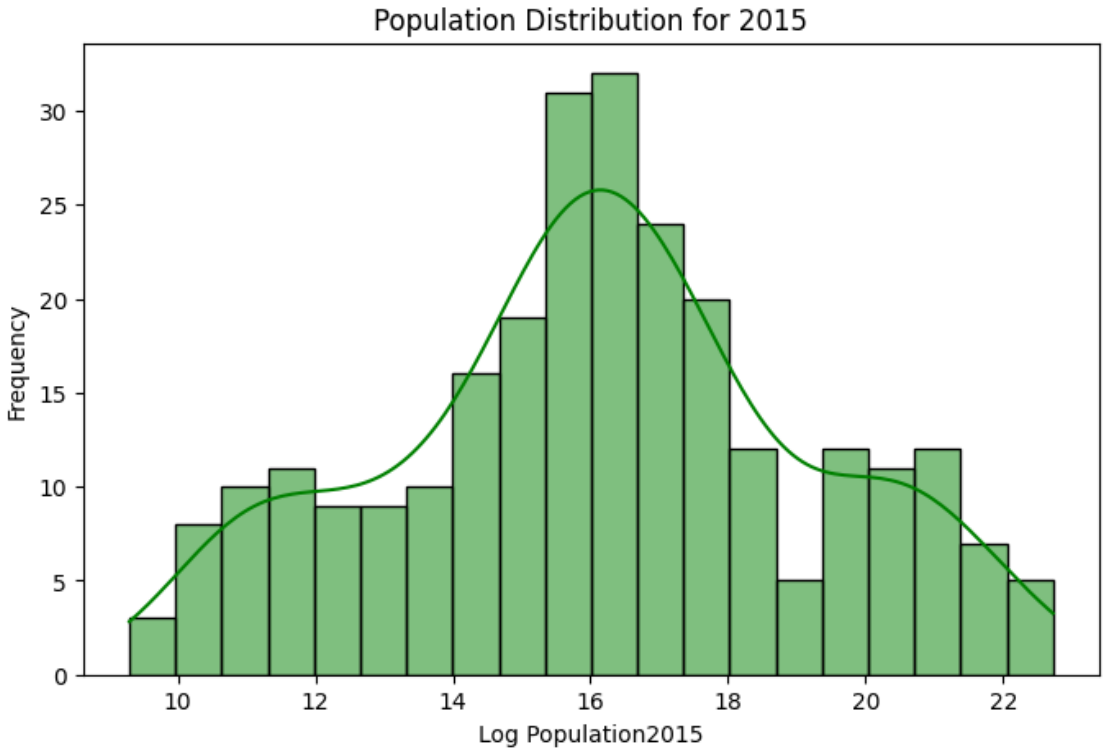

Population Distribution for 2010
Frequency
Log Population2010
0
5
10
15
20
25
30
10
12
14
16
18
20
22

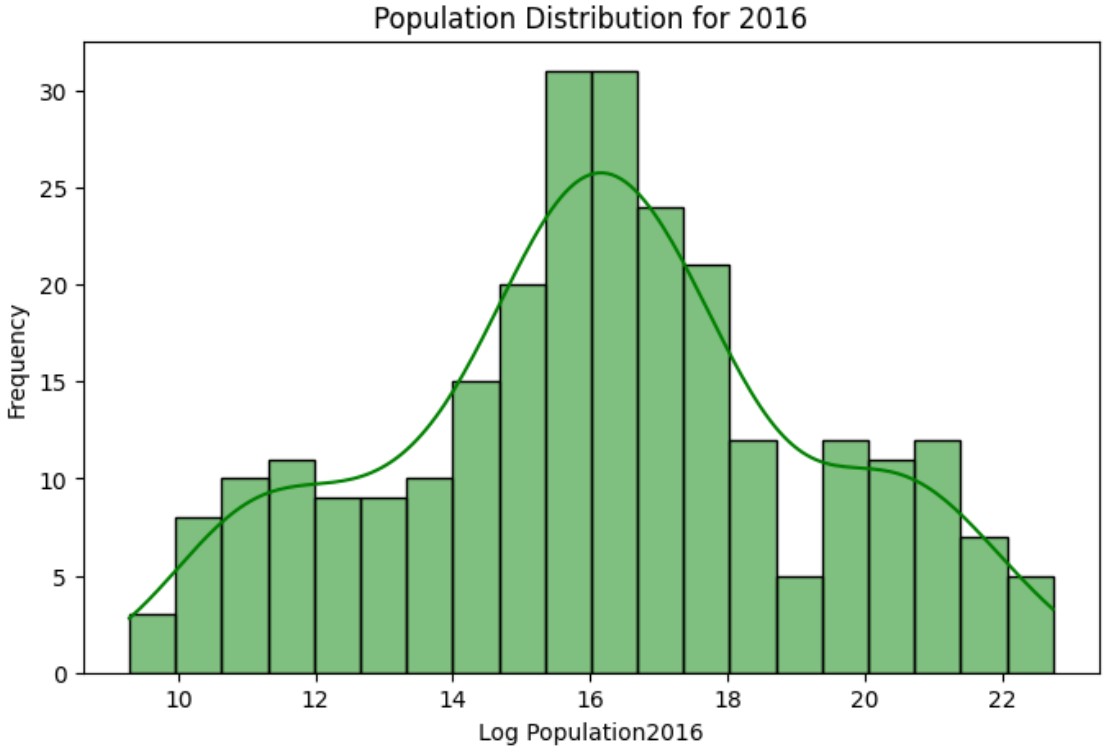

Population Distribution for 2011
Frequency
Log Population2011
0
5
10
15
20
25
30
10
12
14
16
18
20
22

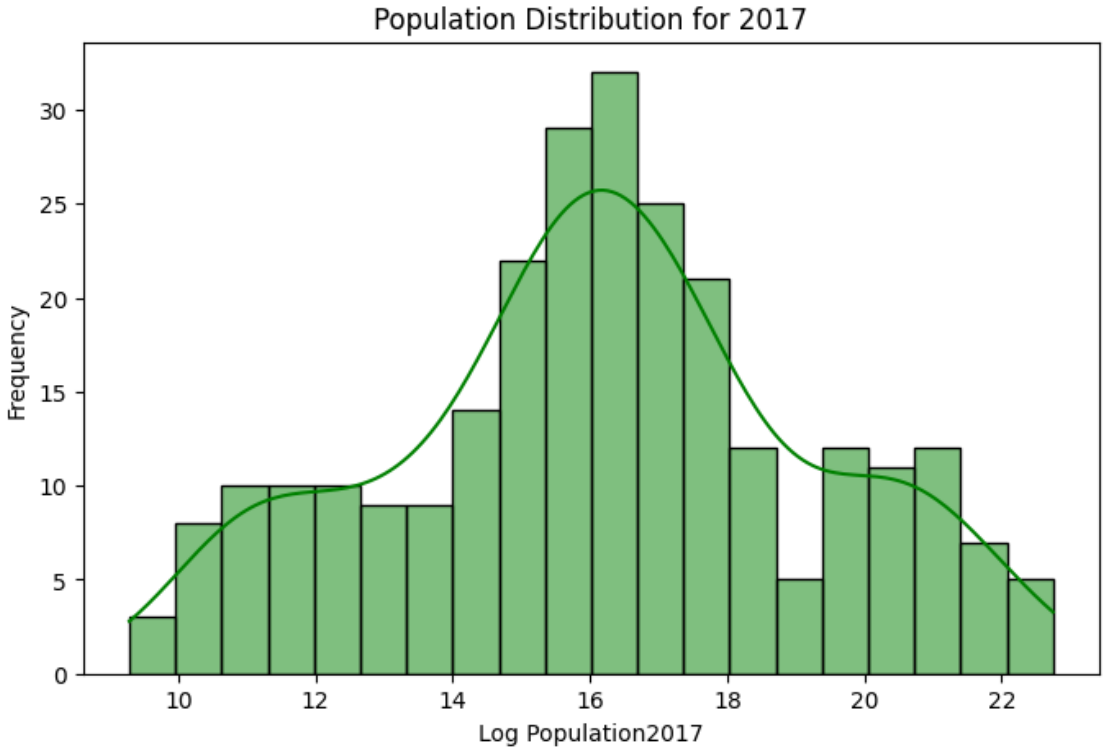

Population Distribution for 2012
Frequency
Log Population2012
0
5
10
15
20
25
30
10
12
14
16
18
20
22

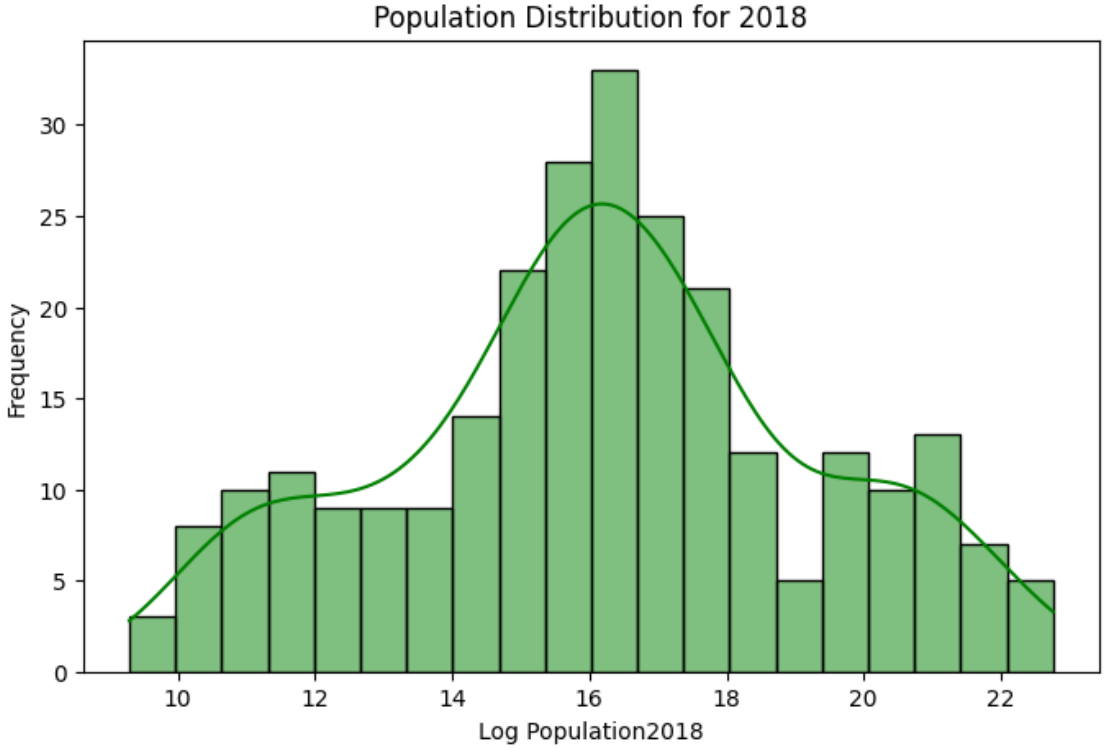

Population Distribution for 2013
Frequency
Log Population2013
0
5
10
15
20
25
30
10
12
14
16
18
20
22

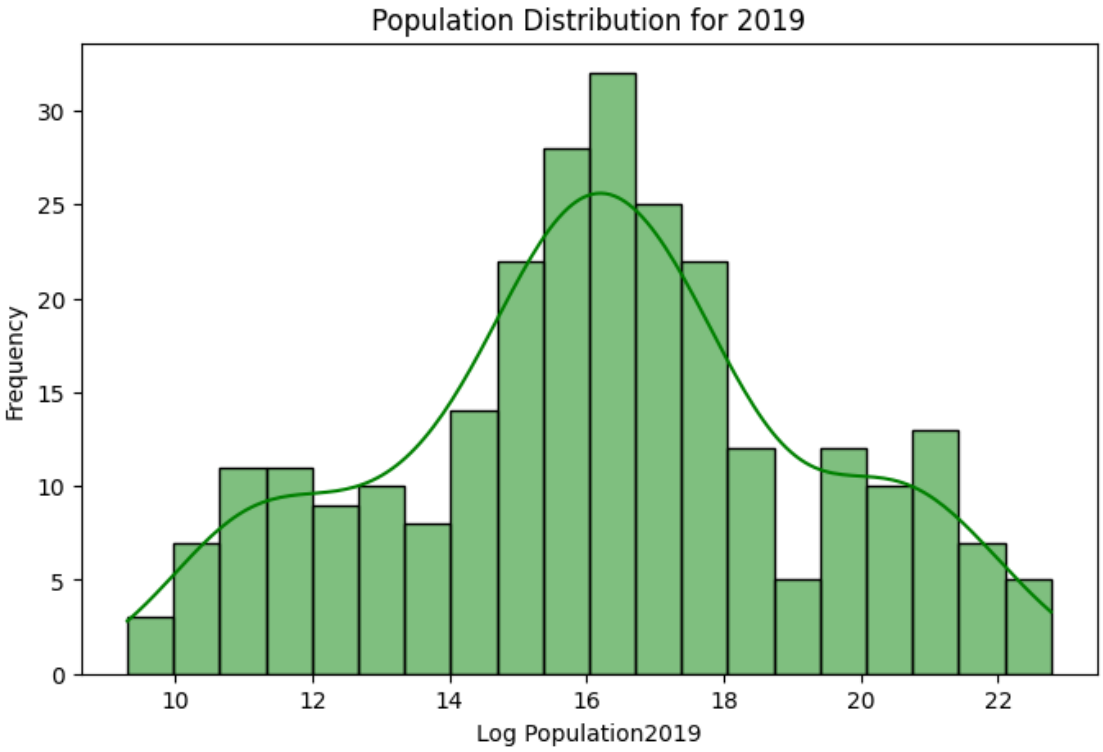

Population Distribution for 2014
Frequency
Log Population2014
0
5
10
15
20
25
30
10
12
14
16
18
20
22

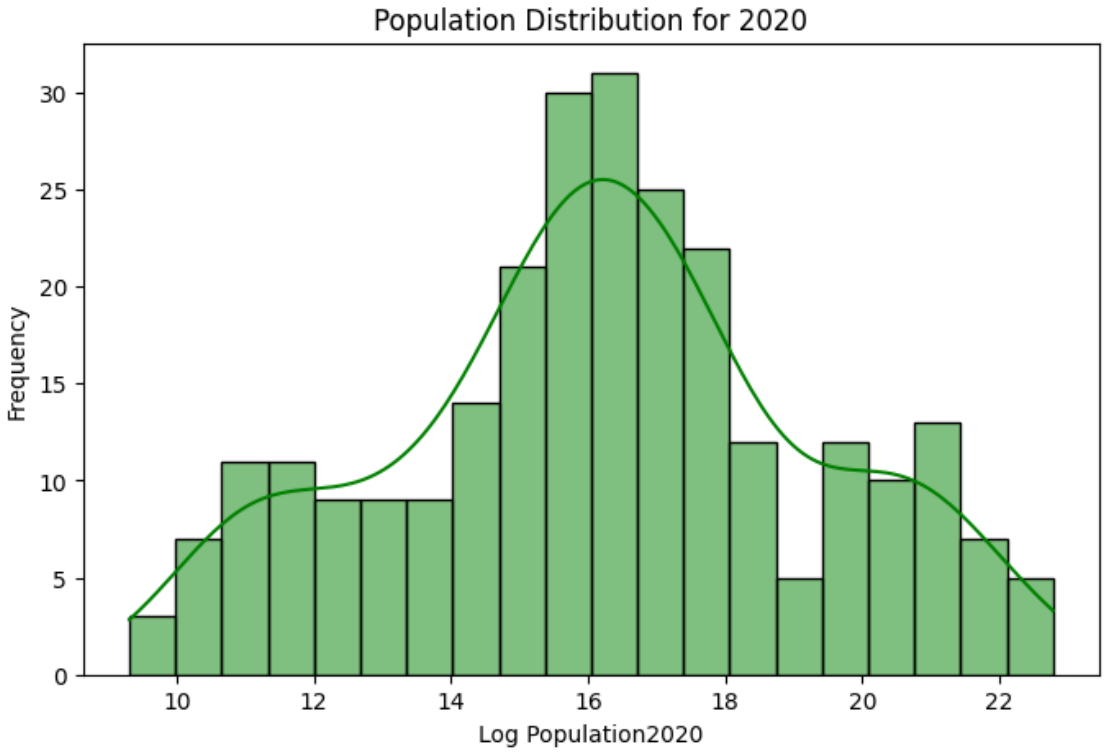

Population Distribution for 2015
Frequency
Log Population2015
0
5
10
15
20
25
30
10
12
14
16
18
20
22

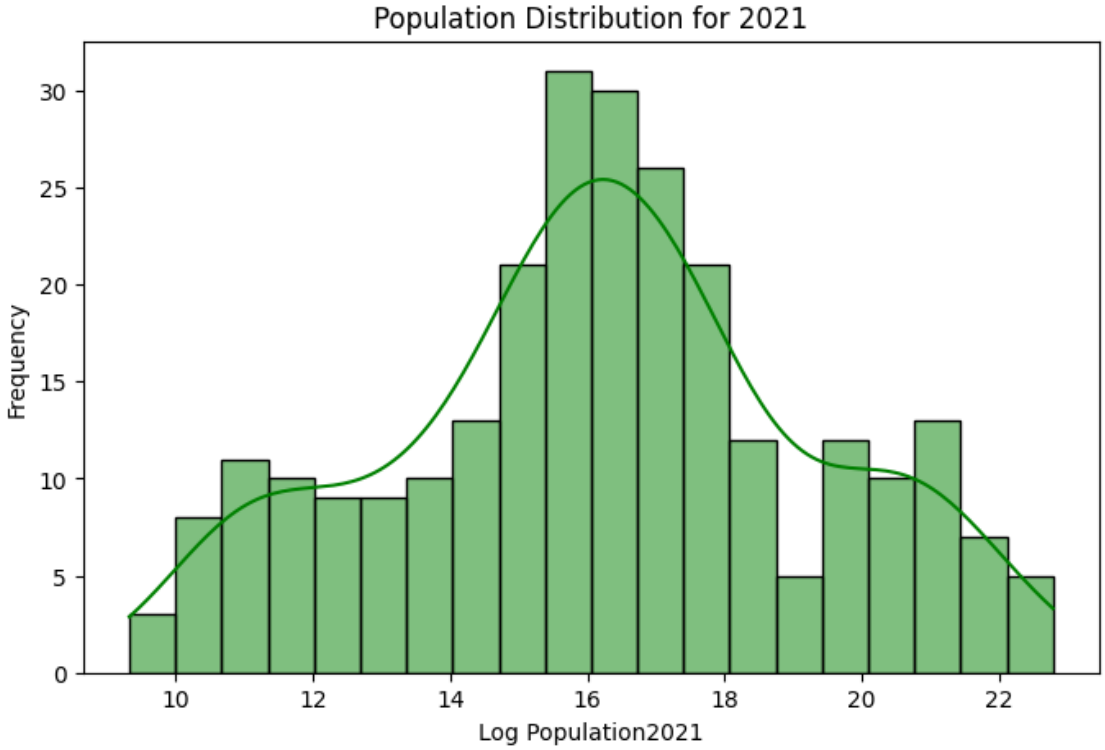

Population Distribution for 2016
Frequency
Log Population2016
0
5
10
15
20
25
30
10
12
14
16
18
20
22

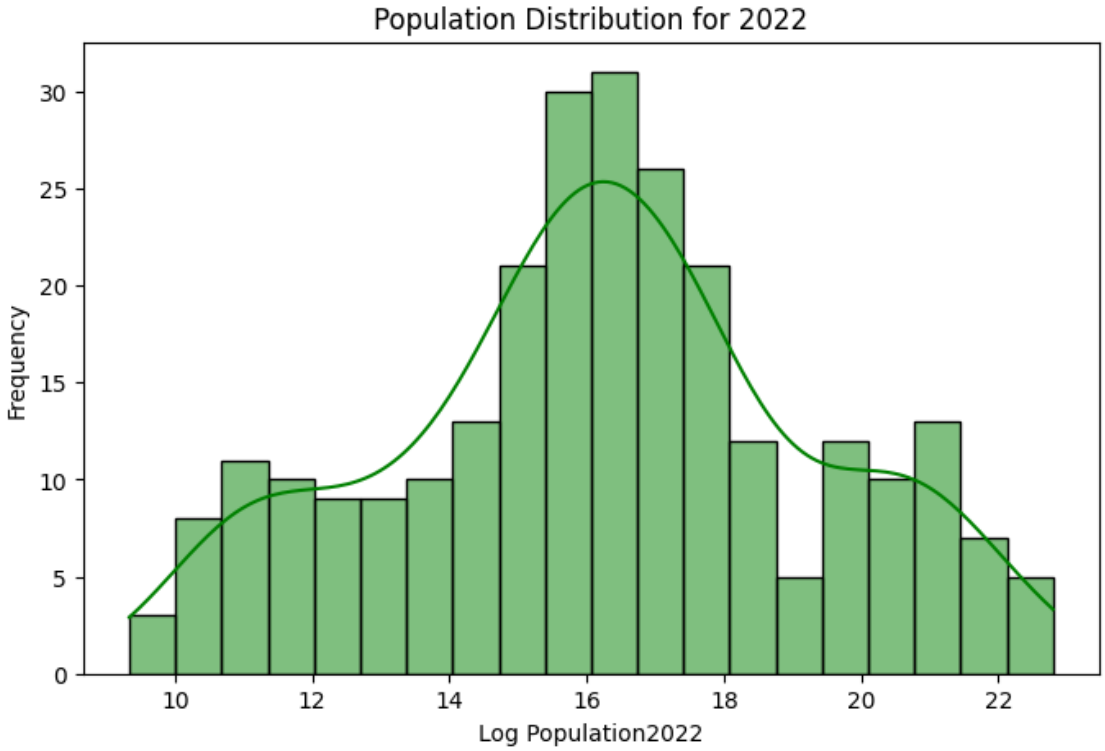

Population Distribution for 2017
Frequency
Log Population2017
0
5
10
15
20
25
30
10
12
14
16
18
20
22


Population Distribution for 2018
Frequency
Log Population2018
0
5
10
15
20
25
30
10
12
14
16
18
20
22


Population Distribution for 2019
Frequency
Log Population2019
0
5
10
15
20
25
30
10
12
14
16
18
20
22


Population Distribution for 2020
Frequency
Log Population2020
0
5
10
15
20
25
30
10
12
14
16
18
20
22


Population Distribution for 2021
Frequency
Log Population2021
0
5
10
15
20
25
30
10
12
14
16
18
20
22



[144]:
```
df.hist(figsize=(20,20))
sns.despine()
```

5 Visualization - Scatter plot for a specific country

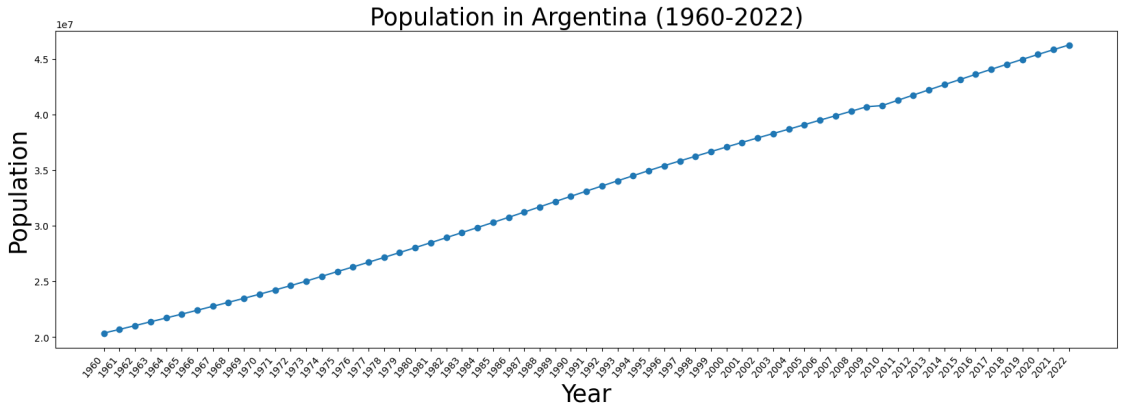
[145]:
```
# Scatter plot for a specific country
def specific_country_scatterpolt(df,country_index_number):
  country=df['Country Name'][country_index_number]
  plt.figure(figsize=(20,6))
  years=df.columns[4:].to_list()
  population =df.iloc[country_index_number,4:]
  plt.plot(years, population, marker='o', linestyle='-')
  plt.xlabel('Year',fontsize=25)
  plt.ylabel('Population',fontsize=25)
```

```
  plt.title(f'Population in {country} (1960-2022)',fontsize=25)
  plt.xticks(rotation=50, ha='right')
```

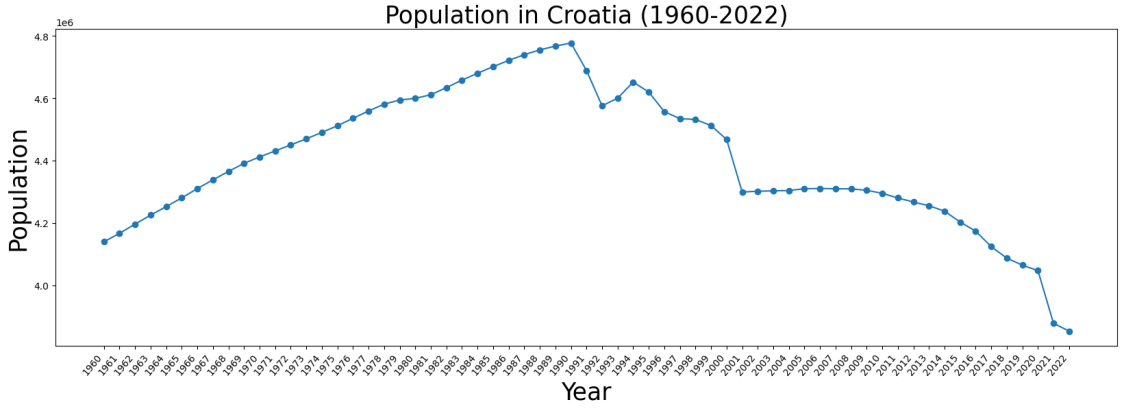
[146]:
```
df['Country Name'][88]
```

[146]:
```
'Equatorial Guinea'
```

[147]:
```
specific_country_scatterpolt(df,9)
```



[148]:
```
specific_country_scatterpolt(df,99)
```



[149]:
```
# BY CHANDRASEKAR
```

[150]:
```
# Happy coding
```