In [3]:
```python
import pandas as pd

# Load the dataset
data = pd.read_csv('world_population.csv')


# Display the first few rows of the dataset to understand its structure
data.head(), data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 17 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Rank                          234 non-null    int64  
 1   CCA3                          234 non-null    object 
 2   Country/Territory            234 non-null    object 
 3   Capital                       234 non-null    object 
 4   Continent                     234 non-null    object 
 5   2022 Population               234 non-null    int64  
 6   2020 Population               234 non-null    int64  
 7   2015 Population               234 non-null    int64  
 8   2010 Population               234 non-null    int64  
 9   2000 Population               234 non-null    int64  
 10  1990 Population               234 non-null    int64  
 11  1980 Population               234 non-null    int64  
 12  1970 Population               234 non-null    int64  
 13  Area (km²)                    234 non-null    int64  
 14  Density (per km²)             234 non-null    float64
 15  Growth Rate                   234 non-null    float64
 16  World Population Percentage  234 non-null    float64
dtypes: float64(3), int64(10), object(4)
memory usage: 31.2+ KB
```

Loading [MathJax]/extensions/Safe.js

Out[3]:
```
(   Rank CCA3 Country/Territory          Capital Continent  2022 Populatio
n  \
 0    36  AFG       Afghanistan            Kabul      Asia         4112877
1
 1   138  ALB           Albania           Tirana    Europe          284232
1
 2    34  DZA           Algeria          Algiers    Africa         4490322
5
 3   213  ASM    American Samoa         Pago Pago   Oceania            4427
3
 4   203  AND           Andorra  Andorra la Vella    Europe            7982
4

    2020 Population  2015 Population  2010 Population  2000 Population  \
 0         38972230         33753499         28189672         19542982
 1          2866849          2882481          2913399          3182021
 2         43451666         39543154         35856344         30774621
 3            46189            51368            54849            58230
 4            77700            71746            71519            66097

    1990 Population  1980 Population  1970 Population  Area (km²)  \
 0         10694796         12486631         10752971      652230
 1          3295066          2941651          2324731       28748
 2         25518074         18739378         13795915     2381741
 3            47818            32886            27075         199
 4            53569            35611            19860         468

    Density (per km²)  Growth Rate  World Population Percentage
 0            63.0587       1.0257                         0.52
 1            98.8702       0.9957                         0.04
 2            18.8531       1.0164                         0.56
 3           222.4774       0.9831                         0.00
 4           170.5641       1.0100                         0.00  ,
 None)
```

In [4]:
```
print(data.head())
```

Loading [MathJax]/extensions/Safe.js

```
   Rank CCA3 Country/Territory           Capital Continent  2022 Population
\
0    36  AFG       Afghanistan             Kabul      Asia         41128771
1   138  ALB           Albania            Tirana    Europe          2842321
2    34  DZA           Algeria           Algiers    Africa         44903225
3   213  ASM    American Samoa         Pago Pago   Oceania            44273
4   203  AND           Andorra  Andorra la Vella    Europe            79824

   2020 Population  2015 Population  2010 Population  2000 Population  \
0         38972230         33753499         28189672         19542982
1          2866849          2882481          2913399          3182021
2         43451666         39543154         35856344         30774621
3            46189            51368            54849            58230
4            77700            71746            71519            66097

   1990 Population  1980 Population  1970 Population  Area (km²)  \
0         10694796         12486631         10752971      652230
1          3295066          2941651          2324731       28748
2         25518074         18739378         13795915     2381741
3            47818            32886            27075         199
4            53569            35611            19860         468

   Density (per km²)  Growth Rate  World Population Percentage
0            63.0587       1.0257                         0.52
1            98.8702       0.9957                         0.04
2            18.8531       1.0164                         0.56
3           222.4774       0.9831                         0.00
4           170.5641       1.0100                         0.00
```

In [5]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.subplots as sp
import plotly.graph_objects as go
```

In [6]:
```
from plotly.subplots import make_subplots
import warnings
 # Suppress FutureWarning messages
warnings.simplefilter(action='ignore', category=FutureWarning)
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplo
init_notebook_mode(connected=True)
```

In [7]:
```
data.shape
```

Out[7]:
```
(234, 17)
```

In [8]:
```
print(f"Amount of duplicates: {data.duplicated().sum()}")
```

```
Amount of duplicates: 0
```

In [9]:
```
# Check for missing values
missing_values = data.isnull().sum()
print(missing_values)
```

Loading [MathJax]/extensions/Safe.js

```
Rank                            0
CCA3                            0
Country/Territory               0
Capital                         0
Continent                       0
2022 Population                 0
2020 Population                 0
2015 Population                 0
2010 Population                 0
2000 Population                 0
1990 Population                 0
1980 Population                 0
1970 Population                 0
Area (km²)                      0
Density (per km²)               0
Growth Rate                     0
World Population Percentage     0
dtype: int64
```

In [10]:
```
# Standardize column names
data.columns = ['Rank', 'CCA3', 'Country/Territory', 'Capital', 'Continent',
                '2015 Population', '2010 Population', '2000 Population', '1990
                '1970 Population', 'Area (km²)', 'Density (per km²)', 'Growth
```

In [11]:
```
data.head()
```

Out[11]:

	Rank	CCA3	Country/Territory	Capital	Continent	2022 Population	2020 Population
0	36	AFG	Afghanistan	Kabul	Asia	41128771	38972230
1	138	ALB	Albania	Tirana	Europe	2842321	2866849
2	34	DZA	Algeria	Algiers	Africa	44903225	43451666
3	213	ASM	American Samoa	Pago Pago	Oceania	44273	46189
4	203	AND	Andorra	Andorra la Vella	Europe	79824	77700

In [12]:
```
# Convert columns to appropriate data types
data['2022 Population'] = pd.to_numeric(data['2022 Population'], errors='coe
data['2020 Population'] = pd.to_numeric(data['2020 Population'], errors='coe
data['2015 Population'] = pd.to_numeric(data['2015 Population'], errors='coe
data['2010 Population'] = pd.to_numeric(data['2010 Population'], errors='coe
data['2000 Population'] = pd.to_numeric(data['2000 Population'], errors='coe
data['1990 Population'] = pd.to_numeric(data['1990 Population'], errors='coe
data['1980 Population'] = pd.to_numeric(data['1980 Population'], errors='coe
data['1970 Population'] = pd.to_numeric(data['1970 Population'], errors='coe
data['Area (km²)'] = pd.to_numeric(data['Area (km²)'], errors='coerce')
data['Density (per km²)'] = pd.to_numeric(data['Density (per km²)'], errors=
data['Growth Rate'] = pd.to_numeric(data['Growth Rate'], errors='coerce')
data['World Population Percentage'] = pd.to_numeric(data['World Population F
```

Loading [MathJax]/extensions/Safe.js

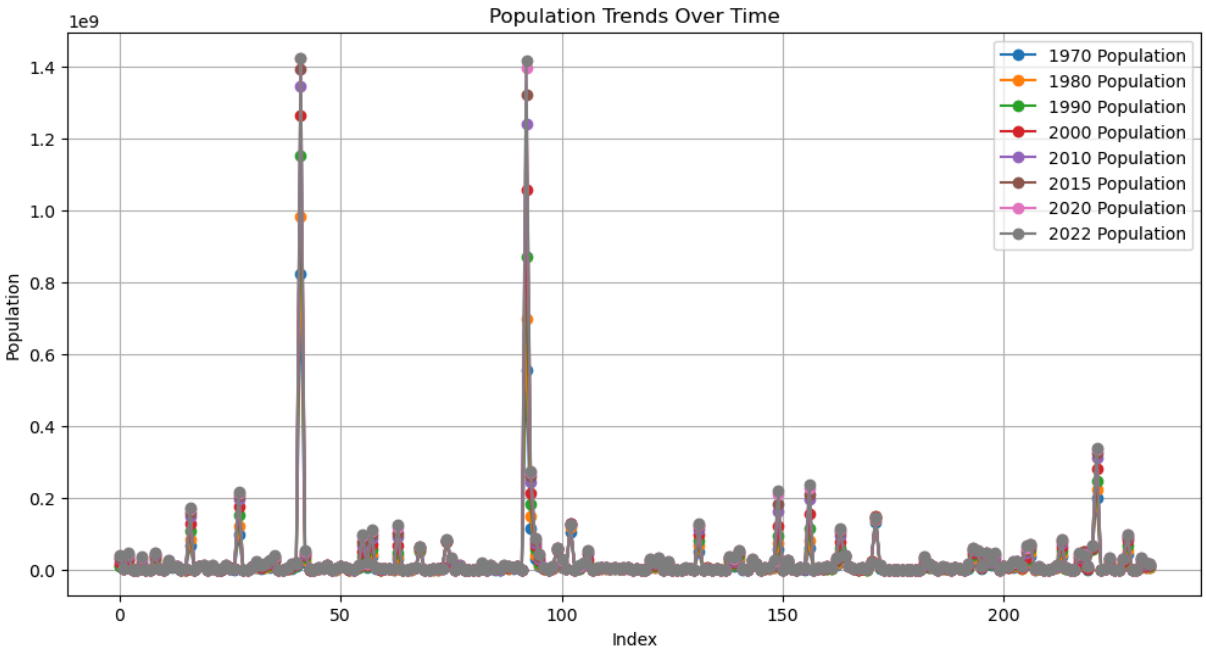
In [13]:
```
# Get basic statistics for numerical columns
stats = data.describe()
print("Basic statistics:")
print(stats)
```

```
Basic statistics:
              Rank  2022 Population  2020 Population  2015 Population  \
count  234.000000     2.340000e+02     2.340000e+02     2.340000e+02
mean   117.500000     3.407441e+07     3.350107e+07     3.172996e+07
std     67.694165     1.367664e+08     1.355899e+08     1.304050e+08
min      1.000000     5.100000e+02     5.200000e+02     5.640000e+02
25%     59.250000     4.197385e+05     4.152845e+05     4.046760e+05
50%    117.500000     5.559944e+06     5.493074e+06     5.307400e+06
75%    175.750000     2.247650e+07     2.144798e+07     1.973085e+07
max    234.000000     1.425887e+09     1.424930e+09     1.393715e+09

       2010 Population  2000 Population  1990 Population  1980 Population  \
count     2.340000e+02     2.340000e+02     2.340000e+02     2.340000e+02
mean      2.984524e+07     2.626947e+07     2.271022e+07     1.898462e+07
std       1.242185e+08     1.116982e+08     9.783217e+07     8.178519e+07
min       5.960000e+02     6.510000e+02     7.000000e+02     7.330000e+02
25%       3.931490e+05     3.272420e+05     2.641158e+05     2.296142e+05
50%       4.942770e+06     4.292907e+06     3.825410e+06     3.141146e+06
75%       1.915957e+07     1.576230e+07     1.186923e+07     9.826054e+06
max       1.348191e+09     1.264099e+09     1.153704e+09     9.823725e+08

       1970 Population    Area (km²)  Density (per km²)  Growth Rate  \
count     2.340000e+02  2.340000e+02         234.000000   234.000000
mean      1.578691e+07  5.814494e+05         452.127044     1.009577
std       6.779509e+07  1.761841e+06        2066.121904     0.013385
min       7.520000e+02  1.000000e+00           0.026100     0.912000
25%       1.559970e+05  2.650000e+03          38.417875     1.001775
50%       2.604830e+06  8.119950e+04          95.346750     1.007900
75%       8.817329e+06  4.304258e+05         238.933250     1.016950
max       8.225344e+08  1.709824e+07       23172.266700     1.069100

       World Population Percentage
count                   234.000000
mean                      0.427051
std                       1.714977
min                       0.000000
25%                       0.010000
50%                       0.070000
75%                       0.280000
max                      17.880000
```

In [14]:
```
import matplotlib.pyplot as plt

# Plot population over the years
plt.figure(figsize=(12, 6))
plt.plot(data['1970 Population'], label='1970 Population', marker='o')
plt.plot(data['1980 Population'], label='1980 Population', marker='o')
plt.plot(data['1990 Population'], label='1990 Population', marker='o')
plt.plot(data['2000 Population'], label='2000 Population', marker='o')
plt.plot(data['2010 Population'], label='2010 Population', marker='o')
plt.plot(data['2015 Population'], label='2015 Population', marker='o')
```

Loading [MathJax]/extensions/Safe.js

```
plt.plot(data['2020 Population'], label='2020 Population', marker='o')
plt.plot(data['2022 Population'], label='2022 Population', marker='o')
plt.xlabel('Index')
plt.ylabel('Population')
plt.title('Population Trends Over Time')
plt.legend()
plt.grid(True)
plt.show()
```



In [15]:

```
import pandas as pd

# Assuming 'data' is already loaded as a DataFrame
# First, print the column names to verify the exact names
print("Column names in the dataset:")
print(data.columns)

# Strip any leading/trailing spaces from column names
data.columns = data.columns.str.strip()

# Check for missing values in the relevant columns
missing_values = data[['2022 Population', '2000 Population']].isnull().sum()
print("Missing values in columns used for growth rate calculation:")
print(missing_values)

# Drop rows with missing values in these columns or fill them as needed
data.dropna(subset=['2022 Population', '2000 Population'], inplace=True)

# Calculate the growth rate
data['Growth Rate (2000-2022)'] = ((data['2022 Population'] - data['2000 Pop

# Ensure 'Country/Region' exists in the DataFrame
if 'Country/Territory' in data.columns:
    # Print the growth rate and corresponding regions
    print("Growth Rate (2000-2022):")
    print(data[['Country/Territory', 'Growth Rate (2000-2022)']])
```

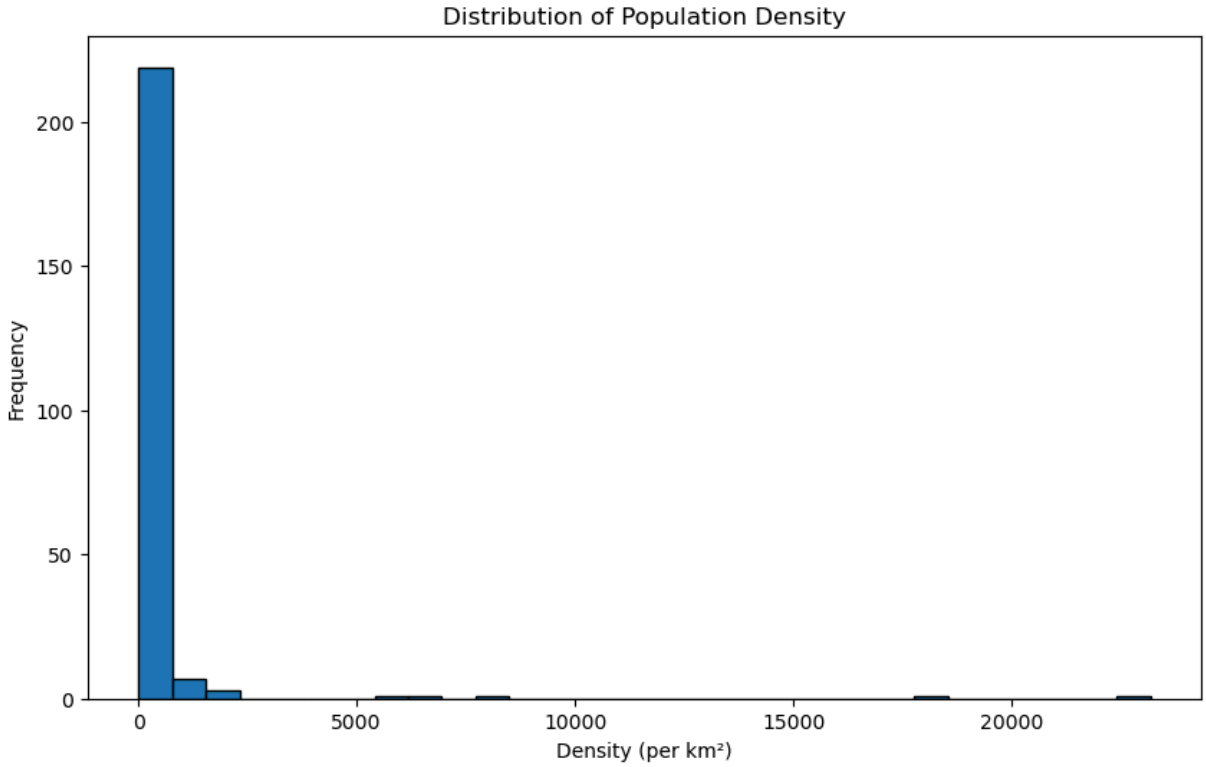
Loading [MathJax]/extensions/Safe.js

```
else:
    print("'Country/Territory' column is not in the DataFrame.")
```

```
Column names in the dataset:
Index(['Rank', 'CCA3', 'Country/Territory', 'Capital', 'Continent',
       '2022 Population', '2020 Population', '2015 Population',
       '2010 Population', '2000 Population', '1990 Population',
       '1980 Population', '1970 Population', 'Area (km²)', 'Density (per km
²)',
       'Growth Rate', 'World Population Percentage'],
      dtype='object')
Missing values in columns used for growth rate calculation:
2022 Population    0
2000 Population    0
dtype: int64
Growth Rate (2000-2022):
     Country/Territory  Growth Rate (2000-2022)
0          Afghanistan               110.452893
1              Albania               -10.675605
2              Algeria                45.909920
3       American Samoa               -23.968745
4              Andorra                20.767962
..                 ...                      ...
229  Wallis and Futuna               -21.401888
230     Western Sahara               113.032270
231              Yemen                80.885483
232             Zambia               102.379939
233           Zimbabwe                37.904384

[234 rows x 2 columns]
```

In [16]:
```
# Example: Plot population density
plt.figure(figsize=(10, 6))
plt.hist(data['Density (per km²)'].dropna(), bins=30, edgecolor='black')
plt.xlabel('Density (per km²)')
plt.ylabel('Frequency')
plt.title('Distribution of Population Density')
plt.show()
```

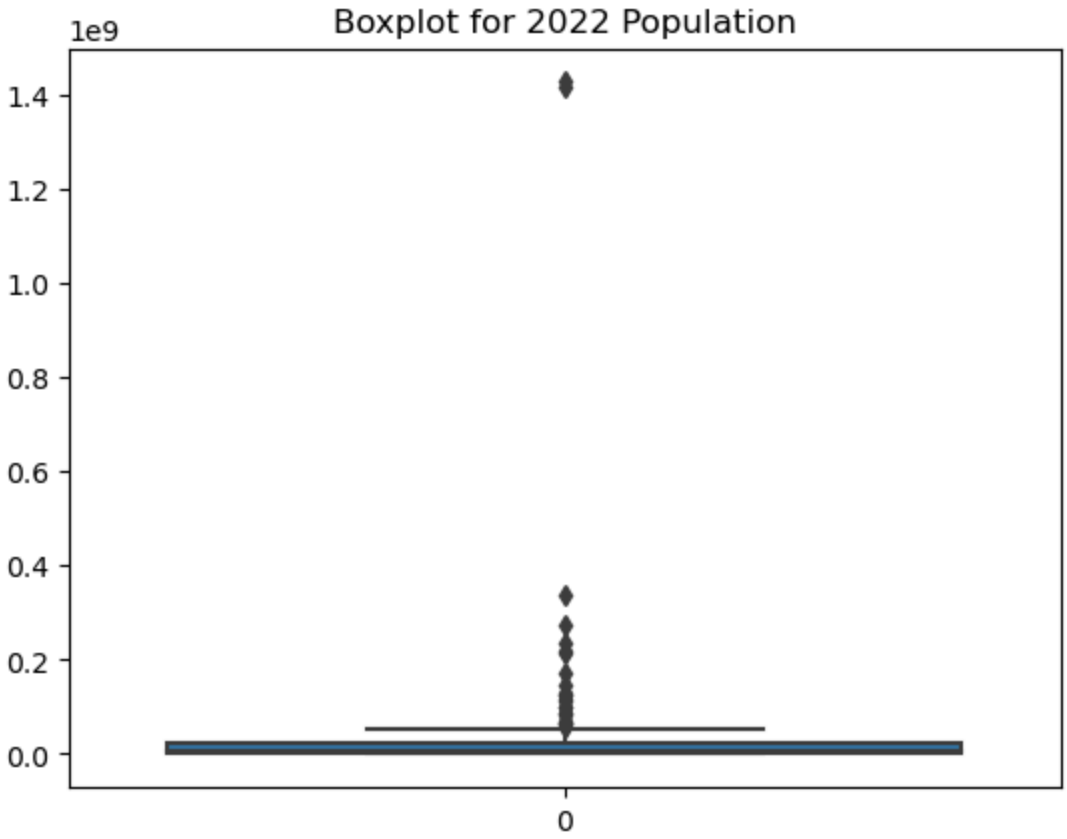
Loading [MathJax]/extensions/Safe.js

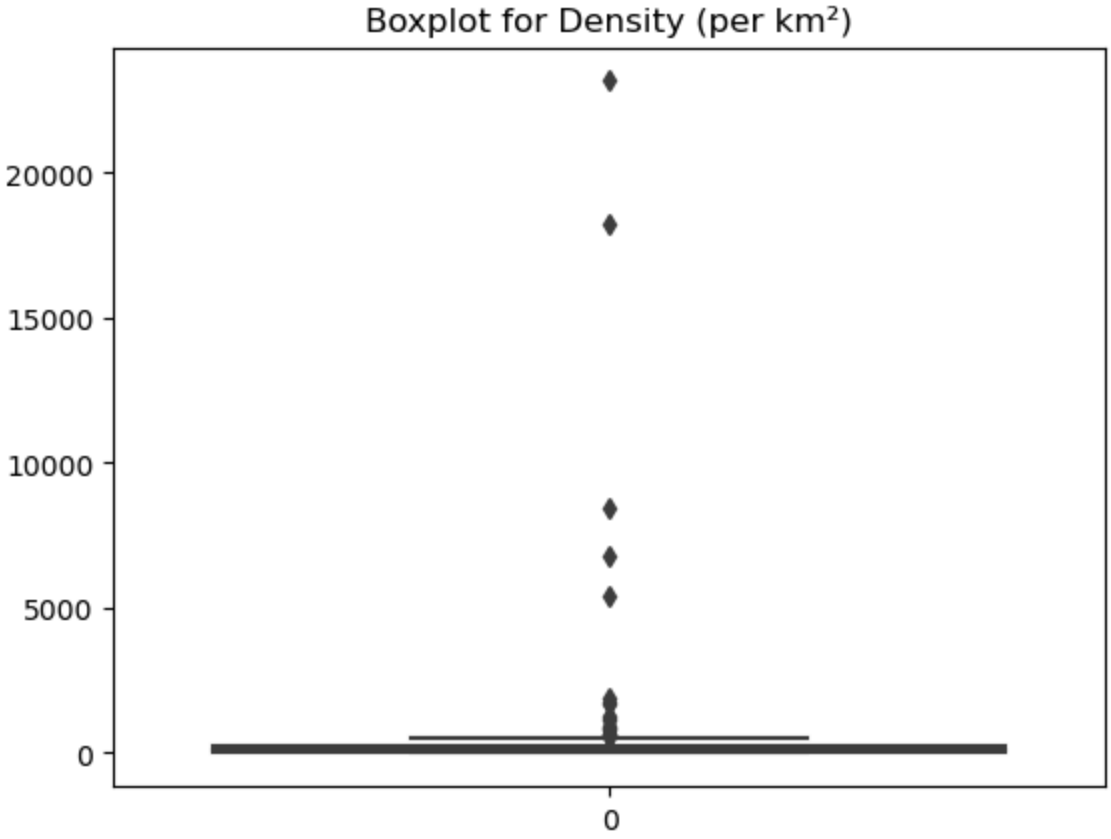


In [17]:

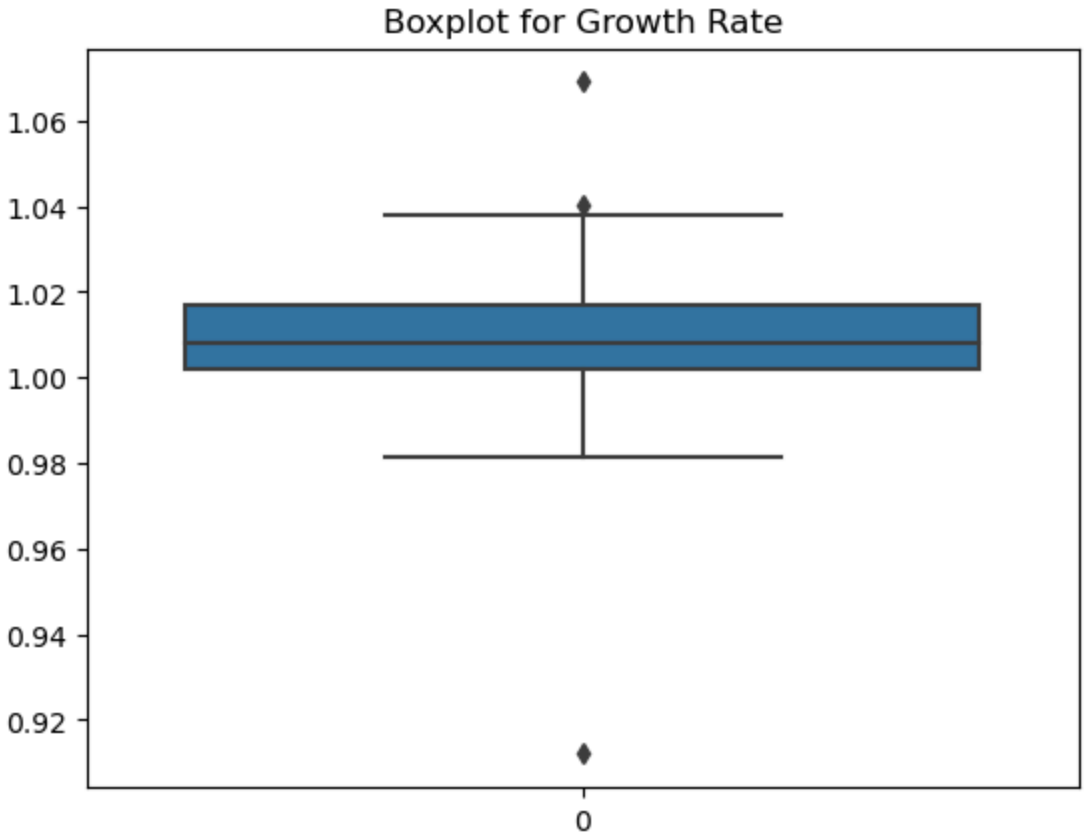
```
import seaborn as sns
import matplotlib.pyplot as plt

# Plot boxplots for numeric columns
numeric_cols = ['2022 Population', 'Density (per km²)', 'Growth Rate', 'Worl
for col in numeric_cols:
    sns.boxplot(data[col])
    plt.title(f'Boxplot for {col}')
    plt.show()

# Calculate z-scores to identify outliers
from scipy import stats

# Calculate z-scores
data['zscore_2022_pop'] = stats.zscore(data['2022 Population'])

# Filter rows with z-scores above a threshold (e.g., z-score > 3)
outliers = data[data['zscore_2022_pop'].abs() > 3]
print(outliers[['Country/Territory', '2022 Population', 'zscore_2022_pop']])
```

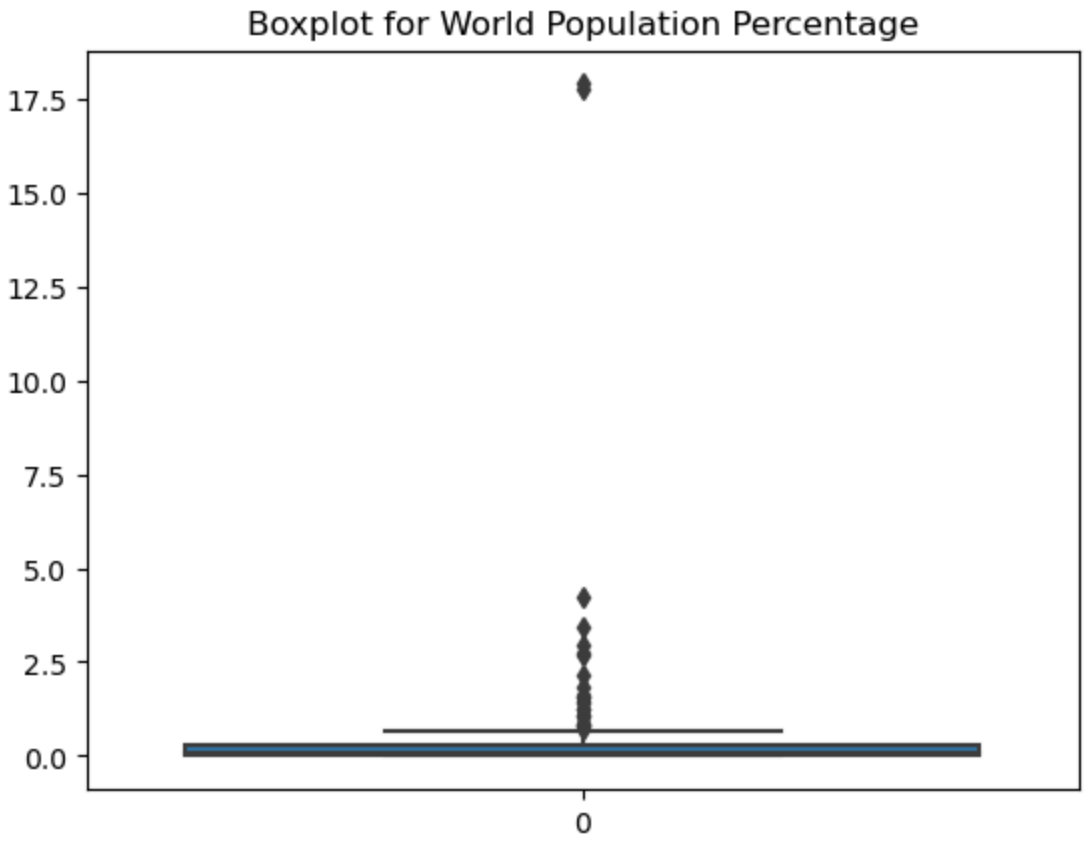
Loading [MathJax]/extensions/Safe.js





Loading [MathJax]/extensions/Safe.js





	Country/Territory	2022 Population	zscore_2022_pop
41	China	1425887337	10.198383
92	India	1417173173	10.134531

Loading [MathJax]/extensions/Safe.js

In [18]:
```
# Drop 'CCA3' and 'Capital' columns since we won't be using them in the anal
data.drop(['CCA3', 'Capital'], axis=1, inplace=True)
```

In [19]:
```
# View the first few rows
data.head()
```

Out[19]:

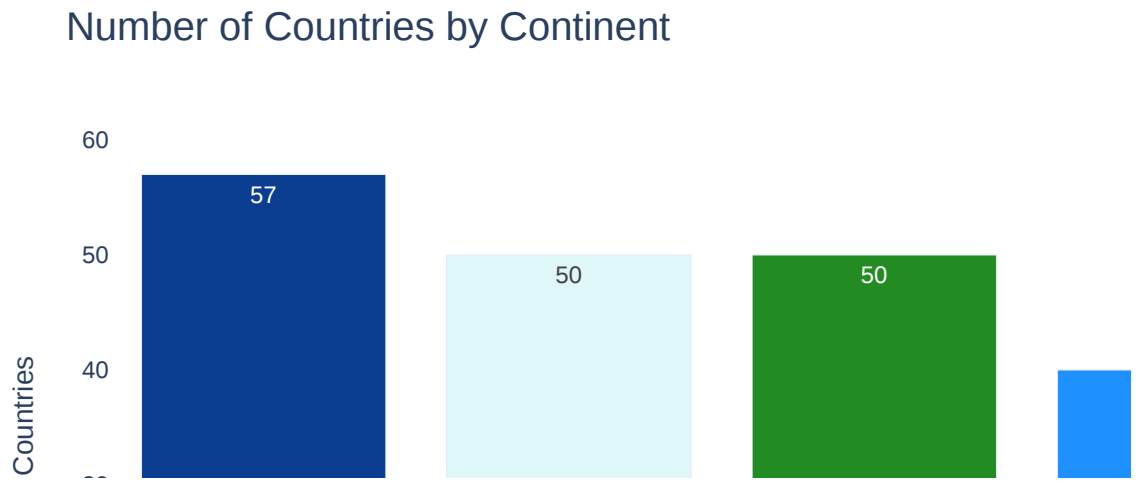
	Rank	Country/Territory	Continent	2022 Population	2020 Population	2015 Population	Po
0	36	Afghanistan	Asia	41128771	38972230	33753499	2
1	138	Albania	Europe	2842321	2866849	2882481	
2	34	Algeria	Africa	44903225	43451666	39543154	3
3	213	American Samoa	Oceania	44273	46189	51368	
4	203	Andorra	Europe	79824	77700	71746	

In [20]:
```
# Define custom color palette
custom_palette = ['#0b3d91', '#e0f7fa', '#228b22', '#1e90ff', '#8B4513', '#D
```

In [21]:
```
# Number of countries by continent
countries_by_continent = data['Continent'].value_counts().reset_index()
countries_by_continent.columns = ['Continent', 'count']
```

In [22]:
```
# Create the bar chart
fig = px.bar(
    countries_by_continent,
    x='Continent',
    y='count',
    color='Continent',
    text='count',
    title='Number of Countries by Continent',
    color_discrete_sequence=custom_palette
)
fig.update_layout(
    xaxis_title='Continents',
    yaxis_title='Number of Countries',
    plot_bgcolor='rgba(0,0,0,0)', # Set the background color to transparent
    font_family='Arial', # Set font family
    title_font_size=20 # Set title font size
)
fig.show()
```

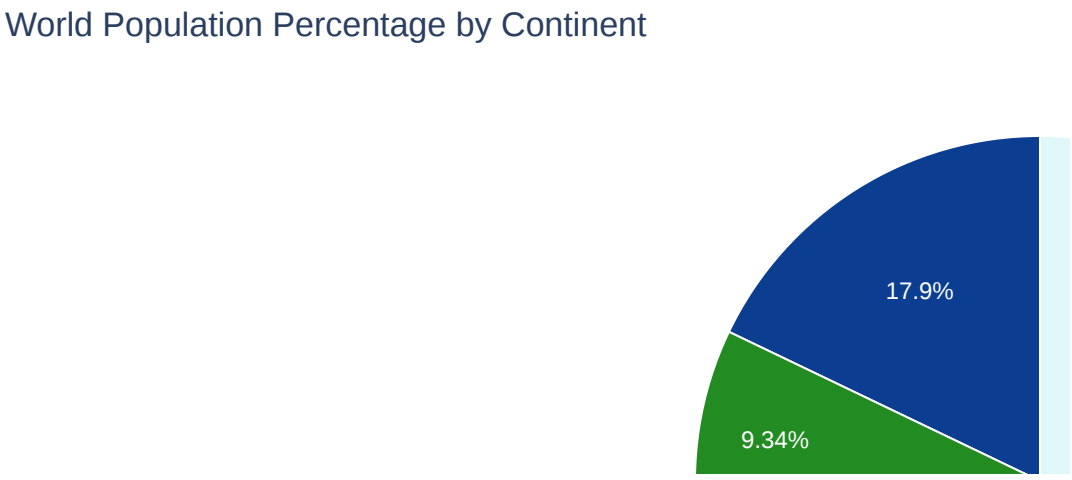
Loading [MathJax]/extensions/Safe.js



In [23]:
```
# World population percentage by continent
continent_population_percentage = data.groupby('Continent')['World Populatic
```

In [24]:
```
# Create the pie chart
fig = go.Figure(data=[go.Pie(labels=continent_population_percentage['Contine
                             values=continent_population_percentage['World F
fig.update_layout(
    title='World Population Percentage by Continent',
    template='plotly',
    paper_bgcolor='rgba(255,255,255,0)', # Set the paper background color to
    plot_bgcolor='rgba(255,255,255,0)' # Set the plot background color to tr
)
fig.update_traces(marker=dict(colors=custom_palette, line=dict(color='#FFFFF
fig.show()
```

Loading [MathJax]/extensions/Safe.js



In [25]:
```
# Melt the DataFrame to have a long format
data_melted = data.melt(id_vars=['Continent'],
                        value_vars=['2022 Population', '2020 Population', '2
                                    '2010 Population', '2000 Population', '1
                                    '1980 Population', '1970 Population'],
                        var_name='Year',
                        value_name='Population')
```

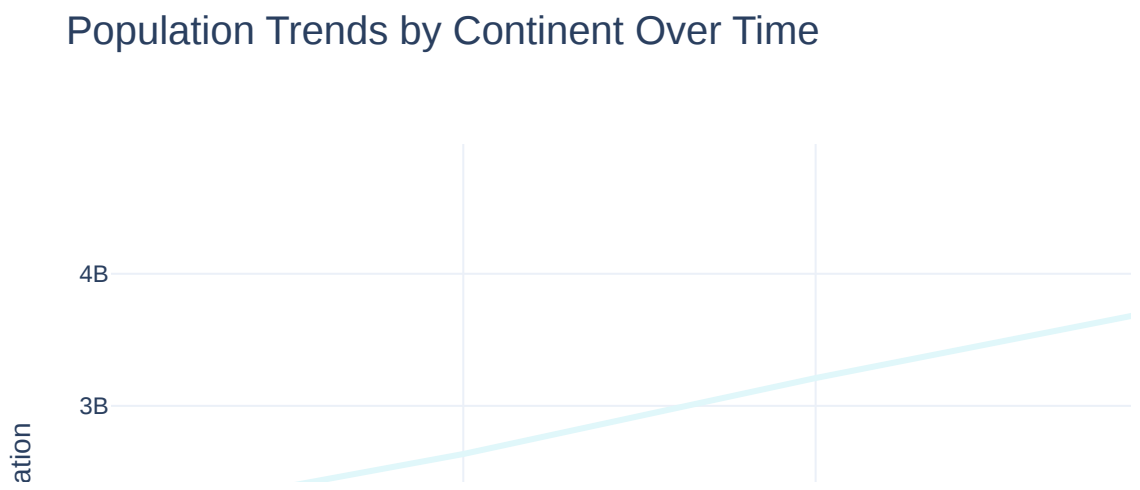
In [26]:
```
# Convert 'Year' to a more suitable format
data_melted['Year'] = data_melted['Year'].str.split().str[0].astype(int)
```

In [27]:
```
# Aggregate population by continent and year
population_by_continent = data_melted.groupby(['Continent', 'Year']).sum().r
```

In [28]:
```
# Create the line plot
fig = px.line(population_by_continent, x='Year', y='Population', color='Cont
              title='Population Trends by Continent Over Time',
              labels={'Population': 'Population', 'Year': 'Year'},
              color_discrete_sequence=custom_palette)
fig.update_layout(
    template='plotly_white',
    xaxis_title='Year',
    _title='Population',
```

Loading [MathJax]/extensions/Safe.js

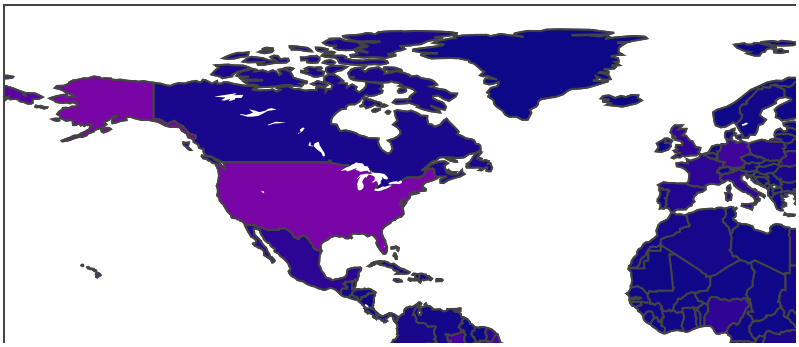
```
    font_family='Arial',
    title_font_size=20,
)
fig.update_traces(line=dict(width=3))
fig.show()
```



In [35]:
```
# World Population Comparison: 1970 to 2020
features = ['1970 Population', '2020 Population','2022 Population']
for feature in features:
    fig = px.choropleth(data,
                        locations='Country/Territory',
                        locationmode='country names',
                        color=feature,
                        hover_name='Country/Territory',
                        template='plotly_white',
                        title=feature)
    fig.show()
```

Loading [MathJax]/extensions/Safe.js

1970 Population



Loading [MathJax]/extensions/Safe.js

2020 Population

Loading [MathJax]/extensions/Safe.js

2022 Population

In [30]:
```
# Add 'Area per Person' to the DataFrame
data['Area per Person'] = data['Area (km²)'] / data['2022 Population']

# Get the countries with the most and least land available per capita
most_land_available = data.groupby('Country/Territory')['Area per Person'].s
least_land_available = data.groupby('Country/Territory')['Area per Person'].
```

In [31]:
```
import plotly.subplots as sp
import plotly.graph_objects as go

# Create subplots
fig = sp.make_subplots(rows=1, cols=2, subplot_titles=("Countries with Most

# Plot countries with the most land available per capita
fig.add_trace(
    go.Bar(x=most_land_available.index, y=most_land_available.values,
           name='Most Land Available Per Capita', marker_color='blue'),
    row=1, col=1
)

# Plot countries with the least land available per capita
fig.add_trace(
    go.Bar(x=least_land_available.index, y=least_land_available.values,
```

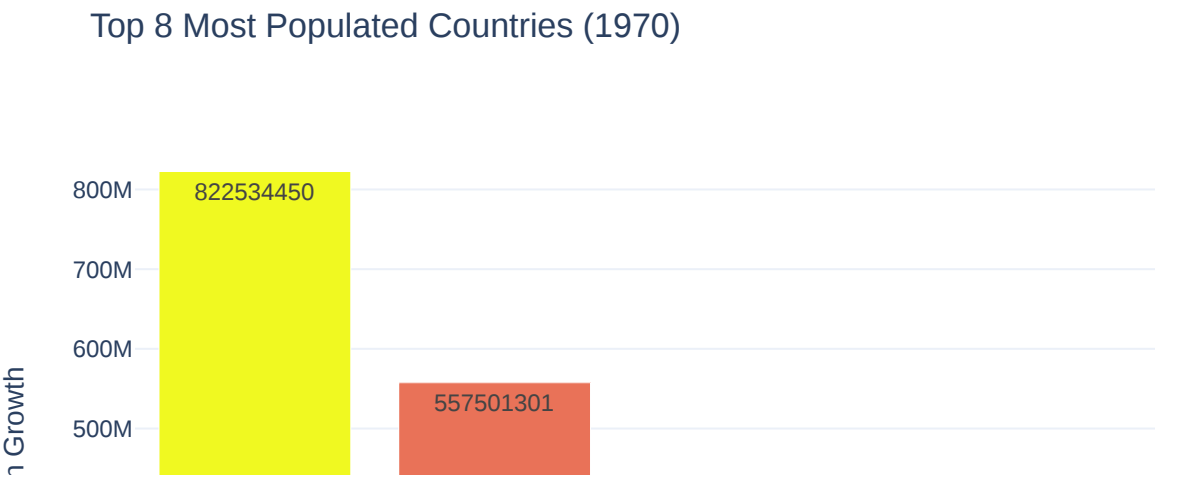
Loading [MathJax]/extensions/Safe.js

```
            name='Least Land Available Per Capita', marker_color='orange'),
    row=1, col=2
)

# Update layout
fig.update_layout(
    title_text="Distribution of Available Land Area by Country Per Capita",
    template='plotly_white',
    showlegend=False
)
fig.update_yaxes(title_text="Land Available Per Person", row=1, col=1)
fig.update_yaxes(title_text="Land Available Per Person", row=1, col=2)

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

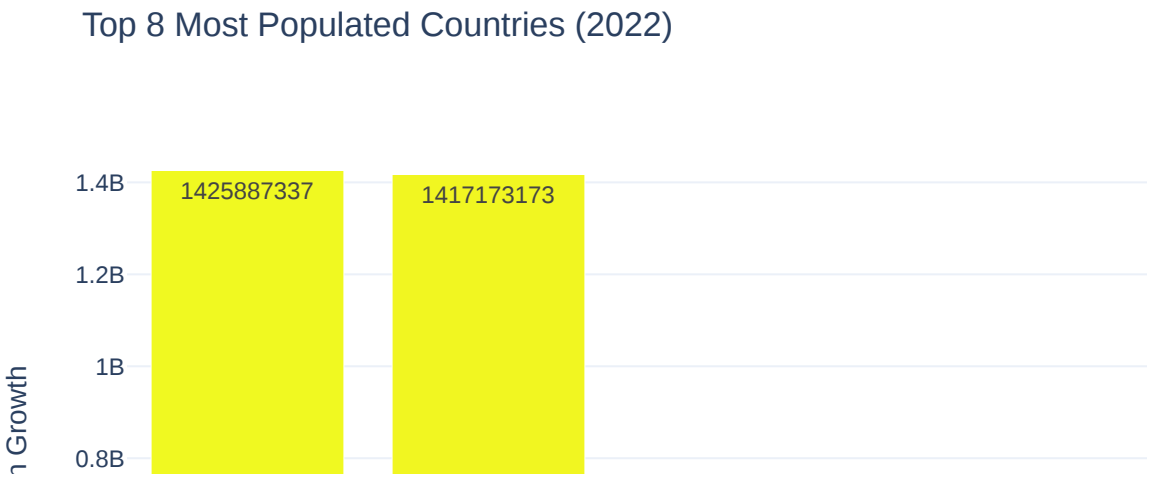
In [32]:
```
# Top 8 most populated countries in 1970 and 2022
top_8_populated_countries_1970 = data.groupby('Country/Territory')['1970 Pop
top_8_populated_countries_2022 = data.groupby('Country/Territory')['2022 Pop

features = {'top_8_populated_countries_1970': top_8_populated_countries_1970
            'top_8_populated_countries_2022': top_8_populated_countries_2022

for feature_name, feature_data in features.items():
```

Loading [MathJax]/extensions/Safe.js

```
    year = feature_name.split('_')[-1] # Extract the year from the feature r
    fig = px.bar(x=feature_data.index,
                 y=feature_data.values,
                 text=feature_data.values,
                 color=feature_data.values,
                 title=f'Top 8 Most Populated Countries ({year})',
                 template='plotly_white')
    fig.update_layout(xaxis_title='Country',
                      yaxis_title='Population Growth')
    fig.show()
```

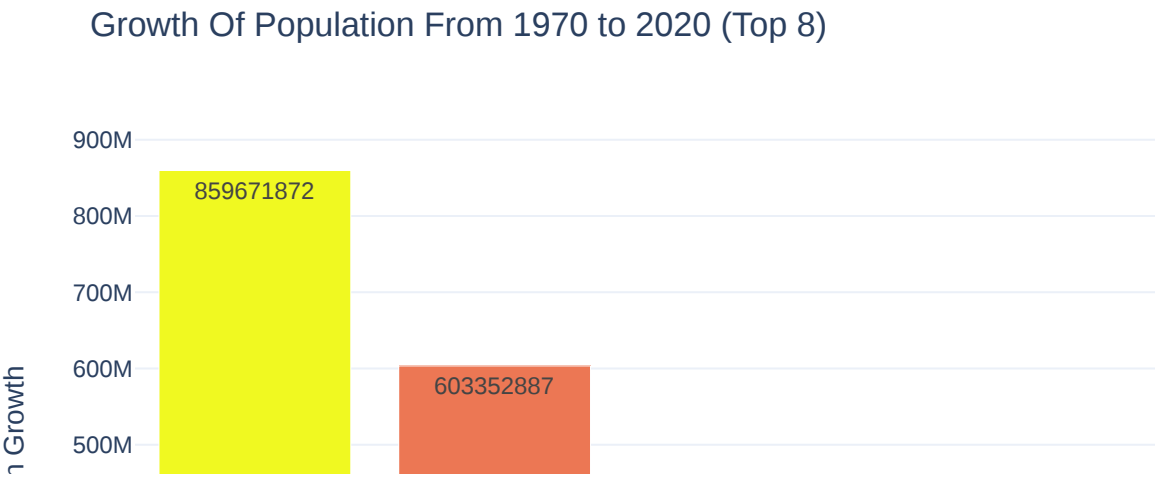


Loading [MathJax]/extensions/Safe.js



In [33]:
```
# Population growth from 1970 to 2020 (Top 8)
growth = (data.groupby(by='Country/Territory')['2022 Population'].sum() -
          data.groupby(by='Country/Territory')['1970 Population'].sum()).sor

fig = px.bar(x=growth.index,
             y=growth.values,
             text=growth.values,
             color=growth.values,
             title='Growth Of Population From 1970 to 2020 (Top 8)',
             template='plotly_white')
fig.update_layout(xaxis_title='Country',
                  yaxis_title='Population Growth')
fig.show()
```

Loading [MathJax]/extensions/Safe.js



In [46]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[46], line 2
      1 # List of fastest growing countries
----> 2 fastest_countries = top_fastest['Country/Territory'].tolist()
      4 # Plotting trends for fastest growing countries
      5 plot_population_trends(fastest_countries, data)

NameError: name 'top_fastest' is not defined
```

In [ ]:

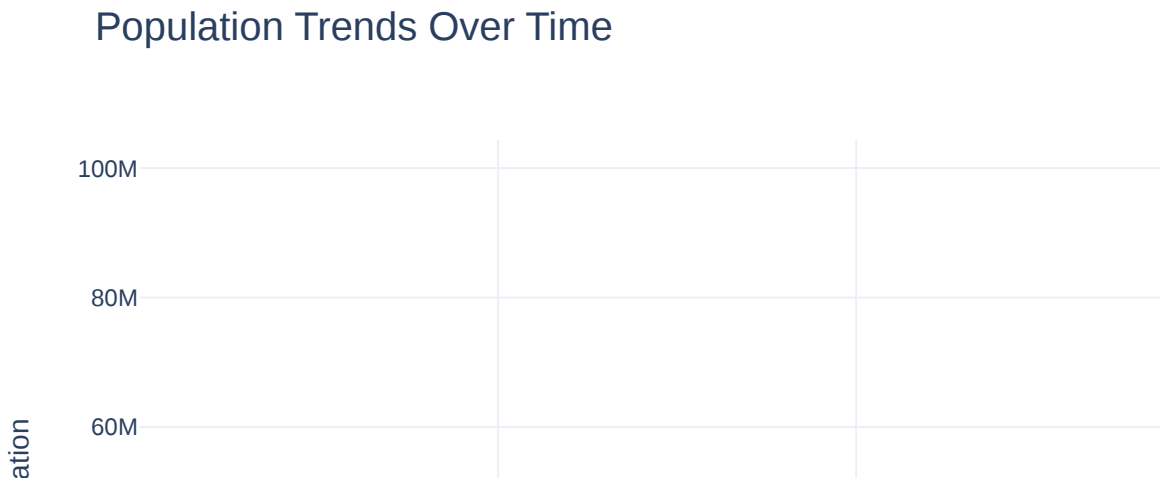
In [36]:

```
import plotly.express as px

def plot_population_trends(countries):
    # Filter data for the specified countries
    filtered_data = data[data['Country/Territory'].isin(countries)]

    # Melt the DataFrame to have a long format
    data_melted = filtered_data.melt(id_vars=['Country/Territory'],
                                     value_vars=['2022 Population', '2020 Po
                                                 '2010 Population', '2000 Po
```

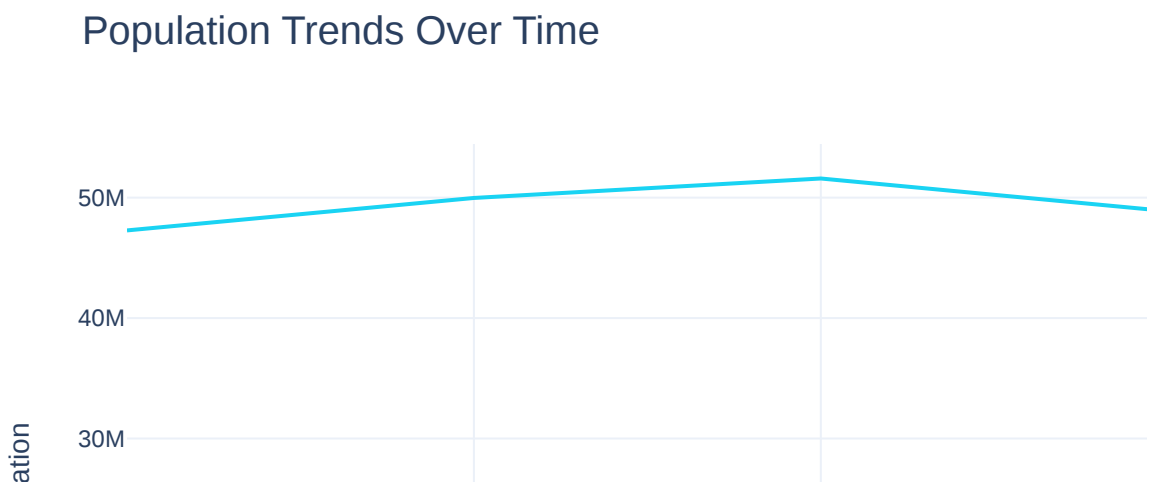
Loading [MathJax]/extensions/Safe.js

```
                                                    '1980 Population', '1970 Po
                                        var_name='Year',
                                        value_name='Population')

    # Convert 'Year' to a more suitable format
    data_melted['Year'] = data_melted['Year'].str.split().str[0].astype(int)

    # Create the line plot
    fig = px.line(data_melted, x='Year', y='Population', color='Country/Terr
                  title='Population Trends Over Time',
                  labels={'Population': 'Population', 'Year': 'Year'})
    fig.update_layout(
        template='plotly_white',
        xaxis_title='Year',
        yaxis_title='Population',
        font_family='Arial',
        title_font_size=20,
    )
    fig.show()

# Plotting trends for fastest growing countries
plot_population_trends(['Moldova', 'Poland', 'Niger', 'Syria', 'Slovakia', '

# Plotting trends for slowest growing countries
plot_population_trends(['Latvia', 'Lithuania', 'Bulgaria', 'American Samoa',
```

Loading [MathJax]/extensions/Safe.js



Loading [MathJax]/extensions/Safe.js



In [37]:
```
# List of slowest growing countries
slowest_countries = top_slowest['Country/Territory'].tolist()

# Plotting trends for slowest growing countries
plot_population_trends(slowest_countries)
```

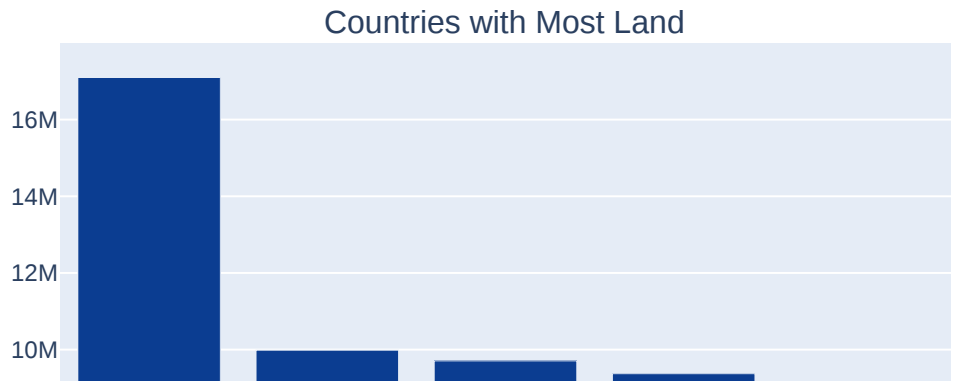
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[37], line 2
      1 # List of slowest growing countries
----> 2 slowest_countries = top_slowest['Country/Territory'].tolist()
      4 # Plotting trends for slowest growing countries
      5 plot_population_trends(slowest_countries)

NameError: name 'top_slowest' is not defined
```

In [38]:
```
#Land area by country
land_by_country = data.groupby('Country/Territory')['Area (km²)'].sum().sort
most_land = land_by_country.head(5)
least_land = land_by_country.tail(5)
```

In [40]:
```
# Create subplots
fig = sp.make_subplots(rows=1, cols=2, subplot_titles=("Countries with Most
```

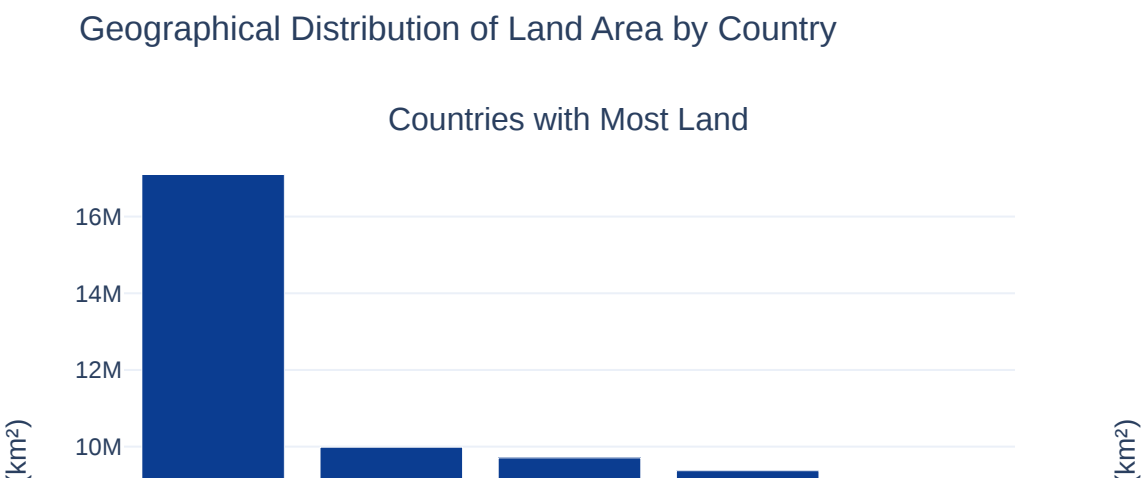
Loading [MathJax]/extensions/Safe.js

In [41]:
```
# Plot countries with the most land
fig.add_trace(go.Bar(x=most_land.index, y=most_land.values, name='Most Land'
```



In [42]:
```
# Plot countries with the least land
fig.add_trace(go.Bar(x=least_land.index, y=least_land.values, name='Least La

fig.update_layout(
    title_text="Geographical Distribution of Land Area by Country",
    showlegend=False,
    template='plotly_white'
)
fig.update_yaxes(title_text="Area (km²)", row=1, col=1)
fig.update_yaxes(title_text="Area (km²)", row=1, col=2)
fig.show()
```

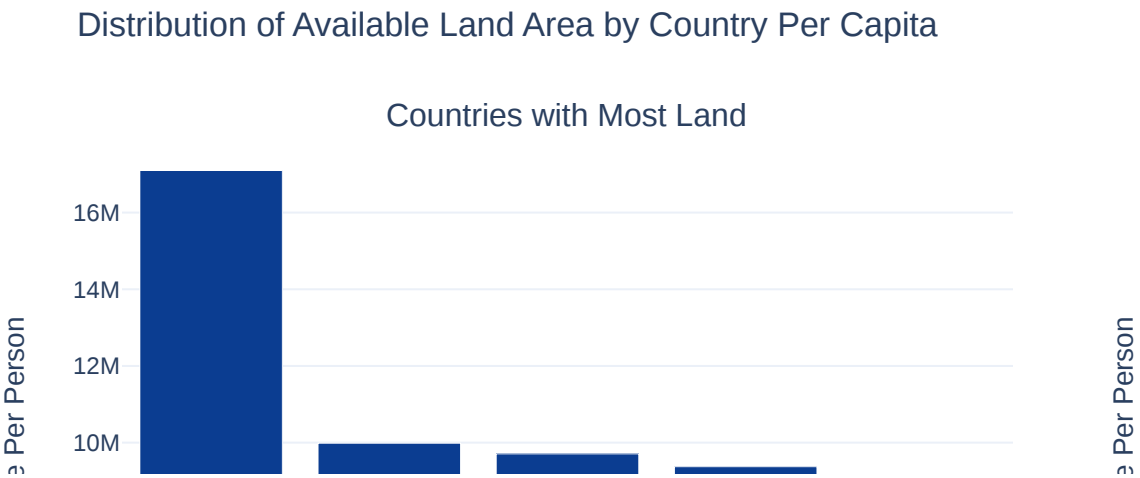
Loading [MathJax]/extensions/Safe.js



In [43]:
```
 #Land area per person by country
data['Area per Person'] = data['Area (km²)'] / data['2022 Population']
country_area_per_person = data.groupby('Country/Territory')['Area per Persor
most_land_available = country_area_per_person.sort_values(ascending=False).h
least_land_available = country_area_per_person.sort_values(ascending=False).
```

In [44]:
```
# Plot countries with the least land
fig.add_trace(go.Bar(x=least_land_available.index, y=least_land_available.va
                     name='Least Land', marker_color=custom_palette[3]), row

fig.update_layout(
    title_text="Distribution of Available Land Area by Country Per Capita",
    showlegend=False,
    template='plotly_white'
)
fig.update_yaxes(title_text="Land Available Per Person", row=1, col=1)
fig.update_yaxes(title_text="Land Available Per Person", row=1, col=2)
fig.show()
```

Loading [MathJax]/extensions/Safe.js



In [ ]:

This notebook was converted with convert.ploomber.io

Loading [MathJax]/extensions/Safe.js