

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

from prettytable import PrettyTable

import geopandas as gpd
from geopy.geocoders import Nominatim
import folium
from folium import Marker, Choropleth

sns.set(style='darkgrid', palette = 'pastel')
pd.set_option('display.float_format', lambda x: '%.2f' % x)
pd.set_option('display.max_columns', 50)
pd.set_option('display.max_colwidth', None)

import warnings
warnings.filterwarnings("ignore" )
```

C:\Users\guym\anaconda3\lib\site-packages\pandas\core\computation\expressions.py:21: UserWarning: Pandas requires version '2.8.4' or newer of 'numexpr' (version '2.7.3' currently installed).
from pandas.core.computation.check import NUMEXPR_INSTALLED
C:\Users\guym\anaconda3\lib\site-packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck' (version '1.3.2' currently installed).
from pandas.core import (
C:\Users\guym\AppData\Local\Temp\ipykernel_24632\3921357982.py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at <https://github.com/pandas-dev/pandas/issues/54466>

import pandas as pd
C:\Users\guym\anaconda3\lib\site-packages\scipy__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.26.3
warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}")

```
In [2]: import matplotlib
print(sns.__version__)
print(pd.__version__)
print(matplotlib.__version__)
print(np.__version__)
```

0.13.2
2.2.0
3.7.0
1.26.3

Load Data

```
In [3]: df = pd.read_table('idb5yr.txt', sep='|')
```

```
In [4]: df.shape
```

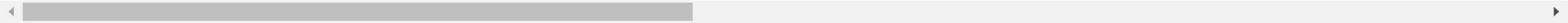
Out[4]: (34237, 115)

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

Out[5]:

	#YR	GEO_ID	AREA_KM2	ASFR15_19	ASFR20_24	ASFR25_29	ASFR30_34	ASFR35_39	ASFR40_44	ASFR45_49	CBR	CDR	E0	E0_F	E0_M	FMR0_4	FMR1_4	FPOP	FPOP0_4	FPOP10_14	FPOP100_	FPOP15_19	FPOP20_24	F
0	1950	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1951	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1952	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1953	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	1954	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 115 columns



variable definitions

```
In [6]: print('Dataframe contains a total of ' + str(len(df.GEO_ID.unique())) + ' countries.')
print('Dataframe contains data from Year ' + str(df['#YR'].min()) + ' to Year ' + str(df['#YR'].max()) )
```

Dataframe contains a total of 227 countries.
Dataframe contains data from Year 1950 to Year 2100

Extract data from year 1950 to year 2050

```
In [7]: data = df[df['#YR']<2051].copy()
```

```
In [8]: print('Row number decreased to', data.shape[0], 'rows')
```

Row number decreased to 22927 rows

Combine GEO_ID with country names

geoid.csv is generated from geoid_etl.py

```
In [9]: df_geoid = pd.read_csv('geoid.csv', encoding='iso-8859-1' )
```

```
In [10]: df_geoid.head()
```

Out[10]:

	NAME	GEO_ID	POP	YR	AGE	SEX
0	Andorra	W140000WOAD	432	2023	15	2
1	United Arab Emirates	W140000WOAE	52476	2023	15	2
2	Afghanistan	W140000WOAF	422527	2023	15	2
3	Antigua and Barbuda	W140000WOAG	724	2023	15	2

	NAME	GEO_ID	POP	YR	AGE	SEX
4	Anguilla	W140000WOAI	127	2023	15	2

```
In [11]: # take the first two columns
df_geoid = df_geoid.iloc[:,[0,1]]

df_geoid.rename(columns={'NAME':'COUNTRY'}, inplace=True)
```

```
In [12]: # merge df_geoid with data

data = data.merge(df_geoid, how='left', on='GEO_ID')
```

```
In [13]: # move country name to the front

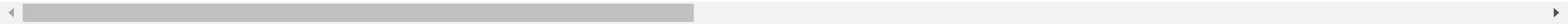
country = data.iloc[:,-1]
data.drop(columns=['COUNTRY'], inplace=True)
data.insert(loc=1, column='COUNTRY', value=country)
```

```
In [14]: data.head(5)
```

Out[14]:

	#YR	COUNTRY	GEO_ID	AREA_KM2	ASFR15_19	ASFR20_24	ASFR25_29	ASFR30_34	ASFR35_39	ASFR40_44	ASFR45_49	CBR	CDR	E0	E0_F	E0_M	FMR0_4	FMR1_4	FPOP	FPOP0_4	FPOP10_14	FPOP100_	FPOP15_19	FPOP20_24
0	1950	Andorra	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1951	Andorra	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1952	Andorra	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1953	Andorra	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	1954	Andorra	W140000WOAD	468.00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 116 columns



```
In [15]: print('There are ', len(data.COUNTRY.unique()), 'countries/territories in data')
```

There are 227 countries/territories in data

```
In [16]: data_name = data.COUNTRY.unique().tolist()
```

World Administrative Boundaries

world-administrative-boundaries.geojson is from [opendatasoft](#)

```
In [17]: df_boundaries = gpd.read_file('world-administrative-boundaries.geojson')
```

```
In [18]: df_boundaries_name = df_boundaries.name.unique().tolist()
```

```
In [19]: print('There are in total', df_boundaries.shape[0], 'administrative boundaries in the geojson file.')
```

There are in total 256 administrative boundaries in the geojson file.

```
In [20]: [name for name in data_name if name not in df_boundaries_name]
```

```
Out[20]: ['Antigua and Barbuda',
          'Bosnia and Herzegovina',
          'Saint Barthelemy',
          'Brunei',
          'Bahamas, The',
          'Congo (Kinshasa)',
          'Congo (Brazzaville)',
          'Cabo Verde',
          'Curaçao',
          'Czechia',
          'Micronesia, Federated States of',
          'United Kingdom',
          'Gambia, The',
          'Iran',
          'Korea, North',
          'Korea, South',
          'Laos',
          'Libya',
          'Moldova',
          'Saint Martin',
          'North Macedonia',
          'Burma',
          'Macau',
          'Saint Pierre and Miquelon',
          'Russia',
          'Saint Helena, Ascension, and Tristan da Cunha',
          'Sint Maarten',
          'Syria',
          'Eswatini',
          'Tanzania',
          'United States',
          'Virgin Islands, British',
          'Virgin Islands, U.S.',
          'Wallis and Futuna',
          'Kosovo']
```

```
In [21]: # change country names in df_boundaries to match names in data
name_dict = {'Antigua & Barbuda': 'Antigua and Barbuda',
             'Bosnia & Herzegovina': 'Bosnia and Herzegovina',
             'Brunei Darussalam': 'Brunei',
             'Bahamas': 'Bahamas, The',
             'Democratic Republic of the Congo': 'Congo (Kinshasa)',
             'Congo': 'Congo (Brazzaville)',
             'Cape Verde': 'Cabo Verde',
             'Netherlands Antilles': 'Curaçao',
             'Czech Republic': 'Czechia',
             'Micronesia (Federated States of)': 'Micronesia, Federated States of',
             'U.K. of Great Britain and Northern Ireland': 'United Kingdom',
```

```
'Gambia': 'Gambia, The',
'Iran (Islamic Republic of)': 'Iran',
'Democratic People's Republic of Korea': 'Korea, North',
'Republic of Korea': 'Korea, South',
'Lao People's Democratic Republic': 'Laos',
'Libyan Arab Jamahiriya': 'Libya',
'Moldova, Republic of': 'Moldova',
'The former Yugoslav Republic of Macedonia': 'North Macedonia',
'Myanmar': 'Burma',
'Russian Federation': 'Russia',
'Syrian Arab Republic': 'Syria',
'Swaziland': 'Eswatini',
'United Republic of Tanzania': 'Tanzania',
'United States of America': 'United States',
'British Virgin Islands': 'Virgin Islands, British',
'United States Virgin Islands': 'Virgin Islands, U.S.'}
```

Update df_boundaries with name_dict, Non-Exhaustive Mapping

```
In [22]: df_boundaries['name'] = df_boundaries['name'].map(name_dict).fillna(df_boundaries['name'])
```

```
In [23]: # double check
df_boundaries_name = df_boundaries.name.unique().tolist()
[name for name in data_name if name not in df_boundaries_name]
```

```
Out[23]: ['Saint Barthelemy',
'Saint Martin',
'Macau',
'Saint Pierre and Miquelon',
'Saint Helena, Ascension, and Tristan da Cunha',
'Sint Maarten',
'Wallis and Futuna',
'Kosovo']
```

df_boundaries is missing Saint Martin , Saint Barthelemy , Macau , Saint Pierre and Miquelon , Saint Helena, Ascension, and Tristan da Cunha , Sint Maarten , Wallis and Futuna , Kosovo

Get coordinates from geojson.io

```
In [24]: additional_geo = {"type": "FeatureCollection",
"features": [{"type": "Feature",
"geometry": {
"coordinates": [[[-62.91132219738526, 17.96343155866785], [-62.907476690909206, 17.950424674152387], [ -62.86303972718257, 17.890662016508315], [ -62.8121935859958, 17.865043283470357], [-62.767476690909206, 17.850424674152387], [-62.722976690909206, 17.835824674152387], [-62.678476690909206, 17.821224674152387], [-62.633976690909206, 17.806624674152387], [-62.589476690909206, 17.792024674152387], [-62.544976690909206, 17.777424674152387], [-62.500476690909206, 17.762824674152387], [-62.455976690909206, 17.748224674152387], [-62.411476690909206, 17.733624674152387], [-62.366976690909206, 17.719024674152387], [-62.322476690909206, 17.704424674152387], [-62.277976690909206, 17.689824674152387], [-62.233476690909206, 17.675224674152387], [-62.188976690909206, 17.660624674152387], [-62.144476690909206, 17.646024674152387], [-62.100076690909206, 17.631424674152387], [-62.055576690909206, 17.616824674152387], [-62.011076690909206, 17.602224674152387], [-61.966576690909206, 17.587624674152387], [-61.922076690909206, 17.573024674152387], [-61.877576690909206, 17.558424674152387], [-61.833076690909206, 17.543824674152387], [-61.788576690909206, 17.529224674152387], [-61.744076690909206, 17.514624674152387], [-61.699576690909206, 17.500024674152387], [-61.655076690909206, 17.485424674152387], [-61.610576690909206, 17.470824674152387], [-61.566076690909206, 17.456224674152387], [-61.521576690909206, 17.441624674152387], [-61.477076690909206, 17.427024674152387], [-61.432576690909206, 17.412424674152387], [-61.388076690909206, 17.397824674152387], [-61.343576690909206, 17.383224674152387], [-61.299076690909206, 17.368624674152387], [-61.254576690909206, 17.354024674152387], [-61.210076690909206, 17.339424674152387], [-61.165576690909206, 17.324824674152387], [-61.121076690909206, 17.310224674152387], [-61.076576690909206, 17.295624674152387], [-61.032076690909206, 17.281024674152387], [-60.987576690909206, 17.266424674152387], [-60.943076690909206, 17.251824674152387], [-60.898576690909206, 17.237224674152387], [-60.854076690909206, 17.222624674152387], [-60.809576690909206, 17.208024674152387], [-60.765076690909206, 17.193424674152387], [-60.720576690909206, 17.178824674152387], [-60.676076690909206, 17.164224674152387], [-60.631576690909206, 17.149624674152387], [-60.587076690909206, 17.135024674152387], [-60.542576690909206, 17.120424674152387], [-60.498076690909206, 17.105824674152387], [-60.453576690909206, 17.091224674152387], [-60.409076690909206, 17.076624674152387], [-60.364576690909206, 17.062024674152387], [-60.320076690909206, 17.047424674152387], [-60.275576690909206, 17.032824674152387], [-60.231076690909206, 17.018224674152387], [-60.186576690909206, 17.003624674152387], [-60.142076690909206, 16.989024674152387], [-60.097576690909206, 16.974424674152387], [-60.053076690909206, 16.959824674152387], [-60.008576690909206, 16.945224674152387], [-59.964076690909206, 16.930624674152387], [-59.919576690909206, 16.916024674152387], [-59.875076690909206, 16.901424674152387], [-59.830576690909206, 16.886824674152387], [-59.786076690909206, 16.872224674152387], [-59.741576690909206, 16.857624674152387], [-59.697076690909206, 16.843024674152387], [-59.652576690909206, 16.828424674152387], [-59.608076690909206, 16.813824674152387], [-59.563576690909206, 16.799224674152387], [-59.519076690909206, 16.784624674152387], [-59.474576690909206, 16.770024674152387], [-59.430076690909206, 16.755424674152387], [-59.385576690909206, 16.740824674152387], [-59.341076690909206, 16.726224674152387], [-59.296576690909206, 16.711624674152387], [-59.252076690909206, 16.697024674152387], [-59.207576690909206, 16.682424674152387], [-59.163076690909206, 16.667824674152387], [-59.118576690909206, 16.653224674152387], [-59.074076690909206, 16.638624674152387], [-59.029576690909206, 16.624024674152387], [-58.985076690909206, 16.609424674152387], [-58.940576690909206, 16.594824674152387], [-58.896076690909206, 16.580224674152387], [-58.851576690909206, 16.565624674152387], [-58.807076690909206, 16.551024674152387], [-58.762576690909206, 16.536424674152387], [-58.718076690909206, 16.521824674152387], [-58.673576690909206, 16.507224674152387], [-58.629076690909206, 16.492624674152387], [-58.584576690909206, 16.478024674152387], [-58.540076690909206, 16.463424674152387], [-58.495576690909206, 16.448824674152387], [-58.451076690909206, 16.434224674152387], [-58.406576690909206, 16.419624674152387], [-58.362076690909206, 16.405024674152387], [-58.317576690909206, 16.390424674152387], [-58.273076690909206, 16.375824674152387], [-58.228576690909206, 16.361224674152387], [-58.184076690909206, 16.346624674152387], [-58.139576690909206, 16.332024674152387], [-58.095076690909206, 16.317424674152387], [-58.050576690909206, 16.302824674152387], [-58.006076690909206, 16.288224674152387], [-57.961576690909206, 16.273624674152387], [-57.917076690909206, 16.259024674152387], [-57.872576690909206, 16.244424674152387], [-57.828076690909206, 16.229824674152387], [-57.783576690909206, 16.215224674152387], [-57.739076690909206, 16.200624674152387], [-57.694576690909206, 16.186024674152387], [-57.650076690909206, 16.171424674152387], [-57.605576690909206, 16.156824674152387], [-57.561076690909206, 16.142224674152387], [-57.516576690909206, 16.127624674152387], [-57.472076690909206, 16.113024674152387], [-57.427576690909206, 16.098424674152387], [-57.383076690909206, 16.083824674152387], [-57.338576690909206, 16.069224674152387], [-57.294076690909206, 16.054624674152387], [-57.249576690909206, 16.040024674152387], [-57.205076690909206, 16.025424674152387], [-57.160576690909206, 16.010824674152387], [-57.116076690909206, 15.996224674152387], [-57.071576690909206, 15.981624674152387], [-57.027076690909206, 15.967024674152387], [-56.982576690909206, 15.952424674152387], [-56.938076690909206, 15.937824674152387], [-56.893576690909206, 15.923224674152387], [-56.849076690909206, 15.908624674152387], [-56.804576690909206, 15.894024674152387], [-56.760076690909206, 15.879424674152387], [-56.715576690909206, 15.864824674152387], [-56.671076690909206, 15.850224674152387], [-56.626576690909206, 15.835624674152387], [-56.582076690909206, 15.821024674152387], [-56.537576690909206, 15.806424674152387], [-56.493076690909206, 15.791824674152387], [-56.448576690909206, 15.777224674152387], [-56.404076690909206, 15.762624674152387], [-56.359576690909206, 15.748024674152387], [-56.315076690909206, 15.733424674152387], [-56.270576690909206, 15.718824674152387], [-56.226076690909206, 15.704224674152387], [-56.181576690909206, 15.689624674152387], [-56.137076690909206, 15.675024674152387], [-56.092576690909206, 15.660424674152387], [-56.048076690909206, 15.645824674152387], [-56.003576690909206, 15.631224674152387], [-55.959076690909206, 15.616624674152387], [-55.914576690909206, 15.602024674152387], [-55.870076690909206, 15.587424674152387], [-55.825576690909206, 15.572824674152387], [-55.781076690909206, 15.558224674152387], [-55.736576690909206, 15.543624674152387], [-55.692076690909206, 15.529024674152387], [-55.647576690909206, 15.514424674152387], [-55.603076690909206, 15.500024674152387], [-55.558576690909206, 15.485424674152387], [-55.514076690909206, 15.470824674152387], [-55.469576690909206, 15.456224674152387], [-55.425076690909206, 15.441624674152387], [-55.380576690909206, 15.427024674152387], [-55.336076690909206, 15.412424674152387], [-55.291576690909206, 15.397824674152387], [-55.247076690909206, 15.383224674152387], [-55.202576690909206, 15.368624674152387], [-55.158076690909206, 15.354024674152387], [-55.113576690909206, 15.339424674152387], [-55.069076690909206, 15.324824674152387], [-55.024576690909206, 15.310224674152387], [-54.980076690909206, 15.295624674152387], [-54.935576690909206, 15.281024674152387], [-54.891076690909206, 15.266424674152387], [-54.846576690909206, 15.251824674152387], [-54.802076690909206, 15.237224674152387], [-54.757576690909206, 15.222624674152387], [-54.713076690909206, 15.208024674152387], [-54.668576690909206, 15.193424674152387], [-54.624076690909206, 15.178824674152387], [-54.579576690909206, 15.164224674152387], [-54.535076690909206, 15.149624674152387], [-54.490576690909206, 15.135024674152387], [-54.446076690909206, 15.120424674152387], [-54.401576690909206, 15.105824674152387], [-54.357076690909206, 15.091224674152387], [-54.312576690909206, 15.076624674152387], [-54.268076690909206, 15.062024674152387], [-54.223576690909206, 15.047424674152387], [-54.179076690909206, 15.032824674152387], [-54.134576690909206, 15.018224674152387], [-54.090076690909206, 15.003624674152387], [-54.045576690909206, 14.989024674152387], [-54.001076690909206, 14.974424674152387], [-53.956576690909206, 14.959824674152387], [-53.912076690909206, 14.945224674152387], [-53.867576690909206, 14.930624674152387], [-53.823076690909206, 14.916024674152387], [-53.778576690909206, 14.901424674152387], [-53.734076690909206, 14.886824674152387], [-53.689576690909206, 14.872224674152387], [-53.645076690909206, 14.857624674152387], [-53.600576690909206, 14.843024674152387], [-53.556076690909206, 14.828424674152387], [-53.511576690909206, 14.813824674152387], [-53.467076690909206, 14.799224674152387], [-53.422576690909206, 14.784624674152387], [-53.378076690909206, 14.770024674152387], [-53.333576690909206, 14.755424674152387], [-53.289076690909206, 14.740824674152387], [-53.244576690909206, 14.726224674152387], [-53.200076690909206, 14.711624674152387], [-53.155576690909206, 14.697024674152387], [-53.111076690909206, 14.682424674152387], [-53.066576690909206, 14.667824674152387], [-53.022076690909206, 14.653224674152387], [-52.977576690909206, 14.638624674152387], [-52.933076690909206, 14.624024674152387], [-52.888576690909206, 14.609424674152387], [-52.844076690909206, 14.594824674152387], [-52.799576690909206, 14.580224674152387], [-52.755076690909206, 14.565624674152387], [-52.710576690909206, 14.551024674152387], [-52.666076690909206, 14.536424674152387], [-52.621576690909206, 14.521824674152387], [-52.577076690909206, 14.507224674152387], [-52.532576690909206, 14.492624674152387], [-52.488076690909206, 14.478024674152387], [-52.443576690909206, 14.463424674152387], [-52.399076690909206, 14.448824674152387], [-52.354576690909206, 14.434224674152387], [-52.310076690909206, 14.419624674152387], [-52.265576690909206, 14.405024674152387], [-52.221076690909206, 14.390424674152387], [-52.176576690909206, 14.375824674152387], [-52.132076690909206, 14.361224674152387], [-52.087576690909206, 14.346624674152387], [-52.043076690909206, 14.332024674152387], [-52.000076690909206, 14.317424674152387], [-51.955576690909206, 14.302824674152387], [-51.911076690909206, 14.288224674152387], [-51.866576690909206, 14.273624674152387], [-51.822076690909206, 14.259024674152387], [-51.777576690909206, 14.244424674152387], [-51.733076690909206, 14.229824674152387], [-51.688576690909206, 14.215224674152387], [-51.644076690909206, 14.200624674152387], [-51.599576690909206, 14.186024674152387], [-51.555076690909206, 14.171424674152387], [-51.510576690909206, 14.156824674152387], [-51.466076690909206, 14.142224674152387], [-51.421576690909206, 14.127624674152387], [-51.377076690909206, 14.113024674152387], [-51.332576690909206, 14.098424674152387], [-51.288076690909206, 14.083824674152387], [-51.243576690909206, 14.069224674152387], [-51.199076690909206, 14.054624674152387], [-51.154576690909206, 14.040024674152387], [-51.110076690909206, 14.025424674152387], [-51.065576690909206, 14.010824674152387], [-51.021076690909206, 13.996224674152387], [-50.976576690909206, 13.981624674152387], [-50.932076690909206, 13.967024674152387], [-50.887576690909206, 13.952424674152387], [-50.843076690909206, 13.937824674152387], [-50.798576690909206, 13.923224674152387], [-50.754076690909206, 13.908624674152387], [-50.709576690909206, 13.894024674152387], [-50.665076690909206, 13.879424674152387], [-50.620576690909206, 13.864824674152387], [-50.576076690909206, 13.850224674152387], [-50.531576690909206, 13.835624674152387], [-50.487076690909206, 13.821024674152387], [-50.442576690909206, 13.806424674152387], [-50.398076690909206, 13.791824674152387], [-50.353576690909206, 13.777224674152387], [-50.309076690909206, 13.762624674152387], [-50.264576690909206, 13.748024674152387], [-50.220076690909206, 13.733424674152387], [-50.175576690909206, 13.718824674152387], [-50.131076690909206, 13.704224674152387], [-50.086576690909206, 13.689624674152387], [-50.042076690909206, 13.675024674152387], [-50.000076690909206, 13.660424674152387], [-49.955576690909206, 13
```

```
    "type": "Polygon"},
    "properties": {
      "geo_point_2d": {
        "lon": -63.08380212192628,
        "lat": 18.066775058581598
      },
      "iso3": "MAF",
      "status": "French Republic",
      "color_code": "MAF",
      "name": "Saint Martin",
      "continent": "Americas",
      "region": "Caribbean",
      "iso_3166_1_alpha_2_codes": "MF",
      "french_short": "Saint Martin"
    }
  },
  {
    "type": "Feature",
    "geometry": {
      "coordinates": [
        [
          [
            113.54412109659705, 22.216468895070577
          ],
          [
            113.53488535516442, 22.20294715848827
          ],
          [
            113.52801224526104, 22.18345765868783
          ],
          [
            113.53982540290883, 22.15839575803186
          ],
          [
            113.54990111111111, 22.15839575803186
          ],
          [
            113.54412109659705, 22.216468895070577
          ]
        ]
      ],
      "type": "Polygon"
    },
    "properties": {
      "geo_point_2d": {
        "lon": 113.5406845416461,
        "lat": 22.190020751444877
      },
      "iso3": "MAC",
      "status": "CN Special Administrative Region",
      "color_code": "CHN",
      "name": "Macau",
      "continent": "Asia",
      "region": "Eastern Asia",
      "iso_3166_1_alpha_2_codes": "MO",
      "french_short": "Macau"
    }
  },
  {
    "type": "Feature",
    "geometry": {
      "coordinates": [
        [
          [
            -56.36097272057373, 47.14882610748785
          ],
          [
            -56.423344443486585, 47.10727482074435
          ],
          [
            -56.37006859683218, 46.97888011389537
          ],
          [
            -56.36227213146806, 46.89103998607763
          ],
          [
            -56.42074111111111, 47.14882610748785
          ]
        ]
      ],
      "type": "Polygon"
    },
    "properties": {
      "geo_point_2d": {
        "lon": -56.17854216659805,
        "lat": 46.77725962397696
      },
      "iso3": "SPM",
      "status": "French archipelago",
      "color_code": "SPM",
      "name": "Saint Pierre and Miquelon",
      "continent": "Americas",
      "region": "Northern America",
      "iso_3166_1_alpha_2_codes": "PM",
      "french_short": "Saint Pierre and Miquelon"
    }
  },
  {
    "type": "Feature",
    "geometry": {
      "coordinates": [
        [
          [
            -5.704634894341467, -15.90296453512316
          ],
          [
            -5.754876246669795, -15.942378369744631
          ],
          [
            -5.793548866554687, -15.993541227517525
          ],
          [
            -5.748265542415652, -16.029442745995965
          ],
          [
            -5.704634894341467, -15.90296453512316
          ]
        ]
      ],
      "type": "Polygon"
    },
    "properties": {
      "geo_point_2d": {
        "lon": -5.692074556259314,
        "lat": -15.94873471510013
      },
      "iso3": "SHN",
      "status": "British Overseas Territory",
      "color_code": "SHN",
      "name": "Saint Helena, Ascension, and Tristan da Cunha",
      "continent": "Africa",
      "region": "Western Africa",
      "iso_3166_1_alpha_2_codes": "SH",
      "french_short": "Saint Helena, Ascension, and Tristan da Cunha"
    }
  },
  {
    "type": "Feature",
    "geometry": {
      "coordinates": [
        [
          [
            -63.0135527824156, 18.053619679889835
          ],
          [
            -63.02333308922579, 18.05723583961033
          ],
          [
            -63.0303966441448, 18.054480677047763
          ],
          [
            -63.03727908227043, 18.057924623501805
          ],
          [
            -63.03980111111111, 18.053619679889835
          ]
        ]
      ],
      "type": "Polygon"
    },
    "properties": {
      "geo_point_2d": {
        "lon": -63.042712586053455,
        "lat": 18.045353892620525
      },
      "iso3": "SXM",
      "status": "Constituent country in the Kingdom of the Netherlands",
      "color_code": "SXM",
      "name": "Sint Maarten",
      "continent": "North America",
      "region": "Caribbean",
      "iso_3166_1_alpha_2_codes": "SX",
      "french_short": "Sint Maarten"
    }
  },
  {
    "type": "Feature",
    "geometry": {
      "coordinates": [
        [
          [
            -63.042712586053455, 18.045353892620525
          ],
          [
            -63.042712586053455, 18.045353892620525
          ]
        ]
      ],
      "type": "Point"
    },
    "properties": {
      "geo_point_2d": {
        "lon": -63.042712586053455,
        "lat": 18.045353892620525
      },
      "iso3": "SXM",
      "status": "Constituent country in the Kingdom of the Netherlands",
      "color_code": "SXM",
      "name": "Sint Maarten",
      "continent": "North America",
      "region": "Caribbean",
      "iso_3166_1_alpha_2_codes": "SX",
      "french_short": "Sint Maarten"
    }
  }
]
```

```
        "color_code": "SXM",
        "name": "Sint Maarten",
        "continent": "Americas",
        "region": "Caribbean",
        "iso_3166_1_alpha_2_codes": 'SX',
        "french_short": "Sint Maarten"}}},

{"type": "Feature",
 "geometry": {
    "coordinates": [[[-176.20631916144734, -13.218566032523611], [-176.2248694765352, -13.233012662024095], [-176.23043457106155, -13.267921812149822], [-176.22919788338902, -13.280560096158837]],
    "type": "MultiPolygon"},
 "properties": {"geo_point_2d": {"lon": -176.1859138148507, "lat": -13.276347407901156},
    "iso3": 'WLF',
    "status": 'Overseas collectivity of France',
    "color_code": "WLF",
    "name": "Wallis and Futuna",
    "continent": "Oceania",
    "region": "Melanesia",
    "iso_3166_1_alpha_2_codes": 'WF',
    "french_short": "Wallis-et-Futuna"}}},

{"type": "Feature",
 "geometry": {
    "coordinates": [[[20.81501184605702, 43.26992240967351], [20.59053394284237, 43.20450741746396], [20.695290297675427, 43.11171597794174], [20.654136015419198, 43.078932416498986], [20.67284256]],
    "type": "MultiPolygon"},
 "properties": {"geo_point_2d": {"lon": 20.946610412739773, "lat": 42.69993248951508},
    "iso3": 'XKK',
    "status": 'autonomous province of Serbia with significant autonomy',
    "color_code": "XKK",
    "name": "Kosovo",
    "continent": "Europe",
    "region": "Southern Europe",
    "iso_3166_1_alpha_2_codes": 'XK',
    "french_short": "Kosovo"}}}

]
```

```
In [25]: # save dict as geojson file

with open('additional_boundaries.geojson','w') as f:
    json.dump(additional_geo, f)
```

```
In [26]: additional_boundaries = gpd.read_file('additional_boundaries.geojson')
```

```
In [27]: #update df_boundaries

df_boundaries = pd.concat([df_boundaries, additional_boundaries])
```

```
In [28]: # set country name as index
df_boundaries.set_index('name', inplace=True)
```

Plot boundaries on the folium map

Double check that the additional boundaries we added were successfully drawn.

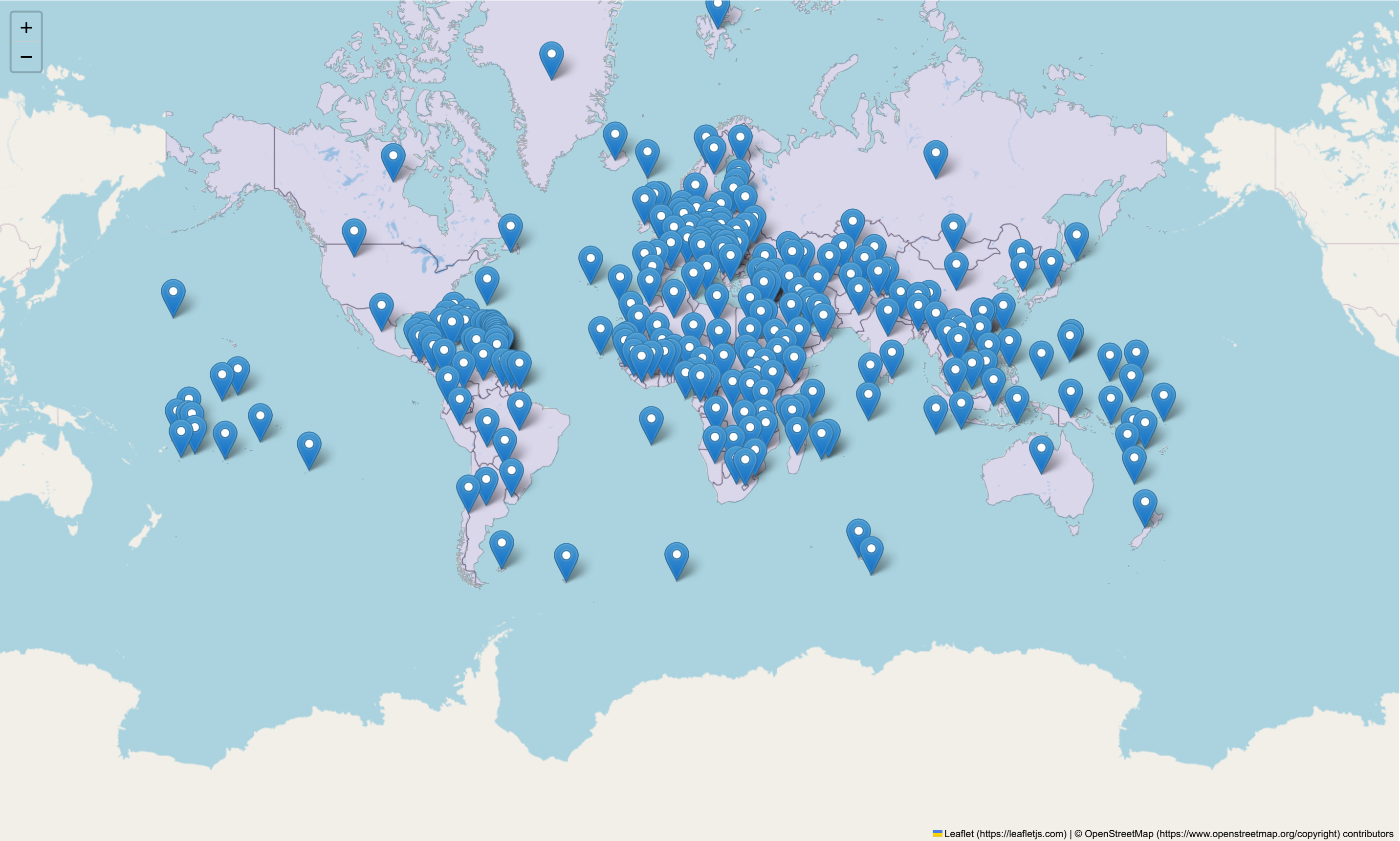
In [29]:

```
m = folium.Map(location=[0,0], tiles="OpenStreetMap", zoom_start=1.5)

folium.Choropleth(geo_data=df_boundaries,
                  fill_opacity=0.1,
                  line_opacity=0.2
                  ).add_to(m)
for idx, row in df_boundaries.iterrows():
    Marker([row['geo_point_2d']['lat'], row['geo_point_2d']['lon']], popup=row.index).add_to(m)

m
```


Out[29]:



Tops of interest

Population & Aging: DEPN, POP, MEDAGE, DEPND014, *DEPND65*

Fertility: ASFR, BIRTH, GRR, TFR, RNI

Immigration:NIM, NMR

```
In [30]: # extract year 2023 data

df_2023 = data[data['#YR']==2023][['#YR', 'COUNTRY', 'POP','MEDAGE', 'DEPND0_14', 'DEPND65_', 'GRR', 'TFR', 'RNI', 'CBR', 'CDR', 'NIM', 'NMR']].reset_index(drop=True)
```

```
In [31]: df_2023
```

Out[31]:

	#YR	COUNTRY	POP	MEDAGE	DEPND0_14	DEPND65_	GRR	TFR	RNI	CBR	CDR	NIM	NMR
0	2023	Andorra	85468	48.10	18.10	28.60	0.70	1.46	-0.11	6.87	7.98	0.00	0.00
1	2023	United Arab Emirates	9973449	35.70	19.80	2.40	0.79	1.62	0.91	10.76	1.62	-33367.00	-3.35
2	2023	Afghanistan	39232003	19.90	69.40	5.00	2.21	4.53	2.27	34.79	12.08	-3754.00	-0.10
3	2023	Antigua and Barbuda	101489	33.60	32.30	14.80	0.95	1.94	0.93	15.01	5.69	204.00	2.01
4	2023	Anguilla	19079	36.80	30.90	16.40	0.85	1.72	0.72	11.90	4.72	200.00	10.48
...
222	2023	West Bank	3176549	21.70	61.80	6.30	1.72	3.54	2.48	28.31	3.48	-12271.00	-3.86
223	2023	Yemen	31565602	21.60	57.10	5.40	1.42	2.91	1.85	24.05	5.54	-5898.00	-0.19
224	2023	South Africa	59795503	30.10	42.00	11.20	1.14	2.31	1.11	18.28	7.18	-17397.00	-0.29
225	2023	Zambia	20216029	18.20	77.60	5.00	2.21	4.49	2.85	34.48	6.02	3144.00	0.16
226	2023	Zimbabwe	16819805	21.00	66.80	6.70	1.73	3.51	2.27	29.41	6.68	-47935.00	-2.85

227 rows × 13 columns

```
In [32]: # Create a map
# m = folium.Map(location=[54, 15], tiles='openstreetmap', zoom_start=2)
# Marker([location.point.latitude, location.point.longitude]).add_to(m)
```

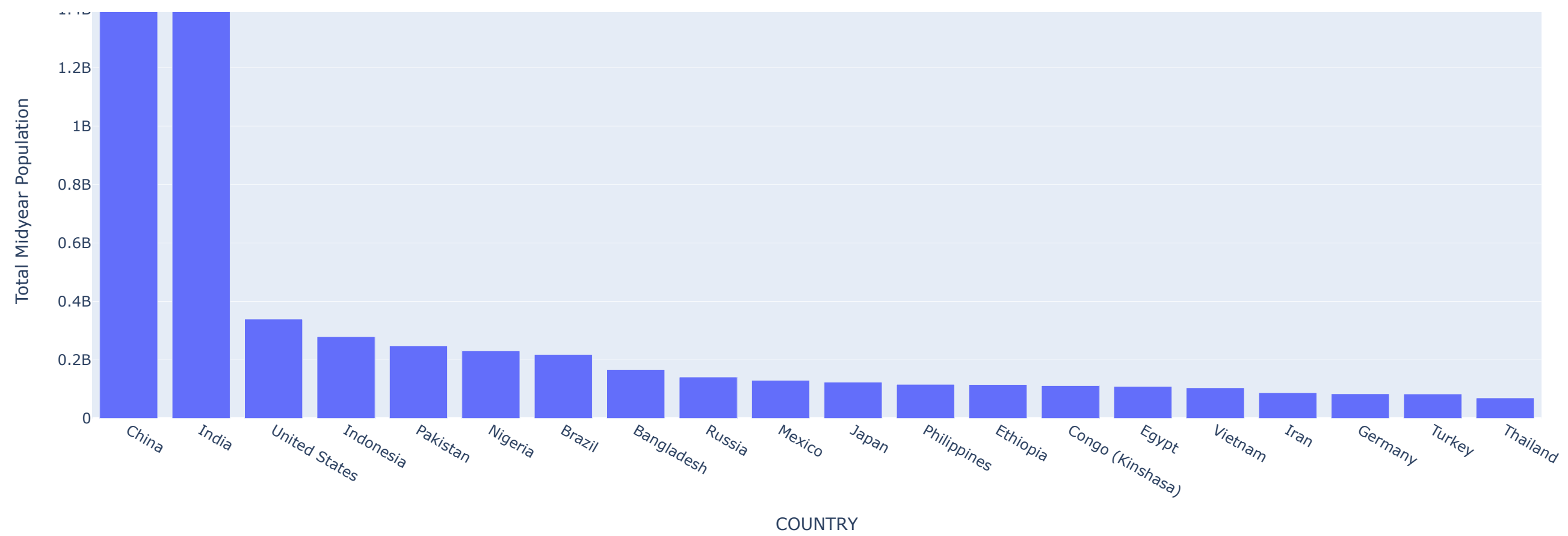
1. 1 Top 20 most populated countries in 2023

- POP Total midyear population

```
In [33]: top_20_pop = df_2023.sort_values('POP', ascending=False)[['COUNTRY', 'POP']].reset_index(drop=True).iloc[:20]
```

```
In [34]: fig = px.bar(x='COUNTRY', y='POP', data_frame=top_20_pop)
fig.update_layout(title_text="Top 20 most populated countries in 2023", title_x=0.5,
                  yaxis_title="Total Midyear Population")
```

Top 20 most populated countries in 2023



Let's see their locations

```
In [35]: df_a = pd.merge(top_20_pop, df_boundaries, left_on='COUNTRY', right_on='name', how='left')
```

```
In [36]: m_1 = folium.Map(location=[0,0], tiles="OpenStreetMap", zoom_start=1.5)

for idx, row in df_a.iterrows():
    Marker([row['geo_point_2d']['lat'], row['geo_point_2d']['lon']], popup=row['COUNTRY']).add_to(m_1)

m_1
```

Out[36]:



We can see that within TOP 20 list, there are two countries from Europe, one country from South America. Majority of the countries are located in Asia.

1.2 How has the population growth changed over time for these top 20 countries?

In [37]:

```
top_20_trend = data[data.COUNTRY.isin(top_20_pop.COUNTRY)][['#YR', 'COUNTRY', 'POP']].reset_index(drop=True)
```

```
In [38]: # pivot table
top_20_trend = top_20_trend.pivot_table(index='#YR', values='POP', columns='COUNTRY')
```

```
In [39]: top_20_trend.head()
```

Out[39]:

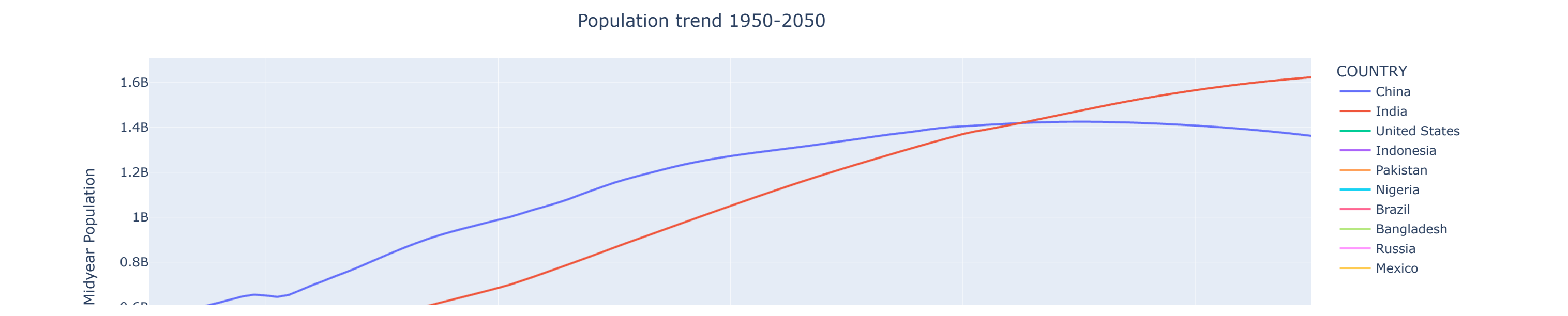
COUNTRY	Bangladesh	Brazil	China	Congo (Kinshasa)	Egypt	Ethiopia	Germany	India	Indonesia	Iran	Japan	Mexico	Nigeria	Pakistan	Philippines	Russia	Thailand
#YR																	
1950	45645964.00	53443075.00	562579779.00	13568762.00	21197691.00	20174562.00	68374572.00	369880000.00	82978392.00	16357000.00	83805000.00	28485180.00	31796939.00	40382206.00	21131264.00	101936816.00	20041628.00
1951	46149840.00	54973775.00	567159896.00	13831813.00	21704443.00	20511408.00	68875884.00	376182850.00	84113761.00	16810772.00	85163848.00	29296235.00	32492088.00	41346560.00	21736410.00	103506916.00	20653334.00
1952	46881899.00	56557783.00	574656098.00	14100005.00	22223309.00	20860941.00	69145952.00	382791319.00	85418340.00	17275640.00	86459025.00	30144317.00	33207747.00	42342412.00	22358886.00	105385090.00	21289402.00
1953	47652925.00	58197231.00	584374120.00	14373435.00	22754580.00	21223618.00	69550236.00	389691318.00	86890805.00	17747610.00	87655163.00	31031279.00	33944600.00	43372063.00	22999187.00	107302806.00	21964158.00
1954	48592988.00	59894345.00	594972939.00	14657484.00	23298551.00	21599912.00	69868115.00	396850768.00	88521458.00	18233684.00	88753892.00	31959113.00	34933877.00	44434445.00	23657826.00	109208917.00	22684974.00

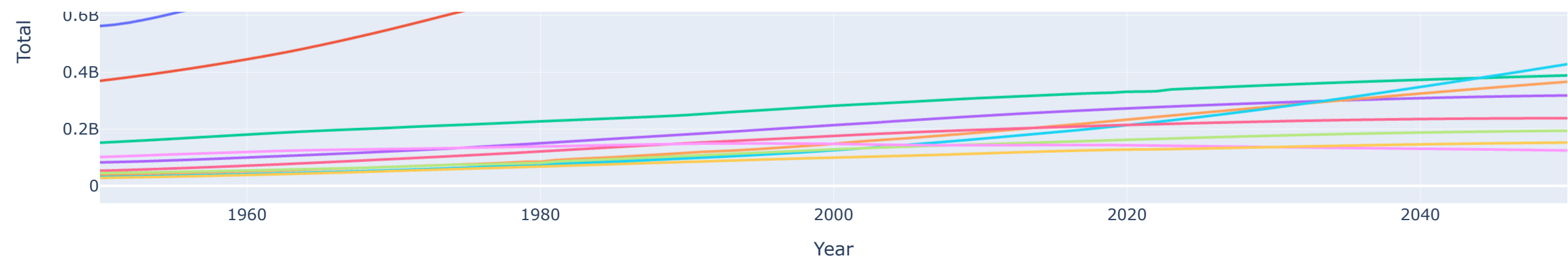
```
In [40]: # reorder the columns

col_order = top_20_pop.COUNTRY.tolist()
top_20_trend = top_20_trend.reindex(col_order, axis=1)
```

```
In [41]: def plot_line(df):
    fig = px.line(df)
    fig.update_layout(xaxis_title="Year", title_x=0.45, title_text="Population trend 1950-2050",
                      yaxis_title="Total Midyear Population")
    fig.show()
```

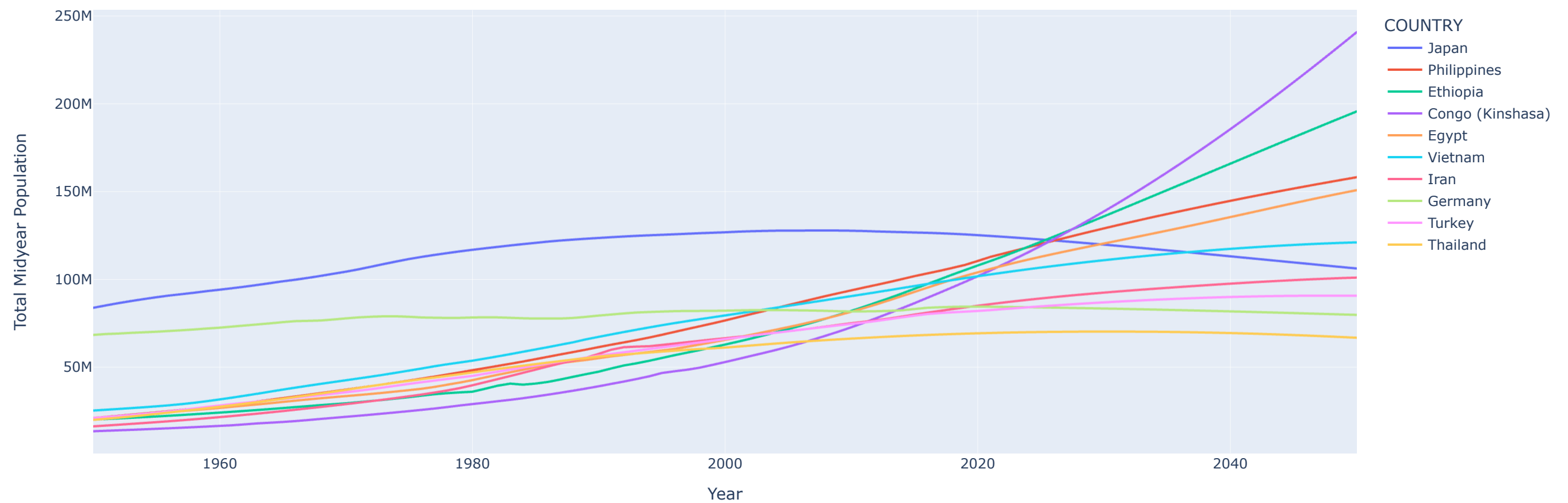
```
In [42]: plot_line(top_20_trend.iloc[:, :10])
```





```
In [43]: plot_line(top_20_trend.iloc[:,10:20])
```

Population trend 1950-2050



- The population of India will exceed China in year 2024; India will be the country with the most population in the world.
- The population of China and Japan exhibits a noticeable downward trend compared to others.

2. Rate of natural increase, crude birth rate, and crude death rate in 2023

The rate of natural increase (RNI) is a measure of how quickly a population is growing or declining.

$$(\text{Crude Birth Rate}/1,000) - (\text{Crude Death Rate}/1000) = \text{CBR} - \text{CDR} = \text{RNI}$$

NATINCR : Natural increase, both sexes

```
In [44]: df_RNI = df_2023.sort_values('CBR', ascending=False)[['COUNTRY', 'CBR', 'CDR', 'RNI', 'POP']].reset_index(drop=True )
```

```
In [45]: print(f'There are {df_RNI[df_RNI.RNI >0].count()[0]} countries with positive RNI, {df_RNI[df_RNI.RNI <=0].count()[0]} with 0 or negative RNI.')
```

There are 194 countries with positive RNI, 33 with 0 or negative RNI.

Choropleth map to show RNI around the world

```
In [46]: # Set COUNTRY as index for df_RNI

df_RNI.set_index('COUNTRY', inplace=True)
```

```
In [47]: df_RNI['RNI'].describe()
```

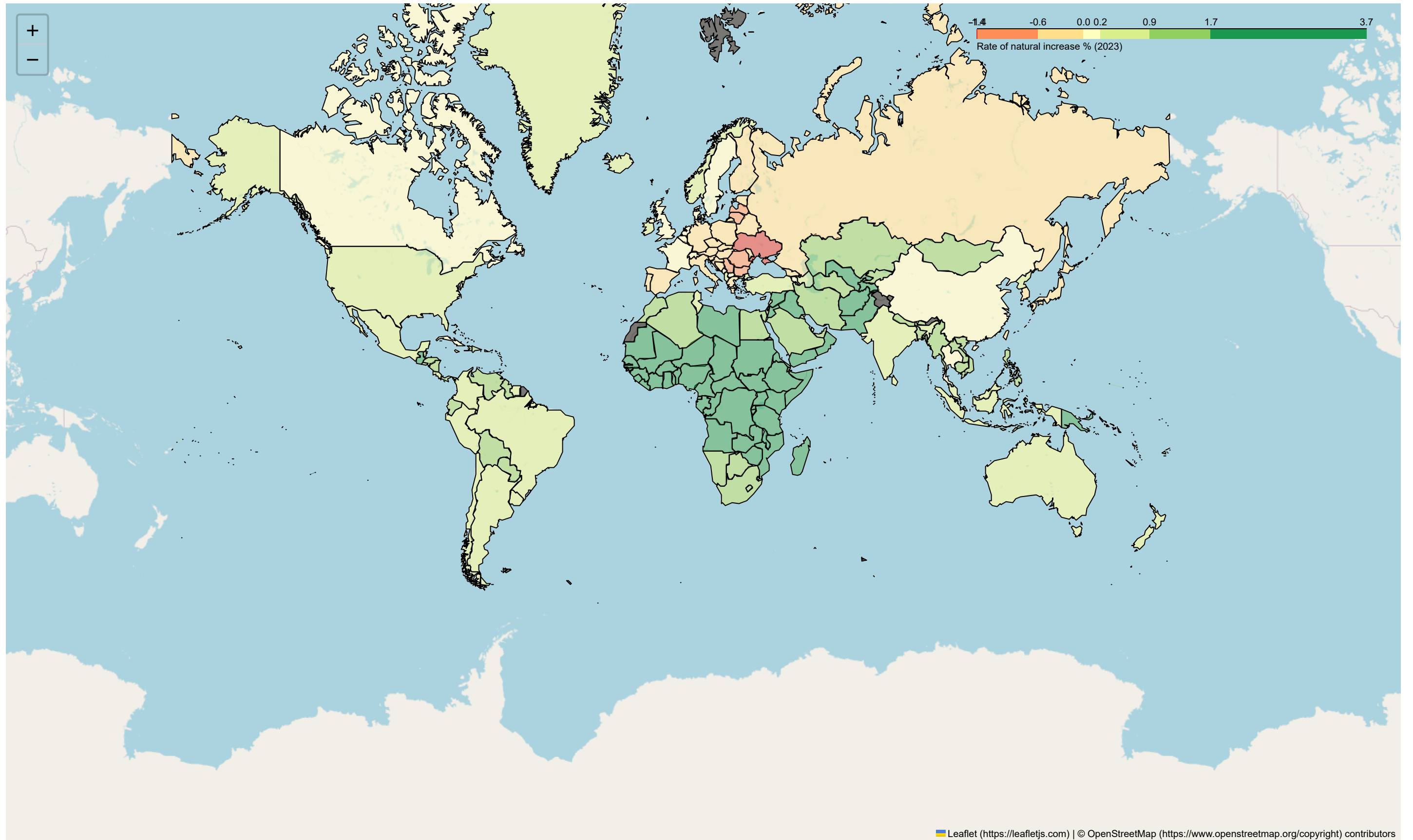
```
Out[47]: count    227.00
mean       1.00
std        1.00
min        -1.40
25%        0.22
50%        0.87
75%        1.68
max         3.72
Name: RNI, dtype: float64
```

```
In [48]: m_2 = folium.Map(location=[0,0], tiles="OpenStreetMap", zoom_start=1.5)

choropleth = Choropleth(geo_data=df_boundaries,
                        data=df_RNI['RNI'],
                        key_on="feature.id",
                        fill_color='RdYlGn',
                        bins=[-1.41, -1.39, -0.6, 0, 0.22, 0.87, 1.68, 3.73],
                        fill_opacity = 0.5,
                        highlight=True,
                        legend_name='Rate of natural increase % (2023)'
                        ).add_to(m_2)

m_2
```


Out[48]:

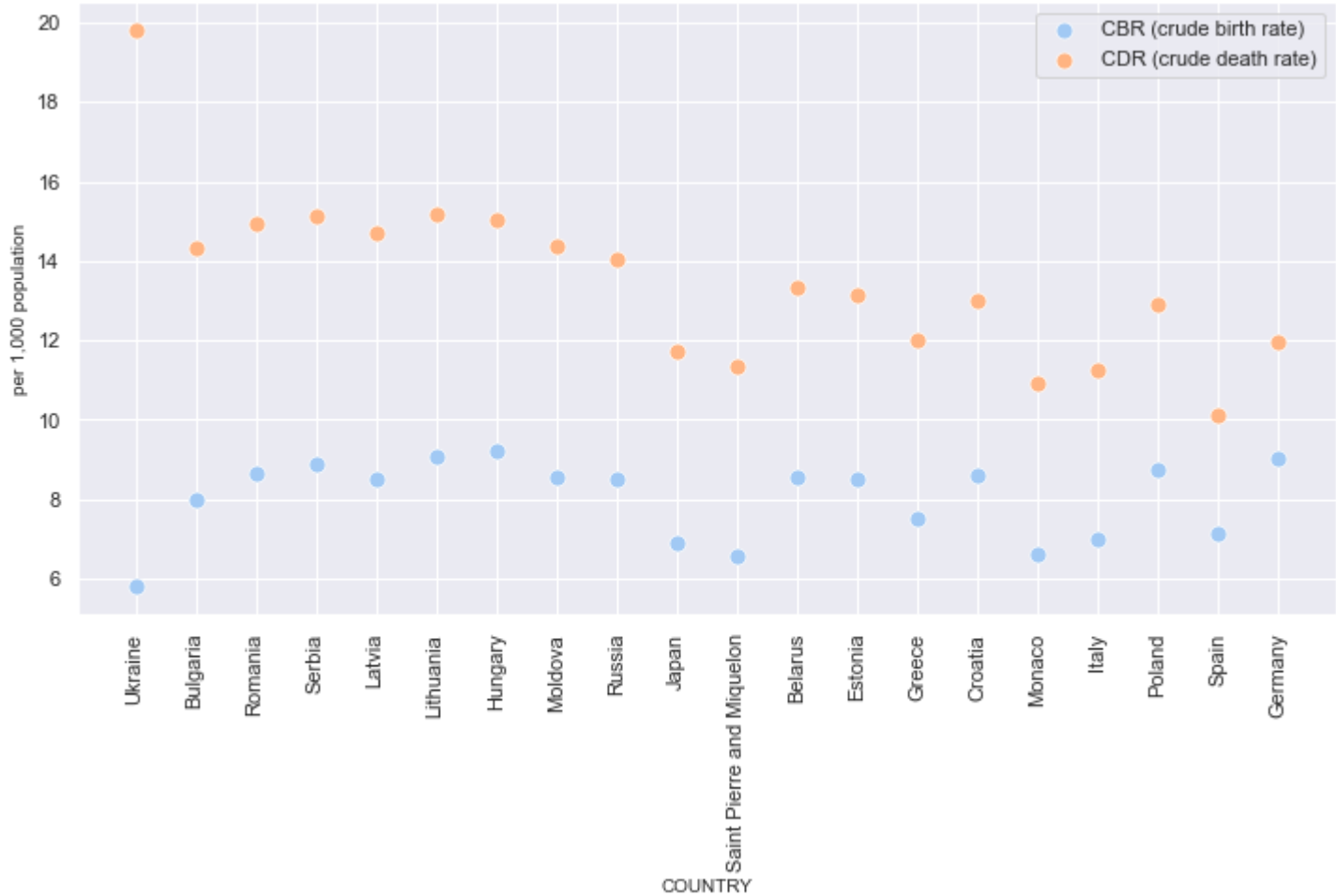


- Ukraine has the lowest RNI due to the ongoing war.
- Africa has the highest average RNI among all continents.
- Europe has the lowest average RNI among all continents.

Let's check crude birth rates, crude death rate for the 20 countries with least RNI

```
In [49]: df_RNI_20 = df_RNI.sort_values('RNI', ascending=True)[:20]
```

```
In [50]: plt.figure(figsize=(12,6))
ax = sns.scatterplot(data = df_RNI_20, x= df_RNI_20.index, y='CBR', s=80)
ax = sns.scatterplot(data = df_RNI_20, x= df_RNI_20.index, y='CDR', s=80)
_ = plt.xticks(rotation=90)
plt.legend(['CBR (crude birth rate)', 'CDR (crude death rate)'])
_ = plt.ylabel('per 1,000 population')
```



3. Population aging in 2023

MEDAGE Median age of the population, both sexes

DEPND0_14 The youth dependency ratio is the ratio of the youth population (ages 0-14) per 100 people of working age (ages 15-64). A high youth dependency ratio indicates that a greater investment needs to be made in schooling and other services for children.

DEPND65_ The elderly dependency ratio is the ratio of the elderly population (ages 65+) per 100 people of working age (ages 15-64). Increases in the elderly dependency ratio put added pressure on governments to fund pensions and healthcare.

Working age population population aged 15 to 64

source: <https://www.cia.gov/the-world-factbook/field/dependency-ratios/>

```
In [51]: df_aging = df_2023[['COUNTRY', 'MEDAGE', 'DEPND0_14', 'DEPND65_', 'POP']].reset_index(drop=True)
```

```
In [52]: df_aging.set_index('COUNTRY', inplace=True)
```

```
In [53]: df_aging[df_aging.index=='United States']
```

Out[53]:

	MEDAGE	DEPND0_14	DEPND65_	POP
COUNTRY				
United States	NaN	28.50	28.40	339665118

we don't have MEDAGE data for United States

Let's see how the MEDAGE changes over the years for United States

```
In [54]: df_medage = data[(data['#YR']<2023) & (data['COUNTRY']=='United States')][['#YR', 'MEDAGE']]
```

```
In [55]: df_medage.dropna(subset=['MEDAGE'])
```

Out[55]:

	#YR	MEDAGE
21169	2010	37.20
21170	2011	37.30
21171	2012	37.40
21172	2013	37.60
21173	2014	37.70
21174	2015	37.80
21175	2016	37.90
21176	2017	38.00
21177	2018	38.20
21178	2019	38.40
21179	2020	38.50
21180	2021	38.70
21181	2022	38.90

Let's assume the median age increased by 0.1 years to 39.0 years, in 2023.

```
In [56]: df_aging.loc['United States', 'MEDAGE'] = 39.0
```

```
In [57]: df_aging.describe().T
```

Out[57]:

	count	mean	std	min	25%	50%	75%	max
MEDAGE	227.00	32.40	9.24	15.10	24.65	32.00	40.60	56.20
DEPND0_14	227.00	40.22	19.13	15.30	25.55	33.60	49.85	104.40
DEPND65_	227.00	16.68	11.07	1.70	7.30	12.70	25.80	66.50
POP	227.00	35163080.17	137599206.28	5195.00	621830.00	5953730.00	23261072.00	1413142846.00

```
In [58]: df_aging.sort_values(by='MEDAGE', ascending=False)
```

Out[58]:

	MEDAGE	DEPND0_14	DEPND65_	POP
COUNTRY				
Monaco	56.20	17.30	66.50	31597
Saint Pierre and Miquelon	50.60	21.60	39.70	5195
Japan	49.50	21.00	50.00	123719238
Andorra	48.10	18.10	28.60	85468
Italy	48.10	18.70	36.10	61021855
...
Chad	16.50	90.20	4.90	18523165
Mali	16.30	94.30	6.20	21359722
Angola	16.20	93.50	4.60	35981281
Uganda	16.10	94.00	4.70	47729952
Niger	15.10	104.40	5.70	25396840

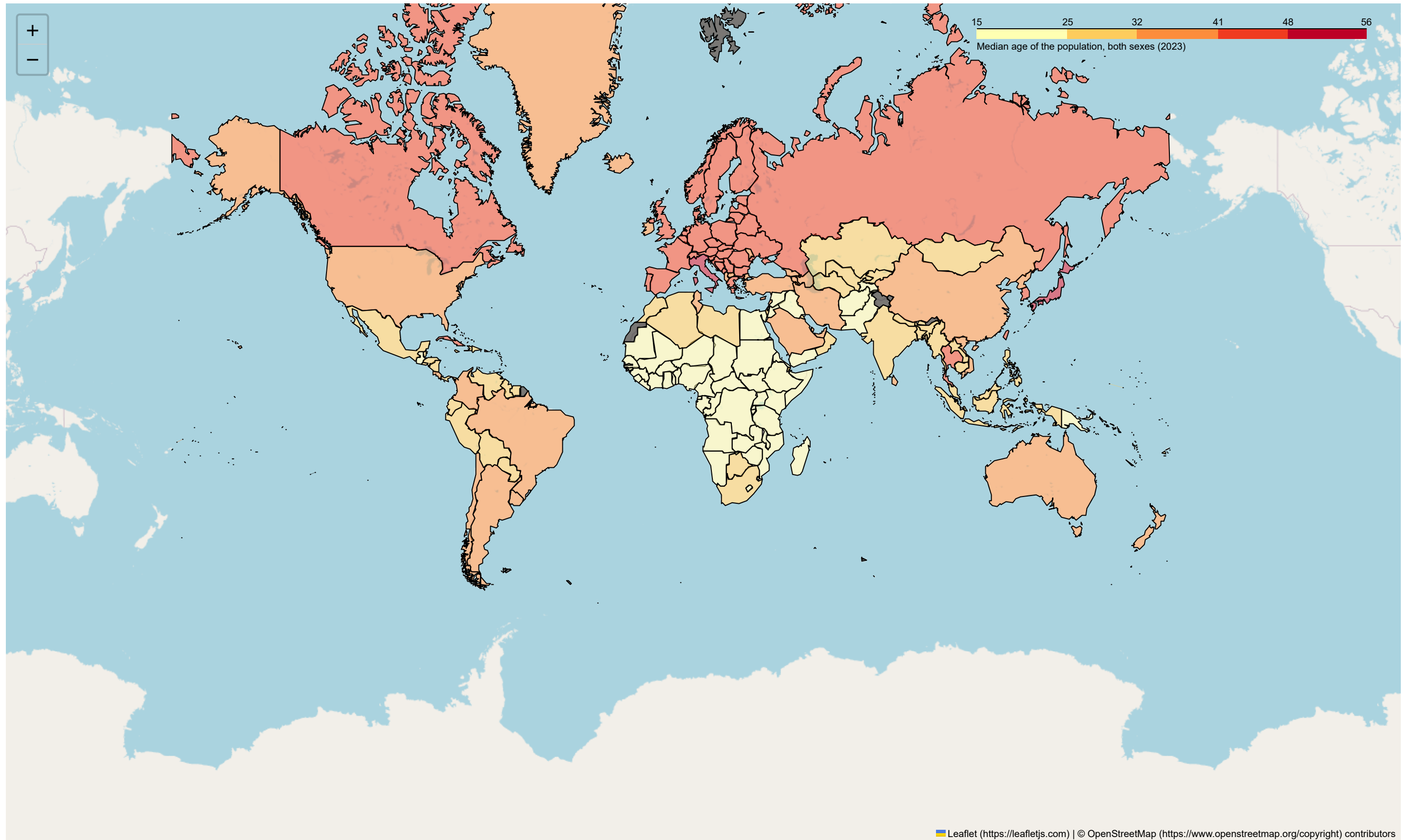
227 rows × 4 columns

```
In [59]: m_3 = folium.Map(location=[0,0], tiles="OpenStreetMap", zoom_start=1.5)

choropleth = Choropleth(geo_data=df_boundaries,
                        data=df_aging['MEDAGE'],
                        key_on="feature.id",
                        fill_color='YlOrRd',
                        bins=[15, 24.65, 32, 40.6, 48, 56.3],
                        fill_opacity = 0.5,
                        highlight=True,
                        legend_name='Median age of the population, both sexes (2023)'
                        ).add_to(m_3)

m_3
```

Out[59]:



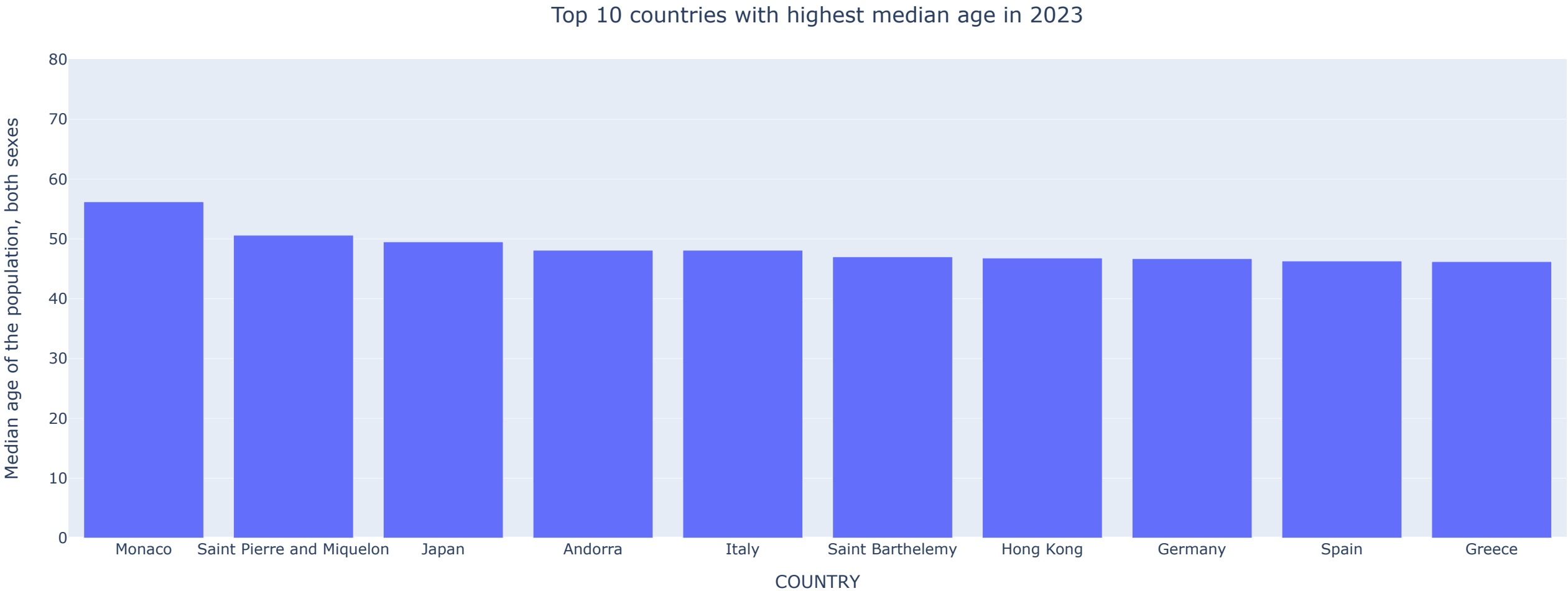
We can see that -

- Africa is the youngest continent in the world
- Almost the whole Europe has median age > 40
- East Asia and Southern Europe has highest median age.

Let's plot 10 countries with highest MEDAGE, and 10 countries with lowest MEDAGE.

```
In [60]: df_aging_old = df_aging.sort_values(by='MEDAGE', ascending=False).reset_index()[:10]

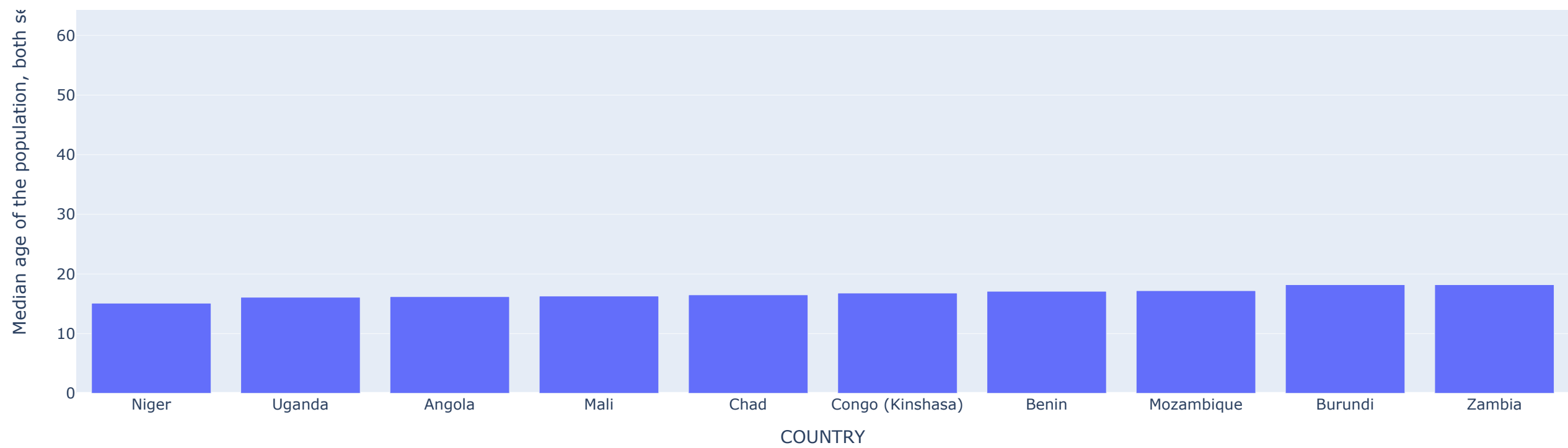
fig = px.bar(x='COUNTRY', y='MEDAGE', data_frame=df_aging_old)
fig.update_layout(title_text="Top 10 countries with highest median age in 2023", title_x=0.5,
                  yaxis_title="Median age of the population, both sexes",
                  yaxis_range=[0,80])
```



```
In [61]: df_aging_young = df_aging.sort_values(by='MEDAGE', ascending=True).reset_index()[:10]

fig = px.bar(x='COUNTRY', y='MEDAGE', data_frame=df_aging_young)
fig.update_layout(title_text="Top 10 countries with lowest median age in 2023", title_x=0.5,
                  yaxis_title="Median age of the population, both sexes",
                  yaxis_range=[0,80])
```





Let's take a look at the dependency ratio.

A high total dependency ratio indicates that the working-age population and the overall economy face a greater burden to support and provide social services for youth and elderly persons, who are often economically dependent.

source: <https://www.cia.gov/the-world-factbook/field/dependency-ratios/>

```
In [62]: df_aging['DEPN'] = df_aging['DEPND0_14'] + df_aging['DEPND65_']

In [63]: df_aging['DEPN'].describe()

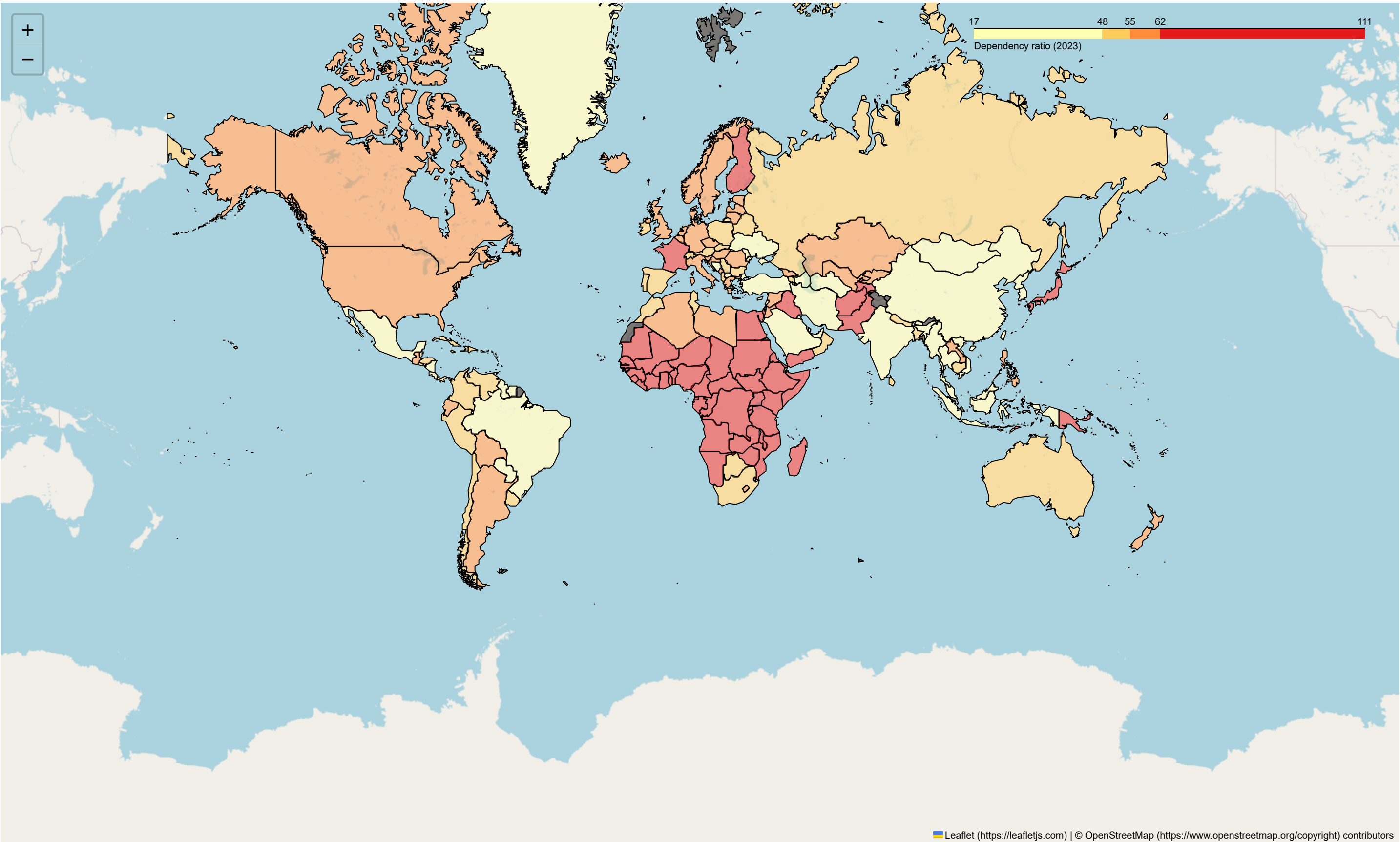
Out[63]: count    227.00
mean       56.90
std        13.92
min        17.00
25%        47.90
50%        54.50
75%        61.80
max        110.10
Name: DEPN, dtype: float64

In [64]: m_4 = folium.Map(location=[0,0], tiles="OpenStreetMap", zoom_start=1.5)

choropleth = Choropleth(geo_data=df_boundaries,
                        data=df_aging['DEPN'],
                        key_on="feature.id",
                        fill_color='YlOrRd',
                        bins=[16.9, 47.9, 54.5, 61.8, 111],
                        fill_opacity = 0.5,
                        highlight=True,
                        legend_name='Dependency ratio (2023)'
                        ).add_to(m_4)

m_4
```

Out[64]:



```
In [65]: df_aging['more_young'] = df_aging['DEPND0_14'] - df_aging['DEPND65_']
```

```
In [66]: df_aging.sort_values(by='more_young', ascending=True)[:20]
```

Out[66]:

	MEDAGE	DEPND0_14	DEPND65_	POP	DEPN	more_young
COUNTRY						
Monaco	56.20	17.30	66.50	31597	83.80	-49.20
Japan	49.50	21.00	50.00	123719238	71.00	-29.00
Saint Pierre and Miquelon	50.60	21.60	39.70	5195	61.30	-18.10
Puerto Rico	45.60	20.40	38.40	3057311	58.80	-18.00
Italy	48.10	18.70	36.10	61021855	54.80	-17.40
Germany	46.70	21.80	37.00	84220184	58.80	-15.20
Greece	46.20	22.40	37.20	10497595	59.60	-14.80
Portugal	46.00	19.70	33.70	10223150	53.40	-14.00
Croatia	44.80	22.00	35.70	4169239	57.70	-13.70
Slovenia	45.90	23.20	36.20	2099790	59.40	-13.00
Malta	43.20	23.10	36.00	467138	59.10	-12.90
Saint Barthelemy	47.00	22.10	34.70	7093	56.80	-12.60
Ukraine	45.30	17.40	29.40	34831102	46.80	-12.00
Hong Kong	46.80	19.80	31.70	7288167	51.50	-11.90
Finland	43.20	26.90	38.40	5614571	65.30	-11.50
Spain	46.30	20.20	31.00	47222613	51.20	-10.80
Latvia	45.20	23.50	34.30	1821750	57.80	-10.80
Estonia	44.70	25.00	35.60	1202762	60.60	-10.60
Romania	45.10	25.20	35.80	18326327	61.00	-10.60
Andorra	48.10	18.10	28.60	85468	46.70	-10.50

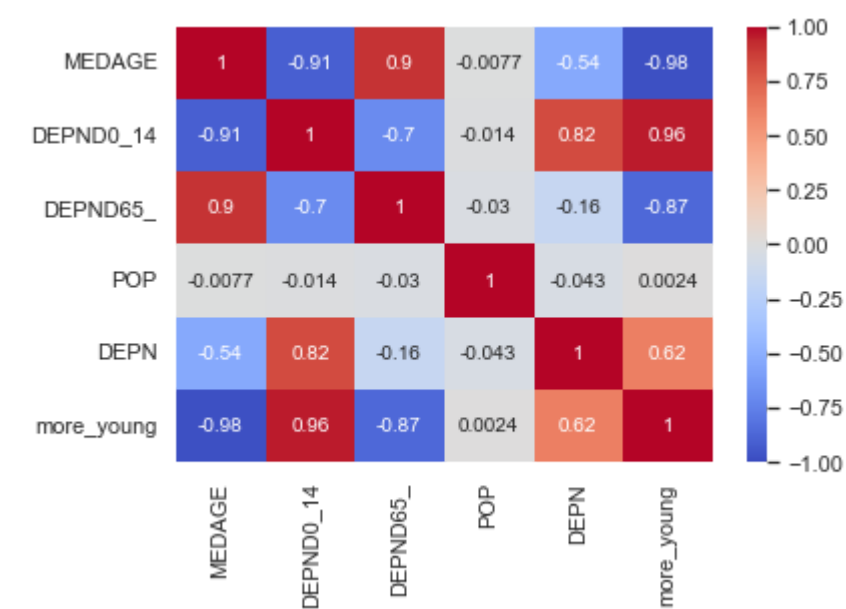
Besides vacation/resort places, Japan and Hong Kong, most of the countries in this top 20 list located in Europe.

In [67]:

```
sns.heatmap(df_aging.corr(), vmin=-1, vmax=1, cmap='coolwarm', annot=True)
```

Out[67]:

<Axes: >



MEDAGE and DEPN are highly correlated.

4. Fertility rate in 2023

Total fertility rate (TFR) is the total number of children a woman would bear during her lifetime if she were to experience the prevailing age-specific fertility rates of women and survive until the end of her reproductive life.

```
In [71]: df_fertility = df_2023[['COUNTRY', 'GRR', 'TFR']]
```

```
In [72]: df_fertility.set_index('COUNTRY', inplace=True)
```

```
In [73]: df_fertility.describe().T
```

```
Out[73]:
```

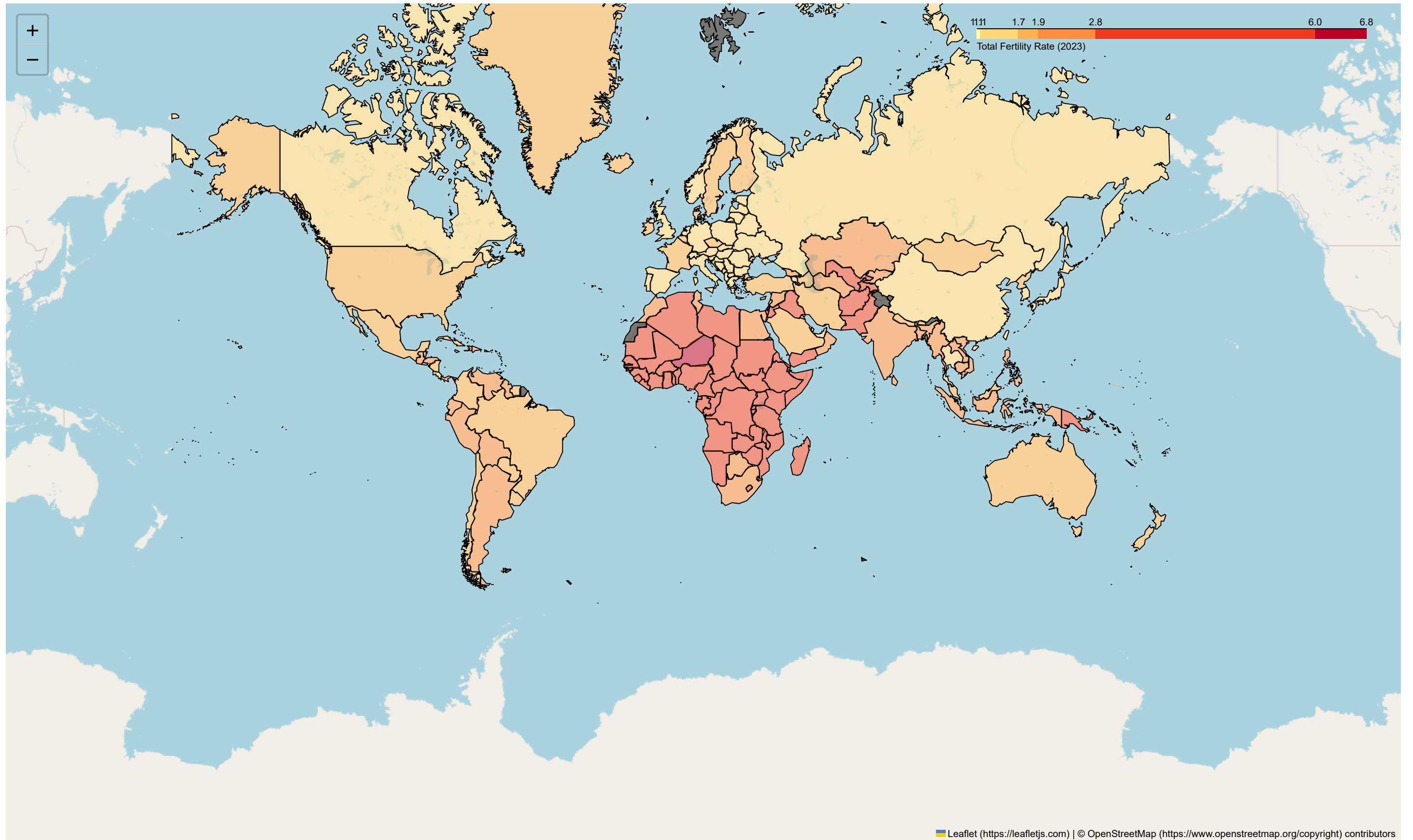
	count	mean	std	min	25%	50%	75%	max
GRR	227.00	1.17	0.54	0.53	0.80	0.95	1.36	3.31
TFR	227.00	2.39	1.08	1.09	1.66	1.95	2.79	6.73

```
In [76]: m_5 = folium.Map(location=[0,0], tiles="OpenStreetMap", zoom_start=1.5)

choropleth = Choropleth(geo_data=df_boundaries,
                        data=df_fertility['TFR'],
                        key_on="feature.id",
                        fill_color='YlOrRd',
                        bins=[1.05, 1.1, 1.66,1.95,2.79,6,6.75],
                        fill_opacity = 0.5,
                        highlight=True,
                        legend_name='Total Fertility Rate (2023)'
                        ).add_to(m_5)

m_5
```

Out[76]:



The average TFR of 2023 is 2.3.

- Africa shows much higher TFR than the world average value.
- Mid east and southern Asia have high TFR too.

In [78]:

df_fertility.sort_values(by='TFR', ascending=True)[:20]

Out[78]:

	GRR	TFR
COUNTRY		
Taiwan	0.53	1.09
Korea, South	0.54	1.11
Singapore	0.57	1.17
Ukraine	0.59	1.22
Hong Kong	0.60	1.23
Macau	0.60	1.23
Italy	0.60	1.24
Moldova	0.60	1.25
Puerto Rico	0.61	1.25
Spain	0.63	1.29
Poland	0.64	1.31
Montserrat	0.62	1.32
Mauritius	0.65	1.35
Virgin Islands, British	0.67	1.37
Bosnia and Herzegovina	0.66	1.37
Japan	0.68	1.39
Costa Rica	0.68	1.40
Greece	0.68	1.40
Portugal	0.70	1.44
Bahamas, The	0.71	1.44

Again, Southern Europe and Ease Asia show low TFR.