In [1]:
```
# 1) Using the Web scraping, read the list of the Sovereign States from
# https://en.wikipedia.org/wiki/List_of_sovereign_states
```

In [ ]:
```
# Creating an empty dataframe with the desired columns
```

In [2]:
```
import pandas as pd
```

In [3]:
```
columns = ['Country', 'Link', 'Description','Status', 'Capital','Total area' ,'Water
df = pd.DataFrame(columns=columns)
```

In [4]:
```
import requests
from bs4 import BeautifulSoup
import re
```

In [5]:
```
# Making a GET request to the Wikipedia page
url = 'https://en.wikipedia.org/wiki/List_of_sovereign_states'
response = requests.get(url)
```

In [6]:
```
# Parsing the HTML content using BeautifulSoup
soup = BeautifulSoup(response.content, 'html.parser')
```

In [7]:
```
# Finding the table containing the list of sovereign states
table = soup.find(class_ = 'sortable wikitable')
```

In [8]:
```
# Extracting the names of the countries and their corresponding href links
countries = {}
for row in table.find_all('tr')[3:]:
    cells = row.find_all('td')

    names = cells[0].text
    name = re.split(r"\s*[→—]\s*", names)[0]
    name = name.split(",")[0]
    name = name.split("[")[0]
    name = name.replace("ZZZ","").replace("\xa0","").replace("\n","")

    link = cells[0].find('a')
    if link is not None:
            link = link['href']
            countries[name] = link


countries = {k: v for k, v in countries.items() if not v.startswith('#')}
del countries['↑ UN member states and General Assembly observer states ↑']
print(countries)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
{'Abkhazia': '/wiki/Abkhazia', 'Afghanistan': '/wiki/Afghanistan', 'Albania': '/wiki/A
lbania', 'Algeria': '/wiki/Algeria', 'Andorra': '/wiki/Andorra', 'Angola': '/wiki/Ango
la', 'Antigua and Barbuda': '/wiki/Antigua_and_Barbuda', 'Argentina': '/wiki/Argentin
a', 'Armenia': '/wiki/Armenia', 'Artsakh': '/wiki/Republic_of_Artsakh', 'Australia':
'/wiki/Australia', 'Austria': '/wiki/Austria', 'Azerbaijan': '/wiki/Azerbaijan', 'Baha
mas': '/wiki/The_Bahamas', 'Bahrain': '/wiki/Bahrain', 'Bangladesh': '/wiki/Banglades
h', 'Barbados': '/wiki/Barbados', 'Belarus': '/wiki/Belarus', 'Belgium': '/wiki/Belgiu
m', 'Belize': '/wiki/Belize', 'Benin': '/wiki/Benin', 'Bhutan': '/wiki/Bhutan', 'Boliv
ia': '/wiki/Bolivia', 'Bosnia and Herzegovina': '/wiki/Bosnia_and_Herzegovina', 'Botsw
ana': '/wiki/Botswana', 'Brazil': '/wiki/Brazil', 'Brunei': '/wiki/Brunei', 'Bulgari
a': '/wiki/Bulgaria', 'Burkina Faso': '/wiki/Burkina_Faso', 'Burundi': '/wiki/Burund
i', 'Cambodia': '/wiki/Cambodia', 'Cameroon': '/wiki/Cameroon', 'Canada': '/wiki/Canad
a', 'Cape Verde': '/wiki/Cape_Verde', 'Central African Republic': '/wiki/Central_Afric
an_Republic', 'Chad': '/wiki/Chad', 'Chile': '/wiki/Chile', 'Colombia': '/wiki/Colombi
a', 'Comoros': '/wiki/Comoros', 'Congo': '/wiki/Republic_of_the_Congo', 'Cook Island
s': '/wiki/Cook_Islands', 'Costa Rica': '/wiki/Costa_Rica', 'Croatia': '/wiki/Croati
a', 'Cuba': '/wiki/Cuba', 'Cyprus': '/wiki/Cyprus', 'Czech Republic': '/wiki/Czech_Rep
ublic', 'Denmark': '/wiki/Danish_Realm', 'Djibouti': '/wiki/Djibouti', 'Dominica': '/w
iki/Dominica', 'Dominican Republic': '/wiki/Dominican_Republic', 'East Timor': '/wiki/
East_Timor', 'Ecuador': '/wiki/Ecuador', 'Egypt': '/wiki/Egypt', 'El Salvador': '/wik
i/El_Salvador', 'Equatorial Guinea': '/wiki/Equatorial_Guinea', 'Eritrea': '/wiki/Erit
rea', 'Estonia': '/wiki/Estonia', 'Eswatini': '/wiki/Eswatini', 'Ethiopia': '/wiki/Eth
iopia', 'Fiji': '/wiki/Fiji', 'Finland': '/wiki/Finland', 'France': '/wiki/France', 'G
abon': '/wiki/Gabon', 'Gambia': '/wiki/The_Gambia', 'Georgia': '/wiki/Georgia_(countr
y)', 'Germany': '/wiki/Germany', 'Ghana': '/wiki/Ghana', 'Greece': '/wiki/Greece', 'Gr
enada': '/wiki/Grenada', 'Guatemala': '/wiki/Guatemala', 'Guinea': '/wiki/Guinea', 'Gu
inea-Bissau': '/wiki/Guinea-Bissau', 'Guyana': '/wiki/Guyana', 'Haiti': '/wiki/Haiti',
'Honduras': '/wiki/Honduras', 'Hungary': '/wiki/Hungary', 'Iceland': '/wiki/Iceland',
'India': '/wiki/India', 'Indonesia': '/wiki/Indonesia', 'Iran': '/wiki/Iran', 'Iraq':
'/wiki/Iraq', 'Ireland': '/wiki/Republic_of_Ireland', 'Israel': '/wiki/Israel', 'Ital
y': '/wiki/Italy', 'Ivory Coast': '/wiki/Ivory_Coast', 'Jamaica': '/wiki/Jamaica', 'Ja
pan': '/wiki/Japan', 'Jordan': '/wiki/Jordan', 'Kazakhstan': '/wiki/Kazakhstan', 'Keny
a': '/wiki/Kenya', 'Kiribati': '/wiki/Kiribati', 'Korea': '/wiki/South_Korea', 'Kosov
o': '/wiki/Kosovo', 'Kuwait': '/wiki/Kuwait', 'Kyrgyzstan': '/wiki/Kyrgyzstan', 'Lao
s': '/wiki/Laos', 'Latvia': '/wiki/Latvia', 'Lebanon': '/wiki/Lebanon', 'Lesotho': '/w
iki/Lesotho', 'Liberia': '/wiki/Liberia', 'Libya': '/wiki/Libya', 'Liechtenstein': '/w
iki/Liechtenstein', 'Lithuania': '/wiki/Lithuania', 'Luxembourg': '/wiki/Luxembourg',
'Madagascar': '/wiki/Madagascar', 'Malawi': '/wiki/Malawi', 'Malaysia': '/wiki/Malaysi
a', 'Maldives': '/wiki/Maldives', 'Mali': '/wiki/Mali', 'Malta': '/wiki/Malta', 'Marsh
all Islands': '/wiki/Marshall_Islands', 'Mauritania': '/wiki/Mauritania', 'Mauritius':
'/wiki/Mauritius', 'Mexico': '/wiki/Mexico', 'Micronesia': '/wiki/Federated_States_of_
Micronesia', 'Moldova': '/wiki/Moldova', 'Monaco': '/wiki/Monaco', 'Mongolia': '/wiki/
Mongolia', 'Montenegro': '/wiki/Montenegro', 'Morocco': '/wiki/Morocco', 'Mozambique':
'/wiki/Mozambique', 'Myanmar': '/wiki/Myanmar', 'Namibia': '/wiki/Namibia', 'Nauru':
'/wiki/Nauru', 'Nepal': '/wiki/Nepal', 'Netherlands': '/wiki/Kingdom_of_the_Netherland
s', 'New Zealand': '/wiki/New_Zealand', 'Nicaragua': '/wiki/Nicaragua', 'Niger': '/wik
i/Niger', 'Nigeria': '/wiki/Nigeria', 'Niue': '/wiki/Niue', 'North Macedonia': '/wiki/
North_Macedonia', 'Northern Cyprus': '/wiki/Northern_Cyprus', 'Norway': '/wiki/Norwa
y', 'Oman': '/wiki/Oman', 'Pakistan': '/wiki/Pakistan', 'Palau': '/wiki/Palau', 'Pales
tine': '/wiki/State_of_Palestine', 'Panama': '/wiki/Panama', 'Papua New Guinea': '/wik
i/Papua_New_Guinea', 'Paraguay': '/wiki/Paraguay', 'Peru': '/wiki/Peru', 'Philippine
s': '/wiki/Philippines', 'Poland': '/wiki/Poland', 'Portugal': '/wiki/Portugal', 'Qata
r': '/wiki/Qatar', 'Romania': '/wiki/Romania', 'Russia': '/wiki/Russia', 'Rwanda': '/w
iki/Rwanda', 'Sahrawi Arab Democratic Republic': '/wiki/Sahrawi_Arab_Democratic_Republ
ic', 'Saint Kitts and Nevis': '/wiki/Saint_Kitts_and_Nevis', 'Saint Lucia': '/wiki/Sai
nt_Lucia', 'Saint Vincent and the Grenadines': '/wiki/Saint_Vincent_and_the_Grenadine
s', 'Samoa': '/wiki/Samoa', 'San Marino': '/wiki/San_Marino', 'São Tomé and Príncipe':
'/wiki/S%C3%A3o_Tom%C3%A9_and_Pr%C3%ADncipe', 'Saudi Arabia': '/wiki/Saudi_Arabia', 'S
enegal': '/wiki/Senegal', 'Serbia': '/wiki/Serbia', 'Seychelles': '/wiki/Seychelles',
'Sierra Leone': '/wiki/Sierra_Leone', 'Singapore': '/wiki/Singapore', 'Slovakia': '/wi
ki/Slovakia', 'Slovenia': '/wiki/Slovenia', 'Solomon Islands': '/wiki/Solomon_Island
s', 'Somalia': '/wiki/Somalia', 'Somaliland': '/wiki/Somaliland', 'South Africa': '/wi
ki/South_Africa', 'South Ossetia': '/wiki/South_Ossetia', 'South Sudan': '/wiki/South_
Sudan', 'Spain': '/wiki/Spain', 'Sri Lanka': '/wiki/Sri_Lanka', 'Suriname': '/wiki/Sur
iname', 'Sweden': '/wiki/Sweden', 'Switzerland': '/wiki/Switzerland', 'Syria': '/wiki/
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Syria', 'Tajikistan': '/wiki/Tajikistan', 'Tanzania': '/wiki/Tanzania', 'Thailand': '/
wiki/Thailand', 'Togo': '/wiki/Togo', 'Tonga': '/wiki/Tonga', 'Transnistria': '/wiki/T
ransnistria', 'Trinidad and Tobago': '/wiki/Trinidad_and_Tobago', 'Tunisia': '/wiki/Tu
nisia', 'Turkey': '/wiki/Turkey', 'Turkmenistan': '/wiki/Turkmenistan', 'Tuvalu': '/wi
ki/Tuvalu', 'Uganda': '/wiki/Uganda', 'Ukraine': '/wiki/Ukraine', 'United Arab Emirate
s': '/wiki/United_Arab_Emirates', 'United Kingdom': '/wiki/United_Kingdom', 'United St
ates': '/wiki/United_States', 'Uruguay': '/wiki/Uruguay', 'Uzbekistan': '/wiki/Uzbekis
tan', 'Vanuatu': '/wiki/Vanuatu', 'Vatican City': '/wiki/Vatican_City', 'Venezuela':
'/wiki/Venezuela', 'Vietnam': '/wiki/Vietnam', 'Yemen': '/wiki/Yemen', 'Zambia': '/wik
i/Zambia', 'Zimbabwe': '/wiki/Zimbabwe', 'Taiwan': '/wiki/Taiwan'}
```

In [13]:

```
# Loop over the dictionary keys and values
for key, value in countries.items():
#import itertools

#for key, value in itertools.islice(countries.items(), 2):
    # Make a GET request to the website

    country_request = requests.get(('https://en.wikipedia.org' + value))

    website_soup = BeautifulSoup(country_request.content, 'html.parser')

    # Find the "mw-parser-output" class
    mw_content_container = website_soup.find('div', class_='mw-content-container')
    first_h2 = mw_content_container.find('h2')
    paragraphs = []
    sibling = first_h2.previous_sibling

    while sibling is not None and sibling.name != 'h2':
        if sibling.name == 'p':
            paragraphs.append(sibling)
        sibling = sibling.previous_sibling

    description = ''
    # Extract the text content of each paragraph
    for paragraph in reversed(paragraphs):
        text = paragraph.text.strip()
        description += text

# Initialize variables for all the columns you want to extract
    status = ''
    capital = ''
    languages = ''
    ethnic_groups = ''
    religion = ''
    demonym = ''
    government = ''
    hdi = ''
    currency = ''
    time_zone = ''
    driving_side = ''
    calling_code = ''
    iso_code = ''
    internet_tld = ''
    total_area = ''
    water = ''
    population_density =''
    population = ''
    GDPppp = ''
    GDPppp_pc = ''
    GDPnominal = ''
    GDPnominal_pc = ''
    date_format = ''
    gini = ''
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
 # Find the "mw-parser-output" class
    infobox_table = website_soup.find('table', class_='infobox ib-country vcard')

# Find all table rows (tr) in the infobox table
    rows = infobox_table.find_all('tr')

# Loop over each row in the info table
    for row in infobox_table.find_all('tr'):
        header = row.find('th', class_='infobox-label')
        data = row.find('td', class_='infobox-data')

# Check the header and extract data for the corresponding column
        if header and 'Status' in header.text:
            status = data.text.strip()
        elif header and 'Date format' in header.text:
            date_format = data.text.strip()
        elif header and 'Gini' in header.text:
            gini = data.text.strip()
        elif header and 'Water' in header.text:
            water = data.text.strip()
            previous_row = row.find_previous_sibling('tr')
            previous_row_cells = previous_row.find_all(['th', 'td'])
            total_area = previous_row_cells[1].text.strip()
        elif header and 'Density' in header.text:
            population_density = data.text.strip()
            previous_row = row.find_previous_sibling('tr')
            previous_row_cells = previous_row.find_all(['th', 'td'])
            population = previous_row_cells[1].text.strip()
        elif header and 'GDP' in header.text and 'PPP' in header.text:
            next_row = row.find_next_sibling('tr')
            next_row_cells = next_row.find_all(['th', 'td'])
            GDPppp = next_row_cells[1].text.strip()
            nnext_row = next_row.find_next_sibling('tr')
            nnext_row_cells = nnext_row.find_all(['th', 'td'])
            GDPppp_pc = nnext_row_cells[1].text.strip()
        elif header and 'GDP' in header.text and 'nominal' in header.text:
            next_row = row.find_next_sibling('tr')
            next_row_cells = next_row.find_all(['th', 'td'])
            GDPnominal = next_row_cells[1].text.strip()
            nnext_row = next_row.find_next_sibling('tr')
            nnext_row_cells = nnext_row.find_all(['th', 'td'])
            GDPnominal_pc = nnext_row_cells[1].text.strip()
        elif header and 'Capital' in header.text:
            capital = data.text.strip()
        elif header and 'language' in header.text:
            languages = data.text.strip()
        elif header and 'Ethnic' in header.text:
            ethnic_groups = data.text.strip()
        elif header and 'Religion' in header.text:
            religion = data.text.strip()
        elif header and 'Demonym' in header.text:
            demonym = data.text.strip()
        elif header and 'Government' in header.text:
            government = data.text.strip()
        elif header and 'HDI' in header.text:
            hdi = data.text.strip()
        elif header and 'Currency' in header.text:
            currency = data.text.strip()
        elif header and 'Time' in header.text:
            time_zone = data.text.strip()
        elif header and 'Driving' in header.text:
            driving_side = data.text.strip()
        elif header and 'Calling' in header.text:
            calling_code = data.text.strip()
        elif header and 'ISO' in header.text:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
            iso_code = data.text.strip()
        elif header and 'Internet' in header.text:
            internet_tld = data.text.strip()

# Append the extracted information to the dataframe
    df = df.append({
            'Country': key,
            'Link': 'https://en.wikipedia.org' + value,
            'Status': status,
            'Capital': capital,
            'Official languages': languages,
            'Ethnic groups': ethnic_groups,
            'Religion': religion,
            'Demonym': demonym,
            'Government': government,
            'HDI': hdi,
            'Currency': currency,
            'Time zone': time_zone,
            'Driving side': driving_side,
            'Calling code': calling_code,
            'ISO 3166 code': iso_code,
            'Internet TLD': internet_tld,
            'Total area' : total_area,
            'Water area %' : water,
            'Population Density' : population_density,
            'Population - 2022 estimate' : population,
            'GDP (PPP) - total' : GDPppp,
            'GDP (PPP) - per capita' : GDPppp_pc,
            'GDP (nominal) - total' : GDPnominal,
            'GDP (nominal) - per capita' : GDPnominal_pc,
            'Date format' : date_format,
            'Gini' : gini,
            'Country': key,
            'Link': 'https://en.wikipedia.org' + value,
            'Description': description

        }, ignore_index=True)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[13], line 125
    122             internet_tld = data.text.strip()
    124 # Append the extracted information to the dataframe
--> 125     df = df.append({
    126             'Country': key,
    127             'Link': 'https://en.wikipedia.org' + value,
    128             'Status': status,
    129             'Capital': capital,
    130             'Official languages': languages,
    131             'Ethnic groups': ethnic_groups,
    132             'Religion': religion,
    133             'Demonym': demonym,
    134             'Government': government,
    135             'HDI': hdi,
    136             'Currency': currency,
    137             'Time zone': time_zone,
    138             'Driving side': driving_side,
    139             'Calling code': calling_code,
    140             'ISO 3166 code': iso_code,
    141             'Internet TLD': internet_tld,
    142             'Total area' : total_area,
    143             'Water area %' : water,
    144             'Population Density' : population_density,
    145             'Population - 2022 estimate' : population,
    146             'GDP (PPP) - total' : GDPppp,
    147             'GDP (PPP) - per capita' : GDPppp_pc,
    148             'GDP (nominal) - total' : GDPnominal,
    149             'GDP (nominal) - per capita' : GDPnominal_pc,
    150             'Date format' : date_format,
    151             'Gini' : gini,
    152             'Country': key,
    153             'Link': 'https://en.wikipedia.org' + value,
    154             'Description': description
    155 
    156         }, ignore_index=True)

File /Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages/p
andas/core/generic.py:5989, in NDFrame.__getattr__(self, name)
   5982 if (
   5983     name not in self._internal_names_set
   5984     and name not in self._metadata
   5985     and name not in self._accessors
   5986     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5987 ):
   5988     return self[name]
-> 5989 return object.__getattribute__(self, name)

AttributeError: 'DataFrame' object has no attribute 'append'
```

In [15]:
```
df
```

Out[15]:

Country	Link	Description	Status	Capital	Total area	Water area %	Population Density	Population - 2022 estimate	Official languages

0 rows × 27 columns

In [16]:
```
df.to_csv('countries_data.csv', index=False)
```

In [17]:
```
df.to_excel('countries.xlsx', index=False)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [ ]:

```
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js