Programming for Data Science 2023

Homework Assigment Two

Homework activities aim at testing not only your ability to put into practice the concepts you have learned during the Lectures and Labs, but also your ability to explore the Python documentation as a resource. Above all, it is an opportunity for you to challenge yourself and practice. If you are having difficulties with the assignment reach out for support.

Description

This homework assignment will test your capacity to **load and manipulate data with Pandas**.

The goal is to develop intuition on filtering, arranging, and merging data, which will be useful for the next homework assignments.
Fill the empty cells with your code and deliver a copy of this notebook to Moodle.

Your submission will be graded according to the following guidelines:

1. **Execution** (does your program does what is asked from the exercise?
2. **Objectivity** (are you using the adequate libraries? are you using a library ...)
3. **Readability** of your code (including comments, variables naming, supporting text, etc ...)

Comment your code properly, which includes naming your variables in a meaningful manner. Badly documented code will be penalized.

This assignment is to be done in pairs, as in the first one, but remember that **you can't have the same pair as you had in Homework 1**.

Students that are caught cheating will obtain a score of 0 points.

Homework 2 is worth 25% of your final grade.

The submission package should correspond to a .zip archive (.rar files are not accepted) with the following files:

1. Jupyter Notebook with the output of all the cells;
2. HTML print of your Jupyter Notebook (in Jupyter go to File -> Download as -> HTML);
3. All text or .csv files are exported as part of the exercises. Please don't upload the files downloaded/imported as part of the exercises.

Please change the name of the notebook to "H2.\<student_1*id*>\ <student_2_id>.ipynb", replacing \<student_id> by your student_id.

Submission is done through the respective Moodle activity, and only one of the group members has to submit the files.

The deadline is the **12th of October at 23:59**.
A penalty of 1 point per day late will be applied to late deliveries.
In this notebook, you are allowed to use Pandas and Numpy.

In [215…

```
import numpy as np
import scipy
import pandas as pd
```

Start Here

[Please Complete the following form with your details]

Student Name - Marta Jesus
Student id - 20230464
Contact e-mail - 20230464@novaims.unl.pt / marta.jesus9@gmail.com

Student Name - Pedro Cerejeira
Student id - 20230442
Contact e-mail - 20230442@novaims.unl.pt / pcerejeira@gmail.com

Part 1 - Get the Data

Download and Load the World Development Indicators Dataset

We will work with the **World Development Indicators dataset**, which should be downloaded from the world bank databank.
Hence, the very first step is to download the data to your computer, you can do this by running the cell below.

NOTE This cell may timeout on slower connections. If you receive an error you will need to download the file manually by pasting the URL into your browser. After downloading the zip archive you will need to move it to the same folder as this notebook and then unzip it to have acees to the required files.

Alternatively you can copy and paste the url inside the .get() method into your browser.

The above code downloads a zip archive to the working folder, which by default is the the location of this notebook in your computer.
Secondly, and since the document downloaded is a zip archive, it extracts the documents from the archive.
The contents include multiple .csv files, however we will be working only with the document 'WDIData.csv'.

1. In the cell bellow, use Pandas to open the file "WDIData.csv" and **save** it to a variable called **wdi**.

NOTE If you see strange characters in the headings or text you may need to specify the option enconding, "ISO-8859-1" has worked previously. Find more information at https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html

In [ ]:
```
# importing libraries
import requests, zipfile, io

#note this can take several minutes depending on your internet connection
r = requests.get('http://databank.worldbank.org/data/download/WDI_csv.zip')
z = zipfile.ZipFile(io.BytesIO(r.content))
z.extractall()

# let us free the variales we used above
del z
del r
```

In [230…
```
# open and read the csv file using pandas
wdi=pd.read_csv('WDIData.csv')
```

2. Check the top of the dataframe to ensure it loaded correctly.

In [231…
```
# head fuction to see the top of the dataframe
wdi.head()
```

Out[231]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	1965	...
0	Africa Eastern and Southern	AFE	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...
1	Africa Eastern and Southern	AFE	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.RU.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...
2	Africa Eastern and Southern	AFE	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.UR.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...
3	Africa Eastern and Southern	AFE	Access to electricity (% of population)	EG.ELC.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...
4	Africa Eastern and Southern	AFE	Access to electricity, rural (% of rural popul...	EG.ELC.ACCS.RU.ZS	NaN	NaN	NaN	NaN	NaN	NaN	...

5 rows × 67 columns

Download and Load the Penn World Table V9.0

We will additionally use data from the pwt v9.0 tables.

Again **run the following cell to download the dataset**. This time using the library urllib.

In [138…
```
import urllib
urllib.request.urlretrieve("https://www.rug.nl/ggdc/docs/pwt90.xlsx", "pwt90.xlsx")
```

Out[138]:
```
('pwt90.xlsx', <http.client.HTTPMessage at 0x21478a3f9d0>)
```

3. In the following cell, open and read the file 'pwt90.xlsx' and **save** it into variable **pwt**.

In [232…
```
# open and read the xlsx file using pandas and select the sheet with the data we ne
pwt=pd.read_excel('pwt90.xlsx', sheet_name='Data')
pwt
```

Out[232]:

	countrycode	country	currency_unit	year	rgdpe	rgdpo	pop	e
0	ABW	Aruba	Aruban Guilder	1950	NaN	NaN	NaN	N
1	ABW	Aruba	Aruban Guilder	1951	NaN	NaN	NaN	N
2	ABW	Aruba	Aruban Guilder	1952	NaN	NaN	NaN	N
3	ABW	Aruba	Aruban Guilder	1953	NaN	NaN	NaN	N
4	ABW	Aruba	Aruban Guilder	1954	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	
11825	ZWE	Zimbabwe	US Dollar	2010	20652.718750	21053.855469	13.973897	6.298
11826	ZWE	Zimbabwe	US Dollar	2011	20720.435547	21592.298828	14.255592	6.518
11827	ZWE	Zimbabwe	US Dollar	2012	23708.654297	24360.527344	14.565482	6.248
11828	ZWE	Zimbabwe	US Dollar	2013	27011.988281	28157.886719	14.898092	6.287
11829	ZWE	Zimbabwe	US Dollar	2014	28495.554688	29149.708984	15.245855	6.499

11830 rows × 47 columns

4. Check the top of the dataframe to ensure it was loaded correctly.

In [233…
```
# check the top rows of the dataframe using the pandas head fuction
pwt.head()
```

Out[233]:

	countrycode	country	currency_unit	year	rgdpe	rgdpo	pop	emp	avh	hc	...	csh_g	c
0	ABW	Aruba	Aruban Guilder	1950	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
1	ABW	Aruba	Aruban Guilder	1951	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
2	ABW	Aruba	Aruban Guilder	1952	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
3	ABW	Aruba	Aruban Guilder	1953	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	
4	ABW	Aruba	Aruban Guilder	1954	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	

5 rows × 47 columns

Part 2 - Data Processing

Data Wrangling

Now that we have loaded our data we are ready to start playing with it.

5. Start by printing all the column values in the cell bellow.

In [235…

```
# check and print (in a list) the column names
print(wdi.columns.values)
```

```
['Country Name' 'Country Code' 'Indicator Name' 'Indicator Code' '1960'
 '1961' '1962' '1963' '1964' '1965' '1966' '1967' '1968' '1969' '1970'
 '1971' '1972' '1973' '1974' '1975' '1976' '1977' '1978' '1979' '1980'
 '1981' '1982' '1983' '1984' '1985' '1986' '1987' '1988' '1989' '1990'
 '1991' '1992' '1993' '1994' '1995' '1996' '1997' '1998' '1999' '2000'
 '2001' '2002' '2003' '2004' '2005' '2006' '2007' '2008' '2009' '2010'
 '2011' '2012' '2013' '2014' '2015' '2016' '2017' '2018' '2019' '2020'
 '2021' 'Unnamed: 66']
```

6. List the values in the column 'Country Name'.You will get a list with repeated values, **delete all duplicates** to ease your analysis.

Tip: There is a method in the pandas library that allows to do this easily.

In [236…

```
# the unique fuction will show all the unique values (no duplicates)
wdi['Country Name'].unique().tolist() # not saving to the dataframe, since we are j
```

Out[236]:
```
['Africa Eastern and Southern',
 'Africa Western and Central',
 'Arab World',
 'Caribbean small states',
 'Central Europe and the Baltics',
 'Early-demographic dividend',
 'East Asia & Pacific',
 'East Asia & Pacific (excluding high income)',
 'East Asia & Pacific (IDA & IBRD countries)',
 'Euro area',
 'Europe & Central Asia',
 'Europe & Central Asia (excluding high income)',
 'Europe & Central Asia (IDA & IBRD countries)',
 'European Union',
 'Fragile and conflict affected situations',
 'Heavily indebted poor countries (HIPC)',
 'High income',
 'IBRD only',
 'IDA & IBRD total',
 'IDA blend',
 'IDA only',
 'IDA total',
 'Late-demographic dividend',
 'Latin America & Caribbean',
 'Latin America & Caribbean (excluding high income)',
 'Latin America & the Caribbean (IDA & IBRD countries)',
 'Least developed countries: UN classification',
 'Low & middle income',
 'Low income',
 'Lower middle income',
 'Middle East & North Africa',
 'Middle East & North Africa (excluding high income)',
 'Middle East & North Africa (IDA & IBRD countries)',
 'Middle income',
 'North America',
 'Not classified',
 'OECD members',
 'Other small states',
 'Pacific island small states',
 'Post-demographic dividend',
 'Pre-demographic dividend',
 'Small states',
 'South Asia',
 'South Asia (IDA & IBRD)',
 'Sub-Saharan Africa',
 'Sub-Saharan Africa (excluding high income)',
 'Sub-Saharan Africa (IDA & IBRD countries)',
 'Upper middle income',
 'World',
 'Afghanistan',
 'Albania',
 'Algeria',
 'American Samoa',
 'Andorra',
 'Angola',
 'Antigua and Barbuda',
 'Argentina',
 'Armenia',
 'Aruba',
 'Australia',
 'Austria',
 'Azerbaijan',
 'Bahamas, The',
 'Bahrain',
```

```
'Bangladesh',
'Barbados',
'Belarus',
'Belgium',
'Belize',
'Benin',
'Bermuda',
'Bhutan',
'Bolivia',
'Bosnia and Herzegovina',
'Botswana',
'Brazil',
'British Virgin Islands',
'Brunei Darussalam',
'Bulgaria',
'Burkina Faso',
'Burundi',
'Cabo Verde',
'Cambodia',
'Cameroon',
'Canada',
'Cayman Islands',
'Central African Republic',
'Chad',
'Channel Islands',
'Chile',
'China',
'Colombia',
'Comoros',
'Congo, Dem. Rep.',
'Congo, Rep.',
'Costa Rica',
"Cote d'Ivoire",
'Croatia',
'Cuba',
'Curacao',
'Cyprus',
'Czechia',
'Denmark',
'Djibouti',
'Dominica',
'Dominican Republic',
'Ecuador',
'Egypt, Arab Rep.',
'El Salvador',
'Equatorial Guinea',
'Eritrea',
'Estonia',
'Eswatini',
'Ethiopia',
'Faroe Islands',
'Fiji',
'Finland',
'France',
'French Polynesia',
'Gabon',
'Gambia, The',
'Georgia',
'Germany',
'Ghana',
'Gibraltar',
'Greece',
'Greenland',
'Grenada',
```

```
             'Guam',
             'Guatemala',
             'Guinea',
             'Guinea-Bissau',
             'Guyana',
             'Haiti',
             'Honduras',
             'Hong Kong SAR, China',
             'Hungary',
             'Iceland',
             'India',
             'Indonesia',
             'Iran, Islamic Rep.',
             'Iraq',
             'Ireland',
             'Isle of Man',
             'Israel',
             'Italy',
             'Jamaica',
             'Japan',
             'Jordan',
             'Kazakhstan',
             'Kenya',
             'Kiribati',
             "Korea, Dem. People's Rep.",
             'Korea, Rep.',
             'Kosovo',
             'Kuwait',
             'Kyrgyz Republic',
             'Lao PDR',
             'Latvia',
             'Lebanon',
             'Lesotho',
             'Liberia',
             'Libya',
             'Liechtenstein',
             'Lithuania',
             'Luxembourg',
             'Macao SAR, China',
             'Madagascar',
             'Malawi',
             'Malaysia',
             'Maldives',
             'Mali',
             'Malta',
             'Marshall Islands',
             'Mauritania',
             'Mauritius',
             'Mexico',
             'Micronesia, Fed. Sts.',
             'Moldova',
             'Monaco',
             'Mongolia',
             'Montenegro',
             'Morocco',
             'Mozambique',
             'Myanmar',
             'Namibia',
             'Nauru',
             'Nepal',
             'Netherlands',
             'New Caledonia',
             'New Zealand',
             'Nicaragua',
```

```
 'Niger',
 'Nigeria',
 'North Macedonia',
 'Northern Mariana Islands',
 'Norway',
 'Oman',
 'Pakistan',
 'Palau',
 'Panama',
 'Papua New Guinea',
 'Paraguay',
 'Peru',
 'Philippines',
 'Poland',
 'Portugal',
 'Puerto Rico',
 'Qatar',
 'Romania',
 'Russian Federation',
 'Rwanda',
 'Samoa',
 'San Marino',
 'Sao Tome and Principe',
 'Saudi Arabia',
 'Senegal',
 'Serbia',
 'Seychelles',
 'Sierra Leone',
 'Singapore',
 'Sint Maarten (Dutch part)',
 'Slovak Republic',
 'Slovenia',
 'Solomon Islands',
 'Somalia',
 'South Africa',
 'South Sudan',
 'Spain',
 'Sri Lanka',
 'St. Kitts and Nevis',
 'St. Lucia',
 'St. Martin (French part)',
 'St. Vincent and the Grenadines',
 'Sudan',
 'Suriname',
 'Sweden',
 'Switzerland',
 'Syrian Arab Republic',
 'Tajikistan',
 'Tanzania',
 'Thailand',
 'Timor-Leste',
 'Togo',
 'Tonga',
 'Trinidad and Tobago',
 'Tunisia',
 'Turkiye',
 'Turkmenistan',
 'Turks and Caicos Islands',
 'Tuvalu',
 'Uganda',
 'Ukraine',
 'United Arab Emirates',
 'United Kingdom',
 'United States',
```

```
 'Uruguay',
 'Uzbekistan',
 'Vanuatu',
 'Venezuela, RB',
 'Vietnam',
 'Virgin Islands (U.S.)',
 'West Bank and Gaza',
 'Yemen, Rep.',
 'Zambia',
 'Zimbabwe']
```

You might notice that while the bottom rows represent Countries, the top rows represent aggregates of countries (e.g., world regions).
However we are only interested in **working with country-level data**, and as such we need to filter out all the unnecessary rows.

7. Save all the values of column 'Country Name' in a variable called **cnames**.

In [237…

```
# create a new variable called cnames to save all the Country Names in a list
cnames=wdi['Country Name'].tolist() # since the question specifies "values", we are
                                    # saving a series
cnames[0:20] # just showing a subset of the list, otherwise it will impact the disp
```

Out[237]:

```
['Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern',
 'Africa Eastern and Southern']
```

7.1. Delete all duplicate values.

In [238…

```
# now using drop_duplicates fuction to delete the duplicated values before listing
cnames=wdi['Country Name'].drop_duplicates().tolist()
cnames
```

Out[238]:
```
['Africa Eastern and Southern',
 'Africa Western and Central',
 'Arab World',
 'Caribbean small states',
 'Central Europe and the Baltics',
 'Early-demographic dividend',
 'East Asia & Pacific',
 'East Asia & Pacific (excluding high income)',
 'East Asia & Pacific (IDA & IBRD countries)',
 'Euro area',
 'Europe & Central Asia',
 'Europe & Central Asia (excluding high income)',
 'Europe & Central Asia (IDA & IBRD countries)',
 'European Union',
 'Fragile and conflict affected situations',
 'Heavily indebted poor countries (HIPC)',
 'High income',
 'IBRD only',
 'IDA & IBRD total',
 'IDA blend',
 'IDA only',
 'IDA total',
 'Late-demographic dividend',
 'Latin America & Caribbean',
 'Latin America & Caribbean (excluding high income)',
 'Latin America & the Caribbean (IDA & IBRD countries)',
 'Least developed countries: UN classification',
 'Low & middle income',
 'Low income',
 'Lower middle income',
 'Middle East & North Africa',
 'Middle East & North Africa (excluding high income)',
 'Middle East & North Africa (IDA & IBRD countries)',
 'Middle income',
 'North America',
 'Not classified',
 'OECD members',
 'Other small states',
 'Pacific island small states',
 'Post-demographic dividend',
 'Pre-demographic dividend',
 'Small states',
 'South Asia',
 'South Asia (IDA & IBRD)',
 'Sub-Saharan Africa',
 'Sub-Saharan Africa (excluding high income)',
 'Sub-Saharan Africa (IDA & IBRD countries)',
 'Upper middle income',
 'World',
 'Afghanistan',
 'Albania',
 'Algeria',
 'American Samoa',
 'Andorra',
 'Angola',
 'Antigua and Barbuda',
 'Argentina',
 'Armenia',
 'Aruba',
 'Australia',
 'Austria',
 'Azerbaijan',
 'Bahamas, The',
 'Bahrain',
```

```
          'Bangladesh',
          'Barbados',
          'Belarus',
          'Belgium',
          'Belize',
          'Benin',
          'Bermuda',
          'Bhutan',
          'Bolivia',
          'Bosnia and Herzegovina',
          'Botswana',
          'Brazil',
          'British Virgin Islands',
          'Brunei Darussalam',
          'Bulgaria',
          'Burkina Faso',
          'Burundi',
          'Cabo Verde',
          'Cambodia',
          'Cameroon',
          'Canada',
          'Cayman Islands',
          'Central African Republic',
          'Chad',
          'Channel Islands',
          'Chile',
          'China',
          'Colombia',
          'Comoros',
          'Congo, Dem. Rep.',
          'Congo, Rep.',
          'Costa Rica',
          "Cote d'Ivoire",
          'Croatia',
          'Cuba',
          'Curacao',
          'Cyprus',
          'Czechia',
          'Denmark',
          'Djibouti',
          'Dominica',
          'Dominican Republic',
          'Ecuador',
          'Egypt, Arab Rep.',
          'El Salvador',
          'Equatorial Guinea',
          'Eritrea',
          'Estonia',
          'Eswatini',
          'Ethiopia',
          'Faroe Islands',
          'Fiji',
          'Finland',
          'France',
          'French Polynesia',
          'Gabon',
          'Gambia, The',
          'Georgia',
          'Germany',
          'Ghana',
          'Gibraltar',
          'Greece',
          'Greenland',
          'Grenada',
```

```
 'Guam',
 'Guatemala',
 'Guinea',
 'Guinea-Bissau',
 'Guyana',
 'Haiti',
 'Honduras',
 'Hong Kong SAR, China',
 'Hungary',
 'Iceland',
 'India',
 'Indonesia',
 'Iran, Islamic Rep.',
 'Iraq',
 'Ireland',
 'Isle of Man',
 'Israel',
 'Italy',
 'Jamaica',
 'Japan',
 'Jordan',
 'Kazakhstan',
 'Kenya',
 'Kiribati',
 "Korea, Dem. People's Rep.",
 'Korea, Rep.',
 'Kosovo',
 'Kuwait',
 'Kyrgyz Republic',
 'Lao PDR',
 'Latvia',
 'Lebanon',
 'Lesotho',
 'Liberia',
 'Libya',
 'Liechtenstein',
 'Lithuania',
 'Luxembourg',
 'Macao SAR, China',
 'Madagascar',
 'Malawi',
 'Malaysia',
 'Maldives',
 'Mali',
 'Malta',
 'Marshall Islands',
 'Mauritania',
 'Mauritius',
 'Mexico',
 'Micronesia, Fed. Sts.',
 'Moldova',
 'Monaco',
 'Mongolia',
 'Montenegro',
 'Morocco',
 'Mozambique',
 'Myanmar',
 'Namibia',
 'Nauru',
 'Nepal',
 'Netherlands',
 'New Caledonia',
 'New Zealand',
 'Nicaragua',
```

```
'Niger',
'Nigeria',
'North Macedonia',
'Northern Mariana Islands',
'Norway',
'Oman',
'Pakistan',
'Palau',
'Panama',
'Papua New Guinea',
'Paraguay',
'Peru',
'Philippines',
'Poland',
'Portugal',
'Puerto Rico',
'Qatar',
'Romania',
'Russian Federation',
'Rwanda',
'Samoa',
'San Marino',
'Sao Tome and Principe',
'Saudi Arabia',
'Senegal',
'Serbia',
'Seychelles',
'Sierra Leone',
'Singapore',
'Sint Maarten (Dutch part)',
'Slovak Republic',
'Slovenia',
'Solomon Islands',
'Somalia',
'South Africa',
'South Sudan',
'Spain',
'Sri Lanka',
'St. Kitts and Nevis',
'St. Lucia',
'St. Martin (French part)',
'St. Vincent and the Grenadines',
'Sudan',
'Suriname',
'Sweden',
'Switzerland',
'Syrian Arab Republic',
'Tajikistan',
'Tanzania',
'Thailand',
'Timor-Leste',
'Togo',
'Tonga',
'Trinidad and Tobago',
'Tunisia',
'Turkiye',
'Turkmenistan',
'Turks and Caicos Islands',
'Tuvalu',
'Uganda',
'Ukraine',
'United Arab Emirates',
'United Kingdom',
'United States',
```

```
 'Uruguay',
 'Uzbekistan',
 'Vanuatu',
 'Venezuela, RB',
 'Vietnam',
 'Virgin Islands (U.S.)',
 'West Bank and Gaza',
 'Yemen, Rep.',
 'Zambia',
 'Zimbabwe']
```

7.2. Print the names that do not represent countries.

In [239…

```
cnames=cnames[0:49] # We confirmed manually with Excel that, the first 50 elements,
                    # we are using this approach since it is allowed, otherwise we
cnames
```

Out[239]:
```
['Africa Eastern and Southern',
 'Africa Western and Central',
 'Arab World',
 'Caribbean small states',
 'Central Europe and the Baltics',
 'Early-demographic dividend',
 'East Asia & Pacific',
 'East Asia & Pacific (excluding high income)',
 'East Asia & Pacific (IDA & IBRD countries)',
 'Euro area',
 'Europe & Central Asia',
 'Europe & Central Asia (excluding high income)',
 'Europe & Central Asia (IDA & IBRD countries)',
 'European Union',
 'Fragile and conflict affected situations',
 'Heavily indebted poor countries (HIPC)',
 'High income',
 'IBRD only',
 'IDA & IBRD total',
 'IDA blend',
 'IDA only',
 'IDA total',
 'Late-demographic dividend',
 'Latin America & Caribbean',
 'Latin America & Caribbean (excluding high income)',
 'Latin America & the Caribbean (IDA & IBRD countries)',
 'Least developed countries: UN classification',
 'Low & middle income',
 'Low income',
 'Lower middle income',
 'Middle East & North Africa',
 'Middle East & North Africa (excluding high income)',
 'Middle East & North Africa (IDA & IBRD countries)',
 'Middle income',
 'North America',
 'Not classified',
 'OECD members',
 'Other small states',
 'Pacific island small states',
 'Post-demographic dividend',
 'Pre-demographic dividend',
 'Small states',
 'South Asia',
 'South Asia (IDA & IBRD)',
 'Sub-Saharan Africa',
 'Sub-Saharan Africa (excluding high income)',
 'Sub-Saharan Africa (IDA & IBRD countries)',
 'Upper middle income',
 'World']
```

You can take advantage of the structure of the dataset to realize that aggregates (Continents, Regions, etc) are all located on the top of the series 'cnames'. Moreover, since the series is small you can easily validate this assumption manually and then use that information to extract a slice of all the entries that represent non-countries entities.

8. In the next cell filter out, from **wdi**, the rows in which 'Country Name' represents an aggregate of countries.

In [240…
```
# Find the index of the first occurrence of the target value
index_of_first_afg = (wdi['Country Name'] == 'Afghanistan').idxmax()

# Filter out all the aggregate countries; they are all above afghanistan (our targe
```

```
wdi = wdi[index_of_first_afg::]
wdi
```

Out[240]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	19
70658	Afghanistan	AFG	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	N
70659	Afghanistan	AFG	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.RU.ZS	NaN	NaN	NaN	NaN	NaN	N
70660	Afghanistan	AFG	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.UR.ZS	NaN	NaN	NaN	NaN	NaN	N
70661	Afghanistan	AFG	Access to electricity (% of population)	EG.ELC.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	N
70662	Afghanistan	AFG	Access to electricity, rural (% of rural popul...	EG.ELC.ACCS.RU.ZS	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	...	...	...	...	...
383567	Zimbabwe	ZWE	Women who believe a husband is justified in be...	SG.VAW.REFU.ZS	NaN	NaN	NaN	NaN	NaN	N
383568	Zimbabwe	ZWE	Women who were first married by age 15 (% of w...	SP.M15.2024.FE.ZS	NaN	NaN	NaN	NaN	NaN	N
383569	Zimbabwe	ZWE	Women who were first married by age 18 (% of w...	SP.M18.2024.FE.ZS	NaN	NaN	NaN	NaN	NaN	N
383570	Zimbabwe	ZWE	Women's share of population ages 15+ living wi...	SH.DYN.AIDS.FE.ZS	NaN	NaN	NaN	NaN	NaN	N
383571	Zimbabwe	ZWE	Young people (ages 15-24) newly infected with HIV	SH.HIV.INCD.YG	NaN	NaN	NaN	NaN	NaN	N

312914 rows × 67 columns

9. Check that the top of the **wdi** dataframe now only has countries and not aggregates of countries.

In [241…

```
# check the top of the df to confirm that we only have countries in it
wdi.head()
```

Out[241]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963	1964	196
70658	Afghanistan	AFG	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	Na
70659	Afghanistan	AFG	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.RU.ZS	NaN	NaN	NaN	NaN	NaN	Na
70660	Afghanistan	AFG	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.UR.ZS	NaN	NaN	NaN	NaN	NaN	Na
70661	Afghanistan	AFG	Access to electricity (% of population)	EG.ELC.ACCS.ZS	NaN	NaN	NaN	NaN	NaN	Na
70662	Afghanistan	AFG	Access to electricity, rural (% of rural popul...	EG.ELC.ACCS.RU.ZS	NaN	NaN	NaN	NaN	NaN	Na

5 rows × 67 columns

10. Reset the indexes of **wdi**. Perform this operation inplace.

In [242…

```
# inplace fuction so that we dont need to create a new variable, we can modify the
wdi.reset_index(inplace=True)
```

11. Show that the indexes have been reseted.

In [243…

```
wdi
```

Out[243]:

	index	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963
0	70658	Afghanistan	AFG	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.ZS	NaN	NaN	NaN	NaN
1	70659	Afghanistan	AFG	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.RU.ZS	NaN	NaN	NaN	NaN
2	70660	Afghanistan	AFG	Access to clean fuels and technologies for coo...	EG.CFT.ACCS.UR.ZS	NaN	NaN	NaN	NaN
3	70661	Afghanistan	AFG	Access to electricity (% of population)	EG.ELC.ACCS.ZS	NaN	NaN	NaN	NaN
4	70662	Afghanistan	AFG	Access to electricity, rural (% of rural popul...	EG.ELC.ACCS.RU.ZS	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
312909	383567	Zimbabwe	ZWE	Women who believe a husband is justified in be...	SG.VAW.REFU.ZS	NaN	NaN	NaN	NaN
312910	383568	Zimbabwe	ZWE	Women who were first married by age 15 (% of w...	SP.M15.2024.FE.ZS	NaN	NaN	NaN	NaN
312911	383569	Zimbabwe	ZWE	Women who were first married by age 18 (% of w...	SP.M18.2024.FE.ZS	NaN	NaN	NaN	NaN
312912	383570	Zimbabwe	ZWE	Women's share of population ages 15+ living wi...	SH.DYN.AIDS.FE.ZS	NaN	NaN	NaN	NaN
312913	383571	Zimbabwe	ZWE	Young people (ages 15-24) newly infected with HIV	SH.HIV.INCD.YG	NaN	NaN	NaN	NaN

312914 rows × 68 columns

Note that when reseting the index, pandas appends a new column at the begining of the data frame, which holds the previous index values.

Indicator Codes and Indicator Name

12. Select the columns 'Indicator Name' and 'Indicator Code'.Then, delete all the duplicates, and print the top 5 and bottom 5 values.

Note: You should be able to do everything in a single line of code for the top 5 values and a single line for the bottom 5 values.

In [244…

```
# acessing both Indicator Name and Indicator Code. Then deleting all the duplicates
    # check the top 5 rows of it with head fuction
print(wdi[["Indicator Name", "Indicator Code"]].drop_duplicates().head(5))

    # check the bottom 5 rows of it with tail fuction
print(wdi[["Indicator Name", "Indicator Code"]].drop_duplicates().tail(5))
```

```
                                      Indicator Name     Indicator Code
0  Access to clean fuels and technologies for coo...     EG.CFT.ACCS.ZS
1  Access to clean fuels and technologies for coo...  EG.CFT.ACCS.RU.ZS
2  Access to clean fuels and technologies for coo...  EG.CFT.ACCS.UR.ZS
3            Access to electricity (% of population)     EG.ELC.ACCS.ZS
4  Access to electricity, rural (% of rural popul...  EG.ELC.ACCS.RU.ZS
                                         Indicator Name     Indicator Code
1437  Women who believe a husband is justified in be...     SG.VAW.REFU.ZS
1438  Women who were first married by age 15 (% of w...  SP.M15.2024.FE.ZS
1439  Women who were first married by age 18 (% of w...  SP.M18.2024.FE.ZS
1440  Women's share of population ages 15+ living wi...  SH.DYN.AIDS.FE.ZS
1441  Young people (ages 15-24) newly infected with HIV     SH.HIV.INCD.YG
```

13. Create a new DataFrame named **indicators** made up of the columns 'Indicator Name' and 'Indicator Code'. Then, delete all the duplicated entries. Finally, set the column 'Indicator Code' as the index of **indicators**.

Note: Try to perform all these steps in a single line of code.

In [245…

```
# create a new df (with the targeted columns) and set the index to Indicator Code u
indicators = (wdi[["Indicator Name", "Indicator Code"]].drop_duplicates()).set_inde
indicators
```

Out[245]:

Indicator Code	Indicator Name
EG.CFT.ACCS.ZS	Access to clean fuels and technologies for coo...
EG.CFT.ACCS.RU.ZS	Access to clean fuels and technologies for coo...
EG.CFT.ACCS.UR.ZS	Access to clean fuels and technologies for coo...
EG.ELC.ACCS.ZS	Access to electricity (% of population)
EG.ELC.ACCS.RU.ZS	Access to electricity, rural (% of rural popul...
...	...
SG.VAW.REFU.ZS	Women who believe a husband is justified in be...
SP.M15.2024.FE.ZS	Women who were first married by age 15 (% of w...
SP.M18.2024.FE.ZS	Women who were first married by age 18 (% of w...
SH.DYN.AIDS.FE.ZS	Women's share of population ages 15+ living wi...
SH.HIV.INCD.YG	Young people (ages 15-24) newly infected with HIV

1442 rows × 1 columns

The 'indicators' DataFrame can operate now as a dictionary.
By passing an 'Indicator Code' (key) it returns the associated 'Indicator Name' (value).

14. Using the **indicators** DataFrame, find the 'Indicator Code' associated with the following observables:

1. 'Population', find the 'Indicator Code' of the total population in a country;
2. 'GDP', find the GDP measured in current US Dollars;
3. 'GINI index'

Hint: You can use the method STRING.str.contains('substring') to check whether a string contains a substring.

In [246…

```
def getTotalPop(indicators):
    # to access both the index and row values simultaneously we use iterrows fuctio
    for index, row in indicators.iterrows():
        value = row['Indicator Name']
        # check which index has the substring 'Population, total'
        if 'Population, total' in value:
            # return the value found
            return index

getTotalPop(indicators)
```

Out[246]: 'SP.POP.TOTL'

In [302…

```
def getGdpDollar(indicators):
    for index, row in indicators.iterrows():
        value = row['Indicator Name']
        # check which index has both the substrings 'GDP' and 'current US$' and, at
        # the substring 'per capita'
        if 'GDP' in value and 'current US$' in value and 'per capita' not in value
            # return the value found
            return index
```

```
getGdpDollar(indicators)
```

Out[302]:
```
'NY.GDP.MKTP.CD'
```

In [248…
```
def getGiniIndex(indicators):
    for index, row in indicators.iterrows():
        value = row['Indicator Name']
        # check which index has the substring 'Gini index'
        if 'Gini index' in value:
            # return the value found
            return index

getGiniIndex(indicators)
```

Out[248]:
```
'SI.POV.GINI'
```

Extracting and Cleaning Data from WDI and PWT

15. From **wdi** extract the columns 'Indicator Code', 'Country Code', and '2012'. Save the output in variable **wdi_sample**.

Note: You should be able to perfom all operations in a single line of code.

In [249…
```
# new df with 3 columns from wdi, 'Indicator Code', 'Country Code', and '2012'
wdi_sample = (wdi[["Indicator Code", "Country Code", "2012"]])
wdi_sample
```

Out[249]:

	Indicator Code	Country Code	2012
0	EG.CFT.ACCS.ZS	AFG	23.000000
1	EG.CFT.ACCS.RU.ZS	AFG	8.200000
2	EG.CFT.ACCS.UR.ZS	AFG	74.400000
3	EG.ELC.ACCS.ZS	AFG	69.099998
4	EG.ELC.ACCS.RU.ZS	AFG	60.849155
...	...	...	...
312909	SG.VAW.REFU.ZS	ZWE	NaN
312910	SP.M15.2024.FE.ZS	ZWE	NaN
312911	SP.M18.2024.FE.ZS	ZWE	NaN
312912	SH.DYN.AIDS.FE.ZS	ZWE	58.900000
312913	SH.HIV.INCD.YG	ZWE	22000.000000

312914 rows × 3 columns

16. Select from **wdi_sample** the lines associated with all the Indicator Codes that you found above, which concern the data of the 'GINI index', 'GDP', and 'Population total'.

In [250…
```
[getTotalPop(indicators), getGdpDollar(indicators), getGiniIndex(indicators)]
```

Out[250]:
```
['SP.POP.TOTL', 'NY.GDP.MKTP.CD', 'SI.POV.GINI']
```

In [251…

```
# creating an array with the 3 Indicator Codes found above (in exercise 14)
indicator_codes = [getTotalPop(indicators), getGdpDollar(indicators), getGiniIndex(

# create a new df that stores all the rows, for which the Indicator Code is one of
selected_rows = wdi_sample[wdi_sample["Indicator Code"].isin(indicator_codes)]
selected_rows
```

Out[251]:

	Indicator Code	Country Code	2012
467	NY.GDP.MKTP.CD	AFG	2.020357e+10
491	SI.POV.GINI	AFG	NaN
1062	SP.POP.TOTL	AFG	3.046648e+07
1909	NY.GDP.MKTP.CD	ALB	1.231983e+10
1933	SI.POV.GINI	ALB	2.900000e+01
...	...	...	...
310521	SI.POV.GINI	ZMB	NaN
311092	SP.POP.TOTL	ZMB	1.474466e+07
311939	NY.GDP.MKTP.CD	ZWE	1.711485e+10
311963	SI.POV.GINI	ZWE	NaN
312534	SP.POP.TOTL	ZWE	1.326533e+07

651 rows × 3 columns

17. Create a pivot table, in which **values** are the column '2012', the **index** is the 'Country Code', and the **columns** are the Indicator Codes.

Hint: Pandas has a very useful method to create pivot tables.

In [252…

```
# create a pivot table with the parameters required
wdi_sample = pd.pivot_table(selected_rows, values='2012', index='Country Code', co
wdi_sample
```

Out[252]:

Indicator Code	NY.GDP.MKTP.CD	SI.POV.GINI	SP.POP.TOTL
Country Code			
ABW	2.615084e+09	NaN	102112.0
AFG	2.020357e+10	NaN	30466479.0
AGO	1.249982e+11	NaN	25188292.0
ALB	1.231983e+10	29.0	2900401.0
AND	3.188809e+09	NaN	71013.0
...	...	...	...
XKX	6.163785e+09	29.0	1807106.0
YEM	3.540132e+10	NaN	26223391.0
ZAF	4.344005e+11	NaN	53145033.0
ZMB	2.550306e+10	NaN	14744658.0
ZWE	1.711485e+10	NaN	13265331.0

217 rows × 3 columns

18. Rename the column names of **wdi_sample** to 'Population', 'GDP', and 'GINI', accordingly.

In [253…

```
# renaming the columns as required
wdi_sample = wdi_sample.rename(columns={'SP.POP.TOTL': 'Population', 'NY.GDP.MKTP.(
wdi_sample
```

Out[253]:

Indicator Code	GDP	GINI	Population
Country Code			
ABW	2.615084e+09	NaN	102112.0
AFG	2.020357e+10	NaN	30466479.0
AGO	1.249982e+11	NaN	25188292.0
ALB	1.231983e+10	29.0	2900401.0
AND	3.188809e+09	NaN	71013.0
...	...	...	...
XKX	6.163785e+09	29.0	1807106.0
YEM	3.540132e+10	NaN	26223391.0
ZAF	4.344005e+11	NaN	53145033.0
ZMB	2.550306e+10	NaN	14744658.0
ZWE	1.711485e+10	NaN	13265331.0

217 rows × 3 columns

19. From **pwt** select only the values of the year 2012.
Then, extract the columns 'countrycode' and 'hc' into a new variable **pwt_sample**.
Rename 'countrycode' to 'Country Code', so that it matches the same column in **wdi_sample**

Note: in this case 'hc' stands for the Human Capital Index.

In [254…

```
# selecting just the rows from the year 2012
pwt[pwt["year"] == 2012]
```

Out[254]:

	countrycode	country	currency_unit	year	rgdpe	rgdpo	pop	
62	ABW	Aruba	Aruban Guilder	2012	3744.970947	3576.219971	0.102393	0.
127	AGO	Angola	Kwanza	2012	172309.781250	192991.562500	22.685632	7.
192	AIA	Anguilla	East Caribbean Dollar	2012	362.970490	284.968933	0.014133	
257	ALB	Albania	Lek	2012	28811.394531	30245.425781	2.880667	0.
322	ARE	United Arab Emirates	UAE Dirham	2012	558347.437500	609734.437500	8.952542	5.
...	...	...	...	...	...	...	...	
11567	VNM	Viet Nam	Dong	2012	444350.312500	446818.343750	90.335547	50.
11632	YEM	Yemen	Yemeni Rial	2012	85529.218750	92789.835938	24.882792	5.
11697	ZAF	South Africa	Rand	2012	635144.250000	627725.187500	52.837274	16.
11762	ZMB	Zambia	Kwacha	2012	52006.925781	50184.925781	14.786581	4.
11827	ZWE	Zimbabwe	US Dollar	2012	23708.654297	24360.527344	14.565482	6.

182 rows × 47 columns

In [255…

```
# create a new df that will store the columns 'countrycode' and 'hc' for the rows w
pwt_sample = pwt[pwt["year"] == 2012][["countrycode", "hc"]]
pwt_sample
```

Out[255]:

	countrycode	hc
62	ABW	NaN
127	AGO	1.431295
192	AIA	NaN
257	ALB	2.917346
322	ARE	2.723864
...	...	...
11567	VNM	2.532614
11632	YEM	1.488720
11697	ZAF	2.596012
11762	ZMB	2.330032
11827	ZWE	2.459828

182 rows × 2 columns

In [256…

```
# rename the column to Country Code so it has the same name as the wdi_sample colum
pwt_sample = pwt_sample.rename(columns={"countrycode": "Country Code"})
pwt_sample
```

Out[256]:

	Country Code	hc
62	ABW	NaN
127	AGO	1.431295
192	AIA	NaN
257	ALB	2.917346
322	ARE	2.723864
...	...	...
11567	VNM	2.532614
11632	YEM	1.488720
11697	ZAF	2.596012
11762	ZMB	2.330032
11827	ZWE	2.459828

182 rows × 2 columns

20. Finally, create a new dataframe named **data** that contains the columns from **wdi_sample** and **pwt_sample**, matched by 'Country Code'.

Hint: Use the method concat(), and make sure both dataframes have the same index ('Country Code').

In [258…

```
# setting the index as the "Country Code"
data_aux = pwt_sample.set_index("Country Code")
data_aux
```

Out[258]:

	hc
Country Code	
ABW	NaN
AGO	1.431295
AIA	NaN
ALB	2.917346
ARE	2.723864
...	...
VNM	2.532614
YEM	1.488720
ZAF	2.596012
ZMB	2.330032
ZWE	2.459828

182 rows × 1 columns

In [259…

```
# Concatenate DataFrames based on matching indexes
data = pd.concat([data, wdi_sample], axis=1)
data
```

Out[259]:

	hc	GDP	GINI	Population
Country Code				
ABW	NaN	2.615084e+09	NaN	102112.0
AGO	1.431295	1.249982e+11	NaN	25188292.0
AIA	NaN	NaN	NaN	NaN
ALB	2.917346	1.231983e+10	29.0	2900401.0
ARE	2.723864	3.846101e+11	NaN	8664969.0
...	...	...	...	...
TUV	NaN	3.934562e+07	NaN	10854.0
VIR	NaN	4.089000e+09	NaN	108188.0
VUT	NaN	7.478397e+08	NaN	257313.0
WSM	NaN	7.731575e+08	NaN	198124.0
XKX	NaN	6.163785e+09	29.0	1807106.0

220 rows × 4 columns

Part 3 - Analysing a Dataset

21. Perform the necessary manipulations to answer the following questions, unless otherwise stated you can use the country codes to represent the countries in your solutions:

1. Which countries have a **population size of 10 million habitations +/- 1 million**?
2. What is the **average** and the **standard deviation in GDP** of countries listed in 1?
3. What is the **average** and the **standard deviation in the GDP per capita** of countries listed in 1?
4. Consider the following classification of country size:
   Tiny - population < 1 000 000
   Very Small - 1 000 000 <= population < 5 000 000
   Small - 5 000 000 <= population < 15 000 000
   Medium - 15 000 000 <= population < 30 000 000
   Large - 30 000 000 <= population < 100 000 000
   Huge - 100 000 000 <= population
   What is the **average** and the **standard deviation in the GDP per capita of countries in each size classification**?
5. Create a **function** that will take a dataframe and a column name, and **return** a series with binary values indicating whether the **values from the column are above the mean value of that column** (indicated with a value of 1 or 0 otherwise). If the value in the column is missing (NaN) the value in the series should also be missing (NaN). Test your function. *Hint:* search how to check if something is None so that we can return None.
6. What is the **average GDP per capita of the countries after being grouped by size classification and whether or not the human capital is above average**?
7. What is the **average GDP per capita of countries after being grouped by whether or not the human capital is above average and whether or not the gini coefficient is above average?**
8. What is the **name of the country** that has the **highest GDP per capita + a Gini coefficient below average and a level of human capital below average**?
9. What is the **name of the country** that has the **highest GDP per capita + a Gini coefficient below average for its size classification and a level of human capital below average for its size classification**?
10. What is the **name of the country** that has the **largest % increase in GDP between 1980 and 2010?** *HINT: You will need to use the wdi dataframe.*

Write the code necessary to answer each question in a single cell.
Print the answer at the end of that cell.

In [260…

```
#1
# Extract the valid entries (population from 9m to 11m). Save their country codes
matching_country_codes = data[(data["Population"] >= 9000000) & (data["Population"]
matching_country_codes
```

Out[260]:
```
['AZE',
 'BDI',
 'BEN',
 'BLR',
 'BOL',
 'CZE',
 'DOM',
 'GIN',
 'HTI',
 'HUN',
 'PRT',
 'RWA',
 'SWE',
 'SSD']
```

In [261…
```
#2
# Use boolean indexing to filter data based on matching_country_codes
filtered_concat_df = data.loc[data.index.isin(matching_country_codes)]

print("The average GDP of the countries listed in 1 is:", filtered_concat_df["GDP"]
print("The standard deviation of the GDP of the countries listed in 1 is:", filtere
```

```
The average GDP of the countries listed in 1 is: 98851346744.4546 dollars
The standard deviation of the GDP of the countries listed in 1 is: 149245940074.66
046 dollars
```

In [262…
```
#3

# create a copy of the dataframe so we can work and modify it without changing the
filtered_df_per_capita = filtered_concat_df.copy()

# GDP per capita is the GDP divided by the population
# we could just create a new variable and work with that; however we would have a l

# creating a new column with country 'GDP per capita'
filtered_df_per_capita["GDP per capita"] = filtered_concat_df["GDP"] / filtered_co

# .mean() and .std() to calculate the average and the standard deviation
print("The average GDP per capita of the countries listed in 1 is:", (filtered_df_p
print("The standard deviation of the GDP per capita of the countries listed in 1 is
```

```
The average GDP per capita of the countries listed in 1 is: 9982.43842280551 dolla
rs
The standard deviation of the GDP per capita of the countries listed in 1 is: 1547
3.803230921145 dollars
```

In [263…
```
#4

# Define the population classifications and their ranges
population_ranges = [
    (0, 1000000, 'Tiny'),
    (1000000, 5000000, 'Very Small'),
    (5000000, 15000000, 'Small'),
    (15000000, 30000000, 'Medium'),
    (30000000, 100000000, 'Large'),
    (100000000, np.inf, 'Huge')
]

#create a fuction to check the range of each population
def classify_population(population):
    for min_pop, max_pop, classification in population_ranges:
        if min_pop <= population < max_pop:
            return classification
    # if country has no population defined, return "Unknown"
    return "Unknown"
```

```
# create a copy of the dataframe so we can work and modify it without changing the
filtered_df_per_capita = concat_df.copy()

# creating a new column with country 'GDP per capita'
filtered_df_per_capita["GDP per capita"] = concat_df["GDP"] / concat_df["Populatio

# create a new column for the classifications
filtered_df_per_capita['Population Classification'] = filtered_df_per_capita['Popul

# Group the DataFrame by the new classification column and show their respective me
grouped = filtered_df_per_capita.groupby('Population Classification')["GDP per cap

print(grouped)
```

```
                                   mean           std
Population Classification
Huge                       14106.483034  18628.054216
Large                      13169.183311  15601.727837
Medium                      8894.046643  16312.507343
Small                      16501.754630  23701.728931
Tiny                       28173.577452  36335.068274
Unknown                             NaN           NaN
Very Small                 14143.764480  18407.134058
```

In [265…

```
#5

def create_binary_series(df, column):
    # Get mean from the selected column
    column_mean = df[column].mean()

    # Create a Series following the given rules
    binary_series = df[column].apply(lambda x: 1 if not pd.isna(x) and x > column_m

    return binary_series

result_series = create_binary_series(concat_df, 'GDP')

print(result_series)
```

```
Country Code
ABW    0.0
AGO    0.0
AIA    NaN
ALB    0.0
ARE    1.0
      ...
TUV    0.0
VIR    0.0
VUT    0.0
WSM    0.0
XKX    0.0
Name: GDP, Length: 220, dtype: float64
```

In [269…

```
#6

# create a copy of the dataframe so we can work and modify it without changing the
filtered_df_per_capita = concat_df.copy()

# create a new column in the df with the boolean result of hc above average
filtered_df_per_capita["hc above average"] = create_binary_series(filtered_df_per_c

# creating a new column with country 'GDP per capita'
filtered_df_per_capita["GDP per capita"] = concat_df["GDP"] / concat_df["Populatio
```

```
# create a new column for the classifications
filtered_df_per_capita['Population Classification'] = filtered_df_per_capita['Popul

# Group the DataFrame by the new 'classification' and 'hc above average' columns. S
grouped = filtered_df_per_capita.groupby(['Population Classification', 'hc above av

grouped
```

Out[269]:

Population Classification	hc above average	mean
Huge	0.0	2686.227455
	1.0	27810.789730
Large	0.0	3674.566136
	1.0	24235.276159
Medium	0.0	1323.697495
	1.0	19398.142872
Small	0.0	3081.423432
	1.0	33784.346526
Tiny	0.0	7447.432572
	1.0	41094.901824
Unknown	1.0	NaN
Very Small	0.0	9522.938477
	1.0	19089.144112

In [268…

```
#7

# create a copy of the dataframe so we can work and modify it without changing the
filtered_df_per_capita = concat_df.copy()

# create a new column in the df with the boolean result of hc above average
filtered_df_per_capita["hc above average"] = create_binary_series(filtered_df_per_c

# create a new column in the df with the boolean result of gini above average
filtered_df_per_capita["gini above average"] = create_binary_series(filtered_df_per

# creating a new column with country 'GDP per capita'
filtered_df_per_capita["GDP per capita"] = concat_df["GDP"] / concat_df["Populatior

# Group the DataFrame by the 'hc above average' and 'gini above average' columns. S
grouped = filtered_df_per_capita.groupby(['hc above average', 'gini above average']

grouped
```

Out[268]:

hc above average	gini above average	mean
0.0	0.0	6090.769470
	1.0	4350.769217
1.0	0.0	32178.530201
	1.0	12566.243331

In [271…

```
#8

# create a copy of the dataframe so we can work and modify it without changing the
filtered_df_per_capita = concat_df.copy()

# create a new column in the df with the boolean result of hc above average
filtered_df_per_capita["hc above average"] = create_binary_series(filtered_df_per_c

# create a new column in the df with the boolean result of gini above average
filtered_df_per_capita["gini above average"] = create_binary_series(filtered_df_per

# creating a new column with country 'GDP per capita'
filtered_df_per_capita["GDP per capita"] = concat_df["GDP"] / concat_df["Population

# filtering the whole dataframe by the rows which have gini and hc below the averag
filtered_df_per_capita = filtered_df_per_capita[(filtered_df_per_capita['gini above

# selecting the countrycode of the country with the highest 'GDP per capita' in thi
max_gdp_idx = filtered_df_per_capita['GDP per capita'].idxmax()

# matching the countrycode in the pwt dataframe (since it also has the countries no
# we do not want all the matching rows in the df
selected_country = pwt[pwt['countrycode'] == max_gdp_idx]['country'].head(1).values

selected_country
```

Out[271]: 'Portugal'

In [272…

```
#9

# create a copy of the dataframe so we can work and modify it without changing the
filtered_df_per_capita = concat_df.copy()

# for this exercise we are dropping countries with 'hc' or 'gini' with 'nan' values
# as below the average, which is not what we are looking for
filtered_df_per_capita = filtered_df_per_capita.dropna(subset=['hc', 'GINI'])

# create a new column for the classifications
filtered_df_per_capita['Population Classification'] = filtered_df_per_capita['Popul

# create a new column with 'GDP per capita' for each country
filtered_df_per_capita["GDP per capita"] = concat_df["GDP"] / concat_df["Population

# Group the DataFrame by the classification and fetch the average GINI per group
filtered_df_per_capita['avg gini per class'] = filtered_df_per_capita.groupby('Popu
# Group the DataFrame by the classification and fetch the average HC per group
filtered_df_per_capita['avg hc per class'] = filtered_df_per_capita.groupby('Popula

# create a new column in the df with the boolean according to hc relationship with
filtered_df_per_capita['hc above mean per class'] = np.where(filtered_df_per_capita
```

```
# create a new column in the df with the boolean according to GINI relationship wit
filtered_df_per_capita['gini above mean per class'] = np.where(filtered_df_per_cap:

# filtering the whole dataframe by the specified rules in the exercise
filtered_df_per_capita = filtered_df_per_capita[(filtered_df_per_capita['gini above

# selecting the countrycode of the country with the highest 'GDP per capita' in thi
max_gdp_idx = filtered_df_per_capita['GDP per capita'].idxmax()

# matching the countrycode in the pwt dataframe (since it also has the countries nc
# we do not want all the matching rows in the df
selected_country = pwt[pwt['countrycode'] == max_gdp_idx]['country'].head(1).values

selected_country
```

Out[272]: 'Iceland'

In [318…

```
#10

# Reusing the function created in the previous exercises to filter the df 'getGdpDc
def getGdpDollarRow(indicators):
    for index, row in indicators.iterrows():
        value = row['Indicator Name']
        # check which index has both the substrings 'GDP' and 'current US$' and, at
        # the substring 'per capita'
        if 'GDP' in value and 'current US$' in value and 'per capita' not in value
            # return the value found
            return row

wdi_aux = wdi[wdi["Indicator Code"] == getGdpDollarRow(wdi)["Indicator Code"]]

wdi_filtered = wdi_aux.copy()

# If a country has no GDP registered for 1980, using the first actual value of the
wdi_filtered['first_valid_year_after_1980'] = wdi_filtered.apply(
    lambda row: row.loc['1981':'2010'].first_valid_index() if pd.isna(row['1980'])
    axis=1
)

# Filling nans (if a country has no GDP registered from 1980 to 2010) with the valu
wdi_filtered['first_valid_year_after_1980'] = wdi_filtered['first_valid_year_after_

# Finding the first valud GDP after 1980 by using the first valid year previously 1
wdi_filtered['first_valid_gdp_after_1980'] = wdi_filtered.apply(
    lambda row: row[row['first_valid_year_after_1980']],
    axis=1
)

# Calculate the GDP % growth for each country, using the GDP's found above
wdi_filtered['gdp_growth'] = (
    (wdi_filtered['2010'] - wdi_filtered['first_valid_gdp_after_1980']) /
    wdi_filtered['first_valid_gdp_after_1980']
) * 100

# Identify the country with the biggest gdp_growth (%)
max_growth_country = wdi_filtered.loc[wdi_filtered['gdp_growth'].idxmax(), 'Country

print(f"The country with the largest GDP growth from the first valid year after 198
```

```
The country with the largest GDP growth from the first valid year after 1980 to 20
10 is: Equatorial Guinea
```