PYTHON FOR DATA SCIENCE COURSE
FINAL ASSIGNMENT

Pedro Almada

17TH of December 2024

Oxford, United Kingdom

Second-Assignment

December 17, 2024

1 Second Assignment: World Progress

In this project, you'll explore data from Gapminder.org, a website dedicated to providing a fact-based view of the world and how it has changed. That site includes several data visualizations and presentations, but also publishes the raw data that we will use in this project to recreate and extend some of their most famous visualizations.

The Gapminder website collects data from many sources and compiles them into tables that describe many countries around the world. All of the data they aggregate are published in the Systema Globalis. Their goal is "to compile all public statistics; Social, Economic and Environmental; into a comparable total dataset." All data sets in this project are copied directly from the Systema Globalis without any changes.

This project is dedicated to Hans Rosling (1948-2017), who championed the use of data to understand and prioritize global development challenges.

1.0.1 Logistics

The first assignment was worth 35 points. This second assignment contains 13 questions - each correct answer will be worth 5 points, for a total of 65 points. In order to pass overall you will need to reach at least 40 points out of 100 from the two assignments.

Advice: Develop your answers incrementally. To perform a complicated table manipulation, break it up into steps, perform each step on a different line, give a new name to each result, and check that each intermediate result is what you expect. You can add any additional names or functions you want to the provided cells.

Submission: Please send your solutions to the Weekly Classes Office (weekly-classes@conted.ox.ac.uk) with a signed DoA (Declaration of Authorship) form. Please send the notebook with your name appended to the file name.

Deadline: Please see Canvas

To get started, load `pandas`, `numpy`, and `matplotlib`.

[ ]:
```
import pandas as pd
import numpy as np
%matplotlib inline
import matplotlib.pyplot as plots
```

1.1 1. Global Population Growth

The global population of humans reached 1 billion around 1800, 3 billion around 1960, and 7 billion around 2011. The potential impact of exponential population growth has concerned scientists, economists, and politicians alike.

The UN Population Division estimates that the world population will likely continue to grow throughout the 21st century, but at a slower rate, perhaps reaching 11 billion by 2100. However, the UN does not rule out scenarios of more extreme growth.

In this section, we will examine some of the factors that influence population growth and how they are changing around the world.

The first table we will consider is the total population of each country over time. Run the cell below.

[ ]:
```
# The population.csv file should also be found online here:
# https://github.com/open-numbers/ddf--gapminder--systema_globalis/raw/master/
↪ddf--datapoints--population_total--by--geo--time.csv
# The version in this project was downloaded in February, 2017.
population = pd.read_csv('../datasets/population.csv')
population.head(3)
```

[ ]:

	geo	time	population_total
0	abw	1800	19286
1	abw	1801	19286
2	abw	1802	19286

1.1.1 Bangladesh

In the `population` table, the `geo` column contains three-letter codes established by the International Organization for Standardization (ISO) in the Alpha-3 standard. We will begin by taking a close look at Bangladesh. Inspect the standard to find the 3-letter code for Bangladesh.

Question 1. Create a `DataFrame` called `b_pop` that has two columns labeled `time` and `population_total`. The first column should contain the years from 1970 through 2015 (including both 1970 and 2015) and the second should contain the population of Bangladesh in each of those years.

[ ]:
```
# Step 1: Filter for Bangladesh using its 3-letter code 'BGD'
#I made the mistake of assuming the three-letter coded were in uppercase, and␣
↪overlookeds the display of the three firest collums above. I then oppened␣
↪the csv and realized the mistake.
b_pop = population[population['geo'] == 'bgd']

# Step 2: filtering the rows where 'time' is between 1970 and 2015
b_pop = b_pop[(b_pop['time'] >= 1970) & (b_pop['time'] <= 2015)]

# Step 3: Keeping only the 'time' and 'population_total' columns
#Use double brackets [['time', 'population_total']] to keep those columns.
```

```
b_pop = b_pop[['time', 'population_total']]

# At the end finally Display the first few rows to confirm all ius good
b_pop.head()
```

[ ]:

	time	population_total
7509	1970	65048701
7510	1971	66417450
7511	1972	67578486
7512	1973	68658472
7513	1974	69837960

Create a table called `b_five` that has the population of Bangladesh every five years. At a glance, it appears that the population of Bangladesh has been growing quickly indeed!

[ ]:
```
fives = np.arange(1970, 2016, 5) # 1970, 1975, 1980, ...
b_five = ...
b_five
```

[ ]:
```
fives = np.arange(1970, 2016, 5)  # 1970, 1975, 1980, ..., 2015 #This is to␣
↪generate the years every 5 years using np.arrange (creates an array from␣
↪1970 to 2015 in steps of 5)
b_five = b_pop[b_pop['time'].isin(fives)] #filtering b_pop to include only rows␣
↪where 'time' is in the 'fives' array
# I'm using the .isin() method to check if 'time' matches any value in 'fives'
b_five
```

[ ]:

	time	population_total
7509	1970	65048701
7514	1975	71247153
7519	1980	81364176
7524	1985	93015182
7529	1990	105983136
7534	1995	118427768
7539	2000	131280739
7544	2005	142929979
7549	2010	151616777
7554	2015	160995642

Question 2. Create a table called `b_five_growth` that includes three columns, `time`, `population_total`, and `annual_growth`. There should be one row for every five years from 1970 through 2010 (but not 2015). The first two columns are the same as `b_five`. The third column is the **annual** growth rate for each five-year period. For example, the annual growth rate for 1975 is the yearly exponential growth rate that describes the total growth from 1975 to 1980 when applied 5 times.

Hint: Only your `b_five_growth` table will be scored for correctness; the other names are suggestions that you are welcome to use, change, or delete.

[ ]:

```python
b_1970_through_2010 = ...
initial = ...
changed = ...
b_five_growth = ...
```

[ ]:

```python
# Step 1: Filtering b_five to exclude 2015
# I don't want 2015 because there's no "next" row for it to calculate growth
b_1970_through_2010 = b_five[b_five['time'] <= 2010].reset_index(drop=True)

# Step 2: Calculating the annual growth rate
# I need two versions of the population_total column to calculate growth:
# "initial" is population at time t (skip the last row)
# 'changed" is population at time t+5 (skip the first row)
initial = b_1970_through_2010['population_total'][:-1].values  # Population at␣
↪time t
changed = b_1970_through_2010['population_total'][1:].values   # Population at␣
↪time t+5

# Using the formula for exponential growth
# Annual growth = (changed / initial) ** (1/5) - 1
annual_growth = (changed / initial)**(1/5) - 1

# Step 3: Add the growth rate to the table
# I copy b_1970_through_2010 and drop the last row since it has no "next" value
b_five_growth = b_1970_through_2010[:-1].copy()  # Drop the last row because it␣
↪has no next value
b_five_growth['annual_growth'] = annual_growth

# Displaying e the final table
b_five_growth
```

[ ]:

	time	population_total	annual_growth
0	1970	65048701	0.018370
1	1975	71247153	0.026912
2	1980	81364176	0.027127
3	1985	93015182	0.026447
4	1990	105983136	0.022453
5	1995	118427768	0.020821
6	2000	131280739	0.017149
7	2005	142929979	0.011870

While the population has grown every five years since 1970, the annual growth rate decreased dramatically from 1985 to 2005. Let's look at some other information in order to develop a possible explanation. Run the next cell to load three additional tables of measurements about countries over time.

[ ]:
```
life_expectancy = pd.read_csv('../datasets/life_expectancy.csv')
child_mortality = pd.read_csv('../datasets/child_mortality.csv')
fertility = pd.read_csv('../datasets/fertility.csv')
```

The `life_expectancy` table contains a statistic that is often used to measure how long people live, called *life expectancy at birth*. This number, for a country in a given year, does not measure how long babies born in that year are expected to live. Instead, it measures how long someone would live, on average, if the *mortality conditions* in that year persisted throughout their lifetime. These "mortality conditions" describe what fraction of people at each age survived the year. So, it is a way of measuring the proportion of people that are staying alive, aggregated over different age groups in the population.

[ ]:
```
# I want start  by displaying the first five rows of wach dataset.
print("Life Expectancy Table:")
print(life_expectancy.head())

print("\nChild Mortality Table:")
print(child_mortality.head())

print("\nFertility Table:")
print(fertility.head())
```

```
Life Expectancy Table:
   geo  time  life_expectancy_years
0  afg  1800                  28.21
1  afg  1801                  28.20
2  afg  1802                  28.19
3  afg  1803                  28.18
4  afg  1804                  28.17

Child Mortality Table:
   geo  time  child_mortality_0_5_year_olds_dying_per_1000_born
0  afg  1800                                              468.6
1  afg  1801                                              468.6
2  afg  1802                                              468.6
3  afg  1803                                              468.6
4  afg  1804                                              468.6

Fertility Table:
   geo  time  children_per_woman_total_fertility
0  afg  1800                                 7.0
1  afg  1801                                 7.0
2  afg  1802                                 7.0
3  afg  1803                                 7.0
4  afg  1804                                 7.0
```

Question 3. Perhaps population is growing more slowly because people aren't living as long. Use the `life_expectancy` table to draw a line graph with the years 1970 and later on the horizontal axis that shows how the *life expectancy at birth* has changed in Bangladesh.

```
[ ]: ...
```

Does the graph above help directly explain why the population growth rate decreased from 1985 to 2010 in Bangladesh? Why or why not? What happened in Bangladesh in 1991, and does that event explain the change in population growth rate?

No, the graph does not directly explain why the population growth rate decreased in Bangladesh from 1985 to 2010. This graph shows global trends in population growth, not specific trends for Bangladesh. While it highlights the global slowdown in population growth projections, it does not provide any country-specific information. To explain the decrease in Bangladesh's population growth rate, we wpuld need country-level factors like life expectancy, fertility rates, or child mortality data — which we now have in the other tables.

To answer what happened in Bangladesh in 1991 I had to search the information to have a clearer picture. This is what I found: One of the most powerful tropical cyclones ever recorded in the basin, the tropical cyclone caused a 6.1 m (20 ft) storm surge, which inundated the coastline. The storm hit near the Chittagong region, one of the most populated areas in Bangladesh. An estimated 140,000 people were killed by the storm, as many as 10 million people lost their homes, and overall property damage was in the billions of dollars. This evennt likely does not explain the long-term decrease in population growth rate in Bangladesh from 1985 to 2010.While the cyclone caused a spike in deaths (140,000) in 1991, it does not explain the consistent, long-term decline in growth rates seen between 1985 and 2010. I't could be argued that the economic impact and people losing their homes could also play a role, but I would need more data on that.

The `fertility` table contains a statistic that is often used to measure how many babies are being born, the *total fertility rate*. This number describes the number of children a woman would have in her lifetime, on average, if the current rates of birth by age of the mother persisted throughout her child bearing years, assuming she survived through age 49.

```
[ ]: ##The link for the "number of children a woman would have in her lifetime" is
     ↪not currently working
```

Question 4. Write a function `fertility_over_time` that takes the Alpha-3 code of a `country` and a `start` year. It returns a two-column table with labels "`Year`" and "`Children per woman`" that can be used to generate a line chart of the country's fertility rate each year, starting at the `start` year. The plot should include the `start` year and all later years that appear in the `fertility` table.

Then, in the next cell, call your `fertility_over_time` function on the Alpha-3 code for Bangladesh and the year 1970 in order to plot how Bangladesh's fertility rate has changed since 1970. **The expression that draws the line plot is provided for you; please don't change it.**

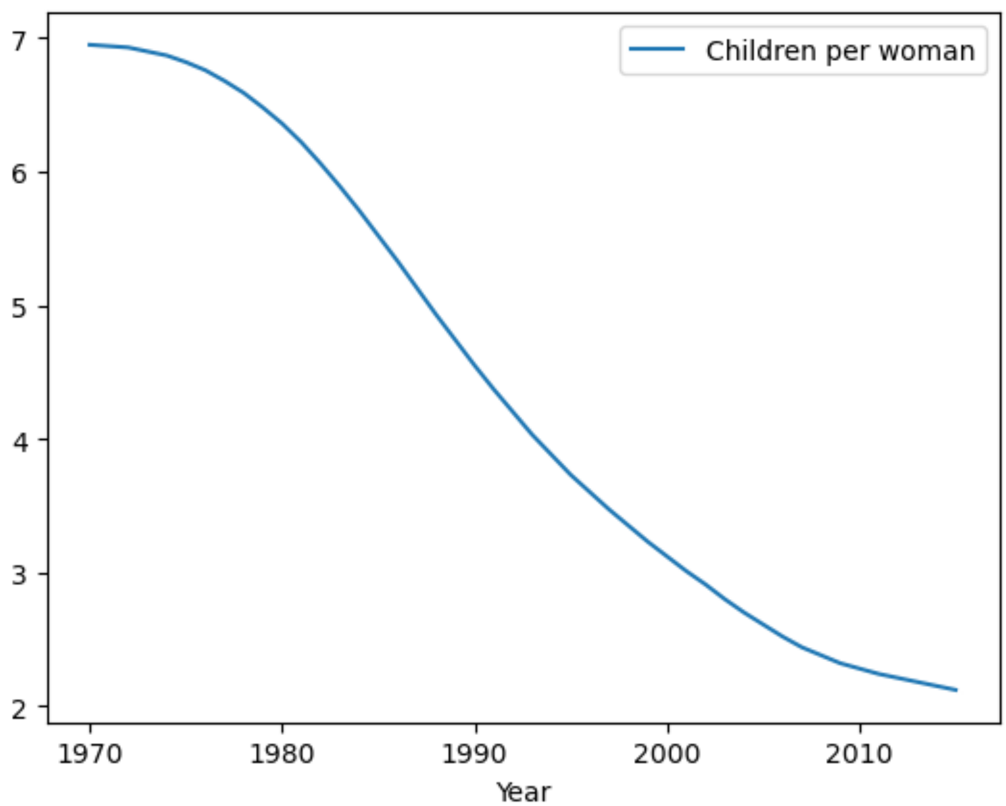
```
[ ]: def fertility_over_time(country: str, start: int) -> pd.DataFrame:
         """
         Create a two-column DataFrame that describes a country's total fertility
     ↪rate each year.
         :param country - alpha-3 code for the country
         :param start - start year
         """
         pass  # remove this line and write your solution here
```

```
[ ]: def fertility_over_time(country: str, start: int) -> pd.DataFrame:
         """
         Create a two-column DataFrame that describes a country's total fertility␣
      ↪rate each year.
         :param country: alpha-3 code for the country
         :param start: start year
         :return: DataFrame with two columns: 'Year' and 'Children per woman'
         """
         # Step 1: Filtering the fertility table for the specified country
         # I use 'geo' == country to get rows specific to the input country
         country_fertility = fertility[fertility['geo'] == country]

         # Step 2: Filtering  for years greater than or equal to 'start'
         # I only keep years greater than or equal to 'start'

         country_fertility = country_fertility[country_fertility['time'] >= start]

         # Step 3: Renameing  the columns to 'Year' and 'Children per woman'
         # 'time' becomes 'Year' for clarity
         # 'children_per_woman_total_fertility' becomes 'Children per woman'
         country_fertility = country_fertility[['time',␣
      ↪'children_per_woman_total_fertility']]
         country_fertility.columns = ['Year', 'Children per woman']

         #return
         return country_fertility
```

```
[ ]: bangladesh_code = ...
     bangladesh_code.plot(0, 1) # You should *not* change this line.
```

```
[ ]: #copying and pasting the code from above
     # Continuing with Step 5: Calling the function for Bangladesh ('bgd') starting␣
      ↪from 1970
     bangladesh_code = fertility_over_time('bgd', 1970)

     # Step 6: Ploting  the results using the provided line of code
     bangladesh_code.plot(0, 1)  # Do not change this line
```

```
[ ]: <Axes: xlabel='Year'>
```



Question 5. Does the graph above help directly explain why the population growth rate decreased from 1985 to 2010 in Bangladesh? Why or why not?

Yes, this graoh provides some strong evidence on why the population growth rate decreased in Bangladesh from 1985 to 2010. The graph shows that the total fertility rate in Bangladesh decreased significantly from nearly 7 children per woman in 1970 to just over 2 children per woman by 2010. This decline in fertility directly contributes to a slower population growth rate since fewer children are being born.

It has been observed that lower fertility rates are often associated with lower child mortality rates. The link has been attributed to family planning: if parents can expect that their children will all survive into adulthood, then they will choose to have fewer children. We can see if this association is evident in Bangladesh by plotting the relationship between total fertility rate and child mortality rate per 1000 children.

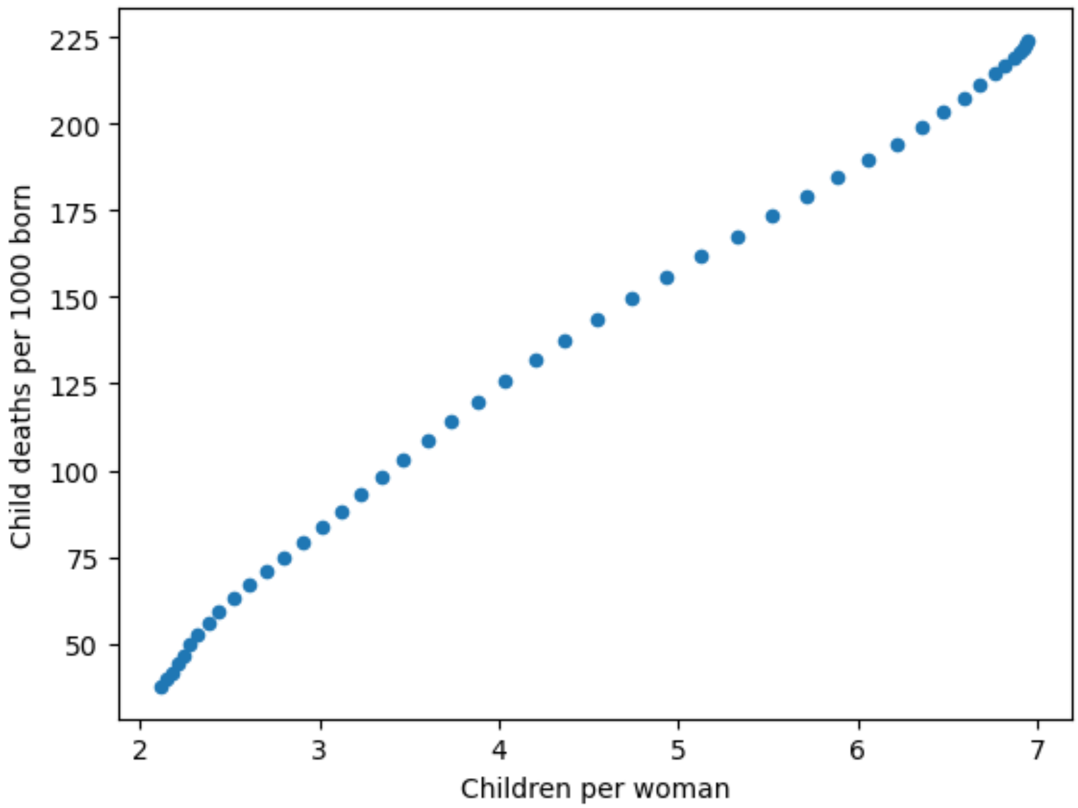
Question 6. Using both the `fertility` and `child_mortality` tables, draw a scatter diagram with one point for each year, starting with 1970, that has Bangladesh's total fertility on the horizontal axis and its child mortality on the vertical axis.

The expression that draws the scatter diagram is provided for you; please don't change it. Instead, create a table called `fertility_and_child_mortality` with the appropriate column labels and data in order to generate the chart correctly. Use the label "`Children per woman`" to describe total fertility and the label "`Child deaths per 1000 born`" to describe child mortality.

[ ]:
```
fertility_and_child_mortality = ...

# You should *not* change the statement below
fertility_and_child_mortality.plot.scatter(
    'Children per woman', 'Child deaths per 1000 born'
)
```

[ ]:
```
# First, Step 1: Filtering fertility table for Bangladesh ('bgd') starting from␣
 ↪1970
# Keeping only rows where 'geo' == 'bgd' and 'time' >= 1970
bangladesh_fertility = fertility[(fertility['geo'] == 'bgd') &␣
 ↪(fertility['time'] >= 1970)]

# Step 2: Filtering child_mortality table for Bangladesh starting from 1970
bangladesh_mortality = child_mortality[(child_mortality['geo'] == 'bgd') &␣
 ↪(child_mortality['time'] >= 1970)]

# Step 3: Merging the two tables on 'geo' and 'time' columns
# This combines fertility and child mortality data year by year

fertility_and_child_mortality = pd.merge(bangladesh_fertility,␣
 ↪bangladesh_mortality, on=['geo', 'time'])

# Step 4: Renaming columns to match the scatter plot labels
fertility_and_child_mortality = fertility_and_child_mortality.rename(columns={
    'children_per_woman_total_fertility': 'Children per woman',
    'child_mortality_0_5_year_olds_dying_per_1000_born': 'Child deaths per 1000␣
 ↪born'
})

# Step 5: Selecting   only the columns we need
# Keeping 'time', 'Children per woman', and 'Child deaths per 1000 born'
fertility_and_child_mortality = fertility_and_child_mortality[['time',␣
 ↪'Children per woman', 'Child deaths per 1000 born']]

# Step 6: Is to plot  the scatter diagram (provided code)
fertility_and_child_mortality.plot.scatter(
    'Children per woman', 'Child deaths per 1000 born'
)
```

[ ]:
```
<Axes: xlabel='Children per woman', ylabel='Child deaths per 1000 born'>
```



In one or two sentences, describe the association (if any) that is illustrated by this scatter diagram. Does the diagram show that reduced child mortality causes parents to choose to have fewer children?

The scatter diagram, does show an association. In this case, a positvie association between child mortality and fertility rates. This means that as child mortality decreses, the number of children per woman also decreses. However, the scatter diagram does not prove causation. The diagram only illustrates a correlation. The graph does not prove that reduced child mortality causes parents to have fewer children; it only shows a correlation. While reduced child mortality is likely a contributing factor in parents choosing to have fewer childre, there might as well be other contributing factors.

[ ]:

```

```

1.1.2 The World

The change observed in Bangladesh since 1970 can also be observed in many other developing countries: health services improve, life expectancy increases, and child mortality decreases. At the same time, the fertility rate often plummets, and so the population growth rate decreases despite increasing longevity.

Run the cell below to generate two overlaid histograms, one for 1960 and one for 2010, that show the distributions of total fertility rates for these two years among all 201 countries in the `fertility` table.

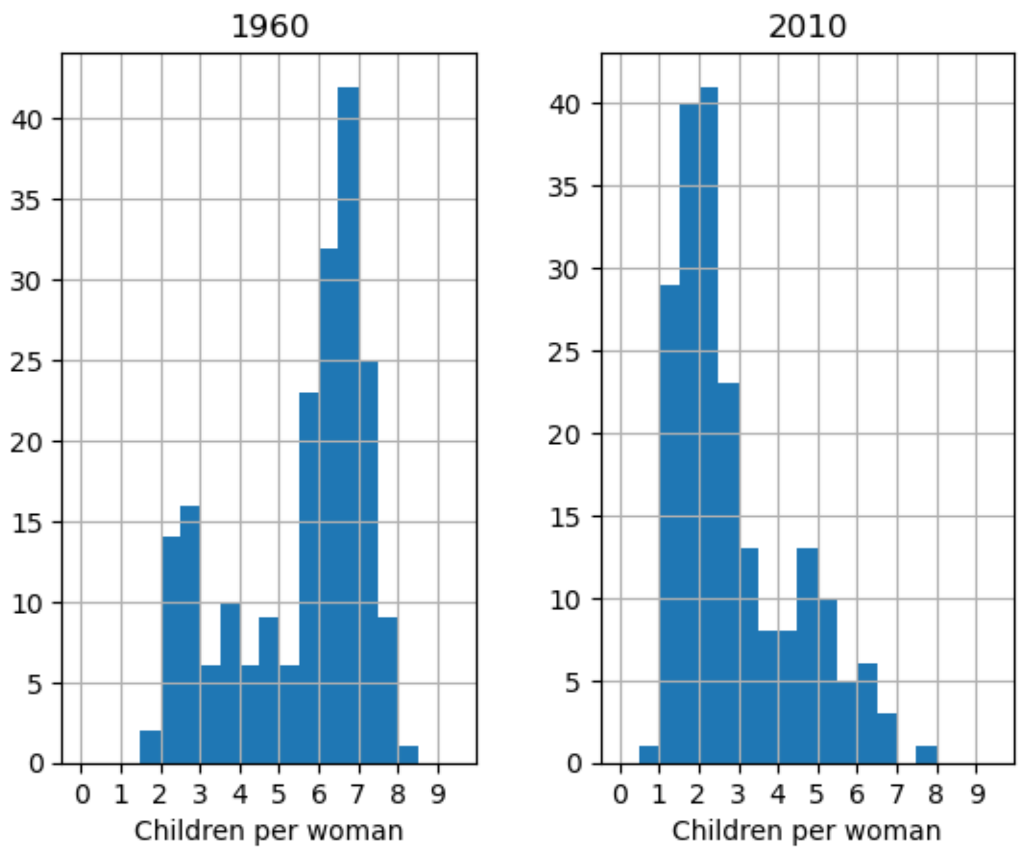
[ ]:
```
fertility[fertility.time == 1960]
```

[ ]:
```
        geo  time  children_per_woman_total_fertility
160     afg  1960                                7.67
376     alb  1960                                6.19
592     dza  1960                                7.65
808     ago  1960                                7.32
1024    atg  1960                                4.43
…     …    …                                  …
42492   vnm  1960                                6.35
42708   vir  1960                                5.62
42924   yem  1960                                7.29
43140   zmb  1960                                7.02
43356   zwe  1960                                7.16

[201 rows x 3 columns]
```

[ ]:
```
fertility[fertility.time == 1960].iloc[:, [0, 2]].set_index("geo")
```

[ ]:
```
     children_per_woman_total_fertility
geo
afg                                7.67
alb                                6.19
dza                                7.65
ago                                7.32
atg                                4.43
..                                  …
vnm                                6.35
vir                                5.62
yem                                7.29
zmb                                7.02
zwe                                7.16

[201 rows x 1 columns]
```

[ ]:
```
ax = pd.concat([
    fertility[fertility.time == 1960].iloc[:, [0, 2]].rename(
        columns={"children_per_woman_total_fertility": "1960"}
    ).set_index("geo"),
    fertility[fertility.time == 2010].iloc[:, [0, 2]].rename(
        columns={"children_per_woman_total_fertility": "2010"}
    ).set_index("geo")
], axis=1).hist(bins=np.arange(0, 10, 0.5))
for ix in range(len(ax[0])):
    ax[0][ix].set_xlabel('Children per woman')
    ax[0][ix].set_xticks(np.arange(10))
```



Question 7. Assign `fertility_statements` to a list of the numbers for each statement below that can be correctly inferred from these histograms. 1. About the same number of countries had a fertility rate between 3.5 and 4.5 in both 1960 and 2010. 1. In 2010, about 40% of countries had a fertility rate between 1.5 and 2. 1. In 1960, less than 20% of countries had a fertility rate below 3. 1. More countries had a fertility rate above 3 in 1960 than in 2010. 1. At least half of countries had a fertility rate between 5 and 8 in 1960. 1. At least half of countries had a fertility rate below 3 in 2010.

[ ]:
```
fertility_statements = ...
```

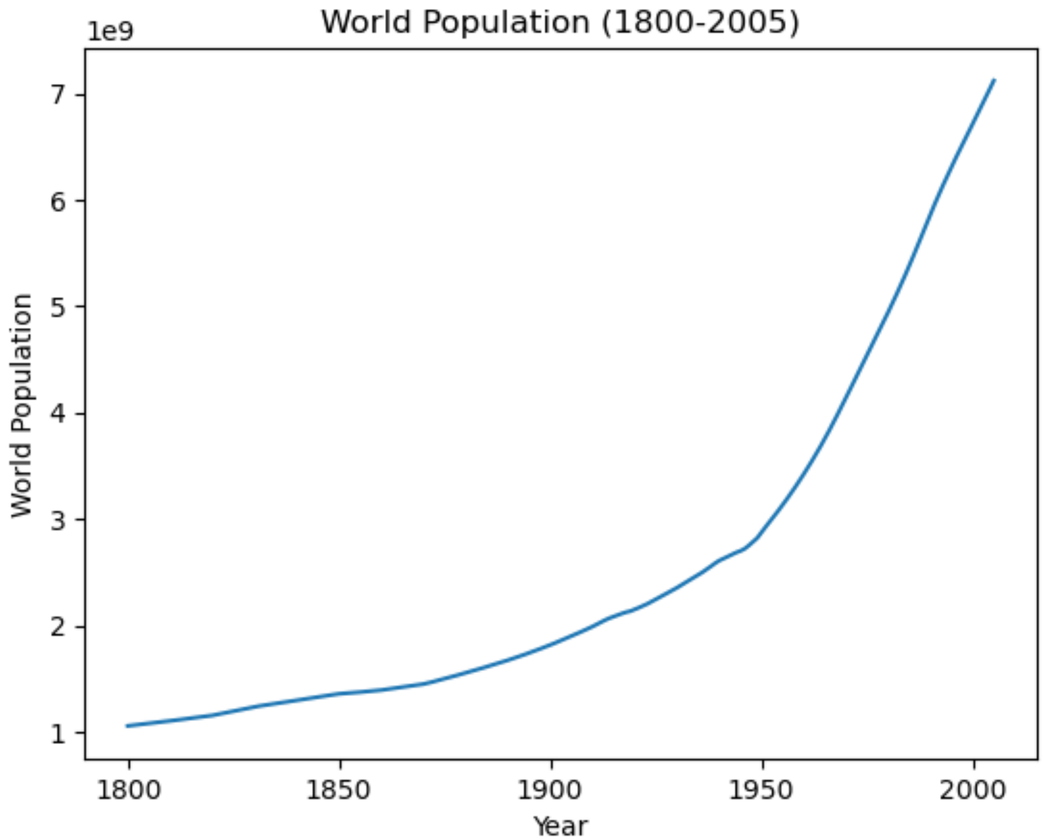
[ ]:
```
# List of correct statements based on the analysis of the histograms
# 2: In 2010, about 40% of countries had a fertility rate between 1.5 and 2.
# 3: In 1960, less than 20% of countries had a fertility rate below 3.
# 4: More countries had a fertility rate above 3 in 1960 than in 2010.
# 5: At least half of countries had a fertility rate between 5 and 8 in 1960.
# 6: At least half of countries had a fertility rate below 3 in 2010.

fertility_statements = [2, 3, 4, 5, 6]

#This will in principle create a variable called fertility_statements.
```

```
# The variable just holds the answers. Meaning that this is only to represent␣
↪the answers. !!
```

Question 8. Draw a line plot of the world population from 1800 through 2005. The world population is the sum of all the country's populations.

```
[ ]: # write your solution here
...
# Step 1: Filtering the population table for years between 1800 and 2005
world_population = population[(population['time'] >= 1800) &␣
↪(population['time'] <= 2005)]

# Step 2: Grouping by 'time' and sum the population for each year
world_population_sum = world_population.groupby('time')['population_total'].
↪sum()

# Step 3: Ploting the world population as a line plot
# 'time' is the index after grouping, and 'population_total' is summed .
world_population_sum.plot(
    kind='line',
    title='World Population (1800-2005)',
    xlabel='Year',
    ylabel='World Population'
)
```

```
[ ]: <Axes: title={'center': 'World Population (1800-2005)'}, xlabel='Year',
ylabel='World Population'>
```



Question 9. Create a function `stats_for_year` that takes a `year` and returns a table of statistics. The table it returns should have four columns: `geo`, `population_total`, `children_per_woman_total_fertility`, and `child_mortality_under_5_per_1000_born`. Each row should contain one Alpha-3 country code and three statistics: population, fertility rate, and child mortality for that `year` from the `population`, `fertility` and `child_mortality` tables. Only include rows for which all three statistics are available for the country and year.

In addition, restrict the result to country codes that appears in `big_50`, an array of the 50 most populous countries in 2010. This restriction will speed up computations later in the project.

[ ]:
```
big_50 = population[(population["time"] == 2010)].sort_values(
    by="population_total", ascending=False
).head(50)
population_of_big_50 = big_50.population_total
big_50.sample(5)
```

[ ]:

	geo	time	population_total
30411	gha	2010	24317734
81152	uga	2010	33149417
61088	pak	2010	170043918
59484	npl	2010	26875910

```
15158   chn   2010         1340968737
```

[ ]:
```python
# We first create a population table that only includes the
# 50 countries with the largest 2010 populations. We focus on
# these 50 countries only so that plotting later will run faster.

def stats_for_year(year: int) -> pd.DataFrame:
    """
    Return a DataFrame with the stats for each country that year.
    """
    pass
```

[ ]:
```python
# We first create a population table that only includes the
# 50 countries with the largest 2010 populations. We focus on
# these 50 countries only so that plotting later will run faster.

# First I  Filter the big_50 country codes
big_50_codes = big_50['geo']



# This defines  the function stats_for_year
def stats_for_year(year: int) -> pd.DataFrame:
    """
    Return a DataFrame with the stats for each country that year.
    """

    # Filtering the data for the given year
    pop_data = population[population['time'] == year][['geo',␣
 ↪'population_total']]
    fert_data = fertility[fertility['time'] == year][['geo',␣
 ↪'children_per_woman_total_fertility']]
    mort_data = child_mortality[child_mortality['time'] == year][['geo',␣
 ↪'child_mortality_0_5_year_olds_dying_per_1000_born']]

    # Then we need to merge the three tables on 'geo'
    merged_data = pop_data.merge(fert_data, on='geo').merge(mort_data, on='geo')

    # Next, restricting to the top 50 populous countries
    merged_data = merged_data[merged_data['geo'].isin(big_50_codes)]

    # Droping rows with missing values
    merged_data = merged_data.dropna()

    return merged_data
```

[ ]:
```python
# You should verify the function you have written is correct
# by running this cell for year 2010
```

```python
stats_for_year(2010)

#This is to veryfy the function for the year 2010
```

[ ]:

```
     geo  population_total  children_per_woman_total_fertility  \
1    afg          27962207                                5.66
5    arg          41222875                                2.22
15   bgd         151616777                                2.28
23   bra         198614208                                1.84
29   can          34126173                                1.63
33   chn        1340968737                                1.54
36   cod          65938712                                6.25
38   col          45918101                                2.38
45   deu          80435307                                1.39
49   dza          36036159                                2.82
51   egy          82040994                                2.88
54   esp          46601492                                1.46
56   eth          87561814                                4.90
59   fra          62961136                                1.98
62   gbr          62716684                                1.90
64   gha          24317734                                4.05
81   idn         241613126                                2.43
82   ind        1230984504                                2.56
84   irn          74253373                                1.90
85   irq          30868156                                4.21
88   ita          59588007                                1.44
91   jpn         127319802                                1.37
93   ken          40328313                                4.62
97   kor          49090041                                1.27
110  mar          32107739                                2.58
114  mex         118617542                                2.28
118  mmr          51733013                                2.00
121  moz          24321457                                5.41
126  mys          28119500                                2.00
131  nga         159424742                                6.02
135  npl          26875910                                2.62
138  pak         170043918                                3.43
140  per          29373644                                2.51
141  phl          93038902                                3.15
143  pol          38574682                                1.37
145  prk          24500506                                2.00
153  rus         143158099                                1.57
155  sau          28090647                                2.83
156  sdn          36114885                                4.64
174  tha          66692024                                1.44
181  tur          72310416                                2.10
182  tza          45648525                                5.43
```

183	uga	33149417	6.16
184	ukr	45647497	1.44
186	usa	309876170	1.93
187	uzb	27739764	2.41
189	ven	28995745	2.47
191	vnm	88357775	1.82
194	yem	23591972	4.50
195	zaf	51621594	2.47

	child_mortality_0_5_year_olds_dying_per_1000_born
1	105.0
5	14.6
15	49.6
23	16.7
29	5.6
33	15.7
36	116.1
38	18.5
45	4.2
49	27.4
51	29.0
54	4.6
56	75.7
59	4.3
62	5.2
64	74.7
81	33.1
82	59.9
84	19.2
85	36.6
88	4.0
91	3.2
93	63.6
97	4.1
110	33.1
114	16.8
118	59.3
121	103.8
126	8.3
131	130.3
135	45.4
138	91.8
140	21.0
141	31.9
143	5.8
145	31.3
153	12.0

```
155                                                     17.0
156                                                     80.2
174                                                     14.5
181                                                     19.1
182                                                     62.3
183                                                     79.5
184                                                     11.8
186                                                      7.4
187                                                     46.1
189                                                     16.6
191                                                     24.8
194                                                     58.8
195                                                     54.4
```

Question 10. Create a table called `pop_by_decade` with two columns called `decade` and `population`. It has a row for each `year` since 1960 that starts a decade. The `population` column contains the total population of all countries included in the result of `stats_for_year(year)` for the first `year` of the decade. For example, 1960 is the first year of the 1960's decade. You should see that these countries contain most of the world's population.

Hint: One approach is to define a function `pop_for_year` that computes this total population, then `apply` it to the `decade` column.

[ ]:
```python
decades = pd.DataFrame({
    'decade': np.arange(1960, 2011, 10)
    }
)

def pop_for_year(year):
    ...

pop_by_decade = ...
```

[ ]:
```python
decades = pd.DataFrame({
    'decade': np.arange(1960, 2011, 10)  # First we need to creat  a data frame␣
 ↪with decades... Generate years: 1960, 1970, ..., 2010
})

# Then we need to define the function to calculate total population for a year
def pop_for_year(year):
    """
    Calculate the total population for the given year from stats_for_year.
    """
    # We need to use the stats_for_year function and sum up the␣
 ↪'population_total' column
    return stats_for_year(year)['population_total'].sum()
```

```
# Then we need to apply the function to the 'decade' column and create a new␣
↪column 'population'
pop_by_decade = decades.copy()
pop_by_decade['population'] = pop_by_decade['decade'].apply(pop_for_year)


pop_by_decade #show the data
```

[ ]:

	decade	population
0	1960	2624944597
1	1970	3211487418
2	1980	3880722003
3	1990	4648434558
4	2000	5367553063
5	2010	6040810517

The `countries` table describes various characteristics of countries. The `country` column contains the same codes as the `geo` column in each of the other data tables (`population`, `fertility`, and `child_mortality`). The `world_6region` column classifies each country into a region of the world. Run the cell below to inspect the data.

[ ]:
```
countries = pd.read_csv('../datasets/countries.csv')
countries[['country', 'name', 'world_6region']]
```

[ ]:

	country	name	world_6region
0	abkh	Abkhazia	europe_central_asia
1	afg	Afghanistan	south_asia
2	akr_a_dhe	Akrotiri and Dhekelia	europe_central_asia
3	alb	Albania	europe_central_asia
4	dza	Algeria	middle_east_north_africa
..	…	…	…
270	yem	Yemen	middle_east_north_africa
271	yug	Yugoslavia	europe_central_asia
272	zmb	Zambia	sub_saharan_africa
273	zwe	Zimbabwe	sub_saharan_africa
274	ala	Åland	europe_central_asia

[275 rows x 3 columns]

1.2 2. Global Poverty

In 1800, 85% of the world's 1 billion people lived in *extreme poverty*, defined by the United Nations as "a condition characterized by severe deprivation of basic human needs, including food, safe drinking water, sanitation facilities, health, shelter, education and information." A common measure of extreme poverty is a person living on less than $1.25 per day.

In 2015, the proportion of people living in extreme poverty was estimated to be 12%. Although the world rate of extreme poverty has declined consistently for hundreds of years, the number of people living in extreme poverty is still close to 1 billion. The United Nations recently adopted an ambitious goal: "By 2030, eradicate extreme poverty for all people everywhere." In this section, we will examine extreme poverty trends around the world.

First, load the population and poverty rate by country and year and the country descriptions. While the `population` table has values for every recent year for many countries, the `poverty` table only includes certain years for each country in which a measurement of the rate of extreme poverty was available.

[ ]:
```
population = pd.read_csv('../datasets/population.csv')
# NOTE: The code here below is 'pseudo-code' and you have to modify it in order␣
↪to have it working accordingly
population.head(5)
```

[ ]:

	geo	time	population_total
0	abw	1800	19286
1	abw	1801	19286
2	abw	1802	19286
3	abw	1803	19286
4	abw	1804	19286

[ ]:
```
countries =  pd.read_csv('../datasets/countries.csv')
countries = countries[countries['country'].isin(population.geo.unique())]
countries.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 255 entries, 1 to 274
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   country                 255 non-null    object
 1   gwid                    254 non-null    object
 2   name                    255 non-null    object
 3   world_6region           254 non-null    object
 4   income_groups           213 non-null    object
 5   landlocked              254 non-null    object
 6   g77_and_oecd_countries  253 non-null    object
 7   main_religion_2008      216 non-null    object
 8   gapminder_list          254 non-null    object
 9   alternative_1           69 non-null     object
 10  alternative_2           37 non-null     object
 11  alternative_3           23 non-null     object
 12  alternative_4_cdiac     215 non-null    object
 13  pandg                   147 non-null    object
 14  god_id                  254 non-null    object
 15  alt_5                   19 non-null     object
 16  upper_case_name         239 non-null    object
 17  code                    231 non-null    object
 18  number                  231 non-null    float64
 19  arb1                    21 non-null     object
```

```
 20  arb2                    8 non-null      object
 21  arb3                    3 non-null      object
 22  arb4                    3 non-null      object
 23  arb5                    1 non-null      object
 24  arb6                    1 non-null      object
 25  is--country             255 non-null    bool
 26  world_4region           255 non-null    object
 27  latitude                238 non-null    float64
 28  longitude               238 non-null    float64
dtypes: bool(1), float64(3), object(25)
memory usage: 58.0+ KB
```

[ ]:
```
poverty = pd.read_csv('../datasets/poverty.csv')
poverty.sample(3)
```

[ ]:

	geo	time	extreme_poverty_percent_people_below_125_a_day
51	aus	2001	1.35
550	jam	1988	3.97
817	per	2011	2.97

Question 11. Assign `latest` to a three-column table with one row for each country that appears in the `poverty` table. The first column should contain the 3-letter code for the country. The second column should contain the *most recent year* for which an extreme poverty rate is available for the country. The third column should contain the poverty rate in that year. **Do not change the last line, so that the labels of your table are set correctly.**

Hint: the `first` function may be helpful, but you are not required to use it.

[ ]:
```
def first(values):
    return values.item(0)

latest = ...

# This line **should** work as it is, but you can change it
# as you see fit
latest.rename(
    columns={0: 'geo', 1: 'time', 2: 'poverty_percent'}
)
```

[ ]:
```
# We start by defining the helper function `first`
def first(values):
    return values.item(0)  # Extracts the first item of a series

# Using Group by 'geo' and find the most recent year for each country
latest = (
    poverty.sort_values('time', ascending=False)  # Sort by time (latest first)
    .groupby('geo')  # Group by country code
    .first()  # Take the first row in each group (most recent year)
    .reset_index()  # Reset index to get 'geo' as a column
```

```
)

# Theb we keep only the required columns
latest = latest[['geo', 'time',␣
↪'extreme_poverty_percent_people_below_125_a_day']]

# I renamed for clarity
latest = latest.rename(
    columns={
        'geo': 'geo',
        'time': 'time',
        'extreme_poverty_percent_people_below_125_a_day': 'poverty_percent'
    }
)

# display the first 5 with head .
latest.head()
```

[ ]:

	geo	time	poverty_percent
0	ago	2009	43.37
1	alb	2012	0.46
2	arg	2011	1.41
3	arm	2012	1.75
4	aus	2003	1.36

Question 12. Using both `latest` and `population`, create a four-column table called `recent` with one row for each country in `latest`. The four columns should have the following labels and contents: 1. `geo` contains the 3-letter country code, 1. `poverty_percent` contains the most recent poverty percent, 1. `population_total` contains the population of the country in 2010, 1. `poverty_total` contains the number of people in poverty **rounded to the nearest integer**, based on the 2010 population and most recent poverty rate.

[ ]:
```
poverty_and_pop = ...
recent = ...
recent
```

[ ]:
```
#First we need to filter the population table for the year 2010
population_2010 = population[population['time'] == 2010][['geo',␣
↪'population_total']]

# Then we merge the latest poverty data with the 2010 population data
poverty_and_pop = latest.merge(population_2010, on='geo')

# We then need to calculate  the total number of people in poverty
# Using the formula: (poverty_percent / 100) * population_total
poverty_and_pop['poverty_total'] = (
```

```
    (poverty_and_pop['poverty_percent'] / 100) *␣
  ↪poverty_and_pop['population_total']
).round(0).astype(int)  # Round to nearest integer and convert to int

# We keep only the required columns
recent = poverty_and_pop[['geo', 'poverty_percent', 'population_total',␣
  ↪'poverty_total']]

# Display the resulting table (first 5)
recent.head()
```

[ ]:

	geo	poverty_percent	population_total	poverty_total
0	ago	43.37	21219954	9203094
1	alb	0.46	2901883	13349
2	arg	1.41	41222875	581243
3	arm	1.75	2963496	51861
4	aus	1.36	22162863	301415

Question 13. Assuming that the `poverty_total` numbers in the `recent` table describe *all* people in 2010 living in extreme poverty, assign the name `poverty_percent` to the percentage of the world's 2010 population that were living in extreme poverty. You should find a number that is somewhat above the 2015 global estimate of 12%, since many country-specific poverty rates are older than 2015.

Hint: The sum of the `population_total` column in the `recent` table is not the world population, because only a subset of the world's countries have known poverty rates. Use the `population` table to compute the world's 2010 total population.

[ ]:

```
poverty_percent = ...
poverty_percent
```

[ ]:

```
#Total number of people in extreme poverty from the recent table
#*Hint*: The sum of the `population_total` column in the `recent` table is not␣
  ↪the world population, because only a subset of the world's countries have␣
  ↪known poverty rates.
total_poverty = recent['poverty_total'].sum()

# We need to calculate the world's total population in 2010 from the population␣
  ↪table
world_population_2010 = population[population['time'] ==␣
  ↪2010]['population_total'].sum()

# we compute the global poverty percentage
poverty_percent = (total_poverty / world_population_2010) * 100

poverty_percent
```

[ ]: 14.299370218520854

The `countries` table includes not only the name and region of countries, but also their positions on the globe.

```
[ ]: countries[['country', 'name', 'world_4region', 'latitude', 'longitude']]
```

```
[ ]:      country                   name world_4region  latitude  longitude
     1         afg            Afghanistan          asia  33.00000     66.000
     2    akr_a_dhe  Akrotiri and Dhekelia        europe       NaN        NaN
     3         alb                Albania        europe  41.00000     20.000
     4         dza                Algeria        africa  28.00000      3.000
     5         asm         American Samoa          asia -11.05600   -171.082
     ..        …                      …             …         …          …
     270       yem                  Yemen          asia  15.50000     47.500
     271       yug             Yugoslavia        europe       NaN        NaN
     272       zmb                 Zambia        africa -14.33333     28.500
     273       zwe               Zimbabwe        africa -19.00000     29.750
     274       ala                  Åland        europe  60.25000     20.000

     [255 rows x 5 columns]
```

You're finished! Congratulations on mastering data visualization and table manipulation. Time to submit.

UNIVERSITY OF OXFORD

Declaration of Authorship

Final Assignment - Python for Data Science Course

I, Pedro Jorge Almada Melendez, confirm that the work I am submitting for the assignment titled "Second-Assignment_Pedro_Almada" is my own work and has been completed by the requirements of the assignment.
Date: 17/12/2024

Signature:

UNIVERSITY OF OXFORD, DEPARTMENT FOR CONTINUING EDUCATION
Ewert House, Ewert Place, Oxford, OX2 7DD. Tel: Oxford (01865) 280900

WEEKLY CLASS ASSESSMENT - DECLARATION OF AUTHORSHIP

Name (in capitals):	PEDRO JORGE ALMADA MELENDEZ	**Assignment Deadline:**
Course Title:	Python for Data Science – Introduction	20/12/2024
Term:		**Date Submitted:**
Tutor:	Cristian Soitu	**17/20/2024**
Title of Assignment: ***Not required for Language courses***	Second Assignment	**Word Count:** (If applicable)

Please sign to confirm the following:

I, declare that:

- I am aware of the University's guidance on plagiarism https://www.ox.ac.uk/students/academic/guidance/skills/plagiarism?wssl=1
- I have read and understood the Department's information and guidance on academic good practice and plagiarism given in the Guide to Producing Coursework
- The work I am submitting is entirely my own work except where otherwise indicated
- It has not been submitted, either wholly or substantially, for another course of this Department or University, or for a course at any other institution
- I have clearly indicated the presence of all material I have quoted from other sources, including any images, diagrams, charts, tables or graphs
- I have clearly signalled the presence of quoted or paraphrased material and referenced all sources
- I have acknowledged appropriately any assistance I have received in addition to that provided by my tutor
- I have not copied from the work of any other student
- I have not used the services of any agency providing specimen, model or ghost-written work in the preparation of this submitted work
- I agree to retain a copy of this work until receipt of my final result
- I agree to make any such electronic copy available to the Director of Weekly Classes should it be necessary to confirm my word count or to check for plagiarism
- I have not used the services of any agency providing specimen, model or ghost-written work in the preparation of this [thesis/dissertation/extended essay/assignment/project/other submitted work]. (See also section 2.4 of Statute XI: University Discipline under which members of the University are prohibited from providing material of this nature for candidates in examinations at this University or elsewhere). Sections 3 and 4 of Proctors' Regulations 1 of 2003 dealing with unauthorised use of AI, also apply.

Candidate's Signature:		**Date:**	17/12/2024

DA Form