Coronavirus World Data Analysis

- Remember to uncomment the line assigning the variable to your answer and don't change the variable or function names.
- Use copies of the original or previous DataFrames to make sure you do not overwrite them by mistake.

First of all, run the following cell to:

- import `pandas` with an alias of `pd`
- read a CSV containing the data to work with
- convert the `date` column to the `datetime` format
- create a DataFrame `df` containing the data for only 1st July 2020
- take a look at the first few rows of the DataFrame

```
import pandas as pd

data = pd.read_csv('data/owid-covid-data.csv')
data['date'] = pd.to_datetime(data['date'])
df = data.data['date'] == '2020-07-01']

df.head()
```

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_...
173	AFG	Asia	Afghanistan	2020-07-01	31517.0	279.0	746.0	13.0	...
300	ALB	Europe	Albania	2020-07-01	2535.0	69.0	62.0	4.0	...
491	DZA	Africa	Algeria	2020-07-01	13907.0	336.0	912.0	7.0	...
613	AND	Europe	Andorra	2020-07-01	855.0	0.0	52.0	0.0	11...
727	AGO	Africa	Angola	2020-07-01	284.0	8.0	13.0	2.0	...

5 rows × 34 columns

`df` now has one row of data for each country with data present for July 1st 2020. However, it also has a row with a `location` of `World` which contains aggregated values for all countries.

Q1. Create a new DataFrame which is the same as `df` but with the `World` row removed.

Assign this new DataFrame to the variable `countries` do not modify `df`.

```
countries = df[df['location'] != 'World'].copy()
countries.head()
```

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_p...
173	AFG	Asia	Afghanistan	2020-07-01	31517.0	279.0	746.0	13.0	...
300	ALB	Europe	Albania	2020-07-01	2535.0	69.0	62.0	4.0	...
491	DZA	Africa	Algeria	2020-07-01	13907.0	336.0	912.0	7.0	...
613	AND	Europe	Andorra	2020-07-01	855.0	0.0	52.0	0.0	...
727	AGO	Africa	Angola	2020-07-01	284.0	8.0	13.0	2.0	...
...	...	...	...	...	...	...	...	...	...
29332	VNM	Asia	Vietnam	2020-07-01	355.0	0.0	0.0	0.0	...
29411	ESH	Africa	Western Sahara	2020-07-01	770.0	172.0	1.0	0.0	...
29506	YEM	Asia	Yemen	2020-07-01	1158.0	30.0	312.0	8.0	...
29623	ZMB	Africa	Zambia	2020-07-01	1594.0	26.0	24.0	2.0	...
29738	ZWE	Africa	Zimbabwe	2020-07-01	591.0	17.0	7.0	0.0	...

210 rows × 34 columns

Q2. Check the shape of your DataFrame to confirm that `countries` has one row fewer than `df`:

```
print(df.shape, countries.shape)
```

```
(211, 34) (210, 34)
```

```
cols = ['continent', 'location', 'total_deaths_per_million']
```

Q3. Define a DataFrame based on the `countries` DataFrame, but which only contains the columns in `cols` (defined above) and assign this to a variable called `countries_dr`.

Order this DataFrame by `total_deaths_per_million`, with the highest numbers at the top.

```
countries_dr = countries[cols].sort_values('total_deaths_per_million', ascending=False)
countries_dr
```

	continent	location	total_deaths_per_million
23306	Europe	San Marino	1237.551
2917	Europe	Belgium	841.615
613	Europe	Andorra	673.008
28347	Europe	United Kingdom	644.168
25362	Europe	Spain	606.633
...	...	...	...
23111	North America	Saint Vincent and the Grenadines	0.000
23926	Africa	Seychelles	0.000
15734	Africa	Lesotho	0.000
10808	Europe	Gibraltar	0.000
12195	Asia	Hong Kong	NaN

210 rows × 3 columns

Q4. Using the `countries` DataFrame we created earlier, find the sum of `total_tests` for countries in `Africa`, *assigning the result, as an integer,* to `africa_tests`.

```
africa_tests = countries[countries["continent"] == "Africa"]["total_tests"].sum().astype(int)
africa_tests
```

```
3445134
```

Q5. How many countries in Africa have no value recorded for the number of `total_tests`? Assign the result to `africa_missing_test_data`.

You may find the pandas `.isnull()` method useful.

```
africa_missing_test_data = countries[countries["continent"] == "Africa"]["total_tests"].isnull().sum()
africa_missing_test_data
```

```
45
```

Q6. How many countries have a higher value for `total_tests` than the `United Kingdom`? Assign your answer to a variable called `countries_more_tests`.

Remember to work from the `countries` DataFrame rather than `df`. You should avoid modifying any existing DataFrames.

```
uk = countries.loc[countries["location"] == "United Kingdom"].index[0], "total_tests"]
countries_more_tests = countries["total_tests"] > uk).value_counts()[True]

countries_more_tests
```

```
3
```

Q7. Create a DataFrame called `beds_dr` which is based on the `countries` DataFrame, but contains only the columns `hospital_beds_per_thousand` and `total_deaths_per_million`.

Your answer should only include rows where there are values present in both of these columns. *You may find the `.dropna()` method useful.*

```
beds_dr = countries[["hospital_beds_per_thousand", "total_deaths_per_million"]].dropna()
beds_dr
```

	hospital_beds_per_thousand	total_deaths_per_million
173	0.50	19.163
300	2.89	21.544
491	1.90	20.798
952	3.80	30.635
1081	5.00	28.919
...	...	...
29136	0.80	1.794
29332	2.60	0.000
29506	0.70	10.461
29623	2.00	1.305
29738	1.70	0.471

164 rows × 2 columns

```
beds_dr
```

	hospital_beds_per_thousand	total_deaths_per_million
173	0.50	19.163
300	2.89	21.544
491	1.90	20.798
952	3.80	30.635
1081	5.00	28.919
...	...	...
29136	0.80	1.794
29332	2.60	0.000
29506	0.70	10.461
29623	2.00	1.305
29738	1.70	0.471

164 rows × 2 columns

Q8. What is the average `total_deaths_per_million` for entries in `beds_dr` where `hospital_beds_per_thousand` is greater than the mean?

Assign the answer to `dr_high_bed_ratio`.

```
dr_high_bed_ratio = beds_dr[beds_dr["hospital_beds_per_thousand"] > beds_dr['hospital_beds_per_thousand'].mean()]['total_deaths_per_million'].mean()
dr_high_bed_ratio
```

```
90.18423728813558
```

Q9. What is the average `total_deaths_per_million` for entries in `beds_dr` where `hospital_beds_per_thousand` is less than the mean?

Assign the answer to `dr_low_bed_ratio`.

```
dr_low_bed_ratio = beds_dr[beds_dr['hospital_beds_per_thousand'] < beds_dr['hospital_beds_per_thousand'].mean()]['total_deaths_per_million'].mean()
dr_low_bed_ratio
```

```
54.29455742857174
```

Q10. Create a DataFrame called `no_new_cases` which contains only rows from `countries` with zero `new_cases`.

```
no_new_cases = countries[countries['new_cases'] == 0].copy()
no_new_cases
```

	iso_code	continent	location	date	total_cases	new_cases	total_deaths	new_deaths	total_cases_per_...
613	AND	Europe	Andorra	2020-07-01	855.0	0.0	52.0	0.0	11...
836	AIA	North America	Anguilla	2020-07-01	3.0	0.0	0.0	0.0	...
952	ATG	North America	Antigua and Barbuda	2020-07-01	66.0	0.0	3.0	0.0	...
1381	ABW	North America	Aruba	2020-07-01	103.0	0.0	3.0	0.0	...
2080	BHS	North America	Bahamas	2020-07-01	104.0	0.0	11.0	0.0	...
...	...	...	...	...	...	...	...	...	...
27103	TLS	Asia	Timor	2020-07-01	24.0	0.0	0.0	0.0	...
27725	TCA	North America	Turks and Caicos Islands	2020-07-01	41.0	0.0	2.0	1.0	1...
28654	VIR	North America	United States Virgin Islands	2020-07-01	84.0	0.0	6.0	0.0	...
29016	VAT	Europe	Vatican	2020-07-01	12.0	0.0	0.0	0.0	14...
29332	VNM	Asia	Vietnam	2020-07-01	355.0	0.0	0.0	0.0	...

62 rows × 34 columns

Q11. Which country in `no_new_cases` has had the highest number of `total_cases`? Assign your answer to `highest_no_new`.

```
highest = no_new_cases['total_cases'].max()

index = no_new_cases[no_new_cases['total_cases'] == highest].index[0]

highest_no_new = no_new_cases.loc[index, 'location']
highest_no_new
```

```
'Cameroon'
```

Q12. What is the sum of the `population` of all countries which have had zero `total_deaths`?

Assign your answer to `sum_populations_no_deaths`. Your answer should be in millions, rounded to the nearest whole number, and converted to an integer.

```
sum_populations_no_deaths = (df[df['total_deaths'] == 0]['population'].sum() / 1000000).astype(int)
sum_populations_no_deaths
```

```
192
```

Q13. Create a function called `country_metric` which accepts the following three parameters:

- a DataFrame (which can be assumed to be of a similar format to `countries`)
- a location (i.e. a string which will be found in the `location` column of the DataFrame)
- a string (which can be assumed to be a column (other than `location`) which will be found in the DataFrame)

The function should return only the value from the first row for a given `location` and `metric`. *You may find `.iloc[]` useful.*

```
def country_metric(df, location, metric):
    return df.loc[df['location'] == location, metric].values[0]
```

Q.14 Use your function to collect the value for `Vietnam` for the metric `aged_70_older`, assigning the result to `vietnam_older_70`.

```
vietnam_older_70 = country_metric(countries, 'Vietnam', 'aged_70_older')
vietnam_older_70
```

```
4.718
```

Q.15 Create another function called `countries_average`, which accepts the following three parameters:

- a DataFrame (which can be assumed to be such as `countries`)
- a list of countries (which can be assumed to be a column (other than `location`) which will be found in the DataFrame)
- a string (which can be assumed to be a column (other than `location`) which will be found in the DataFrame)

The function should return the average value for the given metric for the given list of countries.

```
def countries_average(df, countries, metric):
    val = 0
    for c in countries:
        c = country_metric(df, c, metric)
        val+=c
    return val/val / len(val)
```

```
g7 = ['United States', 'Italy', 'Canada', 'Japan', 'United Kingdom', 'Germany', 'France']
```

Q16. Use your `countries_average` function to find out the average `life_expectancy` of countries in the `g7` list defined above. Assign the result to the variable `g7_avg_life_expectancy`.

```
g7_avg_life_expectancy = countries_average(countries, g7, 'life_expectancy')
g7_avg_life_expectancy
```

```
82.13571428571429
```

Q.17 Find the country with lowest value for `life_expectancy` in the `countries` DataFrame, and create a string which is formatted as follows:

'(country) has a life expectancy of (d)f) years lower than the G7 average.'

Assign your string to the variable `headline` and ensure it is formatted exactly as above, with:

- (country) being replaced by the value in the `location` column of the DataFrame
- (d)f) being replaced by a float **rounded to one decimal place**, of the value from the `life_expectancy` column subtracted from `g7_avg_life_expectancy`.
- Please note that (d)f) should be a positive value

```
abs(-12)
```

```
12
```

```
index = countries[countries['life_expectancy'] == countries['life_expectancy'].min()].index[0]
row = countries.loc[index]
diff = abs(g7_avg_life_expectancy - row['life_expectancy']).round(1)

headline = "{} has a life expectancy of {} years lower than the G7 average.".format(row['location'], diff)
headline
```

```
'Central African Republic has a life expectancy of 28.8 years lower than the G7 average.'
```