FIT1043 Introduction to Data Science Assignment 1

Ian WONG

Student ID: 30612616

Importing Libraries

The first library we will be importing is **pandas**, which is used to create data structures called *DataFrames*. These dataframes allow use to manage and manipulate data more easily.

We will also be importing **matplotlib.pyplot**, which is used to plot and visualise data in Python. This can also be done later on, if you prefer to order your code sequentially.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

Reading datasets

Using the `read_csv` command from **pandas**, we are reading in the datasets from the data folder and assigning them to:

- `vaccinations` for 'Country-Vaccinations.csv'
- `gdp` for '2020-GDP.csv'
- `pop` for '2020-Population.csv'

In [2]:
```
vaccinations = pd.read_csv('data/Country-Vaccinations.csv')
gdp = pd.read_csv('data/2020-GDP.csv')
pop = pd.read_csv('data/2020-Population.csv')
```

To make sure we have read the data in properly, we will be checking the head, tail, and a random sample of each dataset:

In [3]:
```
vaccinations.head()
```

Out[3]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN	
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN	

In [4]:
```
vaccinations.tail()
```

Out[4]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	da
36731	Zimbabwe	ZWE	2021-08-04	2604265.0	1740598.0	863667.0	
36732	Zimbabwe	ZWE	2021-08-05	2701095.0	1780541.0	920554.0	
36733	Zimbabwe	ZWE	2021-08-06	2784270.0	1817598.0	966672.0	
36734	Zimbabwe	ZWE	2021-08-07	2853668.0	1851407.0	1002261.0	
36735	Zimbabwe	ZWE	2021-08-08	2886822.0	1864204.0	1022618.0	

In [5]:
```
vaccinations.sample(5)
```

Out[5]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	da
10686	Ethiopia	ETH	2021-04-13	NaN	NaN	NaN	
11992	Gambia	GMB	2021-03-13	NaN	NaN	NaN	
14240	Honduras	HND	2021-03-19	37317.0	37317.0	NaN	
33168	Timor	TLS	2021-05-10	NaN	NaN	NaN	
27657	Russia	RUS	2021-07-01	40875355.0	23352344.0	17523011.0	

In [6]:
```
gdp.head()
```

Out[6]:

	Unnamed: 0	Gross domestic product 2020	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	(millions of	NaN
2	NaN	Ranking	NaN	Economy	US dollars)	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN
4	USA	1	NaN	United States	20,936,600	NaN

In [7]:
```
gdp.tail()
```

Out[7]:

	Unnamed: 0	Gross domestic product 2020	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
324	NaN	NaN	NaN	NaN	NaN	NaN
325	NaN	NaN	NaN	NaN	NaN	NaN
326	NaN	NaN	NaN	NaN	NaN	NaN
327	NaN	NaN	NaN	NaN	NaN	NaN
328	NaN	NaN	NaN	NaN	NaN	NaN

In [8]:
```
gdp.sample(5)
```

Out[8]:

	Unnamed: 0	Gross domestic product 2020	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
201	TON	198	NaN	Tonga	512	NaN
68	ECU	65	NaN	Ecuador	98,808	NaN
100	NPL	97	NaN	Nepal	33,657	NaN
296	NaN	NaN	NaN	NaN	NaN	NaN
91	TKM	88	NaN	Turkmenistan	45,231	NaN

In [9]:
```
pop.head()
```

Out[9]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unna
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	United Nations	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	Population Division	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

5 rows × 78 columns

In [10]:
```
pop.tail()
```

Out[10]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7
300	285	Estimates	Bermuda	14	60	Country/Area	918	37
301	286	Estimates	Canada	NaN	124	Country/Area	918	13 733
302	287	Estimates	Greenland	26	304	Country/Area	918	23
303	288	Estimates	Saint Pierre and Miquelon	2	666	Country/Area	918	5
304	289	Estimates	United States of America	35	840	Country/Area	918	158 804

5 rows × 78 columns

In [11]:
```
pop.sample(5)
```

Out[11]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7
138	123	Estimates	Nepal	NaN	524	Country/Area	5501	8 483
155	140	Estimates	Lao People's Democratic Republic	NaN	418	Country/Area	920	1 683
120	105	Estimates	Syrian Arab Republic	NaN	760	Country/Area	922	3 413
66	51	Estimates	Chad	NaN	148	Country/Area	911	2 502
225	210	Estimates	Solomon Islands	NaN	90	Country/Area	928	90

5 rows × 78 columns

Data Wrangling

In this section, we will be sub-setting the necessary data that is required.

Vaccination DataFrame

For the vaccination related DataFrame, we are keeping the columns:

- `country` ,
- `people_fully_vaccinated` ,
- `total_vaccinations` (as an aggregation of `daily_vaccinations`),

- and `vaccines`

Using `.groupby()` and `.agg()`, we are grouping the dataframe by country and finding:

- max of people fully vaccinated
- sum of daily vaccinations
- vaccines available in a country

In [12]:
```
fun = {'people_fully_vaccinated':'max',
       'daily_vaccinations':'sum',
       'vaccines':'max'}
```

In [13]:
```
vaccinations = vaccinations.groupby(['country']).agg(fun)
vaccinations
```

Out[13]:

country	people_fully_vaccinated	daily_vaccinations	vaccines
Afghanistan	219159.0	1649463.0	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
Albania	553482.0	1237306.0	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, ...
Algeria	724812.0	4075025.0	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
Andorra	33904.0	76689.0	Oxford/AstraZeneca, Pfizer/BioNTech
Angola	722610.0	1690469.0	Oxford/AstraZeneca
...	...	...	...
Wales	2115477.0	4396339.0	Moderna, Oxford/AstraZeneca, Pfizer/BioNTech
Wallis and Futuna	4659.0	8937.0	Oxford/AstraZeneca
Yemen	13322.0	302943.0	Oxford/AstraZeneca
Zambia	193603.0	475566.0	Oxford/AstraZeneca, Sinopharm/Beijing
Zimbabwe	1022618.0	2693298.0	Sinopharm/Beijing, Sinovac, Sputnik V

222 rows × 3 columns

Using `.rename()`, we are renaming the columns into more appropriate and formatted labels:

In [14]:
```
vaccinations.rename(
    columns = {'people_fully_vaccinated':'People Fully Vaccinated',
               'daily_vaccinations':'Total Vaccinations',
               'vaccines':'Vaccines Available'},
    inplace = True
)
vaccinations
```

Out[14]:

	People Fully Vaccinated	Total Vaccinations	Vaccines Available

country	People Fully Vaccinated	Total Vaccinations	Vaccines Available
country			
Afghanistan	219159.0	1649463.0	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
Albania	553482.0	1237306.0	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, ...
Algeria	724812.0	4075025.0	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
Andorra	33904.0	76689.0	Oxford/AstraZeneca, Pfizer/BioNTech
Angola	722610.0	1690469.0	Oxford/AstraZeneca
...	...	...	...
Wales	2115477.0	4396339.0	Moderna, Oxford/AstraZeneca, Pfizer/BioNTech
Wallis and Futuna	4659.0	8937.0	Oxford/AstraZeneca
Yemen	13322.0	302943.0	Oxford/AstraZeneca
Zambia	193603.0	475566.0	Oxford/AstraZeneca, Sinopharm/Beijing
Zimbabwe	1022618.0	2693298.0	Sinopharm/Beijing, Sinovac, Sputnik V

222 rows × 3 columns

To flatten the columns, we use `.reset_index()` :

In [15]:
```
vaccinations = vaccinations.reset_index()
vaccinations
```

Out[15]:

	country	People Fully Vaccinated	Total Vaccinations	Vaccines Available
0	Afghanistan	219159.0	1649463.0	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
1	Albania	553482.0	1237306.0	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, ...
2	Algeria	724812.0	4075025.0	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
3	Andorra	33904.0	76689.0	Oxford/AstraZeneca, Pfizer/BioNTech
4	Angola	722610.0	1690469.0	Oxford/AstraZeneca
...	...	...	...	...
217	Wales	2115477.0	4396339.0	Moderna, Oxford/AstraZeneca, Pfizer/BioNTech
218	Wallis and Futuna	4659.0	8937.0	Oxford/AstraZeneca
219	Yemen	13322.0	302943.0	Oxford/AstraZeneca
220	Zambia	193603.0	475566.0	Oxford/AstraZeneca, Sinopharm/Beijing

	country	People Fully Vaccinated	Total Vaccinations	Vaccines Available
221	Zimbabwe	1022618.0	2693298.0	Sinopharm/Beijing, Sinovac, Sputnik V

222 rows × 4 columns

Check that we didn't drop any columns:

In [16]:
```
vaccinations.columns
```

Out[16]:
```
Index(['country', 'People Fully Vaccinated', 'Total Vaccinations',
       'Vaccines Available'],
      dtype='object')
```

Use `.rename()` to format the column name for countries:

In [17]:
```
vaccinations.rename(columns = {'country':'Country'},inplace = True)
vaccinations
```

Out[17]:

	Country	People Fully Vaccinated	Total Vaccinations	Vaccines Available
0	Afghanistan	219159.0	1649463.0	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...
1	Albania	553482.0	1237306.0	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, ...
2	Algeria	724812.0	4075025.0	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...
3	Andorra	33904.0	76689.0	Oxford/AstraZeneca, Pfizer/BioNTech
4	Angola	722610.0	1690469.0	Oxford/AstraZeneca
...	...	...	...	...
217	Wales	2115477.0	4396339.0	Moderna, Oxford/AstraZeneca, Pfizer/BioNTech
218	Wallis and Futuna	4659.0	8937.0	Oxford/AstraZeneca
219	Yemen	13322.0	302943.0	Oxford/AstraZeneca
220	Zambia	193603.0	475566.0	Oxford/AstraZeneca, Sinopharm/Beijing
221	Zimbabwe	1022618.0	2693298.0	Sinopharm/Beijing, Sinovac, Sputnik V

222 rows × 4 columns

GDP DataFrame

For the GDP DataFrame, we are keeping the GDP values for each country.

First, by looking at the structure of the GDP DataFrame below, we can determine that the column headings are down below on index 2, and the data starts on index 4.

Therefore, we slice the data and rename to format the data:

In [18]:
```
gdp.head()
```

Out[18]:

	Unnamed: 0	Gross domestic product 2020	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	(millions of	NaN
2	NaN	Ranking	NaN	Economy	US dollars)	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN
4	USA	1	NaN	United States	20,936,600	NaN

Slicing the data from index 4:

In [19]:
```
gdp = gdp[4:]
gdp
```

Out[19]:

	Unnamed: 0	Gross domestic product 2020	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
4	USA	1	NaN	United States	20,936,600	NaN
5	CHN	2	NaN	China	14,722,731	NaN
6	JPN	3	NaN	Japan	5,064,873	NaN
7	DEU	4	NaN	Germany	3,806,060	NaN
8	GBR	5	NaN	United Kingdom	2,707,744	NaN
...	...	...	...	...	...	...
324	NaN	NaN	NaN	NaN	NaN	NaN
325	NaN	NaN	NaN	NaN	NaN	NaN
326	NaN	NaN	NaN	NaN	NaN	NaN
327	NaN	NaN	NaN	NaN	NaN	NaN
328	NaN	NaN	NaN	NaN	NaN	NaN

325 rows × 6 columns

Using `.loc` , we keep certain columns:

In [20]:
```
gdp = gdp.loc[:, ('Unnamed: 3', 'Unnamed: 4')]
gdp
```

Out[20]:

	Unnamed: 3	Unnamed: 4
4	United States	20,936,600
5	China	14,722,731
6	Japan	5,064,873
7	Germany	3,806,060
8	United Kingdom	2,707,744

	Unnamed: 3	Unnamed: 4
...	...	...
324	NaN	NaN
325	NaN	NaN
326	NaN	NaN
327	NaN	NaN
328	NaN	NaN

325 rows × 2 columns

Renaming the rows from to `Country` and `GDP` :

In [21]:
```
gdp.rename(
    columns = {'Unnamed: 3': 'Country',
               'Unnamed: 4': 'GDP'},
    inplace = True
)
gdp
```

Out[21]:

	Country	GDP
4	United States	20,936,600
5	China	14,722,731
6	Japan	5,064,873
7	Germany	3,806,060
8	United Kingdom	2,707,744
...	...	...
324	NaN	NaN
325	NaN	NaN
326	NaN	NaN
327	NaN	NaN
328	NaN	NaN

325 rows × 2 columns

Dropping rows with `NaN` values and resetting the index:

In [22]:
```
gdp = gdp.dropna()
gdp = gdp.reset_index().drop(columns = ['index'])
gdp
```

Out[22]:

	Country	GDP
0	United States	20,936,600
1	China	14,722,731
2	Japan	5,064,873

	Country	GDP
3	Germany	3,806,060
4	United Kingdom	2,707,744
...	...	...
224	Sub-Saharan Africa	1,685,632
225	Low income	546,661
226	Lower middle income	7,328,774
227	Upper middle income	23,630,843
228	High income	53,255,167

229 rows × 2 columns

Population DataFrame

For the population dataframe, we are keeping the population values for each country as of 2020.

By observing the first 20 indexes of the pop dataframe, we can see that the headings start on index 15 and the data starts on index 16.

Therefore, we slice the data from index 16 onwards and rename the headers:

In [23]:
```
pop.head(20)
```

Out[23]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN
3	United Nations	NaN	NaN	NaN	NaN	NaN
4	Population Division	NaN	NaN	NaN	NaN	NaN
5	Department of Economic and Social Affairs	NaN	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	NaN	NaN
7	World Population Prospects 2019	NaN	NaN	NaN	NaN	NaN
8	File POP/1-1: Total population (both sexes com...	NaN	NaN	NaN	NaN	NaN
9	Estimates, 1950 - 2020	NaN	NaN	NaN	NaN	NaN
10	POP/DB/WPP/Rev.2019/POP/F01-1	NaN	NaN	NaN	NaN	NaN
11	© August 2019 by United Nations, made availabl...	NaN	NaN	NaN	NaN	NaN
12	Suggested citation: United Nations, Department...	NaN	NaN	NaN	NaN	NaN

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
13	NaN	NaN	NaN	NaN	NaN	NaN
14	NaN	NaN	NaN	NaN	NaN	NaN
15	Index	Variant	Region, subregion, country or area *	Notes	Country code	Type
16	1	Estimates	WORLD	NaN	900	World
17	2	Estimates	UN development groups	a	1803	Label/Separator
18	3	Estimates	More developed regions	b	901	Development Group
19	4	Estimates	Less developed regions	c	902	Development Group

20 rows × 78 columns

Slicing the dataframe from index 16 onwards and keeping the columns that correspond to country and the population as of 2020:

In [24]:
```
pop = pop[16:]
pop = pop.loc[:, ('Unnamed: 2', 'Unnamed: 77')]
pop
```

Out[24]:

	Unnamed: 2	Unnamed: 77
16	WORLD	7 794 799
17	UN development groups	...
18	More developed regions	1 273 304
19	Less developed regions	6 521 494
20	Least developed countries	1 057 438
...	...	...
300	Bermuda	62
301	Canada	37 742
302	Greenland	57
303	Saint Pierre and Miquelon	6
304	United States of America	331 003

289 rows × 2 columns

Using `.rename()`, we are renaming the columns to `Country` and `Population`.

Afterwards, we use `.reset_index()` and `.drop()` to reset the index:

In [25]:
```
pop.rename(
    columns = {'Unnamed: 2':'Country',
               'Unnamed: 77':'Population'},
    inplace = True
)
pop = pop.reset_index().drop(columns = ['index'])
```

In [26]:
```
pop
```

Out[26]:

	Country	Population
0	WORLD	7 794 799
1	UN development groups	...
2	More developed regions	1 273 304
3	Less developed regions	6 521 494
4	Least developed countries	1 057 438
...	...	...
284	Bermuda	62
285	Canada	37 742
286	Greenland	57
287	Saint Pierre and Miquelon	6
288	United States of America	331 003

289 rows × 2 columns

Subset of Countries

In this assignment, we need to work with a subset of data that includes

- Indonesia,
- Malaysia,
- Singapore,
- Thailand,
- Philipines,
- and Australia

We put them into a list so that we can use it later on in conjunction with `.isin()`.

In [27]:
```
countries = ['Indonesia', 'Malaysia', 'Singapore', 'Thailand', 'Philippines', 'Austr
```

Now, using `.isin` subset the data to only the countries that are needed:

In [28]:
```
vaccinations = vaccinations[vaccinations['Country'].isin(countries)]
gdp = gdp[gdp['Country'].isin(countries)]
```

```
pop = pop[pop['Country'].isin(countries)]
```

Merging datasets

Using the `.merge()` command from **pandas**, we are combining the datasets together into one:

In [29]:
```
merged = pd.merge(vaccinations,gdp,on=['Country'])
merged = pd.merge(merged,pop,on=['Country'])
```

In [30]:
```
merged
```

Out[30]:

	Country	People Fully Vaccinated	Total Vaccinations	Vaccines Available	GDP	Population
0	Australia	4614203.0	13222783.0	Oxford/AstraZeneca, Pfizer/BioNTech	1,330,901	25 500
1	Indonesia	24481296.0	72386296.0	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	1,058,424	273 524
2	Malaysia	9048634.0	23687251.0	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	336,664	32 366
3	Philippines	11614590.0	23230492.0	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	361,489	109 581
4	Singapore	3862510.0	7911869.0	Moderna, Pfizer/BioNTech	339,998	5 850
5	Thailand	4277071.0	18349011.0	Oxford/AstraZeneca, Sinovac	501,795	69 800

Creating a new column 'perCapitaGDP'

We are creating a new column called `perCapitaGDP` which is the GDP divided by population.

First, we check the types of the columns:

In [31]:
```
merged.dtypes
```

Out[31]:
```
Country                   object
People Fully Vaccinated  float64
Total Vaccinations       float64
Vaccines Available        object
GDP                       object
Population                object
dtype: object
```

We can see that the data type for GDP and Population are objects. Therefore, we need to change the data type to numeric in order to divide GDP by Population.

To do this, we need to replace ',' and blank spaces in `GDP` and `Population` in order to convert them into numeric data types:

In [32]:
```
merged['GDP'] = pd.to_numeric(
    merged['GDP'].str.replace(',', ''), errors='coerce')
merged['Population'] = pd.to_numeric(
    merged['Population'].str.replace(' ', ''), errors='coerce')
```

Then we proceed to create `perCapitaGDP`:

In [33]:
```
merged['perCapitaGDP'] = merged['GDP']/merged['Population']
```

In [34]:
```
merged
```

Out[34]:

	Country	People Fully Vaccinated	Total Vaccinations	Vaccines Available	GDP	Population	perCapitaGDP
0	Australia	4614203.0	13222783.0	Oxford/AstraZeneca, Pfizer/BioNTech	1330901	25500	52.192196
1	Indonesia	24481296.0	72386296.0	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	1058424	273524	3.869584
2	Malaysia	9048634.0	23687251.0	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	336664	32366	10.401780
3	Philippines	11614590.0	23230492.0	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	361489	109581	3.298829
4	Singapore	3862510.0	7911869.0	Moderna, Pfizer/BioNTech	339998	5850	58.119316
5	Thailand	4277071.0	18349011.0	Oxford/AstraZeneca, Sinovac	501795	69800	7.189040

Statistical Description

Looking at the data in `merged`, there are a few things that we can note:

- Indonesia has the most total number of vaccinations among the countries,
- Singapore has the most proportion of their population fully vaccinated at ~66%
- The mean perCapitaGDP is 22.51 thousand USD
- The mean GDP is 86,103.5 mil USD

Question 1

In order to visualise the estimated number of people vaccinated for each vaccine type, we could use a **pie graph** that shows the proportion of the population that is vaccinated for each vaccine type.

If you wanted to compare the countries' populations, you could also use a segmented bar graph.

First, we want to find the number of vaccines available to each country.

In [35]:
```
merged['Vaccines Available Count'] = merged['Vaccines Available'].str.count(',') + 1
merged
```

Out[35]:

	Country	People Fully Vaccinated	Total Vaccinations	Vaccines Available	GDP	Population	perCapitaGDP	Va Av
0	Australia	4614203.0	13222783.0	Oxford/AstraZeneca, Pfizer/BioNTech	1330901	25500	52.192196	
1	Indonesia	24481296.0	72386296.0	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	1058424	273524	3.869584	
2	Malaysia	9048634.0	23687251.0	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	336664	32366	10.401780	
3	Philippines	11614590.0	23230492.0	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	361489	109581	3.298829	
4	Singapore	3862510.0	7911869.0	Moderna, Pfizer/BioNTech	339998	5850	58.119316	
5	Thailand	4277071.0	18349011.0	Oxford/AstraZeneca, Sinovac	501795	69800	7.189040	

For a certain country (say the Philippines), we can find all the vaccines available and split them into a list using `.str.split(',')` and splitting where there are commas:

In [36]:
```
phil_vaccines = merged.loc[merged['Country'] == 'Philippines']['Vaccines Available']
phil_vaccines
```

Out[36]:
```
['Johnson&Johnson',
 ' Moderna',
 ' Oxford/AstraZeneca',
 ' Pfizer/BioNTech',
 ' Sinovac',
 ' Sputnik V']
```

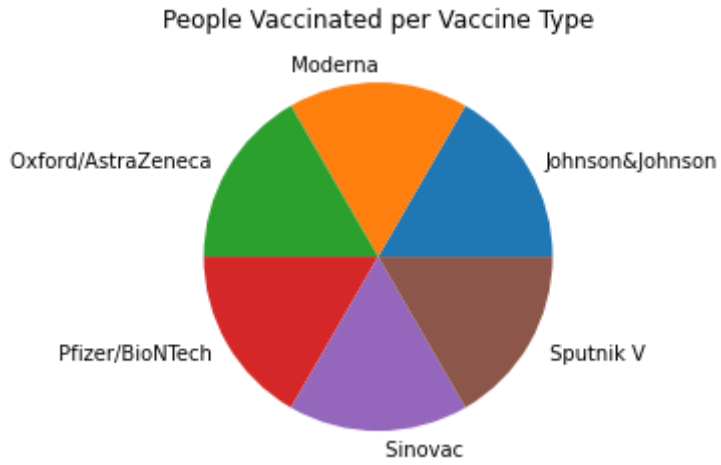
Using basic python, we create a list of numbers that are split evenly into the number of vaccines available:

In [37]:
```
phil_vaccine_n = merged.loc[merged['Country']=='Philippines']['Vaccines Available Co
phil_vaccine_n
n_list = [1/phil_vaccine_n]*phil_vaccine_n
```

Lastly, we make the pie graph showing the number of people vaccinated per vaccine type if they were equally distributed to the country's population:

In [38]:
```
plt.pie(n_list, labels=phil_vaccines)
plt.title('People Vaccinated per Vaccine Type')
plt.show
```

Out[38]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



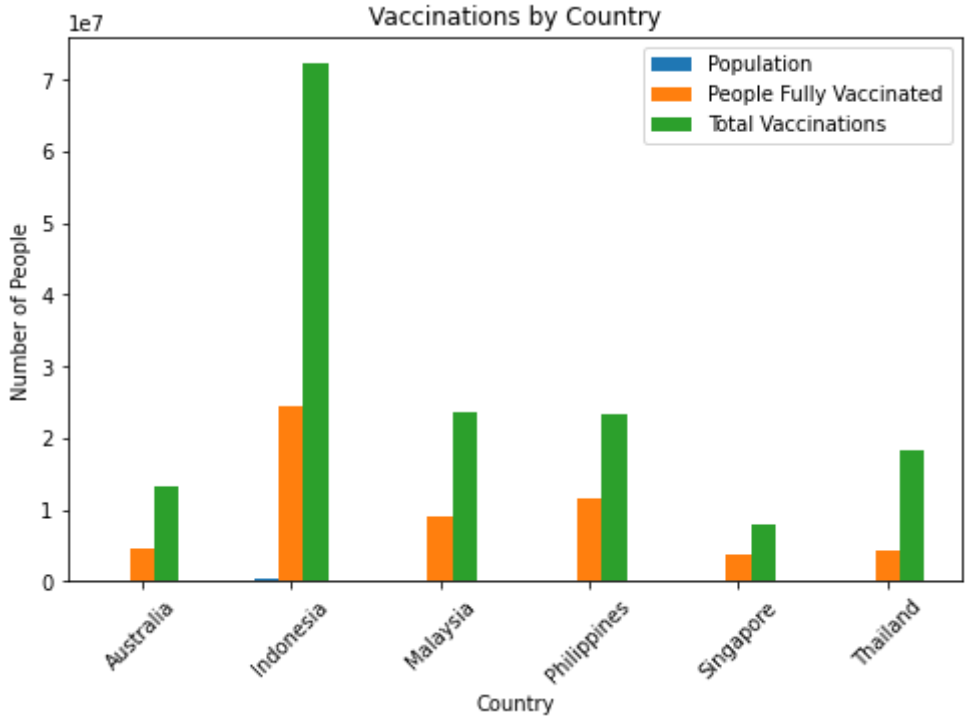
As one can see, the graph is straightforward as the vaccines have been distributed equally. Therefore, the population is approximately using each vaccine equally, creating an equal pie graph.

Question 2

For each country, we are going to plot a bar graph with side-by-side bars for `Population`, `Total Vaccinations`, and `People Fully Vaccinated`.

Using `plt.bar` from the `matplotlib` library, we are creating three side by side bars for each country:

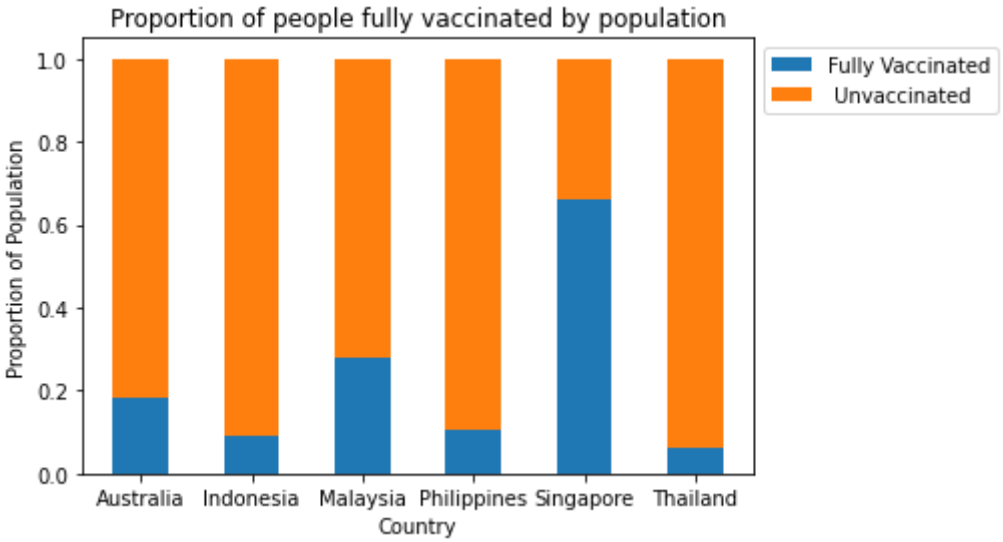
In [39]:
```
merged1 = merged.loc[:, ('Population', 'People Fully Vaccinated', 'Total Vaccination
ax = merged1.plot.bar(figsize = (8,5))
ax.set_xticklabels(merged['Country'], rotation = 45)
plt.xlabel('Country')
plt.ylabel('Number of People')
plt.title('Vaccinations by Country')
plt.show()
```



As one can see, this graph does not make a good visualisation due to the large differences in numbers.

In order to address the large differences in the population numbers, you can use segmented bar charts that show `Total Vaccinations` and `People Fully Vaccinated` as a proportion of the countries population.

In [40]:
```
percentfully = merged['People Fully Vaccinated']/(merged['Population']*1000)

barWidth = 0.5

plt.bar(merged['Country'], percentfully, width = barWidth, label = 'Fully Vaccinated
plt.bar(merged['Country'], 1 - percentfully, bottom = percentfully, width = barWidth

plt.xlabel('Country')
plt.ylabel('Proportion of Population')

plt.legend(loc='upper left', bbox_to_anchor=(1,1), ncol=1)
plt.title('Proportion of people fully vaccinated by population')

plt.show()
```



Question 3

To answer this question, we take the non-aggregated data from `Country-Vaccinations.csv` and read it into a new dataframe:

In [41]:
```
australia = pd.read_csv('data/Country-Vaccinations.csv')
australia.head()
```

Out[41]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN	
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN	

We subset the data for only Australia using `.loc` :

In [42]:
```
australia = australia.loc[australia['country'] == 'Australia']
australia
```

Out[42]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily
1746	Australia	AUS	2021-02-15	0.0	0.0	NaN	
1747	Australia	AUS	2021-02-16	0.0	0.0	NaN	
1748	Australia	AUS	2021-02-17	0.0	0.0	NaN	
1749	Australia	AUS	2021-02-18	0.0	0.0	NaN	
1750	Australia	AUS	2021-02-19	0.0	0.0	NaN	
...	...	...	...	...	...	...	
1917	Australia	AUS	2021-08-05	13030257.0	8775905.0	4254352.0	
1918	Australia	AUS	2021-08-06	13270296.0	8897750.0	4372546.0	
1919	Australia	AUS	2021-08-07	13496355.0	9006750.0	4489605.0	
1920	Australia	AUS	2021-08-08	13636580.0	9062787.0	4573793.0	
1921	Australia	AUS	2021-08-09	13723146.0	9108943.0	4614203.0	

176 rows × 15 columns

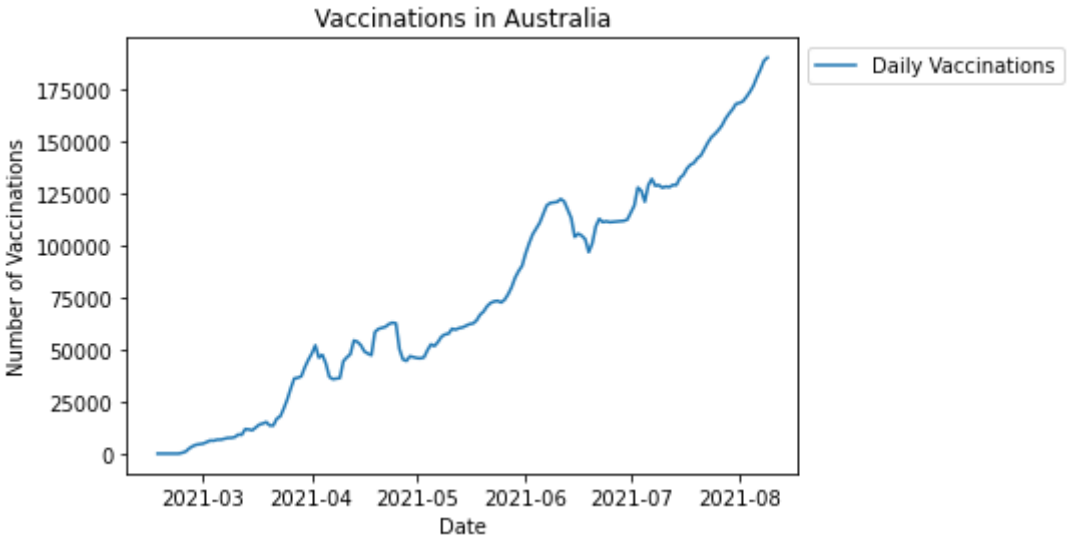
Using `.to_datatime()` , we are converting the `date` variable into datetime data type.

We are also using the `.cumsum()` command to create a new column for the cumulative sum of daily vaccinations over time.

In [43]:
```
australia['date'] = pd.to_datetime(australia['date'])
australia['cum_sum'] = australia['daily_vaccinations'].cumsum()
```

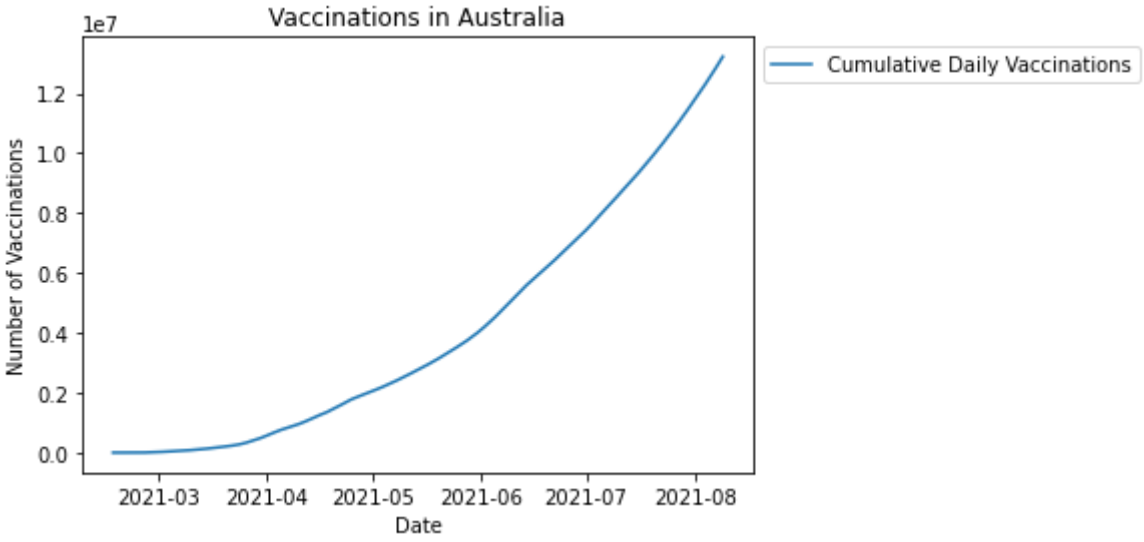
Using `.plot` from the `pyplot` library, we are plotting daily vaccinations in Australia over time:

In [44]:
```
plt.plot(australia['date'], australia['daily_vaccinations'], label = 'Daily Vaccinat

plt.xlabel('Date')
plt.ylabel('Number of Vaccinations')

plt.legend(loc='upper left', bbox_to_anchor=(1,1), ncol=1)
plt.title('Vaccinations in Australia')

plt.show()
```



Using `.plot` , we are plotting the cumulative daily vaccinations over time as well:

In [45]:
```
plt.plot(australia['date'], australia['cum_sum'], label = 'Cumulative Daily Vaccinat

plt.xlabel('Date')
plt.ylabel('Number of Vaccinations')

plt.legend(loc='upper left', bbox_to_anchor=(1,1), ncol=1)
plt.title('Vaccinations in Australia')

plt.show()
```



The first graph could be used when they want to see how daily vaccinations are changing over time.

The second graph would be used to display the total sum of daily vaccinations as it grows with time.

The prime difference between the graphs is that the first graph would be changing over time dropping up and down, whilst the second graph only grows over time. Therefore, it is a better representation of how quickly the data is changing over time.