FIT1043 Introduction to Data Science Assignment 1

Diong Chen Xi 32722656

15th August 2021

Introduction

This is the start of my assignment to investigate and visualize data using Python in the Jupyter Notebook environment.

1. Importing Libraries

First I start by importing the library **pandas**, which is an open source date analysis tool for the Python programming language. This library is used to access the data structure such as DataFrame and its functions to read files such as CSV, Excel, etc. Then, I import the library **matplotlib.pyplot** to use as a data visualizer later on in the assignment. I also included the magic line so that the plots created will be in line.

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

2. Reading datasets

After importing the library **pandas**, I now have access to the function which allows Python to read CSV files. After reading the specific file, I can store it as a data frame for future usage.

Country-Vaccinations.csv file

Here I use the read_csv function from pandas to read the csv file and store it inside the variable *country_vaccinations*.

In [2]:
```
country_vaccinations = pd.read_csv('Country-Vaccinations.csv')
country_vaccinations.head()
```

Out[2]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_
0	Afghanistan	AFG	2021-02-22	0.0	0.0	NaN	NaN	NaN	
1	Afghanistan	AFG	2021-02-23	NaN	NaN	NaN	NaN	1367.0	
2	Afghanistan	AFG	2021-02-24	NaN	NaN	NaN	NaN	1367.0	
3	Afghanistan	AFG	2021-02-25	NaN	NaN	NaN	NaN	1367.0	
4	Afghanistan	AFG	2021-02-26	NaN	NaN	NaN	NaN	1367.0	

The above table displays the first five rows of data in the **Country-Vaccinations.csv** file.

In [3]:
```
country_vaccinations.tail()
```

Out[3]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinatic
36731	Zimbabwe	ZWE	2021-08-04	2604265.0	1740598.0	863667.0	63710.0	46978.0	
36732	Zimbabwe	ZWE	2021-08-05	2701095.0	1780541.0	920554.0	96830.0	53866.0	

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinatic
36733	Zimbabwe	ZWE	2021-08-06	2784270.0	1817598.0	966672.0	83175.0	58416.0	
36734	Zimbabwe	ZWE	2021-08-07	2853668.0	1851407.0	1002261.0	69398.0	62880.0	
36735	Zimbabwe	ZWE	2021-08-08	2886822.0	1864204.0	1022618.0	33154.0	64783.0	

The above table displays the last five rows of data in the **Country-Vaccinations.csv** file.

In [4]:
```
country_vaccinations.sample(5)
```

Out[4]:

	country	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinatio
512	Algeria	DZA	2021-06-10	NaN	NaN	NaN	NaN	25444.0	
29876	Seychelles	SYC	2021-07-17	NaN	NaN	NaN	NaN	106.0	
14112	Guyana	GUY	2021-05-10	NaN	NaN	NaN	NaN	3915.0	
9906	El Salvador	SLV	2021-07-11	NaN	NaN	NaN	NaN	46082.0	
21580	Monaco	MCO	2021-02-16	NaN	NaN	NaN	NaN	294.0	

The above table displays five random rows of data in the **Country-Vaccinations.csv** file.

2020-GDP.csv file

Here I use the read_csv function from pandas to read the csv file and store it inside the variable *gdp*.

In [5]:
```
gdp = pd.read_csv('2020-GDP.csv')
gdp.head()
```

Out[5]:

	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

The above table displays the first five rows of data in the **2020-GDP.csv** file.

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

Out[6]:

	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

The above table displays the last five rows of data in the **2020-GDP.csv** file.

In [7]:
```
gdp.sample(5)
```

Out[7]:

	Unnamed: 0	Gross domestic product 2020	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5

	Unnamed: 0	Gross domestic product 2020	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5
78	AGO	75	NaN	Angola	62,307	NaN
5	CHN	2	NaN	China	14,722,731	NaN
248	NaN	NaN	NaN	NaN	NaN	NaN
281	NaN	NaN	NaN	NaN	NaN	NaN
127	PSE	124	NaN	West Bank and Gaza	15,561	NaN

The above table displays five random rows of data in the **2020-GDP.csv** file.

2020-Population.csv file

Here I use the read_csv function from pandas to read the csv file and store it inside the variable *population*.

In [8]:
```
population = pd.read_csv('2020-Population.csv')
population.head()
```

Out[8]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 68	Unnamed: 69	Unnamed 7(
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaI
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaI
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaI
3	United Nations	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaI
4	Population Division	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaI

5 rows × 78 columns

The above table displays the first five rows of data in the **2020-Population.csv** file.

In [9]:
```
population.tail()
```

Out[9]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 68	Unnamed: 69	Unna
300	285	Estimates	Bermuda	14	60	Country/Area	918	37	38	38	...	65	65	
301	286	Estimates	Canada	NaN	124	Country/Area	918	13 733	14 078	14 445	...	34 539	34 922	3
302	287	Estimates	Greenland	26	304	Country/Area	918	23	23	24	...	57	56	
303	288	Estimates	Saint Pierre and Miquelon	2	666	Country/Area	918	5	5	5	...	6	6	
304	289	Estimates	United States of America	35	840	Country/Area	918	158 804	160 872	163 266	...	311 584	314 044	31

5 rows × 78 columns

The above table displays the last five rows of data in the **2020-Population.csv** file.

In [10]:
```
population.sample(5)
```

Out[10]:

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 68	Unnamed: 69	Unna
59	44	Estimates	United Republic of Tanzania	3	834	Country/Area	910	7 650	7 845	8 051	...	45 674	47 053	4
201	186	Estimates	Panama	NaN	591	Country/Area	916	860	881	904	...	3 706	3 771	
252	237	Estimates	Poland	NaN	616	Country/Area	923	24 824	25 283	25 764	...	38 287	38 227	3
150	135	Estimates	Republic of Korea	NaN	410	Country/Area	906	19 211	19 453	19 818	...	49 786	50 061	5

	Unnamed: 0	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 68	Unnamed: 69	Unna
120	105	Estimates	Syrian Arab Republic	NaN	760	Country/Area	922	3 413	3 499	3 591	...	21 082	20 439	

5 rows × 78 columns

The above table displays five random rows of data in the **2020-Population.csv** file.

3. Wrangling the data

From here I create a function that obtains the sum of daily vaccinations and the sum of people vaccinated.

Country-Vaccinations.csv file

In [11]:
```
fun = {'daily_vaccinations':'sum','people_fully_vaccinated':'max'}
```

Then, I use the function to aggregate the number of total vaccinations and the number of fully vaccinated people of each country, all while keeping the type of vaccines in the same data frame.

In [12]:
```
vaccination = country_vaccinations.groupby(['country','vaccines']).agg(fun)
vaccination
```

Out[12]:

		daily_vaccinations	people_fully_vaccinated
country	**vaccines**		
Afghanistan	**Johnson&Johnson, Oxford/AstraZeneca, Pfizer/BioNTech, Sinopharm/Beijing**	1649463.0	219159.0
Albania	**Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, Sputnik V**	1237306.0	553482.0
Algeria	**Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac, Sputnik V**	4075025.0	724812.0
Andorra	**Oxford/AstraZeneca, Pfizer/BioNTech**	76689.0	33904.0

country	vaccines	daily_vaccinations	people_fully_vaccinated
Angola	**Oxford/AstraZeneca**	1690469.0	722610.0
...	**...**	...	...
Wales	**Moderna, Oxford/AstraZeneca, Pfizer/BioNTech**	4396339.0	2115477.0
Wallis and Futuna	**Oxford/AstraZeneca**	8937.0	4659.0
Yemen	**Oxford/AstraZeneca**	302943.0	13322.0
Zambia	**Oxford/AstraZeneca, Sinopharm/Beijing**	475566.0	193603.0
Zimbabwe	**Sinopharm/Beijing, Sinovac, Sputnik V**	2693298.0	1022618.0

222 rows × 2 columns

Since the columns are not properly indexed and named, I changed the name of the *daily_vaccinations* column to *total_vaccinations* and reassigned index to the data frame.

In [13]:
```
vaccination.rename(columns = {'daily_vaccinations':'total_vaccinations',},inplace = True)
vaccination = vaccination.reset_index()
vaccination
```

Out[13]:

	country	vaccines	total_vaccinations	people_fully_vaccinated
0	Afghanistan	Johnson&Johnson, Oxford/AstraZeneca, Pfizer/Bi...	1649463.0	219159.0
1	Albania	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac, ...	1237306.0	553482.0
2	Algeria	Oxford/AstraZeneca, Sinopharm/Beijing, Sinovac...	4075025.0	724812.0
3	Andorra	Oxford/AstraZeneca, Pfizer/BioNTech	76689.0	33904.0
4	Angola	Oxford/AstraZeneca	1690469.0	722610.0
...	...	...	...	...
217	Wales	Moderna, Oxford/AstraZeneca, Pfizer/BioNTech	4396339.0	2115477.0
218	Wallis and Futuna	Oxford/AstraZeneca	8937.0	4659.0

	country	vaccines	total_vaccinations	people_fully_vaccinated
219	Yemen	Oxford/AstraZeneca	302943.0	13322.0
220	Zambia	Oxford/AstraZeneca, Sinopharm/Beijing	475566.0	193603.0
221	Zimbabwe	Sinopharm/Beijing, Sinovac, Sputnik V	2693298.0	1022618.0

222 rows × 4 columns

2020-GDP.csv file

From the **2020-GDP.csv** file, I renamed the unnamed columns of the country names and the gdp values into *Country* and *GDP_value* respectively for easy access later on.

In [14]:
```
gdp.rename(columns={'Unnamed: 3':'country','Unnamed: 4':'GDP_value'},inplace=True)
gdp
```

Out[14]:

	Unnamed: 0	Gross domestic product 2020	Unnamed: 2	country	GDP_value	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
...	...	...	...	...	...	...
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

329 rows × 6 columns

From here I create a function that takes the sum of the GDP values.

In [15]:
```
fun2 = {'GDP_value':'sum'}
```

Then I use the function to aggregate the GDP values of each country.

In [16]:
```
GDP = gdp.groupby(['country']).agg(fun2)
GDP
```

Out[16]:

country	GDP_value
Afghanistan	19,807
Albania	14,800
Algeria	145,164
American Samoa	638
Andorra	3,155
...	...
West Bank and Gaza	15,561
World	84,705,567
Yemen, Rep.	23,486
Zambia	19,320
Zimbabwe	16,769

230 rows × 1 columns

Since the columns are not properly indexed, I reassigned index to the data frame.

In [17]:
```
GDP = GDP.reset_index()
GDP
```

Out[17]:

	country	GDP_value
0	Afghanistan	19,807
1	Albania	14,800
2	Algeria	145,164
3	American Samoa	638
4	Andorra	3,155
...	...	...
225	West Bank and Gaza	15,561
226	World	84,705,567
227	Yemen, Rep.	23,486
228	Zambia	19,320
229	Zimbabwe	16,769

230 rows × 2 columns

2020-Population.csv file

From the **2020-Populations.csv** file, I renamed the unnamed columns of the country names and the population of year 2020 into *Country* and *Population* respectively for easy access later on.

In [18]:
```
population.rename(columns= {'Unnamed: 2':'country','Unnamed: 77':'Population'},inplace=True)
population
```

Out[18]:

	Unnamed: 0	Unnamed: 1	country	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 68	Unnamed: 69	Unna
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

	Unnamed: 0	Unnamed: 1	country	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	...	Unnamed: 68	Unnamed: 69	Unna
3	United Nations	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
4	Population Division	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
300	285	Estimates	Bermuda	14	60	Country/Area	918	37	38	38	...	65	65	
301	286	Estimates	Canada	NaN	124	Country/Area	918	13 733	14 078	14 445	...	34 539	34 922	3
302	287	Estimates	Greenland	26	304	Country/Area	918	23	23	24	...	57	56	
303	288	Estimates	Saint Pierre and Miquelon	2	666	Country/Area	918	5	5	5	...	6	6	
304	289	Estimates	United States of America	35	840	Country/Area	918	158 804	160 872	163 266	...	311 584	314 044	31

305 rows × 78 columns

From here I create a function that takes the sum of the population in year 2020.

In [19]:
```
fun3 = {'Population':'sum'}
```

Then I use the function to aggregate the population in year 2020 of each country.

In [20]:
```
Population = population.groupby(['country']).agg(fun3)
Population
```

Out[20]:

	Population
country	

country	Population
AUSTRALIA/NEW ZEALAND	30 322
Afghanistan	38 928
Africa	1 340 598
Albania	2 878
Algeria	43 851
...	...
Western Sahara	597
World Bank income groups	...
Yemen	29 826
Zambia	18 384
Zimbabwe	14 863

290 rows × 1 columns

Since the columns are not properly indexed, I reassigned index to the data frame.

In [21]:
```
Population = Population.reset_index()
Population
```

Out[21]:

	country	Population
0	AUSTRALIA/NEW ZEALAND	30 322
1	Afghanistan	38 928
2	Africa	1 340 598
3	Albania	2 878
4	Algeria	43 851
...	...	...

	country	Population
285	Western Sahara	597
286	World Bank income groups	...
287	Yemen	29 826
288	Zambia	18 384
289	Zimbabwe	14 863

290 rows × 2 columns

4. Merging the datasets

As I only need to work with a subset of data that includes data for Indonesia, Malaysia, Singapore, Thailand, Philippines, and Australia, I created a data frame that consists of the list of the respective countries for easier access later on. It is to merge data sets more easily using the *pd.merge()* function.

In [22]:
```
Countries = pd.DataFrame({
    'country' : ['Indonesia', 'Malaysia', 'Singapore', 'Thailand', 'Philippines', 'Australia']
})
Countries
```

Out[22]:

	country
0	Indonesia
1	Malaysia
2	Singapore
3	Thailand
4	Philippines
5	Australia

Then, I merged the previous three datasets (ie. *vaccinations*, *GDP* and *Populations*) with the dataset *Countries*.

In [23]:
```
temp1 = pd.merge(vaccination, Countries, how ="inner", on=['country'])
temp1
```

Out[23]:

	country	vaccines	total_vaccinations	people_fully_vaccinated
0	Australia	Oxford/AstraZeneca, Pfizer/BioNTech	13222783.0	4614203.0
1	Indonesia	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	72386296.0	24481296.0
2	Malaysia	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	23687251.0	9048634.0
3	Philippines	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	23230492.0	11614590.0
4	Singapore	Moderna, Pfizer/BioNTech	7911869.0	3862510.0
5	Thailand	Oxford/AstraZeneca, Sinovac	18349011.0	4277071.0

In [24]:
```
temp2 = pd.merge(temp1, GDP, how="inner", on=['country'])
temp2
```

Out[24]:

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value
0	Australia	Oxford/AstraZeneca, Pfizer/BioNTech	13222783.0	4614203.0	1,330,901
1	Indonesia	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	72386296.0	24481296.0	1,058,424
2	Malaysia	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	23687251.0	9048634.0	336,664
3	Philippines	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	23230492.0	11614590.0	361,489
4	Singapore	Moderna, Pfizer/BioNTech	7911869.0	3862510.0	339,998
5	Thailand	Oxford/AstraZeneca, Sinovac	18349011.0	4277071.0	501,795

In [25]:
```
dataset = pd.merge(temp2, Population, how="inner", on=['country'])
dataset
```

Out[25]:

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value	Population
0	Australia	Oxford/AstraZeneca, Pfizer/BioNTech	13222783.0	4614203.0	1,330,901	25 500
1	Indonesia	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	72386296.0	24481296.0	1,058,424	273 524

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value	Population
2	Malaysia	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	23687251.0	9048634.0	336,664	32 366
3	Philippines	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	23230492.0	11614590.0	361,489	109 581
4	Singapore	Moderna, Pfizer/BioNTech	7911869.0	3862510.0	339,998	5 850
5	Thailand	Oxford/AstraZeneca, Sinovac	18349011.0	4277071.0	501,795	69 800

5. Managing data issues

As the population given is in thousands, I have to show the exact value inside the dataset. (e.g. the population of Australia is 25500 thousand, so it should be displayed as 25500000)
I used the str.split() function to get rid of the whitespace in between, and str.join() to join the numbers back into a string. Then, I used the .astype() function to convert the string into an integer for calculation.

In [26]:
```
dataset.Population = dataset.Population.str.split(' ').str.join('').astype(int)
```

In [27]:
```
dataset['Population'] = dataset['Population'] * 1000
dataset
```

Out[27]:

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value	Population
0	Australia	Oxford/AstraZeneca, Pfizer/BioNTech	13222783.0	4614203.0	1,330,901	25500000
1	Indonesia	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	72386296.0	24481296.0	1,058,424	273524000
2	Malaysia	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	23687251.0	9048634.0	336,664	32366000
3	Philippines	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	23230492.0	11614590.0	361,489	109581000
4	Singapore	Moderna, Pfizer/BioNTech	7911869.0	3862510.0	339,998	5850000
5	Thailand	Oxford/AstraZeneca, Sinovac	18349011.0	4277071.0	501,795	69800000

As the original data of GDP_value is not expressed in integers, I cannot directly use them for calculation to find *perCapitaGDP*.
Therefore I used str.split() to get rid of the unwanted characters in the number strings (i.e. the commas ",") and str.join() to join the numbers back into

a string. Then, I used the .astype() function to convert the string into an integer for calculation.
Also the GDP value given is in millions, I have to show the exact value inside the dataset. (e.g. the GDP value of Australia is 1330901 million, so it should be displayed as 1330901000000)
However, if I saved the data as an "int" type, it would throw an unexepcted result for the data being too big a number. Upon sourcing, I found out that saving the data as a "float" type it would resolve the issue.

In [28]:
```
dataset.GDP_value = dataset.GDP_value.str.split(',').str.join('').astype(float)
```

In [29]:
```
dataset['GDP_value'] = dataset['GDP_value'] * 1000000
dataset
```

Out[29]:

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value	Population
0	Australia	Oxford/AstraZeneca, Pfizer/BioNTech	13222783.0	4614203.0	1.330901e+12	25500000
1	Indonesia	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	72386296.0	24481296.0	1.058424e+12	273524000
2	Malaysia	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	23687251.0	9048634.0	3.366640e+11	32366000
3	Philippines	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	23230492.0	11614590.0	3.614890e+11	109581000
4	Singapore	Moderna, Pfizer/BioNTech	7911869.0	3862510.0	3.399980e+11	5850000
5	Thailand	Oxford/AstraZeneca, Sinovac	18349011.0	4277071.0	5.017950e+11	69800000

Also I would like to show all the data without the ".0" form, so I converted the data in *total_vaccinations* and *people_fully_vaccinated* into integers.

In [30]:
```
dataset['total_vaccinations'] = [int(i) for i in dataset['total_vaccinations']]
dataset['people_fully_vaccinated'] = [int(i) for i in dataset['people_fully_vaccinated']]
dataset
```

Out[30]:

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value	Population
0	Australia	Oxford/AstraZeneca, Pfizer/BioNTech	13222783	4614203	1.330901e+12	25500000
1	Indonesia	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	72386296	24481296	1.058424e+12	273524000
2	Malaysia	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	23687251	9048634	3.366640e+11	32366000
3	Philippines	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	23230492	11614590	3.614890e+11	109581000

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value	Population
4	Singapore	Moderna, Pfizer/BioNTech	7911869	3862510	3.399980e+11	5850000
5	Thailand	Oxford/AstraZeneca, Sinovac	18349011	4277071	5.017950e+11	69800000

6. Engineering the column "perCapitaGDP"

Here, per capita means "for each person", so perCapitaGDP means the GDP value for each person. Therefore I calculate the "perCapitaGDP" using the following formula:

$$perCapitaGDP = GDP\_value / Population$$

In [31]:
```
dataset['perCapitaGDP'] = dataset['GDP_value'] / dataset['Population']
dataset
```

Out[31]:

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value	Population	perCapitaGDP
0	Australia	Oxford/AstraZeneca, Pfizer/BioNTech	13222783	4614203	1.330901e+12	25500000	52192.196078
1	Indonesia	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	72386296	24481296	1.058424e+12	273524000	3869.583656
2	Malaysia	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	23687251	9048634	3.366640e+11	32366000	10401.779645
3	Philippines	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	23230492	11614590	3.614890e+11	109581000	3298.829177
4	Singapore	Moderna, Pfizer/BioNTech	7911869	3862510	3.399980e+11	5850000	58119.316239
5	Thailand	Oxford/AstraZeneca, Sinovac	18349011	4277071	5.017950e+11	69800000	7189.040115

The above is the finalized dataset, consisting of 6 rows (the countries) and 7 columns.

7. Statistical description

In [32]:
```
dataset
```

Out[32]:

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value	Population	perCapitaGDP

	country	vaccines	total_vaccinations	people_fully_vaccinated	GDP_value	Population	perCapitaGDP
0	Australia	Oxford/AstraZeneca, Pfizer/BioNTech	13222783	4614203	1.330901e+12	25500000	52192.196078
1	Indonesia	Moderna, Oxford/AstraZeneca, Sinopharm/Beijing...	72386296	24481296	1.058424e+12	273524000	3869.583656
2	Malaysia	Oxford/AstraZeneca, Pfizer/BioNTech, Sinovac	23687251	9048634	3.366640e+11	32366000	10401.779645
3	Philippines	Johnson&Johnson, Moderna, Oxford/AstraZeneca, ...	23230492	11614590	3.614890e+11	109581000	3298.829177
4	Singapore	Moderna, Pfizer/BioNTech	7911869	3862510	3.399980e+11	5850000	58119.316239
5	Thailand	Oxford/AstraZeneca, Sinovac	18349011	4277071	5.017950e+11	69800000	7189.040115

Interpretation of population and total vaccinations

From the dataset, we can see that Indonesia has the largest populaation, whereas Singapore has the smallest population. We can also see that Indonesia also has the highest amount of total vaccinations and Singapore has the smallest amount of total vaccinations. From that we can deduce that larger population countries have more vaccinations than smaller population countries.

Interpretation of vaccine types

From the dataset, we can see that the Oxford/AstraZeneca vaccine is the most popular type of vaccine used among the six countries, with a total of five countries using the specific vaccine. Also from the dataset we can see that the Sinopharm/Beijing vaccine and the Johnson&Johnson vaccine are the least popular type of vaccines used among the six countries, with only Indonesia and Philippines using respectively.

Interpretation of vaccinations

From the dataset, we can see that Singapore has the highest percentage of fully vaccinated people (the number of fully vaccinated people being the closest to the number of population) and the other countries have relatively low percentage of fully vaccinated people.

Interpretation of GDP values and perCapitaGDP

From the dataset, we can see that Australia has the highest GDP value, whereas Malaysia has the lowest GDP value overall. Also we can interpret from the dataset that Singapore has the highest perCapitaGDP while Philippines has the lowest perCapitaGDP.

Question 1

Firstly, I created a dataframe that includes the required countries, the types of vaccines, and the distribution of the types of vaccines in the countries. Assuming that in a certain country, each type of vaccine will be equally distributed to the country's population.

So, I calculated the number of the types of vaccines used in a country and added them to the dataframe, using the formula

$$number\_of\_vaccines = population / types\_of\_vaccines$$

If the country does not use a certain type of vaccine, I left the respective column as 0.

In [33]:
```
types_of_vaccines = pd.DataFrame({
    'vaccine':['Oxford/AstraZeneca','Pfizer/BioNTech','Moderna','Sinopharm/Beijing','Sinovac','Johnson&Johnson','Sputnik V'],
    'Australia':[dataset.iloc[0,5]//2, dataset.iloc[0,5]//2,0, 0, 0, 0, 0],
    'Indonesia':[dataset.iloc[1,5]//4,0,dataset.iloc[1,5]//4,dataset.iloc[1,5]//4,dataset.iloc[1,5]//4,0,0],
    'Malaysia':[dataset.iloc[2,5]//3,dataset.iloc[2,5]//3,0,0,dataset.iloc[2,5]//3,0,0],
    'Philippines':[dataset.iloc[3,5]//6,dataset.iloc[3,5]//6,dataset.iloc[3,5]//6,0,dataset.iloc[3,5]//6,dataset.iloc[3,5]//6,data
    'Singapore':[0,dataset.iloc[4,5]//2,dataset.iloc[4,5]//2,0,0,0,0],
    'Thailand':[dataset.iloc[5,5]//2,0,0,0,dataset.iloc[5,5]//2,0,0]
})
types_of_vaccines
```

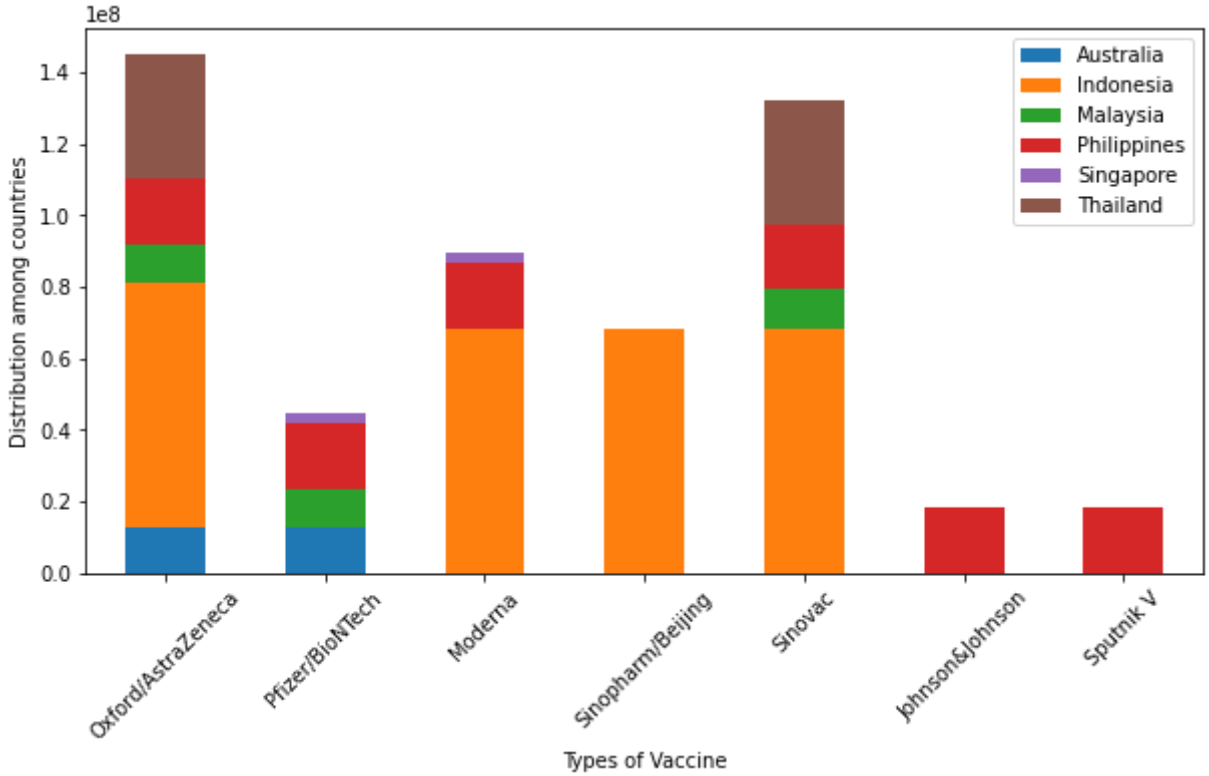
Out[33]:

	vaccine	Australia	Indonesia	Malaysia	Philippines	Singapore	Thailand
0	Oxford/AstraZeneca	12750000	68381000	10788666	18263500	0	34900000
1	Pfizer/BioNTech	12750000	0	10788666	18263500	2925000	0
2	Moderna	0	68381000	0	18263500	2925000	0
3	Sinopharm/Beijing	0	68381000	0	0	0	0
4	Sinovac	0	68381000	10788666	18263500	0	34900000
5	Johnson&Johnson	0	0	0	18263500	0	0
6	Sputnik V	0	0	0	18263500	0	0

Now that I have succesfully created the required dataframe, I can visualize the data using *matplotlib.pyplot*.
I decided to use the stacked bars graph to visualize the data, with the types of vaccines as the x-axis and the distribution among countries as the y-

axis. To me, it gives me a clear sense of which countries are a certain vaccine used in.

In [34]:
```
types_of_vaccines.plot.bar(figsize = (10,5), stacked = True)
plt.xticks((0,1,2,3,4,5,6),('Oxford/AstraZeneca','Pfizer/BioNTech','Moderna','Sinopharm/Beijing','Sinovac','Johnson&Johnson','Sput
plt.xlabel('Types of Vaccine')
plt.ylabel('Distribution among countries')
plt.show()
```



From the graph, we can analyse a lot of different things.
For example, we can see that the Philippines uses the most types of vaccines, while Thailand, Australia and Singapore uses the least types of vaccines.
We can also interpret that the most used type of vaccine is the Oxford/AstraZeneca vaccine, while the least used type of vaccine is the Johnson&Johnson vaccine and the Sputnik V vaccine.

Question 2

Firstly, I made a smaller dataframe from the original *dataset* dataframe to plot the required bar graph with appropriate values.

In [35]:
```
Q2 = pd.DataFrame({
    'country' : dataset['country'],
    'population' : dataset['Population'],
    'total_vaccinations' : dataset['total_vaccinations'],
    'people_fully_vaccinated' : dataset['people_fully_vaccinated']
})
Q2
```

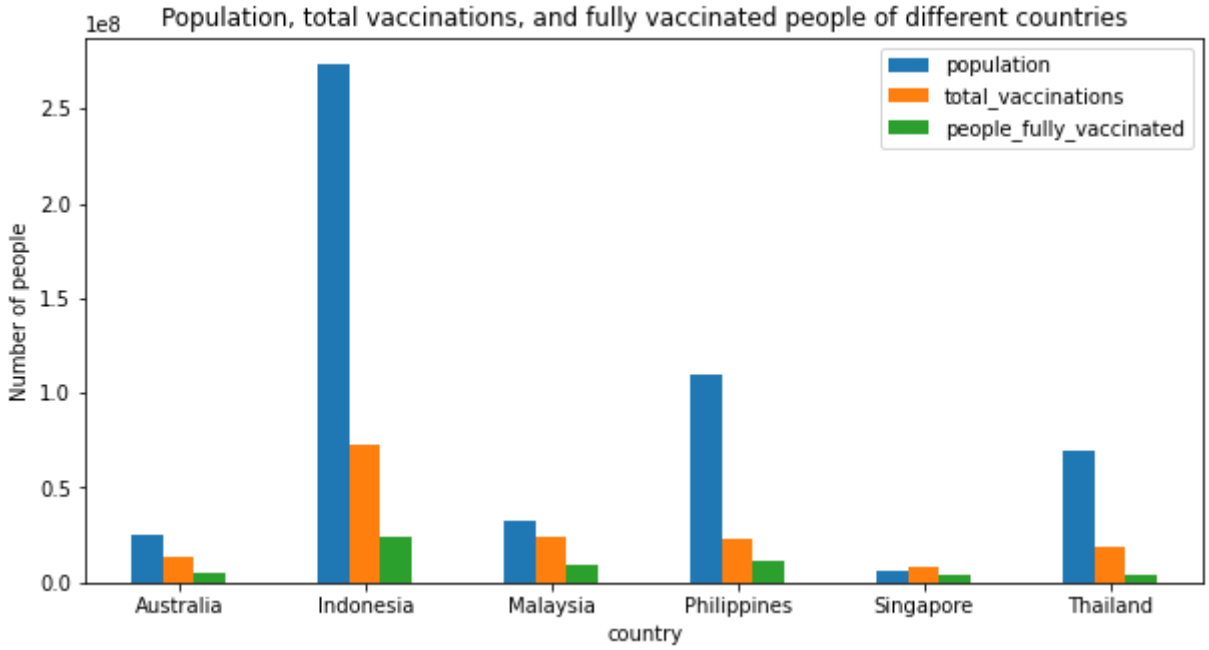
Out[35]:

	country	population	total_vaccinations	people_fully_vaccinated
0	Australia	25500000	13222783	4614203
1	Indonesia	273524000	72386296	24481296
2	Malaysia	32366000	23687251	9048634
3	Philippines	109581000	23230492	11614590
4	Singapore	5850000	7911869	3862510
5	Thailand	69800000	18349011	4277071

Then, I plotted a bar graph displaying side-by-side bars for population, total_vaccinations, and people_fully_vaccinated. The bar graph displays the countries as its x-axis, and the number of people as its y-axis.

In [36]:
```
Q2.plot.bar(figsize = (10,5))
plt.title('Population, total vaccinations, and fully vaccinated people of different countries')
plt.xticks((0,1,2,3,4,5),('Australia','Indonesia','Malaysia','Philippines','Singapore','Thailand'),rotation = 0)
plt.xlabel('country')
plt.ylabel('Number of people')
```

Out[36]:
```
Text(0, 0.5, 'Number of people')
```



However, we can see from the graph that it will be difficult to visualise due to large differences in the numbers.
Also purely using this data may not be accurate as we do not know how many people have not received their vaccination yet. This is because the graph only shows the number of total vaccinations and the number of people fully vaccinated, in which we cannot derive which of the vaccinations given are first dose or second dose and whether the vaccination given is to an unvaccinated person or not.

A way to resolve the overly large data difference is to obtain the logarithmic values of each of the columns of data.
First I import the **numpy** library to access the logarithm function for dataframes.

In [37]:
```
import numpy as np
```

Then I calculate the logarithmic values for the *population*, *total_vaccinations* and *people_fully_vaccinated* and updated the data values inside the Q2 dataframe.

In [38]:
```
Q2['population'] = np.log(Q2['population'])
Q2['total_vaccinations'] = np.log(Q2['total_vaccinations'])
Q2['people_fully_vaccinated'] = np.log(Q2['people_fully_vaccinated'])
Q2
```

Out[38]:

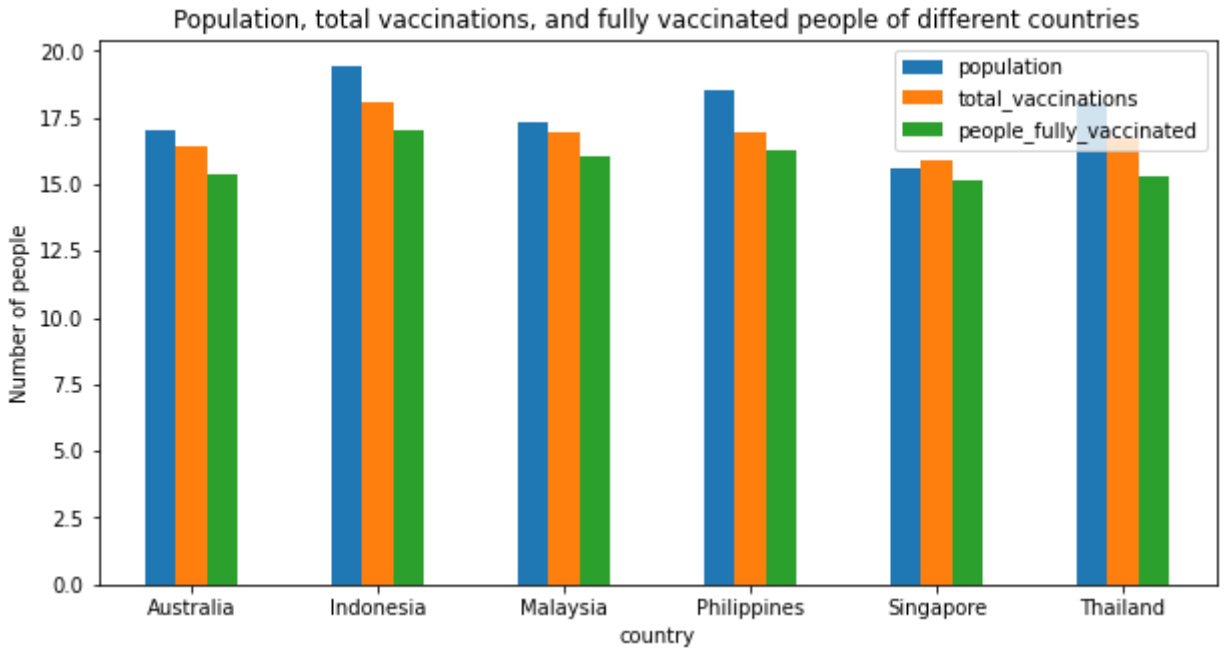
	country	population	total_vaccinations	people_fully_vaccinated
0	Australia	17.054189	16.397452	15.344650
1	Indonesia	19.426900	18.097528	17.013420
2	Malaysia	17.292619	16.980448	16.018124
3	Philippines	18.512175	16.960976	16.267773
4	Singapore	15.581952	15.883875	15.166828
5	Thailand	18.061145	16.725086	15.268779

In [39]:

```
Q2.plot.bar(figsize = (10,5))
plt.title('Population, total vaccinations, and fully vaccinated people of different countries')
plt.xticks((0,1,2,3,4,5),('Australia','Indonesia','Malaysia','Philippines','Singapore','Thailand'),rotation = 0)
plt.xlabel('country')
plt.ylabel('Number of people')
```

Out[39]:

```
Text(0, 0.5, 'Number of people')
```



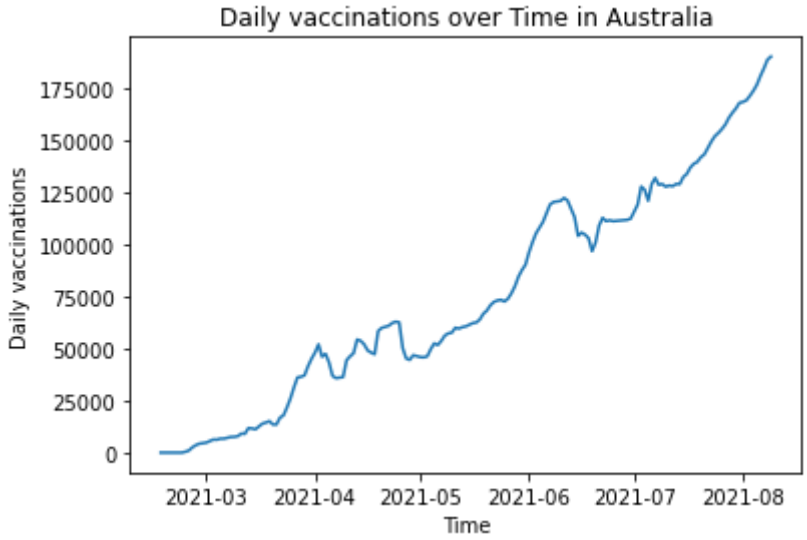
Now that the scale of the graph is shrunk, the data can be better visualised.

Question 3

First line graph

Firstly, I selected the data related to Australia, as it is what I want to work with. Then, I plotted a line graph of daily_vaccinations over time with time as the x-axis and the number of daily vaccinations as the y-axis.
As the data in the *dates* column is too large, I converted the data type into *datetime* format in order to not yield a cramped x-axis.

In [40]:
```
country_vaccinations['date'] = pd.to_datetime(country_vaccinations['date'])
aus_vaccinations = country_vaccinations[country_vaccinations['country'] == 'Australia']
x1 = aus_vaccinations.date
y1 = aus_vaccinations.daily_vaccinations
plt.plot(x1,y1)
plt.xlabel('Time')
plt.ylabel('Daily vaccinations')
plt.title('Daily vaccinations over Time in Australia')
plt.show()
```



In my opinion, I think that this graph would be useful to see how many vaccinations were given each day to get a rough idea of the rate of vaccinations given for a certain country. In this case, we can see that there is a climb in the overall rate of vaccinations for Australia, with some

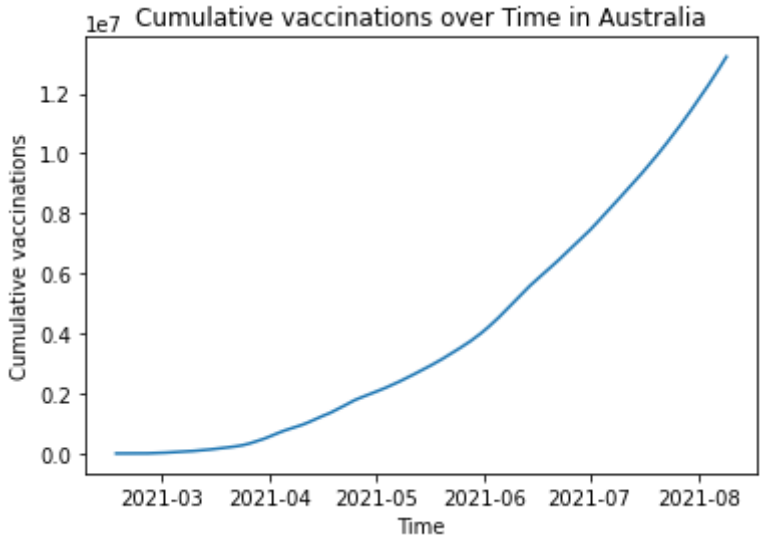
decrement in between.

Second line graph

Then, I plotted a line graph to visualise the cumulative vaccinations over time with time as the x-axis and the cumulative number of vaccinations as the y-axis. Here I made use of the .cumsum() function to calculate the cumulative sum of the daily vaccinations.

In [41]:
```
x2 = aus_vaccinations.date
y2 = aus_vaccinations.daily_vaccinations.cumsum()
graph2 = plt.plot(x2,y2)
plt.xlabel('Time')
plt.ylabel('Cumulative vaccinations')
plt.title('Cumulative vaccinations over Time in Australia')

plt.show()
```



In my opinion, I think that this graph is useful to derive the trend of vaccinations given for a certain country. From the gradient of the line, we can interpret that a larger gradient indicates a larger number of daily vaccinations given by the country , whereas a smaller gradient indicates a smaller number of daily vaccinations given by the country. If the slope is parallel to the x-axis, it means that no vaccinations were given for that period.

Also from the concavity of the graph serves as an indicator of whether the daily vaccination rate is increasing or decreasing. If a certain section of the graph has a concave upward trend, we say that the daily vaccination rate is increasing in that period of time. Otherwise, if a certain section of the

grave has a concave downward trend, we say that the daily vaccination rate is decreasing in that period of time. A linear increase indicates that the daily vaccination rate stays the same in that period of time.