Unit 2

1. Introducing Pandas

Panda's documentation

To begin we need to import pandas When you see pd, know it is referring to pandas

In [44]:
```
import numpy as np
import pandas as pd
```

Pandas is a popular Python library used for working in tabular data (similar to the data stored in a spreadsheet).

There are two main data structure used by pandas

- Series: equivalent to a vector or a list
- Dataframe: equivalent to a table.

Each column in a pandas Dataframe is a pandas Series data structure. We will mainly be looking at the Dataframe.

We can easily create a Pandas Dataframe by reading a .csv file

2. Reading files

We will read the whole file at once using Pandas. Sometimes you might want to read the file line by line, and process each line. Thats possible of course. See for example here.

We will read data on COVID-19 vaccinations

In order to do that, I retrieved the raw data's url

Press on raw either here:

or here:

and retrieve the link:

In [2]:
```
url = 'https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/vaccinations/vaccinations.csv'
vacc_df = pd.read_csv(url)
```

read_csv has about 30 different options. See the documentation

For example, sep='\t' is used for tab delimited files and 'usecol' reads only specific columns.

In [3]:
```
type(vacc_df)
```

Out[3]:
```
pandas.core.frame.DataFrame
```

view the shape of the dataframe:

In [47]:
```
vacc_df.shape
```

Out[47]:
```
(35163, 12)
```

view basic information:

In [5]:
```
vacc_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 35163 entries, 0 to 35162
Data columns (total 12 columns):
 #   Column                               Non-Null Count  Dtype  
---  ------                               --------------  -----  
 0   location                             35163 non-null  object 
 1   iso_code                             35163 non-null  object 
 2   date                                 35163 non-null  object 
 3   total_vaccinations                   20578 non-null  float64
 4   people_vaccinated                    19765 non-null  float64
 5   people_fully_vaccinated              16719 non-null  float64
 6   daily_vaccinations_raw               17387 non-null  float64
 7   daily_vaccinations                   34902 non-null  float64
 8   total_vaccinations_per_hundred       20578 non-null  float64
 9   people_vaccinated_per_hundred        19765 non-null  float64
 10  people_fully_vaccinated_per_hundred  16719 non-null  float64
 11  daily_vaccinations_per_million       34902 non-null  float64
dtypes: float64(9), object(3)
memory usage: 3.2+ MB
```

In [6]:
```
vacc_df.columns
```

Out[6]:
```
Index(['location', 'iso_code', 'date', 'total_vaccinations',
       'people_vaccinated', 'people_fully_vaccinated',
       'daily_vaccinations_raw', 'daily_vaccinations',
       'total_vaccinations_per_hundred', 'people_vaccinated_per_hundred',
       'people_fully_vaccinated_per_hundred',
       'daily_vaccinations_per_million'],
      dtype='object')
```

View the first few rows:

In [51]:
```
vacc_df.head()
```

Out[51]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_va
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	

What do you think that the 'tail' command does? Try it out!

What happens if we just type vacc_df, without a head or a tail?

In [53]:
```
vacc_df.tail()
```

Out[53]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	tota
35158	Zimbabwe	ZWE	2021-07-17	1771434.0	1132045.0	639389.0	39694.0	41958.0	
35159	Zimbabwe	ZWE	2021-07-18	1785533.0	1144379.0	641154.0	14099.0	42019.0	
35160	Zimbabwe	ZWE	2021-07-19	1827638.0	1184435.0	643203.0	42105.0	42253.0	
35161	Zimbabwe	ZWE	2021-07-20	1897337.0	1247494.0	649843.0	69699.0	45971.0	
35162	Zimbabwe	ZWE	2021-07-21	1949472.0	1292642.0	656830.0	52135.0	47976.0	

In [9]:
```
vacc_df.describe()
```

Out[9]:

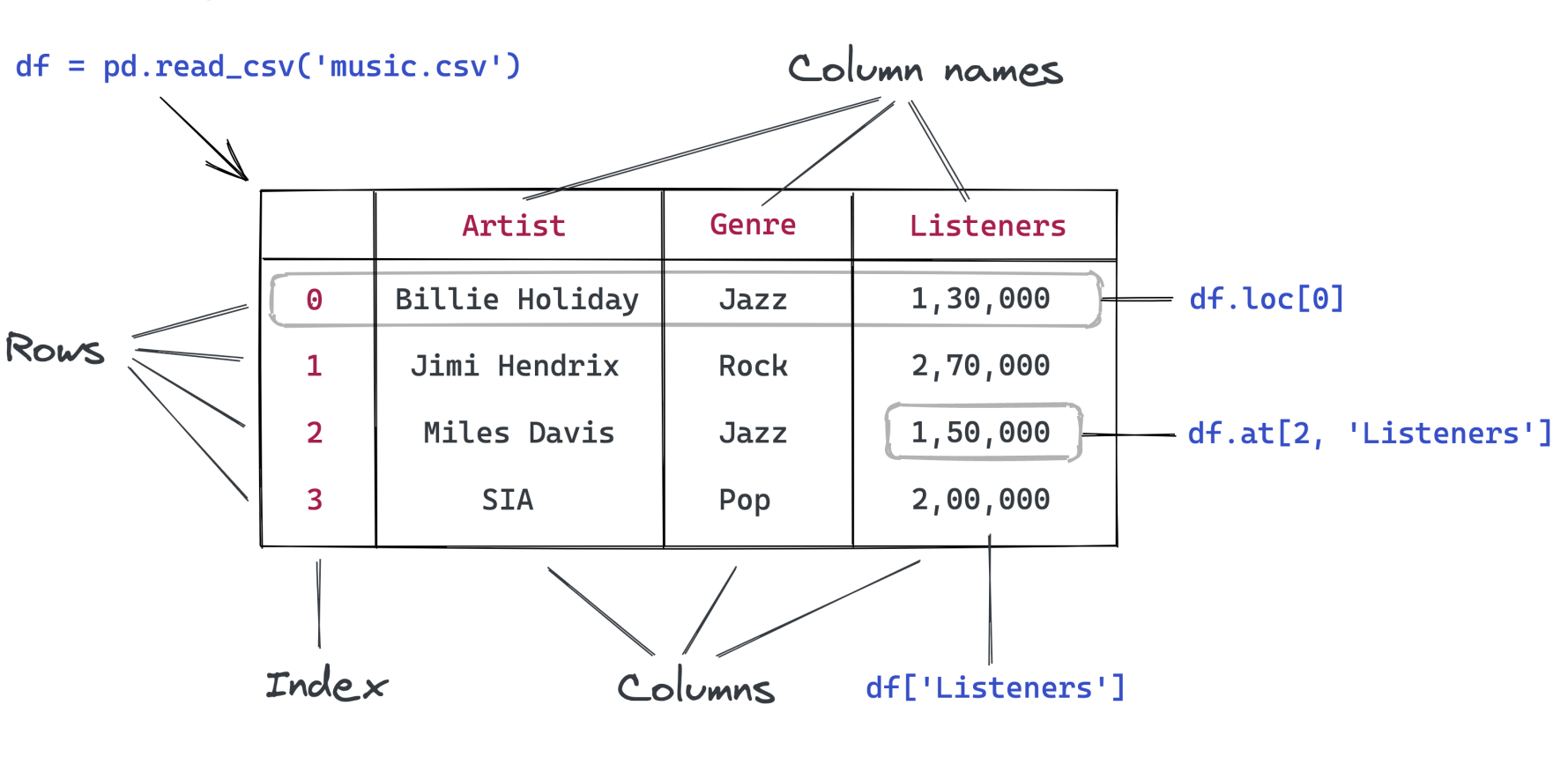
	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_vaccinations_per_hundred
count	2.057800e+04	1.976500e+04	1.671900e+04	1.738700e+04	3.490200e+04	20578.000000
mean	4.834277e+07	2.683580e+07	1.486042e+07	8.780545e+05	4.383342e+05	29.165589
std	2.276768e+08	1.267271e+08	6.475550e+07	3.552802e+06	2.474233e+06	35.025427
min	0.000000e+00	0.000000e+00	1.000000e+00	0.000000e+00	0.000000e+00	0.000000
25%	1.662212e+05	1.402700e+05	6.086300e+04	5.764000e+03	9.600000e+02	2.780000
50%	1.300552e+06	9.153020e+05	5.243060e+05	3.050300e+04	8.131000e+03	14.330000
75%	8.138582e+06	5.149993e+06	3.245194e+06	1.880345e+05	5.346750e+04	44.475000
max	3.787027e+09	2.092904e+09	1.048364e+09	4.794441e+07	4.329428e+07	232.650000

A summary of the functions so far:

- `pd.read_csv` - Read data from a CSV file into a Pandas `DataFrame` object
- `.info()` - View basic infomation about rows, columns & data types
- `.describe()` - View statistical information about numeric columns
- `.columns` - Get the list of column names
- `.shape` - Get the number of rows & columns as a tuple
- `.head()` , `.tail()` - View the beginning/end of the file

---

3. Selecting data



Pandas format is similar to a dictionary, not to a list

a list:

In [10]:
```
covid_data_list = [
    {'date': '2020-08-30', 'new_cases': 1444, 'new_deaths': 1, 'new_tests': 53541},
    {'date': '2020-08-31', 'new_cases': 1365, 'new_deaths': 4, 'new_tests': 42583},
    {'date': '2020-09-01', 'new_cases': 996, 'new_deaths': 6, 'new_tests': 54395},
    {'date': '2020-09-02', 'new_cases': 975, 'new_deaths': 8 },
    {'date': '2020-09-03', 'new_cases': 1326, 'new_deaths': 6},
]
covid_data_list
```

Out[10]:
```
[{'date': '2020-08-30',
  'new_cases': 1444,
  'new_deaths': 1,
  'new_tests': 53541},
 {'date': '2020-08-31',
  'new_cases': 1365,
  'new_deaths': 4,
  'new_tests': 42583},
 {'date': '2020-09-01', 'new_cases': 996, 'new_deaths': 6, 'new_tests': 54395},
 {'date': '2020-09-02', 'new_cases': 975, 'new_deaths': 8},
 {'date': '2020-09-03', 'new_cases': 1326, 'new_deaths': 6}]
```

a dictionary:

In [11]:
```
covid_data_dict = {
    'date':       ['2020-08-30', '2020-08-31', '2020-09-01', '2020-09-02', '2020-09-03'],
    'new_cases':  [1444, 1365, 996, 975, 1326],
    'new_deaths': [1, 4, 6, 8, 6],
    'new_tests': [53541, 42583, 54395, None, None]
}
```

The index of a dataframe doesn't have to be numeric

In [54]:
```
df = pd.DataFrame({'age':[30, 2, 12, 4, 32, 33, 69],
                   'color':['blue', 'green', 'red', 'white', 'gray', 'black', 'red'],
                   'food':['Steak', 'Lamb', 'Mango', 'Apple', 'Cheese', 'Melon', 'Beans'],
                   'height':[165, 70, 120, 80, 180, 172, 150],
                   'score':[4.6, 8.3, 9.0, 3.3, 1.8, 9.5, 2.2],
                   'state':['NY', 'TX', 'FL', 'AL', 'AK', 'TX', 'TX']
                   },
                  index=['Jane', 'Nick', 'Aaron', 'Penelope', 'Dean', 'Christina', 'Cornelia'])
df
```

Out[54]:

	age	color	food	height	score	state
Jane	30	blue	Steak	165	4.6	NY
Nick	2	green	Lamb	70	8.3	TX
Aaron	12	red	Mango	120	9.0	FL
Penelope	4	white	Apple	80	3.3	AL
Dean	32	gray	Cheese	180	1.8	AK
Christina	33	black	Melon	172	9.5	TX
Cornelia	69	red	Beans	150	2.2	TX

In our our file the index is numeric:

In [13]:
```
vacc_df.head()
```

Out[13]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_va
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	

return a single column as a series:

note: using the `.` notation is possible only for columns whose names do not contain spaces or special characters.

In [14]:
```
vacc_df.location
vacc_df['location']
```

Out[14]:
```
0        Afghanistan
1        Afghanistan
2        Afghanistan
3        Afghanistan
4        Afghanistan
            ...     
35158       Zimbabwe
35159       Zimbabwe
35160       Zimbabwe
35161       Zimbabwe
35162       Zimbabwe
Name: location, Length: 35163, dtype: object
```

return a single column as a dataframe:

In [15]:
```
type(vacc_df.location)
```

Out[15]:
```
pandas.core.series.Series
```

In [56]:
```
vacc_df[['location']]
```

Out[56]:

	location	date
0	Afghanistan	2021-02-22
1	Afghanistan	2021-02-23
2	Afghanistan	2021-02-24
3	Afghanistan	2021-02-25
4	Afghanistan	2021-02-26
...	...	...
35158	Zimbabwe	2021-07-17
35159	Zimbabwe	2021-07-18
35160	Zimbabwe	2021-07-19
35161	Zimbabwe	2021-07-20
35162	Zimbabwe	2021-07-21

35163 rows × 2 columns

retrieve a specific cell

In [58]:
```
vacc_df.location[600]
```

Out[58]:
```
'Algeria'
```

retrieve two columns

In [18]:
```
vacc_df[['location','date']]
```

Out[18]:

	location	date
0	Afghanistan	2021-02-22
1	Afghanistan	2021-02-23
2	Afghanistan	2021-02-24
3	Afghanistan	2021-02-25
4	Afghanistan	2021-02-26
...	...	...
35158	Zimbabwe	2021-07-17
35159	Zimbabwe	2021-07-18
35160	Zimbabwe	2021-07-19
35161	Zimbabwe	2021-07-20
35162	Zimbabwe	2021-07-21

35163 rows × 2 columns

Seletcting subsets of rows and columns

One way to do that is iloc.

`.iloc` - selects subsets of rows and columns by integer location only

In [69]:
```
#vacc_df.iloc[0]  #first row
vacc_df.iloc[-1] #last row as a series
vacc_df.iloc[3:5] #last row
```

Out[69]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_va
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 : operator

- when used alone it means "everything"
- also used to indicate a ***slice*** of values

In [20]:
```
vacc_df.iloc[1:2] # second and third row
vacc_df.iloc[[-1,2,22]] #a few specific rows

# Columns:
vacc_df.iloc[:,0] # first column of data frame
vacc_df.iloc[:,1] # second column of data frame
vacc_df.iloc[:,-1] # last column of data frame

#Rows and columns
vacc_df.iloc[0:5] # first five rows of dataframe
vacc_df.iloc[:, 0:2] # first two columns of data frame with all rows
vacc_df.iloc[[0,3,6,24], [0,5,6]] # 1st, 4th, 7th, 25th row + 1st 6th 7th columns.
```

Out[20]:

	location	people_fully_vaccinated	daily_vaccinations_raw
0	Afghanistan	NaN	NaN
3	Afghanistan	NaN	NaN
6	Afghanistan	NaN	NaN
24	Afghanistan	NaN	NaN

In [21]:
```
vacc_df.iloc[0:19]
```

Out[21]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_
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	
5	Afghanistan	AFG	2021-02-27	NaN	NaN	NaN	NaN	1367.0	
6	Afghanistan	AFG	2021-02-28	8200.0	8200.0	NaN	NaN	1367.0	
7	Afghanistan	AFG	2021-03-01	NaN	NaN	NaN	NaN	1580.0	
8	Afghanistan	AFG	2021-03-02	NaN	NaN	NaN	NaN	1794.0	
9	Afghanistan	AFG	2021-03-03	NaN	NaN	NaN	NaN	2008.0	
10	Afghanistan	AFG	2021-03-04	NaN	NaN	NaN	NaN	2221.0	
11	Afghanistan	AFG	2021-03-05	NaN	NaN	NaN	NaN	2435.0	
12	Afghanistan	AFG	2021-03-06	NaN	NaN	NaN	NaN	2649.0	
13	Afghanistan	AFG	2021-03-07	NaN	NaN	NaN	NaN	2862.0	
14	Afghanistan	AFG	2021-03-08	NaN	NaN	NaN	NaN	2862.0	
15	Afghanistan	AFG	2021-03-09	NaN	NaN	NaN	NaN	2862.0	
16	Afghanistan	AFG	2021-03-10	NaN	NaN	NaN	NaN	2862.0	
17	Afghanistan	AFG	2021-03-11	NaN	NaN	NaN	NaN	2862.0	
18	Afghanistan	AFG	2021-03-12	NaN	NaN	NaN	NaN	2862.0	

What if I want to select the 'daily_vaccinations' column, but I don't remember which column it is?

Use `.loc`

`.loc` - selects subsets of rows and columns by label only. Allowed inputs are:

- A single label, e.g. 5 or 'a', (note that 5 is interpreted as a label of the index, and never as an integer position along the index).
- A list or array of labels, e.g. ['a', 'b', 'c'].
- A slice object with labels, e.g. 'a':'f'.

In [75]:
```
vacc_df.loc[2:3,['daily_vaccinations','date']]
```

Out[75]:

	daily_vaccinations	date
2	1367.0	2021-02-24
3	1367.0	2021-02-25

I'm missing the location. Let's add it.

In [23]:
```
vacc_df.loc[0:3,['location','daily_vaccinations']]
```

Out[23]:

	location	daily_vaccinations
0	Afghanistan	NaN
1	Afghanistan	1367.0
2	Afghanistan	1367.0
3	Afghanistan	1367.0

Semantics are similar to iloc. But note:

- `iloc` excludes the last element. `df.iloc[0:1000]` will return entries 0...999
- `loc` , includes the last element. `df.loc[0:1000]` will return entries 0...1000

you try it! What is the difference between:

> vacc_df.iloc[0:5]
>
> vacc_df.loc[0:5]

In [24]:
```
vacc_df.iloc[0:5]
```

Out[24]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_va
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	

In [25]:
```
vacc_df.loc[0:5]
```

Out[25]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_va
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	
5	Afghanistan	AFG	2021-02-27	NaN	NaN	NaN	NaN	1367.0	

Now you:

what do you do to select:

a. first five rows?

b. first two columns, all rows?

c. 1st and 3rd row and 2nd and 4th column?

---

a. the first five rows are:

In [78]:
```
vacc_df.head()
```

Out[78]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_va
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	
5	Afghanistan	AFG	2021-02-27	NaN	NaN	NaN	NaN	1367.0	

In [80]:
```
vacc_df.iloc [:,0:2]
```

Out[80]:

	location	iso_code
0	Afghanistan	AFG
1	Afghanistan	AFG
2	Afghanistan	AFG
3	Afghanistan	AFG
4	Afghanistan	AFG
...	...	...
35158	Zimbabwe	ZWE
35159	Zimbabwe	ZWE
35160	Zimbabwe	ZWE
35161	Zimbabwe	ZWE
35162	Zimbabwe	ZWE

35163 rows × 2 columns

In [28]:
```
vacc_df.iloc[[1,3],[2,4]]
```

Out[28]:

	date	people_vaccinated
1	2021-02-23	NaN
3	2021-02-25	NaN

In [ ]:
```

```

A summary of the functions in this unit:

- `.iloc` - selects rows and columns by integer location
- `.loc` - selects rows and columns by label location

Note: indexing operators as the ones working on dictionaries, will also work in pandas. But for more advanced operations, better get used to loc and iloc.

---

4. Conditional selection

In [29]:
```
vacc_df.loc[:,'location'] == 'Israel'
```

Out[29]:
```
0        False
1        False
2        False
3        False
4        False
         ...  
35158    False
35159    False
35160    False
35161    False
35162    False
Name: location, Length: 35163, dtype: bool
```

This creates a series of true/false

We can insert this into the dataframe to select only that task:

In [30]:
```
vacc_df[vacc_df.loc[:,'location'] == 'Israel']
```

Out[30]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_
15458	Israel	ISR	2020-12-19	60.0	60.0	NaN	NaN	NaN	
15459	Israel	ISR	2020-12-20	7434.0	7434.0	NaN	7374.0	7374.0	
15460	Israel	ISR	2020-12-21	32318.0	32318.0	NaN	24884.0	16129.0	
15461	Israel	ISR	2020-12-22	76933.0	76933.0	NaN	44615.0	25624.0	
15462	Israel	ISR	2020-12-23	139765.0	139765.0	NaN	62832.0	34926.0	
...	...	...	...	...	...	...	...	...	
15669	Israel	ISR	2021-07-18	10980188.0	5744452.0	5235736.0	11665.0	7985.0	
15670	Israel	ISR	2021-07-19	10994393.0	5747537.0	5246856.0	14205.0	9183.0	
15671	Israel	ISR	2021-07-20	11008624.0	5750067.0	5258557.0	14231.0	10060.0	
15672	Israel	ISR	2021-07-21	11022474.0	5752297.0	5270177.0	13850.0	10620.0	
15673	Israel	ISR	2021-07-22	11038265.0	5755067.0	5283198.0	15791.0	11172.0	

216 rows × 12 columns

Another way:

In [31]:
```
vacc_df.loc[vacc_df.location == 'Israel']
```

Out[31]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_
15458	Israel	ISR	2020-12-19	60.0	60.0	NaN	NaN	NaN	
15459	Israel	ISR	2020-12-20	7434.0	7434.0	NaN	7374.0	7374.0	
15460	Israel	ISR	2020-12-21	32318.0	32318.0	NaN	24884.0	16129.0	
15461	Israel	ISR	2020-12-22	76933.0	76933.0	NaN	44615.0	25624.0	
15462	Israel	ISR	2020-12-23	139765.0	139765.0	NaN	62832.0	34926.0	
...	...	...	...	...	...	...	...	...	
15669	Israel	ISR	2021-07-18	10980188.0	5744452.0	5235736.0	11665.0	7985.0	
15670	Israel	ISR	2021-07-19	10994393.0	5747537.0	5246856.0	14205.0	9183.0	
15671	Israel	ISR	2021-07-20	11008624.0	5750067.0	5258557.0	14231.0	10060.0	
15672	Israel	ISR	2021-07-21	11022474.0	5752297.0	5270177.0	13850.0	10620.0	
15673	Israel	ISR	2021-07-22	11038265.0	5755067.0	5283198.0	15791.0	11172.0	

216 rows × 12 columns

Select two countries:

In [82]:
```
two_countries = vacc_df.loc[(vacc_df.location == 'Israel') | (vacc_df.location == 'Denmark')]
two_countries.head()
```

Out[82]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_v
8047	Denmark	DNK	2020-12-17	1.0	1.0	NaN	NaN	NaN	
8048	Denmark	DNK	2020-12-18	2.0	2.0	NaN	1.0	1.0	
8049	Denmark	DNK	2020-12-19	3.0	3.0	NaN	1.0	1.0	
8050	Denmark	DNK	2020-12-20	NaN	NaN	NaN	NaN	1.0	
8051	Denmark	DNK	2020-12-21	4.0	4.0	NaN	NaN	1.0	

only the indexs of the tasks:

In [33]:
```
two_countries.index.values
```

Out[33]:
```
array([ 8047,  8048,  8049,  8050,  8051,  8052,  8053,  8054,  8055,
        8056,  8057,  8058,  8059,  8060,  8061,  8062,  8063,  8064,
        8065,  8066,  8067,  8068,  8069,  8070,  8071,  8072,  8073,
        8074,  8075,  8076,  8077,  8078,  8079,  8080,  8081,  8082,
        8083,  8084,  8085,  8086,  8087,  8088,  8089,  8090,  8091,
        8092,  8093,  8094,  8095,  8096,  8097,  8098,  8099,  8100,
        8101,  8102,  8103,  8104,  8105,  8106,  8107,  8108,  8109,
        8110,  8111,  8112,  8113,  8114,  8115,  8116,  8117,  8118,
        8119,  8120,  8121,  8122,  8123,  8124,  8125,  8126,  8127,
        8128,  8129,  8130,  8131,  8132,  8133,  8134,  8135,  8136,
        8137,  8138,  8139,  8140,  8141,  8142,  8143,  8144,  8145,
        8146,  8147,  8148,  8149,  8150,  8151,  8152,  8153,  8154,
        8155,  8156,  8157,  8158,  8159,  8160,  8161,  8162,  8163,
        8164,  8165,  8166,  8167,  8168,  8169,  8170,  8171,  8172,
        8173,  8174,  8175,  8176,  8177,  8178,  8179,  8180,  8181,
        8182,  8183,  8184,  8185,  8186,  8187,  8188,  8189,  8190,
        8191,  8192,  8193,  8194,  8195,  8196,  8197,  8198,  8199,
        8200,  8201,  8202,  8203,  8204,  8205,  8206,  8207,  8208,
        8209,  8210,  8211,  8212,  8213,  8214,  8215,  8216,  8217,
        8218,  8219,  8220,  8221,  8222,  8223,  8224,  8225,  8226,
        8227,  8228,  8229,  8230,  8231,  8232,  8233,  8234,  8235,
        8236,  8237,  8238,  8239,  8240,  8241,  8242,  8243,  8244,
        8245,  8246,  8247,  8248,  8249,  8250,  8251,  8252,  8253,
        8254,  8255,  8256,  8257,  8258,  8259,  8260,  8261,  8262,
        8263, 15458, 15459, 15460, 15461, 15462, 15463, 15464, 15465,
       15466, 15467, 15468, 15469, 15470, 15471, 15472, 15473, 15474,
       15475, 15476, 15477, 15478, 15479, 15480, 15481, 15482, 15483,
       15484, 15485, 15486, 15487, 15488, 15489, 15490, 15491, 15492,
       15493, 15494, 15495, 15496, 15497, 15498, 15499, 15500, 15501,
       15502, 15503, 15504, 15505, 15506, 15507, 15508, 15509, 15510,
       15511, 15512, 15513, 15514, 15515, 15516, 15517, 15518, 15519,
       15520, 15521, 15522, 15523, 15524, 15525, 15526, 15527, 15528,
       15529, 15530, 15531, 15532, 15533, 15534, 15535, 15536, 15537,
       15538, 15539, 15540, 15541, 15542, 15543, 15544, 15545, 15546,
       15547, 15548, 15549, 15550, 15551, 15552, 15553, 15554, 15555,
       15556, 15557, 15558, 15559, 15560, 15561, 15562, 15563, 15564,
       15565, 15566, 15567, 15568, 15569, 15570, 15571, 15572, 15573,
       15574, 15575, 15576, 15577, 15578, 15579, 15580, 15581, 15582,
       15583, 15584, 15585, 15586, 15587, 15588, 15589, 15590, 15591,
       15592, 15593, 15594, 15595, 15596, 15597, 15598, 15599, 15600,
       15601, 15602, 15603, 15604, 15605, 15606, 15607, 15608, 15609,
       15610, 15611, 15612, 15613, 15614, 15615, 15616, 15617, 15618,
       15619, 15620, 15621, 15622, 15623, 15624, 15625, 15626, 15627,
       15628, 15629, 15630, 15631, 15632, 15633, 15634, 15635, 15636,
       15637, 15638, 15639, 15640, 15641, 15642, 15643, 15644, 15645,
       15646, 15647, 15648, 15649, 15650, 15651, 15652, 15653, 15654,
       15655, 15656, 15657, 15658, 15659, 15660, 15661, 15662, 15663,
       15664, 15665, 15666, 15667, 15668, 15669, 15670, 15671, 15672,
       15673], dtype=int64)
```

the index in the first place:

In [34]:
```
two_countries.index.values[0]
```

Out[34]:
```
8047
```

how many rows for the two countries?

In [81]:
```
len(two_countries)
```

Out[81]:
```
433
```

In [36]:
```
two_countries.count()
```

Out[36]:

```
location                               433
iso_code                               433
date                                   433
total_vaccinations                     376
people_vaccinated                      431
people_fully_vaccinated                399
daily_vaccinations_raw                 371
daily_vaccinations                     431
total_vaccinations_per_hundred         376
people_vaccinated_per_hundred          431
people_fully_vaccinated_per_hundred    399
daily_vaccinations_per_million         431
dtype: int64
```

At the end of the file we have some world data.

Use str.contains if you're not sure how this location is called

In [86]:

```python
vacc_df[vacc_df['location'].str.contains('Worl')]
```

Out[86]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	tot
34612	World	OWID_WRL	2020-12-02	0.000000e+00	0.000000e+00	NaN	NaN	NaN	
34613	World	OWID_WRL	2020-12-03	0.000000e+00	0.000000e+00	NaN	0.0	0.0	
34614	World	OWID_WRL	2020-12-04	1.000000e+00	1.000000e+00	NaN	1.0	0.0	
34615	World	OWID_WRL	2020-12-05	1.000000e+00	1.000000e+00	NaN	0.0	0.0	
34616	World	OWID_WRL	2020-12-06	1.000000e+00	1.000000e+00	NaN	0.0	0.0	
...	...	...	...	...	...	...	...	...	
34840	World	OWID_WRL	2021-07-18	3.662774e+09	2.055135e+09	1.015653e+09	19296764.0	30735269.0	
34841	World	OWID_WRL	2021-07-19	3.695078e+09	2.067338e+09	1.025288e+09	32304332.0	31101056.0	
34842	World	OWID_WRL	2021-07-20	3.727059e+09	2.079488e+09	1.033198e+09	31980155.0	30874061.0	
34843	World	OWID_WRL	2021-07-21	3.760632e+09	2.087375e+09	1.043255e+09	33573399.0	30413169.0	
34844	World	OWID_WRL	2021-07-22	3.787027e+09	2.092904e+09	1.048364e+09	26394993.0	29076881.0	

233 rows × 12 columns

Remove the world data:

In [38]:

```python
vacc_df_noWorld = vacc_df.loc[vacc_df.location != 'World']
vacc_df_noWorld.tail()
```

Out[38]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	tota
35158	Zimbabwe	ZWE	2021-07-17	1771434.0	1132045.0	639389.0	39694.0	41958.0	
35159	Zimbabwe	ZWE	2021-07-18	1785533.0	1144379.0	641154.0	14099.0	42019.0	
35160	Zimbabwe	ZWE	2021-07-19	1827638.0	1184435.0	643203.0	42105.0	42253.0	
35161	Zimbabwe	ZWE	2021-07-20	1897337.0	1247494.0	649843.0	69699.0	45971.0	
35162	Zimbabwe	ZWE	2021-07-21	1949472.0	1292642.0	656830.0	52135.0	47976.0	

Find the country with the maximum vaccinations:

In [87]:

```python
max_vacc = vacc_df_noWorld['total_vaccinations'].max()
max_vacc
```

Out[87]:

```
2383659444.0
```

In [40]:

```python
vacc_df_noWorld.loc[vacc_df_noWorld.total_vaccinations == max_vacc]
```

Out[40]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_
1950	Asia	OWID_ASI	2021-07-22	2.383659e+09	1.246089e+09	461410827.0	19637719.0	19995909.0	

What do you think this function does?

In [41]:

```python
vacc_df_noWorld.total_vaccinations.mean()
```

Out[41]:

```
36918988.861587614
```

Your turn:

Select the number of daily vaccinations in Israel on date 2021-02-06 (hint: use &)

In [88]:

```python
vacc_df.loc[(vacc_df.location == 'Israel') & (vacc_df.date == '2021-07-20')]
```

Out[88]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	total_
15671	Israel	ISR	2021-07-20	11008624.0	5750067.0	5258557.0	14231.0	10060.0	

Find all the countries with more than 3000000 vaccinations

In [43]:

```python
vacc_df_noWorld.loc[(vacc_df.daily_vaccinations > 300000)]
```

Out[43]:

	location	iso_code	date	total_vaccinations	people_vaccinated	people_fully_vaccinated	daily_vaccinations_raw	daily_vaccinations	tot
224	Africa	OWID_AFR	2021-03-27	1.030312e+07	6962741.0	3606163.0	325837.0	301268.0	
225	Africa	OWID_AFR	2021-03-28	1.056197e+07	7218927.0	3608826.0	258849.0	335169.0	
226	Africa	OWID_AFR	2021-03-29	1.077488e+07	7305693.0	3743035.0	212915.0	348122.0	
227	Africa	OWID_AFR	2021-03-30	1.097167e+07	7387045.0	3870510.0	196785.0	355345.0	
267	Africa	OWID_AFR	2021-05-09	2.119979e+07	16063514.0	5518752.0	729471.0	351787.0	
...	...	...	...	...	...	...	...	...	
33674	Upper middle income	OWID_UMC	2021-07-18	1.928636e+09	943605954.0	379618270.0	11998743.0	16733054.0	
33675	Upper middle income	OWID_UMC	2021-07-19	1.943552e+09	945911507.0	381584744.0	14915465.0	16626595.0	
33676	Upper middle income	OWID_UMC	2021-07-20	1.961319e+09	950175010.0	384027832.0	17767193.0	16728652.0	
33677	Upper middle income	OWID_UMC	2021-07-21	1.980014e+09	953111895.0	386240484.0	18694961.0	16468485.0	
33678	Upper middle income	OWID_UMC	2021-07-22	2.000667e+09	955873275.0	388296478.0	20652943.0	16617879.0	

3614 rows × 12 columns

Summary of the functions in this unit:

- `.index.values` - the row indexes of this part of the dataframes
- `.str.contains` - selects rows and columns that contain a string
- `.max` - maximum value
- `.mean` - average value
- `.count` - the number of rows that contain a value
- `len()` - dataframe length