LEX Session: Data analytics tutorial on COVID-19 dataset with Python & SQL

by Neven Dujmovic

In this tutorial, we will learn how to prepare analytics environment on Windows Desktop, install development tools and do a simple data analysis like it could be done with Excel.

My assumption is that you are a total beginner in SQL and Python and that you are using the Microsoft Windows environment. Just to mention that switch to Linux is very easy since SQL and Python are fully multiplatform language.

We will not go in depth with any of the programing languages but instead have real analytics problem that can be resolved with usage of SQL or Python. After your analytics environment is prepared, we will take next five steps that are mandatory in any research with data:

- Set your analytics goal – what is real life problem that needs research
- Collect data – find sources of data
- Prepare data – check data quality, integrity, completeness and perform data cleaning (avoid GIGO – “Garbage In, Garbage Out” situation)
- Analyze data – fun part
- Interpret results – understood results, do reality check, go back to phase 2 or 3 if needed, do data visualization, prepare actions

***

Step 1: Set your data analytics environment

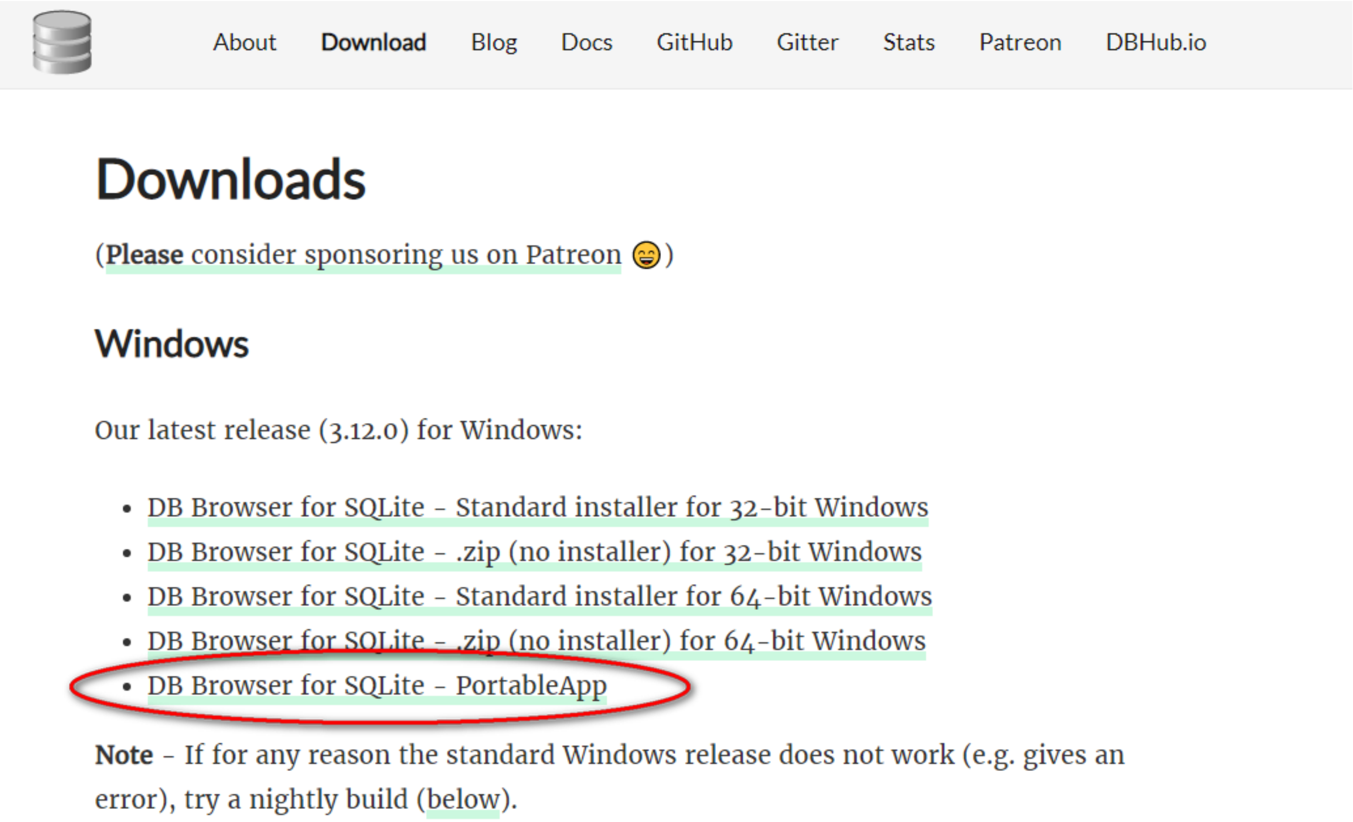
a) Install SQLite on your windows machine

SQLite is a relational database and the special thing about it is that is not a classical client-server database. Rather it is embedded into the end program and it can be used as portable application that can run even without installation on your system.

SQLite is mostly compliant with the SQL standard, generally following PostgreSQL database syntax. It is lightweight, and it can be easily used for small analysis and training.

- Go to web location to download it:

https://sqlitebrowser.org/dl/



Downloads

(**Please** consider sponsoring us on Patreon 😄)

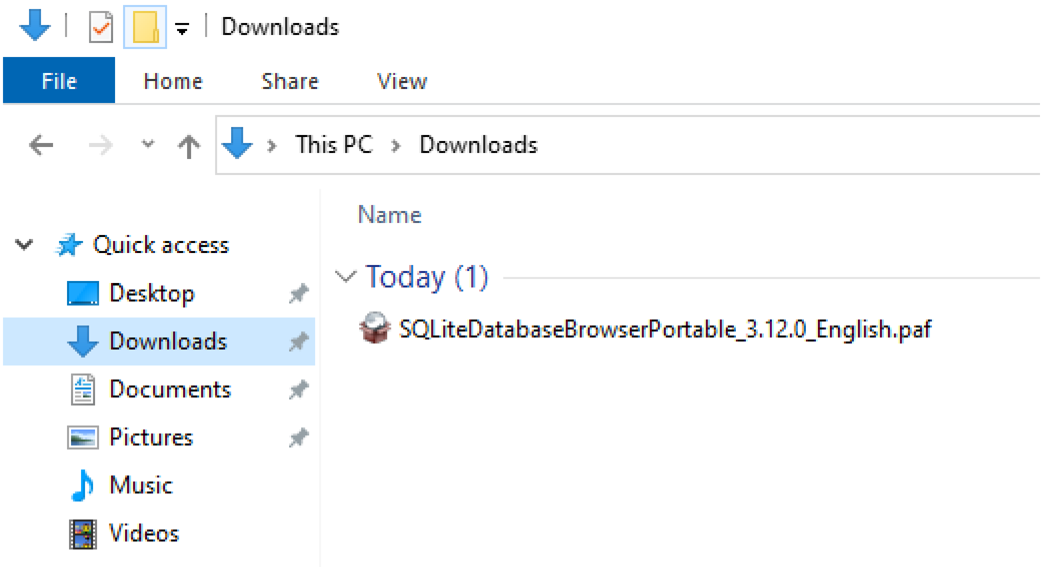
Windows

Our latest release (3.12.0) for Windows:

- DB Browser for SQLite – Standard installer for 32-bit Windows
- DB Browser for SQLite – .zip (no installer) for 32-bit Windows
- DB Browser for SQLite – Standard installer for 64-bit Windows
- DB Browser for SQLite – .zip (no installer) for 64-bit Windows
- DB Browser for SQLite – PortableApp

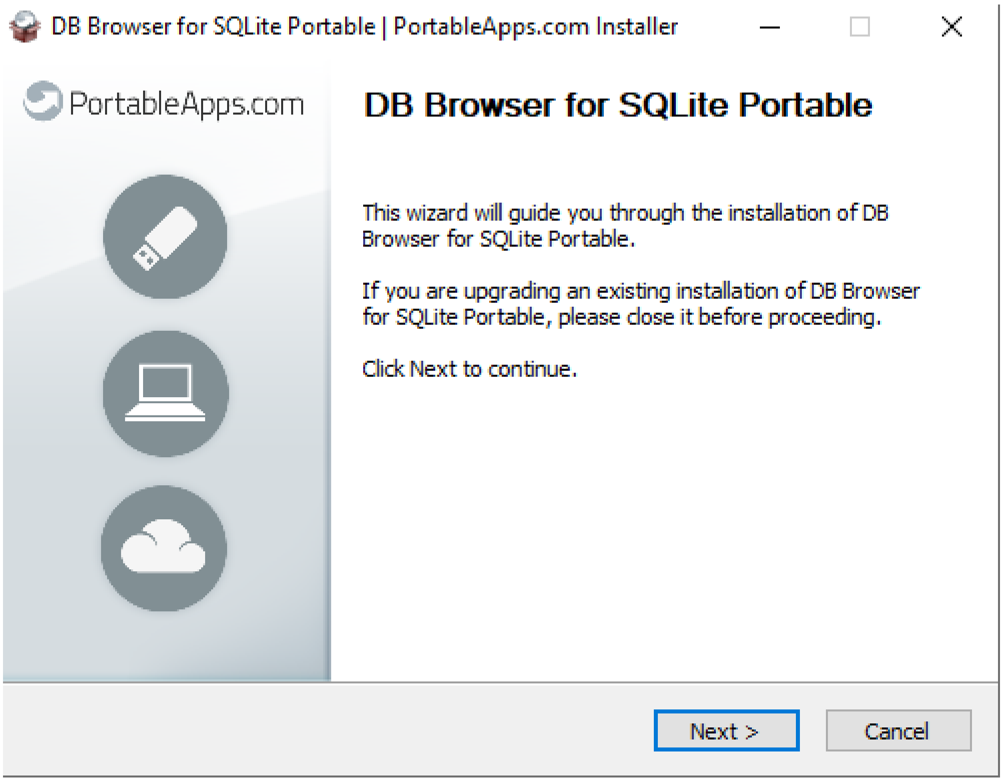
Note – If for any reason the standard Windows release does not work (e.g. gives an error), try a nightly build (below).

- Click on the "DB Browser for SQLite - PortableApp" and download archive file containing SQLite portable application.



SQLiteDatabaseBrowserPortable_3.12.0_English.paf

- Locate the file "SQLiteDatabaseBrowserPortable_3.12.0_English.paf.exe" on your file system and double click on it to run extraction program to unpack portable application.



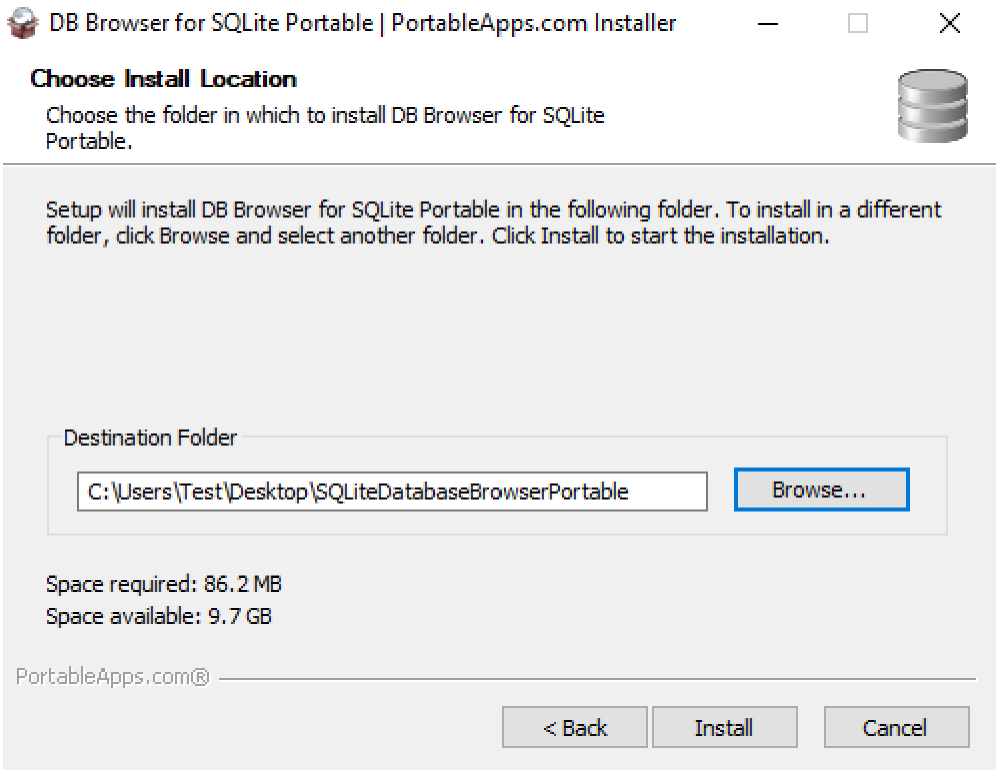
DB Browser for SQLite Portable

This wizard will guide you through the installation of DB Browser for SQLite Portable.

If you are upgrading an existing installation of DB Browser for SQLite Portable, please close it before proceeding.

Click Next to continue.

- Select the location for the portable program and click “Install”.


DB Browser for SQLite Portable | PortableApps.com Installer

Choose Install Location

Choose the folder in which to install DB Browser for SQLite Portable.

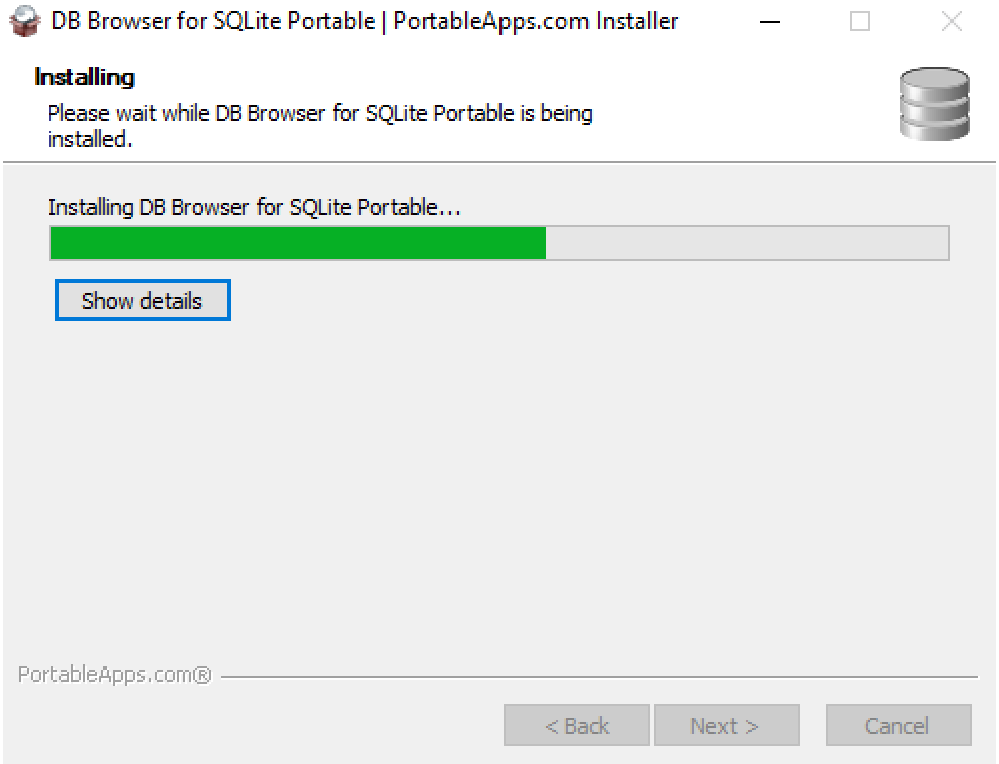
Setup will install DB Browser for SQLite Portable in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

C:\Users\Test\Desktop\SQLiteDatabaseBrowserPortable

Space required: 86.2 MB

Space available: 9.7 GB

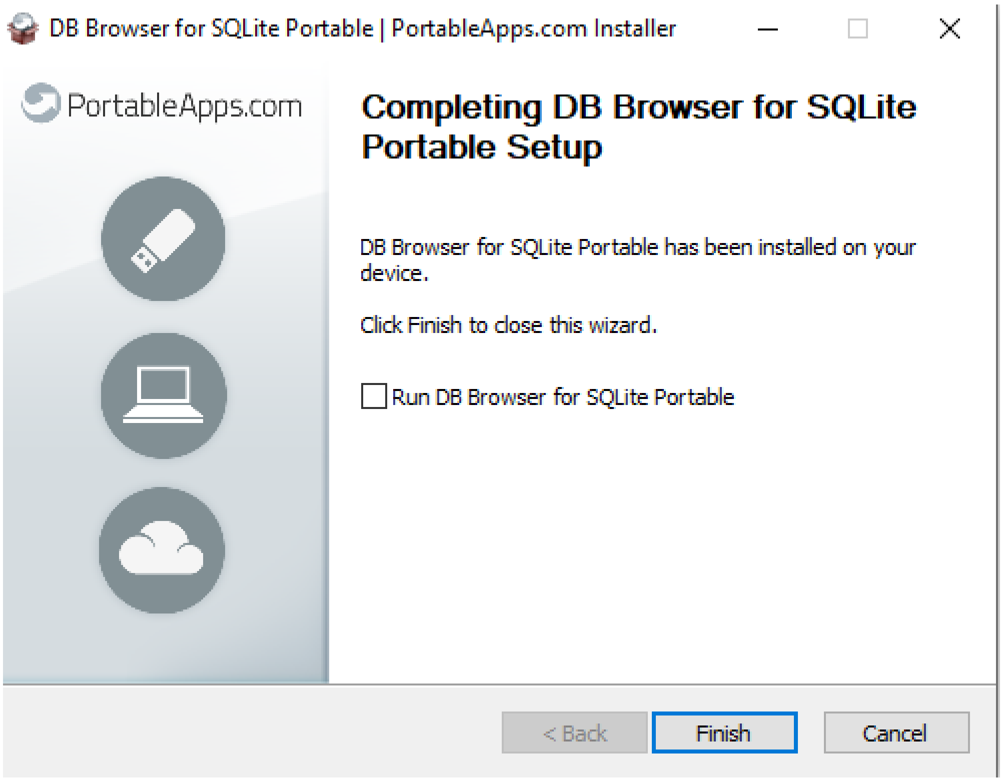

DB Browser for SQLite Portable | PortableApps.com Installer

Installing

Please wait while DB Browser for SQLite Portable is being installed.

Installing DB Browser for SQLite Portable...

- When it is done, click “Finish” to close the dialog.



DB Browser for SQLite Portable | PortableApps.com Installer

PortableApps.com

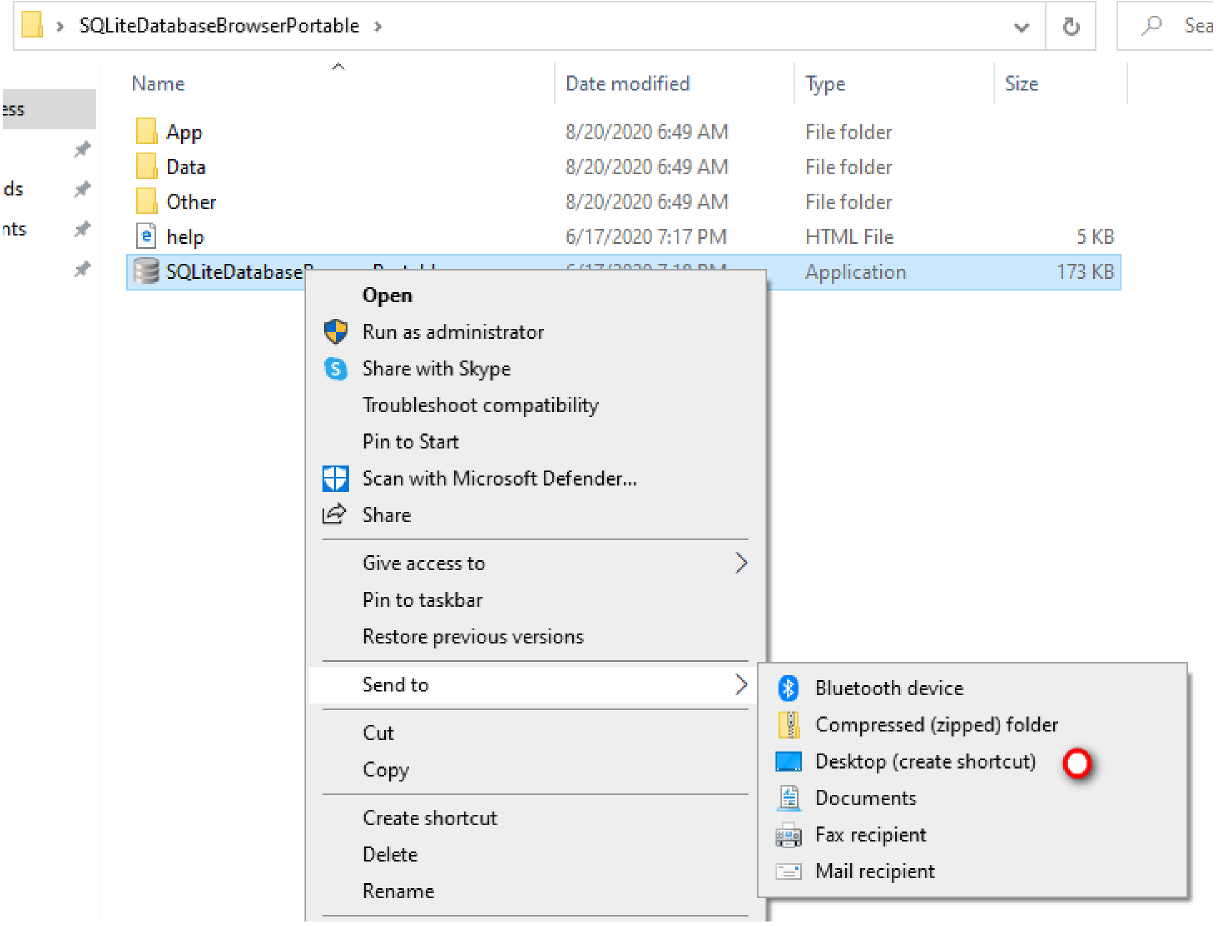
Completing DB Browser for SQLite Portable Setup

DB Browser for SQLite Portable has been installed on your device.

Click Finish to close this wizard.

□ Run DB Browser for SQLite Portable

- Optionally you can create desktop shortcut icon by going to the directory where SQLite portable application was extracted and choosing “Desktop (create shortcut)” on right click menu.

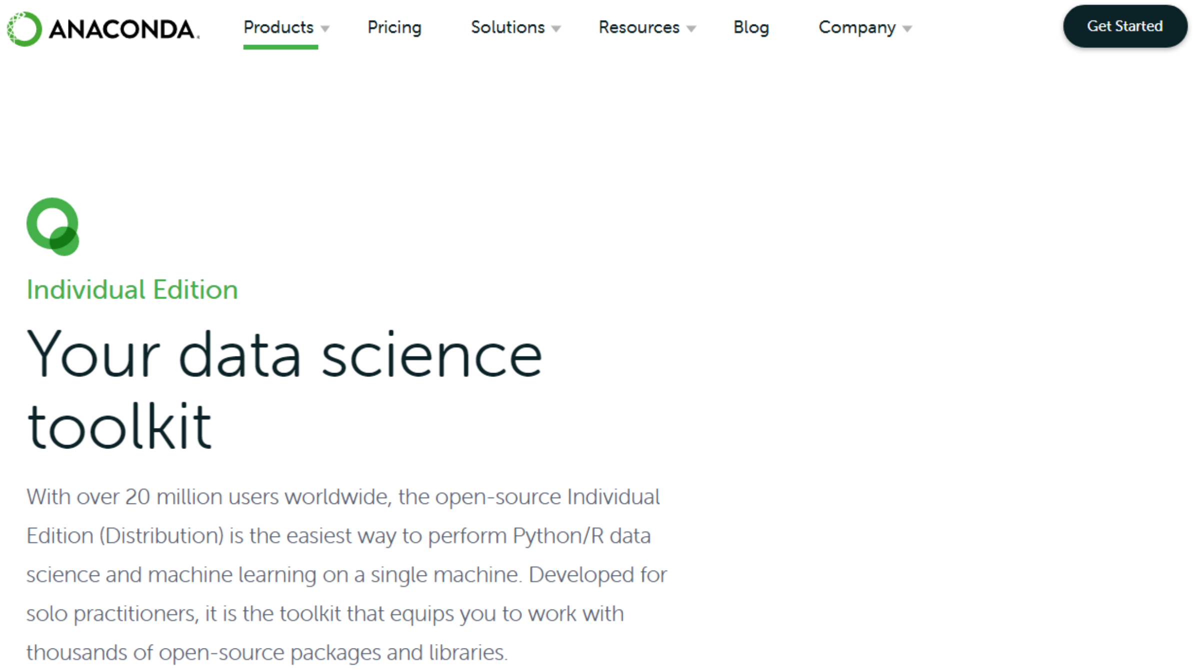


Name	Date modified	Type	Size
App	8/20/2020 6:49 AM	File folder	
Data	8/20/2020 6:49 AM	File folder	
Other	8/20/2020 6:49 AM	File folder	
help	6/17/2020 7:17 PM	HTML File	5 KB
SQLiteDatabase…		Application	173 KB

b) Install Anaconda distribution for the Python on your windows machine

My recommendation is to use Anaconda distribution for the Python for data analysis and other purposes. Anaconda offers a free and open-source distribution of the Python programming language for scientific computing (data science, machine learning applications, etc.). The main advantage is simplified package management and deployment (over 250 packages automatically installed, and over 7,500 additional open-source packages).

- Download Anaconda Individual Edition via link: https://www.anaconda.com/products/individual

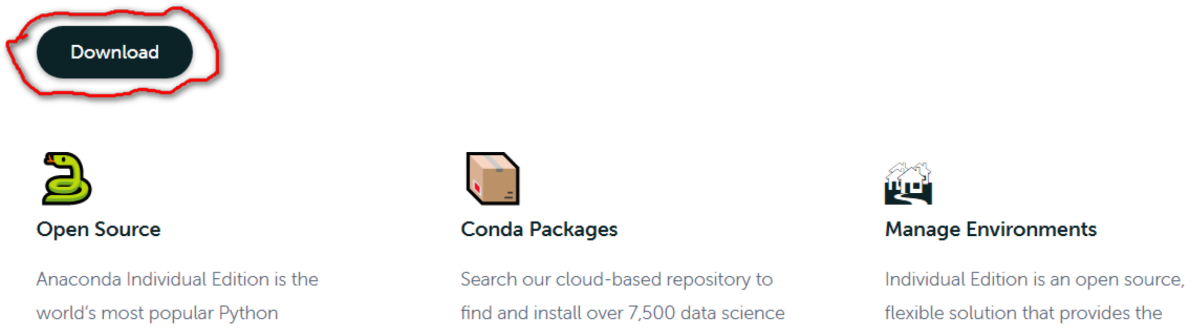


Individual Edition

Your data science toolkit

With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

- Scroll down to “Download” section.



Open Source

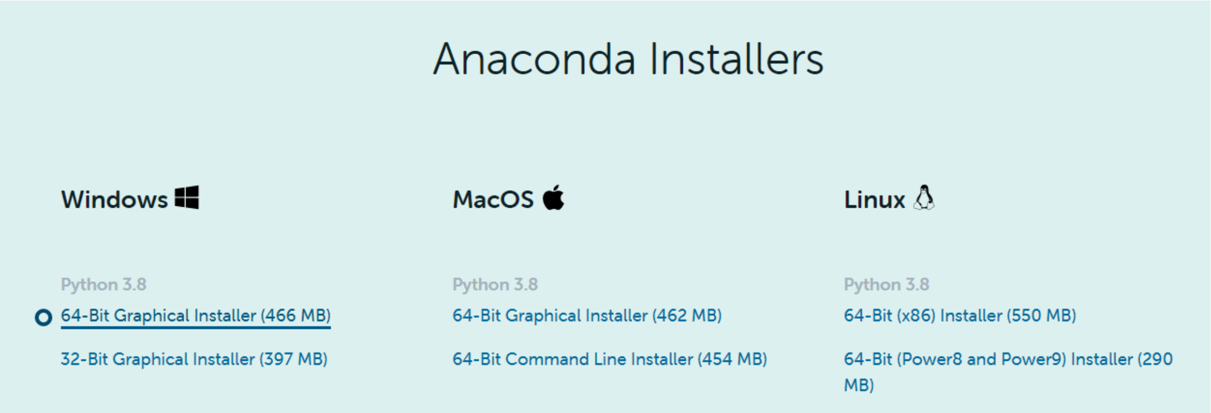
Anaconda Individual Edition is the world’s most popular Python

Conda Packages

Search our cloud-based repository to find and install over 7,500 data science

Manage Environments

Individual Edition is an open source, flexible solution that provides the



Anaconda Installers

Windows

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

MacOS

Python 3.8

64-Bit Graphical Installer (462 MB)

64-Bit Command Line Installer (454 MB)

Linux

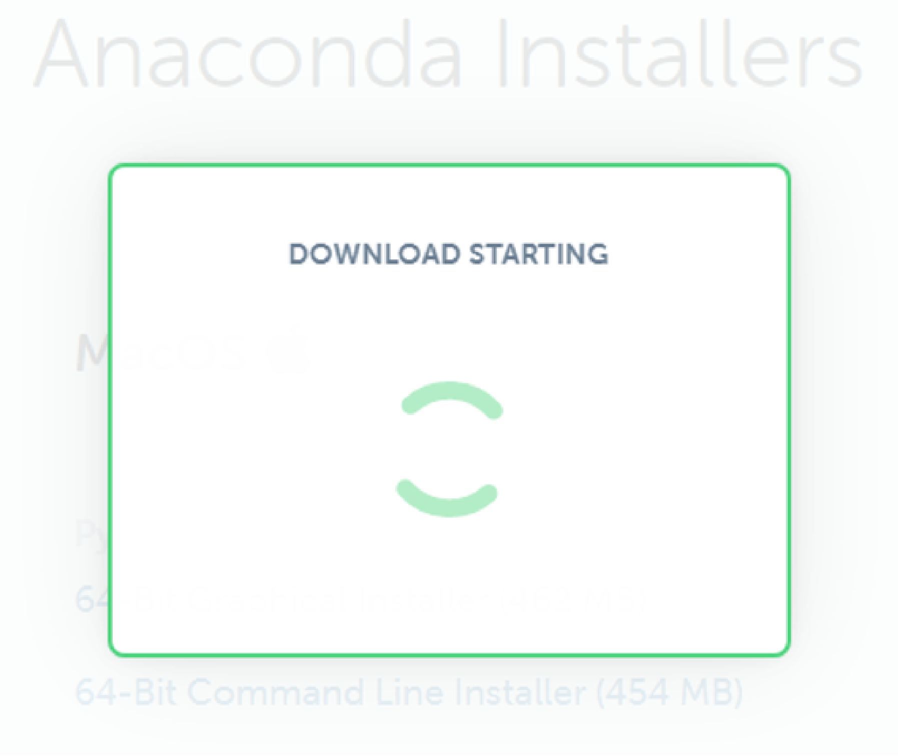
Python 3.8

64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)

- Click on the link for the “64-Bit Graphical Installer (466 MB)” version.

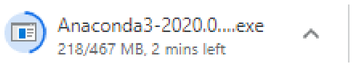
This will start the download and that can take a while.



Anaconda Installers

DOWNLOAD STARTING

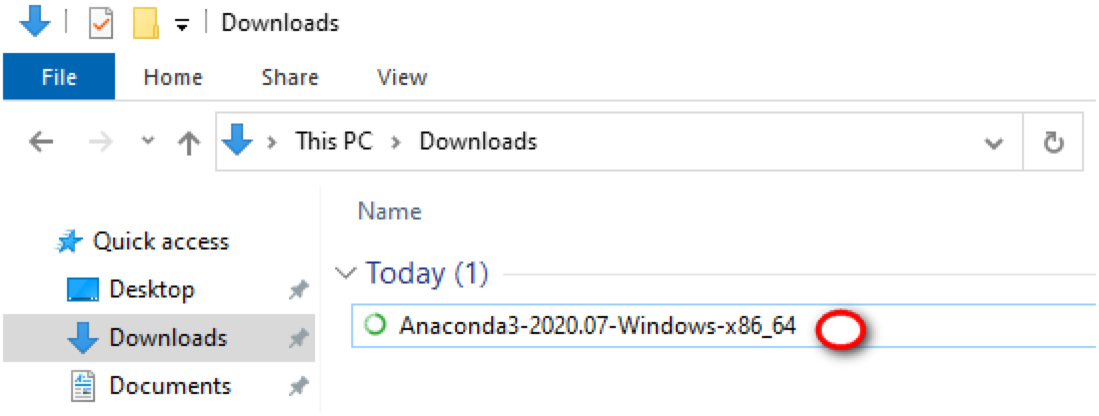
64-Bit Command Line Installer (454 MB)



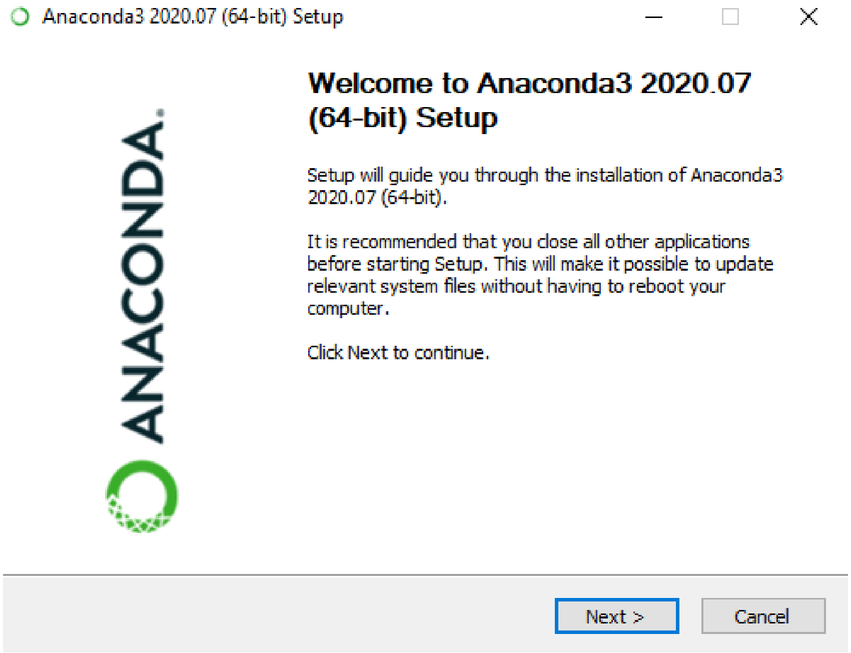
Anaconda3-2020.0....exe

218/467 MB, 2 mins left

- Start the installation by double clicking on the installation file on your download directory



- Install Anaconda Individual Edition with default features.

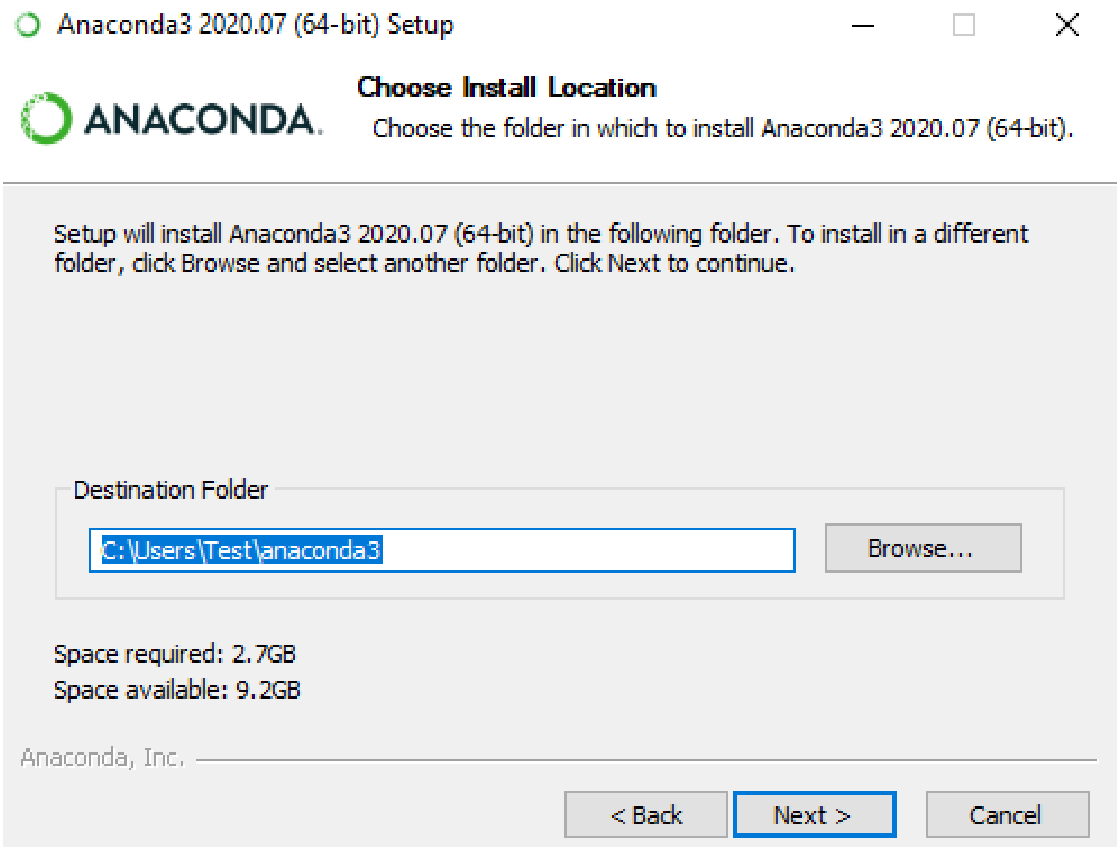


Welcome to Anaconda3 2020.07 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 2020.07 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.



Anaconda3 2020.07 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Anaconda3 2020.07 (64-bit).

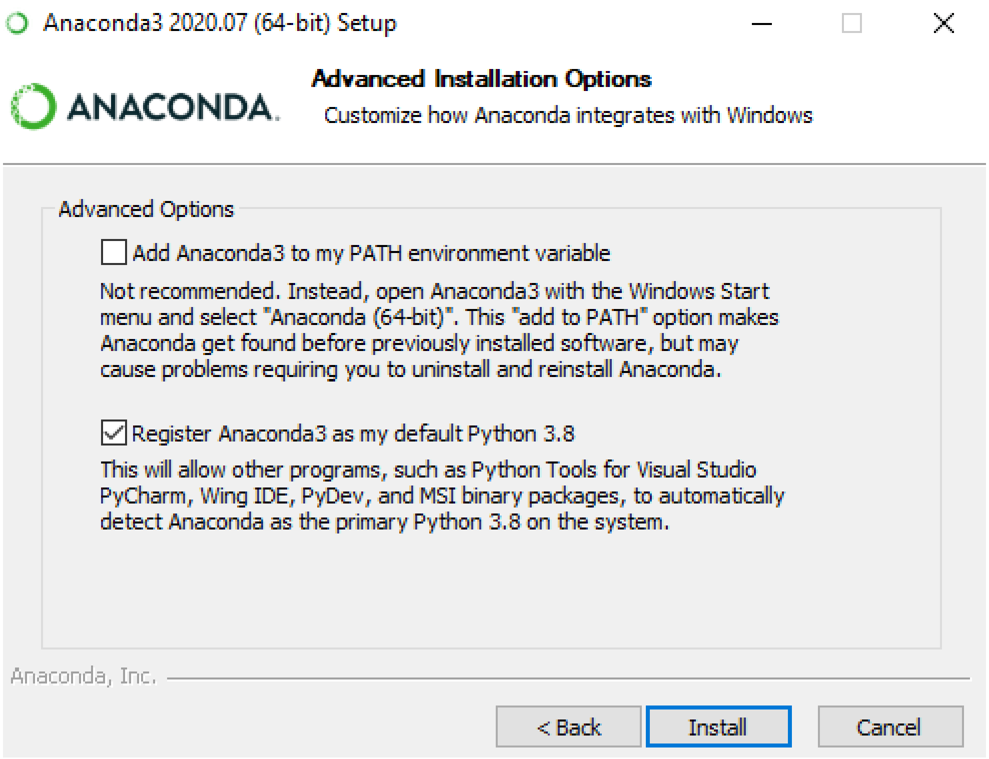
Setup will install Anaconda3 2020.07 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

C:\Users\Test\anaconda3

Space required: 2.7GB
Space available: 9.2GB

Anaconda, Inc.



Anaconda3 2020.07 (64-bit) Setup

Advanced Installation Options

Customize how Anaconda integrates with Windows

Advanced Options

□ Add Anaconda3 to my PATH environment variable

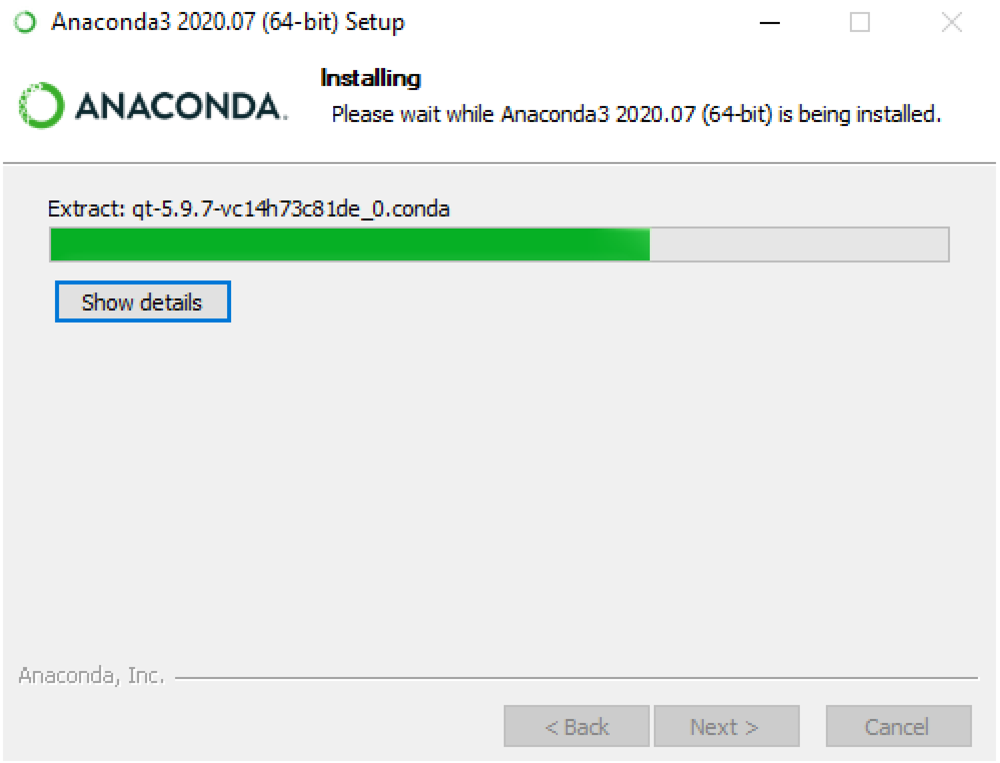
Not recommended. Instead, open Anaconda3 with the Windows Start menu and select "Anaconda (64-bit)". This "add to PATH" option makes Anaconda get found before previously installed software, but may cause problems requiring you to uninstall and reinstall Anaconda.

☑ Register Anaconda3 as my default Python 3.8

This will allow other programs, such as Python Tools for Visual Studio PyCharm, Wing IDE, PyDev, and MSI binary packages, to automatically detect Anaconda as the primary Python 3.8 on the system.

Anaconda, Inc.

Be patient as this can take some time depending on your system performance. A lot of cool packages for data analytics are coming your way! 😊

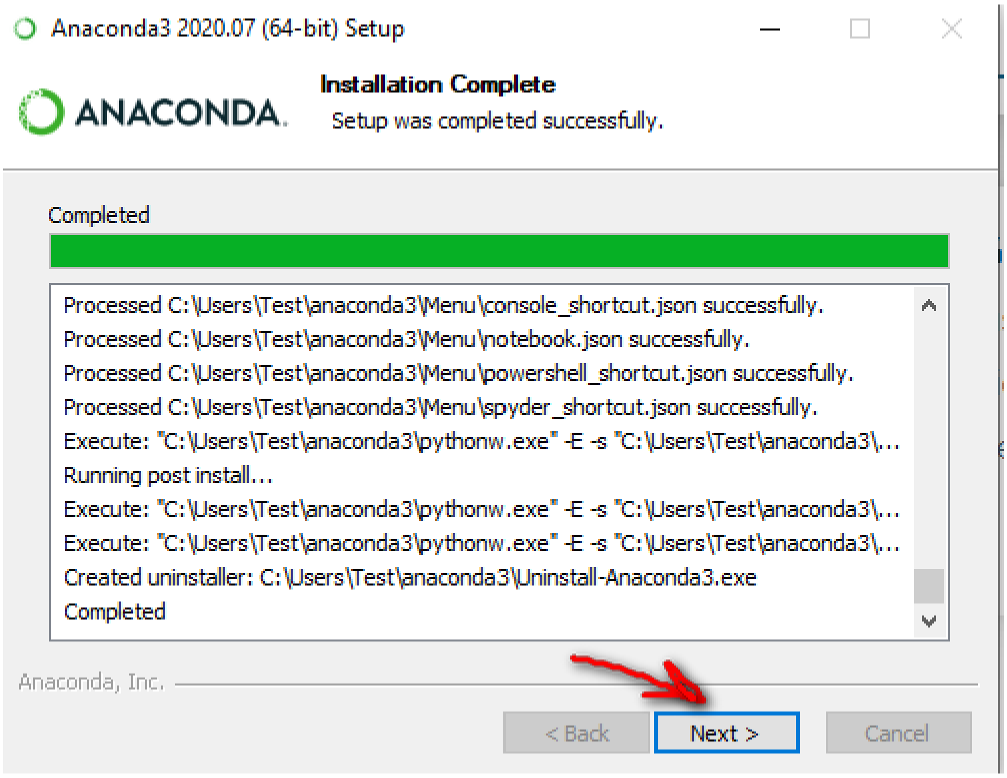


Installing

Please wait while Anaconda3 2020.07 (64-bit) is being installed.

Extract: qt-5.9.7-vc14h73c81de_0.conda

- When it is done, click “Next”, “Next” and “Finish” to end installation process.

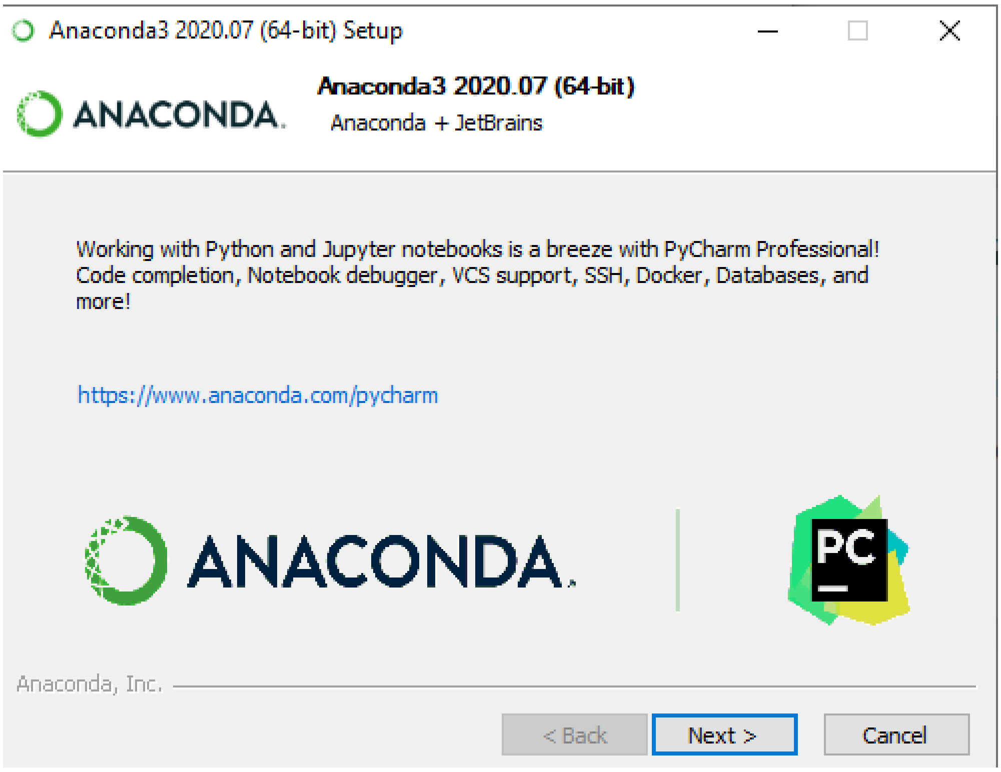


Installation Complete

Setup was completed successfully.

Completed

```
Processed C:\Users\Test\anaconda3\Menu\console_shortcut.json successfully.
Processed C:\Users\Test\anaconda3\Menu\notebook.json successfully.
Processed C:\Users\Test\anaconda3\Menu\powershell_shortcut.json successfully.
Processed C:\Users\Test\anaconda3\Menu\spyder_shortcut.json successfully.
Execute: "C:\Users\Test\anaconda3\pythonw.exe" -E -s "C:\Users\Test\anaconda3\...
Running post install...
Execute: "C:\Users\Test\anaconda3\pythonw.exe" -E -s "C:\Users\Test\anaconda3\...
Execute: "C:\Users\Test\anaconda3\pythonw.exe" -E -s "C:\Users\Test\anaconda3\...
Created uninstaller: C:\Users\Test\anaconda3\Uninstall-Anaconda3.exe
Completed
```

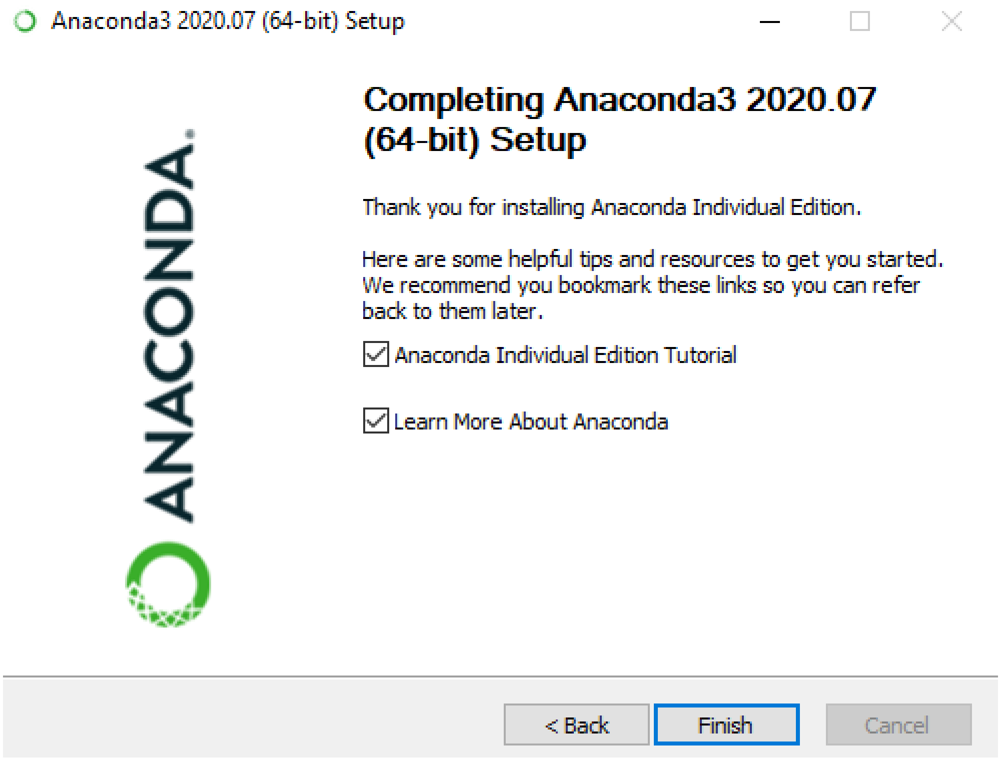


Anaconda3 2020.07 (64-bit) Setup

Anaconda3 2020.07 (64-bit)
Anaconda + JetBrains

Working with Python and Jupyter notebooks is a breeze with PyCharm Professional! Code completion, Notebook debugger, VCS support, SSH, Docker, Databases, and more!

https://www.anaconda.com/pycharm



Anaconda3 2020.07 (64-bit) Setup

Completing Anaconda3 2020.07 (64-bit) Setup

Thank you for installing Anaconda Individual Edition.

Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

☑ Anaconda Individual Edition Tutorial

☑ Learn More About Anaconda

If you have time, I encourage you to review documentation, but since this is a fast tutorial, we will not do it now. Let's better use our data analytics environment to do some serious data processing! 😊

Step 2: Set your data analytics goal

This the shortest section but it most important. Here we define the reason why we are starting with this project task.

Our data analytics project will deal with the following problem:

- Find the effectiveness of testing of COVID-19 per country by comparing the number of tests for a single day and the number of new cases.

Reasoning: High efficiency in testing could mean that there is not enough testing, i.e. that there is a significantly higher number of infected people in the population than it was detected.

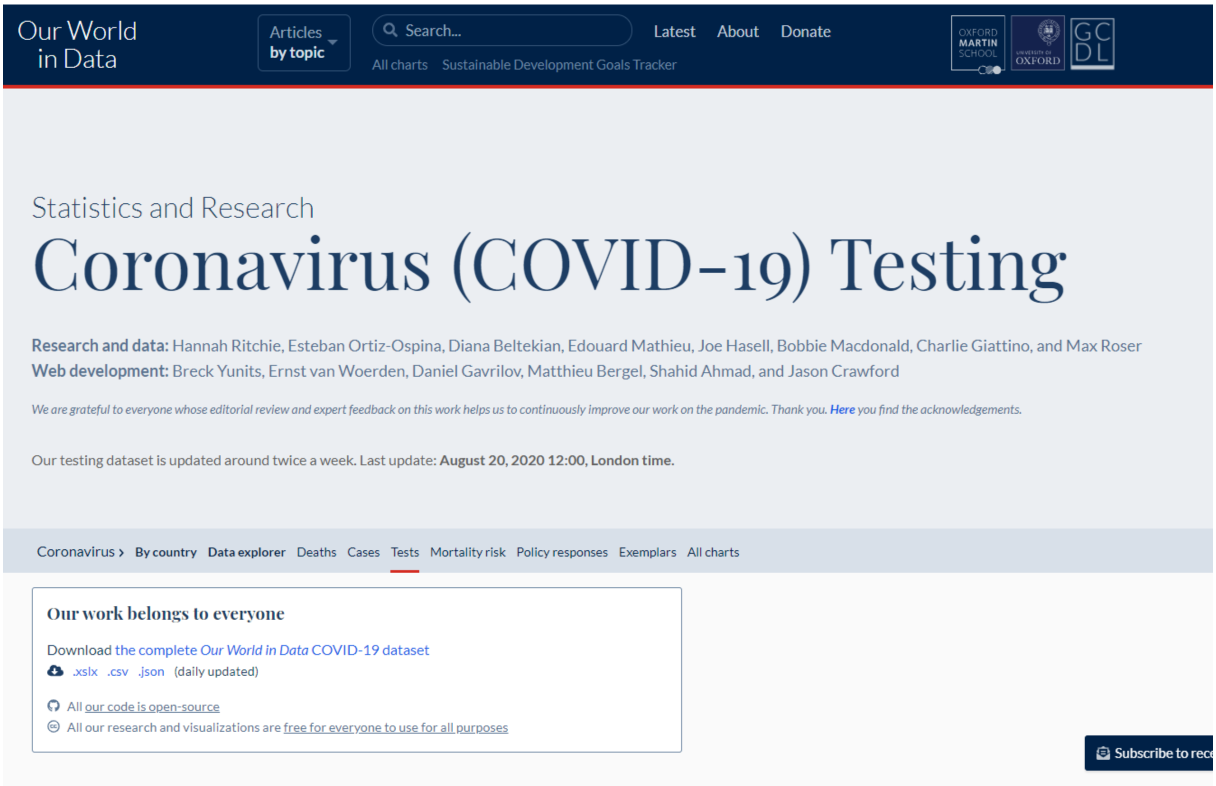
DISCLAIMER: *The purpose of the following data analysis is educational and done on a dataset that was not verified by its accuracy. The results of this analysis, therefore, are not to be interpreted for any other purpose than educational or to be used as accurate reports on COVID-19 tests and cases.*

I had to mention this since there is always a chance that the intention of this exercise could be misinterpreted. I wanted to show you how data analysis is important for real-life situations and this is the main reason to use the COVID-19 dataset. 😊

Step 3: Collect data

A sample data for the analysis could be found on this excellent web resource:

https://ourworldindata.org/coronavirus-testing



Statistics and Research

Coronavirus (COVID-19) Testing

Research and data: Hannah Ritchie, Esteban Ortiz-Ospina, Diana Beltekian, Edouard Mathieu, Joe Hasell, Bobbie Macdonald, Charlie Giattino, and Max Roser
Web development: Breck Yunits, Ernst van Woerden, Daniel Gavrilov, Matthieu Bergel, Shahid Ahmad, and Jason Crawford

We are grateful to everyone whose editorial review and expert feedback on this work helps us to continuously improve our work on the pandemic. Thank you. ***Here*** *you find the acknowledgements.*

Our testing dataset is updated around twice a week. Last update: **August 20, 2020 12:00, London time.**

Our work belongs to everyone

Download the complete *Our World in Data* COVID-19 dataset
.xslx .csv .json (daily updated)

All our code is open-source
All our research and visualizations are free for everyone to use for all purposes

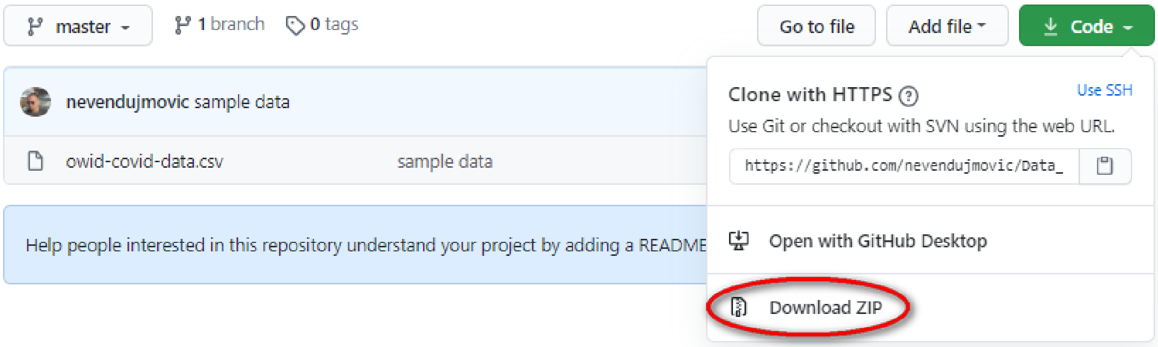
You need to download .CSV text file that contains all data in the comma-separated format:

https://covid.ourworldindata.org/data/owid-covid-data.csv

On the “Our World in Data” web site you can find the actual version of the data. However, I recommend using a version of data that I used for this analysis since it will be easier to follow the exact results and progress of activities. This version can be found on my GitHub location:

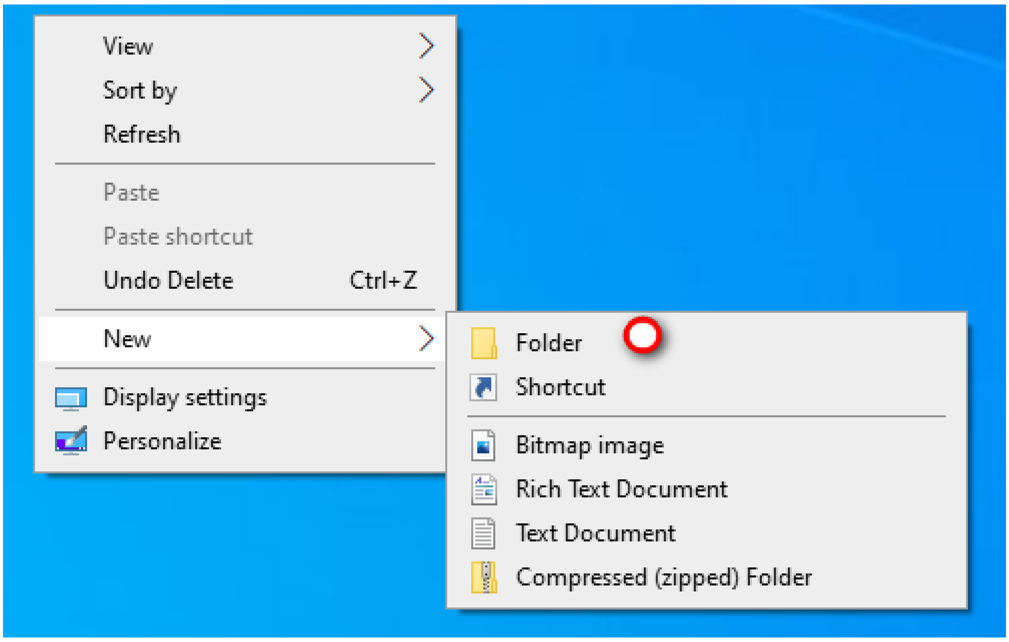
https://github.com/nevendujmovic/Data_analysis_COVID-19_dataset

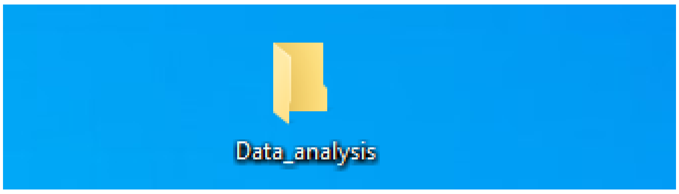
- Click on "**Code**" -> "**Download ZIP**" to download "owid-covid-data.csv" file with sample data.



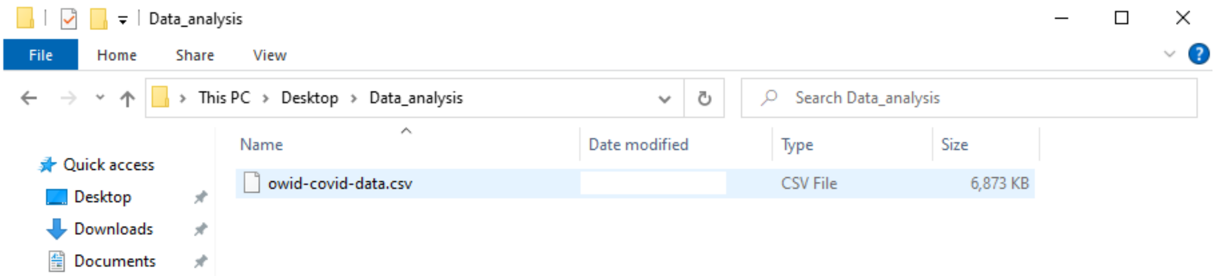
master
1 branch
0 tags
Go to file
Add file
Code
nevendujmovic sample data
owid-covid-data.csv
sample data
Help people interested in this repository understand your project by adding a README
Clone with HTTPS
Use SSH
Use Git or checkout with SVN using the web URL.
https://github.com/nevendujmovic/Data_
Open with GitHub Desktop
Download ZIP

- Make a project directory on your file system
  (e.g. in my case I created the "**Data_analysis**" folder on the desktop).





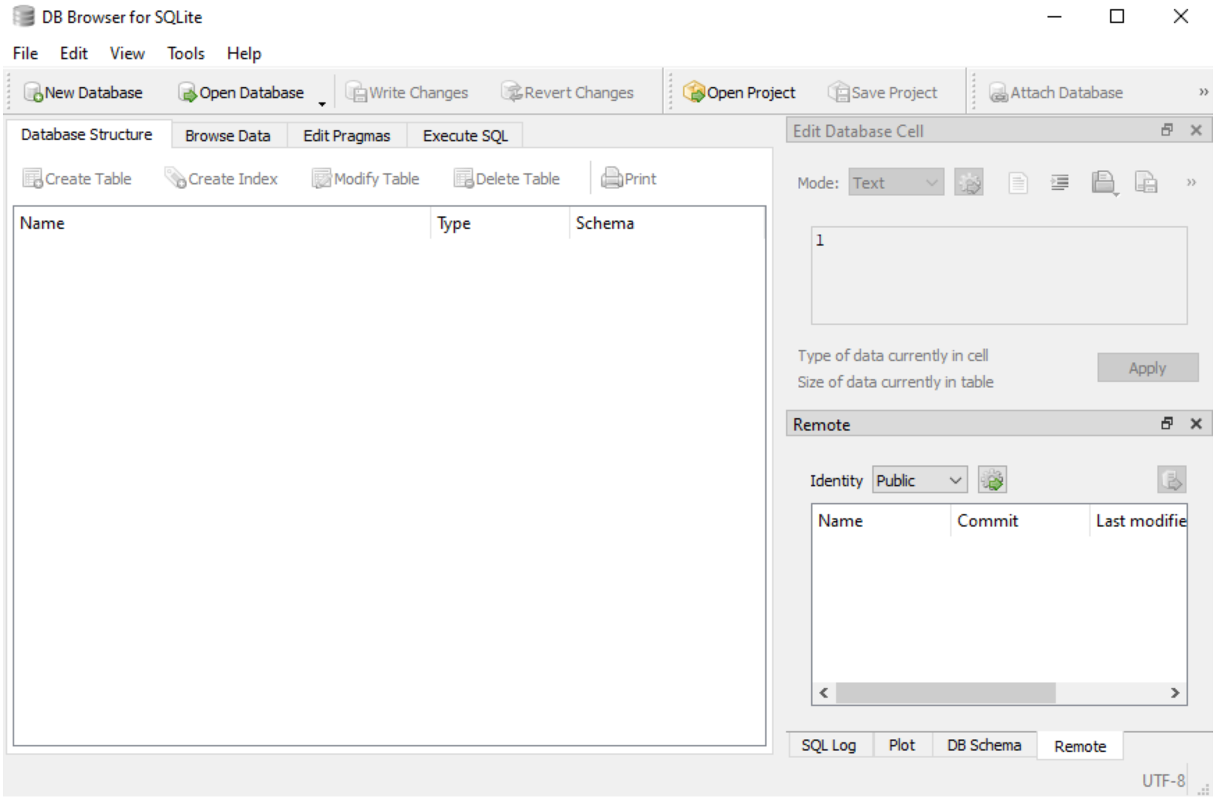
- Extract and copy "**owid-covid-data.csv**" to the project directory
  (e.g. in my case "C:\Users\<user name>\Desktop\Data_analysis")



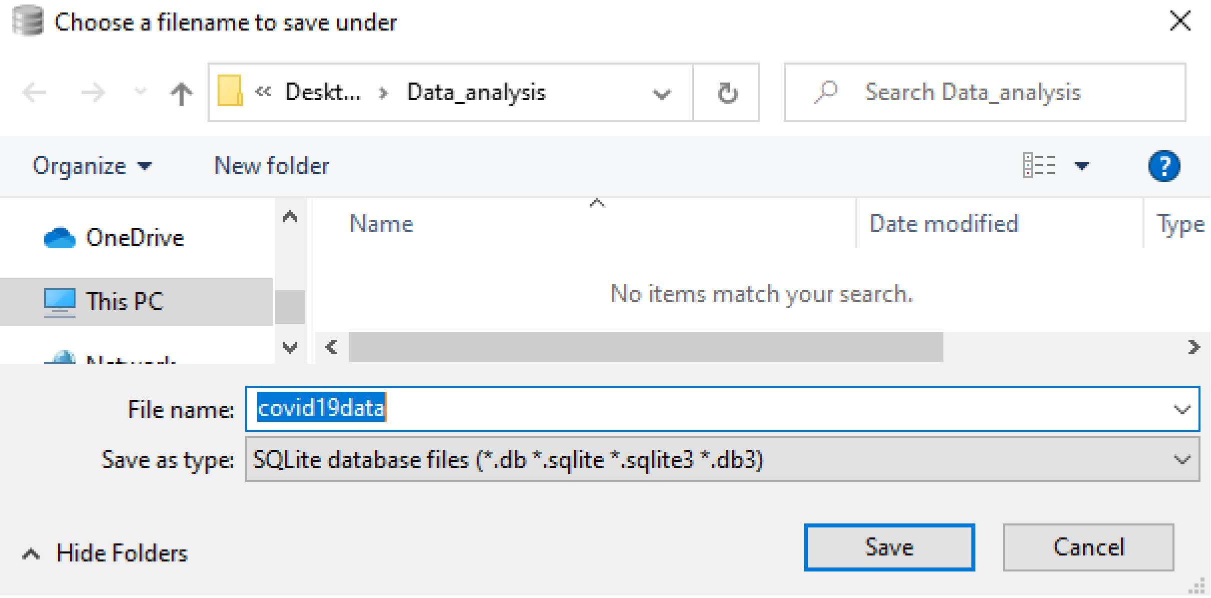
Name	Date modified	Type	Size
owid-covid-data.csv		CSV File	6,873 KB

a) Load data to the SQLite database

- Open SQLite database by clicking on the desktop shortcut or “SQLiteDatabaseBrowserPortable.exe” file in the SQLite portable application directory.



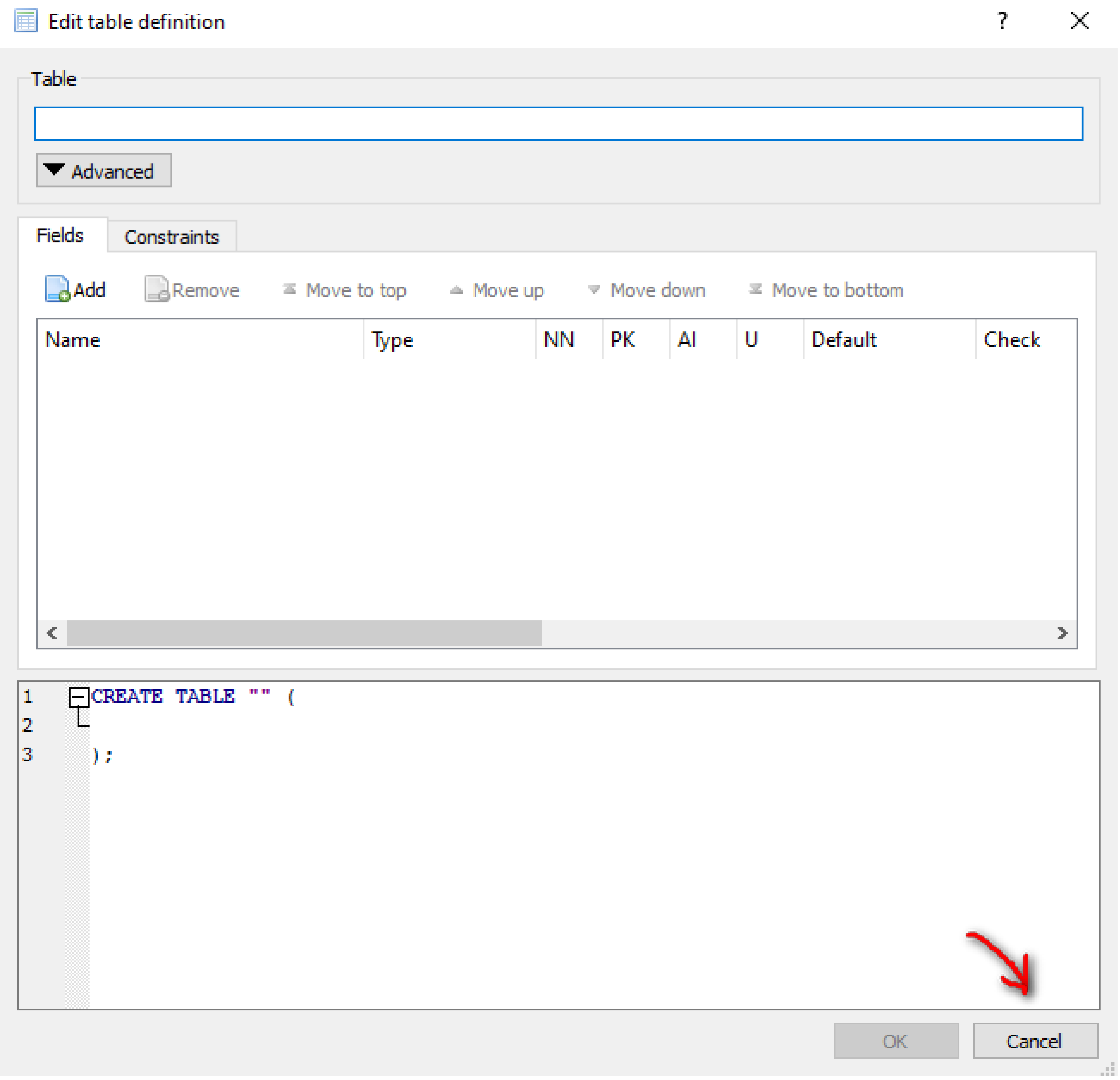
- Select “New database” icon (or on the menu go to “File” -> “New database…”) and in the “Choose a filename to save under” dialog provide name “covid19data” and location (e.g. “data_analysis” directory). Click “Save” to create a database file.



File name: covid19data

Save as type: SQLite database files (*.db *.sqlite *.sqlite3 *.db3)

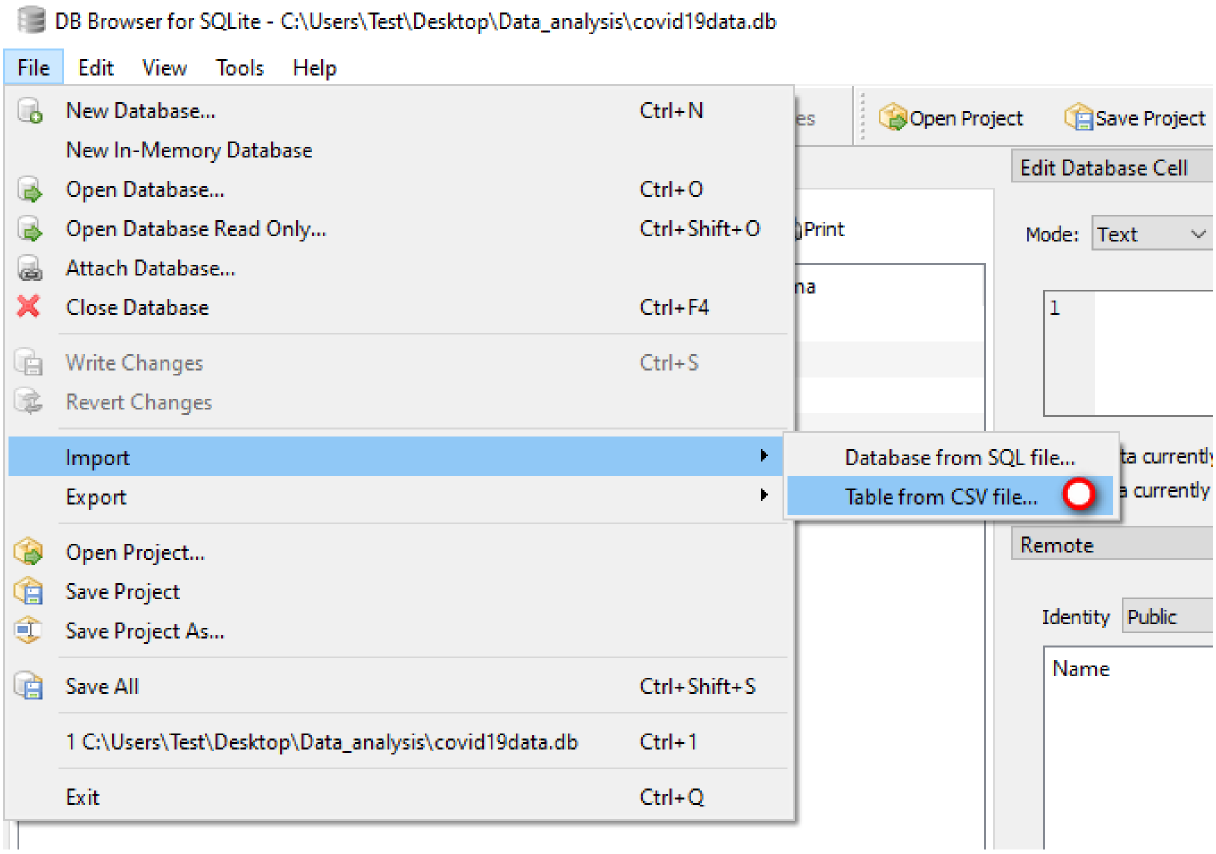
- Click “Cancel” the “Edit table definition” dialog that will open. We will not create a table in this way but rather we will import the table from sample data .CSV file.



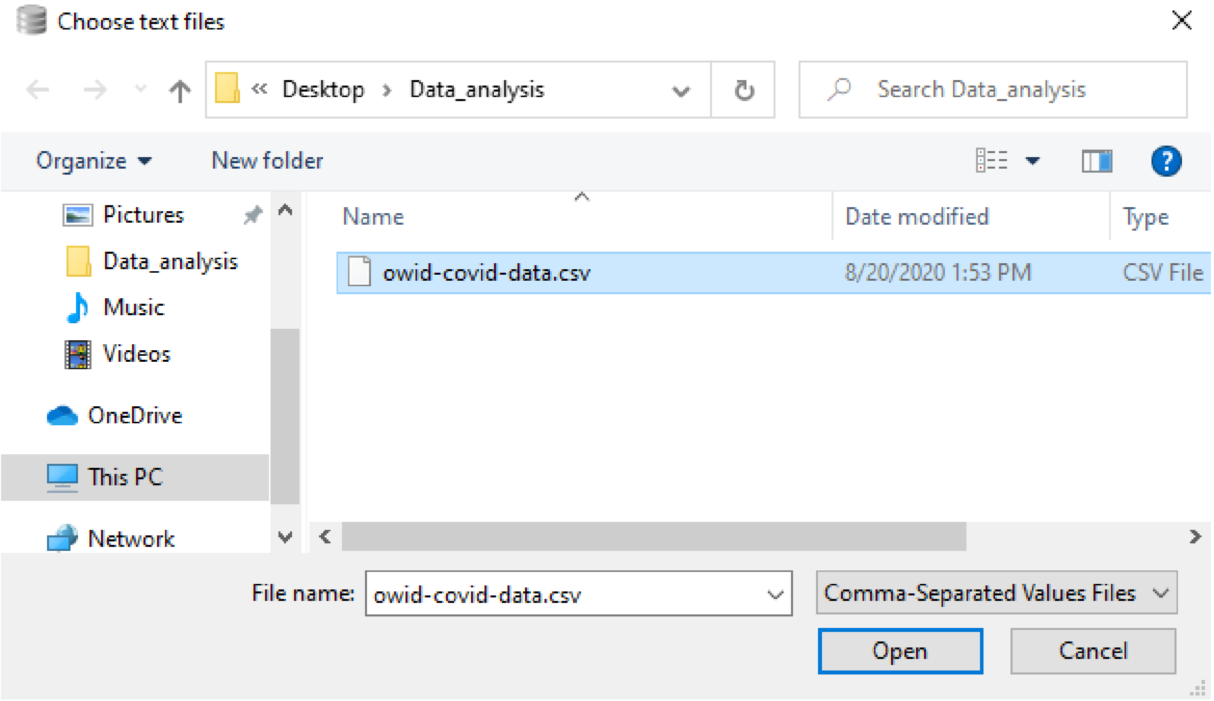
Name	Type	NN	PK	AI	U	Default	Check

```
CREATE TABLE "" (

);
```

- Go to “File” -> “Import” -> “Table from CSV file…”.



- Locate and select “owid-covid-data.csv” file from the “Data_analysis” directory and click “Open”.



Name	Date modified	Type
owid-covid-data.csv	8/20/2020 1:53 PM	CSV File

File name: owid-covid-data.csv

Comma-Separated Values Files

- In the “Import CSV file” dialog check “Column names in first line” and click “OK”.

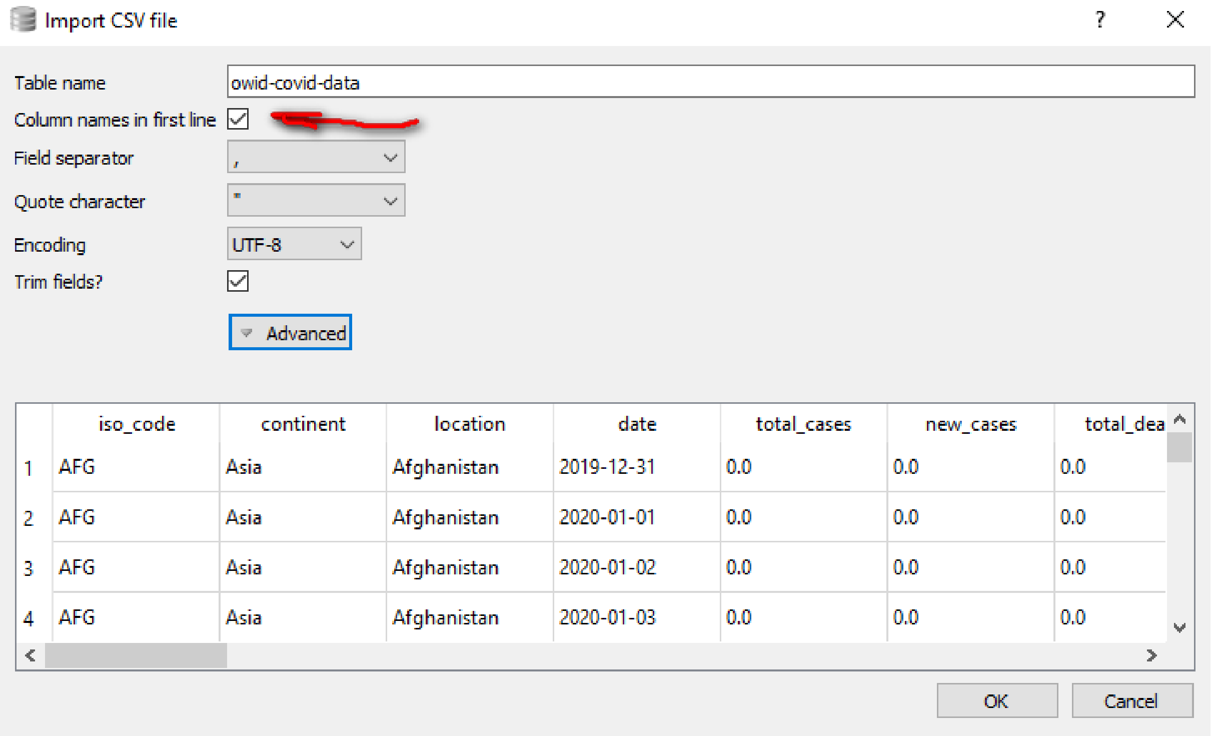


Table name: owid-covid-data

Column names in first line ☑

Field separator: ,

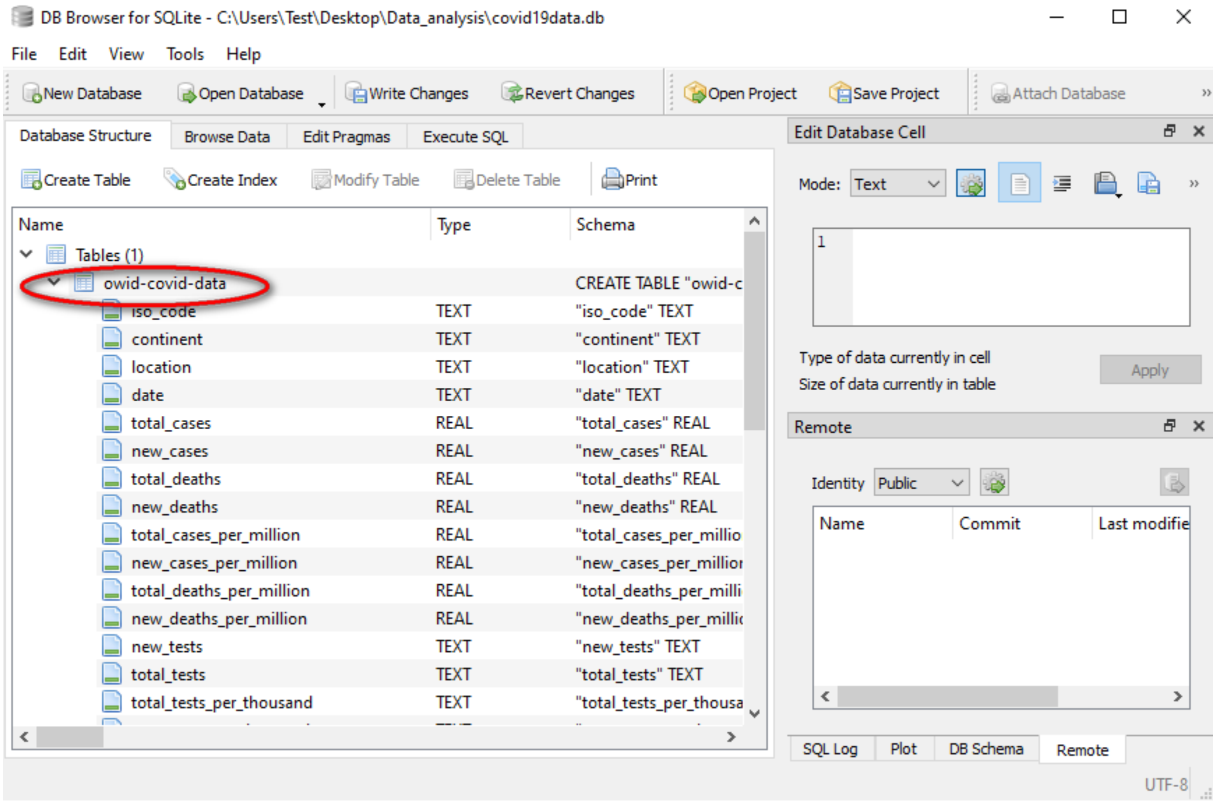
Quote character: "

Encoding: UTF-8

Trim fields? ☑

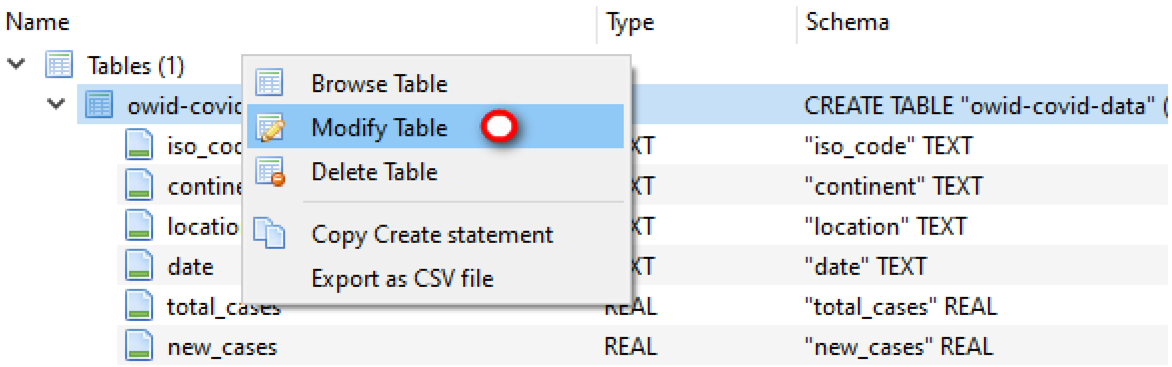
	iso_code	continent	location	date	total_cases	new_cases	total_dea
1	AFG	Asia	Afghanistan	2019-12-31	0.0	0.0	0.0
2	AFG	Asia	Afghanistan	2020-01-01	0.0	0.0	0.0
3	AFG	Asia	Afghanistan	2020-01-02	0.0	0.0	0.0
4	AFG	Asia	Afghanistan	2020-01-03	0.0	0.0	0.0

- New table “owid-covid-data” with all sample data is created. Review the list of columns to get idea of what data we have available.



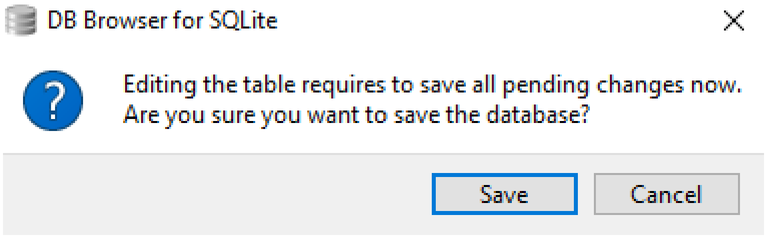
Name	Type	Schema
Tables (1)		
owid-covid-data		CREATE TABLE "owid-c
iso_code	TEXT	"iso_code" TEXT
continent	TEXT	"continent" TEXT
location	TEXT	"location" TEXT
date	TEXT	"date" TEXT
total_cases	REAL	"total_cases" REAL
new_cases	REAL	"new_cases" REAL
total_deaths	REAL	"total_deaths" REAL
new_deaths	REAL	"new_deaths" REAL
total_cases_per_million	REAL	"total_cases_per_millio
new_cases_per_million	REAL	"new_cases_per_millior
total_deaths_per_million	REAL	"total_deaths_per_milli
new_deaths_per_million	REAL	"new_deaths_per_millic
new_tests	TEXT	"new_tests" TEXT
total_tests	TEXT	"total_tests" TEXT
total_tests_per_thousand	TEXT	"total_tests_per_thousa

- Table name has not recommended character “-” and it will be best to change it immediately. Right click on the table name and select “Modify table” option.



Name	Type	Schema
Tables (1)		
owid-covic		CREATE TABLE "owid-covid-data" (
iso_coc	XT	"iso_code" TEXT
contine	XT	"continent" TEXT
locatio	XT	"location" TEXT
date	XT	"date" TEXT
total_cases	REAL	"total_cases" REAL
new_cases	REAL	"new_cases" REAL

- Click “Save” on the dialog that will pop up.



DB Browser for SQLite

Editing the table requires to save all pending changes now.
Are you sure you want to save the database?

- Change name of the table to “CovidData” and click “OK”.

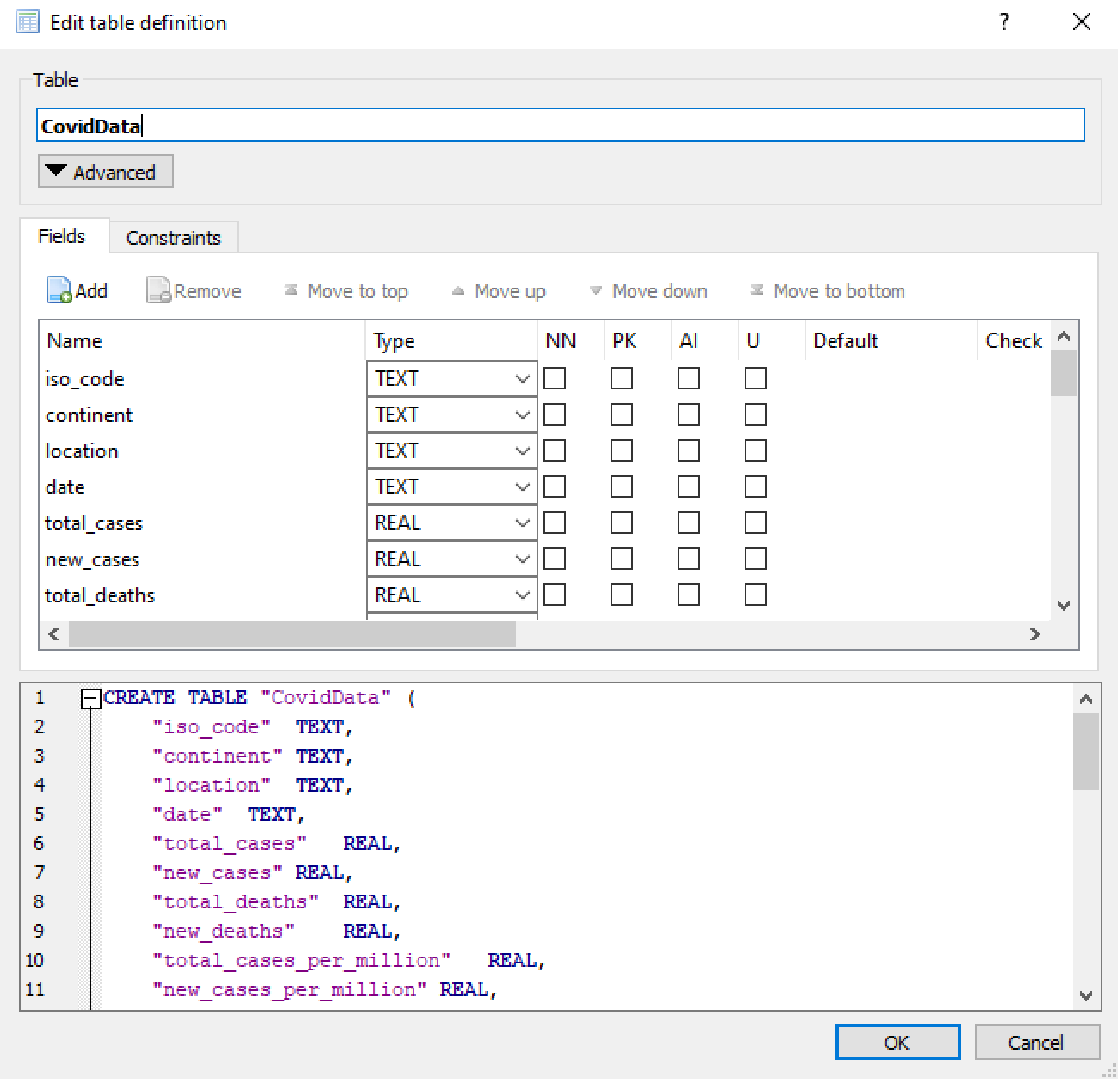


Table: CovidData

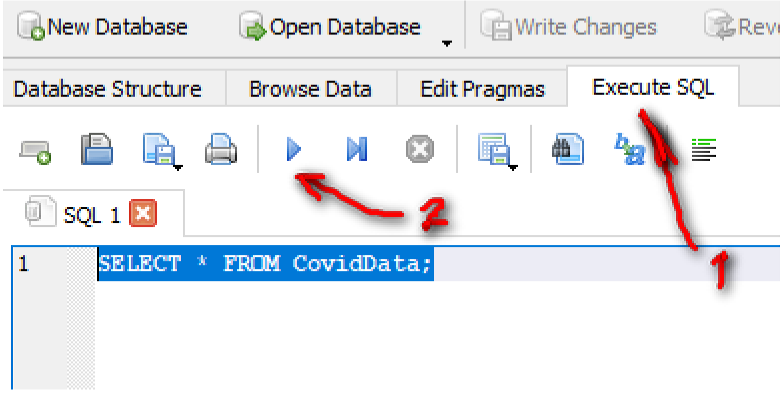
Name	Type	NN	PK	AI	U	Default	Check
iso_code	TEXT	☐	☐	☐	☐		
continent	TEXT	☐	☐	☐	☐		
location	TEXT	☐	☐	☐	☐		
date	TEXT	☐	☐	☐	☐		
total_cases	REAL	☐	☐	☐	☐		
new_cases	REAL	☐	☐	☐	☐		
total_deaths	REAL	☐	☐	☐	☐		

```
CREATE TABLE "CovidData" (
    "iso_code"  TEXT,
    "continent" TEXT,
    "location"  TEXT,
    "date"  TEXT,
    "total_cases"   REAL,
    "new_cases" REAL,
    "total_deaths"  REAL,
    "new_deaths"    REAL,
    "total_cases_per_million"   REAL,
    "new_cases_per_million" REAL,
```

- Now, let’s make a simple query to review data entries and find useful data for our analysis. Go to “Execute SQL” tab and write following SQL statement in the editor.

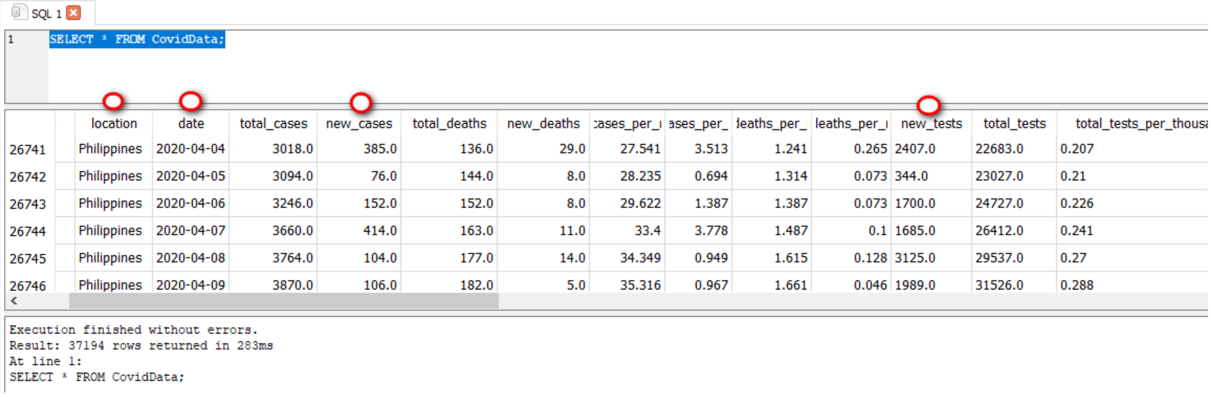
SELECT * FROM CovidData;

- Select text of written SQL statement and run it by clicking little right arrow or by pressing F5 key on keyboard.



```
1   SELECT * FROM CovidData;
```

Data will be displayed and by review of entries, we can detect data columns that are relevant for our analysis.



```
1   SELECT * FROM CovidData;
```

	location	date	total_cases	new_cases	total_deaths	new_deaths	:ases_per_ı	ases_per_	leaths_per_	leaths_per_ı	new_tests	total_tests	total_tests_per_thous
26741	Philippines	2020-04-04	3018.0	385.0	136.0	29.0	27.541	3.513	1.241	0.265	2407.0	22683.0	0.207
26742	Philippines	2020-04-05	3094.0	76.0	144.0	8.0	28.235	0.694	1.314	0.073	344.0	23027.0	0.21
26743	Philippines	2020-04-06	3246.0	152.0	152.0	8.0	29.622	1.387	1.387	0.073	1700.0	24727.0	0.226
26744	Philippines	2020-04-07	3660.0	414.0	163.0	11.0	33.4	3.778	1.487	0.1	1685.0	26412.0	0.241
26745	Philippines	2020-04-08	3764.0	104.0	177.0	14.0	34.349	0.949	1.615	0.128	3125.0	29537.0	0.27
26746	Philippines	2020-04-09	3870.0	106.0	182.0	5.0	35.316	0.967	1.661	0.046	1989.0	31526.0	0.288

```
Execution finished without errors.
Result: 37194 rows returned in 283ms
At line 1:
SELECT * FROM CovidData;
```

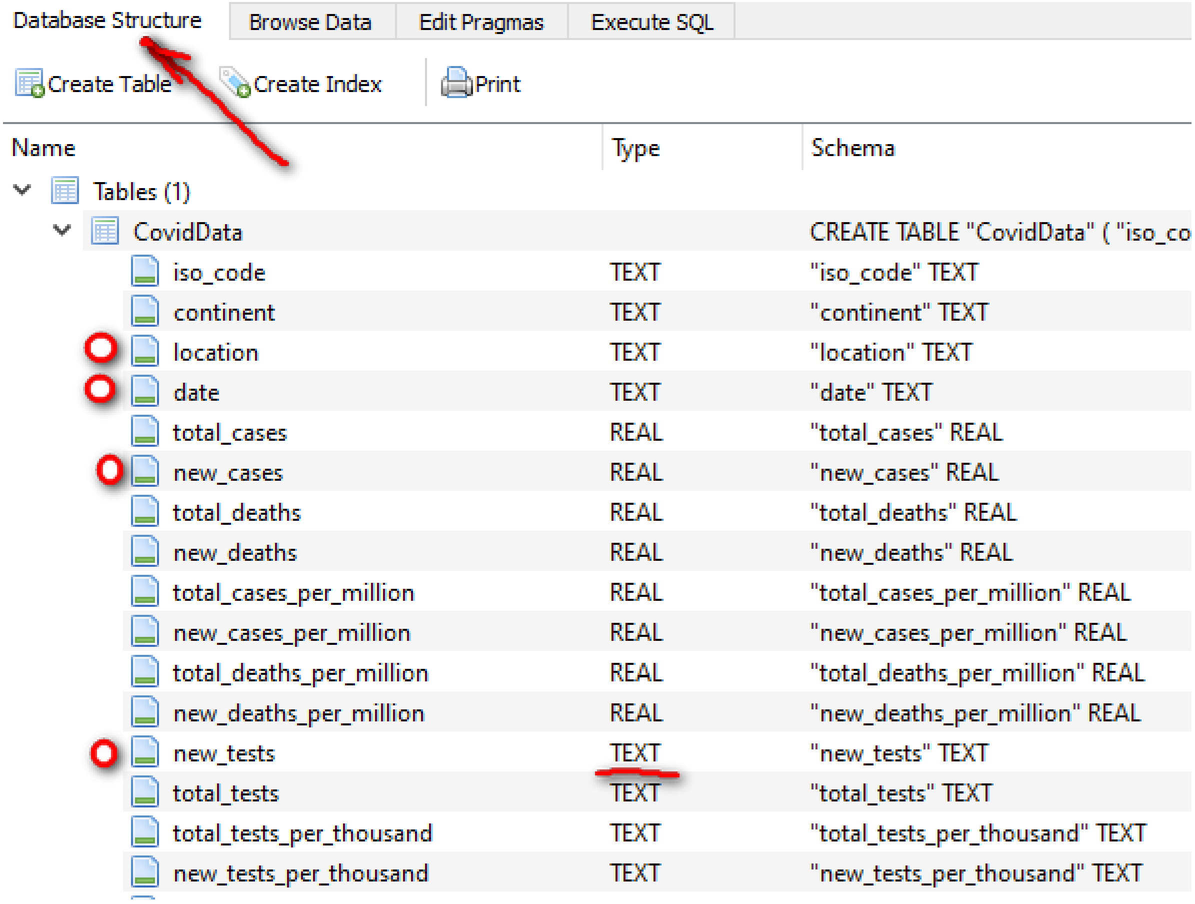
Data columns that we need are the following:

- location – country
- date – specific date
- new_cases – detected positive COVID-19 cases on a specific date
- new_tests – number of tests done on a specific date

You can also see that 37194 rows were returned by this query. That’s is length (number of records) in our dataset.

Let’s check the assigned data types for relevant columns to prevent potential misinterpretation of data due to improper conversion.

- Go to “Data Structure”, expand the “CovidData” table and you will see all columns listed.



Name	Type	Schema
Tables (1)		
CovidData		CREATE TABLE "CovidData" ("iso_co
iso_code	TEXT	"iso_code" TEXT
continent	TEXT	"continent" TEXT
location	TEXT	"location" TEXT
date	TEXT	"date" TEXT
total_cases	REAL	"total_cases" REAL
new_cases	REAL	"new_cases" REAL
total_deaths	REAL	"total_deaths" REAL
new_deaths	REAL	"new_deaths" REAL
total_cases_per_million	REAL	"total_cases_per_million" REAL
new_cases_per_million	REAL	"new_cases_per_million" REAL
total_deaths_per_million	REAL	"total_deaths_per_million" REAL
new_deaths_per_million	REAL	"new_deaths_per_million" REAL
new_tests	TEXT	"new_tests" TEXT
total_tests	TEXT	"total_tests" TEXT
total_tests_per_thousand	TEXT	"total_tests_per_thousand" TEXT
new_tests_per_thousand	TEXT	"new_tests_per_thousand" TEXT

- There is a problem with detected data types regarding the “new_tests” column. For our calculation, we need a numeric value. The reason for that incorrect assignment of data type could be NULL values for that specific column at the beginning of the data file. Also, the “new_cases” is probably the integer, but this is not an issue for or calculation. No problem, we can easily change data type by right-clicking on the “CovidData” table and selecting “Modify table” option as described above.

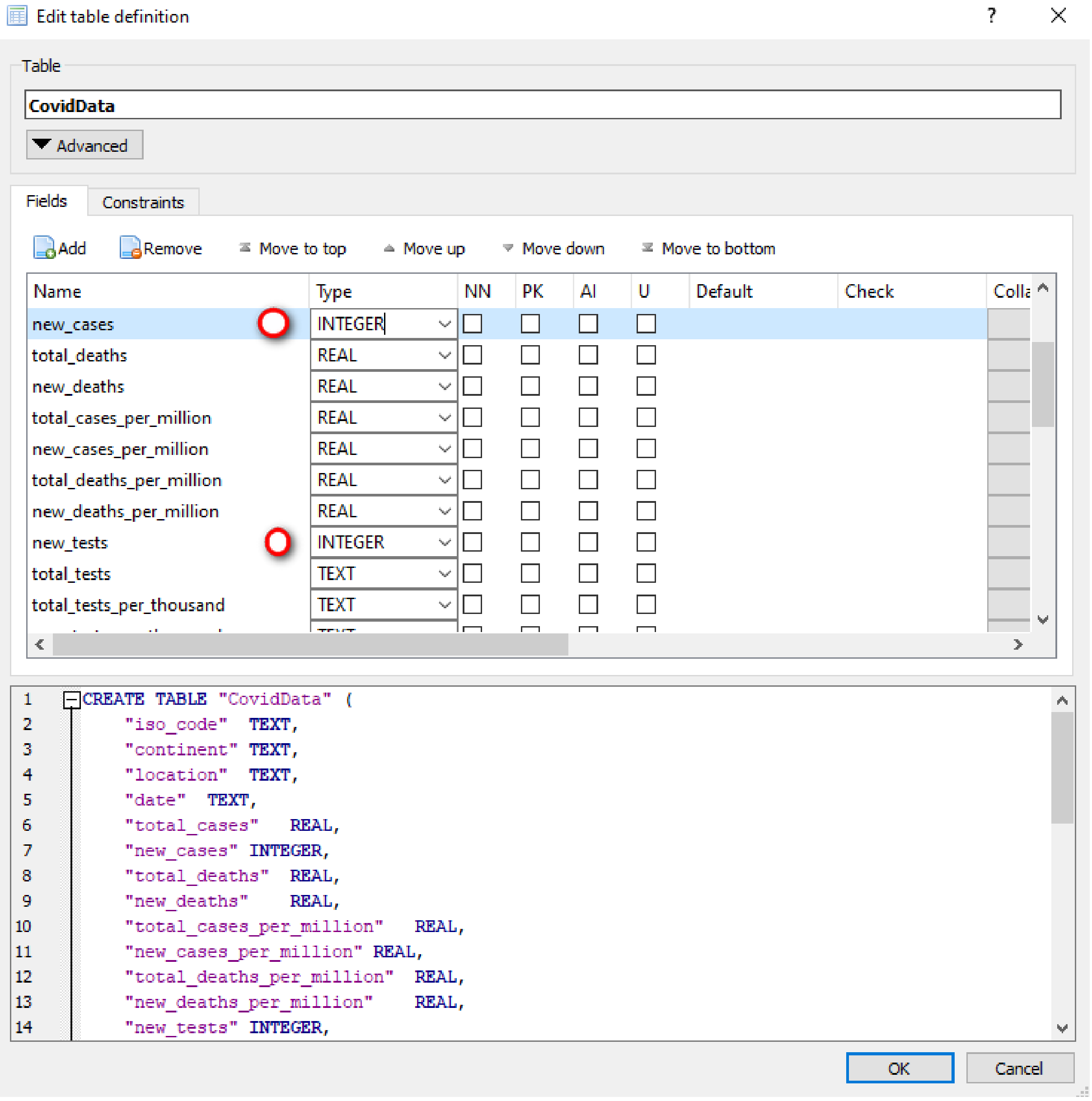
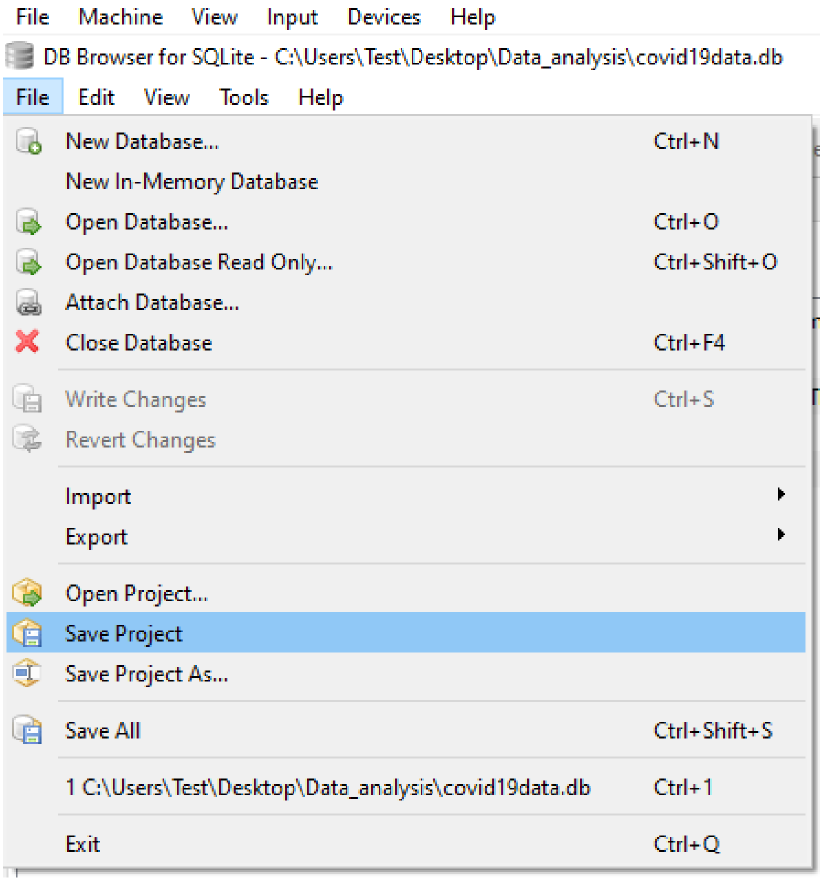


Table: CovidData

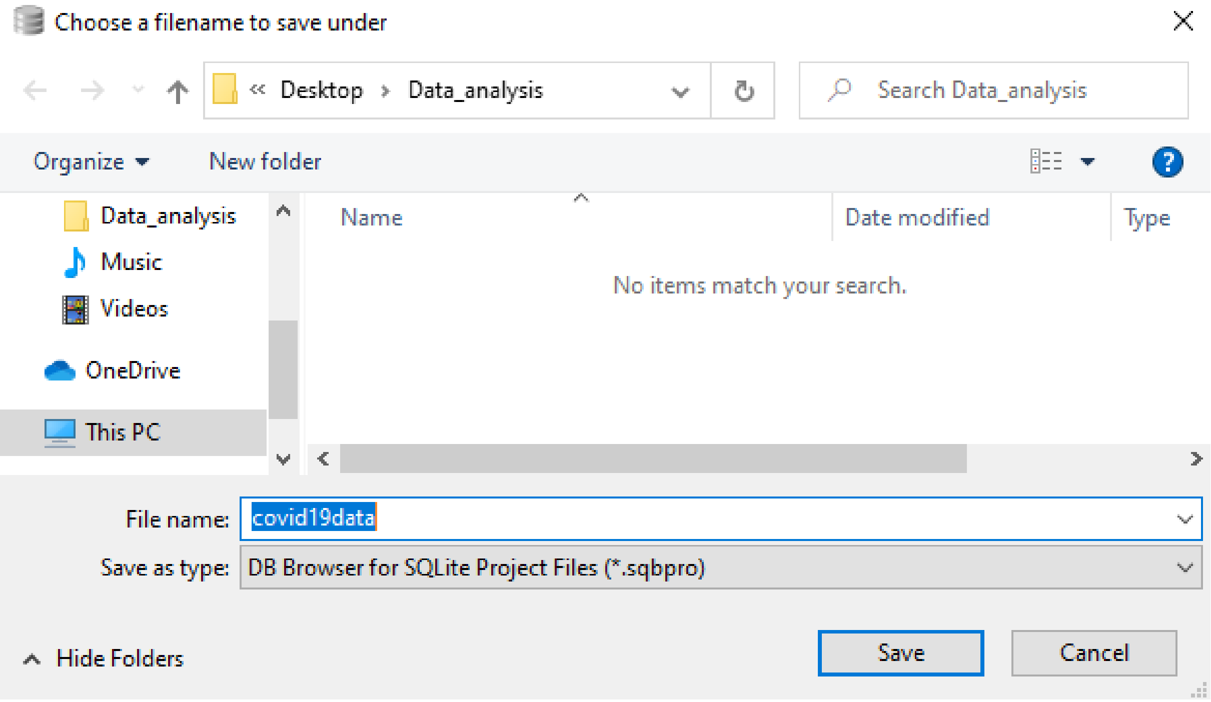
Name	Type	NN	PK	AI	U	Default	Check	Colla
new_cases	INTEGER	□	□	□	□			
total_deaths	REAL	□	□	□	□			
new_deaths	REAL	□	□	□	□			
total_cases_per_million	REAL	□	□	□	□			
new_cases_per_million	REAL	□	□	□	□			
total_deaths_per_million	REAL	□	□	□	□			
new_deaths_per_million	REAL	□	□	□	□			
new_tests	INTEGER	□	□	□	□			
total_tests	TEXT	□	□	□	□			
total_tests_per_thousand	TEXT	□	□	□	□			

```
CREATE TABLE "CovidData" (
    "iso_code"  TEXT,
    "continent" TEXT,
    "location"  TEXT,
    "date"  TEXT,
    "total_cases"   REAL,
    "new_cases" INTEGER,
    "total_deaths"  REAL,
    "new_deaths"    REAL,
    "total_cases_per_million"   REAL,
    "new_cases_per_million" REAL,
    "total_deaths_per_million"  REAL,
    "new_deaths_per_million"    REAL,
    "new_tests" INTEGER,
```

- Change data types for both columns to INTEGER and click “OK”. Now, is also a good time to save our project. Go to “File” -> “Save Project”.


File Machine View Input Devices Help
DB Browser for SQLite - C:\Users\Test\Desktop\Data_analysis\covid19data.db
File Edit View Tools Help
New Database... Ctrl+N
New In-Memory Database
Open Database... Ctrl+O
Open Database Read Only... Ctrl+Shift+O
Attach Database...
Close Database Ctrl+F4
Write Changes Ctrl+S
Revert Changes
Import
Export
Open Project...
Save Project
Save Project As...
Save All Ctrl+Shift+S
1 C:\Users\Test\Desktop\Data_analysis\covid19data.db Ctrl+1
Exit Ctrl+Q

- Save project file in which we will store SQL statements and configuration to the “Data_analysis” directory. You can use “covid19data” as a name.



No items match your search.

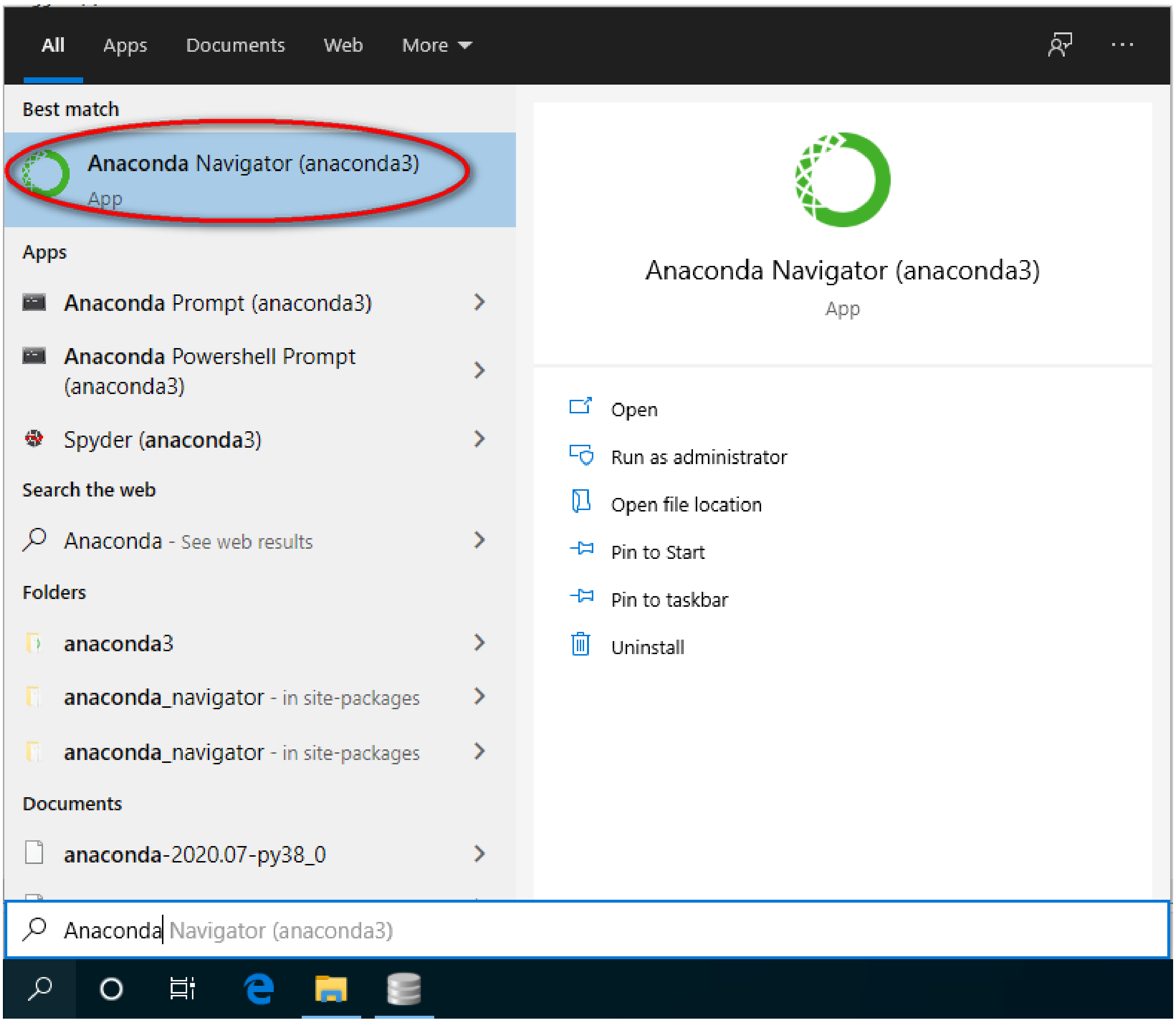
File name: covid19data

Save as type: DB Browser for SQLite Project Files (*.sqbpro)

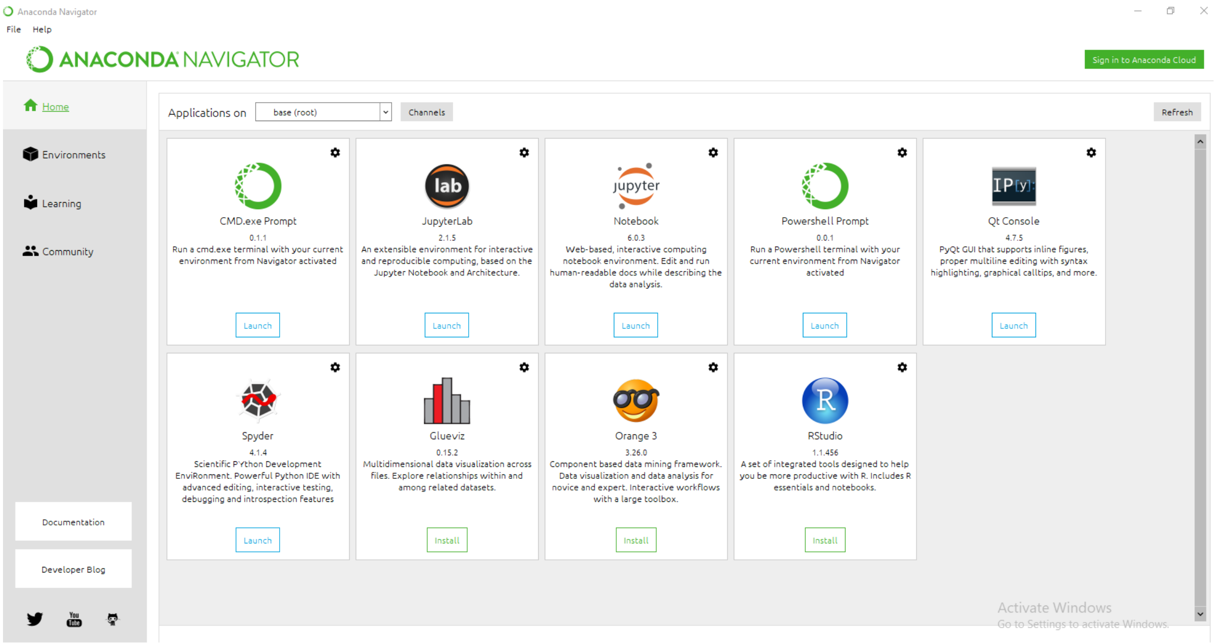
- Go back to the “Execute SQL” to start the next data analytics phase. 😊

b) Load data to the Python Pandas environment

- Use windows search to find “Anaconda Navigator” and start it. This can take some time so wait patiently for everything to load.



All
Apps
Documents
Web
More
Best match
Anaconda Navigator (anaconda3)
App
Apps
Anaconda Prompt (anaconda3)
Anaconda Powershell Prompt (anaconda3)
Spyder (anaconda3)
Search the web
Anaconda - See web results
Folders
anaconda3
anaconda_navigator - in site-packages
anaconda_navigator - in site-packages
Documents
anaconda-2020.07-py38_0
Anaconda Navigator (anaconda3)
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
Uninstall
Anaconda Navigator (anaconda3)



CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
2.1.5
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.0.3
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

Qt Console
4.7.5
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

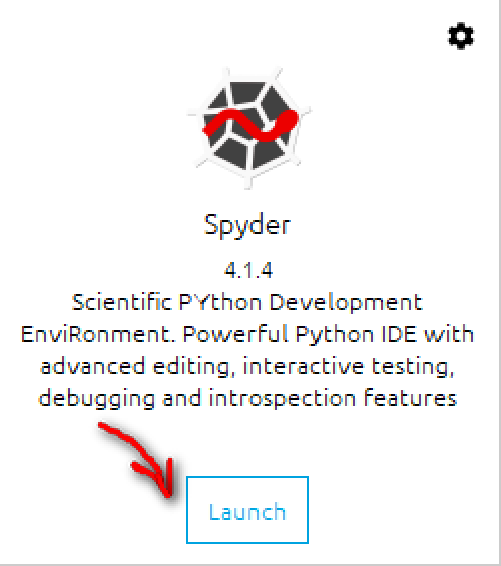
Spyder
4.1.4
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Glueviz
0.15.2
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.26.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

- On the main “Anaconda Navigator” window choose “Spyder” and select “Launch” to load Python development environment.

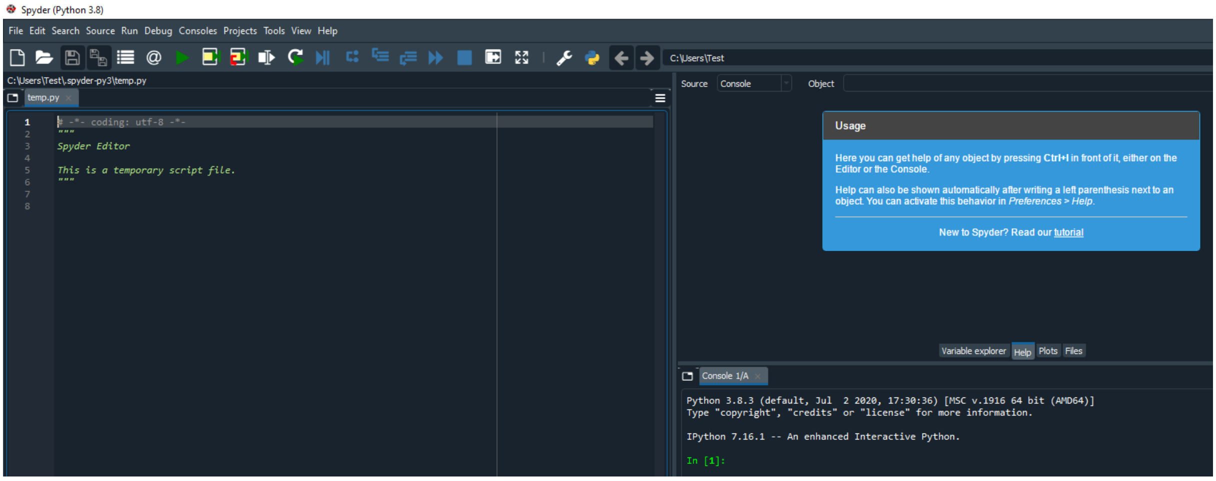


Spyder

4.1.4

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

- When the “Spyder” editor is loaded save the empty Python file at the location of your “Data_analysis” directory.



Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Test
C:\Users\Test\.spyder-py3\temp.py
temp.py
Source
Console
Object
Variable explorer
Help
Plots
Files
Console 1/A

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
```

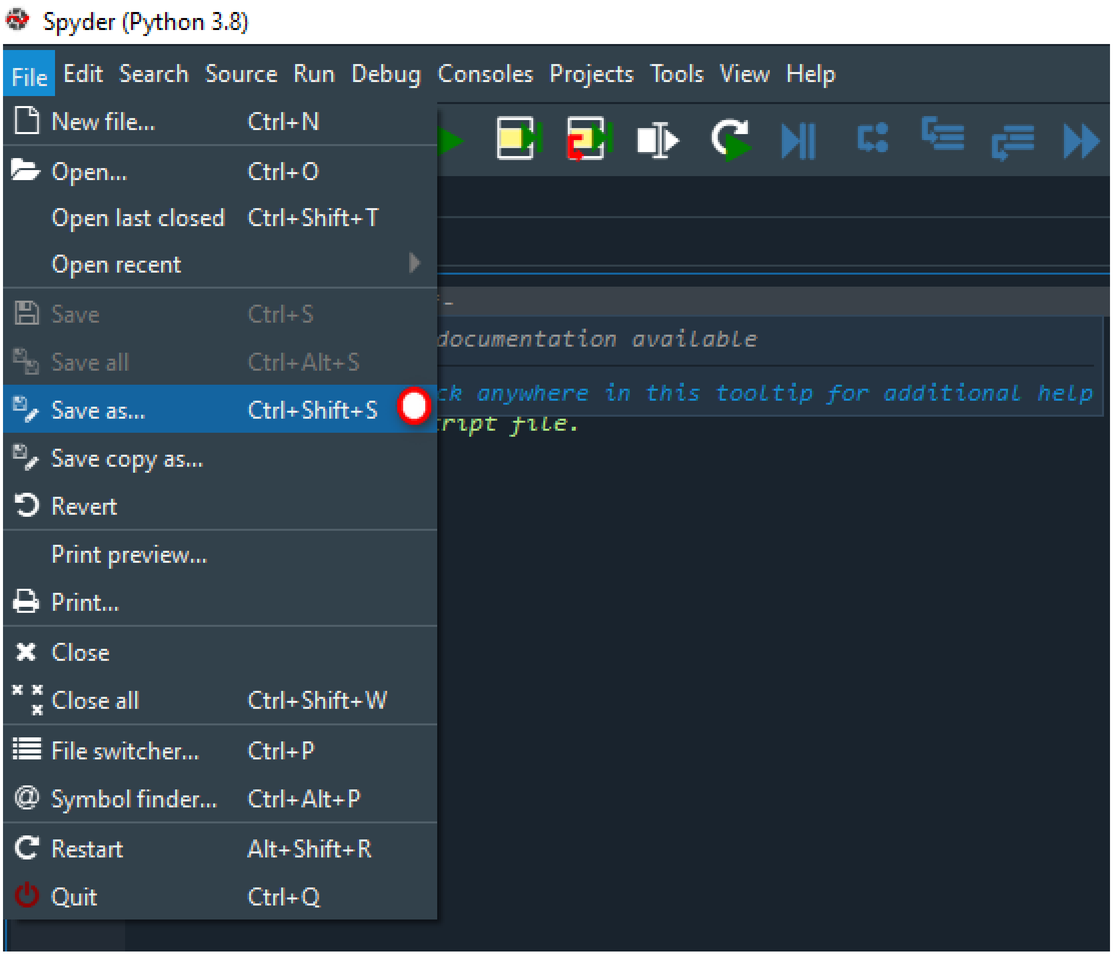
Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

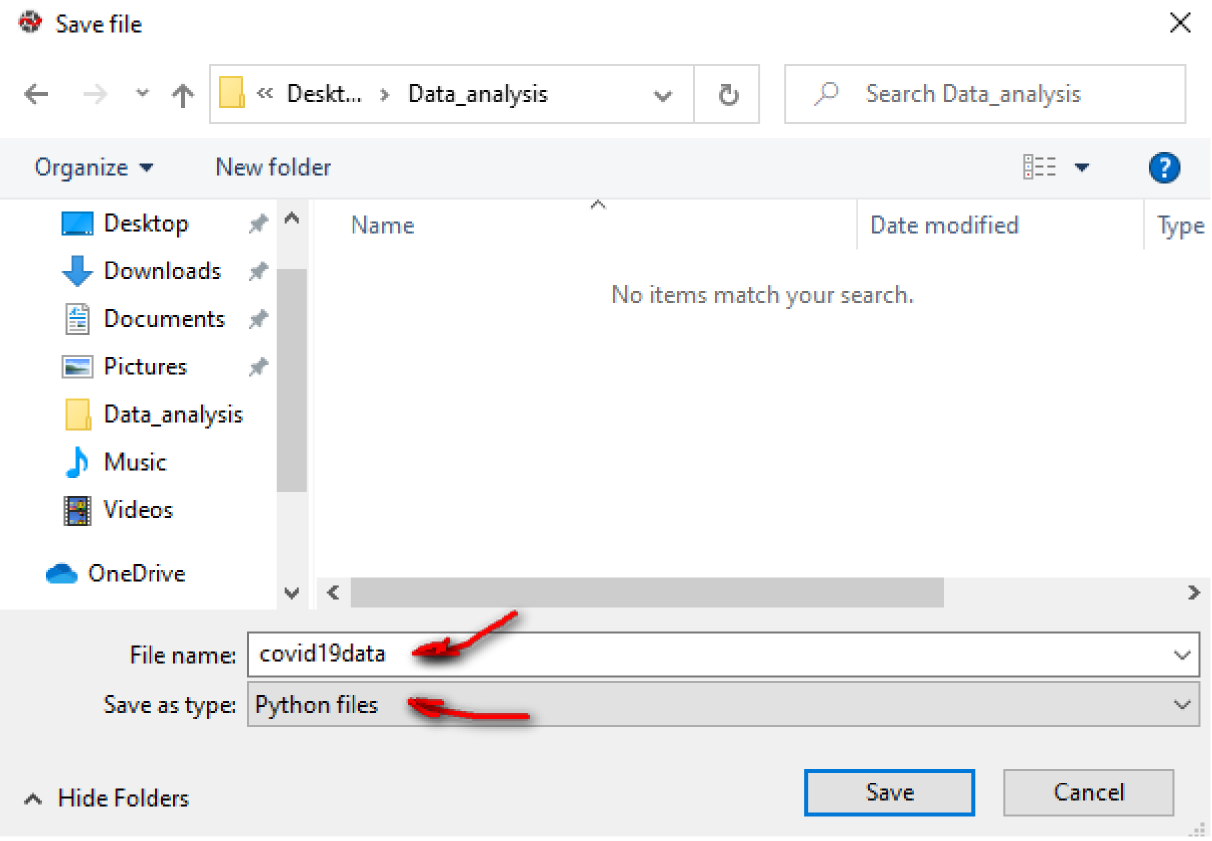
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]:
```



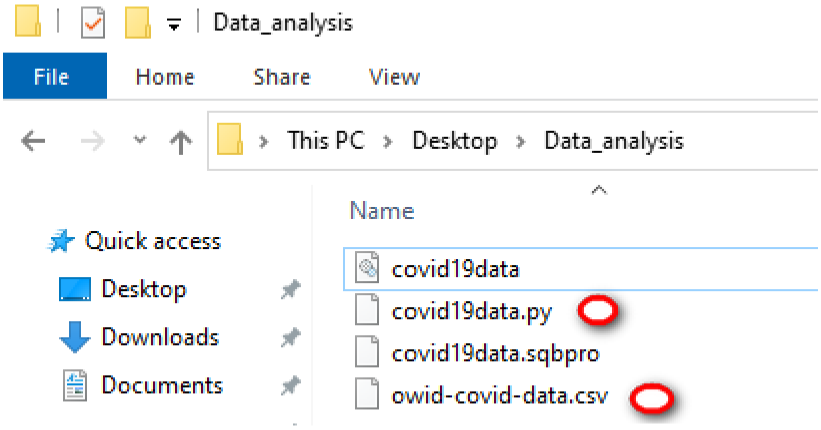
- Provide a name for the file ("covid19data.py") and select "Python files" for type. Click the "Save" button.



File name: covid19data

Save as type: Python files

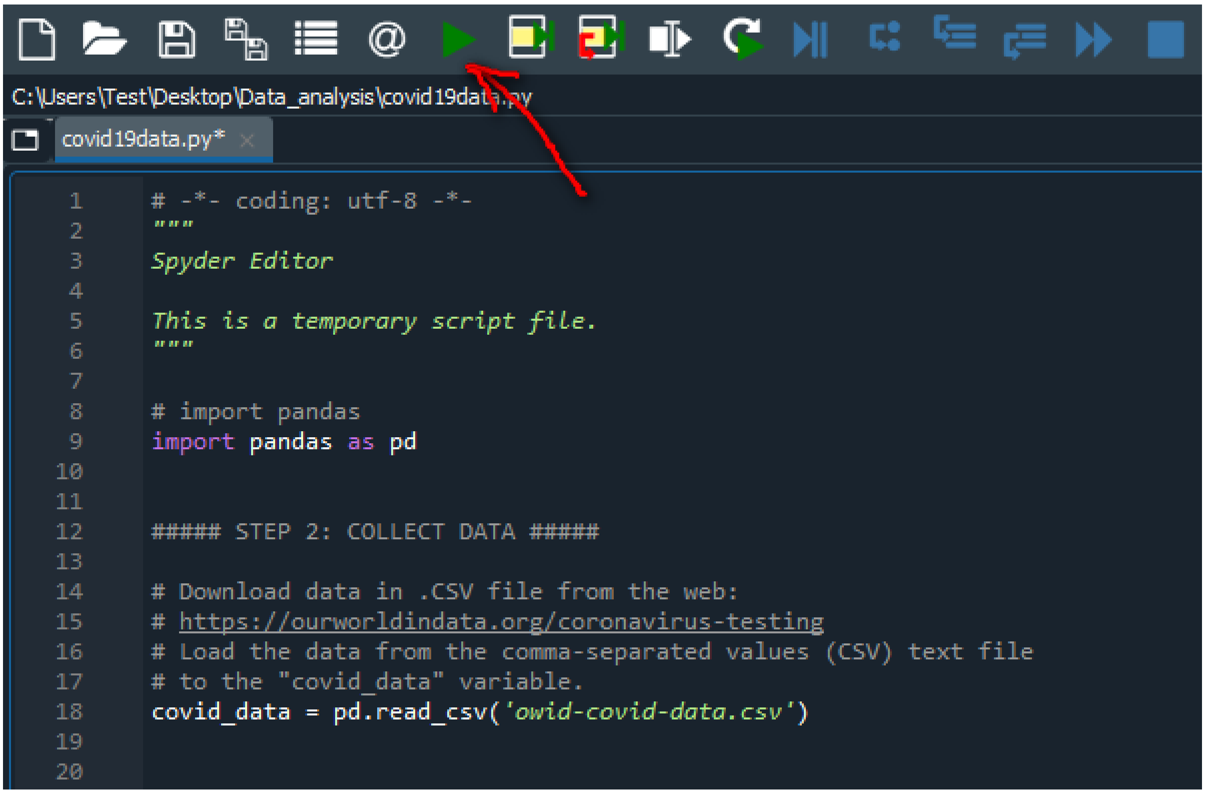
Note: it critical that you place Python script file “covid19data.py” and sample data .CSV file “owid-covid-data.csv” on the same location on your file system (e.g. “Data_analysis” directory). Otherwise, you will have to specify the full path to sample data .CSV file “owid-covid-data.csv” in your Python code.



- Write or paste the following code in your “Spyder” editor and click green arrow (or F5 key) to execute code.

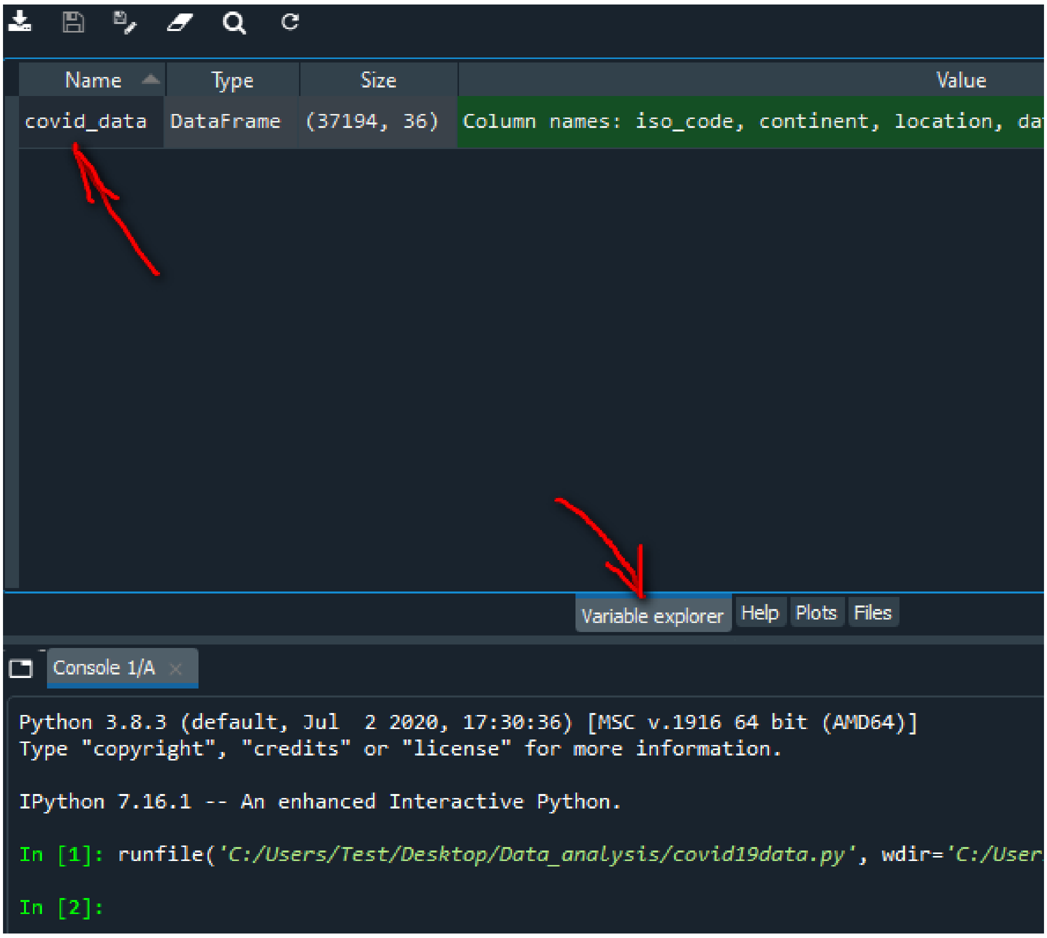
```
# import pandas
import pandas as pd

# Load the data from the comma-separated values (CSV) text file
# to the "covid_data" variable.
covid_data = pd.read_csv('owid-covid-data.csv')
```

Note: On the first line of the code, we imported "pandas" library that holds all you will need for data analytics.



```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

# import pandas
import pandas as pd


##### STEP 2: COLLECT DATA #####

# Download data in .CSV file from the web:
# https://ourworldindata.org/coronavirus-testing
# Load the data from the comma-separated values (CSV) text file
# to the "covid_data" variable.
covid_data = pd.read_csv('owid-covid-data.csv')


```

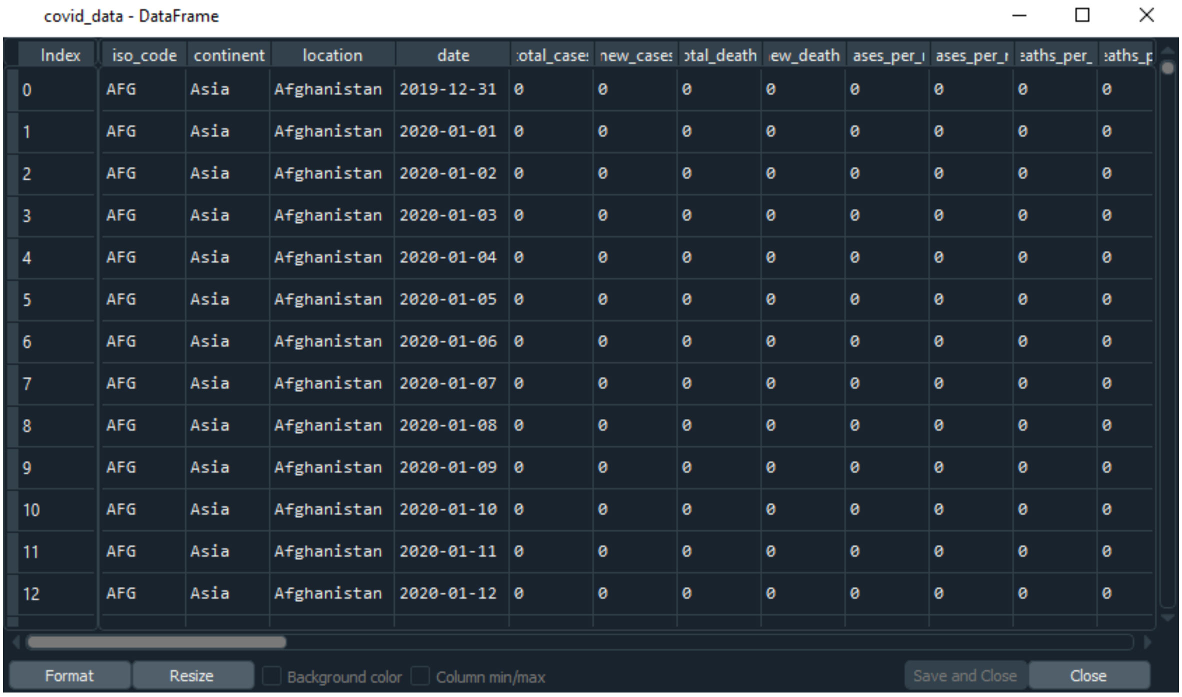
- All data from the sample data .CSV file “owid-covid-data.csv” is loaded to the “covid_data” variable.



Name	Type	Size	Value
covid_data	DataFrame	(37194, 36)	Column names: iso_code, continent, location, da

```
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/Test/Desktop/Data_analysis/covid19data.py', wdir='C:/User

In [2]:
```

- Click on the “Variable explorer” on the right side of the “Spyder” editor window and you will see the “covid_data” variable listed. Click on the “covid_data” variable and window with data will be displayed.



covid_data - DataFrame

Index	iso_code	continent	location	date	:otal_case:	new_case:	:otal_death	ew_death	ases_per_ı	ases_per_ı	:aths_per_	:aths_p
0	AFG	Asia	Afghanistan	2019-12-31	0	0	0	0	0	0	0	0
1	AFG	Asia	Afghanistan	2020-01-01	0	0	0	0	0	0	0	0
2	AFG	Asia	Afghanistan	2020-01-02	0	0	0	0	0	0	0	0
3	AFG	Asia	Afghanistan	2020-01-03	0	0	0	0	0	0	0	0
4	AFG	Asia	Afghanistan	2020-01-04	0	0	0	0	0	0	0	0
5	AFG	Asia	Afghanistan	2020-01-05	0	0	0	0	0	0	0	0
6	AFG	Asia	Afghanistan	2020-01-06	0	0	0	0	0	0	0	0
7	AFG	Asia	Afghanistan	2020-01-07	0	0	0	0	0	0	0	0
8	AFG	Asia	Afghanistan	2020-01-08	0	0	0	0	0	0	0	0
9	AFG	Asia	Afghanistan	2020-01-09	0	0	0	0	0	0	0	0
10	AFG	Asia	Afghanistan	2020-01-10	0	0	0	0	0	0	0	0
11	AFG	Asia	Afghanistan	2020-01-11	0	0	0	0	0	0	0	0
12	AFG	Asia	Afghanistan	2020-01-12	0	0	0	0	0	0	0	0

Great! Data collection and load to analytical tools are finished.

On top of this, you now know how to execute SQL and Python code to get results. We will utilize what we learned in the next section, where we will deal with code and data.

Step 4: Prepare data

In this section we will do some queries both with SQL and Python Pandas in order to check data, clean it, and prepare it for analysis.

This will be done in the following way:

- Python Pandas code will be executed for data extraction or data analytics
- and then checked with the corresponding SQL statement used for data extraction or data analytics.

In real life situations you can, of course, use only one language for your analytical purposes. However, this is a training course and I think is very important to use both SQL and Python Pandas simultaneously to compare tools and language syntax.

Note: As mentioned in the introduction, we will not go in-depth with any of the programming languages. There are a lot of tutorials and free resources online that are dealing with SQL and Python syntax, and I encourage you to use them during this tutorial or later. However, that would take time depending on your previous experience, and this would extend this data analytic education to the level of details that could be too technical. My idea is to explore the steps in the data analysis, look into tools and their usage, and when this was done, you will decide to look under the hood or not to see how it works from a detailed technical side. In other words, the complexity of the system on the low level should not be a stopper of using tools. That is, in a nutshell, what is meant with the "naive" approach in education.

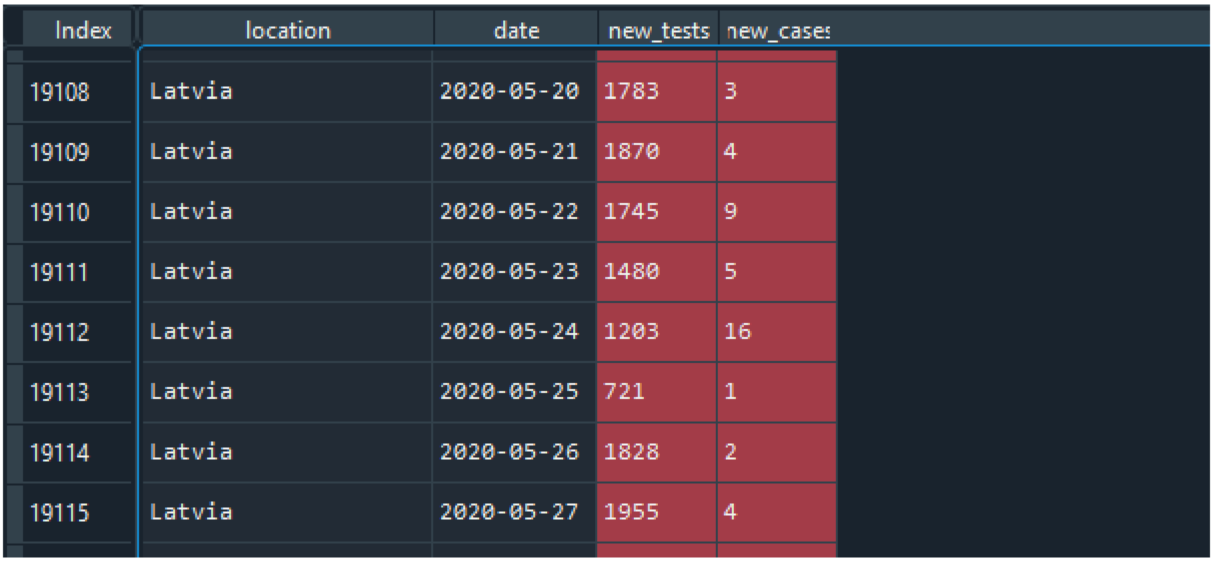
a) Check and remove NULL (empty) values

With the targeted query we will extract only those data that we would require to fulfill our analytical goal. In the initial analysis above we decided that we only need the following data: “location”, “date”, “new_cases”, “new_tests”.

- In the Python (“Spyder” editor) execute the following code:

```
# Make a copy of dataset to use it for data preparation.
ds = covid_data.copy()

# We identified the following three columns that are needed for our analysis.
covid_ds = ds[['location', 'date', 'new_tests', 'new_cases']]
```

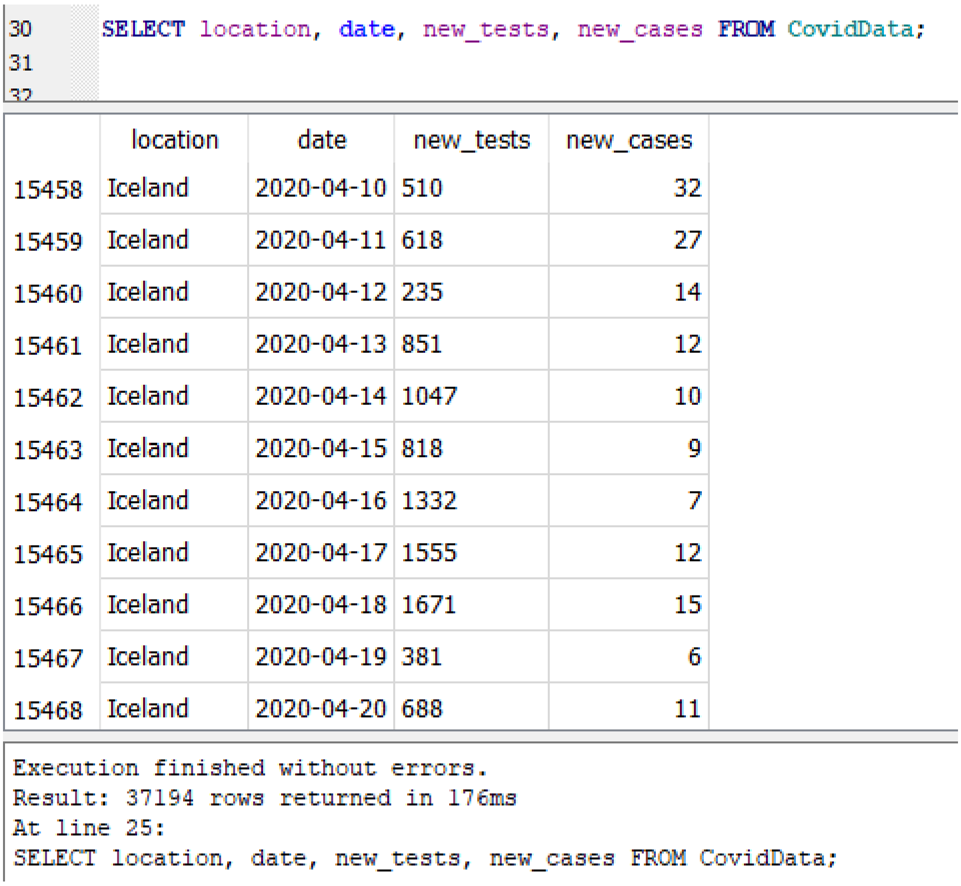
covid_ds - DataFrame



Index	location	date	new_tests	new_cases
19108	Latvia	2020-05-20	1783	3
19109	Latvia	2020-05-21	1870	4
19110	Latvia	2020-05-22	1745	9
19111	Latvia	2020-05-23	1480	5
19112	Latvia	2020-05-24	1203	16
19113	Latvia	2020-05-25	721	1
19114	Latvia	2020-05-26	1828	2
19115	Latvia	2020-05-27	1955	4

- In the SQLite execute the following statement to get the same result:

```
SELECT location, date, new_tests, new_cases FROM CovidData;
```



```
SELECT location, date, new_tests, new_cases FROM CovidData;
```

	location	date	new_tests	new_cases
15458	Iceland	2020-04-10	510	32
15459	Iceland	2020-04-11	618	27
15460	Iceland	2020-04-12	235	14
15461	Iceland	2020-04-13	851	12
15462	Iceland	2020-04-14	1047	10
15463	Iceland	2020-04-15	818	9
15464	Iceland	2020-04-16	1332	7
15465	Iceland	2020-04-17	1555	12
15466	Iceland	2020-04-18	1671	15
15467	Iceland	2020-04-19	381	6
15468	Iceland	2020-04-20	688	11

```
Execution finished without errors.
Result: 37194 rows returned in 176ms
At line 25:
SELECT location, date, new_tests, new_cases FROM CovidData;
```

Let's do a simple check of the quality of data concerning number of records and the number of missing values.

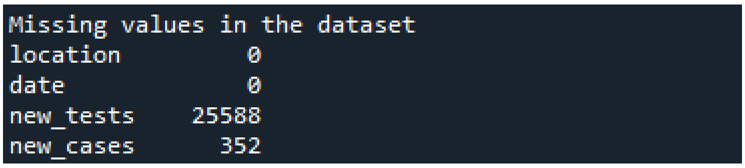
- In the Python (“Spyder” editor) execute the following code:

```
number_of_records1 = covid_ds.count()\
                .reset_index(name='count')\
                .sort_values(['count'], ascending=False)
```

We noticed a different number of records which implies the presence of a significant number of NULL values or as Python calls it "nan" values. This must be interpreted further with a nice Python line of code.

- Let's identify the columns with missing values along with the count and print it on the console.

```
print ('')
print ('Missing values in the dataset')
print (covid_ds.isnull().sum(axis=0))
```



```
Missing values in the dataset
location         0
date             0
new_tests    25588
new_cases      352
```

- We will get the same result if we execute the following SQL statement in the SQLite:

```
SELECT COUNT(*) FROM CovidData WHERE location IS NULL;
-- no NULL entry found

SELECT COUNT(*) FROM CovidData WHERE date IS NULL;
-- no NULL entry found

SELECT COUNT(*) FROM CovidData WHERE new_tests IS NULL;
-- there are 25588 rows where test data for the specific date is missing

SELECT COUNT(*) FROM CovidData WHERE new_cases IS NULL;
-- there are 352 rows where test data for the specific date is missing
```

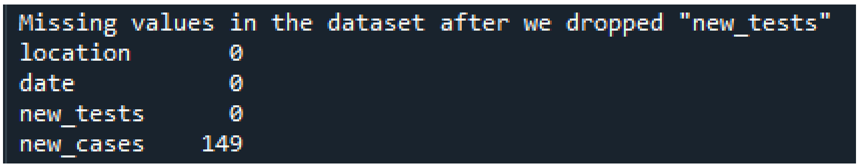
The good news is that the "location" and "date" have no missing values.

The bad news is that the "new_tests" column only 11606 records out of 37194 records have value entry.

DECISION: it is not possible to perform analysis without "new_tests" value. We will remove all rows where the "new_tests" value is missing.

- Execute the following Python code to drop NULL values for the “new_tests” column and print the resulting status after this action:

```
covid_ds = covid_ds.dropna(subset = ["new_tests"])

print ('')
print ('Missing values in the dataset after we dropped \"new_tests\"')
print (covid_ds.isnull().sum(axis=0))
```



```
Missing values in the dataset after we dropped "new_tests"
location        0
date            0
new_tests       0
new_cases     149
```

Also, it is not good that the "new_cases" column has still missing values.

- Let's select records where the "new_cases" column has NaN.

```
new_cases_nan = covid_ds[covid_ds['new_cases'].isnull()]
```

Well, there are some discoveries:

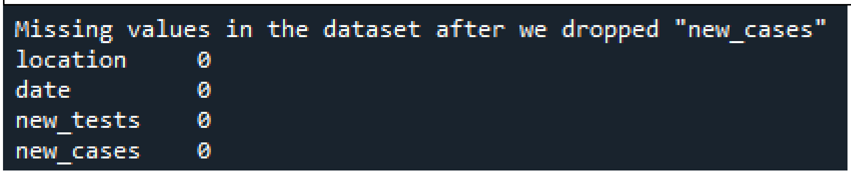
1) Data where "new_tests" are done and "new_cases" are missing are all in the period from 2020-02-06 to 2020-03-19. This was at the beginning of the pandemic, and this could be a test accuracy or reporting problem.
2) Also, we can see that the United Arab Emirates conducted 33555 tests in that period, and data of the "new_cases" are missing. We can say that those records from the United Arab Emirates are clear outliers that will significantly influence results for that country.

Note: Outliers are unusual values in the dataset that significantly vary from other data. Outliers are very problematic for many analyses because they can distort results and cause tests to either miss significant findings.

DECISION: We will remove all rows where the "new_cases" value is missing.

- Execute the following Python code to drop NULL values for the “new_cases” column and print the resulting status after this action:

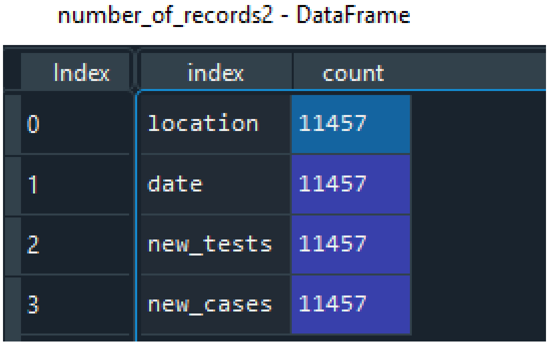
```
covid_ds = covid_ds.dropna(subset = ["new_cases"])
print ('')
print ('Missing values in the dataset after we dropped \"new_cases\"')
print (covid_ds.isnull().sum(axis=0))
```



```
Missing values in the dataset after we dropped "new_cases"
location      0
date          0
new_tests     0
new_cases     0
```

- Again, we will check the number of records for each column.

```
number_of_records2 = covid_ds.count()\
              .reset_index(name='count')\
              .sort_values(['count'], ascending=False)
```

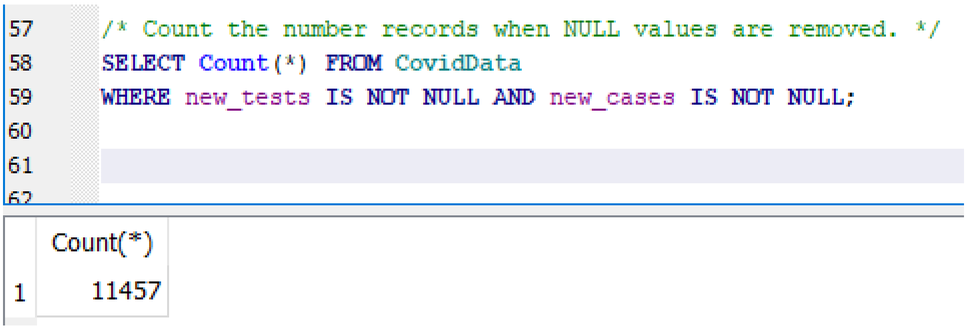


number_of_records2 - DataFrame

Index	index	count
0	location	11457
1	date	11457
2	new_tests	11457
3	new_cases	11457

- SQL statement that will reflect this elimination of NULL values would look like this:

```
/* Count the number records when NULL values are removed. */
SELECT Count(*) FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL;
--11457 records
```



```
/* Count the number records when NULL values are removed. */
SELECT Count(*) FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL;
```

	Count(*)
1	11457

Great! There are now 11457 records without missing values.

b) Check and remove inconsistent and error values

We will now check for possible records with cases where there are more new positive cases for Covid-19 than tests in a single day. This will not make sense for our analysis.

- Execute the following Python code to find those cases:

```
temp1_before = covid_ds\
    .loc[covid_ds['new_cases'] > covid_ds['new_tests']]
```

Unfortunately, we have found such cases. However, this is only for the 65 records. It is also worrying that we have seen negative values. This situation should be an alarm to the analyst. Perhaps, the data collection was not adequate, and it may be best to do a new collection or reach for some other data sources.

DECISION: We will remove all rows where the "new_cases" > "new_tests".

- Execute the following Python code to drop all rows where the "new_cases" > "new_tests":

```
covid_ds.drop(covid_ds[covid_ds['new_cases'] > covid_ds['new_tests']]\
        .index, inplace = True)
temp1_after = covid_ds.loc[covid_ds['new_cases'] > covid_ds['new_tests']]
```

Also, the same number of tests and positive cases could be unrealistic since 100% accuracy of the test is not expected. This scenario is most likely an error.

- Execute the following Python code to find those cases:

```
temp2_before = covid_ds\
    .loc[covid_ds['new_cases'] == covid_ds['new_tests']]
```

There are six records found! The inspection of data shows that it is probably an error.

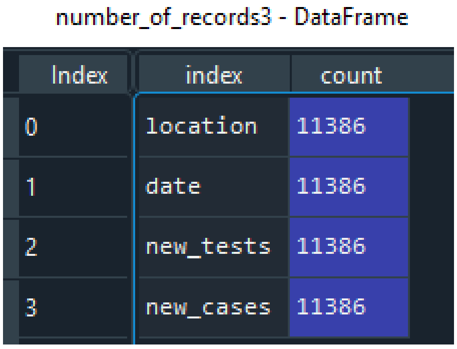
DECISION: We will remove all rows where the "new_cases" == "new_tests".

- Execute the following Python code to drop all rows where the "new_cases" == "new_tests".

```
covid_ds.drop(covid_ds[covid_ds['new_cases'] == covid_ds['new_tests']]\
        .index, inplace = True)
temp2_after = covid_ds\
    .loc[covid_ds['new_cases'] == covid_ds['new_tests']]
```

- Again, we will check the number of records for each column.

```
number_of_records3 = covid_ds.count()\
            .reset_index(name='count')\
            .sort_values(['count'], ascending=False)
```

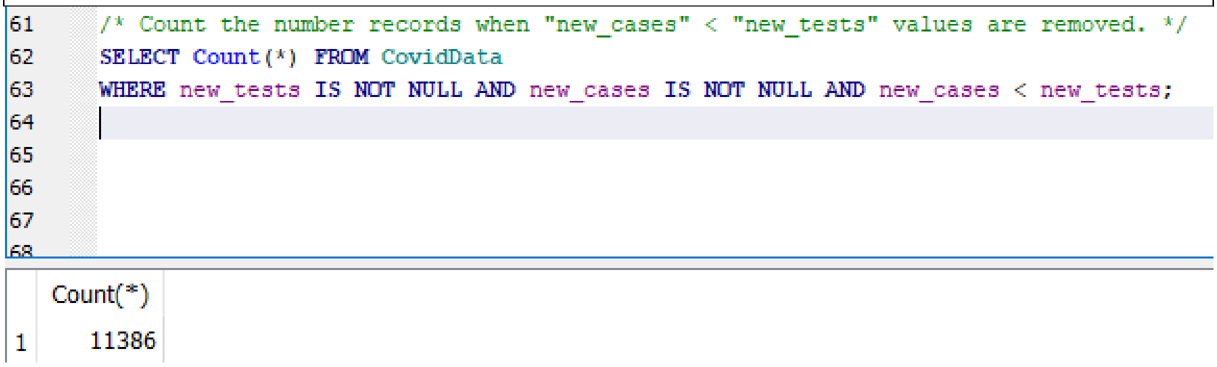


number_of_records3 - DataFrame

Index	index	count
0	location	11386
1	date	11386
2	new_tests	11386
3	new_cases	11386

- SQL statement that will reflect elimination of errors will look like this:

```
SELECT Count(*) FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL AND new_cases < new_tests;
```



```
/* Count the number records when "new_cases" < "new_tests" values are removed. */
SELECT Count(*) FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL AND new_cases < new_tests;
```

	Count(*)
1	11386

Now, we will also check for all negative values for "new_tests" and "new_cases." Those values must be removed for our dataset since it is evident that we are dealing with error entries.

- Execute the following Python code to get all negative entries for the "new_tests" column.

```
temp3 = covid_ds.loc[covid_ds['new_tests'] < 0 ]
```

No action required since there are no records where "new_tests" have a negative value number.

- Execute the following Python code to get all negative entries for the "new_cases" column.

```
temp4_before = covid_ds.loc[covid_ds['new_cases'] < 0]
```

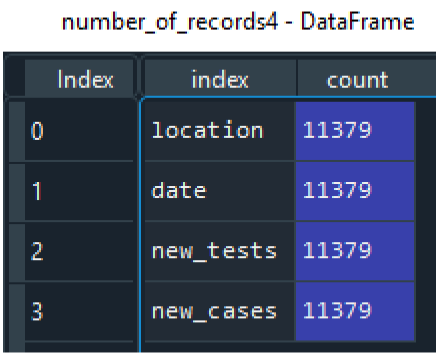
There are 7 records where "new_cases" column has negative value number.

- Execute the following Python code to remove all negative entries for the "new_cases" column.

```
covid_ds.drop(covid_ds[covid_ds['new_cases'] < 0]\
          .index, inplace = True)
temp4_after = covid_ds.loc[covid_ds['new_cases'] < 0]
```

Again, we will check the number of records for each column.

```
number_of_records4 = covid_ds.count()\
              .reset_index(name='count')\
              .sort_values(['count'], ascending=False)
```

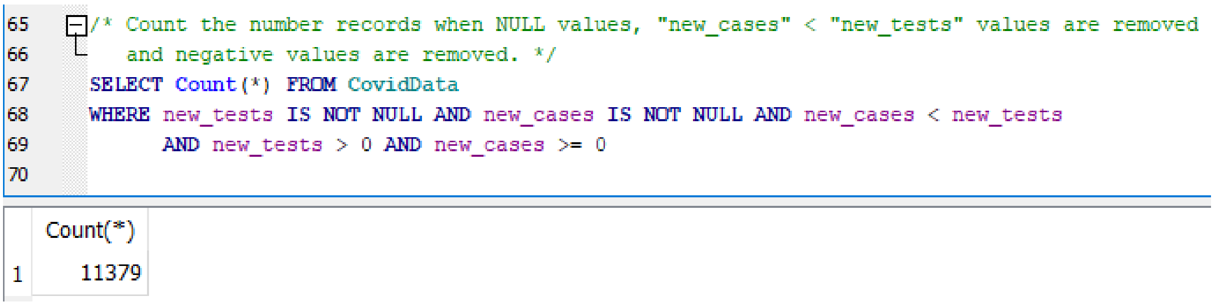


number_of_records4 - DataFrame

Index	index	count
0	location	11379
1	date	11379
2	new_tests	11379
3	new_cases	11379

- SQL statement that will reflect elimination of negative values will look like this:

```
SELECT Count(*) FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL AND new_cases < new_tests AND
new_tests > 0 AND new_cases >= 0
```



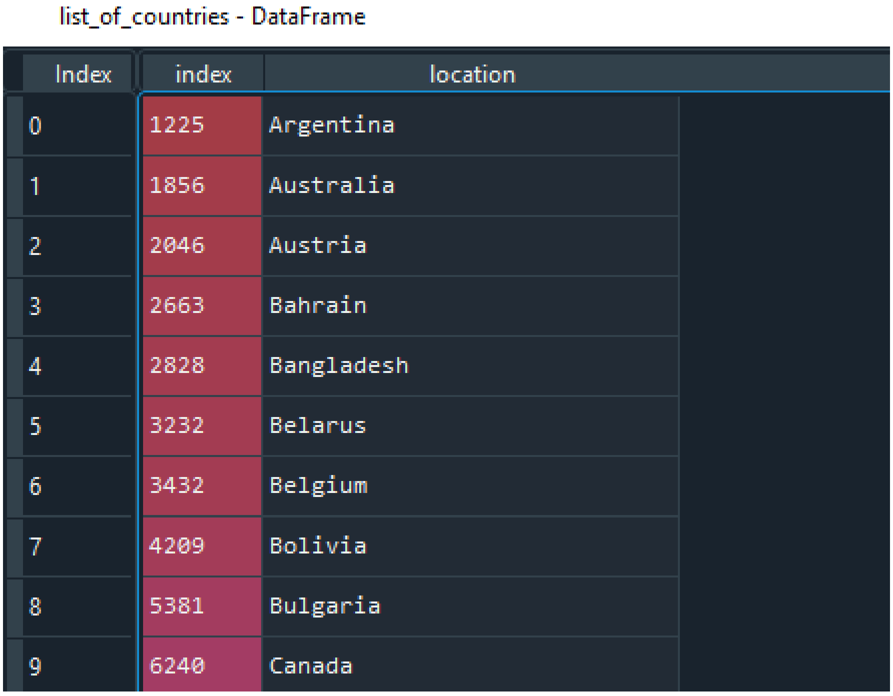
```
/* Count the number records when NULL values, "new_cases" < "new_tests" values are removed
   and negative values are removed. */
SELECT Count(*) FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL AND new_cases < new_tests
      AND new_tests > 0 AND new_cases >= 0
```

	Count(*)
1	11379

- Let's see a list of countries to verify entries by executing following Python code.

```
list_of_countries = covid_ds.drop_duplicates('location')[['location']]\
                        .reset_index()
```

Note: The reset_index() is used to add new index column starting with 0 for the resulting dataset. Usage is optional and if not used, original index entries remain, and in some cases that could be useful to link record back to original data.



list_of_countries - DataFrame

Index	index	location
0	1225	Argentina
1	1856	Australia
2	2046	Austria
3	2663	Bahrain
4	2828	Bangladesh
5	3232	Belarus
6	3432	Belgium
7	4209	Bolivia
8	5381	Bulgaria
9	6240	Canada

- Corresponding SQL statement would look like this:

```
SELECT DISTINCT location AS Country FROM CovidData ORDER BY location;
```

We are left with 86 countries for the analysis after data cleaning activity.

This ends the "data preparation" step, which is most usually the most time consuming and complicated part of the analytical process. Now "covid_ds" dataset is after adjustments and data cleaning prepared for the analyses. As mentioned above, it is not the perfect data source, and its use would probably be questionable regarding data quality, but for this training, it will be just fine.

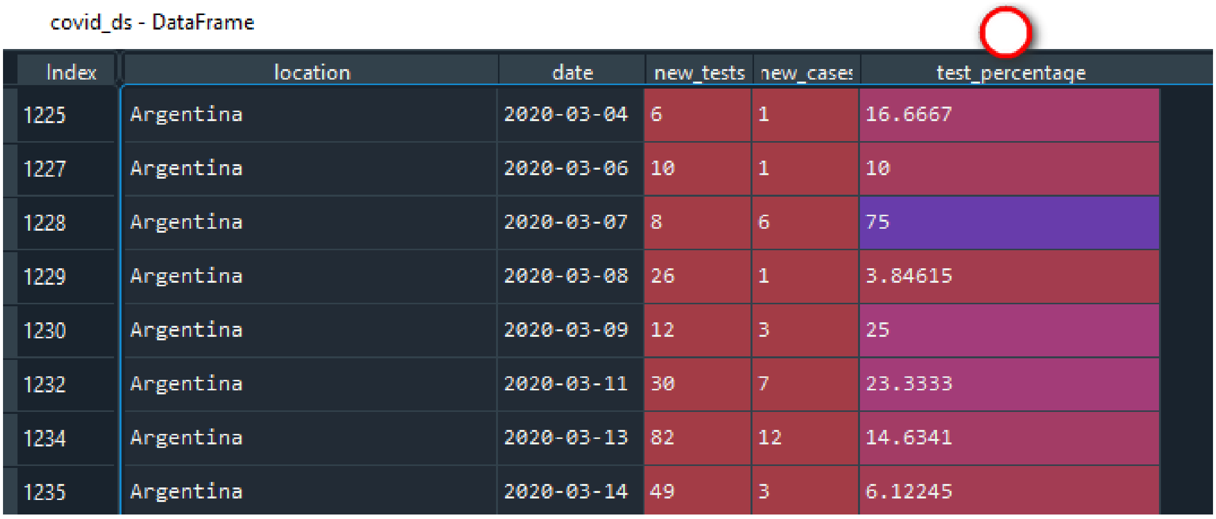
Step 5: Analyze data

Now, is time for the main show! Soon we will see the fruits of our labor. 😊

We will add an additional calculated column ("test_percentage") where the percentage of the ratio between Covid-19 test cases and tests that are done in the specific day will be stored.

- Execute the following Python code to add the calculated column:

```
covid_ds['test_percentage'] = covid_ds['new_cases'] / covid_ds['new_tests'] * 100
```



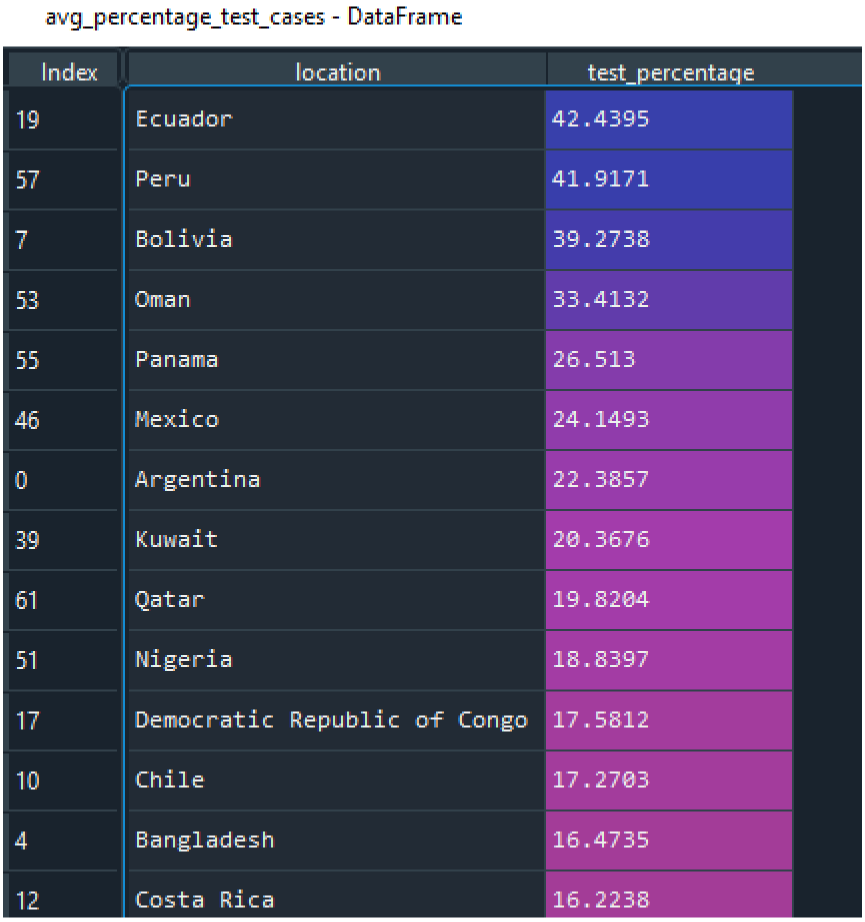
covid_ds - DataFrame

Index	location	date	new_tests	new_cases	test_percentage
1225	Argentina	2020-03-04	6	1	16.6667
1227	Argentina	2020-03-06	10	1	10
1228	Argentina	2020-03-07	8	6	75
1229	Argentina	2020-03-08	26	1	3.84615
1230	Argentina	2020-03-09	12	3	25
1232	Argentina	2020-03-11	30	7	23.3333
1234	Argentina	2020-03-13	82	12	14.6341
1235	Argentina	2020-03-14	49	3	6.12245

Now it is time approach to the goal of our analysis and list average percentages grouped by countries.

- Execute the following Python code get results for our analysis on full data sample:

```
avg_percentage_test_cases =\
        covid_ds.groupby(['location'])['test_percentage']\
        .mean()\
        .reset_index()\
        .sort_values(['test_percentage'], ascending=False)
```

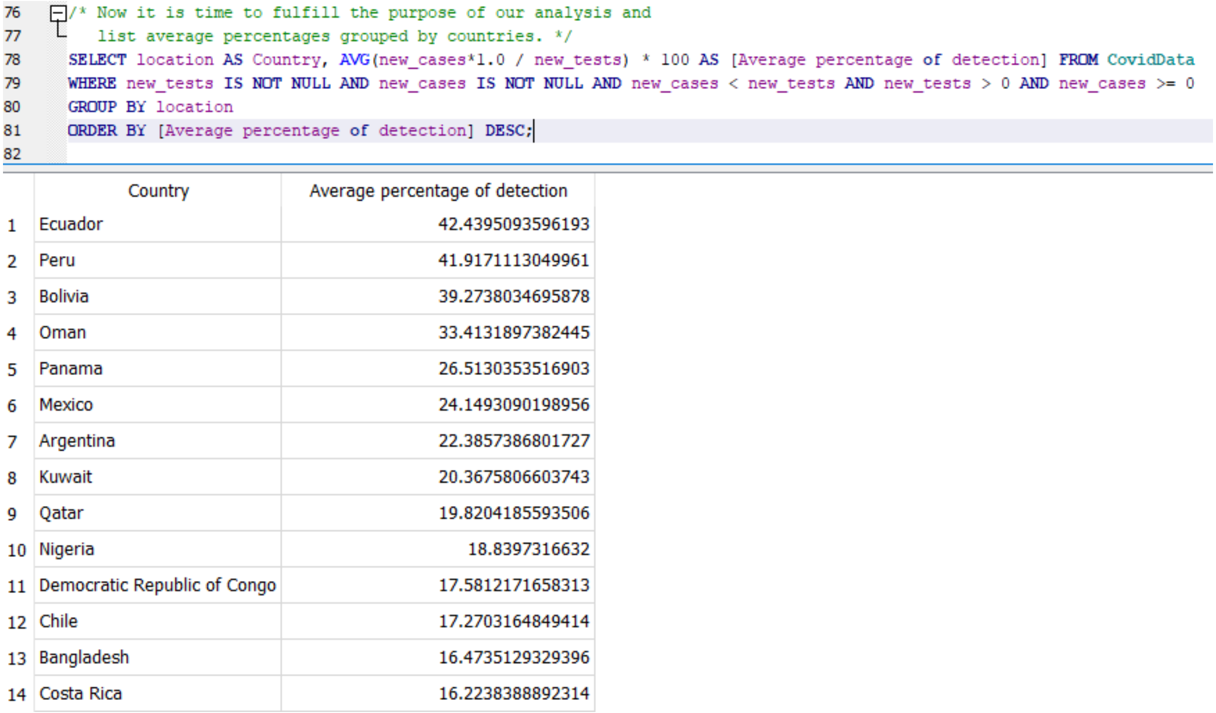


avg_percentage_test_cases - DataFrame

Index	location	test_percentage
19	Ecuador	42.4395
57	Peru	41.9171
7	Bolivia	39.2738
53	Oman	33.4132
55	Panama	26.513
46	Mexico	24.1493
0	Argentina	22.3857
39	Kuwait	20.3676
61	Qatar	19.8204
51	Nigeria	18.8397
17	Democratic Republic of Congo	17.5812
10	Chile	17.2703
4	Bangladesh	16.4735
12	Costa Rica	16.2238

- Execute the SQL statement get same results for our analysis on full data sample:

```
SELECT location AS Country, AVG(new_cases*1.0 / new_tests) * 100 AS [Average percentage of
detection] FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL AND new_cases < new_tests AND
new_tests > 0 AND new_cases >= 0
GROUP BY location
ORDER BY [Average percentage of detection] DESC;
```



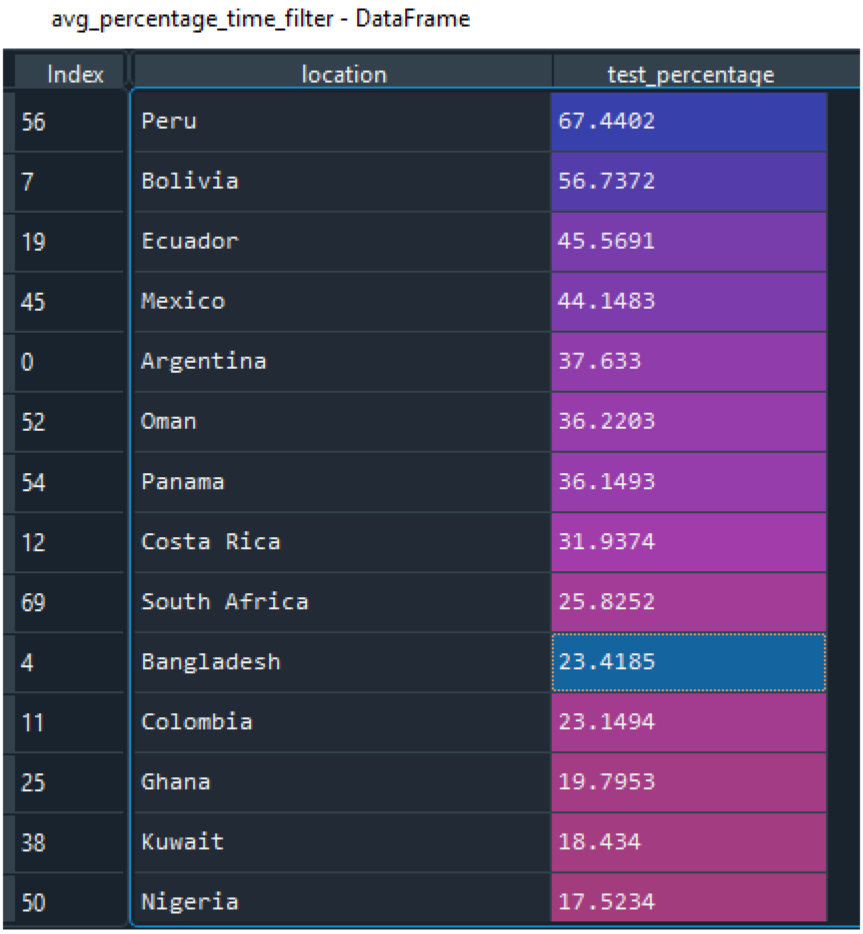
```
/* Now it is time to fulfill the purpose of our analysis and
   list average percentages grouped by countries. */
SELECT location AS Country, AVG(new_cases*1.0 / new_tests) * 100 AS [Average percentage of detection] FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL AND new_cases < new_tests AND new_tests > 0 AND new_cases >= 0
GROUP BY location
ORDER BY [Average percentage of detection] DESC;
```

	Country	Average percentage of detection
1	Ecuador	42.4395093596193
2	Peru	41.9171113049961
3	Bolivia	39.2738034695878
4	Oman	33.4131897382445
5	Panama	26.5130353516903
6	Mexico	24.1493090198956
7	Argentina	22.3857386801727
8	Kuwait	20.3675806603743
9	Qatar	19.8204185593506
10	Nigeria	18.8397316632
11	Democratic Republic of Congo	17.5812171658313
12	Chile	17.2703164849414
13	Bangladesh	16.4735129329396
14	Costa Rica	16.2238388892314

We can also give a time dimension to our query by providing date values to analyze data only in the specific period.

- Execute the following Python code get results for the specific time period:

```
covid_ds_time_filter = covid_ds\
    .loc[(covid_ds['date'] >= '2020-07-01') & (covid_ds['date'] <= '2020-08-01')]

avg_percentage_time_filter =\
      covid_ds_time_filter.groupby(['location'])['test_percentage']\
      .mean()\
      .reset_index()\
      .sort_values(['test_percentage'], ascending=False)
```



avg_percentage_time_filter - DataFrame

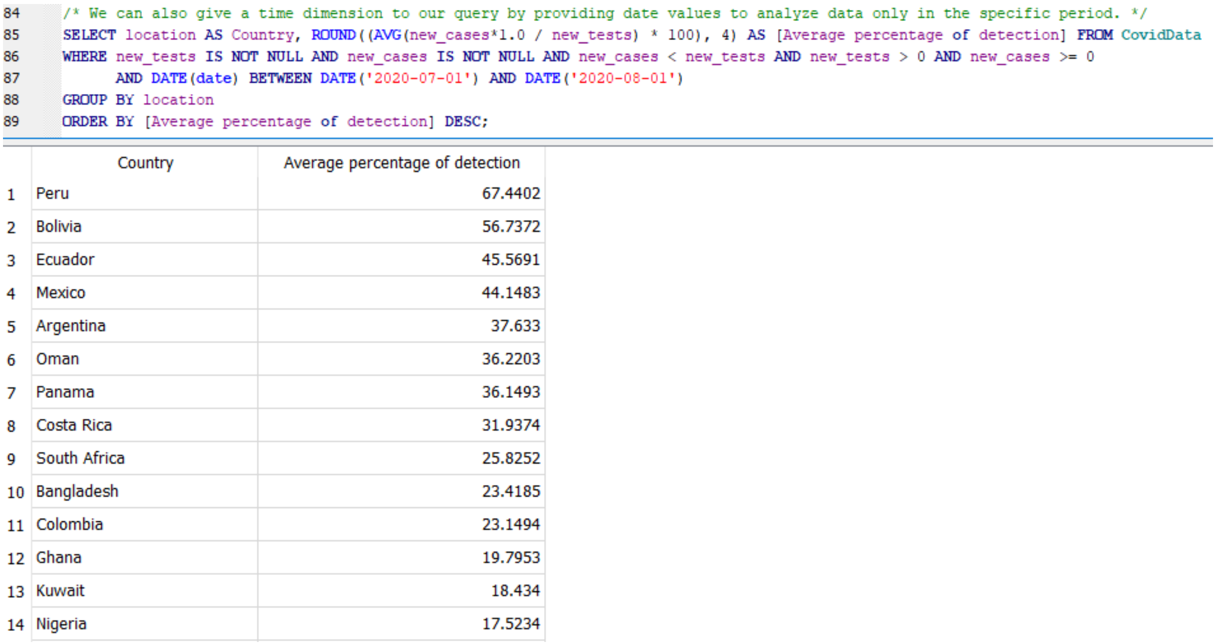
Index	location	test_percentage
56	Peru	67.4402
7	Bolivia	56.7372
19	Ecuador	45.5691
45	Mexico	44.1483
0	Argentina	37.633
52	Oman	36.2203
54	Panama	36.1493
12	Costa Rica	31.9374
69	South Africa	25.8252
4	Bangladesh	23.4185
11	Colombia	23.1494
25	Ghana	19.7953
38	Kuwait	18.434
50	Nigeria	17.5234

- Execute the SQL statement get same results for the specific time period:

```
SELECT location AS Country, ROUND((AVG(new_cases*1.0 / new_tests) * 100), 4) AS [Average
percentage of detection] FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL AND new_cases < new_tests AND
new_tests > 0 AND new_cases >= 0
    AND DATE(date) BETWEEN DATE('2020-07-01') AND DATE('2020-08-01')
GROUP BY location
ORDER BY [Average percentage of detection] DESC;
```

Note: The BETWEEN operator is inclusive. It returns true when the test_expression is less than or equal to high_expression and greater than or equal to the value of low_expression: test_expression >= low_expression AND test_expression <= high_expression.

Also, note the use of a ROUND function to display results with 4 decimal places.



```
/* We can also give a time dimension to our query by providing date values to analyze data only in the specific period. */
SELECT location AS Country, ROUND((AVG(new_cases*1.0 / new_tests) * 100), 4) AS [Average percentage of detection] FROM CovidData
WHERE new_tests IS NOT NULL AND new_cases IS NOT NULL AND new_cases < new_tests AND new_tests > 0 AND new_cases >= 0
      AND DATE(date) BETWEEN DATE('2020-07-01') AND DATE('2020-08-01')
GROUP BY location
ORDER BY [Average percentage of detection] DESC;
```

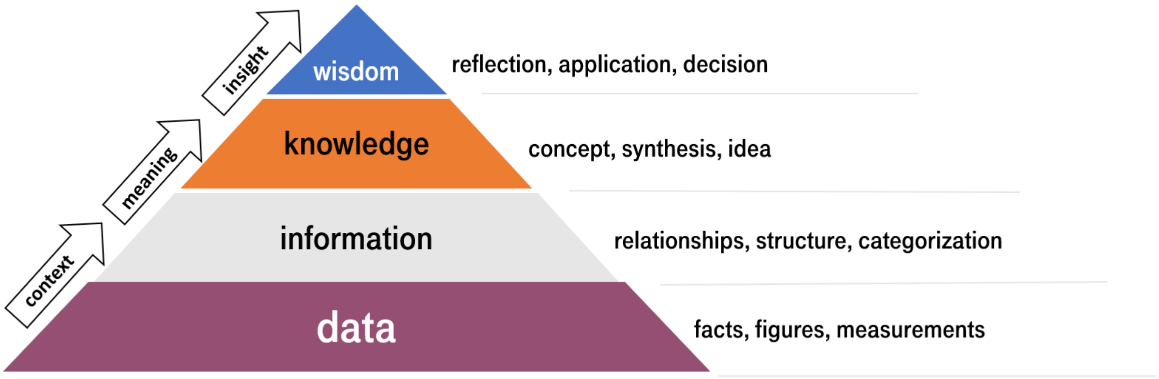
	Country	Average percentage of detection
1	Peru	67.4402
2	Bolivia	56.7372
3	Ecuador	45.5691
4	Mexico	44.1483
5	Argentina	37.633
6	Oman	36.2203
7	Panama	36.1493
8	Costa Rica	31.9374
9	South Africa	25.8252
10	Bangladesh	23.4185
11	Colombia	23.1494
12	Ghana	19.7953
13	Kuwait	18.434
14	Nigeria	17.5234

Step 6: Interpret results

The data analysis in the previous section provided results and we can now put them to real use. It is important to be always critical and do some reality check before making conclusions. As we mentioned there are concerns about the quality of data and the analysis itself could have errors. Therefore, ask yourself before proceeding any further, a simple question:

Does it make sense?

I will leave you with this question and let you decide. Remember that information is power, and as we gain more knowledge, we are becoming more wise human beings.


insight
meaning
context
wisdom
knowledge
information
data
reflection, application, decision
concept, synthesis, idea
relationships, structure, categorization
facts, figures, measurements

But it is not time to finish our lecture yet! Allow me to show you one very cool thing before we end today's session. 😊

How about some visualization of results like it is usual to do in Excel or similar applications?

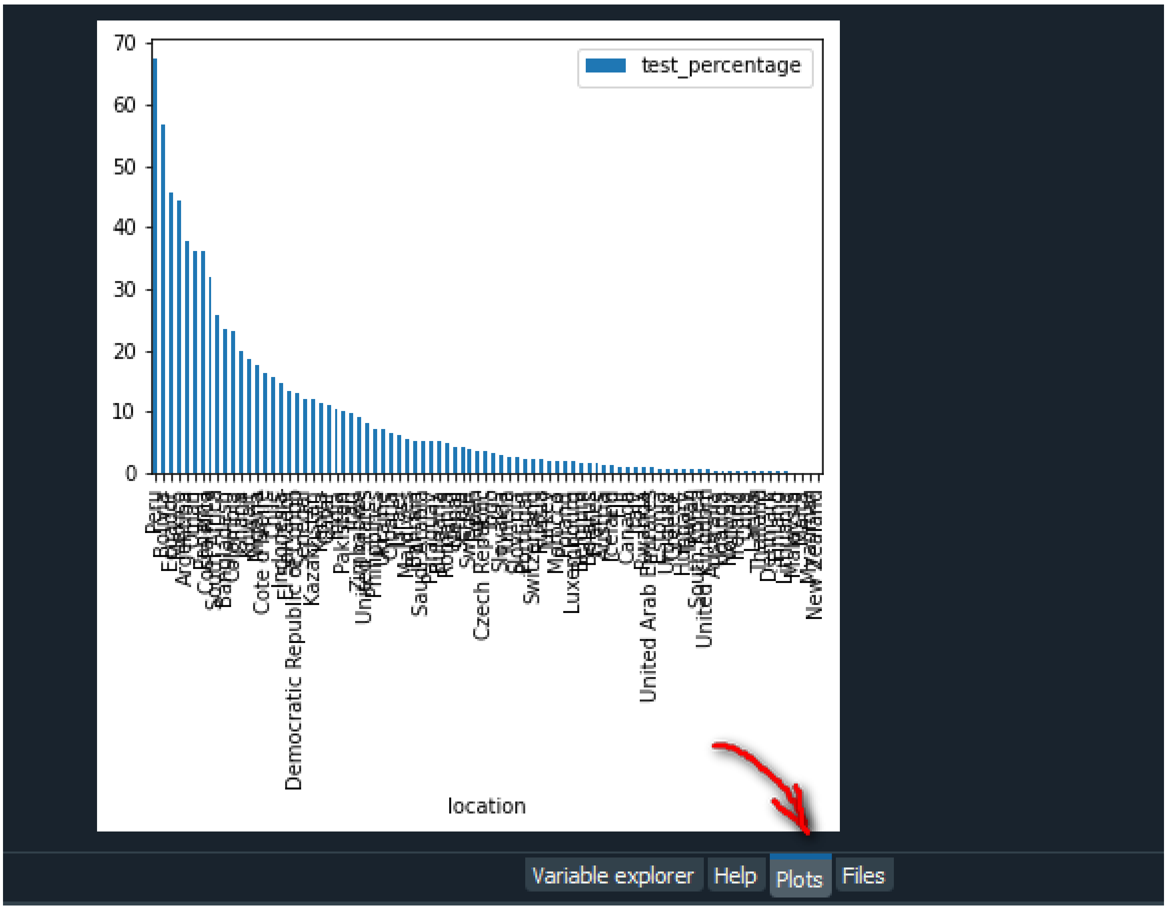
Python Pandas packages provide you with excellent resources for data visualization. We will use the "bar" chart type to show our results with a visual representation. This can be done right from our Python code.

- Input following code to push our result dataset towards visualization object and execute Python script:

```
# import libraries that will enable plotting of graphs
import matplotlib.pyplot as plt

# define x and y axes from the results dataset and select "bar" type graph
ax = avg_percentage_time_filter.plot(x ='location', y='test_percentage', kind = 'bar')

plt.show()
```

All data is there, but x-axes look little overcrowded with labels containing the country's names. With a little Python magic, we can rotate labels with country names by 90 degrees and display only every 3rd country.

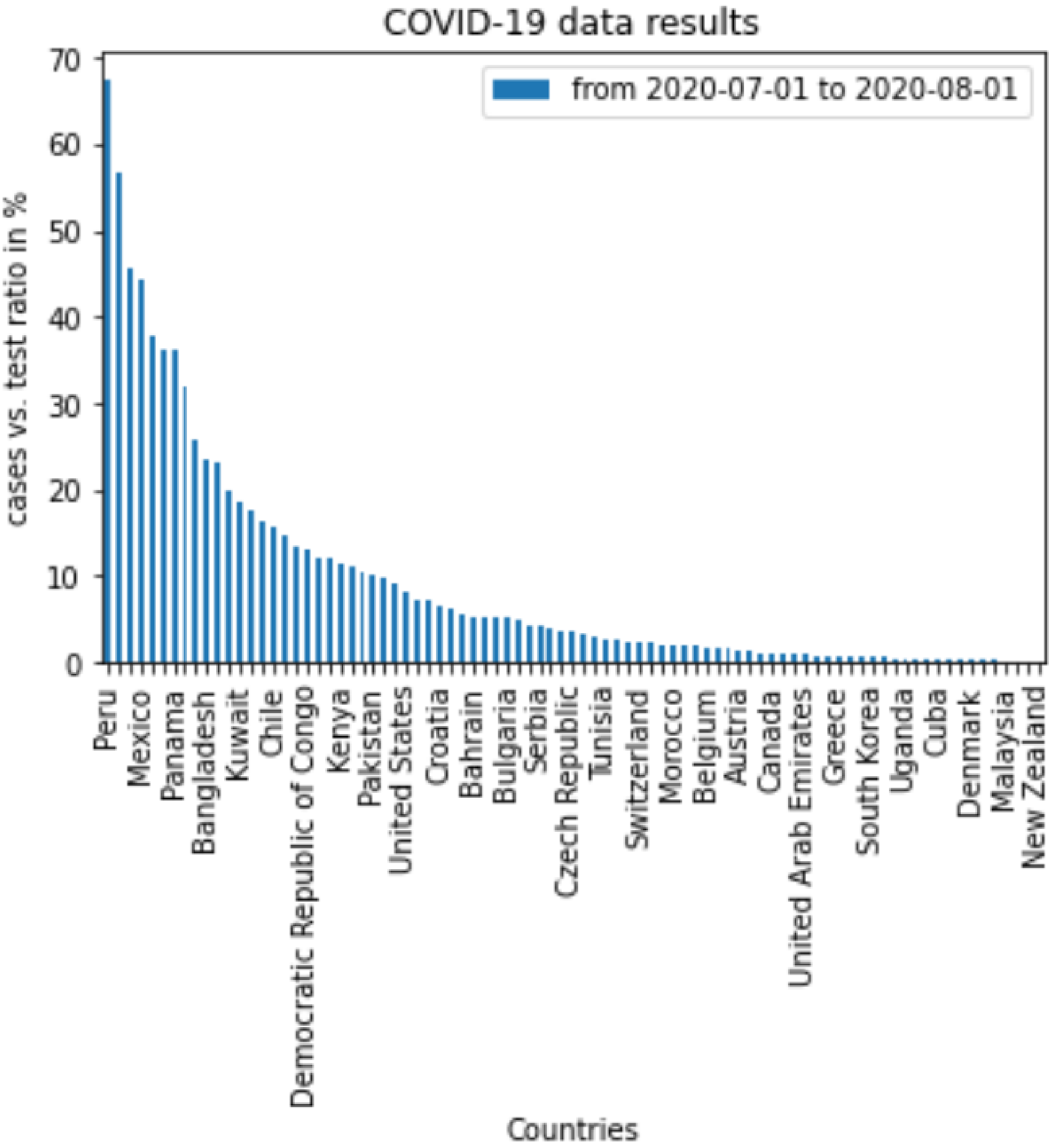
The graph is shown at the right side under the tab "Plots" on the right side of the "Spyder" editor window.



All data is there, but x-axes look little overcrowded with labels containing the country's names. With a little Python magic, and we can rotate labels with country names by 90 degrees and display only every 3rd country. And we can do even more customization by charging axes labels and title of the graph.

```
# import libraries that will enable plotting of graphs
import matplotlib.pyplot as plt

# define x and y axes from the results dataset and select "bar" type graph
ax = avg_percentage_time_filter.plot(x ='location', y='test_percentage', kind = 'bar')

# to avoid crowded labels on x-axes, we will show every third country and
# rotate labels by 90 degrees
for i, t in enumerate(ax.get_xticklabels()):
    if (i % 3) != 0:
        t.set_visible(False)

plt.xticks(rotation=90)

# additionally we can set chart title, legend and axes labels
plt.title('COVID-19 data results')
plt.legend(['from 2020-07-01 to 2020-08-01'])
plt.xlabel('Countries')
plt.ylabel('cases vs. test ratio in %')

plt.show()
```

Now, it looks much more readable and the graph can be saved or copied directly to your presentation.



There are a lot of ways to adjust and use visualization tools in Python, and I encourage you to explore further. This topic is broad and probably will require a separate and dedicated education session that would deal with visualization in detail. But for now, the provided example is enough to get an idea about the basic mechanics of this powerful feature.

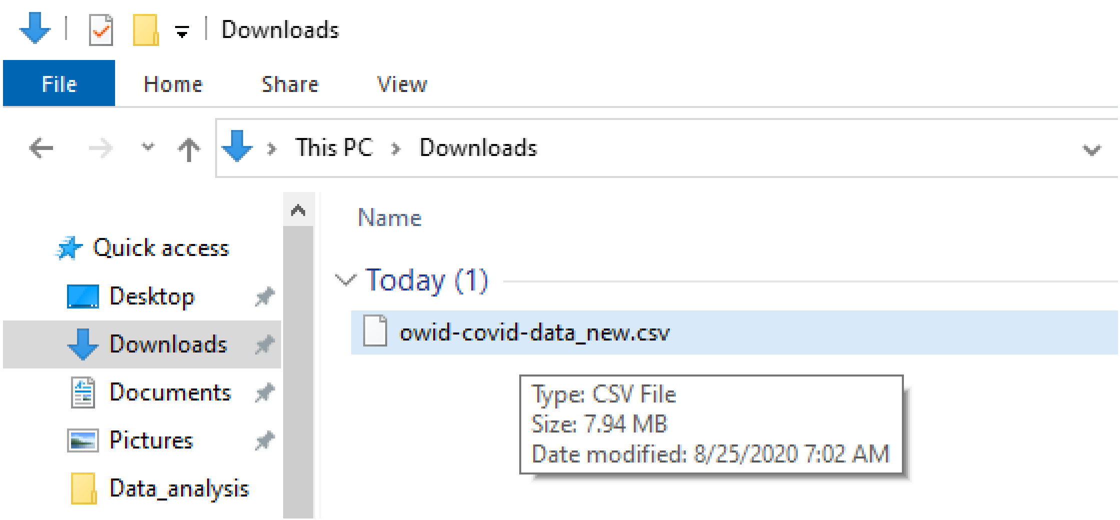
Step 7: Testing scripts on the newest data

Before we finish, I suggest using our script on actual data that can be downloaded from:

https://covid.ourworldindata.org/data/owid-covid-data.csv

As explained above, on the “Our World in Data” web site, you can find the current version of the data.

- Rename .CSV file, copy it to the “Data_analysis” directory, and provide this name to Python script.



owid-covid-data_new.csv

Type: CSV File
Size: 7.94 MB
Date modified: 8/25/2020 7:02 AM

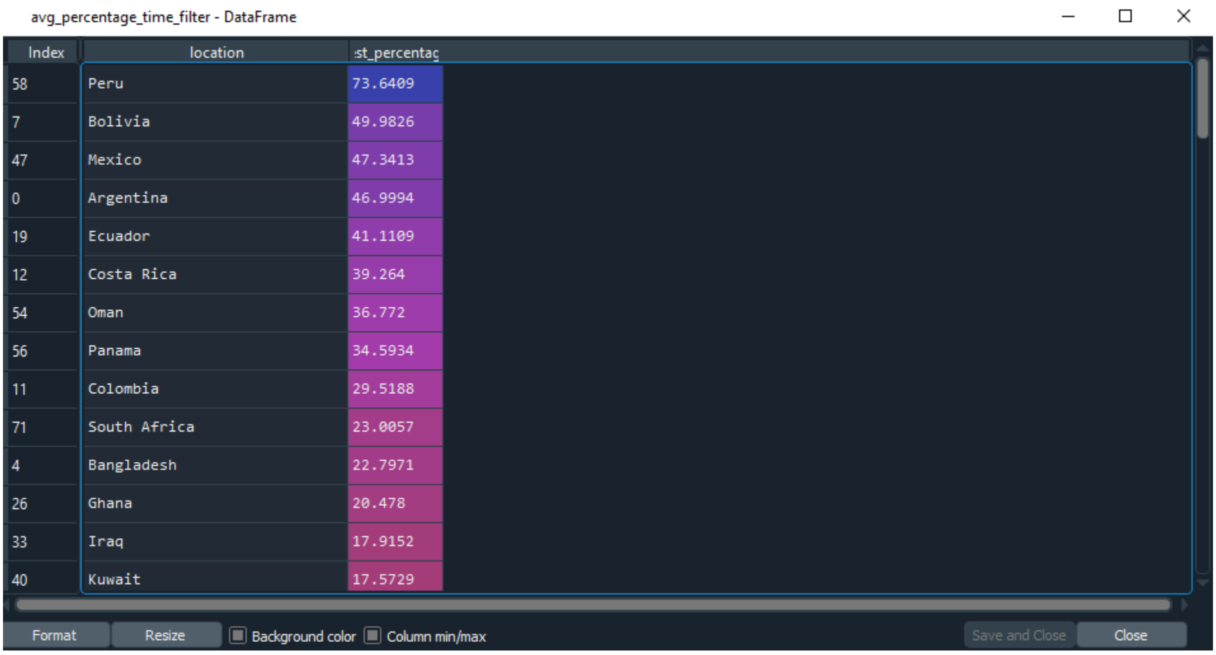
```
covid_data = pd.read_csv('owid-covid-data_new.csv')
```

- Adjust also the time filter to get results for last month and run the Python script.

```
covid_ds_time_filter = covid_ds\
    .loc[(covid_ds['date'] >= '2020-07-25') & (covid_ds['date'] <= '2020-08-25')]

avg_percentage_time_filter =\
      covid_ds_time_filter.groupby(['location'])['test_percentage']\
      .mean()\
      .reset_index()\
      .sort_values(['test_percentage'], ascending=False)

# and on graph legend
plt.legend(['from 2020-07-25 to 2020-08-25'])
```

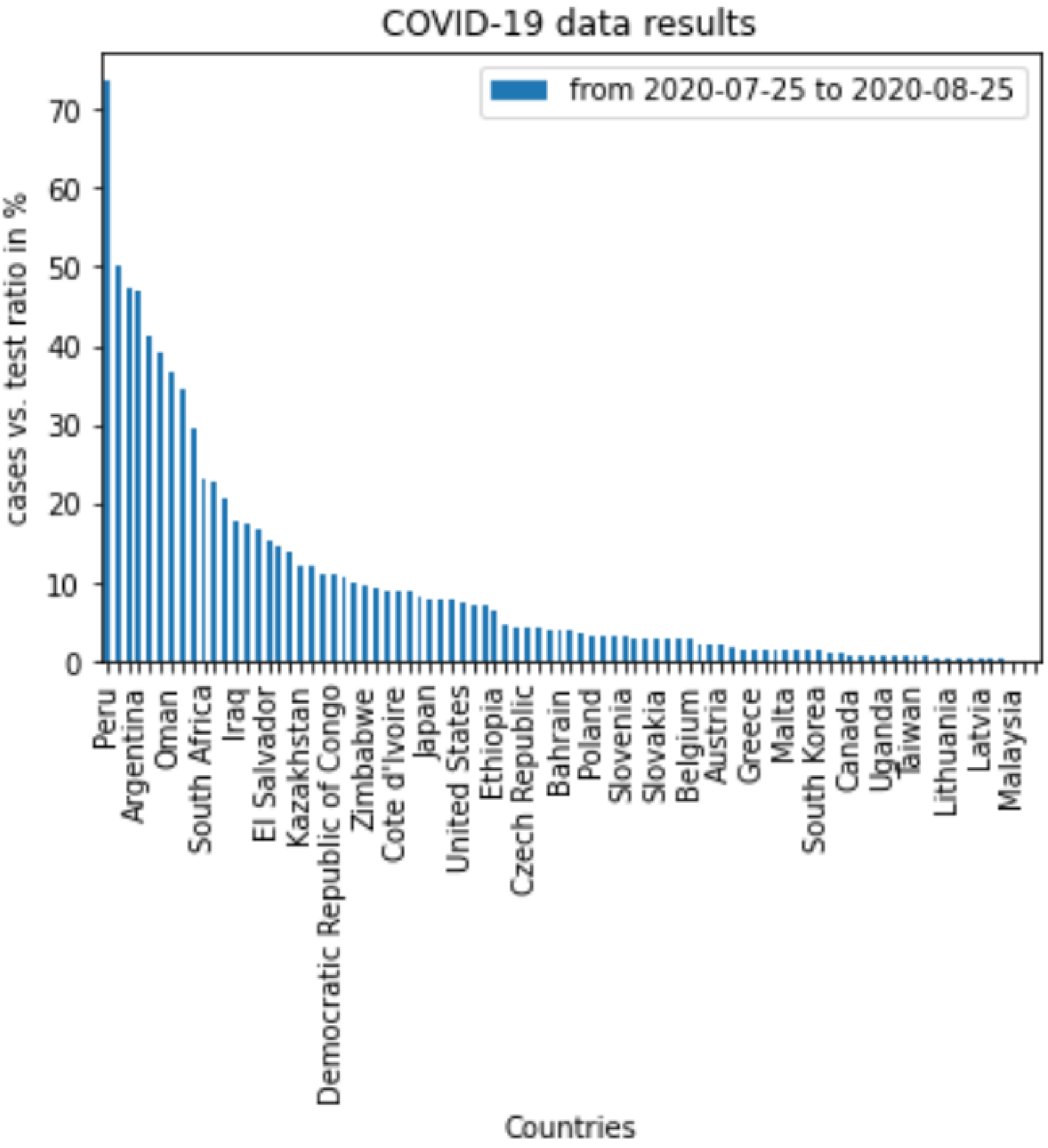
- Execute the Python script to get new results and you can see some changes.

Note that once made script can easily be reused on new set of data. 😊



avg_percentage_time_filter - DataFrame

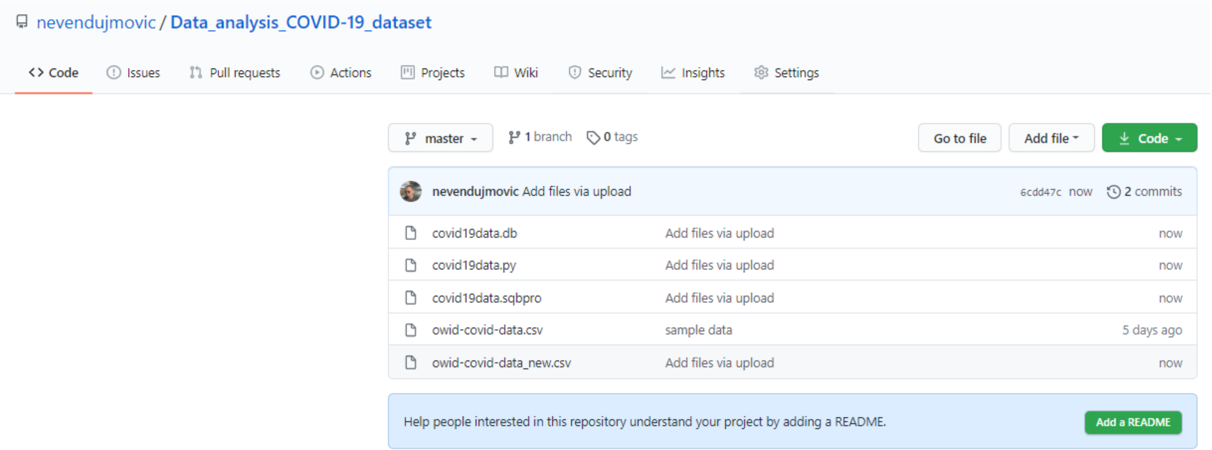
Index	location	st_percentag
58	Peru	73.6409
7	Bolivia	49.9826
47	Mexico	47.3413
0	Argentina	46.9994
19	Ecuador	41.1109
12	Costa Rica	39.264
54	Oman	36.772
56	Panama	34.5934
11	Colombia	29.5188
71	South Africa	23.0057
4	Bangladesh	22.7971
26	Ghana	20.478
33	Iraq	17.9152
40	Kuwait	17.5729



As an exercise, you can perform verification of results with SQL. Once again, you are the one with the task to interpret the results. 😊

Complete source code and data can be found on my GitHub location for this project:

https://github.com/nevendujmovic/Data_analysis_COVID-19_dataset



nevendujmovic / Data_analysis_COVID-19_dataset

nevendujmovic Add files via upload 6cdd47c now 2 commits

covid19data.db	Add files via upload	now
covid19data.py	Add files via upload	now
covid19data.sqbpro	Add files via upload	now
owid-covid-data.csv	sample data	5 days ago
owid-covid-data_new.csv	Add files via upload	now

Help people interested in this repository understand your project by adding a README.

Extra content:

```
covid_ds_pivot = pd.pivot_table(covid_ds_time_filter, index=['date'], values=['test_percentage'],
columns=['location'])

covid_ds_pivot.columns = covid_ds_pivot.columns.droplevel(0)    #remove amount
covid_ds_pivot.columns.name = None                        #remove categories
# covid_ds_pivot = covid_ds_pivot.reset_index()               #index to columns


# covid_ds_pivot_sel = covid_ds_pivot[['Greece', 'Italy', 'Mexico', 'Australia']]
covid_ds_pivot_sel = covid_ds_pivot[['Italy', 'Argentina']]

covid_ds_pivot_sel = covid_ds_pivot_sel.dropna()

covid_ds_pivot_sel.plot(y=['Italy', 'Argentina'], kind = 'line')
plt.show()
```

My friends, this is the end of today's tutorial, and I hope it was time well spent.

Now you know how to, in a short time, prepare your desktop environment for data analysis and perform necessary steps in an analytical process.

I hope you enjoyed using it as much I have enjoyed writing it!

I wish you all the best, and stay healthy!

Author

Neven Dujmovic

P.S. if you like this tutorial be sure to check out other tutorials in my “naive” series on LinkedIn platform.

During the COVID-19 pandemic, I provided free education via the LinkedIn articles platform. All tutorials are covering useful and advanced information technology concepts but are presented on the "easy to learn" way and are adjusted for beginners.

Each tutorial is:

- completely free to use,
- only non-commercial software is used
- tutorials are documented in a high level of details
- live screenshots from used IT systems are provided for each tutorial step
- tutorials are quick to finish (duration max 6-8 hours) and adopted knowledge has direct application
- provided materials can serve as a reference in further usage of the technology.

Following IT educations are available:

- My "naive" Java tutorial with Quarkus on Linux (https://www.linkedin.com/pulse/my-naive-java-tutorial-quarkus-linux-neven-dujmovic/)
- My “naive” Python tutorial (https://www.linkedin.com/pulse/my-naive-python-tutorial-neven-dujmovic/)
- My “naive” virtualization & Linux installation tutorial (https://www.linkedin.com/pulse/my-naive-virtualization-linux-installation-tutorial-neven-dujmovic)
- My “naive” Python development tutorial on Linux (https://www.linkedin.com/pulse/my-naive-python-development-tutorial-linux-neven-dujmovic)
- My “naive” Microsoft SQL Server desktop installation and usage tutorial (https://www.linkedin.com/pulse/my-naive-microsoft-sql-server-desktop-installation-usage-dujmovic/)