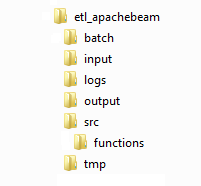
# ETL Process

## Directories and files

For the ETL process to work correctly, the following files and directories must be present after all prerequisites have been installed. In the directory where the project will be created (etl\_apachebeam), the following directories must exist.

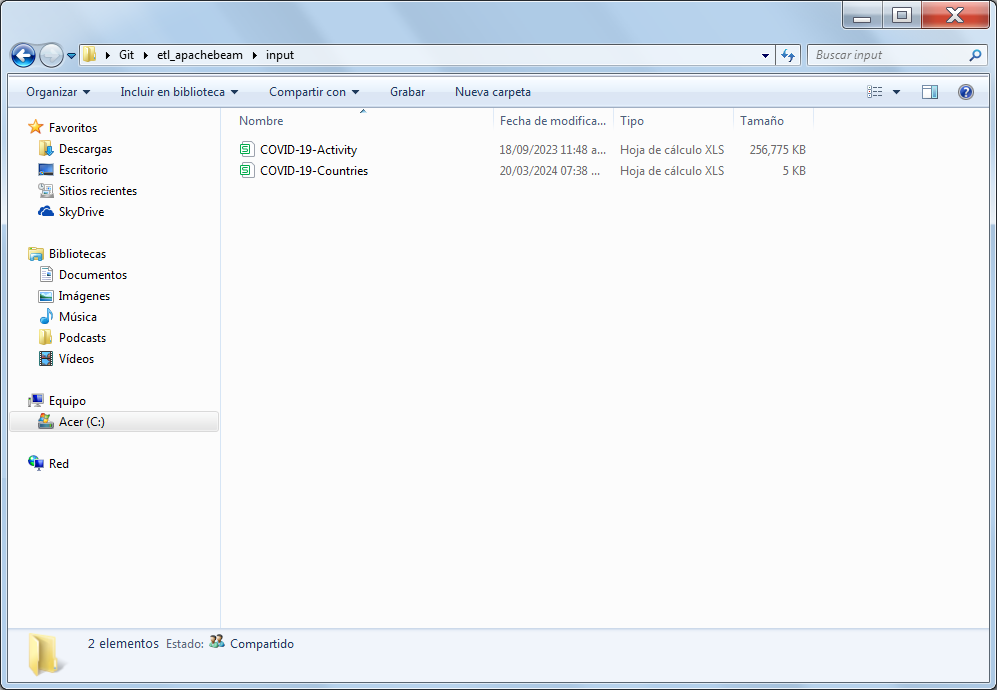


### input directory

The files COVID-19-Countries.csv and COVID-19-Activity.csv must be in the input directory. However, this last file is not in the GitHub repository, because its size exceeds the maximum allowed. To obtain this file you can download it from the following link:

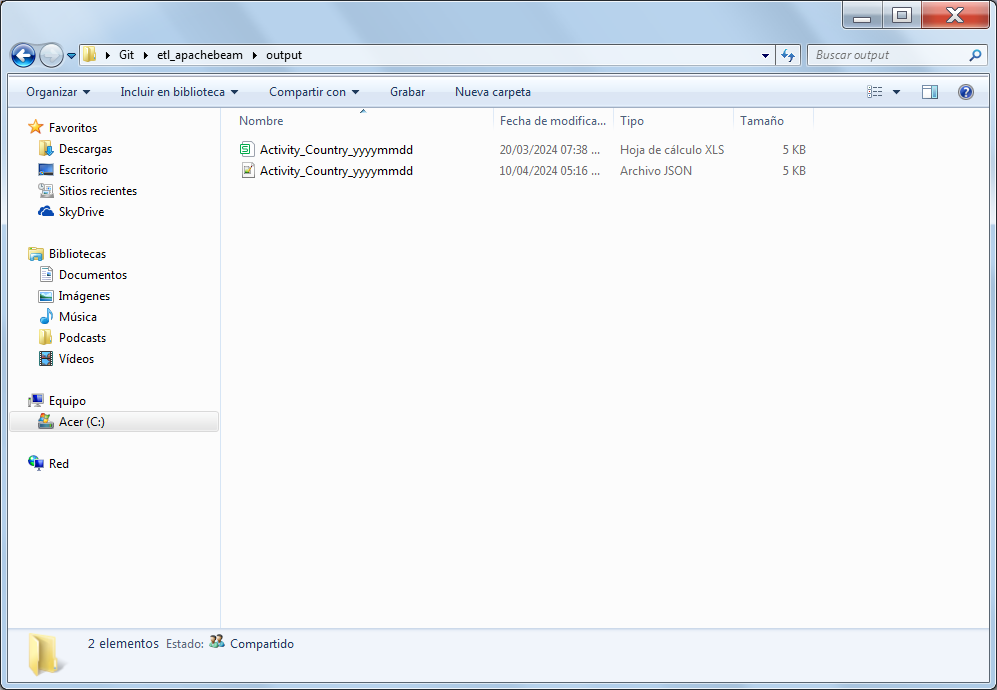
<https://data.world/covid-19-data-resource-hub/covid-19-case-counts>

Once you have downloaded it, you must deposit it in the input directory.



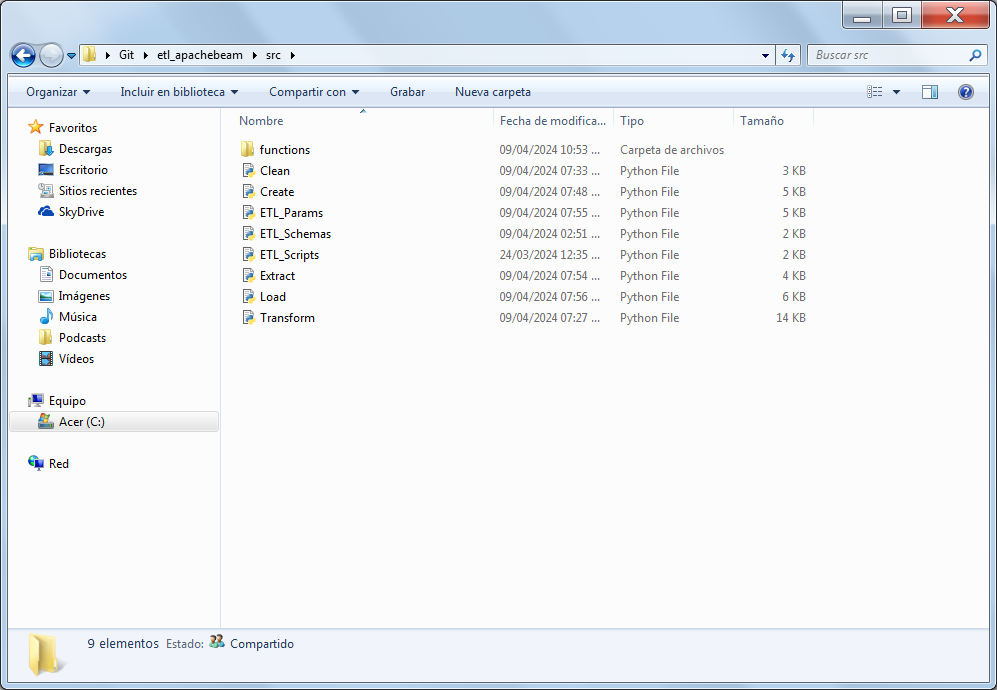
### output directory

The output directory is used to save the final csv files, as well as the json files generated from those files, which will be used to upload to the MongoDB database.



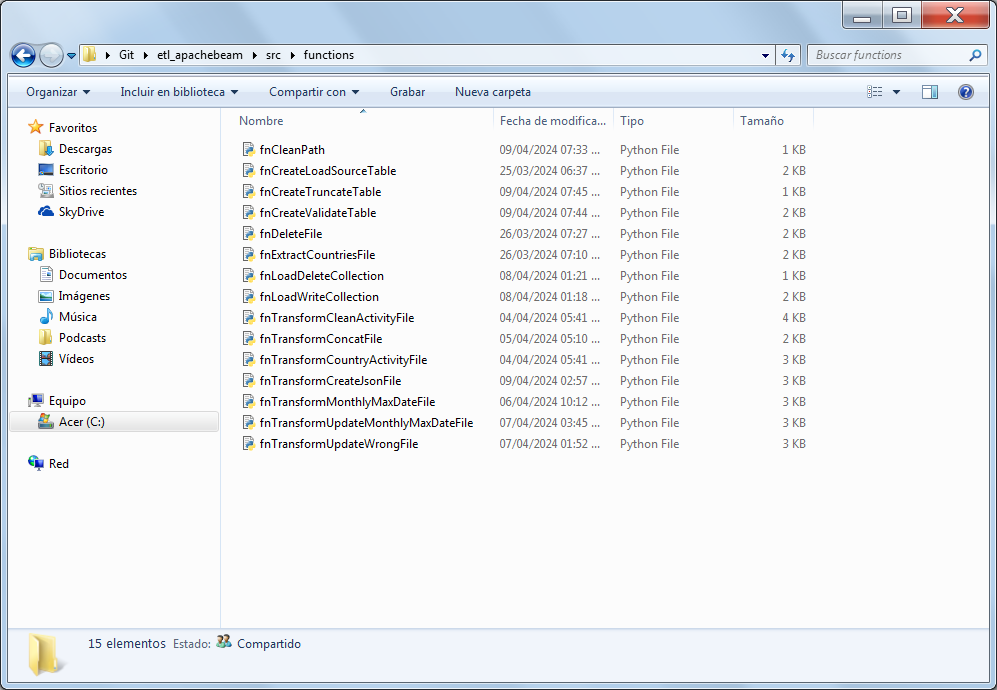
### src directory

The src directory contains the Python files that execute the stages of the ETL process. In addition, there are files that contain parameters, scripts and schemas that the process needs for its execution.



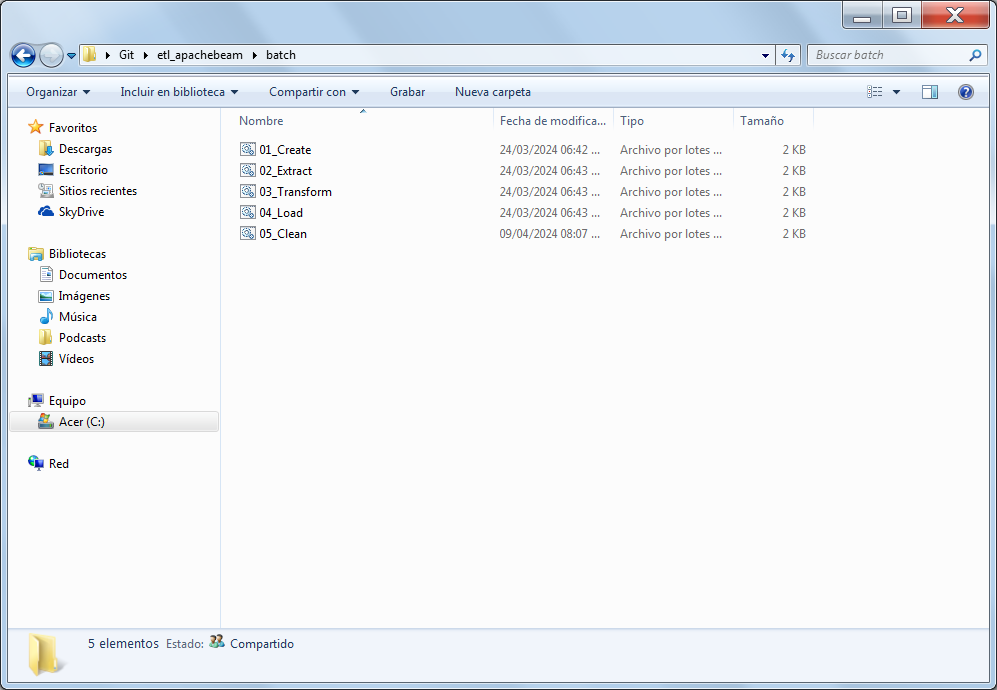
### functions directory

The functions directory contains Python files that help with the execution of the ETL process. These files execute the functions that the Python files in the src directory call.



### batch directory

The batch directory contains the batch files that execute the stages of the ETL process. The files have the .bat extension, since the operating system used is Windows.



The files in this directory are as follows, and the execution order is the same as the number in the file names.

### 01\_Create.bat file

The 01\_Create.bat file creates a date parameter in the format YYYYMMDDHHMISS that maps to the output type file name and executes the Create.py file located at the src path.

### 02\_Extract.bat file

The Extract.bat file creates a date parameter in the format YYYYMMDDHHMISS that maps to the output type file name and executes the Extract.py file located at the src path.

### 03\_Transformation.bat file

The Transformation.bat file creates a date parameter in the format YYYYMMDDHHMISS that maps to the output type file name and executes the Transformation.py file located at the src path.

### 04\_Load.bat file

The Load.bat file creates a date parameter in the format YYYYMMDDHHMISS that maps to the output type file name and executes the Load.py file located at the src path.

### 05\_Clean.bat file

The ETL\_Orchestration.bat file simulates a mesh of processes as it would be in Control - M, since it executes the previous bat files sequentially.

The files in the src and functions directories are as follows:

### Create.py file

The Create.py file runs the process that loads the COVID-19-Countries.csv file into the COVID\_19\_COUNTRIES table in SQL Server. This process was not necessary, since we already have the country origin file, it was only done to have a process prior to the ETL process that would load the information into a table simulating a batch process.

To achieve this, run the next functions from the next files:

fnCreateTruncateTable.fnCreateTruncateTable()

fnCreateLoadSourceTable.fnCreateLoadSourceTable()

fnCreateValidateTable.fnCreateValidateTable()

fnDeleteFile.fnDeleteFile()

### fnCreateTruncateTable.py file

The fnCreateTruncateTable.py file truncates the table in SQL Server. Finally, it gets parameters from the ETL\_Params.py file.

### fnCreateLoadSourceTable.py file

The fnCreateLoadSourceTable.py file loads the COVID-19-Countries.csv file into the COVID\_19\_COUNTRIES table in SQL Server. Finally, it gets parameters from the ETL\_ Params.py file.

### fnCreateValidateTable.py file

The fnCreateValidateTable.py file validates that the table in SQL Server has information loaded. Finally, it gets parameters and scripts from the ETL\_Params.py and ETL\_Scripts.py files respectively.

### fnDeleteFile.py file

The file fnDeleteFile.py deletes a defined file or possible files that exist in the tmp path. This file is not prefixed by any stage of the ETL process, because this file is used in each stage of the ETL process. Finally, it gets parameters from the ETL\_Params.py file.

### Extract.py file

The Extract.py file runs the process that extracts information from the COVID\_19\_ COUNTRIES table in SQL Server into a csv file.

To achieve this, run the next functions from the next files:

fnExtractCountriesFile.fnExtractCountriesFile()

fnDeleteFile.fnDeleteFile()

### fnExtractCountriesFile.py file

The fnExtractCountriesFile.py file extracts information from the COVID\_19\_COUNTRIES table in SQL Server into a csv file. Finally, it gets parameters and scripts from the ETL\_ Params.py and ETL\_Scripts.py files respectively.

### fnDeleteFile.py file

The file fnDeleteFile.py deletes a defined file or possible files that exist in the tmp path. This file is not prefixed by any stage of the ETL process, because this file is used in some stages of the ETL process. Finally, it gets parameters from the ETL\_Params.py file.

### Transform.py file

The Transfom.py file runs the process that generates the json files from the file generated in the extraction process and the file obtained from the link described in the input directory section.

To achieve this, run the next functions from the next files:

fnTransformCleanActivityFile.fnTransformCleanActivityFile()

fnTransformConcatFile.fnTransformConcatFile()

fnTransformCountryActivityFile.fnTransformCountryActivityFile()

fnTransformUpdateWrongFile.fnTransformUpdateWrongFile()

fnTransformMonthlyMaxDateFile.fnTransformMonthlyMaxDateFile()

fnTransformUpdateMonthlyMaxDateFile.fnTransformUpdateMonthlyMaxDateFile()

fnTransformCreateJsonFile.fnTransformCreateJsonFile()

fnDeleteFile.fnDeleteFile()

### fnTransformCleanActivityFile.py file

The fnTransformCleanActivityFile.py file cleans the data from the COVID-19-Activity.csv file, because there are descriptions that come with asterisks and a comma and the comma is the column separator. Generating two files, one with the correct data and another where did data cleaning. Finally, it gets parameters from the ETL\_Params.py file.

### fnTransformConcatFile.py file

The file fnTransformConcatFile.py concatenates the created files into some file, generating a new file. Finally, it gets parameters from the ETL\_Params.py file.

### fnTransformCountryActivityFile.py file

The fnTransformCountryActivityFile.py file splits the file in two. The first has the records with the correct country alpha code and the second has the records without the country alpha code. Finally, it gets parameters from the ETL\_Params.py file.

### fnTransformUpdateWrongFile.py file

The fnTransformUpdateWrongFile.py file updates the file with the empty country alpha code with the correct country alpha code. Finally, it gets parameters from the ETL\_Params.py file.

### fnTransformMonthlyMaxDateFile.py file

The fnTransformMonthlyMaxDateFile.py file creates a file with the record for each county, state and country, with the maximum date per month. Finally, it gets parameters from the ETL\_Params.py file.

### fnTransformUpdateMonthlyMaxDateFile.py file

The fnTransformUpdateMonthlyMaxDateFile.py file of the file with the maximum dates per month adds the missing data to have a layout equal to the COVID-19-Activity.csv file. Finally, it gets parameters from the ETL\_Params.py file.

### fnTransformCreateJsonFile.py file

The file fnTransformCreateJsonFile.py generates json files from the csv files that were generated in the transformation process, these files will be loaded into collections within the MongoDB database. Finally, it gets parameters and schemas from the ETL\_Params.py and ETL\_Schemas.py files respectively.

### fnDeleteFile.py file

The file fnDeleteFile.py deletes a defined file or possible files that exist in the tmp path. This file is not prefixed by any stage of the ETL process, because this file is used in some stages of the ETL process. Finally, it gets parameters from the ETL\_Params.py file.

### Load.py file

The Load.py file runs the process that loads the json files generated in the transformation process into the MongoDB database.

To achieve this, run the next functions from the next files:

fnLoadDeleteCollection.fnLoadDeleteCollection()

fnLoadWriteCollection.fnLoadWriteCollection()

### fnLoadDeleteCollection.py file

The fnLoadDeleteCollection.py file deletes collections from the MongoDB database. Finally, it gets parameters from the ETL\_Params.py file.

### fnLoadWriteCollection.py file

The fnLoadWriteCollection.py file loads the previously generated json files into the MongoDB database collections. Finally, it gets parameters from the ETL\_Params.py file..

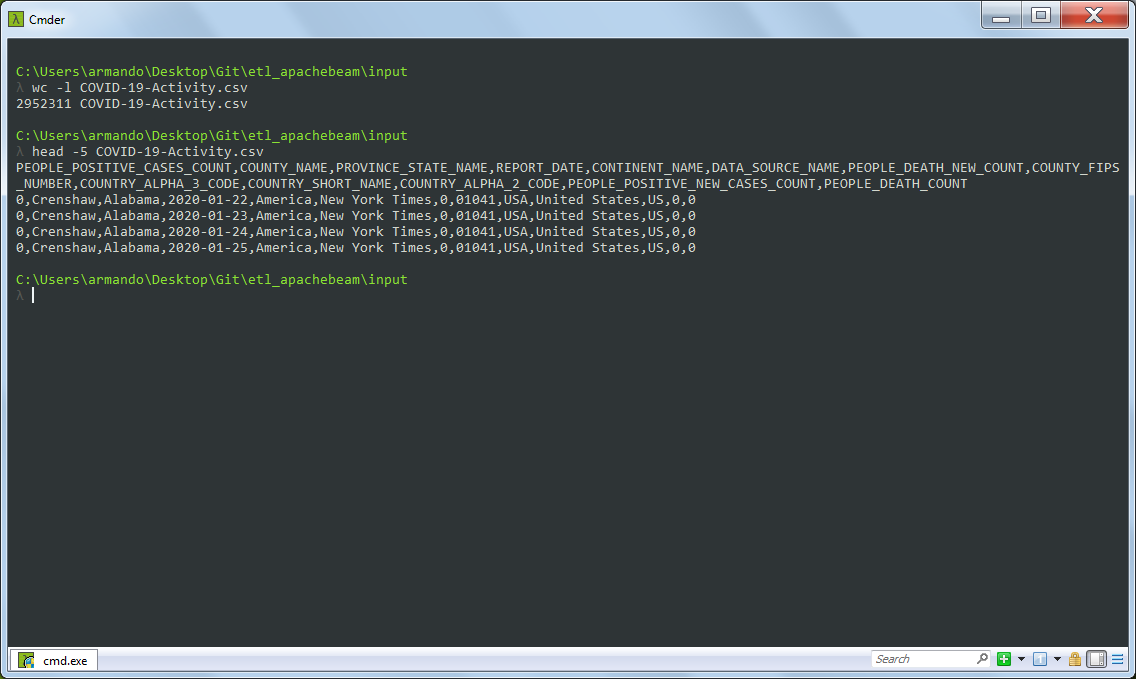
## MongoDB database

MongoDB is a NoSQL database management system (DBMS), which means it's a database that doesn't use the relational model used by traditional SQL databases. Instead, MongoDB uses a flexible data model based on JSON (JavaScript Object Notation) documents, allowing it to store and manage data more dynamically and scalably.

One of MongoDB's distinctive features is its ability to store data in BSON (Binary JSON) documents, enabling efficient representation of information and fast data access. Additionally, MongoDB is known for its ability to horizontally scale, meaning it can distribute and manage large volumes of data across multiple servers efficiently.

Some key features of MongoDB include its ability to handle semi-structured and unstructured data, its capability for performing complex queries, and its flexibility to adapt to changes in the data schema without needing to modify the database structure.

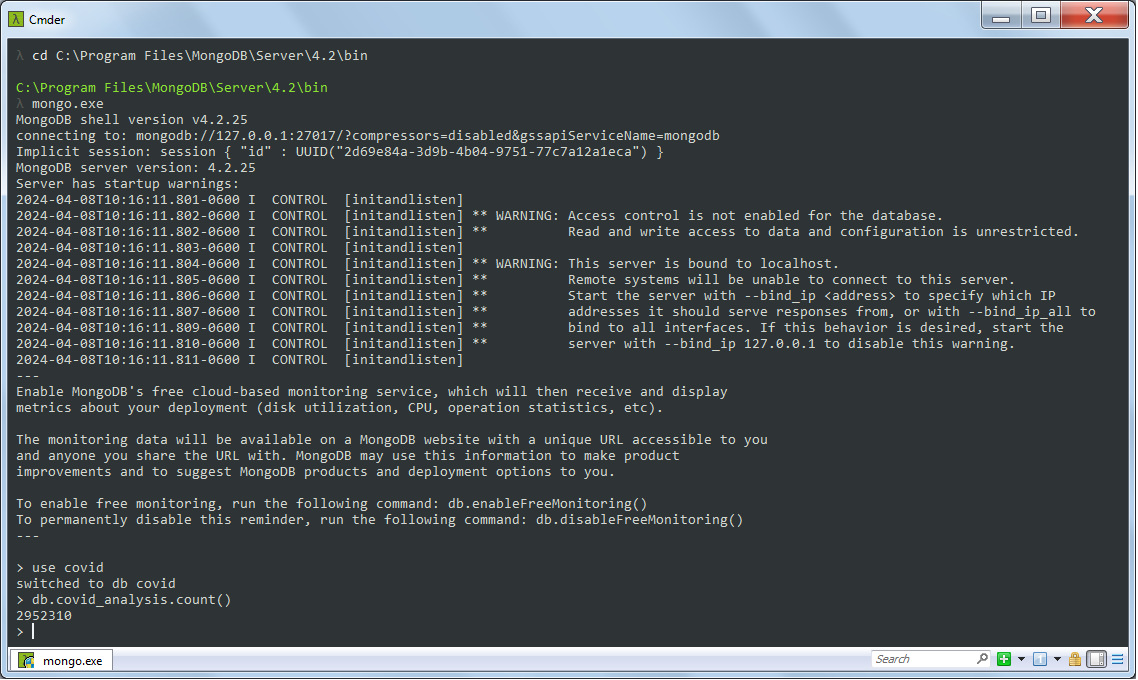
As already mentioned, the process takes as input the COVID-19-Activity.csv file, which contains a daily summary of how the COVID-19 pandemic evolved in several countries around the world.



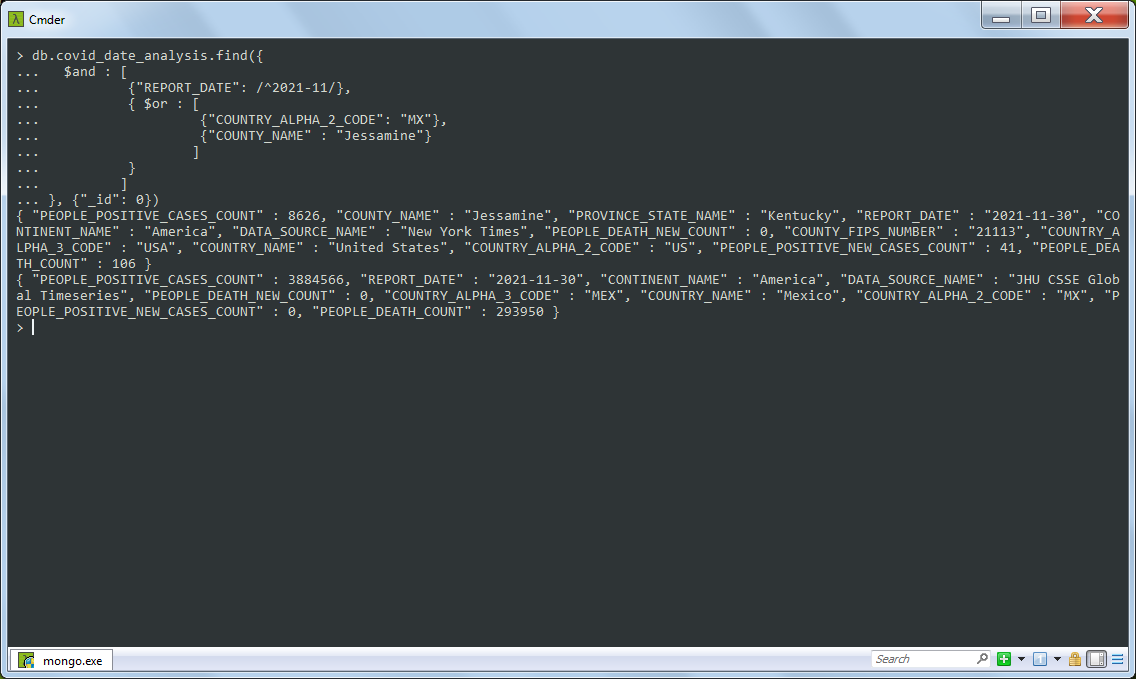
In the figure you can see that the file has almost 3 million records. However, the actual count is 2952310, since the first record is the header. Additionally, when reviewing the file, most of the countries in the county and state columns do not have information.

The latter was why MongoDB was chosen as the target database, since it is a NoSQL, document-oriented database. It is not necessary to assign a schema to its data collections, but it is possible to add indexes to the columns that its collections will have.

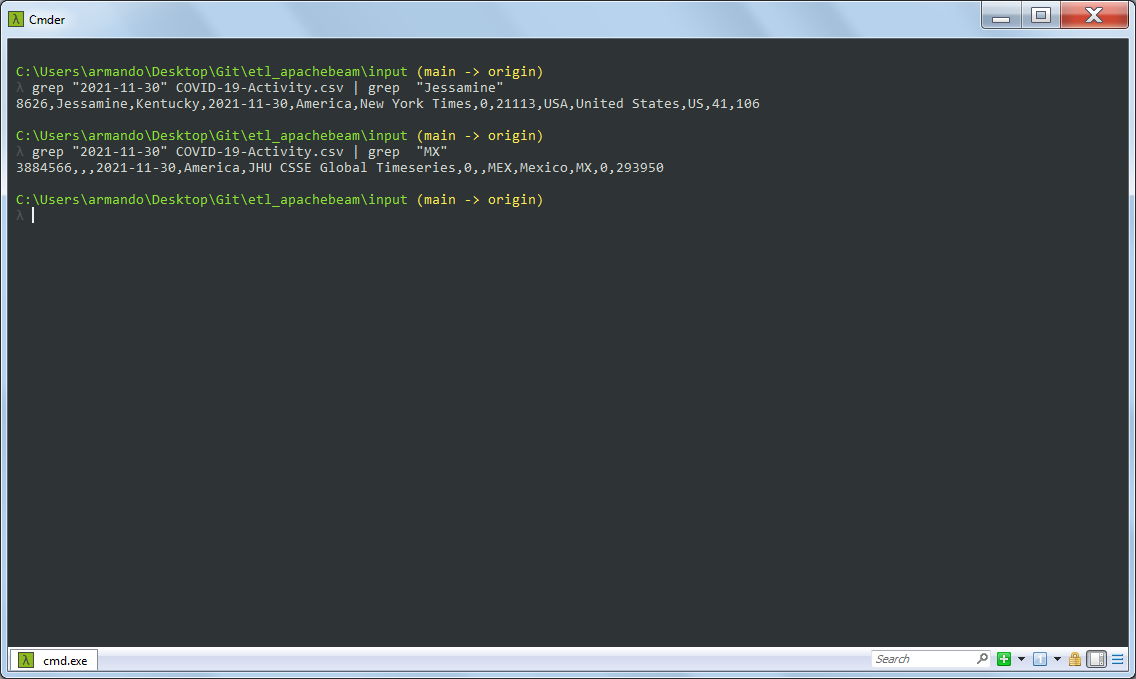
Three collections were created in MongoDB, the first contains the information from the COVID-19-Activity.csv file. Below is the number of documents (records) loaded in the collection which are the same as in the file, excluding the header.



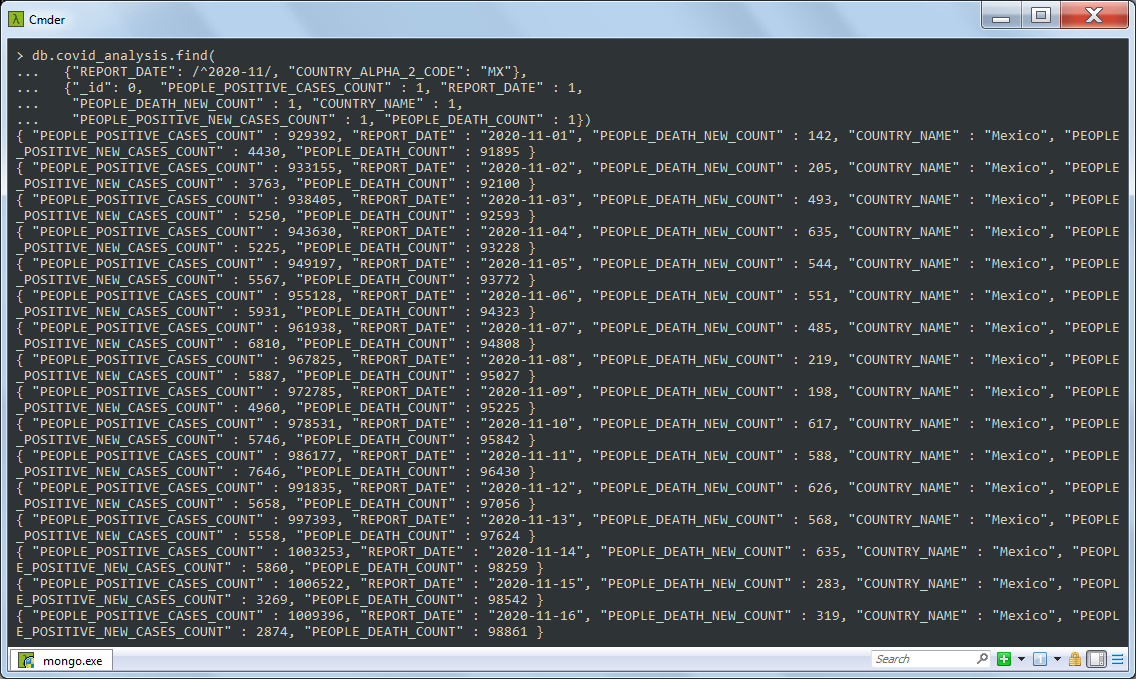
Using the way MongoDB loads data into its collections, records where there was no information in the county and state columns are not included in the json file, shown below.



It can be seen that for the document from Mexico, the county or state does not exist, contrary to the document from the United States, where they do exist. When searching for these records in the COVID-19-Activity.csv file, the same behavior can be observed, but in that case the columns are empty.

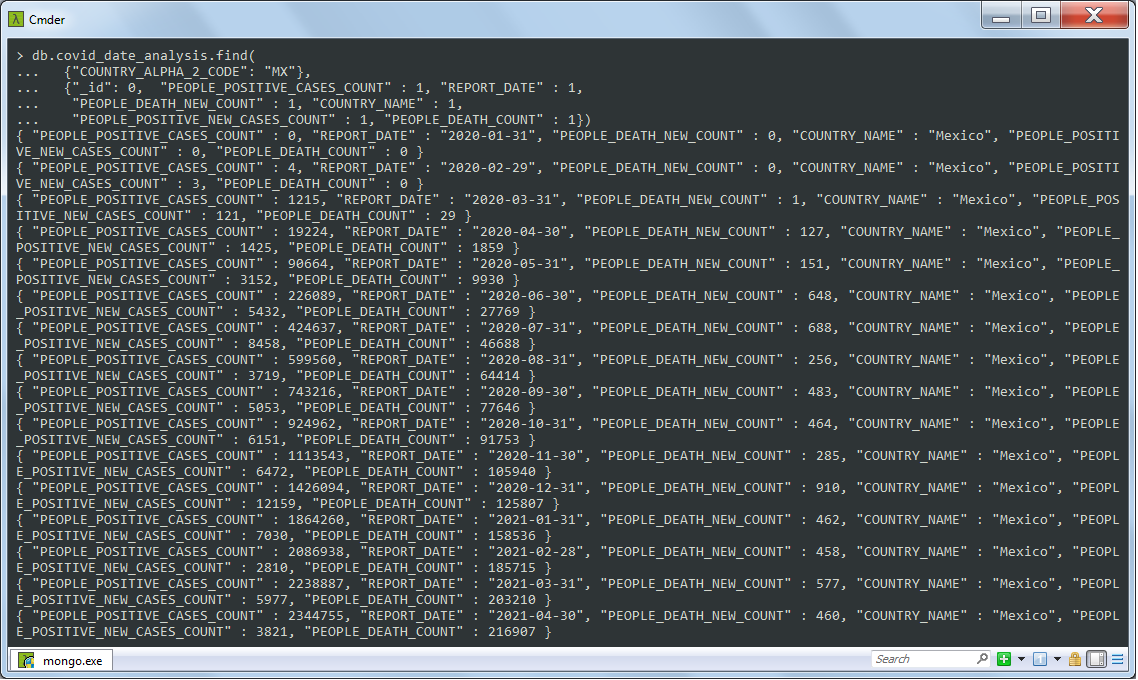


In this collection you can do analysis to see how the progression of COVID was by day, as shown below.

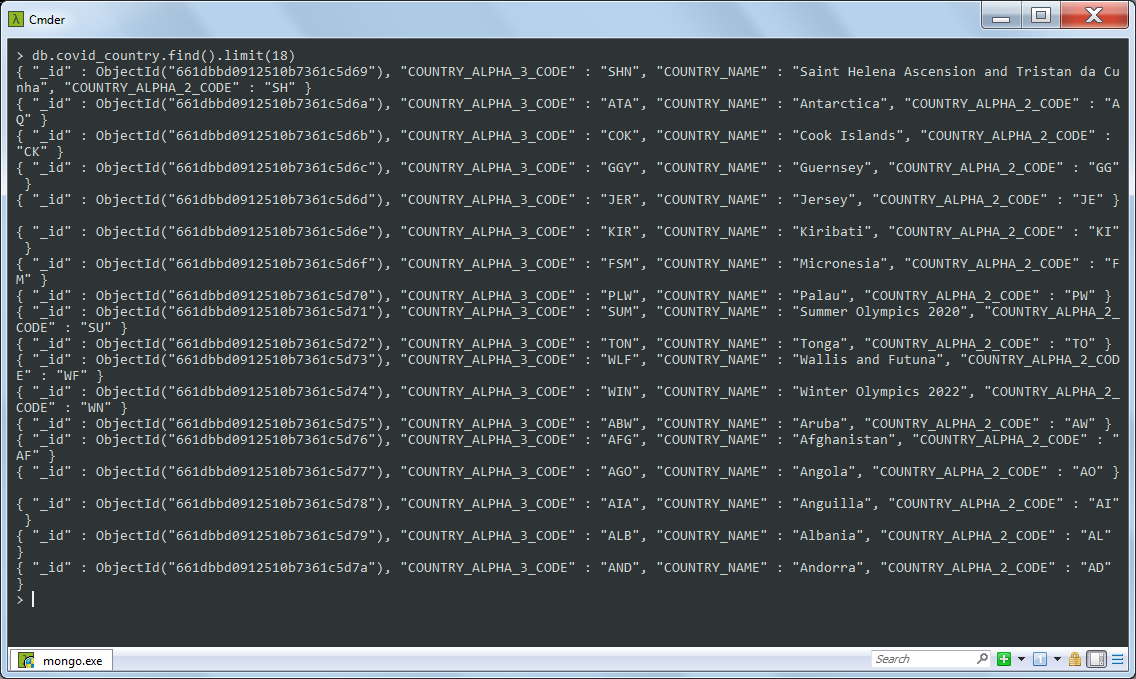


The next collection that was created in MongoDB was to load a file with the information of each county, state and country for the last day of each month. Again it was applied where there is no data in the county and state, the document will not consider those columns to upload them to the collection.

This collection helps when you want to carry out an analysis of how COVID evolved each month.



The last collection loaded the file with the countries, including their alphabetical codes.



In the previous image, you can see how when making a query to the collection of countries, the result sends a field in each document called \_id, this is a special field reserved to store the unique identifier of each document in a collection, which It is generated automatically when each document is loaded into the collection. This field is similar to the concept of primary key in relational databases.

Finally, when performing queries in MongoDB you must take into account that the database is not like SQL type databases, since if you query a collection that has not been created, the database will not return any errors, as shown bellow.

