ETL COVID-19: confirmed cases, deaths, tests, and vaccines.

**Team Members:**

Sonia Suárez

Orlando García

Leo Preciado

Viviana Aragón

# Project Description/Outline:

The aim of this project is to use the method ETL to clean up different data sets and store the information needed for a specific analysis.

* **E**xtract original data sources (CSV, JSON, pgAdmin 4, etc).
* **T**ransform the data sets according to the analysis requirements: cleaning, joining, filtering, aggregating, etc.
* **L**oad the final database, tables/collections, and why this was chosen.

# Data

We selected for our study 4 databases with different information about COVID-19 and an additional one to retrieve latitude and longitudes of countries.

* Confirmed Cases: Raw Global Confirmed Cases.csv <https://www.kaggle.com/antgoldbloom/covid19-data-from-john-hopkins-university>
* Deaths: Raw Global Deaths.csv <https://www.kaggle.com/antgoldbloom/covid19-data-from-john-hopkins-university>
* Vaccination: Country Vaccinations.csv <https://www.kaggle.com/gpreda/covid-world-vaccination-progress?select=country_vaccinations.csv>
* Testing: Testing All Observations.csv

<https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/testing/covid-testing-all-observations.csv>

* Countries: <https://developers.google.com/public-data/docs/canonical/countries_csv>

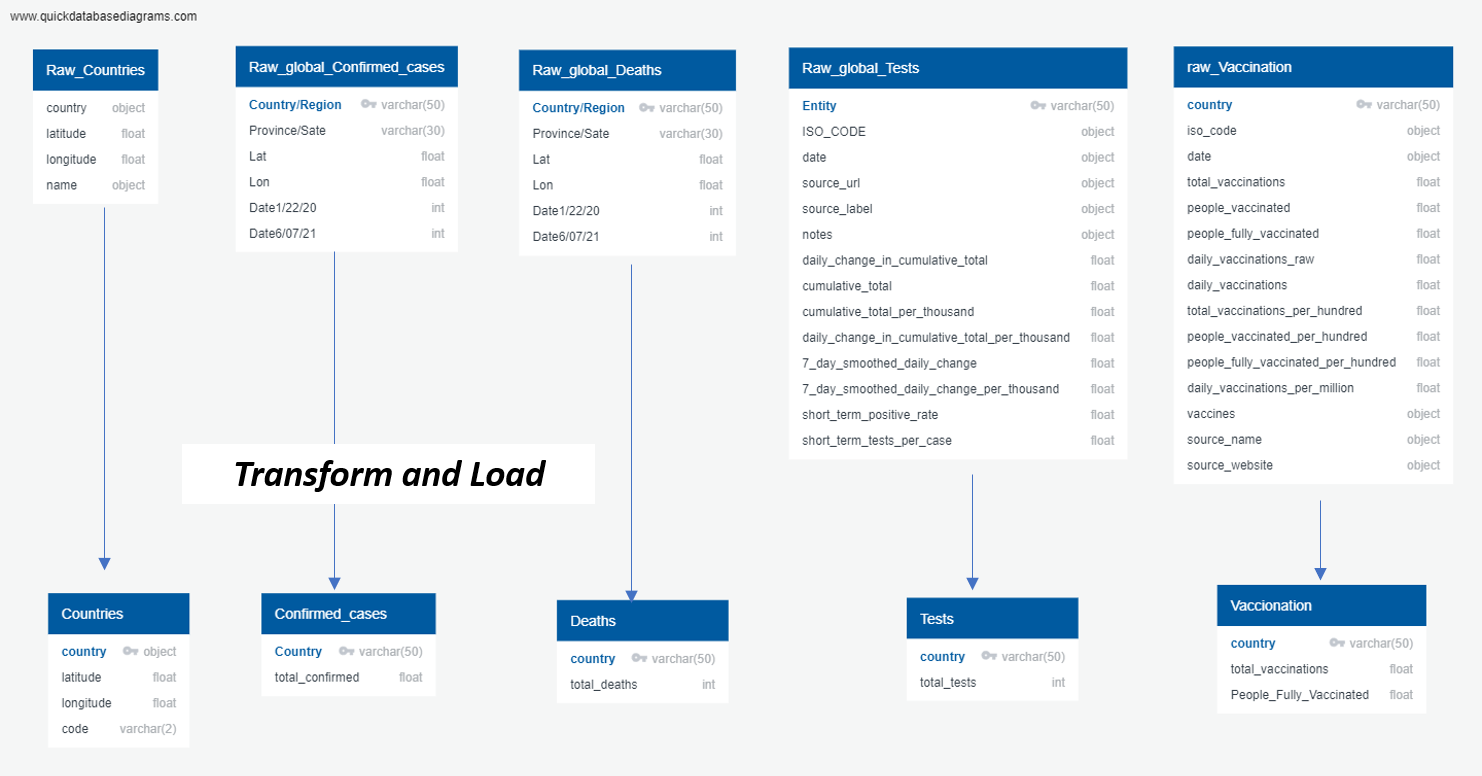
# Description of our analysis

Before starting to clean up our databases, we explore their variables and types to visualize which data we would like to retrieve from each one and how our final database would look like. In the following images, we present our process to get from our 5 databases to our final table COVID-19 to do our analysis about COVID-19.

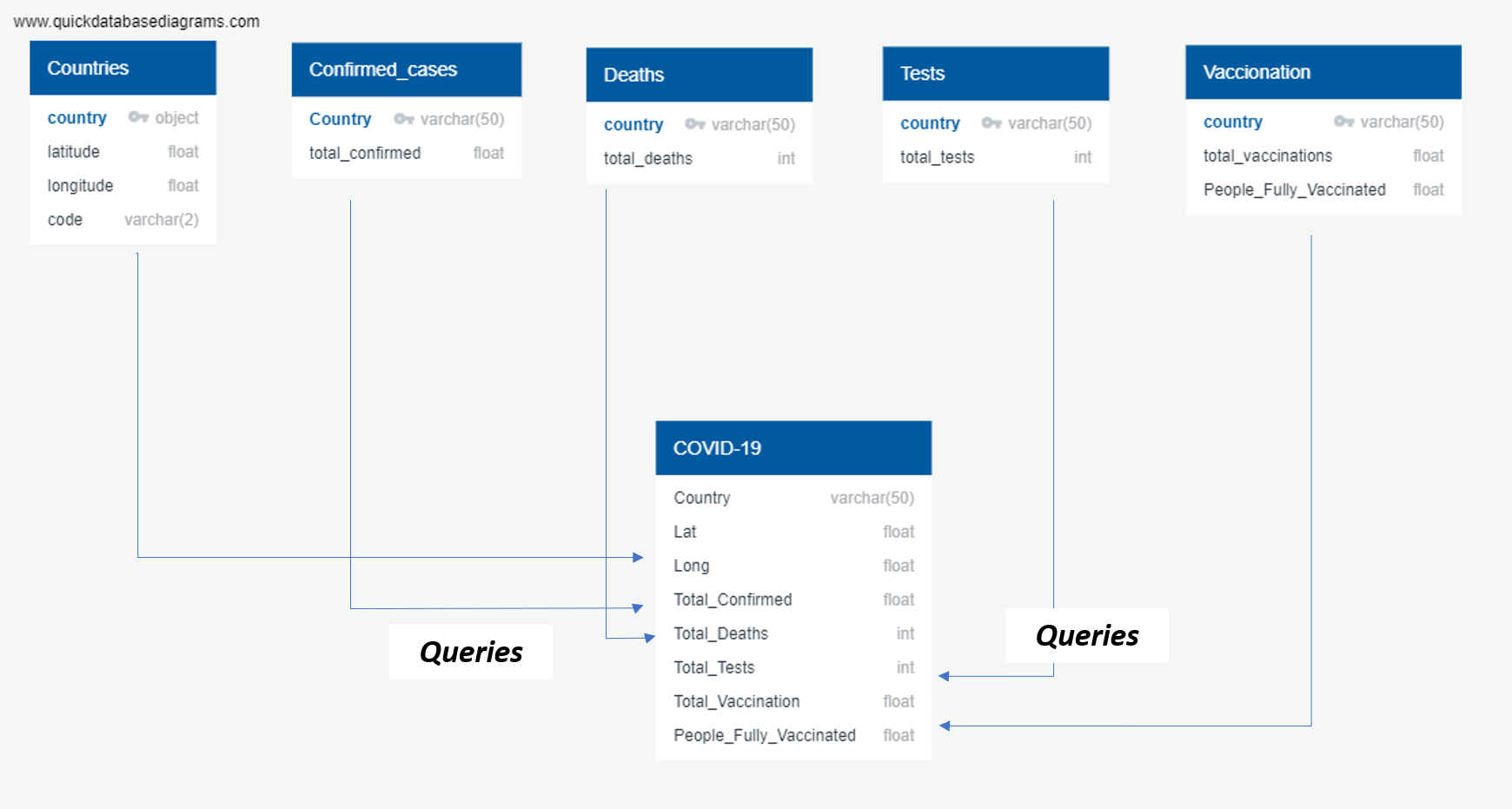
Escala de tiempo

Descripción generada automáticamente con confianza baja

First, we clean each of our databases and get just the variables we need. Second, we store our information in a sql format, given that the data is clearly structured. For this step, we create our tables and variables for each one and save it in a sql schema.



Once we have stored our data, we do the respective queries to join our databases with the variable country as our id of each one.



# Extract, Transform and Load

**Period of Analysis.** Once exploring our databases, we found that each database has information for different periods.

* Confirmed\_cases: 01/22/2020 - 06/07/2021
* Deaths: 01/22/2020 - 06/07/2021
* Tests: 12/02/2020 - 05/28/2021
* Vaccination: 12/02/2020 - 05/28/2021

Given that the vaccination and test database contain information from December 2nd 2020 to May 28th 2021, we selected this as our period for the analysis.

## Confirmed Cases

**EXTRACT.** We extracted information in a CSV from a Kaggle Dataset COVID-19 data from John Hopkins University (<https://www.kaggle.com/antgoldbloom/covid19-data-from-john-hopkins-university>).

We read it with Pandas to explore its variables and types. It contained 507 columns, this is 503 daily confirmed cases data from January 22nd 2020 to June 6th 2021, and 275 rows for 275 countries/regions. Additionally to confirmed cases, we had the latitude and longitude of the country/region.

**TRANSFORM.** We first checked if there were NaN in our dataframe. There were none.

Given that we would like to have the total confirmed cases from December 2nd 2020 to May 28th 2021 per country, we create a new variable as the sum of the columns that correspond to those days by country.

We keep two variables from this database: Country and Total\_Confirmed. We didn't keep latitude and longitude because when we did the sum by country, this function summed the values of the lat and long per country when existing different provinces or states per country.

We verified that there were no repeated values per country to have the country as a unique key, considering that this value will be needed to join our 5 databases.

**LOAD.** Finally, we insert our data to our sql table Total\_confirmed.

## Total Tests

**EXTRACT.** For this part of the exercise we stored a CSV extracted from Our World in Data Coronavirus Testing Database (<https://ourworldindata.org/coronavirus-testing>) which included Entity, ISO Code, Date, Source URL, Source Label, other columns and the one column we studied with entity later named country which is Cumulative Total We generated a dataframe. The Total Table Contained 52235 Rows x 14 Columns. The data showed us two types of columns: object and float64.

**TRANSFORM.** For this part of the cleanup, we did various things. First with str.replace we eliminated the extra text from many countries, such as -test performed, people tested and samples tested. We filtered the data frame with only three columns, which are the only necessary columns for our study: entity (country), date and cumulative total.

We then did another filtering, in which we used only one date, because numbers are shown in a cumulative way. The date used was 2021-05-28 which is the last date used for our study.

We kept the required columns which were Entity and Cumulative Total, using as reference the last date. We used a groupby for entity and did a sum so the numbers were clear. The table made was 117 Rows x 3 Columns and stayed only with the required columns.

**LOAD.** We first changed the column names to country and total\_tests so they could fit into the tables made in PostgreSQL with pgAdmin. From Jupyter Notebook we used Pandas and Psycopg2 to connect to a SQL Local Database and load the data from our dataframe to SQL.

## Deaths

**EXTRACT.** The source file was in CSV format, the first step was to read the file and convert it to a Pandas dataframe.  
(<https://www.kaggle.com/antgoldbloom/covid19-data-from-john-hopkins-university> )

**TRANSFORM.**  The dataset was cleaned removing the NaN. Then we filtered by range of dates (12/2020 to 05/2021) to keep the data of our selected period. To get the information of total deaths, we sum the total cases of this range of dates and the result was assigned to a new column. The new data frame was grouped by Country so we can get rid of the provinces.We only kept the Country and Total death cases columns.

**LOAD.** The final data frame was loaded into a relational database in PgAdmin in order to be able to join this information with the other tables resulting from the analysis.

## Vaccination

**EXTRACT.** For this part of the exercise, we extracted information in a CSV from a Kaggle Dataset COVID-19 World Vaccination Progress and created a dataframe. (<https://www.kaggle.com/gpreda/covid-world-vaccination-progress?select=country_vaccinations.csv>). The CSV was originally 21628 Rows x 15 Columns. We then saw that types of the columns were object and float64.

**TRANSFORM.** We did a filtering of dates, given the range from 2020-12-2 to 2012-5-28. We filtered the data frame into a new data frame with only the required columns which were country, total\_vaccinations and people\_fully\_vaccinated. We did an isnull test and they were no null values. The new dataframe was only 214 Rows x 3 Columns and only carried the necessary columns.

**LOAD.** To load the data frame into the local we had to connect it with an engine, then we used the Pandas as Psycopg libraries so we could load them to SQL.

## Countries

**EXTRACT.** The countries latitudes and longitudes were scraped from <https://developers.google.com/public-data/docs/canonical/countries_csv>

**TRANSFORM.** The info was converted to csv so it can be readable.The csv was imported to pandas to verify integrity.At this point the only transformation needed was to rename the columns so it matches with the DB structure.

**LOAD.** Finally the Data frame was loaded into PgAdmin from the Jupyter notebook using Psycopg

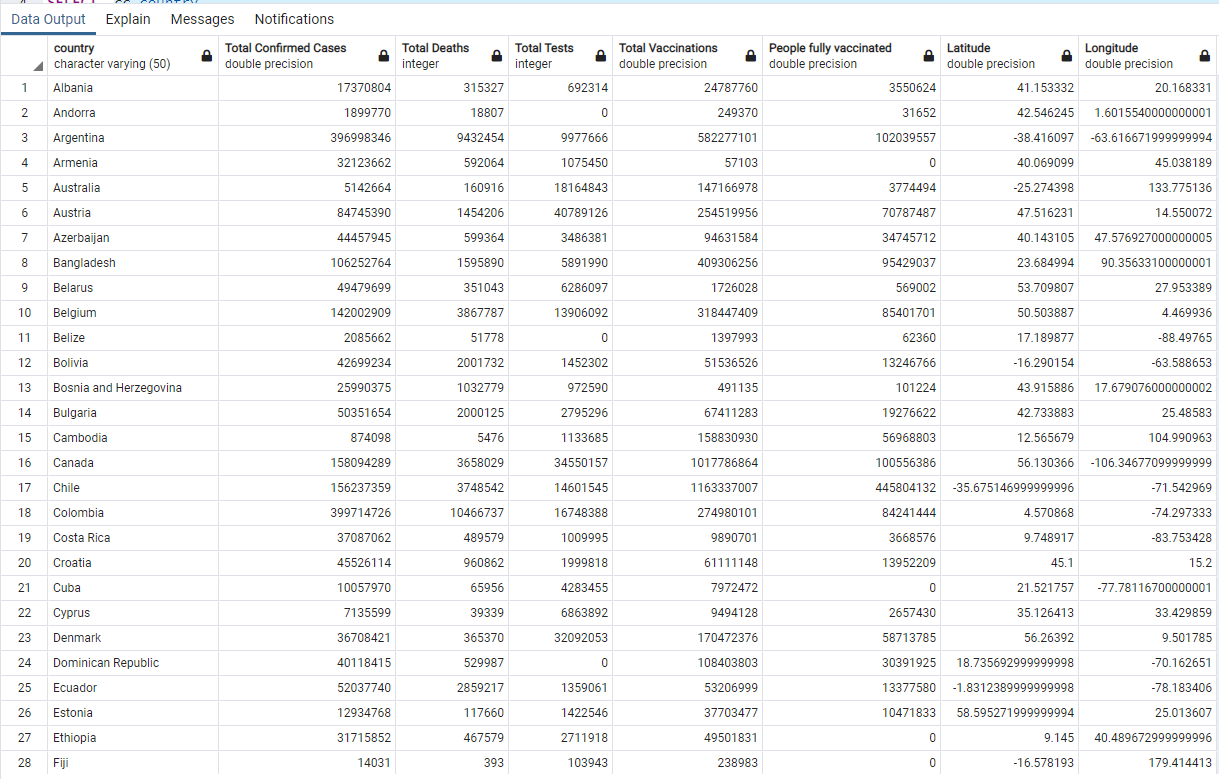
# Final Database

After the loading we end up with 5 tables each one with country as the key, with this in mind we can join all the tables to get the total info depending on the data we want to observe.

The database also includes a table with the geolocalization coordinates of each country and it’s ISO Code, this will allow the final user to create a heat map or any interesting stuff a creative user has in mind.

**Query result**

With the data sets loaded in the DB, we can easily join the information to get a summary for each country about the confirmed cases, deaths, test applied, vaccinations and the people fully vaccinated during the COVID-19 pandemic.

****

# Analysis

Finally, with this clean and structure database of COVID-19, we can analyze and visualize this information in order to respond different questions:

* Which countries have had a higher number of confirmed cases and deaths?
* Is there a possibility, given the latitude and longitude of each country, that neighborhood countries in better conditions could help the most disadvantaged ones?
* Is there a way to create regional programs?
* Is there a way to achieve international cooperation so we could help one another?
* In reality programs have been made, and international cooperation with health issues and vaccines has been happening.

In order to respond to these questions, the database could be completed with other variables like total population to have comparable information per country.