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

Team Members:

Sonia Suárez
Orlando García
Leo Preciado
Viviana Aragón

1 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.

2 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

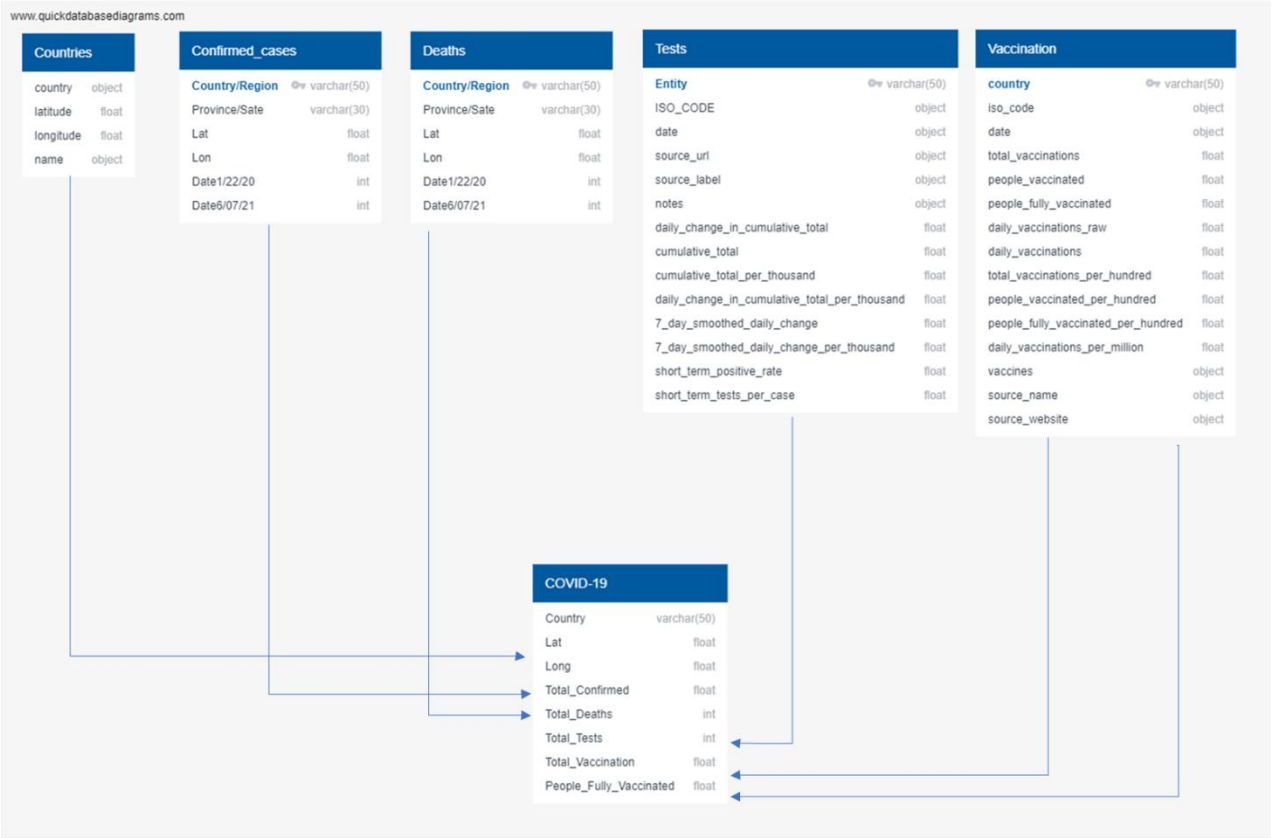
  https://raw.githubusercontent.com/owid/covid-19-data/master/public/data/testing/covid-testing-all-observations.csv

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

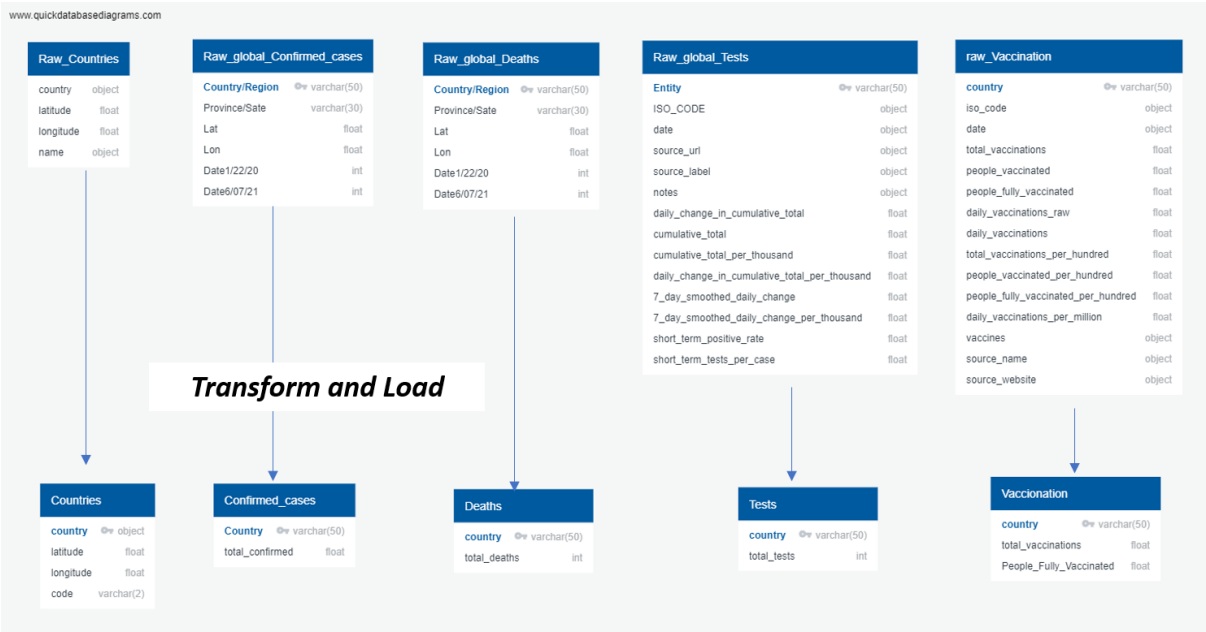
3 Description of our analysis

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

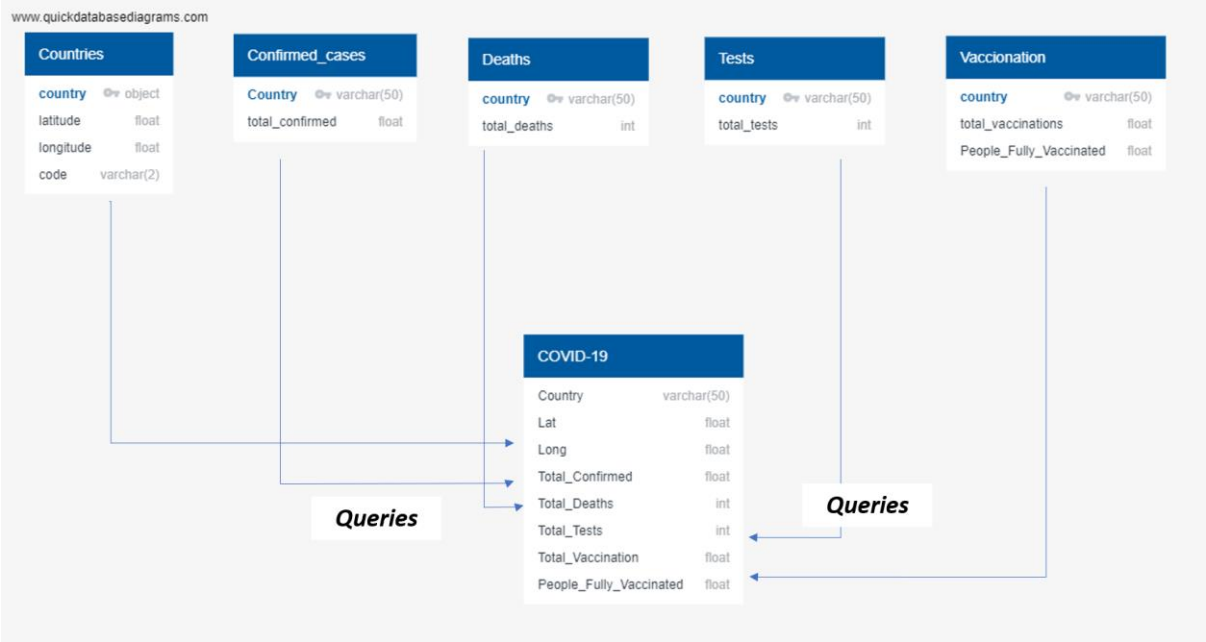
like. In the following images, we present our process to get from our 5 databases to our final table COVID-19 to do our analysis about COVID-19.



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.



4 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.

4.1 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.

4.2 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.

4.3 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.

4.4 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.

4.5 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

5 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.

Data Output | Explain | Messages | Notifications

	country character varying (50)	Total Confirmed Cases double precision	Total Deaths integer	Total Tests integer	Total Vaccinations double precision	People fully vaccinated double precision	Latitude double precision	Longitude double precision
1	Albania	17370804	315327	692314	24787760	3550624	41.153332	20.168331
2	Andorra	1899770	18807	0	249370	31652	42.546245	1.6015540000000001
3	Argentina	396998346	9432454	9977666	582277101	102039557	-38.416097	-63.616671999999994
4	Armenia	32123662	592064	1075450	57103	0	40.069099	45.038189
5	Australia	5142664	160916	18164843	147166978	3774494	-25.274398	133.775136
6	Austria	84745390	1454206	40789126	254519956	70787487	47.516231	14.550072
7	Azerbaijan	44457945	599364	3486381	94631584	34745712	40.143105	47.576927000000005
8	Bangladesh	106252764	1595890	5891990	409306256	95429037	23.684994	90.35633100000001
9	Belarus	49479699	351043	6286097	1726028	569002	53.709807	27.953389
10	Belgium	142002909	3867787	13906092	318447409	85401701	50.503887	4.469936
11	Belize	2085662	51778	0	1397993	62360	17.189877	-88.49765
12	Bolivia	42699234	2001732	1452302	51536526	13246766	-16.290154	-63.588653
13	Bosnia and Herzegovina	25990375	1032779	972590	491135	101224	43.915886	17.679076000000002
14	Bulgaria	50351654	2000125	2795296	67411283	19276622	42.733883	25.48583
15	Cambodia	874098	5476	1133685	158830930	56968803	12.565679	104.990963
16	Canada	158094289	3658029	34550157	1017786864	100556386	56.130366	-106.34677099999999
17	Chile	156237359	3748542	14601545	1163337007	445804132	-35.675146999999996	-71.542969
18	Colombia	399714726	10466737	16748388	274980101	84241444	4.570868	-74.297333
19	Costa Rica	37087062	489579	1009995	9890701	3668576	9.748917	-83.753428
20	Croatia	45526114	960862	1999818	61111148	13952209	45.1	15.2
21	Cuba	10057970	65956	4283455	7972472	0	21.521757	-77.78116700000001
22	Cyprus	7135599	39339	6863892	9494128	2657430	35.126413	33.429859
23	Denmark	36708421	365370	32092053	170472376	58713785	56.26392	9.501785
24	Dominican Republic	40118415	529987	0	108403803	30391925	18.735692999999998	-70.162651
25	Ecuador	52037740	2859217	1359061	53206999	13377580	-1.8312389999999998	-78.183406
26	Estonia	12934768	117660	1422546	37703477	10471833	58.595271999999994	25.013607
27	Ethiopia	31715852	467579	2711918	49501831	0	9.145	40.489672999999996
28	Fiji	14031	393	103943	238983	0	-16.578193	179.414413

6 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.