Introduction to databases in practice with PostgreSQL and Python

Pierre Senellart

4 September 2020
PSL Preparatory Week

In this lab session, we will practice manipulating real-world datasets with PostgreSQL, a popular and powerful open-source relational database management system. You will interact with this DBMS using its command line interface (`psql`), as well as a Python program through the `psycopg2` connector.

Prerequisites

You are given access to a pre-installed and pre-configured instance of PostgreSQL. To access it, you will need the following parameters:

Username: *you have received this by email*

Password: *you have received this by email*

Server: `senellart.com`

Database: `psl`

For this lab session, you need to have installed on your computer the PostgreSQL command-line client, `psql`. You can follow the instructions to install it on various operating systems from
`https://blog.timescale.com/tutorials/how-to-install-psql-on-mac-ubuntu-debian-windows/`

In addition, you will need a working Python development environment, with the following two packages installed:

psycopg2 for connecting to the PostgreSQL database;

scipy to compute some statistics.

Both can easily be installed with the package manager of your choice (`pip`, `conda`, `easy_install`, etc.).

Throughout this lab session, do not hesitate to refer to the following documentation:

- `https://www.postgresql.org/docs/11/index.html` the documentation of the PostgreSQL DBMS. A large part of the documentation is also accessible from the `psql` tool by using the `\help` command followed by a SQL keyword.
- `https://www.psycopg.org/docs/usage.html` the documentation of the `psycopg2` package. You will only need to read the first two sections of the documentation ("Basic module usage" and "Passing parameters to SQL queries") and possibly follow hyperlinks for detailed documentation of the different functions.

Datasets

You will use the following datasets:

- The Google Covid-19 mobility dataset, accessible from `https://www.google.com/covid19/mobility/` that establishes some *mobility* index throughout the world based on Android geolocation data. For different types of locations (businesses, parks, transit stops, workplaces, residential), and different regions over the world (countries and subdivisions thereof), the dataset gives, for a given date, the relative variation of mobility (in percentage points) compared to a period in early 2020, before any lockdown or restrictions were implemented. The documentation of this dataset is at: `https://www.google.com/covid19/mobility/data_documentation.html` Since this dataset is quite large, you will not need to download it yourself: it is already available on the accessible database.
- Oxford University's Covid-19 Government Response Stringency Index, from `https://www.bsg.ox.ac.uk/research/research-projects/coronavirus-government-response-tracker` (look for the Download link). It provides a temporal *stringency* index for every country that indicates how strong the restrictions (social distancing, mass gathering bans, lockdowns) were at a given date. You will need to download this dataset and to import it into the database.

Your main target will be to explore these two datasets and determine how much correlation there is between people's mobility (especially, at work, in transit) and government rules.

As these two datasets use two different standards for indexing countries (respectively, the ISO 3166-1 2-letter code and the ISO 3166-1 3-letter code), you will need an additional dataset to join these two. The official source is from ISO `https://www.iso.org/publication/PUB500001.html` but costs 300 CHF! Fortunately, an equivalent dataset is provided for free on Github by Luke Duncalfe at `https://github.com/lukes/ISO-3166-Countries-with-Regional-Codes/blob/master/all/all.csv`. This also comes with information about continents and subcontinents, which will be helful.

1 Exploring the mobility dataset

1. Connect to the PostgreSQL database using the `psql` tool and the provided parameters. The command line to launch `psql` should be:

   `psql -h` *server database username*

   after which the *password* will be requested.

   Once connected, you can issue SQL statements, and use a few extra commands that start with a '\' character, the most useful of which is "`\d`" that provides you with a list of all tables in the database, and "`\d` *`tablename`*" that shows you the schema of the table *tablename*. See "`\?`" and the PostgreSQL documentation for other such commands. If you issue SQL statements that take too long, you can use CTRL+C to interrupt them. If a command returns many lines, a pager will open, which can be exited by typing "`q`".

2. Using `\d`, determine what the schema of the mobility table is.

3. Using a SQL statement, determine how many entries there are in the mobility table.

4. Using a SQL statement, take a look at the first 10 entries in the mobility table.

5. Using a SQL statement, determine what distinct values the `sub_region_1` and `sub_region_2` columns contain for France. What do you think the entries corresponding to the whole of France (and not a subdivision thereof) look like?

6. Using a SQL statement, determine the minimum and maximum dates in the mobility table.

7. Using a SQL statement, determine the number of distinct countries referenced in the mobility table.

8. Using a SQL statement, find the only country in the world where the average transit mobility is higher in the dataset than it was in early 2020.

9. Using a SQL statement, determine the daily workplace and transit mobility in the Île-de-France region over the whole period recorded in the dataset. Look for high- and low-mobility days, check this makes sense.

10. Continue exploring this dataset a bit! It has a wealth of information.

2 Importing the ISO 3166-1 dataset

1. Open the ISO 3166-1 dataset in a spreadsheet software or text editor, and study its content.

2. Create a table in the database with a `CREATE TABLE` SQL statement that reproduces as faithfully as possible the structure of this dataset. Do not forget primary keys and unique constraints.
3. Using the `\copy` command, copy the dataset from your local machine within the table you created in the database. Check the documentation of this command from `https://www.postgresql.org/docs/11/sql-copy.html` (this discusses the `COPY` SQL statement, which is subtly different from the `\copy psql` command you must use; but the arguments of both commands are the same).
4. Using a SQL statement and both mobility and ISO 3166-1 tables, order all countries in Europe from those with least workplace mobility to highest mobility, averaged over the whole dataset. What is the country that had the highest impact in terms of mobility to the workplace? the lowest?
5. Using a SQL statement and both mobility and ISO 3166-1 tables, order all countries in Europe from those with least workplace mobility to highest mobility, averaged over the second half of August. Is there any country that seems to have stopped favoring remote work?

3 Importing the stringency dataset

1. Open the stringency dataset in a spreadsheet software or text editor, and study its content.
2. Create a table in the database with a `CREATE TABLE` SQL statement that reproduces as faithfully as possible the structure of this dataset. Do not forget primary keys and unique constraints.
3. Using the `\copy` command, copy the dataset from your local machine within the table you created in the database.
4. Using a SQL statement, order countries according to their average stringency index over the whole period recorded in the dataset. Which country was the strictest? Which country of non-negligible size was the most lenient?
5. Using a SQL statement, order countries according to their average stringency index over the last half of August. Has anything changed compared to the previous results?
6. Continue exploring this dataset a bit! It has a wealth of information.

4 Correlating mobility and stringency

1. Using a SQL statement that joins all three tables, determine for every *world subregion* (in the sense of those recorded in the ISO 3166-1 dataset) its average stringency and workplace mobility index. Which subregions do stand out?

Note that the analysis is a bit flawed, as we are averaging countries of very different sizes. A better average would weigh in the population of every country, but this is not in our current datasets. Feel free to add another dataset to fix this!

2. You are now going to write a Python program to compute the correlation between mobility and stringency. For this, you are going to use the `psycopg2` package to connect to the database and issue SQL statements, and the `scipy.stats` package to compute the *Spearman rank-order correlation coefficent* (and its associated p-value), using `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html`. We need to compute this correlation coefficient between two lists: one containing the mobility indices and one containing the stringency indices. Both lists should have the same length, and the kth elements of both lists should refer to the same country.

   Are government restrictions and workplace mobility correlated?

3. Do the same analysis for every continent, instead of for the whole world. Which continent has the least correlation?

5 To go further

You should now have all the tools to manage, process, and integrate data at scale. For example, you could:

- load the WHO dataset of coronavirus cases and deaths, from `https://data.humdata.org/dataset/coronavirus-covid-19-cases-and-deaths` and look for correlations with mobility and stringency indexes;
- do finer analyses by looking for correlations within fixed period of times (e.g., weeeks);
- build data visualizations integrating these datasets.