

# ETL Project:

## Real Estate Market and the Impact of COVID-19

19.12.2020

### Group Members:

**Emilia** Lubanska, **Jorge** Arriola Villafuerte, **Marina** Ercoli, **Monica** Lin, **Ulisses** Pinto

## Overview

This is a continuation of the work we did in Project #1. Back then, we retrieved and transformed data sets regarding two areas:

- 1) the progression of COVID-19 cases in Ontario;
- 2) real estate indicators in Ontario.

Our objective for Project #1 was to assess whether COVID-19 had an impact on real state in Ontario. For project #2, we have added additional data sets that now give us information about how the price and number of houses sold in British Columbia has fluctuated, as well as how the overall house price index in Canada has changed.

The final output of our project was the creation of a database that comprised all the data sets that we have retrieved to date. Of course, these data sets have been transformed into data frames that provide only the information that we find the most valuable, and that can be compared, merged and modified via queries to generate insights.

In our master branch, you will be able to find:

- 1) the Entity Relationship Diagram (ERD) for our database;
- 2) the PostgreSQL schema that was generated based on our ERD;
- 3) a Jupyter notebook showing the process we followed to load our data frames to our data base;
- 4) a SQL file where we run some test queries to ensure that the loading of our data frames was performed successfully.

Lastly, Our GitHub is divided in a way that the viewer can easily find the files we used in each part of the ETL process. These are the branches we created:

- Extraction\_2
- Transformation
- Loading
- Extraction (*not used*)\*

*\*Please, note that there was an issue with our original 'Extraction' branch. After discussing it with a TA, we decided to create an 'Extraction\_2' branch that we used to save our CSV files.*

# ETL Process

## I. Extract

These are the data sources that we used:

### Original data sources

- 1) COVID-2019 cases in Ontario (CSV):  
<https://data.ontario.ca/dataset/confirmed-positive-cases-of-covid-19-in-ontario/resource/455fd63b-603d-4608-8216-7d8647f43350>
- 2) Prices and number of houses sold in Ontario 2019/2020 (API):  
<https://www.quandl.com/data/CMHC-Canadian-Mortgage-and-Housing-Corporation/documentation?anchor=about>
- 3) Canadian Interest Rates - 2019/2020 (CSV):  
<https://www.bankofcanada.ca/rates/interest-rates/canadian-interest-rates/>

### New data sources

- 4) Statistics Canada. Table 18-10-0205-01 New housing price index, monthly (Web scraping):  
<https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1810020501&pickMembers%5B0%5D=1.17&cubeTimeFrame.startMonth=01&cubeTimeFrame.startYear=2019&cubeTimeFrame.endMonth=12&cubeTimeFrame.endYear=2020&referencePeriods=20190101%2C20201201>
- 5) Prices and number of houses sold in British Columbia (API):  
[https://www.quandl.com/data/CMHC/HPPU50\\_BC-Average-Median-and-Price-Percentiles-for-Unabsorbed-Homeowner-and-Condominium-Units-Provinces-British-Columbia](https://www.quandl.com/data/CMHC/HPPU50_BC-Average-Median-and-Price-Percentiles-for-Unabsorbed-Homeowner-and-Condominium-Units-Provinces-British-Columbia)

## Extraction process

### Type: CSV

We saved three CSV datasets under the 'Extraction 2' branch. Below is an example of the code we used to retrieve these data sets using Jupyter notebooks:

```
: #Retrieving CSV file
file_path = os.path.join("../2.CSV_Files/2.1.conposcovidloc.csv")
datafile_df = pd.read_csv(file_path)
#Changing case_reported_date to data format
datafile_df['Case_Reported_Date'] = pd.to_datetime(datafile_df['Case_Reported_Date'])
datafile_df.head()
```

### Type: API

We retrieved datasets from Quandl.com. Below, you can see the code snippet we used to retrieve this API.

```
url = 'https://www.quandl.com/api/v3/datasets/CMHC/HPPA50_ON.json?api_key=9aGzLsxCkNH7a9NEikI5'
response = requests.get(url).json()
pprint(response)
```

### Type: Web Scraping

We retrieved our data from the Statistic Canada site. For this section, we used the BeautifulSoup library to parse HTML file, and the Pandas library to create a data frame with the retrieved data.

```
] import pandas as pd
from bs4 import BeautifulSoup
import requests
from webdriver_manager.chrome import ChromeDriverManager
# How to cite: Statistics Canada. Table 18-10-0205-01 New housing price index, monthly

]: url = 'https://www150.statcan.gc.ca/t1/tb11/en/tv.action?pid=1810020501&pickMembers%5B0%5D=1.17&cubeTimeFrame.startMonth=01&cubeTimeFrame.startYear=2019&cubeTimeFrame.endMonth=12&cubeTimeFrame.endYear=2020&referencePeriods=20190101%2C20201201'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
```

### III. Transform

The transformation step required us to perform a couple of actions:

- 1) Erase columns with information that was not relevant to us: For instance, when determining the number of houses sold in Ontario, the 'type' of house/construction was not relevant for us. Hence, we dropped this column.
- 2) Compile/Aggregate information: For example, the Government of Ontario's data set for COVID-19 cases provides information for each one of the patients that got a positive result for COVID. We had to group the data by month and use the count function to get aggregate figures of how the number of COVID cases has fluctuated throughout 2020.
- 3) Choose a column to do future queries/joins/graphs: We had to identify which column or variable we could use to easily group tables. We decided that all of our data frames would have a 'date' column that showed the date information in the same format and structure 'mm/yyyy'. This proved to be extremely useful in the loading stage, as the date column allowed us to establish relationships between our data frames when creating the ERD.

This is an example of our cleaning process:

- 1) *Cleaning a data frame showing the average price & number of houses/units sold in Ontario in 2019:*

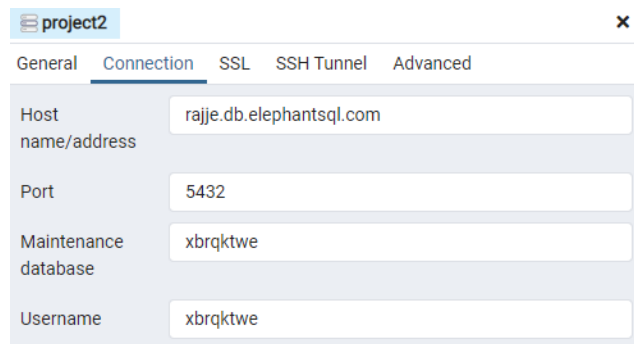
```
In [6]: response_df = pd.DataFrame(response['dataset']['data'])
response_df.columns = response['dataset']['column_names']
response_final_2019 = response_df.loc[(response_df['Date'] > '2019-01-01') & (response_df['Date'] < '2019-12-31'), ['Date', 'Average', 'Units']]
response_final_2019['Date'] = pd.to_datetime(response_final_2019.Date, format='%Y-%m-%d')
response_final_2019['Date'] = response_final_2019['Date'].dt.strftime('%m/%Y')
response_final_2019_sort = response_final_2019.sort_values(by='Date', ascending=True)
response_final_2019_sort = response_final_2019_sort.reset_index(drop=True)
response_final_2019_sort
```

```
Out[6]:
```

	Date	Average	Units
0	01/2019	952417.0	1544.0
1	02/2019	969348.0	1370.0
2	03/2019	822919.0	1090.0
3	04/2019	781375.0	1537.0

## IV. Load

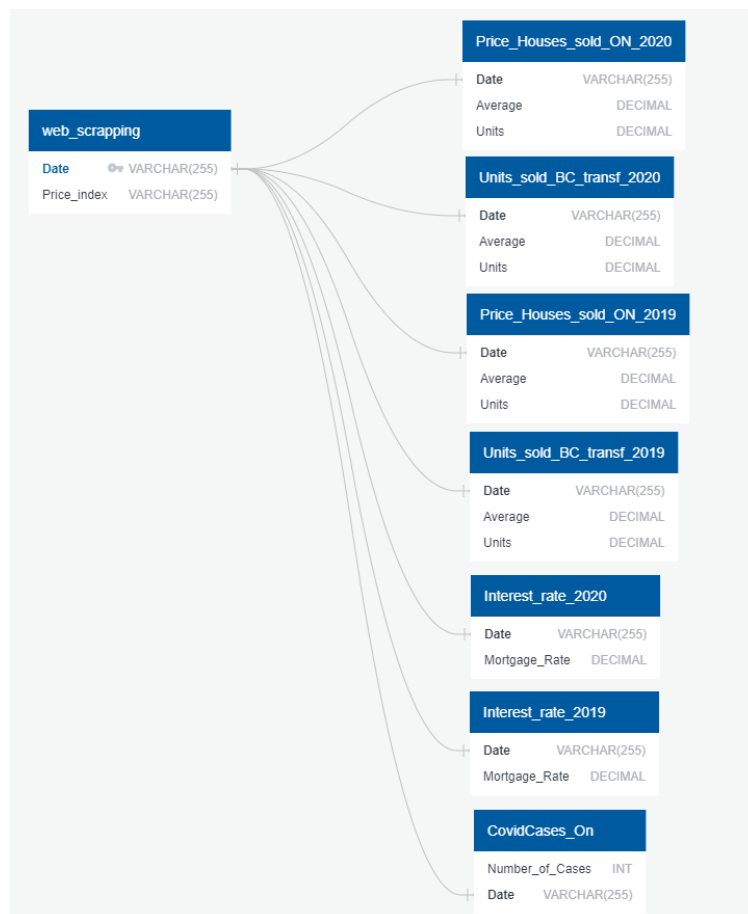
- 1) We created an ElephantSQL server. Each member of our group linked this server to our PG Admin platforms:



The screenshot shows the 'Connection' tab of the ElephantSQL configuration window for a project named 'project2'. The fields are as follows:

Field	Value
Host name/address	rajje.db.elephantsql.com
Port	5432
Maintenance database	xbrqktwe
Username	xbrqktwe

- 2) We used Quick DBD to map out an ERD diagram showing the relation between all of our data frames. We then transferred the PostgreSQL schema generated by Quick DBD to PG Admin. This schema can be found in our main branch as 'PostgreSQL schema.sql'. Below you can observe the ERD diagram that we created:



- 3) After the schema was created, we generated a 'final' Jupyter notebook. In this notebook, we first called all the data frames that we generated during the 'transform' stage using the magic function '%run'. We used the magic function 8 times, as we created 8 data frames:

```
In [3]: # Get final dataset from previous notebooks
%run Transform/Interest_rate_2019.ipynb
```

```
In [4]: # Call final table
MortgageRate2019
```

```
Out[4]:
```

	Date	Estimated variable mortgage rate
0	01/2019	2.7680
1	02/2019	2.8075
2	03/2019	2.8875
3	04/2019	2.8800
4	05/2019	2.8820
5	06/2019	2.9000
6	07/2019	2.9000
7	08/2019	2.9000
8	09/2019	2.9000
9	10/2019	2.9000

- 4) Once all of our data frames were retrieved in this 'final' Jupyter notebook, we coded our engine to connect with our ElephantSQL server:

```
In [21]: Elep_engine=create_engine("postgres://xbrqktwe:E_nywn4IJhz89kqxwG3M498kv8qnq4Z6@rajje.db.elephantsql.com:5432/xbrqktwe")
```

- 5) Finally, we loaded each of our data frames to our database:

```
In [21]: Elep_engine=create_engine("postgres://xbrqktwe:E_nywn4IJhz89kqxwG3M498kv8qnq4Z6@rajje.db.elephantsql.com:5432/xbrqktwe")
```

```
In [22]: resultsDF.to_sql(name="web_scrapping",schema="public", index=False, if_exists="replace", method="multi", con=Elep_engine)
```

```
In [23]: response_final_2019_sort.to_sql(name="Units_sold_BC_transf_2019",schema="public", index=False, if_exists="replace", method="multi", con=Elep_engine)
```

```
In [24]: response_final_2020_sort.to_sql(name="Units_sold_BC_transf_2020",schema="public", index=False, if_exists="replace", method="multi", con=Elep_engine)
```

```
In [25]: Reported_cases_per_month_ON.to_sql(name="CovidCases_On",schema="public", index=False, if_exists="replace", method="multi", con=Elep_engine)
```

```
In [26]: MortgageRate2019.to_sql(name="Interest_rate_2019",schema="public", index=False, if_exists="replace", method="multi", con=Elep_engine)
```

```
In [27]: MortgageRate2020.to_sql(name="Interest_rate_2020",schema="public", index=False, if_exists="replace", method="multi", con=Elep_engine)
```

```
In [28]: response_final_2019_sort.to_sql(name="Price_Houses_sold_ON_2019",schema="public", index=False, if_exists="replace", method="multi", con=Elep_engine)
```

## Discussion / Limitations /Next Steps

- 1) Using the magic function '%run' presented a few challenges. We had to modify the paths to our CSV files to ensure that they worked as required. Using the magic function to call our web scrapping data frame was particularly challenging. Along with a TA, we determined that we had to load a new library ('nest\_asyncio') to ensure that the magic function worked correctly.
- 2) Using GitHub correctly was probably more time consuming for us than the actual ETL requirements for this project. We spent a good portion of the time available solving GitHub errors and ensuring that we were using branches and that our files were organized. As this is a platform that is used widely, and the importance of learning it was highlighted in class, we wanted to make sure we continued learning how to work collaboratively in GitHub.
- 3) We found the web scraping process to be time consuming as many commercial websites protect their data against this form of acquisition. There were some very interesting webpages with information that we would have liked to include in our project, but that did not allow us to retrieve their information. We eventually decided to use an open and free accessible data source.
- 4) Though this was not a requirement for Project #2, we would still like to use GeoPandas down the road. We would be able to provide more granular answers with better visualizations. For instance, we would map out the number of Covid-19 cases, and number/price of houses being sold per city within Ontario and BC.
- 5) Analysis was not a part of this project. However, as a next step we would like to use queries to make changes to our tables, join relevant tables and use functions to generate insights using our final Jupyter Notebook and SQLAlchemy.