**Project Report | ETL Project**

**Compiled By: Eric Meyer, Michelle Simek and Kristen Scaletta**

At the end of the week, your team will submit a Final Report that describes the following:

\* **\*\*E\*\***xtract: your original data sources and how the data was formatted (CSV, JSON, pgAdmin 4, etc).

\* **\*\*T\*\***ransform: what data cleaning or transformation was required.

\* **\*\*L\*\***oad: the final database, tables/collections, and why this was chosen.

**Overview**: For this project, we compared data on city of Chicago sidewalk permits with restaurant citations (see the technical report for data specifics). We were interested in exploring any relationships between sidewalk permits and restaurant citations so needed to extract, transform and load the data into a SQL database to do so. We will perform the analysis at a later phase—the scope of this project was to prepare the data to be able to use it for this analysis.

**Extract**: We obtained two .csv files from open source City of Chicago data (<https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5> and https://data.cityofchicago.org/Community-Economic-Development/Sidewalk-Cafe-Permits-Current/qnjv-hj2q respectively) and loaded them into data frames in Jupyter Lab.

**Transform**: In order to make the data usable for our analysis, we performed a number data cleaning and transformation processes. These are described in greater detail in the related technical report. At a high level, we completed the following:

1. For each dataframe we loaded, we:
   1. Removed columns not needed for our analysis
   2. Updated the Address column to remove spaces in order to facilitate an inner join and renamed the column in one dataframe to match the other.
   3. Ensured the data ranges matched for data in each dataset. The Food Inspection Data had a broader date range so we added a new column to that dataframe that updated the format of the Inspection Date column to be date-time, and filtered the data by 2020 or greater.
2. We then joined the two dataframes on the address column and removed any null values in our new joined dataframe.
3. We wanted to build a relational database, so using the data in our joined dataframe, we created three additional dataframes—one with the ID number and description of violations, one with the ID number and description of the inspection type, and one with the ID number and description of the inspection results. (See the technical report for more specifics on how we did this.)
4. To keep our tables clean, we then replaced the text values of violations, results and inspection type with the unique ID for each in new columns in our joined table.
5. We then removed the columns that contained the text values of violations, results and inspection type, resulting on our fouth and final dataframe.

**Load**:

To load the data, we created a final database called “ADD NAME” that contained four tables in PGAdmin. The four tables are called ADD and they correspond to each of the four dataframes we transformed for this project. We used PGAdmin because our tables related to one another and we knew the data type of each column as well as column name and contents, so we wanted to use SQL to build a relational database. Once the empty tables and database existed, we used SqlAlchemy to load the dataframes into the tables. We then performed simple SQL queries to verify the tables loaded correctly and worked as intended.