# **Chicago Restaurant Food Inspection and Yelp Data**

## ETL Group Project

David Kloepper

Scott Otto

# **Extract**

For the extraction element of our project, we chose to utilize Chicago Food Inspection data sourced from Kaggle (<https://www.kaggle.com/chicago/chicago-food-inspections>) which gave us data from January 1, 2010 to the present. This information was retrieved in .csv format.

We thought it would be interesting if we were to match that information with review data from Yelp. We signed up for the Yelp API and proceeded to construct two different API requests. The Business Match API call (see below) was used to obtain the business ID, and the Business Details API call utilized that business ID to return the full business data, along with rating information for that specific location.

Yelp API - https://www.yelp.com/developers/documentation/v3/get\_started­

API to return business ID: Business Match – uses location data to return a business name.

|  |  |  |
| --- | --- | --- |
| Business Match | [/businesses/matches](https://www.yelp.com/developers/documentation/v3/business_match) | Find the Yelp business that matches an exact input location. Use this to match business data from other sources with Yelp businesses. |

API to return full business information: Business Details - uses business ID to return full business information, along with rating/review information.

|  |  |  |
| --- | --- | --- |
| Business Details | [/businesses/{id}](https://www.yelp.com/developers/documentation/v3/business) | Get rich business data, such as name, address, phone number, photos, Yelp rating, price levels and hours of operation. |

# **Transform**

Below are some of the steps used to clean up the data to make it more useful:

**Kaggle Data**

* Removed anything that had a license ID of zero from the inspection data retrieved from Kaggle.
* Reduced the data to the “Restaurant” category in the inspection data retrieved from Kaggle to eliminate inspection data on daycares, hotdog carts, etc.
* Once the restaurants were isolated, the results were filtered down to the most recent inspection row for each restaurant.
* Once the most recent were returned, those were further reduced to rows where inspection dates were greater than or equal to 1/1/2018.
* Renamed columns to more database friendly names

**Yelp API Data**

* For the data returned from the API requests, the location data was split to create Address, City, State and Zip fields. This may prove to be useful if we were to have a larger dataset that was outside of the greater Chicago area (e.g. if we did the top 20 major markets in the US).
* Columns of the API request dataframe were reordered and renamed (to give more meaningful names)
* To get the data to move to MySQL easier, we incremented the index in each dataframe by one because SQL does not allow for a zero value index.

# **Load**

We created a database with three tables in MySQL.

Database: chicago\_yelp

Table1: food\_inspection

Table2: yelp\_business\_info

Table3: business\_info

* food\_inspection houses the food inspection data retrieved from Kaggle, which has been reduced to restaurants most recent inspections 1/1/18 or later.
* yelp\_business\_info houses the Yelp business information retrieved from the Business Detail API call for restaurants identified in the city of Chicago
* business\_info was created to provide a cleaner connection between a business license number and the corresponding Yelp ID.

We utilized VarChar, Float, Decimal, Integer, DateTime and Boolean field types, as well as primary key auto-incrementing when creating the tables.

We pushed our python dataframes to the appropriate MySQL table using SQLAlchemy.