**ETL Project**

**2/23/2019**

***Chane Donnelly, Nilay Patel, Lucy Shaw, Toni Stapleton***

***Chane:***

Extract:

Found Chicago liquor license data on the City of Chicago Data Portal website. I narrowed down the business licenses to only information on liquor licenses in Chicago. The exact website can be found at: https://data.cityofchicago.org/Community-Economic-Development/Business-Licenses-Current-Liquor-and-Public-Places/nrmj-3kcf. Once I found the data set that would suffice for this project I downloaded the data set to an excel CSV file.

Transform:

Opened at Jupyter notebook and imported the following dependencies: pandas, pymongo, and numpy. I used pandas to read the CSV file and bring the data into my Jupyter notebook to create the liquor\_license\_df data frame. I renamed all of my columns in the data frame to lowercase. Next, I removed all of the data that was located outside the city of Chicago as well as replaced all “NaN” values with “ ” . As a result of running the fillna function I noticed that my latitude and longitude values were rounded to 4 decimal places.

Load:

I created a connection to my local host using pymongo. Then created a database called restaurantsDB and a collaction called db.liquor. I put the liquor\_license\_df into a list of dictionaries and inserted these values into the db.liquor collection inside the restaurantsDB database. We chose to use MongoDB due to the fact that our data was not all structured the same.

***Nilay:***

Extract:

We used the Business Licenses dataset from the City of Chicago Data Portal. The data was contained in a .csv file and it included all the licenses the City of Chicago has granted from 2002 onwards. Included in this dataset was the license information (application data, application type, issue date, description of business activity, expiration date) and the basic information about the business (name, legal name, address, latitude, longitude).

Transform:

1) The first thing to do for the transformation was to select out just the restaurants. The simplest way was to use the “BUSINESS ACTIVITY ID” to select only those data that was for restaurants. There were 3 such ids (775 for “Retail Sales of Perishable Foods”, 782 for “Sale of Food Prepared Onsite Without Dining Area”, and 735 for “Preparation of Food and Dining on Premise With Dining Area”).

2) The next thing was we wanted a table of just active licenses and separate them out from the rest of the dataset. So we took the “LICENSE TERM EXPIRATION DATE” and converted it to datetime because the original dataset contained it as a string, and then created a new column “is\_active” with Boolean values that would be true if the date was greater than today or false otherwise.

3) To check whether this worked, I did a count of the unique values of the account number, which would be tied to a single business, and compared it with the count of the data filtered by “is\_active” equals True. It was clear that there were duplicates, and I guessed that was because businesses would renew licenses before their previous ones expired and so you can have one business entity with two valid licenses at the same time. I performed a groupby on the account number and a count and indeed most of them had only a single count, some had 2, and none of them had more than 2. To work around this, I sorted the expiration column so that the closest expiration date would be at the bottom and the dropped duplicates, reasoning that the duplicates for the closest expiration would be the old licenses still valid.

4) Finally, I created 3 tables from this transformed dataset.

Load:

The 3 tables were loaded to Mongodb, active\_licenses, licenses, and location data. The active\_licenses and licenses data would only have information about the license, entity name, issue date, expiration date, and business type. The active\_licenses would give information about currently active licenses and it would represent the complete information about restaurants currently active in Chicago. The licenses data would contain all the historical data as well, and from this we could look and find historical data (like number of newly issued licenses every year). The location data would give location information only for all the restaurants with active licenses.

There is a license id included in the active\_license and licenses table that would allow cross reference between those 2 tables, and then there is a active\_id in the location and active\_licenses table that allows for cross reference between those 2 tables.

***Lucy:***

Chicago Restaurant Inspections Data

1. Download .csv on Chicago restaurant inspection data from Kaggle
2. Open a jupyter notebook and import dependencies
   * Pandas
   * Pymongo
3. Using pandas, read in the .csv that you downloaded and convert into Dataframe
4. From the dataframe, select only the rows where:
   * City = Chicago
   * Facility Type = Restaurant
5. Select only the column headers required: AKA Name, License #, Facility Type, Inspection Date, Results, Latitude, Longitude
6. Rename the column headers as below:
   * AKA Name: name
   * License #: License #
   * Facility Type: Facility Type
   * Inspection Date: Inspection Date
   * Results: Results
   * Latitude: latitude
   * Longitude: longitude
7. Round the values in the latitude and longitude columns to four decimal places
8. Replace all NAN fields with a blank
9. Create connection to your local host
10. Create a database named restaurantsDB
11. Create a collection called inspections
12. Using the insert\_many function, insert the dataframe into the inspections collection on the restaurantsDB

***Toni:***

Extract: I used the Yelp API Business Search endpoint (<https://www.yelp.com/developers/documentation/v3/business_search>) to pull the top 1000 restaurants in Chicago sorted by number of Yelp ratings. To do this, my parameters included the key term ‘restaurants’ and the location ‘Chicago, IL’. This endpoint only returns 50 results at a time and pulls 1,000 results in total. As a result, I put the API request in a for-loop and changed the offset parameter for each loop to extract all 1,000 records.

Transform: I then cleansed the data by removing all restaurants not in Chicago, as there were some that were in Chicago suburbs. In addition, the price field was reported by varying numbers of dollar signs, I converted this to a number scale so that 1 is the lowest price level and 4 is the highest price level. I also rounded the latitude and longitude fields to 4 decimal places and filled the nan values with empty strings so that the nan would not be loaded into MongoDB.

Load: The data was then loaded into restaurantsDB, our group’s MongoDB database, using PyMongo. This collection is yelpdata.

***Final Notes***:

We tested joining on different fields. Every dataset has geographical coordinates, but this did not always provide for a good join. The two inspection datasets are able to join on license number. Thejoin on business name was not effective, due to small variations in each restaurant name in each dataset. We found joining on the address to be the most effective. If we were to continue, we would look into cleansing the address fields to further improve the join, perhaps only keep the street number and name and removing all “Street”, “St”, “Drive”, “Suite #”, etc. In addition, we could cleanse the business names of all unnecessary terms, punctuation, and/or numbers to try and improve the join.