ETL Project Report:

Kana Moore

Brandon Uhler

*Report Date: 07 March 2020*

**Objective and Scope:**

The goal of this project is to perform what is commonly referred to as ETL. ETL stands for **e**xtract, **t**ransform, and **l**oad. This is a practice that is crucial in creating data that is clean and usable for the purpose of analyzing, storing, etc. The process can be broken down into its individual parts: extracting or acquiring the data from a source or sources, transforming or cleaning it into a form that is more readable and usable, and finally loading the data into a database or table.

**Extract:**

Data is available in a variety of formats from a variety of sources. For this project, data was collected from Kaggle, a free-to-use website that allows the open sharing of databases that cover diverse topics, and NYC Open Data, a free-to-use website that contains various NYC datasets in various formats. The first dataset that was extracted outlines recent NYC property sales around the five boroughs (Manhattan, Brooklyn, The Bronx, Queens, and Staten Island). This dataset was downloaded from Kaggle (https://www.kaggle.com/new-york-city/nyc-property-sales)1, in the format of a CSV file. The second dataset that was extracted is a NYC tree census. This dataset contains information about the trees located throughout the five boroughs of NYC. This was downloaded from the NYC Open Data site (https://data.cityofnewyork.us/Environment/2015-Street-Tree-Census-Tree-Data/pi5s-9p35)2, in the format of a raw JSON file.

**Transform:**

To transform this data into a usable form, Pandas, a popular software library written for the Python programming language, was utilized. Pandas proved to be very effective here due to its ability to easily load data from a file into a pandas dataframe, where data can be manipulated into a form the user finds relevant to their analysis. First, the property sale CSV file was read into a pandas dataframe. The NYC property sales data only contained a borough ID and was missing the borough name. Therefore, a borough id dataframe was created and merged with a borough name dataframe in order to replace the id numbers with the corresponding names. Next, the tree census JSON file in its raw form is a series of nested dictionaries, and therefore a quick look through the raw file was required. The file was broken down into two sections: “meta” and “data.” The data could be easily loaded into the pandas dataframe, however column titles proved to be less straight-forward. An empty list was created, followed by a for loop that would cycle through the meta data and grab all the column titles and populate the list with them. This yielded the dataframe with the correct corresponding columns titles.

Not all the columns are relevant at this time. Therefore the next step in the cleaning was to drop any columns that will not be used later. After dropping irrelevant columns, the dataset we refer to as “property sales,” contained the following columns: id (primary key), borough id, borough, neighborhood, address, and sale price. In the dataset we refer to as “tree census,” the columns that are present include tree id (primary key), tree health, zip code, borough, and address. It was recognized that the two datasets related to each other via information such as boroughs and address. As a result, the loading stage would greatly benefit from a relational database program, i.e. PostgreSQL. In order for the data to be placed in SQL tables (in the hopes of performing a join later), the columns needed to be renamed to match the column names of tables that will eventually be made using PostgreSQL. Additionally, as a precaution, any null values in our dataframes were dropped.

**Load:**

After we were satisfied with the format of our dataframes, they were loaded into tables in Postgres (pgAdmin4) by forming a connection to the jupyter notebook through the creation of an engine (ORM). This involved creating tables in PostgreSQL that we can load our data into using Pandas. Each table is assigned a primary key (typically id) for relational purposes. We created tables with the number and names of columns present in the dataframes we previously created.

**Conclusion:**

The goal(s) of the project were completed successfully. The motivation was not to perform an analysis on the data itself, but rather to prepare said data in such a way that an analysis would not be limited by any inconsistencies or formatting errors due to how the data is stored. Now that these two datasets have had ETL performed on them, they are potentially ready for an analysis. A possible use for the cleaned datasets is to see if there are any correlations between trees located in NYC, and the property values at various locations. For example, are there more trees present in high-value areas? Is the health of the trees correlated with the property values? A potential hypothesis to test here would be that if the trees are located in an area with higher value properties, then the health of the trees would be better than that of trees in low value areas. To accomplish this, a query was also written to demonstrate that an inner join could be performed on ‘address,’ providing information on tree location and the value of the properties where it is located.

**Keywords:** Python, Pandas, Jupyter Notebook (or Jupyter Lab), SQL, Postgres, dataframe, dataset, object-relational mapping (ORM), engine, primary key

**Sources:**

1. [*Via Kaggle*]Dataset is owned, maintained and provided by the City of New York.
2. [*Via NYC Open Data*]Dataset is owned, maintained and provided by the New York City DPR (Department of Parks and Recreation).