**ETL Project: Renewables Revisited**

Graham Penrose

Scott Clark

12/21/19

The goal of this project was to build out a comprehensive database for how energy flows in California. We established a solid set of data on renewable energy in California with our first class project, but we wanted to continue to expand the work. We sought out additional ways to gather data based on the skills taught in class with three particular goals in mind: find renewable energy information for after 2017, incorporate non-renewable energy production data, and identify hourly demand data.

We were successful in locating the data that we sought from the California Independent System Operator (CASIO) and the US Energy Information Administration (USEIA). CAISO provides more than 80% of energy in California, so it serves as a good proxy for how California is making progress on increasing renewable energy production. We pulled production data from CAISO by scraping but were not able to access demand data without API access, that we have not been able to obtain permission to obtain. However, CAISO reports information to the federal government, so by using USEIA API we were able to locate the missing information. The data collected was transformed and loaded into a relational PostgreSQL database.

**Extract**

Demand Data: We received USEIA API access and used it to pull all the available hourly demand data for CAISO specifically. The JSON data was in an easy to use form with all hourly data listed within a single nested dictionary key. We used the pandas module to read the data into a dataframe.

Production Data: Obtaining the comprehensive production data required scraping daily report files on the CAISO website for the text using beautifulsoup, splinter, and pandas. Scraping navigated to a webpage that holds daily reports based off the date given in the url string. The scraping worked, but there were challenges with time-out errors. In a loop, the code hit the CAISO website about 1300 times and turned the displayed data table into two dataframes. Sometimes, the code would still error out due to the website not responding quick enough for the default "requests" module. This was address by using "with requests.get(url) as r:" to open and close the session within every loop, which seemed to help. Sometimes it would still error out, but waiting a minute or two then re-running would allow the loop to finish.

In both demand and production data, there was a stretch of data missing from 2019. It is unclear why this data was missing, but if CAISO failed to produce data for those dates, it makes sense that it was also missing from USEIA datasets. We obtained data from July 1, 2015 to the current day from the above sources.

**Transform**

Demand Data: There were two minor issues with the USEIA data. The timestamp was not in a format that pandas would recognize, as it had a string on the end of it that did not pertain to time. The string was split off and the timestamp worked. The second issue was that there were a handful of duplicate timestamps in the USEIA dataset. As timestamps were used as a primary key in our database, duplicates prevented us from loading our data into pdAdmin. This was resolved by using drop\_duplicates, keeping only one of the timestamps.

Production Data: Of of the dates we collected, there were only two days that didn't have their table formatted the same way. When we concatenated all of the tables scraped, having these two tables with very odd formatting would completely mess up the final table, which was created from stacking all of the daily scraped tables. We found the dates with the inconsistent formatting in the dataframe and added an "if" statement in the scraping loop to skip scraping for that date. Filling nulls and combining numerical rows for calculated columns was also necessary.

For both production and demand data, we used the datetime module to break timestamps into the date and hour categories we needed for our relational database. We structured into First Normal form by removing duplicates and ensuring a single value per field.

**Load**

APostgreSQL database with two schema and five tables was built.

Production:

* + Hourly renewable production by renewable energy source (solar, wind, etc.)
  + Hourly production by all energy sources (solar, wind, nuclear, etc.)
  + Daily percent of production provided by renewable sources
  + Daily percentage of demand met by renewable sources

Demand:

* + Hourly demand

We related all tables on the timestamp, which served as the primary key. We also used flask to create a site where demand and production data could be pulled with API call of individual dates or date ranges.