Project 2: Extract, Transform, and Load

Jongyon Kim, Emily McGrew, Joe Scuteri, Jenny Shea

For this project our team was asked to create a database using 2 sources, transform the data and load it into a database. We chose to use an SQLite database of roughly 18,000 music reviews from the online magazine Pitchfork and a JSON file of Spotify artist information which we created using Spotify's API.

The Pitchfork review database is available on Kaggle, https://www.kaggle.com/nolanbconaway/pitchfork-data. This data was created by Nolan Conaway who scraped the Pitchfork site for the review data. It is a SQLite database with 7 tables of data on artist reviews, genres, and music labels spanning 22 years. We used SQLalchemy to get the table names and then merged the reviews, content, and genre tables together and removed duplicates and irrelevant columns such as day of publication. The merged table uploaded into PGAdmin. Next, we created a list of artists for the Spotify API call.

Using the Spotify API documentation, https://developer.spotify.com/documentation/web-api/, and the Spotipy python library, https://spotipy.readthedocs.io/en/2.19.0/. Spotipy is a python library for accessing Spotify data. We created a dictionary containing the artist name, popularity score, Spotify genres, and number of followers for the artists on the Pitchfork review list. We also printed out a list containing the artists on the Pitchfork list who were not listed on Spotify. Possible reasons why they may not be listed include not being on Spotify, keying errors, and collaborations between artists that may be classified differently on Spotify. After dropping the duplicate listings, we had a total dataset of about 8,000 artists. In total the API call took 33 minutes to run.

We created 2 tables in PGAdmin to store our dataframes using the column names from the Pitchfork and Spotify dataframes. Both tables contain an artist name field. The Spotify artist field has a one-to-many relationship with the artist name is the Pitchfork table. The dataframes were loaded into PGAdmin using the .to_sql method The SQL format was chosen so an end user could use the relationship between the artist columns from both sources to understand artist popularity.