ETL Project – Group 13: What’s On Netflix

Alex Riner

Igor Pavlunin

Kaylon Young

Our primary data source was a csv file containing an up to date list of shows on Netflix. We chose this data source as it seemed that it was updated regularly, an important piece to capture Netflix’s often changing catalogue. The second data source was the open movie database api. This source gave us much more information on the individual shows than was in the Netflix csv file. And the api format meant that we would only have to pull down the information that was actually relevant to our project, instead of downloading or scraping information that would have to be tossed if it didn’t match up with what was in our primary data source.

The first dataset is found on the Kaggle website. It is in \*csv format, containing >6000 TV Shows and movies for the 2007-2020 period of time. We decided to keep all original twelve columns from the same dataset to preserve data integrity after merging with the second dataset (API OMDb). First of all, I’ve defined file location and load the file with Pandas DataFrame. Using “shape” and sorted functions I started exploring Netflix data – 6234 rows and 12 columns.

The next step is checking for Null values – five columns contained a number of Null fields. Then renamed columns for better perception and assigned proper data types using “astype” function. To prevent errors all NA values have been replaced with blank field ‘ ‘. The last clean-up was to drop duplicates in Show\_Title column and keep the first value with “drop\_dublicates” function. This step is done in order to use Show\_Titles column as a unique ID to merge with OMDb dataset. PG Connection not performed at this step since we had to merge with the OMDb dataset and table structure will be different.

After cleanup was performed on the Netflix dataset, it is time to gather data from the Open Movie database (OMDb). Using the Netflix dataframe, code loops through each movies title and searches Open Movie database for that title. Information is retrieved from the OMDb API, and each column is saved to their respective list. A total of 21 lists are created. Not every movie or TV show is found in the OMDb. After data has been retrieved for each movie from OMDb, all these lists are combined into a dictionary and then loaded into another dataframe.

One quirk observed in the OMDb is at times it will return a partial movie title instead of an exact movie title. For this reason, we dropped duplicates out of this open movie dataframe, since we will be using an existing column, show\_id as the primary key. If there is a duplicate row inside the open movie dataframe, it will create a duplicate value in the resulting joined dataframe. Then this dataframe will not be able to be added to Postgres because it will contain a duplicate show\_id.

After duplicates are dropped out of the open movie dataframe, a new dataframe, movies\_combined, is created by joining the Netflix dataframe to the open movie dataframe on title and show\_title. Another new dataframe, movies\_limited, is created with select columns from the movies\_combined dataframe. A quick check is run to make sure they are not duplicate records in the movies\_limited dataframe.

Postgres was chosen as our database. We chose Postgres because it is a relational database. It is also more familiar to team members. Before loading the movies\_limited dataframe into Postgres, a database and table must exist. In our project we are created a movies\_db database, and table is called netflix\_omdb. The netflix\_omdb table contains:

* + Show\_ID INTEGER PRIMARY KEY,
  + Show\_Type VARCHAR(3000) NOT NULL,
  + Show\_Title VARCHAR(3000),
  + Show\_Director VARCHAR(3000),
  + Show\_Cast VARCHAR(3000),
  + Show\_Country VARCHAR(3000),
  + Date\_Added DATE,
  + Release\_Year INTEGER,
  + Show\_Rating VARCHAR(3000),
  + Show\_Duration VARCHAR(3000) NOT NULL,
  + Listed\_In VARCHAR(3000) NOT NULL,
  + Show\_Description VARCHAR(3000) NOT NULL,
  + Awards VARCHAR(1000),
  + Genre VARCHAR(1000),
  + Language VARCHAR(1000),
  + Writer VARCHAR(1000),
  + IMDB\_ID VARCHAR(15),
  + IMDB\_Votes VARCHAR(20),
  + Rating\_IMDB VARCHAR(20),
  + Rating\_Rotten VARCHAR(20),
  + Rating\_Metacritic VARCHAR(20)

Care must be taken to make sure the column names are the same, and the datatypes allow for the data to be loaded into that column. The movies\_limited dataframe is loaded into Postgres using the to\_sql command.

A possible use for this database would be to look for additional movie information but restrict the movies to only those which can be seen on Netflix. If you lack any other streaming services it is a waste of time to be looking at movies that you will be unable to watch. On the other hand, looking at Neflix’s own list of movies lacks data that you may want when selecting what to watch, particularly ratings from entities other than Netflix like rotten tomatoes and metacritic. This could also be useful if one was wanting to do data analysis specifically on those movies that are in Netflix’s catalogue.