**ETL Project: Movie Remakes vs Their Originals - Final Report**

**By Kasie Wong, Jan 2020**

Project Description *(copied from github README)*:

Hollywood production houses love to reuse and recycle popular story ideas, often by telling the same story and updating it for contemporary audiences. While many embrace the nostalgia of watching familiar storylines, most movie buffs insist that the original movie is almost always better. So which is it?

This project is aims to prepare the preliminary data to answer this very question!

Data Sourcing:

Data was pulled from two main sources, Wikipedia’s web page lists of Film Remakes and Original Titles, and OMDb API. Wikipedia was used because it had by far the largest and most complete list of film remakes. OMDb API was used because it was the largest source of standardized movie data available that also included movie ratings.

Extraction & Transformation Process in Detail:

1. *Wikipedia Web Scraping*

Wikipedia’s list of movie remake and original film titles was so large that it was housed on two separate web pages, film titles starting alphabetically A-M, and film titles starting alphabetically N-Z. Each page was scraped using the pd.read\_html function and then appended together in various ways to create separate dataframes for remakes and originals, respectively. There can be multiple remakes per original movie. Unfortunately, the successive remake films were not stored in an easy to scrape format. For the purposes of this project, I primarily scraped the data for the first remake film available if there were multiple options. This is the main element I would have changed in my project had I been given more time. I also made sure to create another data frame that would serve as a reference table (“remakes\_originals\_df”), and tie each film remake with its corresponding original by creating a unique “movie ID” that combined movie titles and years.

1. *OMDb API*

I created a “movies\_all” data frame that contained the full list of remakes and originals. I then used this list to query all available IMDb data in the OMDb API. I stored the results in an “imdb\_df”. I had to use caution in this process and ensure that my indices were frequently reset to be “movie ID”. Otherwise, my API call would end up returning multiple duplicate records due to hidden indexing from previous list-appending.

1. *Transform & Export to CSV’s*

I performed multiple QC checks to remove duplicate records from each of my data frames using the “movie ID”. Columns were renamed for clarity and better data tracking. The results from these data frames were exported as several csv files.

1. *Create Postgres db & Load CSV’s*

I created a movies\_db in Postgres and wrote SQL queries to create five database tables. SQL Alchemy was then used to help load the CSV data into their respective tables. Movie ID served as my primary key for almost all of my tables. I chose to make movies\_db a relational database for this very reason. I wanted to be able to easily query remake and originals data in a variety of different normalizations with a common index.

Potential Database Uses:

The set up of this database allows for easy querying in movie remake analysis. It can help answer questions like, “Are comedy remakes more popular than romance remakes?” “What is the difference in movie remake ratings between critics and the general population?” “Which movie remakes are more popular than their original counterpart?” “Are movie remakes released in the last year more popular than those released in the last 20 years on average”?”

While I do have some redundant data in my movies\_db tables, such as “imdb ID” in originals, remakes, AND imdb tables, I believe that this my database much more likely to be used for any future work. It will make it much easier to join information from other sources of IMDb data other than the OMDb API.