KU Data Analytics Bootcamp

Kill Bill Vol. 3
Our ETL Project

Annie Di Persio, Mark Messick, & Ben Anderson

Introduction

For our project we decided to collect data on the top 50 highest grossing movies of all time. In order to access all the information needed, we had to pull data from several sources.

Data Sources

We first went to Wikipedia and used Pandas to scrape a table containing basic information on the top 50 movies including the title, year of release, and total gross. We then passed a list of all the movie titles to the OMDB API, and pulled the following data: director, MPAA rating, IMDB rating, and total runtime.

As the API data came in it was stored in JSON dictionaries, so we temporarily kept all the useful data in several lists, and then converted those lists into a dataframe.

Transformation

We next needed to perform some transformative steps to get the information we wanted into tables that we could combine. In creating our web scraped data frame, we had already selected the specific data columns we wanted to keep for our purposes so there was no need to add or drop columns. We then set the Rank column to our index to make the index more meaningful.

After we used the title column to iterate through our API to get the second set of data, we renamed the columns so they would fit better with SQL syntax including making all column names lowercase with underscores replacing spaces. Next, once the API data was extracted, we loaded it into a Pandas data frame that matched our other data in format including lowercase column names and underscores.

Database

After collecting and transforming the datasets in Python, we used SQLAlchemy to connect them to a PostgreSQL database in order to store the information. We chose PostgreSQL because it is a relational database, and the datasets we were working with were directly related to each other. Using the pgAdmin platform allowed us to run simple SQL queries that merged the datasets and could be used easily for further analysis.

Tables

Once the data had been transformed in Python, the PostgreSQL database was assembled by creating tables (“wiki” and “imdb”) with an identical structure (table column names) to the corresponding Python DataFrames (“df” and “df_2”). After importing the DataFrames to the tables of the database, we ran a JOIN query that merged those tables to produce a final resulting table titled “movies”:

```
CREATE TABLE movies AS
SELECT DISTINCT wiki.rank, wiki.title, wiki.worldwide_gross, wiki.year,
        omdb.director, omdb.runtime, omdb.rating, omdb.imdb_rating
FROM wiki
FULL OUTER JOIN omdb
ON wiki.title = omdb.title
WHERE wiki.title IS NOT NULL
ORDER BY wiki.rank;
```

This query combined all the columns from each table, merging them on distinct movie *titles*, producing a final table that combined all the data for each movie title into one clean, consolidated result.