Project 2: ETL Challenge

Jacob Avchen, Jacob Hollander, Salvatore Russo

The Jupyter Notebooks in our project are designed to extract, transform, and load data about book reviews, movie reviews, and a link between the two with a table that connects movies with the books they were based on in an effort to see the relation between how well a book was rated and how well the movie based on it was rated.

The program takes 4 tables based on data from Kaggle, IMDB, and Wikipedia and normalizes the data in order to be compared in PostgreSQL.

The following processes were completed using BeautifulSoup and Pandas' built-in read_csv function to perform operations.

EXTRACT:

1. books.csv - source: kaggle.com
2. title.basics.tsv.gz - source: IMDB Data Files
3. title.ratings.tsv.gz - source: IMDB Data Files
4. Wikipedia's list of fiction works with feature film adaptations using BeautifulSoup - source: Wikipedia

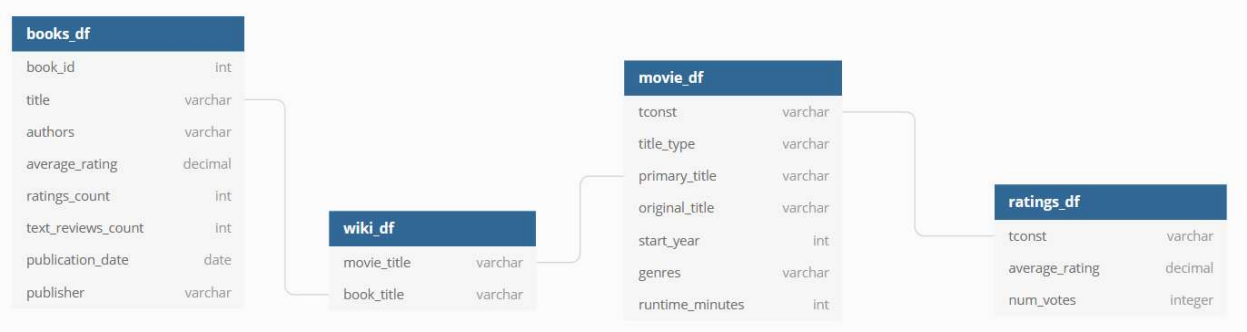
TRANSFORM:

1. Books
    a. Remove unnecessary columns
    b. Clean titles to standard format with no parentheses
    c. Change column names to standard underscore notation
2. Movies
    a. Remove unnecessary columns
    b. Remove entries that are not movies, ie: shorts, tv shows, etc.
    c. Remove entries with no release year, or that have not been released yet
    d. Remove entries with no entry in ratings_df via the tconst column
    e. Change column names to standard underscore notation
3. Ratings
    a. Remove entries with no entry in movie_df via the tconst column
    b. Change column names to standard underscore notation
4. Wiki
    a. Create single A-Z dataframe from 4 separate alphabetized tables
    b. Clean book titles to standard format with no parentheses
    c. Change column names to standard underscore notation

LOAD:

Before we loaded the data into PostgreSQL, we had to create the books_movies_db on our local computers in pgAdmin4 to store it. We loaded the data into PostgreSQL using SQLAlchemy and Pandas' to_sql function to create tables for each dataframe in books_movies_db and load the data in at once. Below are diagrams of the relations between the tables on quickdatabasediagrams.com and a

visualization of the entire ETL process.



EXTRACT:

Source:

https://www.kaggle.com/jealousleopard/goodreadsbooks
books.csv

Raw Data:

bookID
title
authors
average_rating
isbn
isbn13
language_code
num_pages
ratings_count
text_reviews_count
publication_date
publisher
Unnamed:12
# of records: 11,127

TRANSFORM:

From:

Dataframe: books_df

Action:

keep - not null as book_id
keep - not null as title
Remove
keep - not null as average_rating
remove
Remove
Remove
Remove
keep - not null as ratings_count
keep - not null as text_reviews_count
keep - not null as publication_date
keep - not null as publisher
Remove
# of records: 11,127

LOAD:

Into:

Table: books

Table: books

book_id
title
average_rating
ratings_count
text_reviews_count
publication_date
publisher
# of records: 11,127

EXTRACT:

Source:

https://datasets.imdbws.com/
title.basics.tsv.gz

Raw Data:

tconst
titleType
primaryTitle
originalTitle
isAdult
startYear
endYear
runtimeMinutes
genres
# of records: 6,762,842

TRANSFORM:

From:

Dataframe: movie_df

Action:

keep - only values in ratings_df as tconst
keep - "movies" as title_type
keep - not null as primary_title
keep - not null as original_title
Remove
keep - not null as start_year
Remove
keep - as runtime_minutes
keep - not null as genres
# of records: 246,366

LOAD:

Into:

Table: movies

Table: movies

tconst
title_type
primary_title
original_title
start_year
runtime_minutes
genres
# of records: 246,366

EXTRACT:	TRANSFORM:	LOAD:
Source:	**From:**	**Into:**
https://datasets.imdbws.com/ **title.ratings.tsv.gz**	**Dataframe: ratings_df**	**Table: ratings**
Raw Data:	**Action:**	**Table: ratings**
tconst averageRating numVotes **# of records: 1,033,876**	keep - only values in movie_df as tconst keep - not null as average_rating keep - not nul as num_votes **# of records: 246,366**	tconst average_rating num_votes **# of records: 246,366**

EXTRACT:	TRANSFORM:	LOAD:
Source:	**From:**	**Into:**
https://en.wikipedia.org/wiki/Lists_of_fiction_works_made_into_feature_films **page = requests.get(url)** **soup = BeautifulSoup(page.text,"lmxl")**	**Dataframe: wiki_df**	**Table: wiki**
Raw Data:	**Action:**	**Table: wiki**
Book Titles Movie Titles **# of records: 1,637**	keep - not null as book_title keep - not null as movie_title **# of records: 1,637**	book_title movie_title **# of records: 1,637**

Challenges and Limitations:

During this project we ran into a few challenges. Taking data from multiple different sources, each source had their own way of formatting the titles of books and movies. This made it difficult to relate them to each other when one source had a book titled “Harry Potter and the Prisoner of Azkaban (1999)” and the other had it titled “Harry Potter and the Prisoner of Azkaban (Harry Potter #3)”. This was easily solved by removing everything in the parentheses for both columns, but the problem was even worse when relating the movies to each other because, in some cases, multiple movies had been made of the same book. Wikipedia’s table denoted this by having all movies made of a single book in one cell. This led us to relate our own movies.primary_title to wiki.movie_title by looking in wiki.movie_title for the string found in movies.primary_title using the following Join in our SQL Query.

“INNER JOIN wiki AS w ON POSITION (m[ovies].primary_title IN w.movie_title)<>0”

This query allows us to find movie titles in the wiki table even if the title as it’s written in the movies table is only part of the value in the wiki table. In theory this worked and returned the values we needed, but it ran into problems when it looked for simple movie titles like “Brother” and “L” and found every instance of “Brother” and “L” and thought it had found books related to those movies, when in reality it had found “My Blind Brother” for both movies because the title “My Blind Brother” contains both “Brother” and “L”. Given more time, this issue could be solved with more intensive cleaning of the wiki.movie_title column and a stricter query to find only titles that matched exactly.