ETL Project:

Netflix Content Reviews Technical Report

Matthew Werth

Simar Singh

Akash Vaidya

I. INTRODUCTION

For the ETL project, our team chose to use the “Netflix Movies and TV Shows” and “IMDb movies” data sets from the Kaggle.com. We selected these datasets as the information detailed in these files have similar categories and will allow for more seamless integration when merging the files. In this report, we will articulate the processes used to clean and upload these files into a relational database.

II. EXTRACTION

Our team had used three (3) data sets in Jupyter Notebook. The files used were in CSV format. We had used pandas to upload the datasets into data frames into Jupyter Notebook. The CSV format allowed us clarity when reviewing the data in each CSV file. The first two data sets that were uploaded were the IMDb movies and Netflix titles, these files were merged and cleaned, the new CSV file was called Movie Data, to create our first table in PostgreSQL. The third and final data set that was used was IMDb ratings. After cleaning this CSV file, our team merged it with the Movie Data CSV file and the merged file was used to create our second table in PostgreSQL.

III. TRANSFORMATION

In the transformation aspect of our projects, we had cleaned and merged all three CSV files at various stages in this process.

For the IMDb movies and Netflix titles data sets, we reviewed each file on what each file contained. The IMDb file only had information relating to movies, while the Netflix data set had information on television shows and movies. We made the decision to only use information relating to movies in both files. The television show data was missing information on the directors and there were numerous rows with the same name. The Netflix file was split to separate the television shows and movies data.

After splitting the data, we had merged the IMDb data set onto the Netflix data set and were left with 1,704 entries in the newly merged file. The categories used to merge were year, director, and title. A few of the issues that we encountered were the files were difficult to merge due to the huge amount of data in each file and there were eleven (11) entries with the same movie title. To correct the mismatches, we had dropped the duplicated movie titles and conducted a search to ensure there were no duplicated IMDb ids or mismatched titles in the file. Upon further review of the merged file, several columns that were missing a significant amount of information were removed. It was noted that both data sets still were missing information in various columns so we combined the available data from both data sets to fill in the blanks. We removed several columns that contained the same information. The columns cut were cast, year and date published, while actors and release year were kept. There were also two duration columns, we kept the IMDb file duration as it had integer data. Another change that we made was to remove the country data from Netflix as IMDb contained more information in the country column. One final review was conducted of the merged file.

Upon the final review, we made several edits to further clean the file. The first change made was filling the missing rating and language data with a notation of “not available”. Original title data was removed as the information provided was deemed not important. Genre and listed in sections possessed the same data, we cut the listed information data. For the year of release, we had changed the data type to the python functions for data/time format. The columns were rearranged to include the important sections first and renamed for better clarity. The merged file was converted into a CSV file and named Movie Data. After reviewing and cleaning the Movie Data CSV file, we had used SQLAlchemy to push the file from Jupyter Notebook to PostgreSQL/PG Admin 4.

The next CSV file that we reviewed and cleaned was the IMDb ratings data set. Most of the columns in this data set were renamed. The IMDb ratings data set was merged with the merged Movie data set using the IMDb id. After the merge, we had dropped the columns that were not relevant to the ratings data. To avoid putting string data into an integer column in PG Admin 4/PostgreSQL, we opted to not mark the missing section as “not available”. In SQL, the missing data would be shown as null, which would provide more clarity as the ratings data consists of integers/numbers. After cleaning and editing the Ratings file, SQLAlchemy was used to push the rating files onto to PostgreSQL/PG Admin 4.

IV. LOAD

For the relational database, we had decision to use PostgreSQL as the database allowed CSV files to be viewed with clarity and the data we had was reasonably small, highly structured and in a one (1) to many relationship. In PG Admin 4/PostgreSQL, we had created a created a database called Netflix db. We had created two tables in the Netflix database, the first table was called Netflix, while the second table was called ratings. The Movie Data file was added onto the Netflix table and the ratings file was added onto the ratings table. For each table, a primary key was used, and each row had an IMDb id. In addition, the ratings table had a foreign key/constraint that did not allow for any additions to be made without adding corresponding information to the Netflix table.

V. CONCLUSION

The CSV files were compatible with each other that allowed for integration of the files. Each file contained a significant amount of information that we reviewed and then cleaned to ensure relevant information was kept. After uploading the files into SQL and creating the database/tables, the final product of our work allowed for the data provided to be reviewed with simplicity. Our final database would offer openness in regards to general movie information from the Netflix database and also the IMDb ratings for those movies.