ETL Project: Final Report

**Team:** Matt Stieg, Danny Habetyes, Desirae Gagnon

We browsed Kaggle.com data sets and found three internet movie data sources; IMDB-Movie-Data, tmdb\_5000\_movies, and wiki\_movie\_deduped. Each data set had good movie attributes with some overlapping information.

(<https://www.kaggle.com/tmdb/tmdb-movie-metadata#tmdb_5000_movies.csv>)

(<https://www.kaggle.com/jrobischon/wikipedia-movie-plots>)

(<https://www.kaggle.com/PromptCloudHQ/imdb-data>)

**Extract**

We exported each data set into a csv’s and saved locally to review and do some basic analysis of the data in Excel to get column lists, record count, and null counts. The data was then sorted by Title which led to discovery of duplicate records.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TMDB** | 4803 records |  | **IMDB** | 1000 records | |  | **Wiki** | 34887 records |
| Columns |  |  | Columns |  | |  | Columns |  |
| budget | 1037 nulls |  | Rank |  | |  | Release Year |  |
| genres | 28 nulls |  | Title |  | |  | Title |  |
| homepage | 3091 nulls |  | Genre |  | |  | Origin/Ethnicity |  |
| ~~id~~ |  |  | ~~Description~~ |  | |  | Director | 1124 nulls |
| ~~keywords~~ | 4125 nulls |  | Director |  | |  | Cast | 1422 nulls |
| original\_language |  |  | Actors |  | |  | Genre | 6118 nulls |
| ~~original\_title~~ |  |  | Year |  | |  | Wiki Page |  |
| overview |  |  | ~~Runtime (Minutes)~~ |  | |  | ~~Plot~~ |  |
| ~~popularity~~ | 1 null |  | Rating |  | |  |  |  |
| production\_companies | 351 nulls |  | Votes |  | |  |  |  |
| production\_countries | 174 nulls |  | Revenue (Millions) | 76 < Mill | |  |  |  |
| release\_date | 1 null |  | Metascore | 64 nulls | |  |  |  |
| revenue | 1427 nulls |  |  |  | |  |  |  |
| runtime | 35 nulls |  |  |  | |  |  |  |
| spoken\_languages | 86 nulls |  |  |  | |  |  |  |
| ~~status~~ |  |  |  |  | |  |  |  |
| tagline | 844 nulls |  |  |  | |  |  |  |
| title |  |  |  |  | |  |  |  |
| ~~vote\_average~~ | ~~63 nulls~~ |  |  |  | |  |  |  |
| ~~vote\_count~~ | ~~62 nulls~~ |  |  |  | |  |  |  |
|  |  |  |  |  | |  |  |  |
| \*\*\*\*Null count includes unknowns, blanks, zeros, and/or bad data | | | | |  |  |  |  |

**Transform**

To drop duplicates and transform the data we read each csv file into Jupyter Notebook using Pandas and Numpy imports. The first steps were to read the files then create data frames then identify and drop duplicate rows. We were eager to merge the data sets and made several attempts before realizing that the title columns needed indices before the data sets could be merged. The smaller data sets were merged first, then merged with the largest data set. The goal was to eliminate nulls so a combination of left joins and an inner join were used.

Once merged, redundant and unnecessary columns needed to be dropped, the initial analysis documentation was helpful in deciding which columns to keep; again, the goal being to keep the most complete and reliable data.

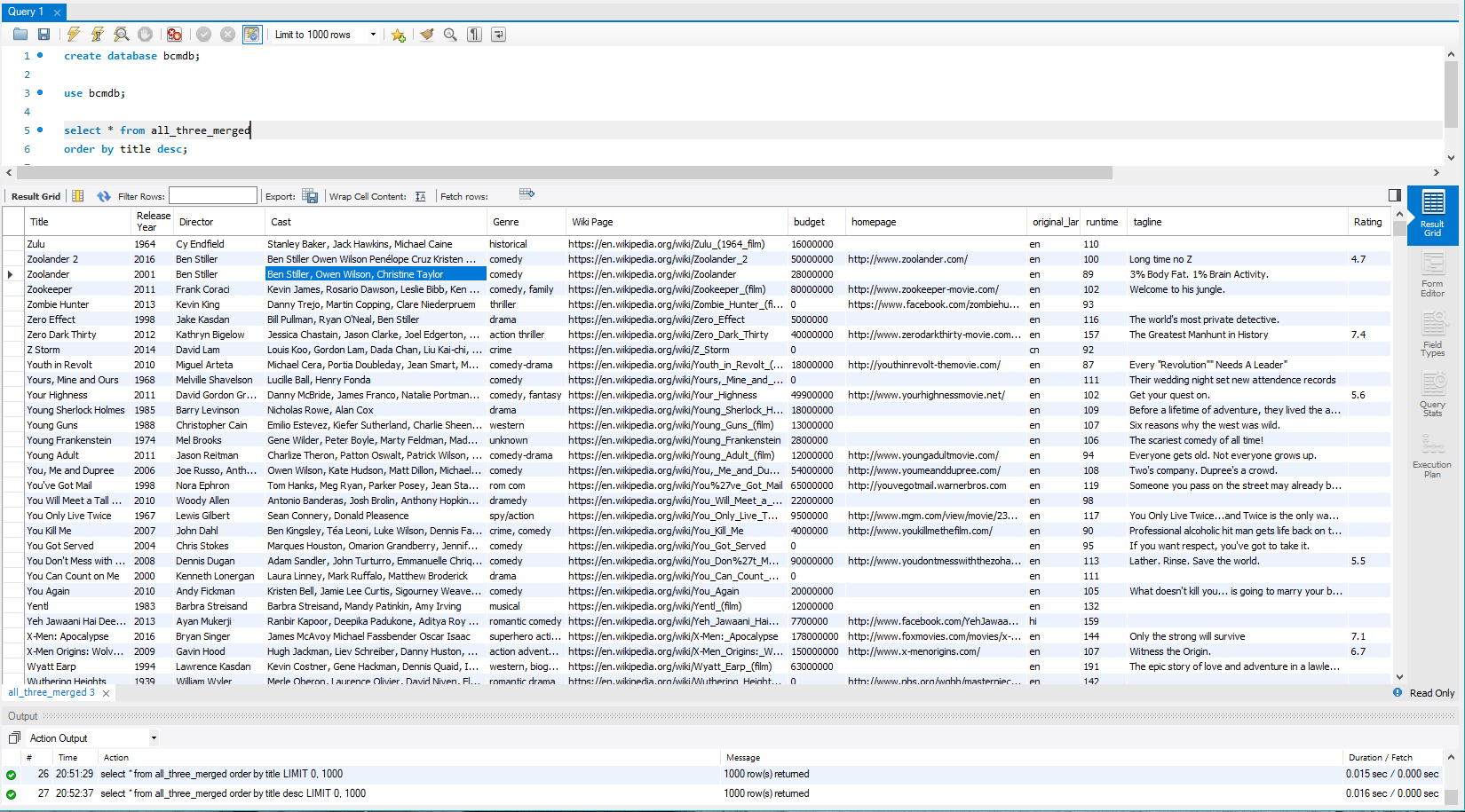
Dropping columns from existing data frames was restricted due to the indices created in order to merge; so the indices had to be dropped, new data frames sans unwanted columns created, indices recreated, then the process of merging data sets repeated.

**Load**

The final data set was easily compatible for loading into either MongoDB or MySQL. Both were tested and successful.

**MySQL**

BCMDB was created and data imported using the import wizard into one table. To test successful loading select statements were written to display table data in ascending and descending order.



**MongoDB**

Due to the structure of the initial data, a noSQL database such as Mongo looked to be one of the better solutions. In the original datasets, several columns (including genre, and production company) collected multiple entries and stored them as dictionaries within a list, each containing an ID as the key. To allow for easier data retrieval for end users, we were faced with two options: 1) Create reference tables and normalize the data; 2) use a noSQL database to allow end users to call specific keys when attempting to search for the contained values. The second option was a lot more efficient and does not require multiple tables to be updated when either a new movie is entered, or a new genre declared. 