**Technical Report Summary**

Data

The datasets used for this project came from two separate Kaggle databases, both relating to movies. The first was “[The Movies Dataset](https://www.kaggle.com/rounakbanik/the-movies-dataset)”, which contained csv files with information about 45,000 movies, including the metadata (i.e. title, budget, revenue, release dates, etc.), ratings, and credits. The second was “[The Oscar Award, 1927-2020](https://www.kaggle.com/unanimad/the-oscar-award)”, which contained a csv file with Oscar winners, awards, and film titles for the years 1927-2020.

Transformation

A similar transformation procedure was completed for both datasets. Namely, after importing the datasets to a Pandas DataFrame, the columns were first filtered to the desired values for our analysis. They were then renamed to match those found in our postgresql “create table” statements. To join the Oscars dataset with the Movies dataset, we first needed to group the Oscars dataset by film title and use the “COUNT” aggregate function. The ratings dataset was also grouped and aggregated by the “MEAN” function to find average ratings for each movie. The Movies dataset was also cleaned to remove movies with allegedly $0 budgets and/or revenues.

Load

Each cleaned DataFrame (i.e “oscarsindex,” “oscars,” “movie\_meta,” “credits,” and “ratings”) was then loaded into postgresql using the Pandas “to\_sql” method. Because the tables were relational, we could then conduct all of our join queries in pgAdmin (see project.sql file).

**Project Report**

Oscar Nominations

Using the Kaggle dataset we downloaded a csv containing all of the Oscar winners since 1927.

To eliminate duplicate movie ids from one movie winning multiple awards in a given year we created a grouped dataframe to show only unique film names; their corresponding row indices were then used as the primary key for SQL. We eliminated several columns showing data that was either confusing or irrelevant and changed the column names to correspond to the table column names in SQL. Following this, using the Pandas LOC function we filtered down to only show Oscar award categories of Best Picture, Directing, Best Actor and Best Actress. We then loaded two tables into SQL; one table showing the films with their matching IDs and one table showing all Oscar winners for the selected categories regardless of one movie winning multiple awards. From this point, we loaded the data into SQL where further joining and querying was to be done.

Movies

For this project, Kaggle’s “The Movies Dataset” was used to obtain information about each movie in the Oscar Nominations dataset.

After importing the necessary dependencies, the metadata, credits, and ratings csv files were then imported into a Pandas DataFrame. The metadata was then filtered from 24 to 11 desired attributes, including the Movie ID, title, release date, runtime, genres, etc. Duplicates and illegitimate rows were then dropped (including those with “$0” budgets and/or revenues). A similar procedure was conducted for the credits DataFrame. For the ratings dataset, the attributes “movieId” and “rating” were selected and then grouped by “movieId” with the aggregate function “MEAN”. This was done to produce a single-valued rating for each movieId in the dataset. Finally, all data was loaded to postgresql with identical column names and data types for each attribute in the datasets.

Limitations and Challenges

* Several columns in the movie datasets save data in dictionary type. It would be better if we could unfold the information. But for this project, we could only save these data in text into SQL without transforming them into a separate table. .
* So far this database is short of some information we were interested in, like the MPAA film rating. For future studies, we may have to find extra sources.
* Time!!