**ETL PROJECT REPORT**

GROUP - (Corona Be Gone!) - Team Members: Rahul Sanghera Arun Singh, Effiba Tumi, Jannat Randhawa.

**Project Overview**

The objective of this project was to create a database to display critic ratings from IMDB rating and rotten tomatoes, in conjunction with academy awards for the movies released in the year 2000 to 2015. Using our data, we created a relational database in postgressql.

**Data Extraction**

We relied on Kaggle to extract our data set for Academy Awards, IMDB movies and Rotten Tomatoes. The **Academy Awards data set** consists of the official record of movie winners and nominees from movies released from 1927 to 2015. The **IMDB data set** is an extensive data set, that contains around 81,273 movies from around the world, ranging from the first movie ever released in 1874 to present. The data set records various movie characteristics such as actors, plots, directors, writers, budget, year of release and ratings among other criterias. These movies in IMDB dataset are characterized by their own unique IMDB ID. The **rotten tomatoes data se**t contains all the movie data such as cast, directors, rating etc, however it lacked the rotten tomato critic ratings. To overcome this issue, we utilised an api key from the OMDB website to extract the rotten tomato critic ratings for the movies in the IMDB dataset, using the unique IMDB ID provided in the data set. We converted each one of the following files to a pandas data frame;

* + Academy Awards Datas as AA\_df.
  + IMDB data as imdb\_movie\_data.
  + Rotten tomatoes as rotten\_tomatoes.
  + OMDB data as omdb\_db.

**Transformations**

* **Academy Awards Data** - The data provided in the academy awards dataset ranged from 1927 to present. The data set had two columns for the Years as Year 1 and Year 2. Secondly, it displayed NAN for the movies that did not win an award/s and 1.0 for the movies that won an award. We created another dataframe called AA\_filtered\_df in which we fixed the following:
* Year 1 and 2 were dropped from the dataset using the “.drop” function and were stored as one Year. The years were originally saved in a string format which were then converted to a numeric value using the “pd.to\_numeric” function.
* To shrink the dataset, it was filtered from the year 2010 to 2015 using the “.loc” function.
* Next using the “.fillna” function we replaced the NAN values with “0” and later used “.replace” to change the 1.0 value to “Won” .
* The column name was changed from “Winners” to “Outcome” and “Name” to “Title”.
* Lastly in order to match with other datasets we changed the column names to lowercase using “.str\_lower” function and made a special column called query\_titles in order to merge with other data tables.
* **IMDB DATA** - The IMDB database queried extensive data documented in 4 different CSV’s, a lot of which was redundant to our project. We chose the csv that contained basic information such as 'imdb id’ ','title','year','genre', 'votes', 'duration', 'country', 'language' and description'.
* Firstly, using the iloc function we extracted data for movies released in the USA during the time period of 2010 to 2015.
* We then created a second data frame “imdb\_movie\_data2” with just the relevant attributes such as 'imdb id’ ','title','year','genre', 'votes', 'duration', 'country', 'language' and description'.
* The imdb\_title\_id column was then renamed to imdb\_id using the “.rename” function.
* We changed the “title” from to “query\_title” to match the query\_title in the “AA\_filtered\_df” data set to ensure smooth merging of the table.
* **OMDB DATA** - We extracted information from the OMDB website using API key, and jsonified the response. We set Years, Country and IMDB\_title\_id as conditions to filter results. Further, we made a loop and queried in data using the “if” and “else” function. From the imdb\_movie\_data2 we set imdb\_title\_id and OMDB api\_key as parameters to query results from the response and append it to the results list.
* OMDB data stored all the critic ratings from IMDB, Rotten Tomatoes and Metacritic in one column as three different dictionaries. We extracted the rotten tomatoes rating by using the “.Ratings.apply(pd.Series)[1].apply(pd.Series)["Value"]” command to the rating section in the omdb\_db dataframe.
* From the omdb\_db we created a new data frame new\_omdb\_db and renamed the columns using the “.rename” function to change 'imdbID’ to 'imdb\_ID' and 'imdbRating' to 'imdb\_rating'.
* Lastly we set imdb\_ID as index using the “.set\_index” function.
* **Rotten Tomatoes -** We used the iloc function to extract data for movies released in the USA during the time period of 2010 to 2015.
* From the old data set 'Title', 'Cast 1', 'Cast 2', 'Rating', 'Director 1', 'Release Date', 'Studio' were selected to make a new data frame called “rotten\_tomatoes\_1”.
* The column names were then changed by converting them lower case using “columns.str.lower()”, space between titles was changed with underscore using “name.replace(' ', '\_') for name in columns”. Lastly the columns were renamed from 'cast\_1' to 'actor\_1', 'cast\_2' to 'actor\_2', 'rating' to 'pg\_rating', 'director\_1'to 'director' using the “.rename” function.
* Lastly to match the other 2 datasets we added another column called “query\_title” and converted the titles from title section to lowercase so it can be easily merge with other datasets.

**LOADING**

* We created a cloud database using AWS and then an engine connection was created using python to connect the jupyter lab datasets to SQL postgres server.
* Lastly, we pushed our tables to this database. We connected to the cloud database using PG Admin and also ran different queries.