**ETL Project Final Report**

Team Members – Erik Lockwood, Maria Lara, Deepika Awasthi

**Data source Description**

Following data sources were used for this ETL project:

1. **CSV files from kaggle link**

<https://www.kaggle.com/stefanoleone992/imdb-extensive-dataset?select=IMDb+title_principals.csv>

* + IMDb Movies.csv
  + IMDb Names.csv
  + IMDb Ratings.csv

1. **CSV files from data world link**

<https://data.world/data-society/imdb-5000-movie-dataset/workspace/file?filename=movie_metadata.csv>

* + movie\_metadata.csv

1. **CSV files from following link**

[https://www.kaggle.com/ruchi798/movies-on-netflix-prime-video-hulu-and-disney](https://slack-redir.net/link?url=https%3A%2F%2Fwww.kaggle.com%2Fruchi798%2Fmovies-on-netflix-prime-video-hulu-and-disney)

* + MoviesOnStreamingPlatforms\_updated.csv

**Extract**

1. **Extracted all the below character details from IMDb Names.csv file**
   1. Name
   2. Birth Name
   3. Height
   4. Birth Date
   5. Death Date
   6. Primary Profession
2. **Extracted all the below Movie details from IMDb Movies.csv file**
   1. Movie Title
   2. Release Year
   3. Duration
   4. Budget
   5. Gross Income
3. **Extracted Movie Genre from IMDb Movies.csv file**
4. **Extracted following details from movie\_metadata.csv file**
   1. Director Facebook Likes
   2. Actor Facebook Likes
5. **Extracted movie’s country from IMDb Movies.csv**
6. **Extracted movie’s language from IMDb Movies.csv**
7. **Extracted movie’s rating from MoviesOnStreamingPlatforms\_updated.csv**

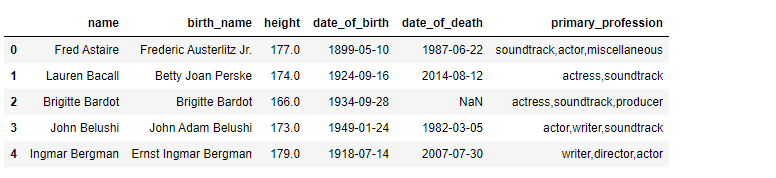
**Transform**

**Movie Table**

1. Read in “IMDb movies.csv”, sorted by year ascending, and removed all entries prior to the year 2000.
2. Created a new DataFrame with only the columns of interest.
   1. ['title', 'year', 'duration', 'budget', 'worlwide\_gross\_income'].
3. Read in “MoviesOnStreamingPlatforms\_updated.csv" for streaming platform availability data, sorted, and dropped all rows before year 2000.
4. Created a new DataFrame with only the columns of interest keeping title & year to join on
   1. ['Title', 'Year', 'Netflix', 'Hulu', 'Prime Video', 'Disney+'].
5. Renamed all columns to align with database schema.
   1. 'movie','movie\_actor','actor','lang\_movie','lang','movie\_genre','genre','movie\_country','country','movie\_drt','director','rating'].
6. Added “joinstring” column to both DF’s that as a concatenation of movie title and year with white space stripped out and characters forced to lower case to create a unique join id that would preempt remakes of a movie with the same title or inconsistent capitalization joining incorrectly or failing to join.
7. Merged on joinstring then dropped joinstring from final merge\_df.
8. Filled all NaN’s in streaming availability columns with “0” indicating not available
9. Updated “0” & “1” values for streaming availability to Boolean values to align with DB schema
10. Stripped out non-numeric characters from budget gross income and changed columns to float to align with DB.
11. Created movie\_id column on finalized merge\_df.
12. Loaded data to postgres DB.

**Actor Table**

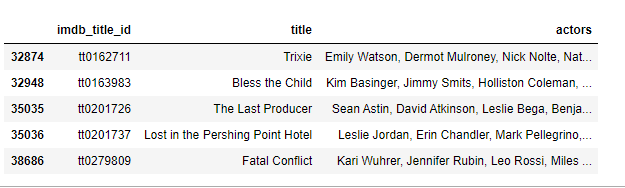
1. Created a data frame “actors\_transformed” to get character details from **IMBD Names.csv** file
2. The primary profession has comma separated profession values so created “actorscleancsv” dataframe by applying a split function.



1. Renamed the column after the split back to Profession heading.
2. Joined the “actors\_transformed” dataframe with “actorscleancsv”.
3. Filtered the data to get rows where profession is “actor” or “actress” and created a dataframe “actors\_filtered”.
4. Renamed the columns “date\_of\_birth” and “date\_of\_death” to “birth\_date” and “death\_date” to match the column name in database.
5. Applied datetime formatting on “birth\_date” and “death\_date” columns.
6. Extract Facebook Likes for the actor
   1. Created dataframe “metadata\_df” from **“movie\_metadata.csv”**
   2. Grab “actor\_name” and “actor facebook likes” column from the dataframe
   3. Rename the columns “actor\_name” and “actor facebook likes”.
   4. Perform a left join with “actors\_clean” dataframe to merge the actor facebook likes and create a merged dataframe “actor\_FB\_merge”.
7. Create a unique “actor\_id” column in “actor\_FB\_merge” dataframe.
8. Uploaded the final “actors\_clean” dataframe to actor database table.

**movie\_actor table**

1. Created a data frame “movies\_file” from **IMBD Movies.csv** file.
2. Filter the dataframe “movies\_file” for year > 2000.
3. Retrieve columns “imdb\_title\_id “ , “title” and “ actors”.
4. Actors value is comma separated so created “actormoviecleancsv” dataframe by applying a split function:

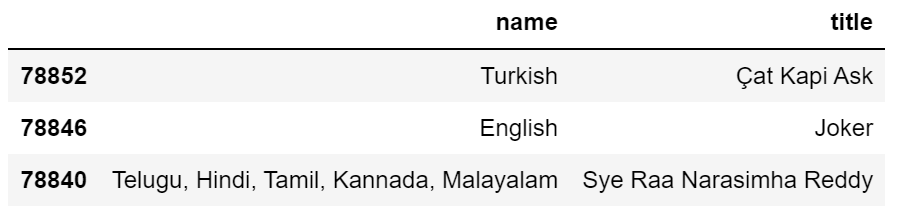


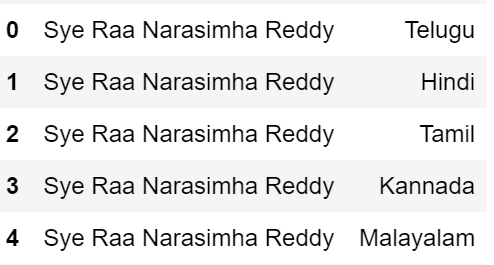
1. Renamed the column to “name”.
2. Merged the actor details with “actormovie\_renamed” to get actorid and actor facebook likes in 1 dataframe.
3. Dropped the unnecessary columns not needed.
4. Merged with movie dataframe to get movie ID for the actor ID combination.
5. Loaded the “actormovie\_final”dataframe to “movie\_actor” table.

**country-Language Tables:**

Source: IMDb Movie.csv

1. Original excel was loaded as a Pandas Dataframe.
2. Data was filtered to get only movies and the rest of columns from 2000 to 2019.
3. Filtered DataFrame was cleaned to show only desirable columns.
4. Languages and Countries were columns with comma separated data, so “split” and “stack” pandas’ functions were needed to define a new row every time a cell contains more than one country or language separated by commas, as follow:





1. Columns renaming process to match same database columns names.
2. lang\_id and country\_id serial id’s generation.
3. Reorganizing columns order.

**“lang\_movie” and “movie\_country” Tables:**

Source: IMDb Movie.csv

1. Join each normalized languages and country tables (without comma separation) with the “Movie Table” that contains “movie\_id”. Then columns were renamed.
2. A second Join between the above merged tables (countr\_movie\_merge, lang\_movies\_id\_merge) with their corresponding country\_table and language\_table.

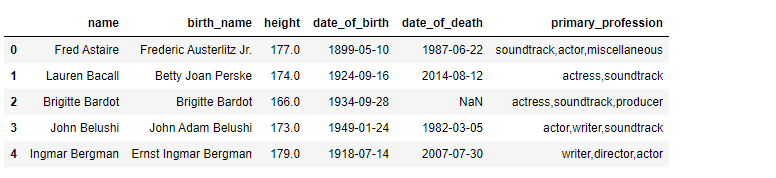
**Rating Table**

Source: IMDb Movie.csv, MoviesOnStreamingPlatforms\_updated.csv

1. Original Movie as Dataframe (movie\_table).
2. Filtered desirable columns.
3. MoviesOnStreamingPlatforms\_updated loaded as DataFrame (rating\_source\_df)
4. Columns renaming process.
5. Column “Rotten Tomatoes” was converted as float, “%” was removed.
6. Column “Rotten Tomatoes” was divided into 100, to ensure a fraction for future calculations.
7. Merge between rating\_source\_df and movie\_table.
8. Data cleaning of the merged dataframe (filtered and renamed columns).

**Director Table**

1. Created a data frame “actors\_ directors\_df” to get character details from **IMDb Names.csv** file
2. The primary profession has comma separated profession values so created “directors\_merge” dataframe by applying a split function:



1. Renamed the column after the split back to Profession heading.
2. Filtered the data to get rows where profession is “director” created a dataframe “directors\_filtered”.
3. Created final “directors\_clean” dataframe with only director names in it
4. Extract Facebook Likes for the director –
   1. Created dataframe “metadata\_df” from **“movie\_metadata.csv”**
   2. Grab “director\_name” and “director\_facebook\_likes” column from the dataframe
   3. Rename the column “director\_name”.
   4. Perform a left join with “directors\_clean” dataframe to merge the director facebook likes and create a merged dataframe “director\_FB\_merge”.
5. Create a unique “director\_id” column in “director\_FB\_merge” dataframe.
6. Uploaded the final “director\_FB\_merge” dataframe to director database table.

**movie\_drt table**

1. Created a data frame “movies\_file” from **IMDb Movies.csv** file.
2. Filter the dataframe “movies\_file” for year > 2000.
3. Retrieve columns “imdb\_title\_id “, “title” and “director”.
4. Director value is comma separated so created “directormoviecleancsv” dataframe by applying a split function.
5. Renamed the column to “name”.
6. Merged the director details with “directormovie\_renamed” to get directorid and director facebook likes in 1 dataframe.
7. Dropped the unnecessary columns not needed.
8. Merged with movie dataframe to get movie ID for the director ID combination.
9. Loaded the “directormovie\_final” dataframe to "movie\_drt" table.

**Genre Table**

1. Created a data frame “movies\_data\_df” to get movie details from **IMDb Movies.csv** file.
2. Filter the dataframe “movies\_file” for year > 2000.
3. Retrieve columns “imdb\_title\_id “and “genre”.
4. The genre has comma separated profession values so created “genrecleancsv” dataframe by applying a split function



1. Renamed the column after the split back to name heading in “genre\_df” dataframe.
2. Create a unique “genre\_id” column in “genre\_df” dataframe.
3. Uploaded the final “genre\_df” dataframe to “Genre” database table.

**movie\_genre table**

1. Created a data frame “select\_imdb\_movies\_df” from **movie\_output** file.
2. Filter the dataframe “movies\_file” for year > 2000.
3. Use the “genrecleancsv” which already have “imdb\_title\_id” and “Genre” values splitted.
4. Merge the dataframe “select\_imdb\_movies\_df” and “genrecleancsv” to get Movie id and “Genre” name together.
5. Merge the result with “genre\_df” dataframe to get “genre\_id”.
6. Drop the unnecessary columns.
7. Loaded the “Movie\_genre\_final” dataframe to " movie\_genre" table.

**Load**

A postgres relation database is used to load the tables. After visualizing and inspecting the data we found an entity relationship between various tables for the data we extracted. Hence decided to use postgres DB to load the data.

Please refer to ERD diagram for all table and relationship details

1. Create a Postgress DB connection.
2. Used “to\_sql” pandas method to load the data to tables.