# ETL Project Group A

## Team Members

Hema Vyas, Matthew Canady, Lindsay Reynolds

## Project Overview/Objective

The project will extract data from CSV files to Pandas. Data will be transformed and loaded to a relational database.

- Movies are filtered for years > 2000
- Ratings are filtered for weighted_average_vote >= 8
- The project is summarized in the Project Report

## Data Destination

A relational database will be created in postgres with a movies table and a ratings table

Structure of movies table (text unless otherwise identified):

- imdb_title_id
- title
- year (INT)
- language
- country

Structure of ratings table (text unless otherwise identified):

- imdb_title_id
- total_votes
- weighted_average_vote (FLOAT)
- mean_vote (FLOAT)

Note: IMDb_title_id is going to be the primary key

## Requirements

- Remove null values and remove duplicated rows
- Filter for the columns needed for each table
- Filter for movies after the year 2000
- Filter for the weighted_average_vote >=8

### Data Sources

Kaggle: IMDB movies extensive dataset
Movies
Ratings

# ETL Project SOP

Step 1: Load dependencies that you will need (pandas). Create config file in your folder to hold the following information (username, password, port, host, and name)

**Extract**
Step 2: Bring in the movies csv file and create movies_path and create a dataframe using read_csv
Step 3: Runs a list to see what columns exist in our dataframe

**Transform**
Step 4: Reduce dataframe to only the columns that we want to import into SQL later
Step 5: Confirms file type
Step 6: Confirms the data types
Step 7: Converts the year data to a string for future conversion
Step 8: Splits the string apart so that we can extract only the year and helps identify outlier
Step 9: Replaces year outlier with correct format
Step 10: Confirm that outliers were removed and file updated
Step 11: Changes string to integers so that we can later filter on date
Step 12: Filters data for years after 2000 and puts in a new dataframe (reduced the number of rows to only desired data)

**Extract**
Step 13: Bring in the ratings csv file and create ratings_path and create a dataframe using read_csv. Then runs a list to see what columns exist in our dataframe.
Step 14: Runs a list to see what columns exist in our dataframe

**Transform**
Step 15: Reduce dataframe to only the columns that we want to import into SQL later
Step 16: Dropped all rows that have an N/A value
Step 17: Filter the dataframe for the desired data containing only movies that were received a weighted average greater than or equal to 8

**Load**
Step 18: Open PGAdmin and create the database
Step 19: Create the connection string and establish connection
Step 20: Loads the movies dataframe into PostgreSQL (if the table already exists this will overwrite current table in the database)
Step 21: Loads the ratings dataframe into PostgreSQL (if the table already exists this will overwrite current table in the database)

**Other**
Step 22: Brings the movies SQL table back into Jupyter Notebook to confirm in PostgreSQL database
Step 23: Brings the ratings SQL table back into Jupyter Notebook to confirm in PostgreSQL database