MIGRATE NETFLIX AND OMBD DATA INTO PRODUCTION DATA BASE

Project Two

March 7ʰ, 2020

TEAM MEMBERS

- Jonpaul Mastropaolo (Data Architect)

- Tiger Cosmos (Data Starlord)

- Tiania Chan (Data Engineer)

- Clayton Skinner (Data Trash Panda)

TABLE OF CONTENTS

- Project Report
  - Data Sources
  - **E**xtract
  - **T**ransform
  - **L**oad
- Questions

DATA SOURCES

<u>OMDB</u>

Source: OMDB

Website: http://www.omdbapi.com

<u>Netflix</u>

CSV: [Kaggle](https://www.kaggle.com/)

EXTRACT

```python
In [3]:  # This cell contains the for loop to get all the data for movies in netflix in the omdb api

         # Creating the list of movie titles to make the API calls on
         filtered_movies = filtered_flix['title']

         # initializing empty lists to append to in the loop
         metascore = []
         imdbrating = []
         imdbvotes = []
         boxoffice = []
         title = []

         # This is the api call for our OMDB dataframe
         for item in filtered_movies:

             # If the loop doesn't work, we will skip the API call and list append function.
             try:
                 # Creating the string for the URL to call the API from.
                 movie = "t=" + item
                 key = "&apikey="+jomdb
                 url = 'http://www.omdbapi.com/?' + movie +  key

                 # Storing the response as a json object
                 response = requests.get(url).json()

                 # Append the items we care about from the json
                 boxoffice.append(response['BoxOffice'])
                 title.append(response['Title'])
                 metascore.append(response["Metascore"])
                 imdbrating.append(response['imdbRating'])
                 imdbvotes.append(response['imdbVotes'])

             except:
                 next
```
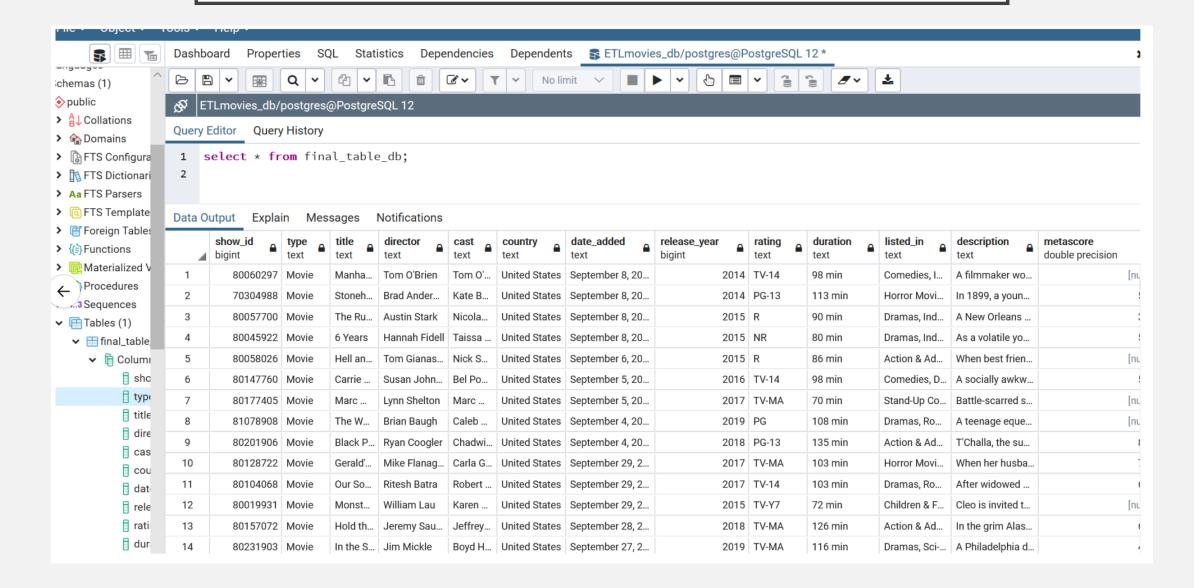
TRANSFORM

```python
# Creating the DataFrame for the OMDB data
omdb_df = pd.DataFrame({'title':title,
                        'metascore':metascore,
                        'imdb_rating':imdbrating,
                        'imdb_votes':imdbvotes,
                        'box_office':boxoffice})
```

```python
# Joining the two tables: the netflix table and the OMDB table
final_table = filtered_flix.join(omdb_df.set_index('title'),on = 'title', how = 'left')
```

```python
# Doing away with the rows that contain "NaN" in any column.
final_table = final_table.dropna()

# Write final table to csv for use later
final_table.to_csv('final_table.csv',index = False, header = True, sep = ',')
```

```python
final_table=pd.read_csv('final_table.csv')

final_table.head()
```

LOAD

```
In [3]:  #connect to local data base
         rds_connection_string = "etl:etl@localhost:5432/ETLmovies_db"
         engine = create_engine(f'postgresql://{rds_connection_string}')
```

```
In [7]:  #check for tables
         engine.table_names()
```

```
Out[7]:  ['final_table_db']
```

```
In [6]:  #upload data frame to database
         final_table.to_sql(name='final_table_db', con=engine, if_exists='append', index=False)
```

```
In [8]:  #check the dataframe made it to the table
         pd.read_sql_query('select * from final_table_db', con=engine).head()
```

LOAD

QUESTIONS

- Open discussion to Q&A from the class.

END