In [1]:
```
import json
import pandas as pd
import numpy as np

import re

from sqlalchemy import create_engine
import psycopg2

from config import db_password

import time
```

In [2]:
```
#  Add the clean movie function that takes in the argument, "movie".
def clean_movie(movie):
    movie = dict(movie) #create a non-destructive copy
    alt_titles = {}
    # combine alternate titles into one list
    for key in ['Also known as','Arabic','Cantonese','Chinese','French'
                'Hangul','Hebrew','Hepburn','Japanese','Literally',
                'Mandarin','McCune-Reischauer','Original title','Polish
                'Revised Romanization','Romanized','Russian',
                'Simplified','Traditional','Yiddish']:
        if key in movie:
            alt_titles[key] = movie[key]
            movie.pop(key)
    if len(alt_titles) > 0:
        movie['alt_titles'] = alt_titles

    # merge column names
    def change_column_name(old_name, new_name):
        if old_name in movie:
            movie[new_name] = movie.pop(old_name)
    change_column_name('Adaptation by', 'Writer(s)')
    change_column_name('Country of origin', 'Country')
    change_column_name('Directed by', 'Director')
    change_column_name('Distributed by', 'Distributor')
    change_column_name('Edited by', 'Editor(s)')
    change_column_name('Length', 'Running time')
    change_column_name('Original release', 'Release date')
    change_column_name('Music by', 'Composer(s)')
    change_column_name('Produced by', 'Producer(s)')
    change_column_name('Producer', 'Producer(s)')
    change_column_name('Productioncompanies ', 'Production company(s)')
    change_column_name('Productioncompany ', 'Production company(s)')
    change_column_name('Released', 'Release Date')
    change_column_name('Release Date', 'Release date')
    change_column_name('Screen story by', 'Writer(s)')
    change_column_name('Screenplay by', 'Writer(s)')
    change_column_name('Story by', 'Writer(s)')
    change_column_name('Theme music composer', 'Composer(s)')
    change_column_name('Written by', 'Writer(s)')

    return movie
```

In [10]:

```
# 1 Add the function that takes in three arguments;
# Wikipedia data, Kaggle metadata, and MovieLens rating data (from Kag

def movies_function():
    # Read in the kaggle metadata and MovieLens ratings CSV files as P
    kaggle_metadata = pd.read_csv ('movies_metadata.csv', low_memory =
    ratings = pd.read_csv('ratings.csv')

    kaggle_metadata_df = pd.DataFrame(kaggle_metadata)
    ratings_df = pd.DataFrame(ratings)

    # Open and read the Wikipedia data JSON file.
    file_dir = "/Users/caroline/Documents/Data Boot Camp/Module 8/Movi
    with open(f'{file_dir}/wikipedia-movies.json', mode='r') as file:
        wiki_movies_raw = json.load(file)

    # 3. Write a list comprehension to filter out TV shows.
    wiki_movies = [movie for movie in wiki_movies_raw if 'No. of episo

    # 4. Write a list comprehension to iterate through the cleaned wik
    # and call the clean_movie function on each movie.
    clean_wiki_movies = [clean_movie(movie) for movie in wiki_movies]

    # 5. Read in the cleaned movies list from Step 4 as a DataFrame.
    wiki_movies_df = pd.DataFrame(clean_wiki_movies)

    # 6. Write a try-except block to catch errors while extracting the
    #  dropping any imdb_id duplicates. If there is an error, capture
    try:
        wiki_movies_df['imdb_id'] = wiki_movies_df['imdb_link'].str.ex

        wiki_movies_df.drop_duplicates(subset = 'imdb_id', inplace = T

    except:
        print("This is an error from step 6")

    #  7. Write a list comprehension to keep the columns that don't ha
    wiki_columns_to_keep = [column for column in wiki_movies_df.column
                            < len(wiki_movies_df) * 0.9]
    wiki_movies_df = wiki_movies_df[wiki_columns_to_keep]

    # 8. Create a variable that will hold the non-null values from the
    box_office = wiki_movies_df['Box office'].dropna()

    # 9. Convert the box office data created in Step 8 to string value
    box_office[box_office.map(lambda x: type(x) != str)]

    # 10. Write a regular expression to match the six elements of "for
    form_one = r'\$\d+\.?\d*\s*[mb]illion'
    matches_form_one = box_office.str.contains(form_one, flags=re.IGNO

    # 11. Write a regular expression to match the three elements of "f
    form_two = r'\$\d{1,3}(?:,\d{3})+'
    matches_form_two = box_office.str.contains(form_two, flags=re.IGNO

    # 12. Add the parse_dollars function.
```

```
    def parse_dollars(s):
        if type(s) != str:
            return np.nan

        if re.match(r'\$\s*\d+\.?\d*\s*milli?on', s, flags=re.IGNORECA
            s = re.sub('\$|\s|[a-zA-Z]','', s)
            value = float(s) * 10**6
            return value

        elif re.match(r'\$\s*\d+\.?\d*\s*billi?on', s, flags=re.IGNORE
            s = re.sub('\$|\s|[a-zA-Z]','', s)
            value = float(s) * 10**9
            return value

        elif re.match(r'\$\s*\d{1,3}(?:[,\.]\d{3})+(?!\s[mb]illion)',
            s = re.sub('\$|,','', s)
            value = float(s)
            return value

        else:
            return np.nan

    # 13. Clean the box office column in the wiki_movies_df DataFrame.
    wiki_movies_df['box_office'] = box_office.str.extract(f'({form_one
    wiki_movies_df.drop('Box office', axis=1, inplace=True)

    # 14. Clean the budget column in the wiki_movies_df DataFrame.
    budget = wiki_movies_df['Budget'].dropna().apply(lambda x: ' '.joi
    budget = budget.str.replace(r'\$.*[-——](?![a-z])', '$', regex=True
    budget = budget.str.replace(r'\[\d+\]\s*', '')
    wiki_movies_df['budget'] = budget.str.extract(f'({form_one}|{form_

    # 15. Clean the release date column in the wiki_movies_df DataFram
    release_date = wiki_movies_df['Release date'].dropna().apply(lambd
    date_form_one = r'(?:January|February|March|April|May|June|July|Au
    date_form_two = r'\d{4}.[01]\d.[123]\d'
    date_form_three = r'(?:January|February|March|April|May|June|July|
    date_form_four = r'\d{4}'
    wiki_movies_df['release_date'] = pd.to_datetime(release_date.str.e

    # 16. Clean the running time column in the wiki_movies_df DataFram
    running_time = wiki_movies_df['Running time'].dropna().apply(lambd
    running_time_extract = running_time.str.extract(r'(\d+)\s*ho?u?r?s
    running_time_extract = running_time_extract.apply(lambda col: pd.t
    wiki_movies_df['running_time'] = running_time_extract.apply(lambda
    wiki_movies_df.drop('Running time', axis=1, inplace=True)

    # 2. Clean the Kaggle metadata.
    kaggle_metadata = kaggle_metadata[kaggle_metadata['adult'] == 'Fal
    kaggle_metadata['video'] = kaggle_metadata['video'] == 'True'
    kaggle_metadata['budget'] = kaggle_metadata['budget'].astype(int)
    kaggle_metadata['id'] = pd.to_numeric(kaggle_metadata['id'], error
    kaggle_metadata['popularity'] = pd.to_numeric(kaggle_metadata['pop
    kaggle_metadata['release_date'] = pd.to_datetime(kaggle_metadata['

    # 3. Merged the two DataFrames into the movies DataFrame.
    movies_df = pd.merge(wiki_movies_df, kaggle_metadata, on='imdb_id'
```

```
114
115     # 4. Drop unnecessary columns from the merged DataFrame.
116     movies_df.drop(columns=['title_wiki','release_date_wiki','Language
117
118     # 5. Add in the function to fill in the missing Kaggle data.
119     def fill_missing_kaggle_data(df, kaggle_column, wiki_column):
120         df[kaggle_column] = df.apply(lambda row: row[wiki_column] if r
121         df.drop(columns=wiki_column, inplace=True)
122
123     # 6. Call the function in Step 5 with the DataFrame and columns as
124     fill_missing_kaggle_data(movies_df, 'runtime', 'running_time')
125     fill_missing_kaggle_data(movies_df, 'budget_kaggle', 'budget_wiki'
126     fill_missing_kaggle_data(movies_df, 'revenue', 'box_office')
127
128     # 7. Filter the movies DataFrame for specific columns.
129     movies_df = movies_df.loc[:, ['imdb_id','id','title_kaggle','origi
130                     'runtime','budget_kaggle','revenue','release_da
131                     'genres','original_language','overview','spoken
132                     'production_companies','production_countries','
133                     'Producer(s)','Director','Starring','Cinematogr
134                    ]]
135
136     # 8. Rename the columns in the movies DataFrame.
137     movies_df.rename({'id':'kaggle_id',
138                   'title_kaggle':'title',
139                   'url':'wikipedia_url',
140                   'budget_kaggle':'budget',
141                   'release_date_kaggle':'release_date',
142                   'Country':'country',
143                   'Distributor':'distributor',
144                   'Producer(s)':'producers',
145                   'Director':'director',
146                   'Starring':'starring',
147                   'Cinematography':'cinematography',
148                   'Editor(s)':'editors',
149                   'Writer(s)':'writers',
150                   'Composer(s)':'composers',
151                   'Based on':'based_on'
152                  }, axis='columns', inplace=True)
153
154     # 9. Transform and merge the ratings DataFrame.
155     rating_counts = ratings.groupby(['movieId','rating'], as_index=Fal
156                 .rename({'userId':'count'}, axis=1) \
157                 .pivot(index='movieId',columns='rating', values='count
158     rating_counts.columns = ['rating_' + str(col) for col in rating_co
159     movies_with_ratings_df = pd.merge(movies_df, rating_counts, left_o
160     movies_with_ratings_df[rating_counts.columns] = movies_with_rating
```

In [13]:
```
#creating a SQL database
db_string = f"postgres://postgres:{db_password}@127.0.0.1:5432/movies_d
engine = create_engine(db_string)
movies_df.to_sql(name='movies', con=engine)

rows_imported = 0
start_time = time.time()
for data in pd.read_csv('ratings.csv', chunksize=1000000):
    print(f'importing rows {rows_imported} to {rows_imported + len(data
    data.to_sql(name='ratings', con=engine, if_exists='append')
    rows_imported += len(data)

    print(f'Done. {time.time() - start_time} total seconds elapsed')
```

```
importing rows 0 to 1000000...Done. 252.13348007202148 total seconds el
apsed
importing rows 1000000 to 2000000...Done. 513.536689043045 total second
s elapsed
importing rows 2000000 to 3000000...Done. 922.1973788738251 total secon
ds elapsed
importing rows 3000000 to 4000000...Done. 1332.1677091121674 total seco
nds elapsed
importing rows 4000000 to 5000000...Done. 1619.4044170379639 total seco
nds elapsed
importing rows 5000000 to 6000000...Done. 1927.9038622379303 total seco
nds elapsed
importing rows 6000000 to 7000000...Done. 2185.834051847458 total secon
ds elapsed
importing rows 7000000 to 8000000...Done. 2467.8625650405884 total seco
nds elapsed
importing rows 8000000 to 9000000...Done. 2729.9847791194916 total seco
nds elapsed
importing rows 9000000 to 10000000...Done. 2983.683515071869 total seco
nds elapsed
importing rows 10000000 to 11000000...Done. 3214.713099002838 total sec
onds elapsed
importing rows 11000000 to 12000000...Done. 3452.246883869171 total sec
onds elapsed
importing rows 12000000 to 13000000...Done. 3677.7869729995728 total se
conds elapsed
importing rows 13000000 to 14000000...Done. 3879.994642972946 total sec
onds elapsed
importing rows 14000000 to 15000000...Done. 4225.885064125061 total sec
onds elapsed
importing rows 15000000 to 16000000...Done. 4503.6516308784485 total se
conds elapsed
importing rows 16000000 to 17000000...Done. 4778.14617395401 total seco
nds elapsed
importing rows 17000000 to 18000000...Done. 5043.797405004501 total sec
onds elapsed
importing rows 18000000 to 19000000...Done. 5295.373776912689 total sec
onds elapsed
importing rows 19000000 to 20000000...Done. 5541.271307229996 total sec
onds elapsed
importing rows 20000000 to 21000000...Done. 5775.770133018494 total sec
onds elapsed
```

```
importing rows 21000000 to 22000000...Done. 5950.776211023331 total sec
onds elapsed
importing rows 22000000 to 23000000...Done. 6132.967226028442 total sec
onds elapsed
importing rows 23000000 to 24000000...Done. 6302.955336093903 total sec
onds elapsed
importing rows 24000000 to 25000000...Done. 6479.958534002304 total sec
onds elapsed
importing rows 25000000 to 26000000...Done. 6669.753993034363 total sec
onds elapsed
importing rows 26000000 to 26024289...Done. 6676.92467212677 total seco
nds elapsed
```

In [14]:

```
# 10. Create the path to your file directory and variables for the thre
file_dir = "/Users/caroline/Documents/Data Boot Camp/Module 8/Movies-ET
# The Wikipedia data
wiki_file = f'{file_dir}/wikipedia.movies.json'
# The Kaggle metadata
kaggle_file = f'{file_dir}/movies_metadata.csv'
# The MovieLens rating data.
ratings_file = f'{file_dir}/ratings.csv'
```

In [ ]:

```
# 11. Set the three variables equal to the function created in D1.
wiki_file, kaggle_file, ratings_file = movies_function()
```