In [20]:

```python
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 [21]:

```python
# 1. 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 [22]:
```
# 2 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)

    # Return three variables. The first is the wiki_movies_df DataFram
    return wiki_movies_df, kaggle_metadata, ratings
```

In [23]:
```
# 17. 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 [24]:
```
# 18. Set the three variables equal to the function created in D1.
wiki_file, kaggle_file, ratings_file = movies_function()
```

In [25]:
```
# 19. Set the wiki_movies_df equal to the wiki_file variable.
wiki_movies_df = wiki_file
```

In [26]:
```
# 20. Check that the wiki_movies_df DataFrame looks like this.
wiki_movies_df.head()
```

Out[26]:

	url	year	imdb_link	tit
0	https://en.wikipedia.org/wiki/The_Adventures_o...	1990.0	https://www.imdb.com/title/tt0098987/	Tl Adventure of Fo Fairlar
1	https://en.wikipedia.org/wiki/After_Dark,_My_S...	1990.0	https://www.imdb.com/title/tt0098994/	After Dar My Swe
2	https://en.wikipedia.org/wiki/Air_America_(film)	1990.0	https://www.imdb.com/title/tt0099005/	/ Americ
3	https://en.wikipedia.org/wiki/Alice_(1990_film)	1990.0	https://www.imdb.com/title/tt0099012/	Alic
4	https://en.wikipedia.org/wiki/Almost_an_Angel	1990.0	https://www.imdb.com/title/tt0099018/	Almost a Ang

5 rows × 23 columns

In [27]:
```
# 21. Check that wiki_movies_df DataFrame columns are correct.
wiki_movies_df.columns.to_list()
```

Out[27]:
```
['url',
 'year',
 'imdb_link',
 'title',
 'Based on',
 'Starring',
 'Cinematography',
 'Release date',
 'Country',
 'Language',
 'Budget',
 'Director',
 'Distributor',
 'Editor(s)',
 'Composer(s)',
 'Producer(s)',
 'Production company(s)',
 'Writer(s)',
 'imdb_id',
 'box_office',
 'budget',
 'release_date',
 'running_time']
```

In [ ]:
```

```