SI 507 Final Project Submission

Zekun Zhao

ID: 01160883

Project code:

Github repo url: https://github.com/cooeoeooc/Zekun-Zhao-Final-Project

Readme:

(The readme file also can be accessed from the github repository)

The main project is "Final Project." I also upload the "secrets", which contains my API key for this program.

There are several important functions in the file, and I also add doc string to every one of them.

make_url_request_using_cache: Check the cache for a saved result for this url. If the result is found, return it. Otherwise send a new request, save it, then return it.

get_imdb_top250: generate a list of tuple, the tuple formed by url, rank, name, year, id of Top 250 IMBD movies.

make_request_api: Make a request to the Web API using the baseurl and params and return a dictionary

create_imdb_database: generate imdb database using information from IMDB(list of tuple)

create_omdb_database: generate imdb database using information from OMDB(list of tuple)

make_language_bar_plot, make_genre_bar_plot, make_country_bar_plot, make_rated_bar_plot: link to database and based on different criteria making bar plots

After open the link, users can click on four links to view graphs about the movies' data, or input a number to search movie's information.

SI 507 Final Project Submission

Zekun Zhao

ID: 01160883

To run this program, users need to import:

sqlite3

requests

BeautifulSoup

Secrets

Plotly

Flask

Os

Data Sources:

I used two websites for collecting data sources. The first one is https://www.imdb.com/chart/top. I **scraped** IMDB top rated 250 movies from this website into **json** file and saved the information into local **cache** file. The second one is http://www.omdbapi.com/ (The documentation is also in the website) Through API search, I found more details about these 250 movies and save them into cache file. I found and save 250 movie records. From IMDB website, I collect fields as following:

- movie_name
- movie_year: the release year of the movie
- movie_rank: the rank of the movie among IMDB TOP 250 Movies
- movie_url: the IMDB url of the movie
- movie_imdb_id: the IMDB id of the movie

From the OMDB API, I collect fields as Following:

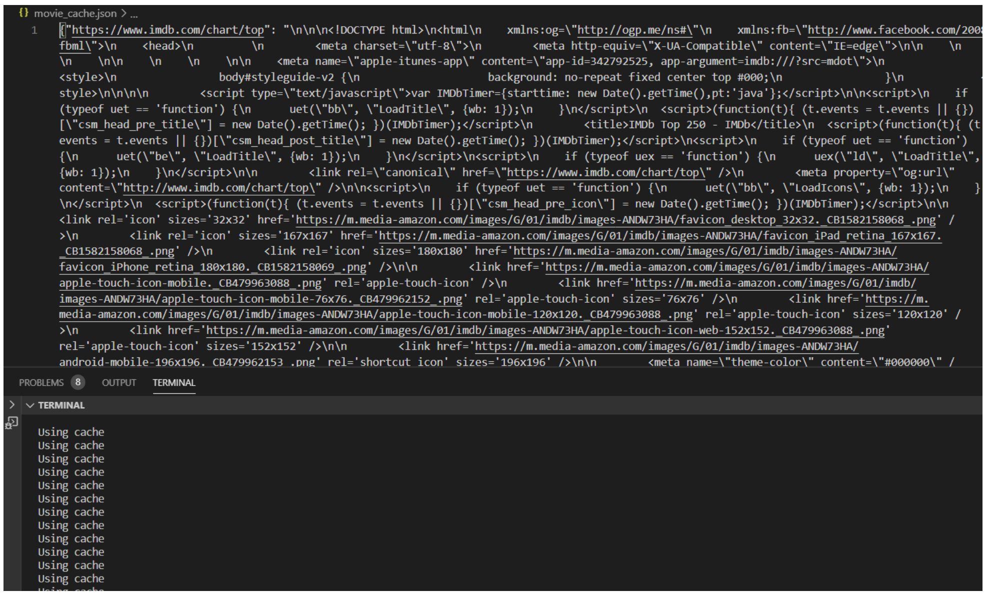
- Movie_Rate(R or PG)
- Movie_Runtime
- Movie_Director
- IMDB_Rating
- Rotten_Tomatoes_Rating
- Metascore: score of Metacritic website
- Movie_Production: The production company of the movie (i.e. Paramount Pictures, Universal Pictures)
- Movie_Genre: (i.e. Comedy, Drama, Action).

SI 507 Final Project Submission

Zekun Zhao

ID: 01160883

Evidence of Caching:

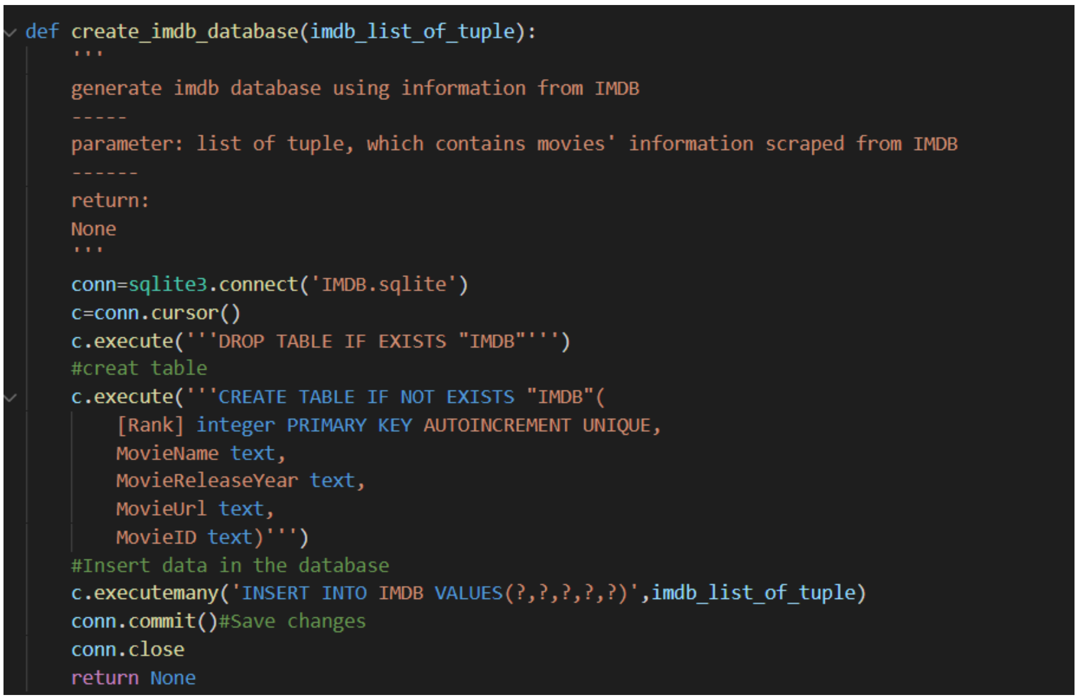


```
{"https://www.imdb.com/chart/top": "\n\n\n<!DOCTYPE html>\n<html\n    xmlns:og=\"http://ogp.me/ns#\"\n    xmlns:fb=\"http://www.facebook.com/200
fbml\">\n    <head>\n         \n        <meta charset=\"utf-8\">\n        <meta http-equiv=\"X-UA-Compatible\" content=\"IE=edge\">\n\n    \n
\n    \n\n    \n    \n    \n\n    <meta name=\"apple-itunes-app\" content=\"app-id=342792525, app-argument=imdb:///?src=mdot\">\n
<style>\n              body#styleguide-v2 {\n                  background: no-repeat fixed center top #000;\n              }\n
style>\n\n\n\n        <script type=\"text/javascript\">var IMDbTimer={starttime: new Date().getTime(),pt:'java'};</script>\n\n<script>\n    if
(typeof uet == 'function') {\n      uet(\"bb\", \"LoadTitle\", {wb: 1});\n    }\n</script>\n  <script>(function(t){ (t.events = t.events || {})
[\"csm_head_pre_title\"] = new Date().getTime(); })(IMDbTimer);</script>\n        <title>IMDb Top 250 - IMDb</title>\n  <script>(function(t){ (t
events = t.events || {})[\"csm_head_post_title\"] = new Date().getTime(); })(IMDbTimer);</script>\n<script>\n    if (typeof uet == 'function')
{\n      uet(\"be\", \"LoadTitle\", {wb: 1});\n    }\n</script>\n<script>\n    if (typeof uex == 'function') {\n      uex(\"ld\", \"LoadTitle\",
{wb: 1});\n    }\n</script>\n\n        <link rel=\"canonical\" href=\"https://www.imdb.com/chart/top\" />\n        <meta property=\"og:url\"
content=\"http://www.imdb.com/chart/top\" />\n\n<script>\n    if (typeof uet == 'function') {\n      uet(\"bb\", \"LoadIcons\", {wb: 1});\n    }
\n</script>\n  <script>(function(t){ (t.events = t.events || {})[\"csm_head_pre_icon\"] = new Date().getTime(); })(IMDbTimer);</script>\n\n
<link rel='icon' sizes='32x32' href='https://m.media-amazon.com/images/G/01/imdb/images-ANDW73HA/favicon_desktop_32x32._CB1582158068_.png' /
>\n        <link rel='icon' sizes='167x167' href='https://m.media-amazon.com/images/G/01/imdb/images-ANDW73HA/favicon_iPad_retina_167x167.
_CB1582158068_.png' />\n        <link rel='icon' sizes='180x180' href='https://m.media-amazon.com/images/G/01/imdb/images-ANDW73HA/
favicon_iPhone_retina_180x180._CB1582158069_.png' />\n\n        <link href='https://m.media-amazon.com/images/G/01/imdb/images-ANDW73HA/
apple-touch-icon-mobile._CB479963088_.png' rel='apple-touch-icon' />\n        <link href='https://m.media-amazon.com/images/G/01/imdb/
images-ANDW73HA/apple-touch-icon-mobile-76x76._CB479962152_.png' rel='apple-touch-icon' sizes='76x76' />\n        <link href='https://m.
media-amazon.com/images/G/01/imdb/images-ANDW73HA/apple-touch-icon-mobile-120x120._CB479963088_.png' rel='apple-touch-icon' sizes='120x120' /
>\n        <link href='https://m.media-amazon.com/images/G/01/imdb/images-ANDW73HA/apple-touch-icon-web-152x152._CB479963088_.png'
rel='apple-touch-icon' sizes='152x152' />\n\n        <link href='https://m.media-amazon.com/images/G/01/imdb/images-ANDW73HA/
android-mobile-196x196._CB479962153_.png' rel='shortcut icon' sizes='196x196' />\n\n        <meta name=\"theme-color\" content=\"#000000\" /
```

```
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
Using cache
```

DataBase:

DataBase Schema:

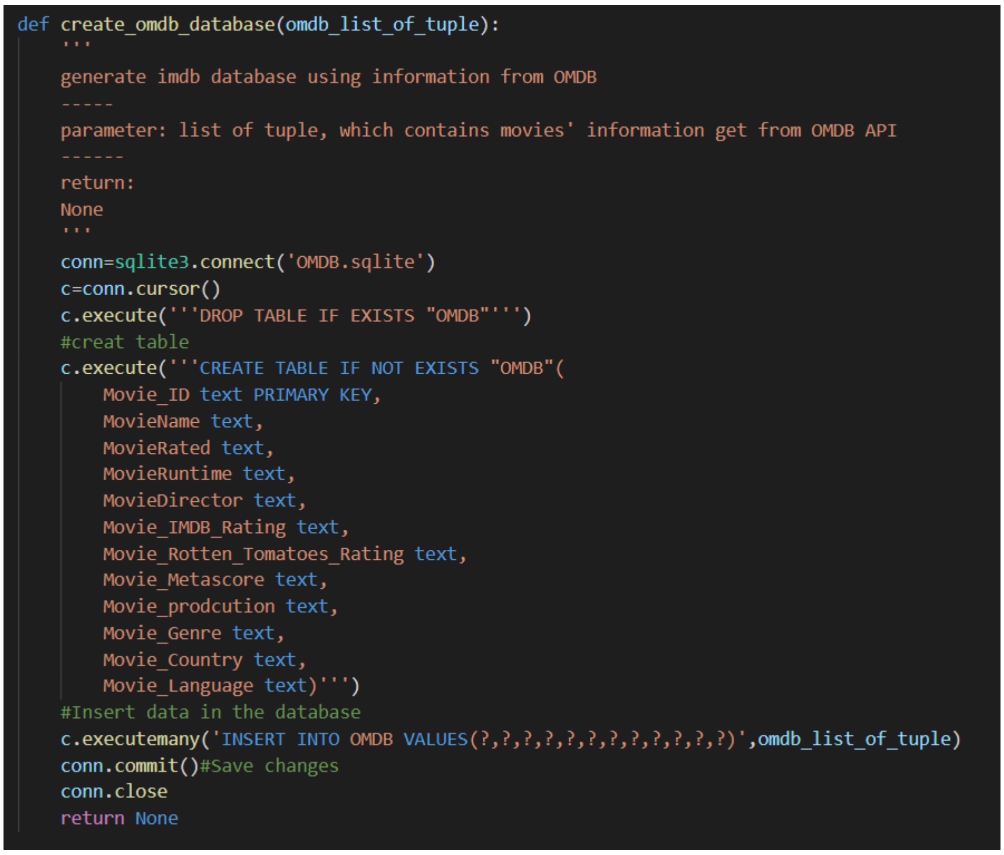


```python
def create_imdb_database(imdb_list_of_tuple):
    '''
    generate imdb database using information from IMDB
    -----
    parameter: list of tuple, which contains movies' information scraped from IMDB
    ------
    return:
    None
    '''
    conn=sqlite3.connect('IMDB.sqlite')
    c=conn.cursor()
    c.execute('''DROP TABLE IF EXISTS "IMDB"''')
    #creat table
    c.execute('''CREATE TABLE IF NOT EXISTS "IMDB"(
        [Rank] integer PRIMARY KEY AUTOINCREMENT UNIQUE,
        MovieName text,
        MovieReleaseYear text,
        MovieUrl text,
        MovieID text)''')
    #Insert data in the database
    c.executemany('INSERT INTO IMDB VALUES(?,?,?,?,?)',imdb_list_of_tuple)
    conn.commit()#Save changes
    conn.close
    return None
```

SI 507 Final Project Submission

Zekun Zhao

ID: 01160883



```python
def create_omdb_database(omdb_list_of_tuple):
    '''
    generate imdb database using information from OMDB
    -----
    parameter: list of tuple, which contains movies' information get from OMDB API
    ------
    return:
    None
    '''
    conn=sqlite3.connect('OMDB.sqlite')
    c=conn.cursor()
    c.execute('''DROP TABLE IF EXISTS "OMDB"''')
    #creat table
    c.execute('''CREATE TABLE IF NOT EXISTS "OMDB"(
        Movie_ID text PRIMARY KEY,
        MovieName text,
        MovieRated text,
        MovieRuntime text,
        MovieDirector text,
        Movie_IMDB_Rating text,
        Movie_Rotten_Tomatoes_Rating text,
        Movie_Metascore text,
        Movie_prodcution text,
        Movie_Genre text,
        Movie_Country text,
        Movie_Language text)''')
    #Insert data in the database
    c.executemany('INSERT INTO OMDB VALUES(?,?,?,?,?,?,?,?,?,?,?,?)',omdb_list_of_tuple)
    conn.commit()#Save changes
    conn.close
    return None
```

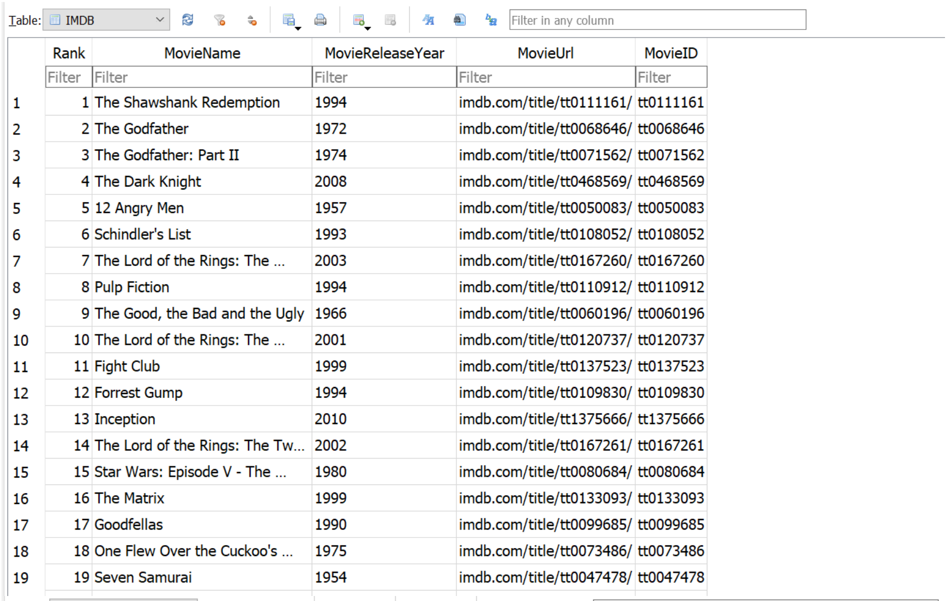
The primary key of my IMDB database is rank, and the primary key of my OMDB database is Movie_ID (the IMDB movie id), which is also a foreign key linked my two tables.

Screenshots of my tables:

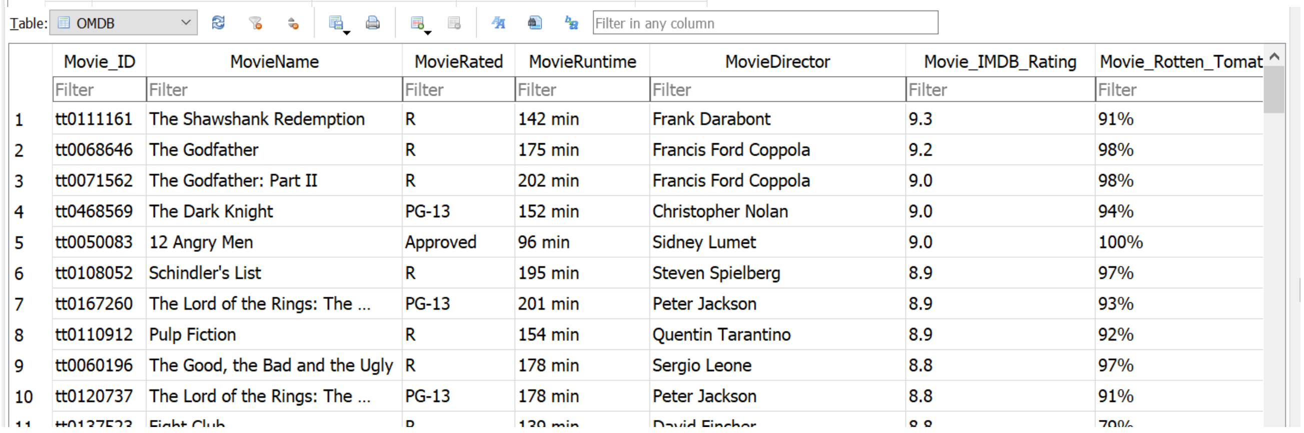
SI 507 Final Project Submission

Zekun Zhao

ID: 01160883



	Rank	MovieName	MovieReleaseYear	MovieUrl	MovieID
	Filter	Filter	Filter	Filter	Filter
1	1	The Shawshank Redemption	1994	imdb.com/title/tt0111161/	tt0111161
2	2	The Godfather	1972	imdb.com/title/tt0068646/	tt0068646
3	3	The Godfather: Part II	1974	imdb.com/title/tt0071562/	tt0071562
4	4	The Dark Knight	2008	imdb.com/title/tt0468569/	tt0468569
5	5	12 Angry Men	1957	imdb.com/title/tt0050083/	tt0050083
6	6	Schindler's List	1993	imdb.com/title/tt0108052/	tt0108052
7	7	The Lord of the Rings: The ...	2003	imdb.com/title/tt0167260/	tt0167260
8	8	Pulp Fiction	1994	imdb.com/title/tt0110912/	tt0110912
9	9	The Good, the Bad and the Ugly	1966	imdb.com/title/tt0060196/	tt0060196
10	10	The Lord of the Rings: The ...	2001	imdb.com/title/tt0120737/	tt0120737
11	11	Fight Club	1999	imdb.com/title/tt0137523/	tt0137523
12	12	Forrest Gump	1994	imdb.com/title/tt0109830/	tt0109830
13	13	Inception	2010	imdb.com/title/tt1375666/	tt1375666
14	14	The Lord of the Rings: The Tw...	2002	imdb.com/title/tt0167261/	tt0167261
15	15	Star Wars: Episode V - The ...	1980	imdb.com/title/tt0080684/	tt0080684
16	16	The Matrix	1999	imdb.com/title/tt0133093/	tt0133093
17	17	Goodfellas	1990	imdb.com/title/tt0099685/	tt0099685
18	18	One Flew Over the Cuckoo's ...	1975	imdb.com/title/tt0073486/	tt0073486
19	19	Seven Samurai	1954	imdb.com/title/tt0047478/	tt0047478



	Movie_ID	MovieName	MovieRated	MovieRuntime	MovieDirector	Movie_IMDB_Rating	Movie_Rotten_Tomat
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	tt0111161	The Shawshank Redemption	R	142 min	Frank Darabont	9.3	91%
2	tt0068646	The Godfather	R	175 min	Francis Ford Coppola	9.2	98%
3	tt0071562	The Godfather: Part II	R	202 min	Francis Ford Coppola	9.0	98%
4	tt0468569	The Dark Knight	PG-13	152 min	Christopher Nolan	9.0	94%
5	tt0050083	12 Angry Men	Approved	96 min	Sidney Lumet	9.0	100%
6	tt0108052	Schindler's List	R	195 min	Steven Spielberg	8.9	97%
7	tt0167260	The Lord of the Rings: The ...	PG-13	201 min	Peter Jackson	8.9	93%
8	tt0110912	Pulp Fiction	R	154 min	Quentin Tarantino	8.9	92%
9	tt0060196	The Good, the Bad and the Ugly	R	178 min	Sergio Leone	8.8	97%
10	tt0120737	The Lord of the Rings: The ...	PG-13	178 min	Peter Jackson	8.8	91%
11	tt0137523	Fight Club	R	139 min	David Fincher	8.8	79%

Interaction and Presentation Plans:

For this part, users can input search key word to find movie's information. For example, if the user type the rank number of a movie, the movie's information should pop out like <movie_name> is a <release_year> movie, and directed by<movie_director>.

For example

1. <movie name> is a <genre> released in <movie_year> directed by<director's name>
2. …
3. …
4. …

Moreover, I use different criteria to group those movies and use plotly to show the number of different groups. For example, I can group by genre, group by language,

SI 507 Final Project Submission

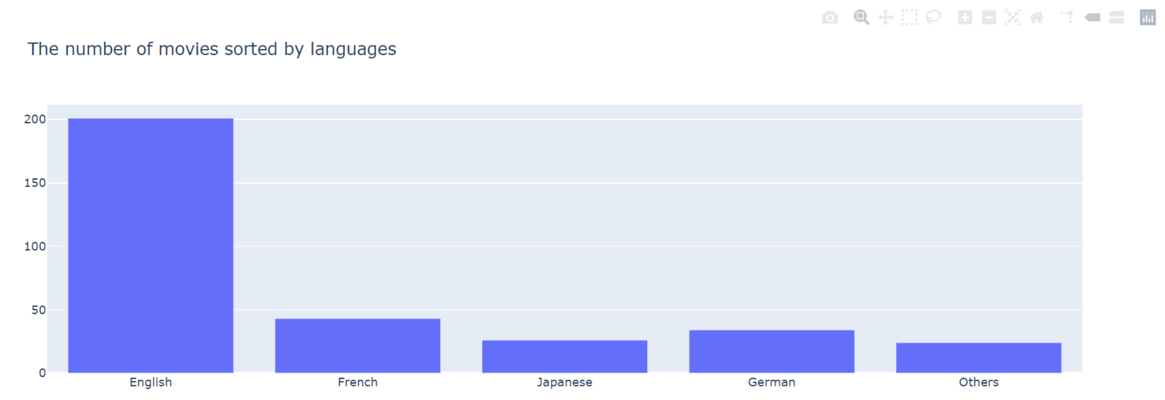
Zekun Zhao

ID: 01160883

group by country, and group by rating. Through click on different links to those graph, users can see different bar plots based on different sorting methods.

Here is an example of those graph:

Graph about IMDB TOP 250 Movies



The link of my demo video: https://youtu.be/7HnkQFJ9-o8