SI 507 Final Project Checkpoint

Zekun Zhao

ID: 01160883

Project code:

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

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:

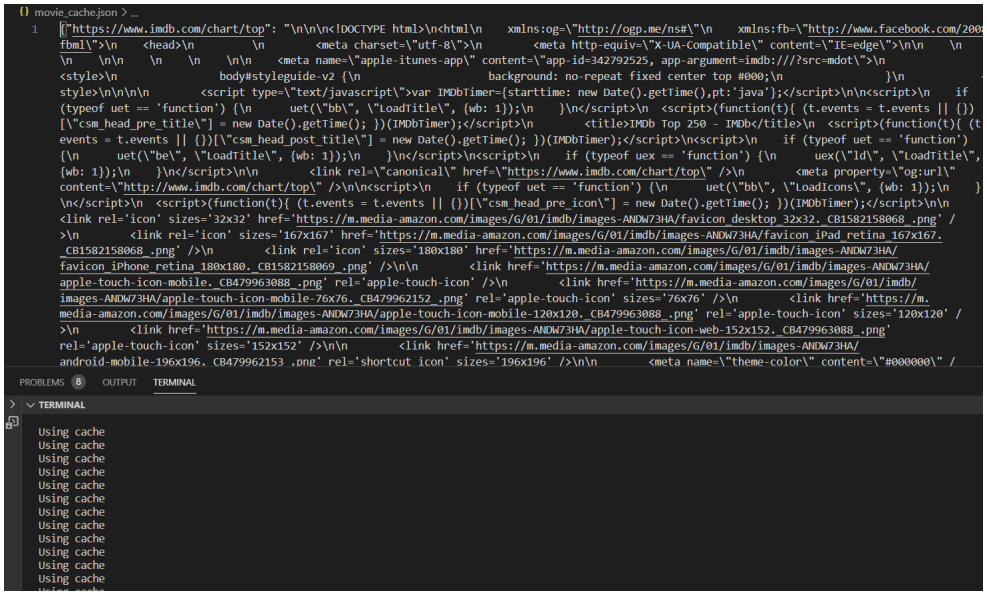
- Movie_Rate(R or PG)
- Movie_Runtime
- Movie_Director
- IMDB_Rating
- Rotten_Tomatoes_Rating
- Metascore: score of Metacritic website
- Movie_Production: The production company of the movie (i.e. Paramount Pictures, Universal Pictures)
- Movie_Genre: (i.e. Comedy, Drama, Action).

Evidence of Caching:

SI 507 Final Project Checkpoint

Zekun Zhao

ID: 01160883



DataBase:

DataBase Schema:

```
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 Checkpoint

Zekun Zhao

ID: 01160883

```
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 Checkpoint

Zekun Zhao

ID: 01160883

Table: IMDB

Filter in any column

	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

Table: OMDB

Filter in any column

	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>.

If the user type one kind of language (i.e. French), then all French movies will form a list like:

1. <movie name> is a <genre> released in <movie_year>
2. …
3. …
4. …

From this step, the user can also choose to type a number to get the details about the movie.

SI 507 Final Project Checkpoint

Zekun Zhao

ID: 01160883

Like language, if a user input a genre type (i.e. “Comedy”), then the program will form a list of comedy, as following:

1.<movie name> is a <genre>, language:<movie_language>, country:<movie_country>

2.…

3.…

4.…

For next step, the user also can input the number of movie to get more details of the movie.

Moreover, I plan to 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, group by country, and group by imdb_rating.