MySQL IMDb Dataset Project

D. L. Whittenbury

February, 2020

Contents

Introduction

In this project we will build a MySQL database using the Internet Movie Database (IMDb) dataset. The dataset consists of 7 compressed tab-separated-value (*.tsv) files, which are explained and available for download from here. The data is refreshed daily, although the data used in this project was obtained on **29/11/2019**.

The purpose of this project is to do the following:

- Learn about and use the database management system MySQL.
- Learn the essentials of database design, e.g., Entity-Relationship diagrams, logical schema, and database normalisation.
- Practice database querying by posing basic and more advanced queries using MySQL directly and also indirectly using python.
- Visualise IMDb data using python.

The tangible steps we will take in this project are:

- Understand the data in the IMDb dataset.
- Design a relational database and store the IMDb data in it.
  - Model the database using an Entity-Relationship (ER) diagram.
  - Perform normalisation and restructure the IMDb data using python.
  - Create MySQL database.
  - Load data into the database.
  - Add primary and foreign key constraints.
  - Create database indexes.
- Ask questions of the IMDb data, so as to practice simple and more advanced SQL queries.
- Perform further exploration and also visualisation of the data using python.
- Throughout we will adhere to SQL style conventions, e.g., please see SQL Style guide https://www.sqlstyle.guide/. In particular, underscores will be used in attribute names rather than camel case, which is used in the IMDb data files.

Dataset details

Each dataset is contained in a gzipped tab-separated-values (TSV) formatted file in the UTF-8 character set. The first line in each file contains headers that describe what is in each column. A "\N" is used to denote that a particular field is missing or has a NULL value for that title or name. It should be noted that the data available for download from the IMDb website is not the full dataset, but it will suffice for our purposes. The available IMDb data files are as follows:

name.basics.tsv.gz

Contains the following information for names:

- nconst (string) - alphanumeric unique identifier of the name/person.
- primaryName (string)– name by which the person is most often credited.
- birthYear – in YYYY format.
- deathYear – in YYYY format if applicable, else "\N".
- primaryProfession (array of strings) – the top-3 professions of the person.
- knownForTitles (array of tconsts) – titles the person is known for.

title.basics.tsv.gz

Contains the following information for titles:

- tconst (string) - alphanumeric unique identifier of the title.
- titleType (string) – the type/format of the title (e.g. movie, short, tvseries, tvepisode, video, etc).
- primaryTitle (string) – the more popular title / the title used by the filmmakers on promotional materials at the point of release.
- originalTitle (string) - original title, in the original language.
- isAdult (boolean) - 0: non-adult title; 1: adult title.
- startYear (YYYY) – represents the release year of a title. In the case of TV Series, it is the series start year.
- endYear (YYYY) – TV Series end year. "\N" for all other title types.
- runtimeMinutes – primary runtime of the title, in minutes.
- genres (string array) – includes up to three genres associated with the title.

title.akas.tsv.gz

Contains the following information for titles:

- titleId (string) - a tconst which is an alphanumeric unique identifier of the title.
- ordering (integer) – a number to uniquely identify rows for a given titleId.
- title (string) – the localised title.
- region (string) - the region for this version of the title.
- language (string) - the language of the title.
- types (array) - Enumerated set of attributes for this alternative title. One or more of the following: "alternative", "dvd", "festival", "tv", "video", "working", "original", "imdbDisplay". New values may be added in the future without warning. **Please note that types is said to be an array. In the data we have this appears to not be true. There appears to be only one string for each pair of titleId and ordering values. Also, there are many NULL (\N) values in this field (~95%).**
- attributes (array) - Additional terms to describe this alternative title, not enumerated. **Please note that attributes is said to be an array. In the data we have this appears to not be true. There appears to be only one string for each pair of titleId and ordering values. There are many NULL (\N) values in this field (~99%).**
- isOriginalTitle (boolean) – 0: not original title; 1: original title.

title.crew.tsv.gz

Contains the director and writer information for all the titles in IMDb. Fields include:

- tconst (string) - alphanumeric unique identifier of the title.
- directors (array of nconsts) - director(s) of the given title.
- writers (array of nconsts) – writer(s) of the given title.

title.episode.tsv.gz

Contains the tv episode information. Fields include:

- tconst (string) - alphanumeric identifier of episode.
- parentTconst (string) - alphanumeric identifier of the parent TV Series.
- seasonNumber (integer) – season number the episode belongs to.
- episodeNumber (integer) – episode number of the tconst in the TV series.

title.principals.tsv.gz

Contains the principal cast/crew for titles

- tconst (string) - alphanumeric unique identifier of the title.
- ordering (integer) – a number to uniquely identify rows for a given titleId.
- nconst (string) - alphanumeric unique identifier of the name/person.
- category (string) - the category of job that person was in.
- job (string) - the specific job title if applicable, else "\N".
- characters (string) - the name of the character played if applicable, else "\N" (It is really "[role1,role2,. . . .]" or "\N").

title.ratings.tsv.gz

Contains the IMDb rating and votes information for titles

- tconst (string) - alphanumeric unique identifier of the title.
- averageRating – weighted average of all the individual user ratings.
- numVotes - number of votes the title has received.

IMDb dataset license details

Subsets of IMDb data are available for access to customers for personal and non-commercial use. You can hold local copies of this data, and it is subject to our terms and conditions. Please refer to the Non-Commercial Licensing and copyright/license and verify compliance.

A few WARNINGs about the IMDb dataset

The IMDb data base contains details of adult titles. If this is likely to offend you or others that see your work, then you can filter these titles out by using the field isAdult in title.basics.tsv.gz.

The IMDb dataset is a noisy dataset. In particular, there are some issues with missing data which affected how we added constraints to the database. These are discussed in the note IMDb_data_issues.md. Of course, missing data could be obtained by scraping the IMDb but this is beyond the scope of the project.

Entity-Relationship (ER) diagram

The IMDb data as provided is not normalised. We first design an entity-relationship diagram for our IMDb relational database. This is shown below.

Logical schema

We then normalise our ER diagram and obtain the logical schema illustrated below. Note the following:

- New tables were created for multi-valued attributes, such as Title_genres.
- We pulled the rating information attributes from the Titles entity, because many titles didn't have a rating. If we were to store them in the Titles table, then we would have stored many NULL values. Instead we decided to separate this information, by putting it into the table Title_ratings.

Prepare the IMDb data and build the IMDb database

1) Preprocess the IMDb dataset

The IMDb data is preprocessed using python. The script `imdb_converter.py` reads in the 7 data files, cleans and normalises the IMDb data. After which, the desired set of tables are output as tab-separate-value (tsv) files.

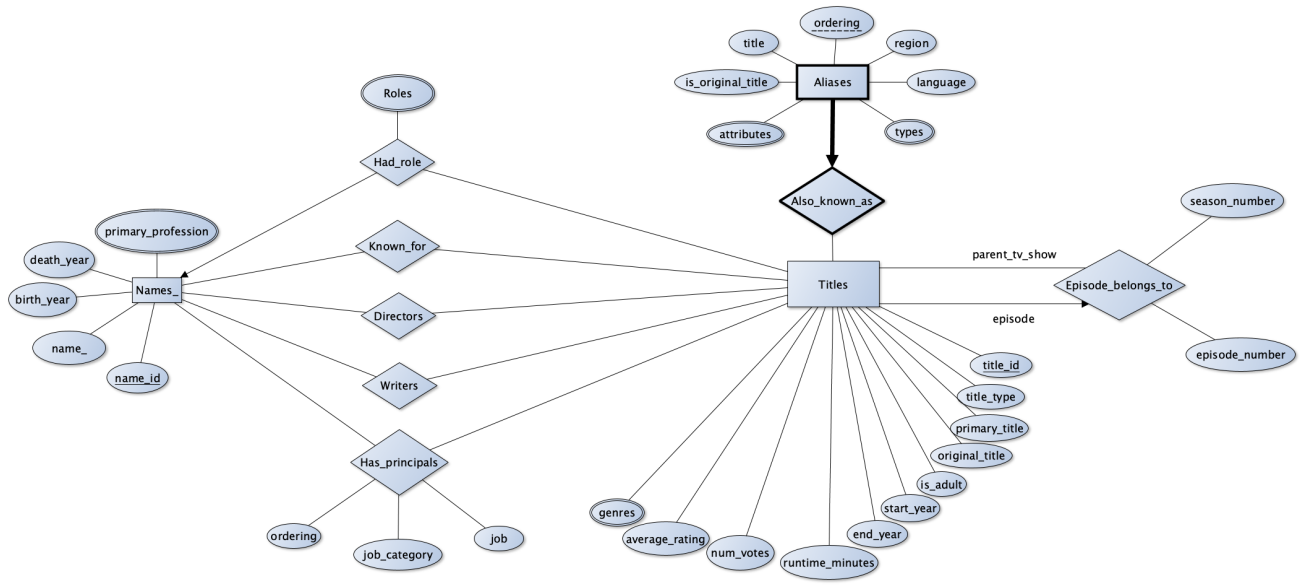


Figure 1: IMDb ER diagram.

2) Build MySQL database

The sequence of steps to build the database are as follows:

1. Open MySQL in terminal:

```
$ mysql -u root -p --local-infile
```

The SQL commands to build the database described above by the ER diagram and logical schema are contained in 4 `*.sql scripts`:

- imdb-create-tables.sql
- imdb-load-data.sql
- imdb-add-constraints.sql
- imdb-index-tables.sql

2. Create IMDb data base in MySQL:

```
mysql> SOURCE /Users/lappy/Git_repos_mine/MySQL_IMDb_Project/imdb-create-tables.sql
```

3. Load data using this script in MySQL:

```
mysql> SOURCE /Users/lappy/Git_repos_mine/MySQL_IMDb_Project/imdb-load-data.sql
```

4. Add constraints to the IMDb database in MySQL

```
mysql> SOURCE /Users/lappy/Git_repos_mine/MySQL_IMDb_Project/imdb-add-constraints.sql
```

5. Add indexes to the IMDb database in MySQL

```
mysql> SOURCE /Users/lappy/Git_repos_mine/MySQL_IMDb_Project/imdb-index-tables.sql
```

SQL Queries using MySQL

After creating and loading data into the database, we can now pose queries to it. In the file `SQL_Queries_1.sql` we consider many questions and answer them by querying the IMDb database. This section is an on going piece of work. We intend to add more questions and queries to the repository as we practice SQL.

For each query in the file `SQL_Queries_1.sql` we create a view by

```
CREATE OR REPLACE VIEW Q1(column_1,column_2)
AS
...
;
```

The result of the query which is stored in the view can be seen by

```
SELECT * FROM Q1;
```

To delete the view

```
DROP VIEW Q1;
```

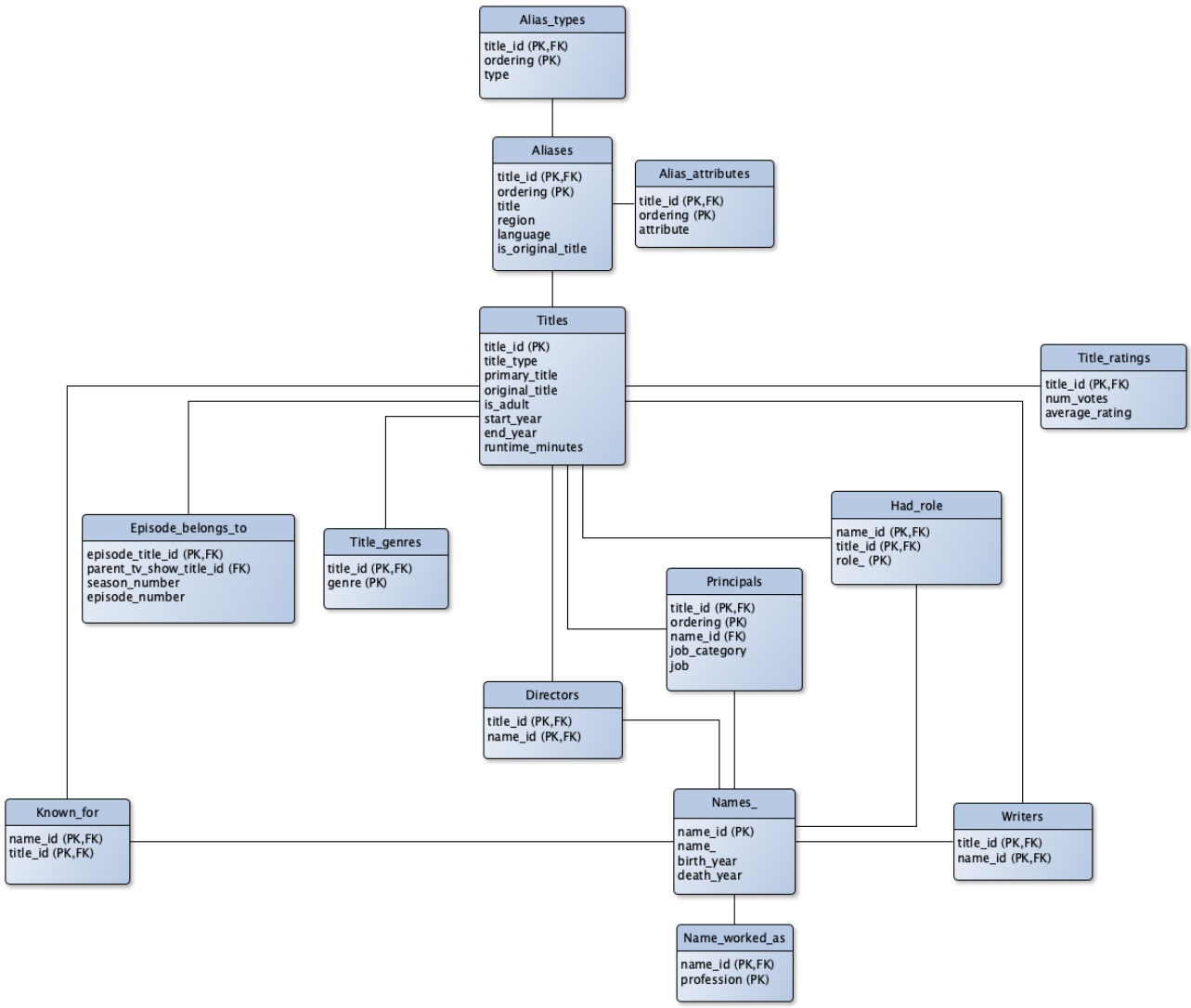


Figure 2: IMDb logical schema diagram.

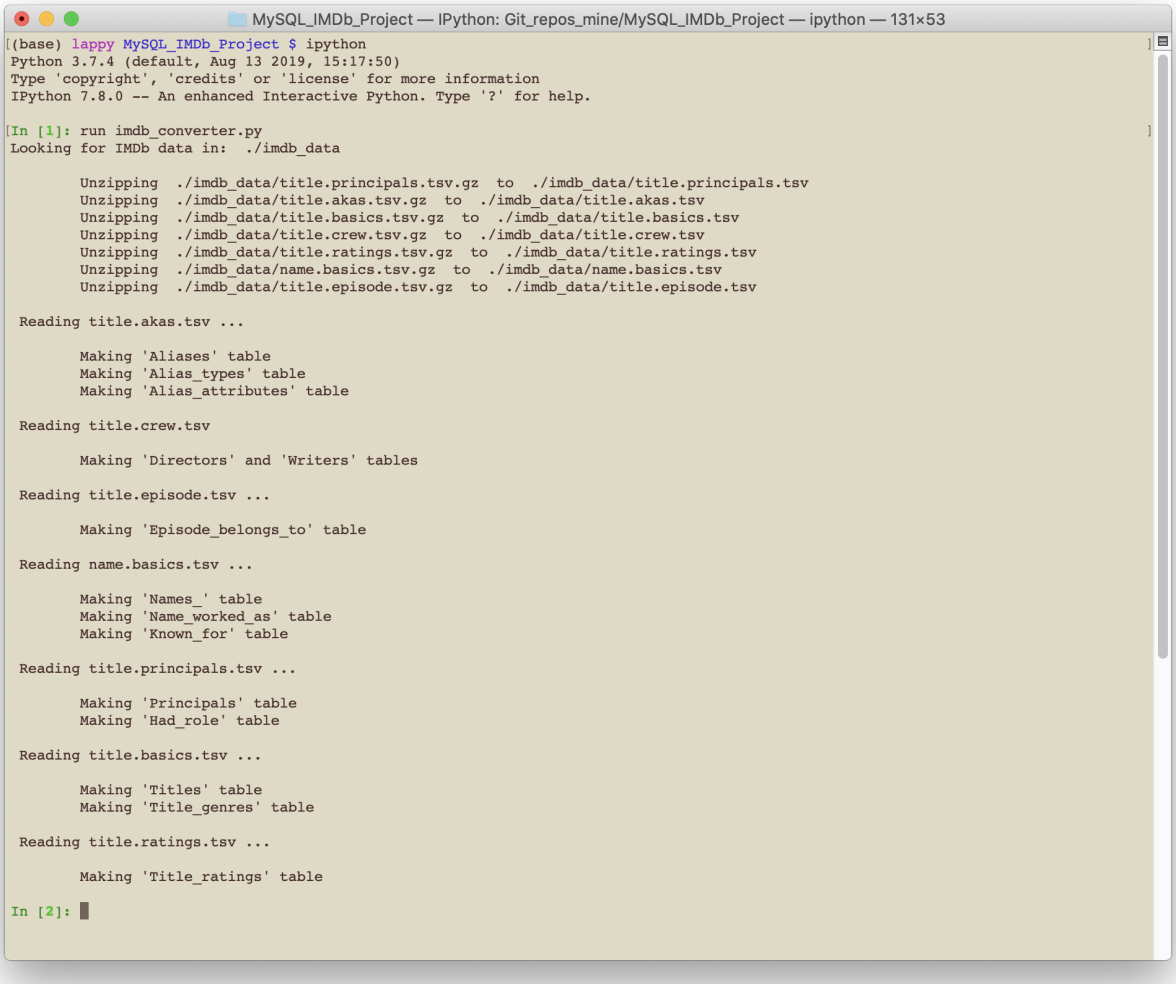


Figure 3: Terminal screenshot imdb_converter.py

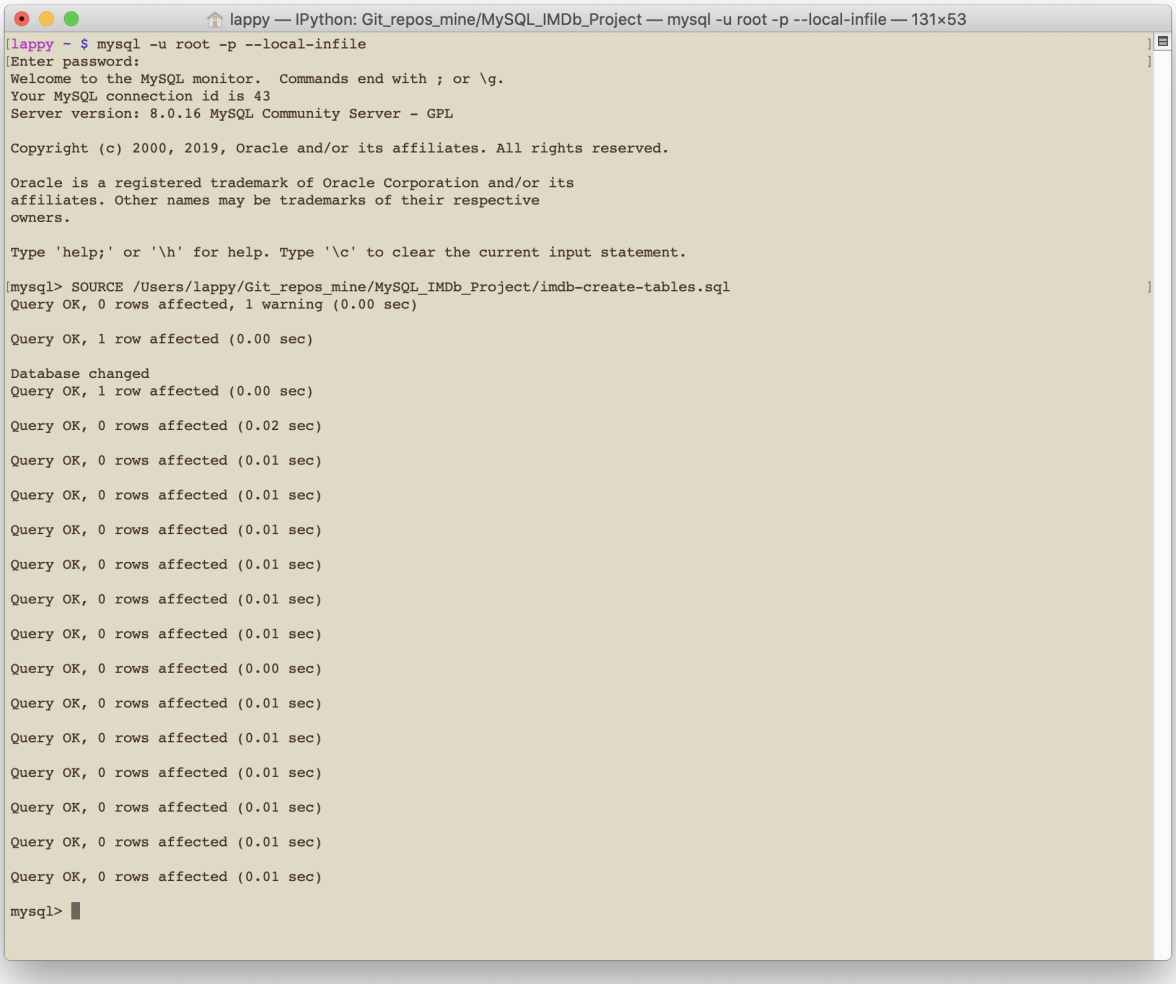


Figure 4: Terminal screenshot of running imdb-create-tables.sql in MySQL

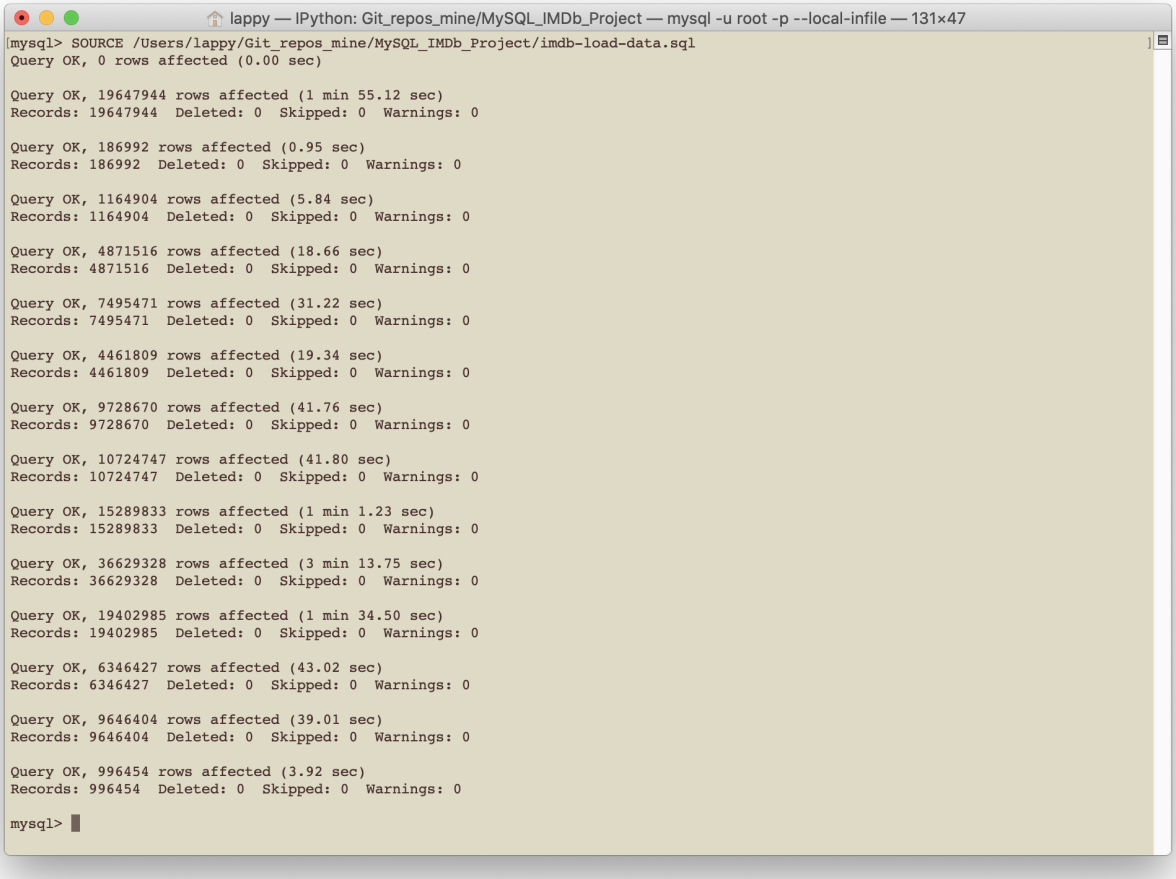


Figure 5: Terminal screenshot of running imdb-load-data.sql in MySQL

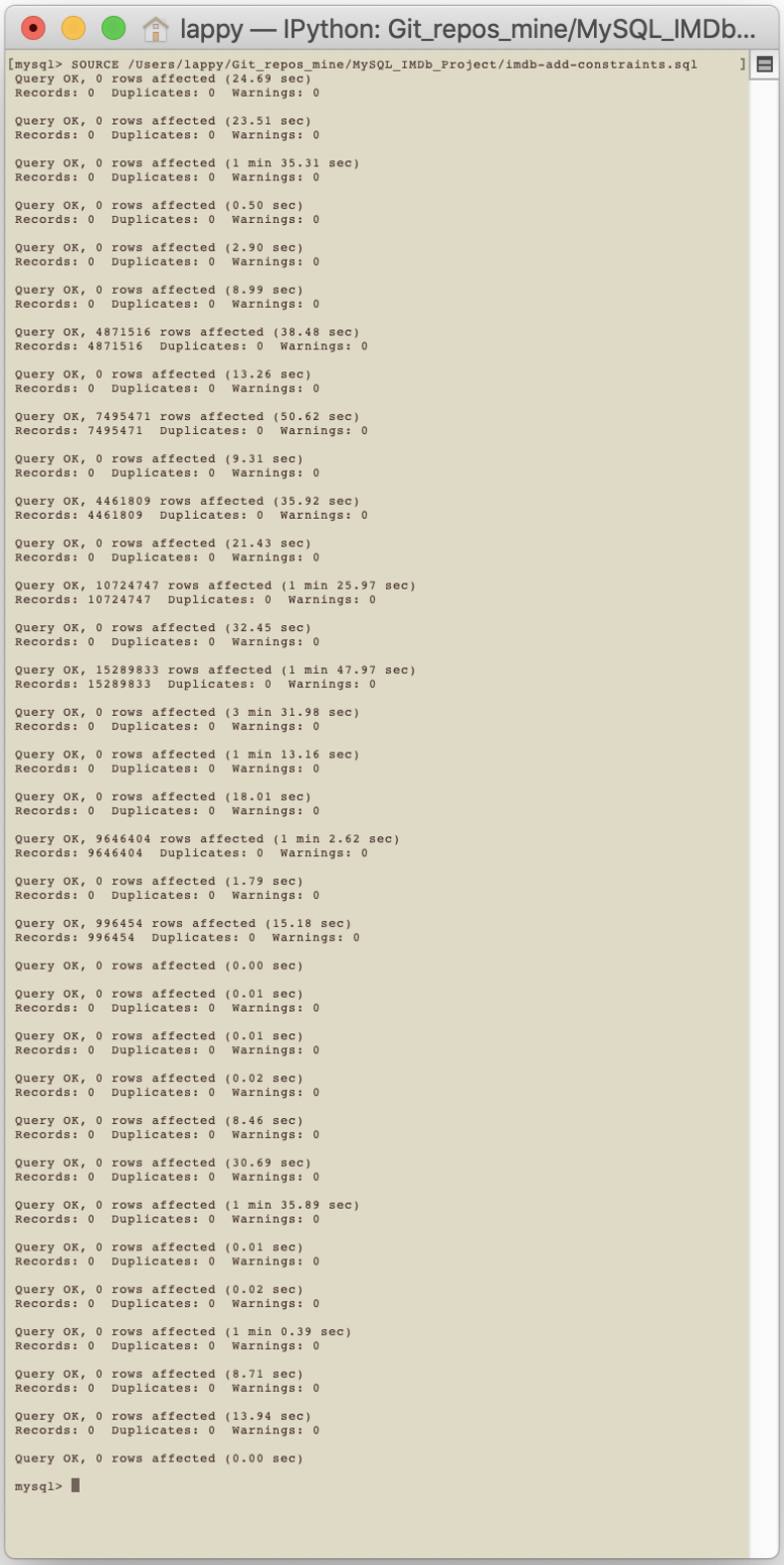


Figure 6: Terminal screenshot of running imdb-add-constraints.sql in MySQL

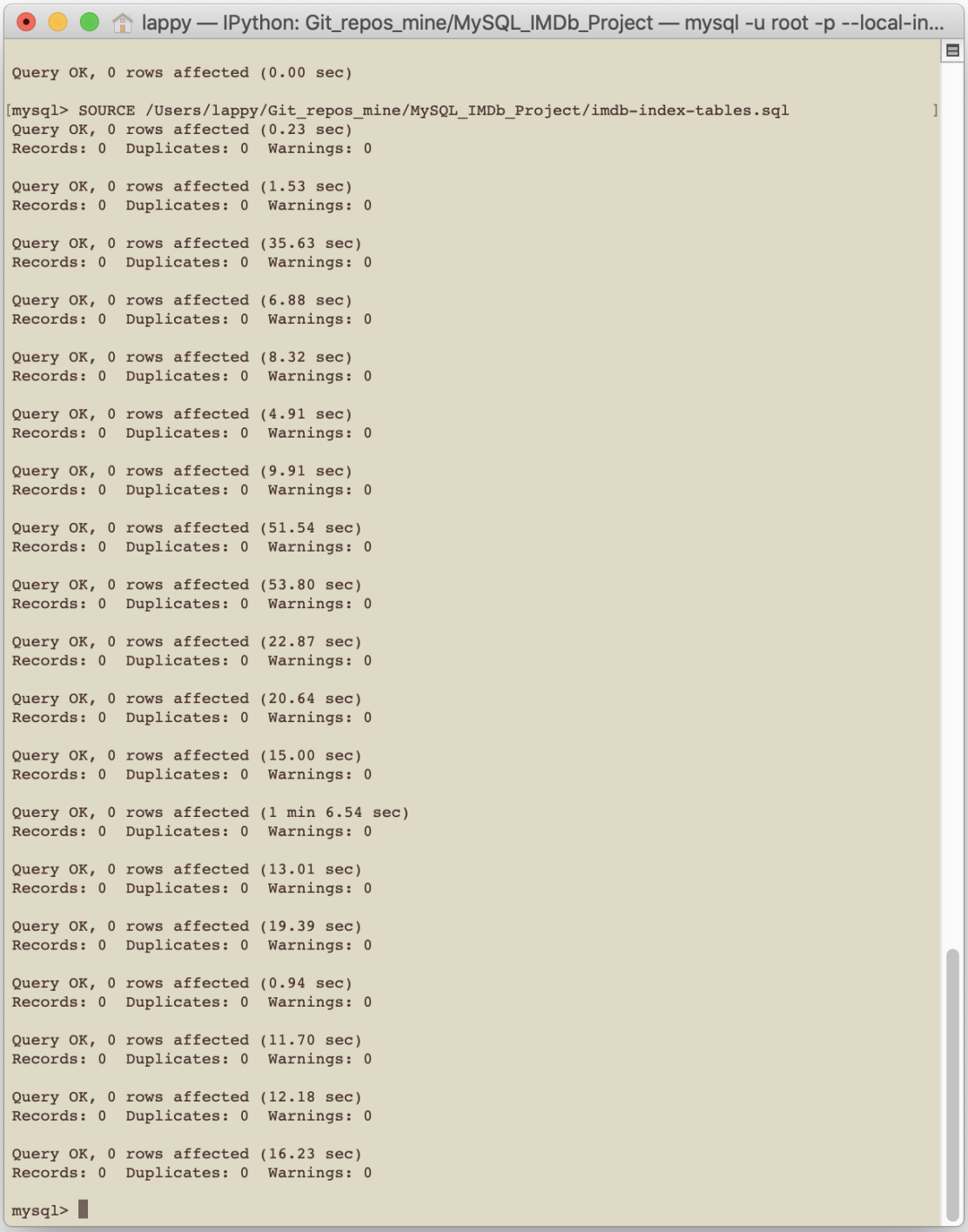


Figure 7: Terminal screenshot of running imdb-index-tables.sql in MySQL

The database is quite large, so for illustration purposes we will quite often limit ourselves to the first few entries only.

Example queries

We will consider a few queries for illustration purposes.

- Query 9: Who are the actors who played James Bond in a movie? How many times did they play the role of James Bond?

```
CREATE OR REPLACE VIEW Q9(name_id,name_,number_of_films)
AS SELECT N.name_id, N.name_, COUNT(*) AS number_of_films
FROM Names_ AS N, Had_role AS H, Titles AS T
WHERE H.role_ LIKE 'James Bond'
AND T.title_type LIKE 'movie'
AND T.title_id = H.title_id
AND N.name_id = H.name_id
GROUP BY N.name_id;
```

To see the results of this query:

```
SELECT * FROM Q9;
```

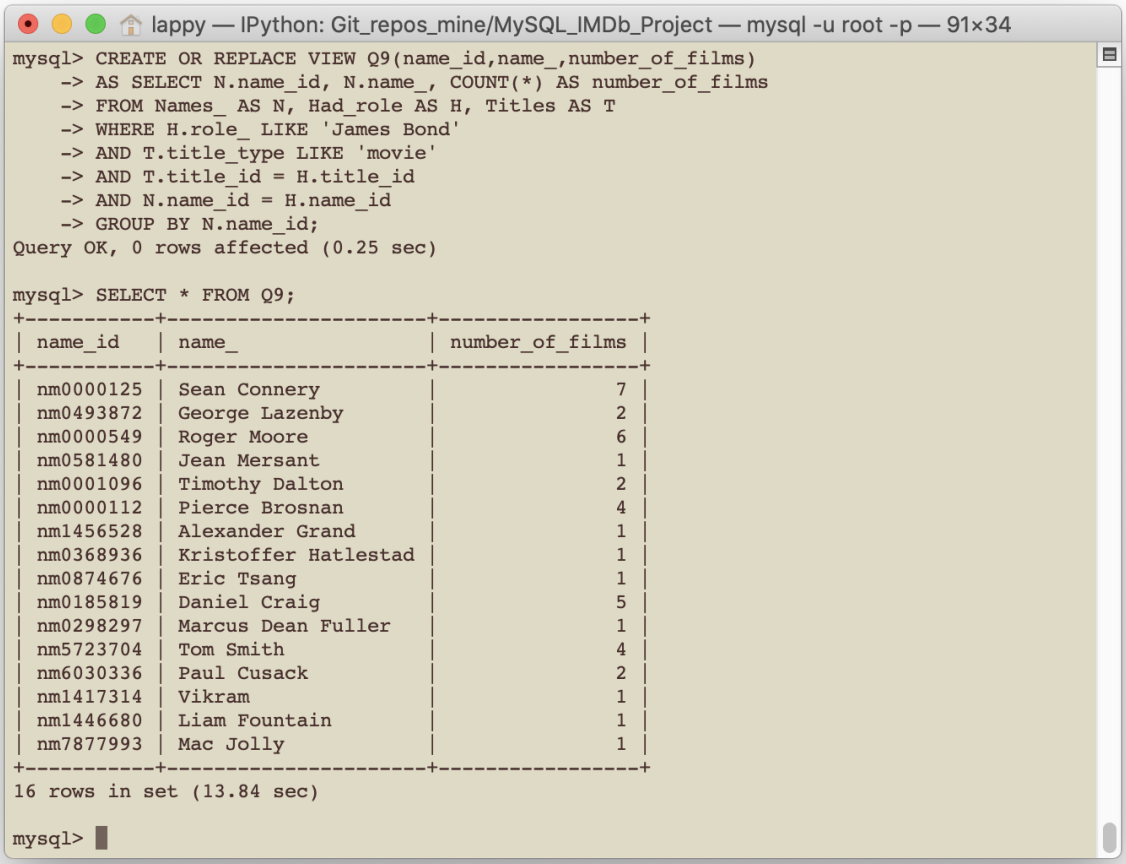


Figure 8: MySQL Query 9.

- Query 10: How many actors played James Bond?

```
CREATE OR REPLACE VIEW Q10(number_of_JB_actors)
AS SELECT COUNT(DISTINCT name_id) AS number_of_JB_actors
FROM Q9;
```

To see the results of this query:

```
SELECT * FROM Q10;
```

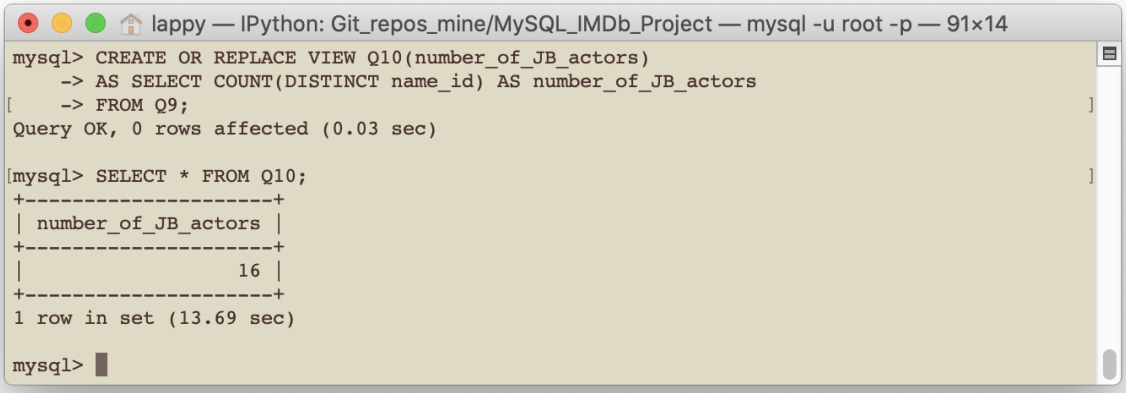


Figure 9: MySQL Query 10.

- Query 11: I don't recognise some of these names lets look at them more closely

```
CREATE OR REPLACE VIEW Q11(name_,title_id,primary_title,start_year)
AS SELECT Q9.name_, T.title_id, T.primary_title, T.start_year
FROM Q9, Titles AS T, Had_role AS H
WHERE Q9.name_id = H.name_id
AND H.role_ LIKE 'James Bond'
AND T.title_id = H.title_id
AND T.title_type LIKE 'movie'
ORDER BY T.start_year DESC;
```

To see the results of this query:

```
SELECT * FROM Q11;
```

Clearly, a few of these movies contain the character James Bond, but are not the James Bond movies we have in mind. In particular, the appearance of the movie `Deadly Hands of Kung Fu` is quite interesting as it looks to be a 1970's kung fu flick. Its IMDb page can be found here. From this page we quote its synopsis:

> "It's one of the"Bruceploitation" films that were made to cash in on Bruce Lee after his death. The story follows Bruce Lee after he dies and ends up in Hell. Once there, he does the logical thing and opens a gym. After fending off the advances of the King Of Hell's naked wives, he discovers that the most evil people in Hell are attempting a takeover, so Bruce sets out to stop it. As if it wasn't weird enough, the evil people are: Zatoichi (the blind swordsman hero of Japanese film), **James Bond**, The Godfather, The Exorcist, Emmanuelle (the "heroine" of many European softcore porn films), Dracula, and, of course, Clint Eastwood (played by a Chinese guy). Aiding Bruce is The One-Armed Swordsman (hero of kung-fu films), Kain from the U.S. tv series, Kung-Fu (actually played by a Chinese guy this time), and Popeye the Sailor Man! Yes, Popeye the Sailor Man. He eats spinach and helps Bruce fight some mummies."

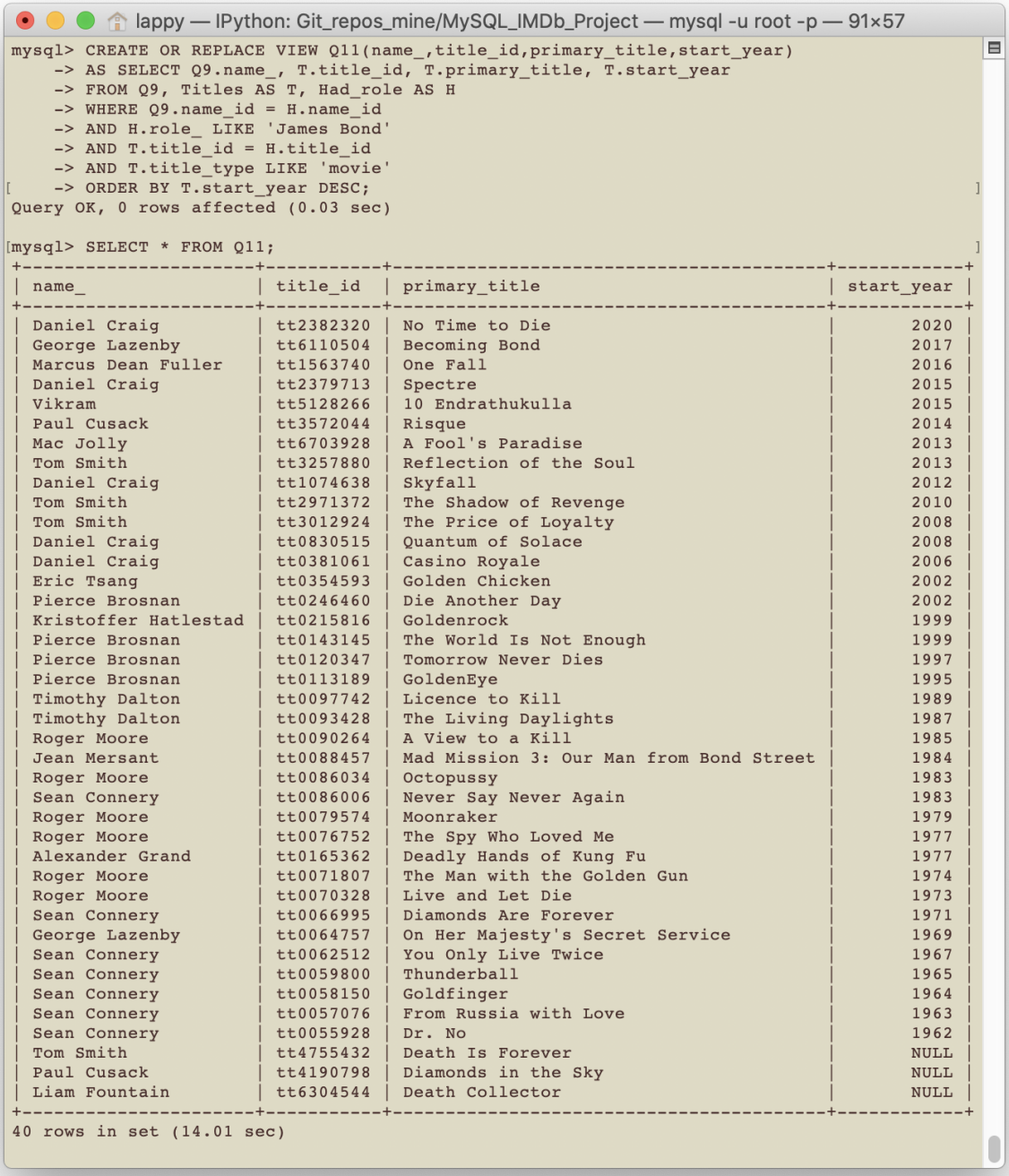
WOW!!! I certainly was not expecting that, I need to see this movie! In the script `imdb_scraper.py` we provide a couple of functions that can be used to extract the url of a movie poster on its IMDb webpage using its `title_id`.

The functions used to scrape the movie poster url made use of BeautifulSoup and urllib.request.

SQL Queries using python and data visualisation

In the notebook `MySQL_IMDb_visualisation.ipynb` we query the IMDb database to explore and visualise the IMDb dataset using pandas and matplotlib. This notebook is by no means a thorough exploration of the IMDb dataset. Its purpose is to practice querying a database using python, then to process and visualise the retrieved data with the pandas package. In particular, we consider the following questions:

- What are the average ratings for the TV show 'The X-files'?
- What genres are there?
- How many movies are there in each genre?
- How many movies are made in each genre each year?
- How do the average ages of leading actors and actresses compare in each genre?
- What is a typical runtime for movies in each genre?



```
lappy — IPython: Git_repos_mine/MySQL_IMDb_Project — mysql -u root -p — 91×57

mysql> CREATE OR REPLACE VIEW Q11(name_,title_id,primary_title,start_year)
    -> AS SELECT Q9.name_, T.title_id, T.primary_title, T.start_year
    -> FROM Q9, Titles AS T, Had_role AS H
    -> WHERE Q9.name_id = H.name_id
    -> AND H.role_ LIKE 'James Bond'
    -> AND T.title_id = H.title_id
    -> AND T.title_type LIKE 'movie'
    -> ORDER BY T.start_year DESC;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM Q11;
+---------------------+-----------+----------------------------------------+------------+
| name_               | title_id  | primary_title                          | start_year |
+---------------------+-----------+----------------------------------------+------------+
| Daniel Craig        | tt2382320 | No Time to Die                         |       2020 |
| George Lazenby      | tt6110504 | Becoming Bond                          |       2017 |
| Marcus Dean Fuller  | tt1563740 | One Fall                               |       2016 |
| Daniel Craig        | tt2379713 | Spectre                                |       2015 |
| Vikram              | tt5128266 | 10 Endrathukulla                       |       2015 |
| Paul Cusack         | tt3572044 | Risque                                 |       2014 |
| Mac Jolly           | tt6703928 | A Fool's Paradise                      |       2013 |
| Tom Smith           | tt3257880 | Reflection of the Soul                 |       2013 |
| Daniel Craig        | tt1074638 | Skyfall                                |       2012 |
| Tom Smith           | tt2971372 | The Shadow of Revenge                  |       2010 |
| Tom Smith           | tt3012924 | The Price of Loyalty                   |       2008 |
| Daniel Craig        | tt0830515 | Quantum of Solace                      |       2008 |
| Daniel Craig        | tt0381061 | Casino Royale                          |       2006 |
| Eric Tsang          | tt0354593 | Golden Chicken                         |       2002 |
| Pierce Brosnan      | tt0246460 | Die Another Day                        |       2002 |
| Kristoffer Hatlestad | tt0215816 | Goldenrock                             |       1999 |
| Pierce Brosnan      | tt0143145 | The World Is Not Enough                |       1999 |
| Pierce Brosnan      | tt0120347 | Tomorrow Never Dies                    |       1997 |
| Pierce Brosnan      | tt0113189 | GoldenEye                              |       1995 |
| Timothy Dalton      | tt0097742 | Licence to Kill                        |       1989 |
| Timothy Dalton      | tt0093428 | The Living Daylights                   |       1987 |
| Roger Moore         | tt0090264 | A View to a Kill                       |       1985 |
| Jean Mersant        | tt0088457 | Mad Mission 3: Our Man from Bond Street |       1984 |
| Roger Moore         | tt0086034 | Octopussy                              |       1983 |
| Sean Connery        | tt0086006 | Never Say Never Again                  |       1983 |
| Roger Moore         | tt0079574 | Moonraker                              |       1979 |
| Roger Moore         | tt0076752 | The Spy Who Loved Me                   |       1977 |
| Alexander Grand     | tt0165362 | Deadly Hands of Kung Fu                |       1977 |
| Roger Moore         | tt0071807 | The Man with the Golden Gun            |       1974 |
| Roger Moore         | tt0070328 | Live and Let Die                       |       1973 |
| Sean Connery        | tt0066995 | Diamonds Are Forever                   |       1971 |
| George Lazenby      | tt0064757 | On Her Majesty's Secret Service        |       1969 |
| Sean Connery        | tt0062512 | You Only Live Twice                    |       1967 |
| Sean Connery        | tt0059800 | Thunderball                            |       1965 |
| Sean Connery        | tt0058150 | Goldfinger                             |       1964 |
| Sean Connery        | tt0057076 | From Russia with Love                  |       1963 |
| Sean Connery        | tt0055928 | Dr. No                                 |       1962 |
| Tom Smith           | tt4755432 | Death Is Forever                       |       NULL |
| Paul Cusack         | tt4190798 | Diamonds in the Sky                    |       NULL |
| Liam Fountain       | tt6304544 | Death Collector                        |       NULL |
+---------------------+-----------+----------------------------------------+------------+
40 rows in set (14.01 sec)
```

Figure 10: MySQL Query 11.

```
MySQL_IMDb_Project — IPython: Git_repos_mine/MySQL_IMDb_Project — ipython — 131×17
(base) lappy MySQL_IMDb_Project $ ipython
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.8.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: run imdb_scraper.py

In [2]: title_id = 'tt0165362'

In [3]: bs = get_title_webpage(title_id)

In [4]: poster_link = get_imdb_title_poster_url(bs)

In [5]: print(poster_link)
https://m.media-amazon.com/images/M/MV5BMzk0NTE1MDA1NF5BMl5BanBnXkFtZTgwOTIxMDAxNDE@._V1_UY268_CR8,0,182,268_AL_.jpg

In [6]:
```

Figure 11: Poster url for Deadly Hands of Kung Fu.

Figure 12: Poster for Deadly Hands of Kung Fu.

This section is also an ongoing piece of work, which will be added to in the future.

As a sample of the kind of querying and visualisation performed in this notebook we show a few of the created figures:

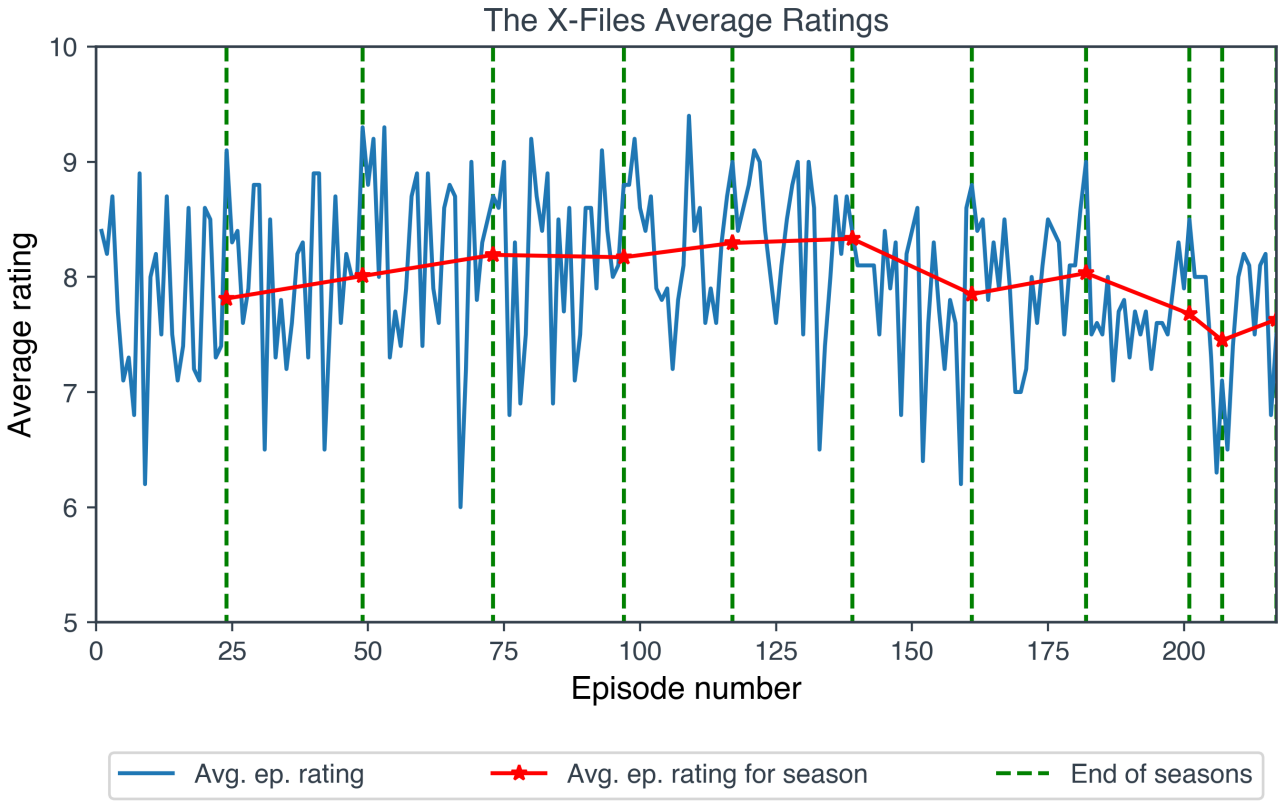


Figure 13: Average ratings of episodes of the X-Files.

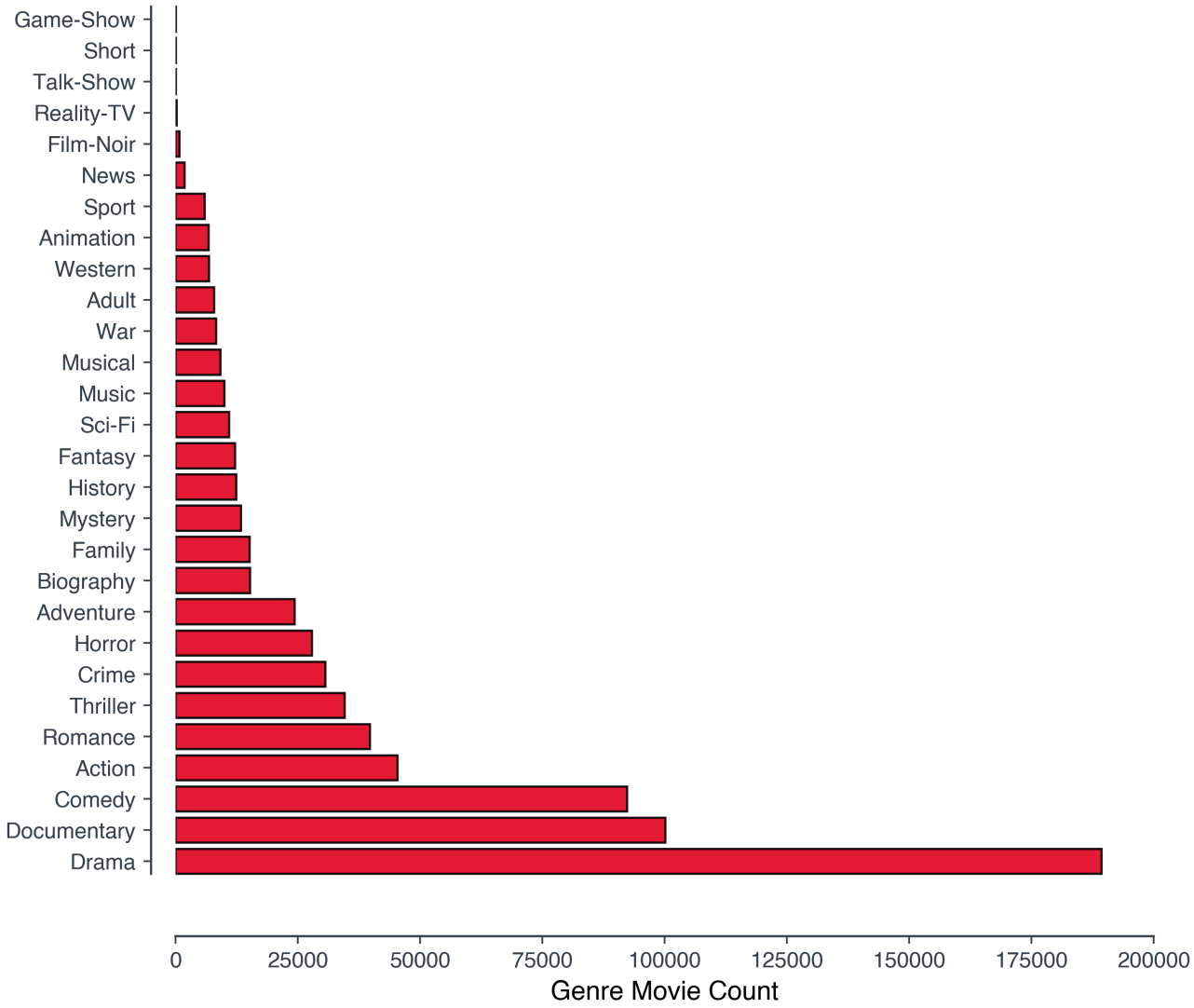


Figure 14: Number of movies in the database in each genre.

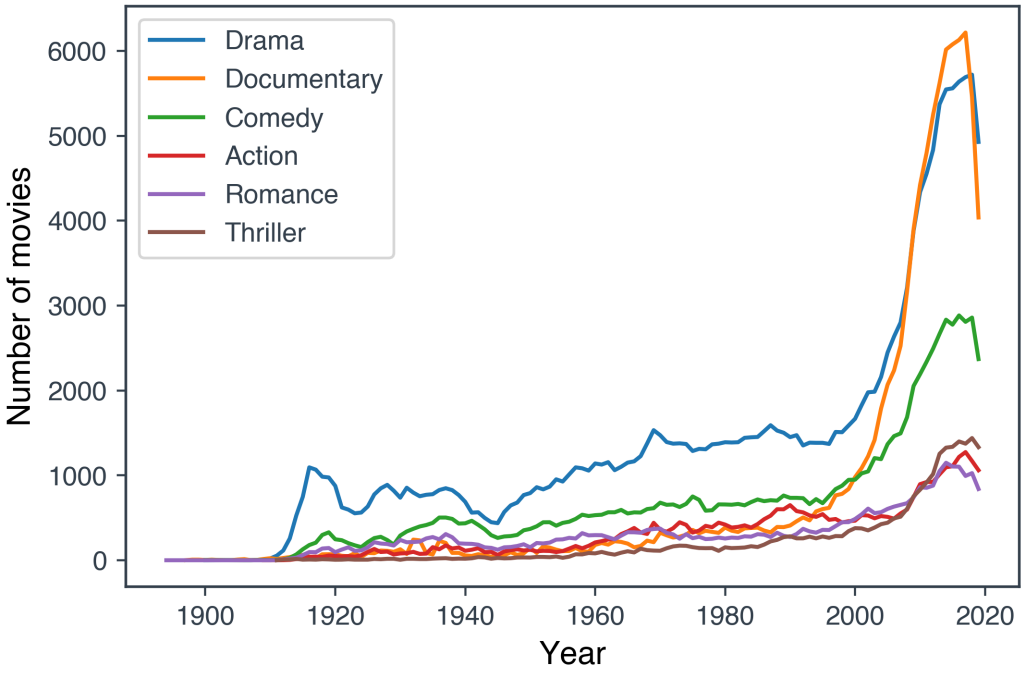


Figure 15: Number of movies per year in each genre.

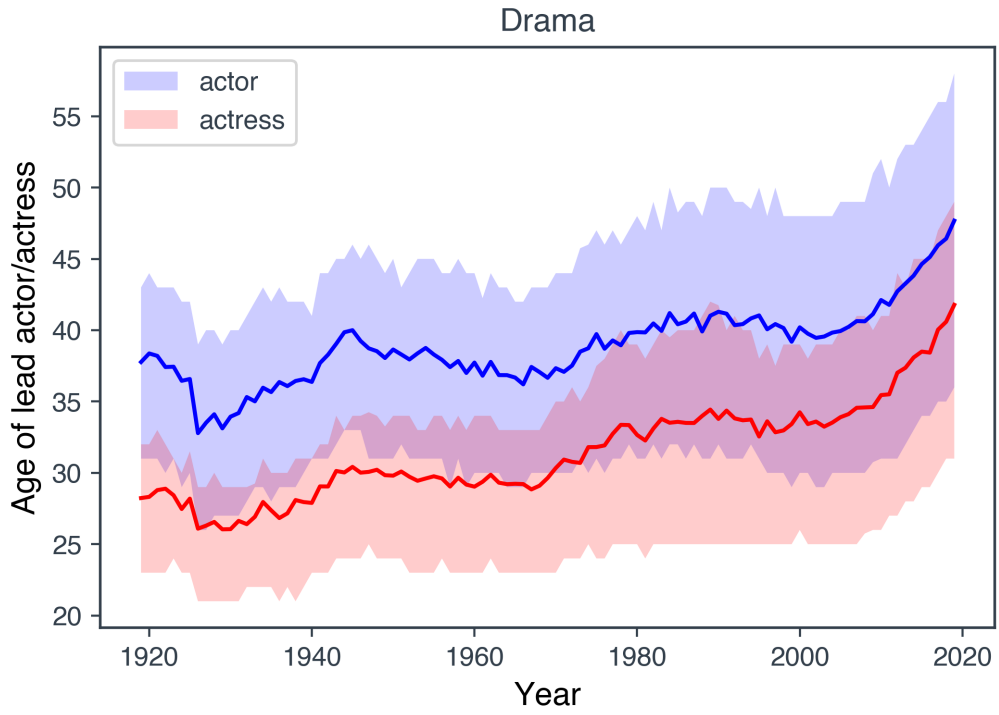


Figure 16: Average ages of leading actors and actresses in drama movies.

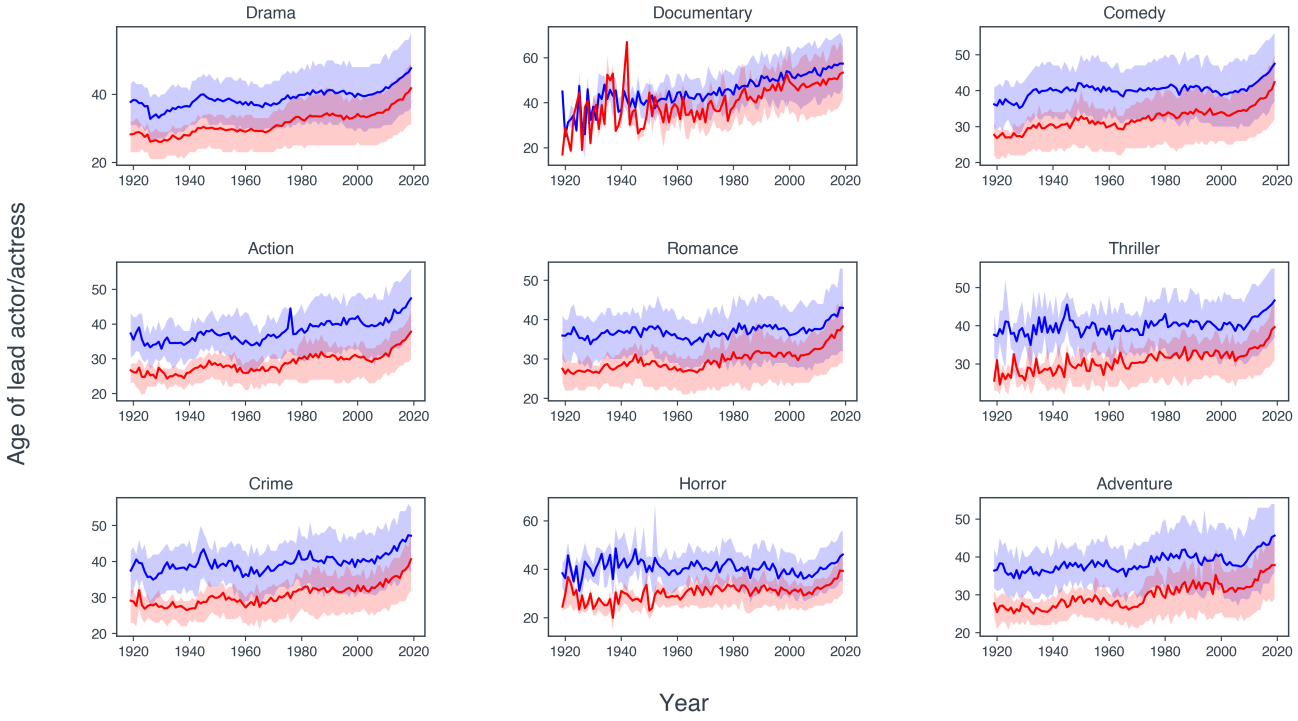


Figure 17: Average ages of leading actors and actresses in the top 9 movie genres.

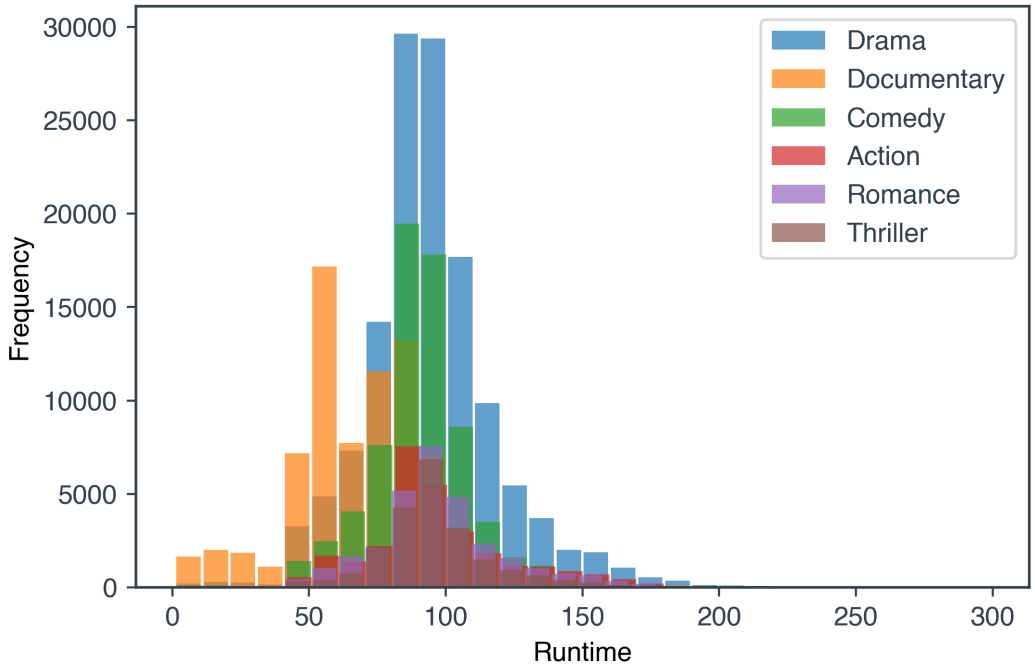


Figure 18: Histogram of movie runtimes for the top 6 movie genres.