**Pre installation requirements:**

* Python
* Mysql ( I used mysql workbench for its relative parts of the project)
* Mysql connector : use command ‘pip install mysql-connector-python’
* Files from this repo

**Installation instructions:**

dump

Use the folder ‘netflixdb\_MySQLDump’ containing the dump of the netflixdb (Netflix database) to set up the database.

**or**

Manual setup

If the dump fails for some reason there is a folder containing the tables called ‘tablesCsvForm’ which has all the data in an excel format.

To create the tables look inside the ‘NetflixSearchEngine.py’ file and at the top is commented out code that will create the tables for a database named ‘netflixdb’. So make sure you’ve created the database on mysql workbench or however named ‘netflixdb’ before continuing.

1. Now that you have the database ready for tables insert information to get your connector working, by default I have host as ‘localhost’, and user & pass as ‘root’. Then uncomment the first create table for table shows.
2. In mysql workbench run this sql script to insert the data from its respective csv file from the ‘tablesCsvForm’ folder. For show we would use the ‘showTableWithstudioid.csv’ file

The script:

LOAD DATA LOCAL INFILE 'C:\\path\\to\\file\\tablesCsvForm\\studioTable.csv' INTO TABLE Studios

FIELDS TERMINATED BY ',' ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 LINES;

1. (comment old create table) Uncomment the next create table and run the python code and use the same script from above on the ‘directorsTable’ file to fill it.

1. The next line of code alters the shows table and adds a foreign key to the directors table. uncomment this one and the db.commit().

4.5) if the command fails foreign key checks may play a weird part into that I wasn’t able to figure out though turning it off fixed it

Use this script in mysql workbench before retrying the python code to add foreign key:

‘SET foreign\_key\_checks = 0’

1. The next two tables are the casts and actors tables. Both of these need to be created before foreign keys can be added from shows as the casts table is the thing linking the actors table to the shows table due to a many to many relationship. Create those two tables and fill them with the mysql script and csv file accordingly
2. Comment out the create tables for the two and uncomment the alter tables and db.commit() And run the code again
3. Lastly uncomment the studio table, fill it with data from studio csv using script, comment it back out and uncomment the alter to add foreign key and run the py code again.

**Running the program:**

Now that you got the database all set up you can use the run the Netflix search engine python file and it will use the user input to modify the cursors queries to give results based on what you may want.

You can find your show based off 13 different search types:

Show title, genre, actor name, director name, studio name, show type (movie or tv series), rating, release year, and the last 5 are some combination of the previous (for example actor and director).

Once you make a selection the user will be prompted to give additional information on their selection such as if they search by genre they can get a list of genres to search by and choose one then the program will filter its results by the genre they’ve chose (we use the LIKE keyword instead of ‘=’ to give the users a bit more leeway with how they type their option) and then they’ll receive a list of movies based on that query. Most of the time the query will show the first 5 results but if they choose an option that gives a lot of results such as genre they’ll receive 10 options at a time instead and they can choose more to receive 5 or 10 more. Once they’ve reached the end of the list or found the option they want they can type the number associated with the movie (which is an indice to a list where the movie is temporarily stored) to see more details of the movie.

**Additional information:**

three required natural joins: To access director, actor, and studio data:

mycursor.execute("SELECT \* FROM Shows NATURAL JOIN directors WHERE name LIKE (%s) ", (directorName,))

mycursor.execute("SELECT \* FROM Shows NATURAL JOIN studios WHERE name LIKE (%s) ", (studioName,))

mycursor.execute("SELECT \* FROM Shows NATURAL JOIN Casts NATURAL JOIN actors WHERE name LIKE (%s) ", (actorName,))

and more in the combinations section in total 7 queries involving natural joins

(actor requires a double natural join because of the Casts table between them linking the two)

Issues and future:

When originally creating the database I used imdbs dataset which consisted of something like 30-50 million rows in total and the data was messy and needed to be cleaned so after spending more time than I was happy with to do so it still wasn’t being easy to work with. So I decided to scrap whatever work ive done and find a new dataset. This new one was based on Netflix and had around 9000 rows and was much more manageable.

The Netflix data was a little messy but nothing that couldn’t be handled. The actor data was originally multivalued attributes which I wasn’t fond of so I redid my schema so that they weren’t required and wrote two scripts to manipulate that data into two tables so that it would properly work with the database with single valued attributes.

If I was going to properly turn this project into a real Netflix search engine, I would give it a proper UI with images associated with the movies/series and a link to a stream where the user would actually be able to watch the show chosen. With that little bit of work added to this program this could potentially be a real Netflix clone (without billing and account system that is).

Table creation:

See ‘relevant stuff.txt’ for more information on the process I used in creating the database