Quynh Tran
CS 351- HW 5 Writeup

Part 1 - Data:
You should include a description of each relation in your schema, including what attributes are primary and foreign keys. Also include an analysis on what normal form this schema is in. You may put the database in a normal form higher than 3NF, but it may not be lower than 3NF.

Movie: contains all the attributes that are in the First Normalization Form
- Attributes: budget, homepage, m_id, original language, original title, overview, popularity, release date, revenue, runtime, status, tagline, title
- Primary key: m_id
- Foreign key: None

genres: contains unique genre ids with corresponding names
- Attribute: id, name
- Primary key: id
- Foreign key: None

genres_and_movie: a join table that contains the relationship between Movie and genres tables
- Attribute: m_id (reference to Movie), g_id (reference to genres)
- Primary key: (m_id, g_id)
- Foreign key: m_id, g_id

production_companies: contains unique name of companies and corresponding ids
- Attribute: name, id
- Primary key: id
- Foreign key: None

company_and_movie: a join table that contains the relationship between Movie and production_companies relations
- Attribute: m_id (reference to Movie), c_id (reference to production_companies)
- Primary key: (m_id, c_id)
- Foreign key: m_id, c_id

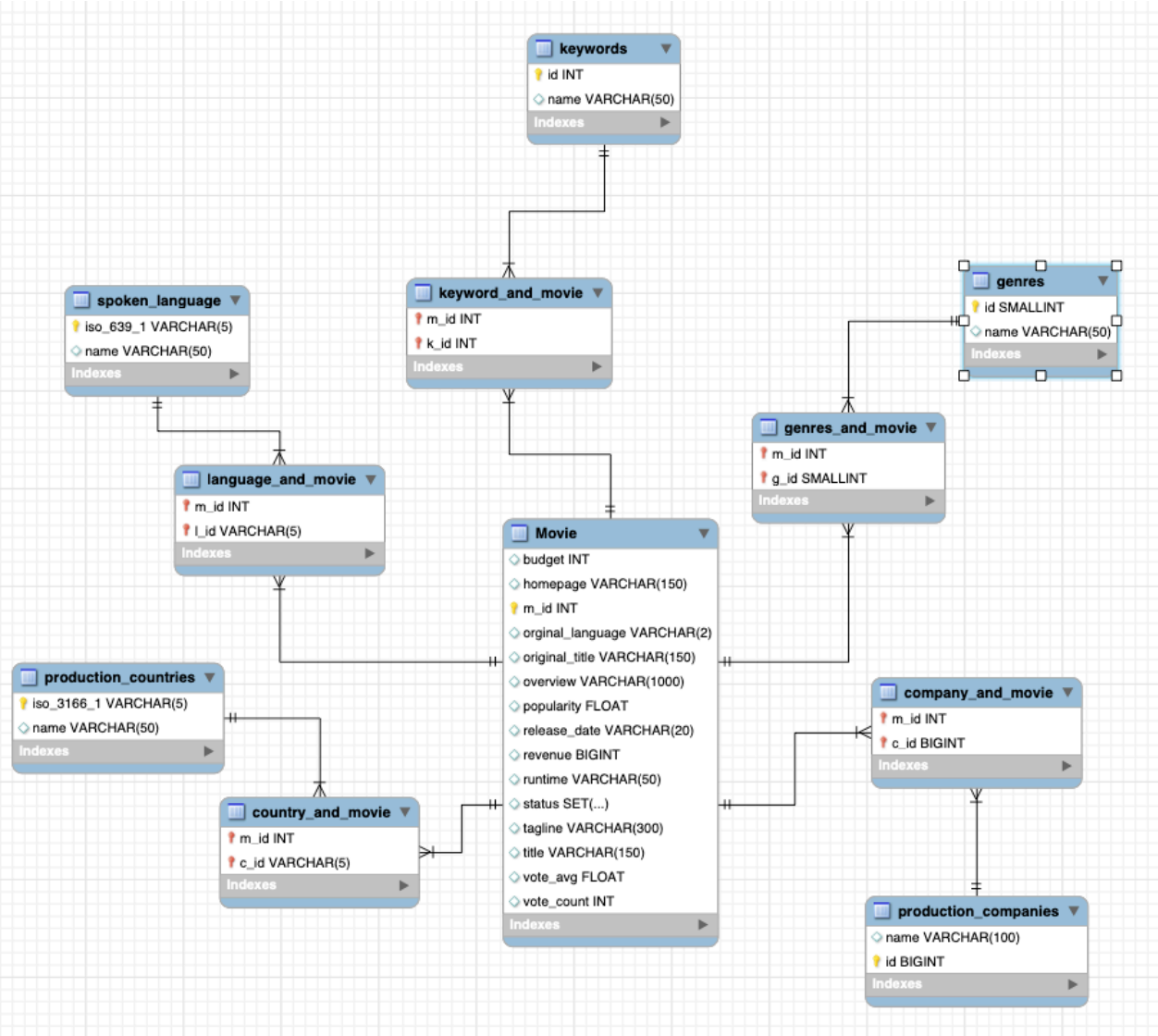
production_countries: contains all unique country ids and corresponding country names
- Attribute: iso_3166_1, name
- Primary key: iso_3166_1
- Foreign key: None

country_and_movie: a join table that contains the relationship between Movie and production_countries
- Attribute: m_id (reference to Movie), c_id (reference to production_countries)

- Primary key: (m_id, c_id)
- Foreign key: m_id, c_id

spoken_language: contains all unique language ids and corresponding language names
- Attribute: iso_639_1, name
- Primary key: iso_639_1
- Foreign key: None

language_and_movie: a join table that contains the relationship between Movie and spoken_language
- Attribute: m_id (reference to Movie), l_id (reference to spoken_language)
- Primary key: (m_id, l_id)
- Foreign key: m_id, l_id



All tables above are in 3NF because all cells hold single values, there are no partial dependencies between non-prime attributes and candidate keys, and each table has a primary key that is also the super key.

Part 2 - Queries:
For the following questions, write SQL queries in Python that answer them. Your code must include a line for each query so that it can be run individually if need be, by either commenting out the lines or providing some command line options. The commands must output the relation in a comma delimited form.

1. 10% What is the average budget of all movies? Your output should include just the average budget value.

```
SELECT AVG(budget) FROM Movie;
```

```
+---------------+
| AVG(budget)   |
+---------------+
| 29045039.8753 |
+---------------+
```

2. 10% Show the movies that were produced in the United States. Your output must include the movie title and the production company name.

```
SELECT Movie.original_title, production_companies.name
FROM Movie
INNER JOIN company_and_movie
ON Movie.m_id=company_and_movie.m_id
INNER JOIN production_companies
ON company_and_movie.c_id = production_companies.id
INNER JOIN country_and_movie
ON Movie.m_id = country_and_movie.m_id
INNER JOIN production_countries
ON country_and_movie.c_id = production_countries.iso_3166_1
WHERE iso_3166_1='US' limit 5;
```

```
+----------------+-----------------------------------------+
| original_title | name                                    |
+----------------+-----------------------------------------+
| Four Rooms     | Miramax Films                           |
| Four Rooms     | A Band Apart                            |
| Star Wars      | Lucasfilm                               |
| Star Wars      | Twentieth Century Fox Film Corporation  |
| Finding Nemo   | Pixar Animation Studios                 |
+--------+-----------------------------------------+
```

3. 10% Show the top 5 movies that made the most revenue. Your output must include the movie title and how much revenue it brought in.

```
SELECT original_title, revenue FROM Movie ORDER BY revenue DESC
limit 5;
```

```
+----------------+------------+
| original_title | revenue    |
+----------------+------------+
| Avatar         | 2787965087 |
| Titanic        | 1845034188 |
| The Avengers   | 1519557910 |
| Jurassic World | 1513528810 |
| Furious 7      | 1506249360 |
+----------------+------------+
```

4. 10% What movies have both the genre Science Fiction and Mystery. Your output must include the movie title and all genres associated with that movie.

```
SELECT Movie.original_title, GROUP_CONCAT(genres.name)
FROM Movie
JOIN genres_and_movie
ON Movie.m_id = genres_and_movie.m_id
JOIN genres
ON genres_and_movie.g_id = genres.id
WHERE Movie.m_id IN (SELECT Movie.m_id
FROM Movie
INNER JOIN genres_and_movie
ON Movie.m_id = genres_and_movie.m_id
INNER JOIN genres ON genres_and_movie.g_id = genres.id
```

```
WHERE genres.name = 'Science Fiction' or genres.name = 'Mystery'
GROUP BY(Movie.original_title) HAVING COUNT(*) =2)
GROUP BY(Movie.original_title) LIMIT 5;
```

```
+-----------------------------------------+--------------------------------------+
| original_title                          | GROUP_CONCAT(genres.name)            |
+-----------------------------------------+--------------------------------------+
| 2001: A Space Odyssey                   | Adventure,Science Fiction,Mystery    |
| Atlas Shrugged Part II                  | Drama,Science Fiction,Mystery        |
| Atlas Shrugged Part III: Who is John Galt? | Drama,Science Fiction,Mystery     |
| Beneath the Planet of the Apes          | Adventure,Science Fiction,Mystery    |
| Blindness                               | Drama,Thriller,Science Fiction,Mystery |
+-----------------------------------------+--------------------------------------+
```

5. 10% Find the movies that have a popularity greater than the average popularity. Your output must include the movie title and their popularity.

```
SELECT original_title, popularity
FROM Movie
WHERE popularity > (SELECT avg(popularity) FROM Movie) LIMIT 5;
```

```
+-----------------+------------+
| original_title  | popularity |
+-----------------+------------+
| Four Rooms      |    22.8762 |
| Star Wars       |    126.394 |
| Finding Nemo    |    85.6888 |
| Forrest Gump    |    138.133 |
| American Beauty |    80.8786 |
+-----------------+------------+
```