CS 351 HW 5

Tam Nguyen

March 2019

Part One:

The relationship table *genresRelation*, *keywordsRelation*, and *productionCoRelation* have identical schema (which we'll refer to as id tables) with each other with addition to *productionCountryRelation* and *languagesRelation* have matching schema (we'll refer this to *abbrev* tables). The id tables primary key (PK) are a combination of both the foreign key (FK), *movieId* (reference to 'movies' table's PK) and *itemID* (reference to its correspondent table's PK). As for the *abbrev* tables, the PK are the *movieId* and *abbrev* (reference to *productionCountry*/languages' PK). The reason why I chose to stick with only two schema layouts is because it'll be easier to work with the SQL code via Python. Additionally, The Tables are all in BCNF.

Part Two:

For the first query, all it needed was the average aggregation since we're looking for just the average budget of all the movies. The Second query was trickier since we needed to combine four tables. So, to achieve this I inner join movies to *productioncountryrelation* on the IDs (*movies.id = productionCountryRelation.movieId*). Then I inner join that new table to *productionCoRelation* on movies id (*movies.id = productionCoRelation.movieId*), and inner join once more to *productionCo* on *productionCo's* ID (*productionCo.id = productionCoRelation.itemID*). Finally, we filter the tuples to match our 'US' requirement by using a restriction where *productionCoRelation's* abbrev attribute matches 'US'.

On the Third query, it was a basic 'get' column but requirements need it to display 5 and in decending order, so we add ' ORDER BY revenue DESC LIMIT 5' to achieve the goal. Since the fourth query wants movie with two genres it'll require subqueries since each query can only pull each movies from a genre. The reason why this is hard is not only do we have to display the movie titles but also *all* the genres for that movie. So, to get our query *SELECT* from subqueries where subquery 1 pulls movies with Mystery, and subquery 2 pulls movies with Science Fiction and *INNER JOIN* those two tables on movies titles (since we only want movies that contains both). From there, we *INNER JOIN* the new table with the *genresrelation* table on the movies' id so that we can get all the genres for a movie. And finally, *INNER JOIN* again with the *genres* table to get the genres. For the last query, I had to do a sub query in a query. So, it's the basic select column from table with our restrictions being popularity attribute is greater than average, but instead of doing "< avg(popularity)", we sub query the average. So, we're left with " popularity > (SELECT avg(popularity) FROM movies)".

Average Budget:

```
~29045039.88
```

US Movies:

```
('Four Rooms', 'Miramax Films')
('Star Wars', 'Lucasfilm')
('Finding Nemo', 'Pixar Animation Studios')
('Forrest Gump', 'Paramount Pictures')
('American Beauty', 'DreamWorks SKG')
```

Top 5 Earnings:

```
('Avatar', 2787965087)
('Titanic', 1845034188)
('The Avengers', 1519557910)
('Jurassic World', 1513528810)
('Furious 7', 1506249360)
```

Science Fiction & Mystery:

```
('2001: A Space Odyssey', 'Adventure')
('2001: A Space Odyssey', 'Science Fiction')
('2001: A Space Odyssey', 'Mystery')
('Star Trek: The Motion Picture', 'Adventure')
('Star Trek: The Motion Picture', 'Science Fiction')
```

Popular:

```
('Four Rooms', 22.87623)
('Star Wars', 126.393695)
('Finding Nemo', 85.688789)
('Forrest Gump', 138.133331)
('American Beauty', 80.878605)
```