CS 377: Database Systems

Project #2 Solutions

SUBMISSION: Please submit the project electronically via Canvas before 11:59 pm. Any submissions afterwards will be marked late. You should use one file for each portion of the problem and follow these guidelines for your project submission:

- If you have comments in your SQL query, use C style comments using /∗ < comment > ∗/
- Any text that is not within the comment brackets will be assumed to be part of your SQL command, and if the file does not execute properly, points will be deducted.
- Each file should be named as "<netID>-project2-<problem>-<part>.sql".
- A README.txt file **must** be submitted that contains the honor code, otherwise points will be deducted.

```
/* THIS CODE IS MY OWN WORK.
IT WAS WRITTEN WITHOUT CONSULTING CODE WRITTEN BY OTHER STUDENTS.
_Your_Name_Here_ */
```

IMDB Database (100 points): We will answer more sophisticated questions from our course's version of the IMDB database. Please see Project #1 and Piazza for the database details. Write SQL queries to answer the following questions. Each file should be named as "<netID>-project2-2-<part>.sql". For example, the instructor's answer to part b of this question would have the file "jho31-project2-2-b.sql".

1. (10 points) List the movies and the year where Matt Damon, George Clooney, and Brad Pitt have all played a role.
2. (10 points) List the movie name and year for the comedy movies that Tom Hanks has either directed or starred in?
3. (10 points) Which actor(s) has only been in Steven Soderbergh's movies?
4. (10 points) What movies has no female actors in 2010?
5. (10 points) For each year, count the number of movies in that year that had only female actors (and at least one actor).
6. (10 points) What actor(s) has been in all of Stephen Chow's comedic movies before 2009?
7. (5 points) List the director, movie name, and year for the biographical movies that were directed in a leap year (assume every year divisible by 4 is a leap year for this purpose).
8. (10 points) Find the actors who have played five or more unique roles in the same movie in 2007. List the actor names, movie names, and number of distinct roles in that movie ordered by movie name in ascending order.
9. (10 points) Find the actress(es) that has been in the largest number of family movies between 2005 and 2010 (inclusive).

10. (15 points) Find the actor(s) that are exactly 2 degrees of separation from Kevin Bacon. For example, Ian McKellen has never been in a movie with Kevin Bacon. But Ian McKellen was in X-Men: Days of Future Past with James McAvoy and Michael Fassbender who were in X-Men: First Class with Kevin Bacon. Note that this example may not be in our version of the database.

(**ANSWER**)

1. Use set intersection of the movie id between all three actors.

```
SELECT name, year
FROM movie
WHERE id IN
    ( SELECT mid
      FROM casts
      WHERE aid = (SELECT id FROM actor WHERE fname = 'Matt' AND lname = 'Damon') )
  AND id in
    ( SELECT mid
      FROM casts
      WHERE aid = (SELECT id FROM actor WHERE fname = 'George' AND lname = 'Clooney') )
  AND id in
    ( SELECT mid
      FROM casts
      WHERE aid = (SELECT id FROM actor WHERE fname = 'Brad' AND lname = 'Pitt') );
```

2. This is an OR query – note this can be done with UNION as well

```
SELECT distinct name, year
FROM movie
WHERE movie.id IN ( SELECT casts.mid
                    FROM actor, casts, genre
                    WHERE fname = 'Tom' AND lname = 'Hanks'
                    AND actor.id = casts.aid
                    AND genre.mid = casts.mid
                    AND genre = 'Comedy' )
OR movie.id in ( SELECT movie_director.mid
                 FROM director, movie_director, genre
                 WHERE fname = 'Tom' AND lname = 'Hanks'
                 AND director.id = movie_director.did
                 AND genre.mid = movie_director.mid
                 AND genre = 'Comedy') ;
```

3. One option is to use the set difference, so you first eliminate all the actors who have been in movies that were not directed by Steven Soderbergh. Once you have that, you want to make sure that the actors you have left have been a movie that was directed by Steven Soderbergh as there are movies that have no directors either.

```
SELECT *
FROM actor
WHERE id NOT IN
( SELECT aid
  FROM casts c, movie_director md, director d
  WHERE md.did = d.id AND md.mid = c.mid
     AND NOT (d.fname = 'Steven' AND d.lname = 'Soderbergh'))
AND id IN
```

```
( SELECT aid
  FROM casts c, movie_director md, director d
  WHERE md.did = d.id AND md.mid = c.mid
     AND d.fname = 'Steven' AND d.lname = 'Soderbergh');
```

4. We will use a set difference, comparing against the movies that female were against all the movies made in 2010.

```
SELECT DISTINCT name, id
FROM casts, movie
WHERE movie.year = 2010
AND movie.id = casts.mid
AND casts.mid NOT IN (SELECT DISTINCT casts.mid
                      FROM actor, casts, movie
                      WHERE gender = 'F'
                      AND movie.year = 2010
                      AND actor.id = casts.aid
                      AND movie.id = casts.mid);
```

5. We can get the movies that have a male actor, and eliminate them from the movie list. Since we also need them to have at least one actor, we need to join it with the casts to make sure there is at least one non-null actor id. Finally we can group by the year and count the movies left in the list.

```
SELECT m.year, COUNT(DISTINCT m.id)
FROM movie m, casts c
WHERE m.id NOT IN
    (SELECT m.id
     FROM movie m, actor a, casts c
     WHERE m.id = c.mid AND a.id = c.aid AND a.gender = 'M')
    AND m.id = c.mid
GROUP BY m.year;
```

6. We will use the division query formulation technique to find these movies. First we find all the movies that were made in 2009 and are comedy with Stephen chow in it. We then find an actor such that the set difference between the Stephen chow movies and all the movies that they are in is empty.

```
SELECT fname, lname
FROM actor
WHERE NOT EXISTS ( SELECT *
                   FROM casts c
                   WHERE c.mid IN (SELECT kb.mid
                                   FROM casts kb, movie, genre
                                   WHERE kb.aid = (SELECT id
                                                   FROM actor
                                                   WHERE fname = 'Stephen' and lname = 'Chow')
                                   AND movie.id = kb.mid
                                   AND year < 2009
                                   AND movie.id = genre.mid
                                   AND genre = 'Comedy')
                   AND c.mid NOT IN (SELECT t.mid
                                     FROM casts t
                                     WHERE t.aid = actor.id) )
AND id <> (SELECT id FROM actor WHERE fname = 'Stephen' AND lname = 'Chow');
```

Note that this takes quite a bit of time since we need to iterate through all the actors to check the where not exists. Instead, we only need to check the actors that have been in his movies. Although this means an additional join, it is a join on a smaller table resulting in a smaller subset of actors which will expedite our query. The new query that will run much faster but looks more complicated:

```
SELECT DISTINCT id, fname, lname
FROM (SELECT aid FROM casts
      WHERE mid IN (SELECT kb.mid
                    FROM casts kb, movie, genre
                    WHERE kb.aid = (SELECT id
                                    FROM actor
                                    WHERE fname = 'Stephen' AND lname = 'Chow')
                    AND movie.id = kb.mid
                    AND year < 2009
                    AND movie.id = genre.mid
                    AND genre = "Comedy")) AS scma, actor
WHERE NOT EXISTS ( SELECT *
                   FROM casts c
                   WHERE c.mid IN (SELECT kb.mid
                                   FROM casts kb, movie, genre
                                   WHERE kb.aid = (SELECT id
                                                   FROM actor
                                                   WHERE fname = 'Stephen'
                                                   AND lname = 'Chow')
                                   AND movie.id = kb.mid
                                   AND year < 2009
                                   AND movie.id = genre.mid
                                   AND genre = "Comedy")
                   AND c.mid NOT IN (SELECT t.mid
                                     FROM casts t
                                     WHERE t.aid = id) )
AND id = scma.aid
AND id <> (SELECT id FROM actor WHERE fname = 'Stephen' AND lname = 'Chow');
```

7. Find the movies with genre = Biography and the year modulo 4 is 0.

```
SELECT fname, lname, name, year
FROM director, movie, movie_director, genre
WHERE director.id = movie_director.did
AND movie.id = movie_director.mid
AND movie.id = genre.mid
AND genre = 'Biography'
AND year % 4 = 0
ORDER BY year, name;
```

8. Use GROUP BY and HAVING to count the number of distinct roles.

```
SELECT fname, lname, name, COUNT(DISTINCT role) as nrole
FROM actor, casts, movie
WHERE year = 2007
AND casts.mid = movie.id
AND actor.id = casts.aid
GROUP BY aid, mid
HAVING count(distinct role) >= 5;
```

9. This one uses the query formulation technique of only. Note that you could use LIMIT 1, and still return the right query.

```
SELECT fname, lname
FROM actor, casts
WHERE gender = 'F'
AND actor.id = casts.aid
AND casts.mid IN ( SELECT movie.id
                   FROM genre, movie
                   WHERE genre.mid = movie.id
                   AND genre = 'Family'
                   AND year <= 2010
                   AND year >= 2005)
GROUP BY aid
HAVING count(mid) >= ALL( SELECT COUNT(mid)
                          FROM actor, casts
                          WHERE gender = 'F'
                          AND actor.id = casts.aid
                          AND casts.mid IN ( SELECT movie.id
                                             FROM genre, movie
                                             WHERE genre.mid = movie.id
                                             AND genre = 'Family'
                                             AND year <= 2010
                                             AND year >= 2005)
                          GROUP BY aid);
```

10. First we introduce the idea of a temporary table, which is similar to a base table except it only persists for the connection session in MySQL. This way we can illustrate how things work before putting it into a single query. The syntax for creating a temporary table is almost the same as creating a table except we add the word temporary prior to table, such that the syntax is **CREATE TEMPORARY TABLE <tblname> ....**

    Step 1: Create a temporary table that represents the movies that Kevin Bacon has been in:

```
CREATE TEMPORARY TABLE kbm AS (SELECT mid
                               FROM actor, casts
                               WHERE fname = 'Kevin' AND lname = 'Bacon'
                               AND actor.id = casts.aid);
```

    Step 2: Create a temporary table that represents the actors that have been in the same movie with Kevin Bacon (also known as actors with 1 degree of Kevin Bacon separation):

```
CREATE TEMPORARY TABLE kbn1 AS (SELECT aid
                                FROM casts
                                WHERE mid IN (SELECT mid FROM kbm)
                                AND aid NOT IN (SELECT id
                                                FROM actor
                                                WHERE fname = 'Kevin' AND lname = 'Bacon'));
```

    Step 3: Create a temporary table that represents the movies that the actors with 1 degree of separation have been in that are not the movies Kevin Bacon has been in:

```
CREATE TEMPORARY TABLE kbn1m AS (SELECT mid
                                 FROM casts
                                 WHERE aid IN (SELECT aid from kbn1)
                                 AND mid NOT IN (SELECT mid from kbm));
```

Step 4: Get the actors in the movies with actors with a separation number of 1:

```
SELECT DISTINCT aid, fname, lname
FROM casts
WHERE mid IN (SELECT mid FROM kbn1m)
AND aid NOT IN (SELECT aid from kbn1)
AND aid <> (SELECT id FROM actor
            WHERE fname = 'Kevin' AND lname = 'Bacon');
```

Step 5: Put all the queries together into 1 single query.

```
SELECT DISTINCT aid, fname, lname
FROM casts, actor
WHERE mid IN (SELECT mid
              FROM casts
              WHERE aid IN (SELECT aid
                            FROM casts
                            WHERE mid IN (SELECT mid
                                          FROM actor, casts
                                          WHERE fname = 'Kevin'
                                          AND lname = 'Bacon'
                                          AND actor.id = casts.aid)
                            AND aid NOT IN (SELECT id
                                            FROM actor
                                            WHERE fname = 'Kevin'
                                            AND lname = 'Bacon'))
              AND mid NOT IN (SELECT mid
                              FROM actor, casts
                              WHERE fname = 'Kevin' AND lname = 'Bacon'
                              AND actor.id = casts.aid))
AND aid NOT IN (SELECT aid
                FROM casts
                WHERE mid IN (SELECT mid
                              FROM actor, casts
                              WHERE fname = 'Kevin' AND lname = 'Bacon'
                              AND actor.id = casts.aid)
                AND aid NOT IN (SELECT id
                                FROM actor
                                WHERE fname = 'Kevin' AND lname = 'Bacon'))
AND aid <> (SELECT id FROM actor
            WHERE fname = 'Kevin' AND lname = 'Bacon')
AND casts.aid = actor.id;
```