Database Systems, CSCI 4380-01
Homework # 4
Due Friday March 11, 2016 at 11:59:59 PM

Introduction.

This homework is also on SQL and furthermore allows you to use any type of SELECT statement. However, all your answers must be a single query, but include as many nested subqueries as you wish in that single query.

Beware: this is a larger database and a bad query is simply going to take too long to run. Make sure to read your queries before you set them on the server! Just for comparison, the largest table in the last homework had 3335 tuples. This time, we have tables with 343,966 and 149,374 tuples.

I am providing the data for this homework as well, but only in the PSQL dump format at http://www.cs.rpi.edu/ sibel/DBS_Past_Materials/Spring2016/. If you want to use it, you need to have PSQL on your computer. Here is the command to push the data into your own database:

```
$ psql

psql> create database imdb ;
psql> \q

$ cat imdb_data_march5_2016.dmp | psql -d imdb
```

Database server is available at: http://csci4380rpi.cloudapp.net/phppgadmin

Follow the same rules as before: be considerate of others using the server and do not start many parallel jobs. Let us know when you see the server showing down. As these are more complex queries, please run them one by one to test to reduce load on the server.

Homework Description: Internet Movie Database

This homework will also a use real data from IMDB (http://www.imdb.com): The Internet Movie database. I have selected a subset of the massive IMDB data corresponding to movies that many users have voted on. I have also added some additional movie ratings extracted from Twitter. All these tables have the prefix Twitter. Take a look at the schema at the end of the homework.

Given this database, write the following queries using SQL. All results are sorted by the id returned in the query unless explicitly specified. Check your queries carefully before running, start with simple parts and then add complexity.

We will take points off for unnecessary DISTINCT statements (DISTINCT after a GROUP BY is not necessary but does not incur any penalty). Check the schema carefully.

1. Return all the Harry Potter movies in descreasing order of year, return the id, name, year of the movie, number of votes and the rating on IMDB, number of tuples (as votes) and average rating on Twitter.

2. Find the id, name and the IMDB rating of movies that have received the lowest ratings in IMDB. Do not hardcode the lowest rating values.

3. Find the id, name and average Twitter rating of movies that have received the average lowest ratings in Twitter. Do not hardcode the lowest rating values.

4. Find actors who have played both in a movie with an IMDB rating below 2 and a movie with IMDB rating above 8 (according to our set of movies). Return their id, name and surname of actors.

5. Find id and name of movies involving a plot with words "destruction" and "Earth" (with this capitalizaton) that is not a movie in genre "Sci-Fi" according to moviegenres.

6. This is a favorite interview question type (no hardcoding of answers):

   Return the name of the second highest rated Harry Potter movie according to IMDB ratings.

7. Find the crankypants in Twitter:

   Find Twitter users who rated at least 10 movies with a rating less than 3 and never rated a movie high than a 8. Return the user id and the link to these users' Twitter pages in the form: `https://twitter.com/intent/user?user_id=<twitterid from database>`

   See for example: https://twitter.com/intent/user?user_id=16027904

8. Find the most contraversial movies:

   Find movies in which Twitter and IMDB do not agree on. Find movies where the difference between the average rating in Twitter for the movie and the IMDB rating are at least 3 points off (in either direction). Return only movies with at least 5 ratings on Twitter, amd return the id, name of the movies, IMDB rating and the average Twitter rating. Order the results by the difference in the rating.

9. For each actor in the database with at least 50 movies, return their id, name, surname, the total number of movies that they acted in, minimum, maximum and average IMDB rating of their movies, and average rating of all the tuples for their movies in Twitter.

   Start with writing two queries: one for IMDB ratings and one for Twitter, then combining them.

   Note that joining all the relations needed for this query will not only give incorrect results, but it will likely never terminate. These are some of the largest relations. Consider using an anonymous relation for computing the Twitter averages for each actor first, and using that in the query.

10. Find directors who directed at least 5 movies and acted in the more than 90% of the movies that they have directed. Return their name.

    A director and actor are considered the same if they have the same name and surname.

    Hint: This is a pretty hard question. Using left join happens to be extremely useful.

Submission Instructions.

Submit a single ASCII text file named `hw4ans.sql` that contains all your queries. Your script should be formatted in the same way as in Homework #3.

Database Schema

```
create table movies(
    id          serial primary key
    , title     text not null
    , full_name text
    , type      varchar(20)
    , ep_name   text
    , ep_num    text
    , status    text
    , years     text
    , movieyear int
    , name      varchar(150)
) ;

create table actors (
    id       serial primary key
    , atype  varchar(10)
      -- actor or actress
    , name   varchar(127)
    , surname varchar(127)
) ;

create table directors (
    id       serial primary key
    , name   varchar(127)
    , surname varchar(127)
) ;

create table moviedirectors (
    directorid int
    , movieid int
    , info    text
    , primary key (directorid, movieid)
);

create table movieroles (
   actorid    int
   , movieid  int
   , info_1   text
   , info_2   text
   , role     text
   , primary key (actorid, movieid)
) ;

create table moviegenres (
   movieid   int
   , genre   text
   , primary key (movieid, genre)
) ;

create table movieplots(
   movieid   int primary key
   , plot    text
) ;
```

```
create table imdbratings(
   movieid   int primary key
   , distribution varchar(127) not null
   , rating  float
   , numvotes int
) ;

-- Data mined from Twitter for movie
    rating
create table twittergenres (
   movieid    int
   , genre    varchar(100)
   , primary key (movieid, genre)
) ;

create table twitterusers (
   id         int  primary key
   , twitterid bigint
);

create table twitterratings (
   userid     int
   , movieid  int
   , rating   int
   , whenrated timestamp
   , primary key (userid, movieid)
);
```