Database Systems, CSCI 4380-01
Homework # 5
Due Friday October 23, 2019 at 11:59:59 PM

Homework Statement. This homework is worth 5% of your total grade. If you choose to skip it, Midterm #2 will be worth 5% more. This homework will concentrate on SQL skills. You will be able to test your queries against real data and expected output.

I will supply expected output for `ministreaming` as in hw#4 to test your queries. I will also supply a second, bigger database for testing as well early next week and test output for it. Queries in this homework are more complex and involved than the first homework, so be very careful in writing correct and efficient queries. We will measure query run times and queries that run unreasonably long may not generate output. So, test your queries carefully before submitting.

1 Problem Description

Write the following queries in SQL. In all your queries, use the simplest possible expression possible. Do not forget your JOIN conditions and pay attention to returned attributes and ordering of tuples.

Query 1 Return the name of all directors (other than `Steven Spielberg`) who have the same number of movies as `Steven Spielberg` in the database. Order by director.

Query 2 For each country in the database (in countryseries), return the country and the total number of series (numseries) they offer in the `TV Comedies` category. Order results by numseries descending and country ascending.

Query 3 Order languages by the number of movies that they have in descending order and by language name in ascending order. Return the languages that rank between 15-40 (inclusive at both ends) in this ordering, order the results in ascending order of number of movies and language ascending. (Note: this was a favorite interview query for a while...)

Query 4 Find movies offered in the highest number of languages. Return the title of the movie, number of languages (nummovies). Order results by title ascending.

Query 5 Find the difference between the average imdbrating of movies in the `Horror` genre released before year 2000 and after year 2000. Return the average rating before 2000 (avgbefore), after 2000 (avgafter) and the difference (avgdiff). Cast all your average values as NUMERIC(5,2). Use the `movies.year` attribute in this query.

Query 6 For the following countries: Turkey, France, China, India, Thailand, Japan, find series with an imdb rating of at least 7.5 and are only available in that country. Return the country name, series title and its imdbrating. Order results by country, imdbrating and series title ascending.

Query 7 This query builds on the logic of the previous query.

For each country (other than United States), find series that are only available in that country. Among these series, find the series with the highest imdbrating for each country. Return the country, title and the imdb rating of these series, order by country and title asc.

Query 8 Find pairs of platforms (renamed platform1, platform2) such that platform2 offers more than 60% of the series offered by platform1. Return pairs of platforms (renamed platform1, platform2) and proportion of series they have in common (numcommon) formatted as numeric(5,2). Order the results by platform1, platform2 and numcommon.

Query 9 Return the name of cast members who have starred in more than two of the same director's movies and did not direct any movies themselves. Order results by cast member name ascending.

SUBMISSION INSTRUCTIONS. You will use SUBMITTY for this homework. Please submit each query to the appropriate box in Submitty.

You must add a semi-colon to the end of each query to make sure it runs properly.

If you want to provide any comments, all SQL comments must be preceded with a dash:

```
-- Example comment.
```

Database Schema

This database is merged from multiple datasets, containing data for movies and TV Series, both appearing in various streaming platforms.

Note that this is real data, scraped from websites and then parsed by me, so it may be noisy. Make a habit of using simple queries to first explore the data to understand various issues.

```
create table movies(
      movieid      int  primary key
      , title      varchar(1000)
      , year       int        -- year the movie was made
      , contentrating varchar(10) -- age group the movie is intended for
      , imdbratin    float      -- imdb rating
      , rottentomatoes float  -- rotten tomatoes rating
      , runtime    int
      , date_added date       -- date movie is added to Netflix
      , duration   varchar(40)
      , description text
) ;

-- Which countries the movie is available in
create table moviescountry (
     movieid int
     , country varchar(100)
     , primary key (movieid, country)
     , foreign key (movieid) references movies1(movieid)
) ;

-- Directors of the movie
create table moviesdirectors (
     movieid int
     , director varchar(100)
     , primary key (movieid, director)
     , foreign key (movieid) references movies1(movieid)
) ;

create table moviesgenres (
     movieid int
```

```
      , genre varchar(100)
      , primary key (movieid, genre)
      , foreign key (movieid) references movies1(movieid)
) ;

-- Languages the movie is available in
create table movieslanguages (
      movieid int
      , language varchar(100)
      , primary key (movieid, language)
      , foreign key (movieid) references movies1(movieid)
) ;

-- People who starred in the movie
create table moviescast (
      movieid int
      , castname varchar(100)
      , primary key (movieid, genre)
      , foreign key (movieid) references movies1(movieid)
) ;

-- Which platform the movie is in (if ispresent is True),
-- or not in (if ispresent is False)
create table moviesonplatform (
      movieid int
      , platform varchar(100)
      , ispresent boolean
      , primary key (movieid, platform)
      , foreign key (movieid) references movies1(movieid)
) ;

-- TV series on various platforms
create table series (
      seriesid int  primary key
      , title varchar(400)
      ,  yearreleased  int
      ,  contentrating varchar(40) -- age group the movie is intended for
      ,  imdbrating    float       -- imdb rating
      ,  rottentomatoes int        -- rotten tomatoes rating
      ,  description   text
      ,  seasons       int         -- how many seasons are available
      ,  date_added    date        -- date series is added to Netflix
) ;

-- Which platform the series is available in
create table seriesonplatform (
      seriesid int
      , platform varchar(100)
      , primary key (seriesid, platform)
      , foreign key (seriesid) references series(id)
) ;

-- People who starred in the series
create table seriescast (
      seriesid int
      , castname varchar(100)
      , primary key (seriesid, castname)
      , foreign key (seriesid) references series(id)
) ;
```

```
-- Categories for series
create table seriescategory (
      seriesid int
      , category varchar(100)
      , primary key (seriesid, category)
      , foreign key (seriesid) references series(id)
) ;

-- Countries the series is available in
create table seriescountry (
      seriesid int
      , country varchar(100)
      , primary key (seriesid, country)
      , foreign key (seriesid) references series(id)
) ;

-- Directors for the series
create table seriesdirectors (
      seriesid int
      , country varchar(100)
      , primary key (seriesid, country)
      , foreign key (seriesid) references series(id)
) ;

-- Genre from series
create table seriesgenres (
      seriesid int
      , genre varchar(100)
      , primary key (seriesid, genre)
      , foreign key (seriesid) references series(id)
) ;
```