Database Systems, CSCI 4380-01
Homework # 6
Due Monday November 2, 2020 at 11:59:59 PM

Homework Statement. This homework is worth 4% of your total grade. If you choose to skip it, Midterm #2 will be worth 4% more. This homework will concentrate on SQL skills using PL/pgSQL.

Since this homework requires you to create new tables and functions, you must your own database, already created for you in the course database. It will be named: `db_⟨yourusername⟩`.

I have already created a small database for testing your function in each of your databases. It will only contain the series related tables. Please do not change the already given tables as it will be costly for me to recreate them. Do all your work by creating new database objects as described below.

Problem Description

Create a single PL/pgSQL function called `recommendation` that takes as input:

- `inputseries`: an array of integers containing seriesid values,
- `topk`: an integer, how many tuples/series to return from the function,
- `w1,w2,w3,w4`: floats to be used as weights.

I have already provided you with a skeleton function description below that illustrates how to use the array data type. Quite simply, you can convert the array to a relation in the FROM statement using: `unnest(inputseries) as input(seriesid)`.

In your hw solution, please keep the function name, input and output types the same as the one I provided you. Simply replace the function body with your solution. Do not create any tables or other objects outside of the function. Inside the function, you can create any number of tables or objects. Simply drop them before the function ends.

We will test your code by first running your submission to create the function, and then run your function on different inputs to test. So, make sure that your function can be executed multiple times without a problem once created.

Functionality

Let's talk about what your function is supposed to do.

You are writing a function to make recommendations to users based on a few series that they like. The `seriesid` of series is given in `inputseries` (input to the function).

Based on this input and the input weights `w1,w2,w3,w4`, compute a match score for every series in the database (except for those in `inputseries`) and return the title and score of the series and return the `topk` tuples (series) with the highest score as a string.

To compute this, simply order series by match score descending and series title ascending and limit results to `topk` values.

The score of any series `S` (that is not in the `inputseries`) is given by: `w1*a1+w2*a2+w3*a3+w4*a4` where:

- `a1` is the total number of categories (not distinct) that series `S` has in common with any series in `inputseries`.
- `a2` is the total number of platforms the series `S` is on.
- `a3` is the total number of people (cast members and directors) (not distinct) series S shares with any of the input series.
- `a4` is the average of IMDB and Rotten tomatoes rating for the series. If one of these two ratings is NULL, only the other rating is used. If both are NULL, then a4 is zero. Note: Rotten tomatoes rating must be divided by 10!

Compute all of `a1,a2,a3,a4` as floats and add them up. Use this for ranking. When displaying the score, format it as `NUMERIC(6,3)`.

A draft of the function definition is given below.

```
CREATE OR REPLACE FUNCTION
    recommendation(inputseries integer array
              , topk int
          , w1 float, w2 float, w3 float, w4 float)

    RETURNS varchar AS $$
DECLARE
   values VARCHAR ;
   myrow RECORD ;
BEGIN
   values = 'You entered: ' || E'\n';

   FOR myrow IN SELECT s.title
         FROM unnest(inputseries) as input(seriesid)
         , series s
         WHERE s.seriesid = input.seriesid
   LOOP
       values = values || myrow.title || E'\n';
   END LOOP ;

   RETURN values ;
END ;
$$ LANGUAGE plpgsql ;
```

To create the function, simply use the `SQL Command` section of the interface first, then run example calls. Here is an example call to this function to test it:

`select recommendation(ARRAY[1587,13,255],10,1,0.2,2,0.3);`

SUBMISSION INSTRUCTIONS. You will use SUBMITTY for this homework. Please submit a single file named `hw6.sql` that creates the function and does not run it.

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

`-- Example comment.`

Please double check your submission before submitting. We will not debug functions that do not run. Use a plain text editor for your submission, similar to what you would use for C++ or Python.

Database Schema

This database contains data for TV Series 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.

```
-- 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
```

```
     , director varchar(100)
     , primary key (seriesid, director)
     , foreign key (seriesid) references series(id)
) ;
```