Database Systems, CSCI 4380-01
Homework # 4
Due Monday February 26, 2018 at 11 PM

Introduction.

This homework is worth 5% of your total grade. If you choose to skip it, Midterm #2 will be worth 5% more. Remember, practice is extremely important to do well in this class. I recommend that not only you solve this homework, but also work on homeworks from past semesters. Link to those is provided in the Piazza resources page.

This homework is on SQL. It attempts to teach you how to write simple SQL queries. All queries in this homework can be solved using basic SQL that we have learnt so far (i.e. using SELECT/FROM/WHERE/GROUP BY/HAVING/ORDER BY/LIMIT and UNION/INTERSECT/EXCEPT). There is no need to use complex nested expressions, anything beyond what is covered here: `http://www.cs.rpi.edu/~sibel/csci4380/spring2018/course_notes/sql_basics.html`

Concentrate on simple structure as much as possible, do not overcomplicate queries. Do not use DISTINCT unless it is necessary. Do not join with relations that are not needed for the query. We might take points off if we see unnecessary relations in the FROM clause and if you are missing join conditions even if your query returns the correct answer. Now is the time to write efficient queries.

As a special treat, I will also provide the correct answers to each query to test against. If we find an error, the correct answers may be updated. So, please remember to check against the latest copy on Piazza.

Database Server Use Rules

If you want to install and create the database on your own computer, you can use the data scripts I used. You do not have to have a database server installed to do this homework. This database is already created as `olympics` on the shared database server at:

`http://rpidbclass.info`

Feel free to use it for testing your queries, but please do considerate of others when using the server. Here are a few ground rules:

- I have no idea how the server will respond to load. This is my first experience with this cloud service. So, be patient if you see some slow down. This is a medium sized database for a 100+ class, so you can expect a serious slow down near the homework deadline. Please do not wait till the last minute to submit your homeworks.
- Test your queries one at a time. Best set up is using a browser and a text editor. Write queries elsewhere and test in the server with cut and paste.

- Make every effort to read your queries before submitting. A forgotten join condition may mean disaster. Check join conditions first, then run your queries.
- Remember if you have an unresponsive query, it will continue to use system resources even if you quit your browser. So, opening a new browser window will solve your problem but may slow things down for everyone. Queries should terminate after 2 minutes, so if you increased the load with a bad query, then wait for your query to end before submitting another.

  If you are experiencing problems with a query, read it carefully before running it again in a separate window. Remember: missing join conditions is the difference between: 2,094,266,610,000 tuples and 3335 tuples.
- If the server is not responsive, let us know on Piazza. I will see if some jobs need to be killed or whether server needs to be made more powerful. Please be patient.
- Please do not include a query that does not run in your homework submission. I will run all your queries in a batch job and an incomplete query will cause me a great deal of problems.

Homework Description



In this homework, you will use a real database of all Olympic medals from winter and summer olympics. I have also included an additional country table for extra data though the correctness of data in this table is suspect. Note that not all countries for athletes have a matching tuple in this table.

As we are dealing with real data, there can be some unusual issues. We will deal with that as we come across it and clarify queries. Note that I purposefully removed all Unicode data to make life easy for all of us. That may have caused issues with the names of your favorite athletes, sorry about that.

Read the schema below carefully. It should give you all the information you need for the queries below.

Given this database, write the following queries using SQL (in no particular order of difficulty):

Query 1. Return the name and type of all events for the discipline `Figure skating`, order by name first and then type.

Query 2. Return the code and name of all countries that won a medal in the `1924 Chamonix Winter` Olympics, order by code first and then name.

Query 3. For each winter olympic game, find the total number of countries (by code) that won a medal that game. Return the year and the number of countries, order in descending order by the count first and then by the year. (Note: In this, you do not need to join with the `countries` relation, just use the country code for athletes. Beware, some athletes have NULL country. Do not count these.)

Query 4. Find all athletes who won a gold medal in the same event in two winter Olympic games 12 years or more apart. Return the name and country of the athlete, the name of the event, the years and cities for the Olympics. Order by Olympic years first, then by athlete name.

Query 5. Find all winter events that have been discontinued in the 21st century. Return their name and corresponding sport discipline, order by name and discipline.

Query 6. Return id, name and country of all atheletes who won a gold medal in both winter and summer olympic games. Order by name and country.

Query 7. Return id, name, country of all athletes who won 6 or more Gold medals in a single Summer Olympic game. Order by name and country.

Query 8. Return the id, name, country of all athletes who won a bronze medal in winter or summer olympic games, whose first name starts with Z and comes from a country with 'U' in its country code. Exclude those names with no commas and order by name and country.

Query 9. Find the top 10 countries with the highest medal count to GDP ratio for summer olympics. Order by the ratio first and country name second. Return country code, name, medal count, GDP and the ratio of medal count to GDP. Exclude countries with no GDP value.

Query 10. Find athletes who won all three types of medals in summer olympics and competed under three different countries. For this query, you will be looking at athletes with the same name but different countries (and different ids).

Query 11. Find pairs of athletes a1, a2 such that a1's first name is a2's last name, and a2's first name is a1's last name. The athletes must not have the same first and last names and must have a comma in their names. For each pair, only return one tuple. Order the results by name.

Query 12. Find athletes who won medals in three different summer sports (i.e. different summer sport name). Note: look for different names as some sports with different id may have the same name.

Submission Instructions.

Submit a single ASCII text file named `username_hw4ans.sql` that contains all your queries. Your script should be formatted as shown below:

```
-- Print your answer and RCS id first
SELECT 'Student: Sibel Adali (adalis@rpi.edu)';

-- Print the name of each query before the query output
-- Pay close attention to the columns requested as well as the
-- requirements for ordering of results for each comparison

SELECT 'Query 1';

-- Replace this with your answer for Query 1.
SELECT count(*) FROM sports ;

--- Repeat this pattern for each query

SELECT 'Query 2';
```

```
-- Replace this with your answer for Query 2.
SELECT count(*) FROM sports ;


SELECT 'Query 3';

-- Replace this with your answer for Query 3.
SELECT count(*) FROM sports ;
```

I am not super sure where the submission will be as I want to have the ASCII files to run and then also the ability to grade on Gradescope. Unfortunately, Gradescope only allows PDF. As soon as I figure this out, I will announce it on Piazza.

Database Schema

```
create table countries(
     name         varchar(255) not null
     , code       char(3) primary key
     , population int
     , gdp        float --gross domestic product
) ;
create table olympics (--all olympic games, winter or summer
      id      int primary key
      , year  int not null
      , city  varchar(255) not null
      , otype char(6) not null --winter or summer
      , check (otype in ('winter','summer'))
);
create table sports (--all sports in winter and summer games
      id          int primary key
      , name      varchar(255) not null
      , discipline varchar(255) not null
      , stype     char(6)     not null  --- winter or summer
) ;
create table events (--all events in winter and summer games
      -- team vs individual events not distinguishable
      id         int primary key
      , sid      int not null -- which sport this event is for
      , name     varchar(255) not null
      , etype    char(1) -- m or f
      , constraint event_fk foreign key (sid)
            references sports(id)
) ;
create table athletes (--all athletes who won a medal in an olympic game
      id       int primary key
      , name   varchar(255) not null
      , country char(3) -- same as countries(code)
      -- but not all countries are present in countries(code)
) ;
create table summer_medals (--medals given in summer olympics
      id      int primary key
      , oid   int not null -- olympic game id
      , aid   int not null -- athlete id
      , eid   int not null -- event id
      , medal varchar(6) not null -- Gold Silver or Bronze
      , constraint sm_fk1 foreign key (oid) references olympics(id)
      , constraint sm_fk2 foreign key (aid) references athletes(id)
      , constraint sm_fk3 foreign key (eid) references events(id)
) ;
create table winter_medals (--medals given in summer olympics
      id      int primary key
      , oid   int not null  -- olympic game id
      , aid   int not null  -- athlete id
      , eid   int not null  -- event id
      , medal varchar(6) not null
      , constraint sm_fk1 foreign key (oid) references olympics(id)
      , constraint sm_fk2 foreign key (aid) references athletes(id)
      , constraint sm_fk3 foreign key (eid) references events(id)
) ;
```