Database Systems, CSCI 4380-01
Homework # 4
Due Thursday October 11, 2018 at 11:59 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.

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 will be created as `hw4` on the shared database server at:

```
http://rpidbclass.info
```

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

- Server response to load can be unpredictable. So, be patient if you see some slow down. This is a medium sized database for a 100+ student 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 Jeopardy contestants. For this homework only, I have scaled the database to the last two seasons to make it easier to query. It is still a fairly large database. As this data was scraped from the web by me, there is likely to be lots of missing data and errors. We will discover them as we use it. Above is a picture of one of our best friends (left in the picture) who became the first streaker (5 consecutive wins) of 2018 in Jeopardy!

If you do not know about Jeopardy, it is a game show in which contestants answer increasingly difficult questions/clues about many topics. Each clue has a point value, but some clues are special (daily doubles) where the contestant can provide a bet and choose the point value. There are three main parts: 'Jeopardy' (easier questions), 'Double Jeopardy' (harder questions) and 'Final Jeopardy' (a single last question). For the Jeopardy and Double Jeopardy, contestants will try to answer as quickly as possible (using a buzzer). If the first person's answer is incorrect, others may also answer. The daily doubles are questions only one contestant can answer (whoever picked it). Finally, in the final jeopardy, all contestants (as long as they have positive scores before that point) will answer and provide a bet. If they answer correctly, they will as many points as their bet and if they answer incorrectly, they will loose as many points as their bet.

You can review the data model to see more details.

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

Query 1. Return the full name of all contestants who had at least 5 consecutive wins (hint: check out the description of contestants). Order by fullname.

Query 2. Return the full name of all constants whose short name start with letter 'b' who answered a daily double clue correctly (`isdd` is True) in the `Double Jeopardy` part of the short (`cat_type` is `'DJ'`). Order by full name.

Query 3. Return the gameid, clue text and category for all final jeopardy clues that were triple stumpers (no contestant has answered them correctly). Order by gameid, clue, category.

Query 4. Return the id of all games, shortname for a pair of contestants in which at least two contestants were tied going into final jeopardy (i.e. according to their scores in Round 3). For each pair of contestants, only return one pair (alphabetically ordered). Order by gameid and names.

Query 5. Return all game rounds in which all clue categories were 11 characters or less (each `cat_type` value `J` or `DJ` is a different game round). Order by game id and category type.

Query 6. Return the full name, final game score and description of contestants from 'Wisconsin' with the highest final score in a sigle game among all other contestants from 'Winconsin'. Order by full name and score.

Query 7. For each contestant who competed in a game that aired in January, return the game id, contestant full name and the total number of questions the contestant answered correctly in the 'Double Jeopardy' round of that game (considering only the games that the database contains some clues for the 'Double Jeopardy' round). Order by number of correct answers, game id and full name.

Query 8. Find all contestants who have a game in which their Coryat score would have been ten times as much as their Final score, even though their final score was more than 1000 (i.e. they are really bad at betting). Return the gameid, shortname, their final score and the Coryat score. Order by gameid and shortname.

Query 9. Return the text of all clues that are about the 'Internet' (either clue text or the category). Order by clue text.

Query 10. Return the gameid, contestant short name and final score of all games in which the contestant had a negative score in Round 2, but eventually won the game (with the highest final score). Order by final score and game id.

Submission Instructions.

Submit a single ASCII text file named `username_hw4ans.sql` that contains all your queries to SUBMITTY. I will post submission instructions later for this on Piazza. We will use Submitty for all SQL homeworks. However, Submitty is not yet set up, so this may take some time.

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 games ;

--- Repeat this pattern for each query

SELECT 'Query 2';

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

```
SELECT 'Query 3';

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

Database Schema

```
-- Each game is in a season, given by id
CREATE TABLE games
     ( id INT  -- season id
       , gameid INT
       , airdate DATE
       , PRIMARY KEY (gameid)
     ) ;

-- Each contestant is identified by a shortname, which is unique for a
-- game.

CREATE TABLE contestants
     ( gameid INT
       , fullname VARCHAR(100)
       , description VARCHAR(255)
       , shortname  VARCHAR(100)
       , PRIMARY KEY (gameid, shortname)
       , FOREIGN KEY (gameid) REFERENCES games(gameid)
     ) ;

-- The overall scores of each contestants after different rounds
-- of the game.
-- Rounds '1', '2' are in the first stage called the 'Jeopardy' stage,
-- Round '3' is after 'Double Jeopardy' before 'Final Jeopardy'.
-- Round 'Final Score' is the actual score of each person
-- Round 'Coryat Score' is the hypothetical score without the bets
-- Round '6' is an error, which needs to be identified later.

CREATE TABLE scores
      ( gameid INT
        , shortname VARCHAR(100)
   , score INT
   , round VARCHAR(20)
   , PRIMARY KEY (gameid, shortname, round)
   , FOREIGN KEY (gameid, shortname)
     REFERENCES contestants(gameid, shortname)
      ) ;

-- Each game has many clues, clue is the question, and correct_answer is the answer
-- value is the dollar value of the clue: amount player wins/looses
-- for correct, incorrect answers
-- category is the named of the category
-- cat_type is one of: 'J': 'Jeopardy' round and 'DJ': 'Double Jeopardy' round
-- isdd is true if the question was a double jeopardy question

CREATE TABLE clues
     ( gameid INT
       , clueid INT
       , clue   TEXT
       , value  INT
       , category VARCHAR(255)
       , cat_type VARCHAR(10)
       , isdd     BOOLEAN
       , correct_answer VARCHAR(255)
       , PRIMARY KEY (gameid, clueid)
       , FOREIGN KEY (gameid) REFERENCES games(gameid)
```

```
     ) ;

-- Each contestant can answer a clue, if the answer is wrong,
-- another contestant can answer. This relation stores all
-- contestants who gave a response (but not what they said).
-- If there is no correct answer for a question here, it means
-- that no contestant answered the question correctly.

CREATE TABLE responses
     ( gameid INT
       , clueid INT
   , shortname VARCHAR(255)
   , iscorrect BOOLEAN
   , PRIMARY KEY (gameid, clueid, shortname)
   , FOREIGN KEY (gameid, clueid) REFERENCES clues(gameid, clueid)
   , FOREIGN KEY (gameid, shortname)
     REFERENCES contestants(gameid, shortname)
     ) ;

-- At the end of the game, there is a single question/clue called
-- the 'Final Jeopardy'. This relation stores the clues for this
-- specific round. There is no dollar value attached to these questions.

CREATE TABLE final_clues
     ( gameid INT
       , clue   TEXT
       , category VARCHAR(255)
       , correct_answer VARCHAR(255)
       , PRIMARY KEY (gameid)
       , FOREIGN KEY (gameid) REFERENCES games(gameid)
     ) ;


-- For the 'final jeopardy', all contestants give an answer and a bet
-- The bet is the dollar amount the contestant will win/loose if they
-- answer correctly. Only contestants with positive winnings/scores
-- at round '3' can participate. This relation stores the bets and
-- whether each person scored correctly or not.

CREATE TABLE final_responses
     ( gameid INT
   , shortname VARCHAR(255)
   , iscorrect BOOLEAN
   , bet  FLOAT -- VARCHAR(10)
   , PRIMARY KEY (gameid, shortname)
   , FOREIGN KEY (gameid, shortname)
     REFERENCES contestants(gameid, shortname)
     ) ;
```