Database Systems, CSCI 4380-01
Homework # 5:
Double Jeopardy and Final Jeopardy
Due Friday October 19, 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.

Some of these queries are difficult. Try to solve any part of it that you can, test simpler conditions before making it more complex. We will give lots of partial credit for queries that return correct (or mostly correct) results but miss some of the final attributes. Please do not turn in grossly inefficient queries or queries that generate massive output. We will take points off for these (my answers to all queries take about 1-2 seconds on my personal computer). Try your best to seek simplest queries you can write and also check for join conditions.

This homework will use the full database of jeopardy, named `jeopardy`. The database will be up in the server soon and I will supply the correct answers very soon as well.

This is a much larger database than the last one (and has many more parsing errors), so be careful when testing your queries. Do not start this homework in the last minute. Give yourself time to implement parts of queries, test and think about how to do the rest. You can also expect heavier load on the server and slower response times.

Homework Description

The data model is the same for this homework except we have added a new `states` table. Given this database, write the following queries using SQL (in no particular order of difficulty):

Query 1. Query 1. Return name of categories that appeared in final jeopardy 6 or more times and never appeared in the other rounds of the game (i.e. in the clues table). Order by category name.

Query 2. Find the full name of the contestant with highest total score (sum of final scores) over all his/her games. You can assume there is a single such person. (Note that the database does not contain the championship rounds in case you are unsatisfied with the answer.)

Query 3. Query 3. Find answers to clues that are longer than 20 characters that were answers to more than 6 clues. Return the answer, the number of clues it was an answer to and the number of categories that had a clue with this correct answer, order by the number of clues.

Query 4. Find categories that have a correct response less 60% of the time, order by the number of different games these categories appeared in and return the top 20 most frequent such categories.

For each category, return the name, number of games it appeared in, and the percentage of correct answers in that category (number of times a correct answer is given for a clue in this category divided by the number of clues in the category). Order by number of games and percent correct.

Query 5. Query 5. Find people who had higher scores than 'Ken Jennings' (fullname) in Round 3. Return their full name, gameid in which this happened and the score difference in Round 3. Order by score difference.

Query 6. Return the gameid and air date of games that has at least 15 triple stumpers in Jeopardy or Double Jeopardy rounds. Order ther results by airdate.

Query 7. Find the full name and total winnings of the lowest total scoring 5 time champion (i.e. sum of final scores of exactly 5 time champions in). (Note: This could be a pretty tricky query. I recommend you use WITH here.)

Query 8. Find categories that appeared in two games more than 30 years between them (meaning the category did not appear between two appearances that are 30 years apart.) Use 365 days to represent a year.

Query 9. For each state, return the state name and the number of contestants per capita from this state as well as the number of contestants (+1) who were at least 5 time champions per capita (percapita5times) from this state. Order by percapita5times. Note: for this query, we added a new table called states(name,population) To find a percapita value, you need to divide the population of the state with number of contestants.

Clarifications. A few clarifications on the results here.

Number of contestants for each state is non-zero, so we compute per capita directly: `population/num_cont` However, number of contestants with at least 5 time champions can be zero, so I compute per capita as: `population/(1+num_5plustimers)`. This avoids division by zero. It is a hack, sorry about that. We will learn case statement soon that will simplify this.

The second one is finding the contestants from a state. This could get complex. So, I am looking for the following pattern:

`...from...,...statename...`

in description for a contestant. Note that any part in red is some other text. This is not perfect, but it should be close enough.

Finally, to simplify the query, I have used the same trick from Hw 4 to find 5 timers (i.e. by checking 5-time consecutive winners based on the desription). I know it is not consistent, but hopefully this will simplify the query.

Query 10. FINAL JEOPARDY! Find the full name of contestants who won at least two games in which they were third going into final jeopardy (i.e. at round 3). Order by name.

Submission Instructions.

Submit a single ASCII text file named `username_hw5ans.sql` that contains all your queries to SUBMITTY. It should have the same format as Homework #4.

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

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 ;
```

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)
     ) ;

CREATE TABLE states
     (  name VARCHAR(50)
        , population INT
     ) ;
```