Name

UIN

Homework Assignment 2

Due Date: March 05, 2024

CS480 - Database Systems

Please leave this empty!

2.1 2.2

2.3 2.4 2.5 2.6 2.7 2.8 2.9 2.10

2.15 2.16 2.17 2.18 2.19 Sum

Instructions

- Try to answer all the questions using what you have learned in class
- **When writing a query, write the query in a way that it would work over all possible database instances and not just for the given example instance!**
- Some questions are marked as bonus. You do not have to answer these questions to get full points for the assignment. However, you can get bonus points for these questions!
- **Please submit the homework electronically using gradescope**
- **Submissions will be automatically graded. You can submit multiple times.**
- **The due date is 03/05!**

Consider the following database schema and example instance storing information about theatre plays:

student

UIN	name	major	year
111111	Peter	CS	1
112333	Bob	CS	2
443223	Alice	EE	3
234242	Veeta	CS	2
366666	Fauly	CS	6
554344	Rufus	BIO	2
777777	Gertrud	CS	1

hackathon

title	start_date	end_date	description
Sparkhacks 2024	2024-02-08	2024-02-11	UIC-based event
Sparkhacks 2023	2023-02-01	2023-02-02	UIC-based event
Chasehacks 2022	2022-01-01	2022-01-02	Bank-organized

team

teamname	hackathon
The bosses	Sparkhacks 2024
The bosses	Sparkhacks 2023
The bosses	Chasehacks 2022
The noobs	Sparkhacks 2024
Hack On	Sparkhacks 2024
Hack On	Sparkhacks 2023

student_team

teamname	hackathon	student
The bosses	Sparkhacks 2024	366666
The bosses	Sparkhacks 2024	443223
The bosses	Sparkhacks 2023	366666
The bosses	Sparkhacks 2023	443223
The bosses	Chasehacks 2022	366666
The bosses	Chasehacks 2022	443223
The bosses	Sparkhacks 2024	111111
The noobs	Sparkhacks 2024	777777
The noobs	Sparkhacks 2024	554344
The noobs	Sparkhacks 2024	443223
Hack On	Sparkhacks 2024	234242
Hack On	Sparkhacks 2024	112333
Hack On	Sparkhacks 2023	234242
Hack On	Sparkhacks 2023	112333

organizer

student	hackathon	role
366666	Sparkhacks 2024	director
234242	Sparkhacks 2024	outreach
366666	Sparkhacks 2023	director
111111	Chasehacks 2022	director

project

teamname	hackathon	title
The bosses	Sparkhacks 2024	Boring React Mockup
The bosses	Sparkhacks 2023	Convoluted Haskell Nonsense
The bosses	Chasehacks 2022	Verbose Java Mess
The noobs	Sparkhacks 2024	Too many fancy frameworks
Hack On	Sparkhacks 2024	Static HTML to the Max
Hack On	Sparkhacks 2023	Unnecessary Assembly

sponsorship

hackathon	sponsor	amount
Sparkhacks 2024	UIC	3500
Sparkhacks 2024	Chase	500
Sparkhacks 2024	Oracle	10000
Sparkhacks 2023	UIC	4500
Chasehacks 2022	Chase	20000
Chasehacks 2022	UIC	250

judge

name	affiliation	position
Lazy young prof	UIC	Assistant Professor
Lazy old prof	UIC	Full Professor
No morals industry guy	UIC	Associate Professor

rating

judge	team	hackathon	project	rating
Lazy young prof	The bosses	Sparkhacks 2024	Boring React Mockup	2.0
Lazy old prof	The bosses	Sparkhacks 2024	Boring React Mockup	5.0
No morals industry guy	The bosses	Sparkhacks 2024	Boring React Mockup	3.5
Lazy young prof	The bosses	Sparkhacks 2023	Convoluted Haskell Nonsense	5.0
Lazy old prof	The bosses	Sparkhacks 2023	Convoluted Haskell Nonsense	1.0
No morals industry guy	The bosses	Chasehacks 2022	Verbose Java Mess	5.0
Lazy young prof	The noobs	Sparkhacks 2024	Too many fancy frameworks	2.5
Lazy old prof	The noobs	Sparkhacks 2024	Too many fancy frameworks	4.0
No morals industry guy	The noobs	Sparkhacks 2024	Too many fancy frameworks	4.5
Lazy young prof	Hack On	Sparkhacks 2024	Static HTML to the Max	1.0
Lazy old prof	Hack On	Sparkhacks 2024	Static HTML to the Max	3.0
No morals industry guy	Hack On	Sparkhacks 2024	Static HTML to the Max	1.5
Lazy young prof	Hack On	Sparkhacks 2023	Unnecessary Assembly	1.0
Lazy old prof	Hack On	Sparkhacks 2023	Unnecessary Assembly	5.0

Hints:

- If you are unsure about the type of an attribute look at the SQL script for the homework. The same applied to foreign key constraints which are defined the DDL script for the homework.
- Attributes with black background form the primary key of an relation.

Part 2.1 SQL DDL (Total: 14 Points)

Question 2.1.1 (5 Points)

Write an SQL statement that adds a *location* attribute to relation *hackathon* which stores the location (a string) where the hackathon takes place. Ensure that no hackathons have a `NULL` location. The default location should be *"TBA"*.

Question 2.1.2 (9 Points)

Write an SQL statement that creates a new table `price`. This table should record information about prices for hackathons. You need to store: (i) the `name` of the price, (ii) the name of the `sponsor` that pays for the price, (iii) the `amount`, (iv) the `hackathon` the price is for, and (v) the `team` that won the price. Prices are uniquely identified through the `hackathon` and `name`. Ensure that the name, sponsor, amount, and hackathon attributes cannot contain null values. Make sure to connect the `price` table to the `hackathon` and `team` tables.

After creating the table insert the following data (you need it to test queries). You can find the SQL code commented out in the SQL script for the homework.

For the autograder submission, we will insert data, just submit the `CREATE TABLE` statement.

```
INSERT INTO price
VALUES
('Chasewin', 'Chase', 350.0, 'Sparkhacks 2024','The bosses'),
('Cloudwin', 'Oracle', 550.0, 'Sparkhacks 2024','The noobs'),
('Good win', 'UIC', 750.0, 'Sparkhacks 2024','The noobs'),
('Good win', 'UIC', 150.0, 'Sparkhacks 2023','The bosses'),
('Chasewin', 'Chase', 250.0, 'Sparkhacks 2023','Hack On'),
('Lota Money', 'Chase', 2000.0, 'Chasehacks 2022','The bosses');
```

Part 2.2 SQL Queries (Total: 56 + 10 BONUS Points)

Question 2.2.1 (5 Points)

Write an SQL query that returns for every judge their average rating, minimum rating, and maximum rating. The result schema should be `(judge, avg_rating, min_rating, max_rating)`.

Question 2.2.2 (5 Points)

Write an SQL query that returns projects (the project title) paired with a student participating in the project, the hackathon the project was developed at, and the description of this hackathon. Return all combinations. If there are multiple students in a team working on a project, return one row for each such student. The result schema should be `(project, student, hackathon, description)`.

Question 2.2.3 (7 Points)

Write an SQL query that returns the names of students that have participated in at least 2 teams. Participating in n teams in a single hackathon should contribute n to the count for a student. Also participating with a team with the same name in multiple hackathons should be considered as multiple teams for a student.

Question 2.2.4 (7 Points)

Write an SQL query that returns the two teams (their team name and average rating) that have received the largest average rating. The result schema should be `(team, avgr)`.

Question 2.2.5 (8 Points)

Write an SQL query that returns for each student a rolling sum of the total amount of prices (using the new price table you have created) they have won. Show this rolling sum over time based on `start_date`'s of hackathons. That is, there should be one result row for each hackathon the student has participated in the total amount of price money the student has won at that hackathon and all earlier hackathons. The money awarded for a price should be considered in full for each member of a team winning the price. The result schema should be `(student, start_date, total_prices)`. Only return each pair of students and hackathon once, even if the student won multiple prices at a hackathon.

Question 2.2.6 (8 Points)

Write an SQL query that returns the names of students that have never served as organizers for a hackathon.

Question 2.2.7 (8 Points)

Write an SQL query that returns names of students that have participated in all hackathons.

Question 2.2.8 (8 Points)

Write an SQL query that calculates for each hackathon the number of student per major that have participated in the hackathon. Do not count students twice if they participated in multiple teams in a hackathon. Then determine for each hackathon the major with the maximum number of participants and count for each major how often it was the major with the most students at a hackathon. Return this list sorted in descending order based on the number of times a major had the most participants. You do not need to return majors that did not have the most participants at any event. The result schema should be `(major, nummax)`.

Question 2.2.9 BONUS (5 Points)

Write an SQL query which returns a rank (*gold*, *silver*, or *bronze*) for each team. The rank of a team is determined based on the *average rating* it has received and the *number of prices* it has won using the rules shown below. The result schema should be `(team,rank)`.

- **gold**: an average rating larger 4.0 or at least 3 prices
- **silver**: an average rating larger 3.0 or at least 1 prices
- **bronze**: every team that does not meet the conditions for silver or gold rank

Question 2.2.10 BONUS (5 Points)

Write an SQL query that normalizes the ratings given by each judge by subtracting from each rating the average rating for the judge that has given the rating. Using the normalized ratings, the query should then return projects with the normalized ratings of two judges for the project such that the difference between the normalized ratings for the two judges for this project is more than 1.5. The result schema should be `(hackathon,team,project,judge_1, judge_2, rating_1,rating_2)`. This condition is symmetric. To avoid returning a pair of judges more than once, ensure that `judge_1` is the one with the lower normalized rating.

Part 2.3 SQL Updates (Total: 30 + 5 BONUS Points)

Question 2.3.1 (7 Points)

Delete all hackathons whose start date is before `2024-01-01`.

Question 2.3.2 (8 Points)

Add a new hackathon named *"Sparkhacks 2025"* taking place from `2025-02-01` to `2025-02-04` with description *"UIC-based event"*. Then copy all sponsorships from *"Sparkhacks 2024"* to *"Sparkhacks 2025"* with their amount increased by 100.0.

Question 2.3.3 (6 Points)

Increase all ratings by judge *"Lazy young prof"* by 1.0. However, do not raise any rating about 5.0.

Question 2.3.4 (9 Points)

Update the `hackathon` table delaying the end date of each hackathon by 1 day if the hackathon is currently 2 days long. Note that in Postgres you can add time intervals to dates using the interval data type (`https://www.postgresql.org/docs/current/datatype-datetime.html`).

Question 2.3.5 BONUS (5 Points)

Delete all ratings from the ratings table that were given by judges whose average rating is below 3.0.