CAS CS 460: Introduction to Database Systems – Fall 2019 – Written Assignment #2

Due: October 20th, 11:59pm in gradescope

Problem 1. SQL. [25pts]

Consider the following relations:

Student(snum: integer, sname: string, major: string, year: string, age: integer)
Class(cname: string, meets_at: time, room: string, fid: integer)
Enrolled(snum: integer, cname: string)
Faculty(fid: integer, fname: string, deptid: integer)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class.
Write the following queries in SQL. No duplicates should be printed in any of the answers.

1. Find the names of all Juniors (year = JR) who are enrolled in a class taught by Professor Jonathan.
2. Find the names of all classes that either meet in room R128 or have five or more students enrolled.
3. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.
4. Find the names of students enrolled in the maximum number of classes.
5. Find the names of students not enrolled in any class.

Problem 2. SQL. [25pts]

Consider the following MovieStars database schema:

MovieStar(SNo, sname, gender), Director(DNo, dname, ddob, dcity)
Movie(MNo, DNo, title, genre, year, length, STNo, budget)
Studio(STNo, sname)
StarsIn(SNo, MNo, wage, rstrole)

The MovieStar relation stores information about movie stars. The Director relation stores information about directors and the (DNo) is the primary key. The Movie relation contains a list of movies. For each movie we store a movie number (MNo), director number (DNo), title, genre, year of production, length in minutes, the studio number where the movie was filmed and the budget. Each movie has a single genre (category). The Studio relation stores information about studios and the StarsIn relation stores information about who played in which movie. We store the star number (SNo), the movie number (MNo), the wages of the actor/actress for this movie and if he/she had a rst role in the movie.

Write the following queries in SQL:

1. Find the names of stars who have appeared in at least one movie where the budget greater than 100M.
2. Find the names of stars who have appeared in at least two movies where the pro ts were greater than 100M.

3. List the name of a star that has also directed a movie (was a director on a movie).

4. For each star, nd the average pro t of the movies he/she has appeared in and display it only if the sum of pro ts of the movies he/she has appeared in is more than 2M.

5. Find the names of stars who acted in at least one movie for each genre (have participated in movies for all genres).

Problem 3. (Based on Exercise 5.10) [25pts]

Consider again the following relational schema. An employee can work in more than one department; the *pct* time eld of the Works relation shows the percentage of time that a given employee works in a given department.

Emp(eid: integer, ename: string, age: integer, salary: real)
Works(eid: integer, did: integer, pct time: integer)
Dept(did: integer, dname: string, budget: real, managerid: integer)

Write SQL integrity constraints (domain, key, foreign key, or CHECK constraints; or assertions) or SQL triggers to ensure each of the following requirements, considered independently. You have to give the CREATE TABLE statements for each relation and you can modify it to add constraints if needed.

1. Employees must make a minimum salary of $1000.

2. Every manager must be also be an employee.

3. The total percentage of all appointments for an employee must be under 100%.

4. Whenever an employee is given a raise, the manager's salary must be increased to be at least as much.

Problem 4. Functional Dependencies. [25pts]

Consider the relation R = (A, B, C, D, E, F, G, H). Assume that the following set of FDs holds:

$F = \{AB \longrightarrow CEFGH,\ A \longrightarrow D,\ F \longrightarrow G,\ BF \longrightarrow H,\ BCH \longrightarrow ADEFG,\ BCF \longrightarrow ADE\}$

Find all the candidate keys of the relation R. Decompose the relation in the following normal forms (if it is not there already): a) 3NF, b) BCNF. For each decomposition that you make, state if it is (i) lossless join and (ii) dependency preserving.