COMP 3005: Database Management Systems **(Due: October 1, 2021 (11:59 PM))**

Assignment #1

Instructor: Ahmed El-Roby *Name: , ID:*

Instructions: Read all the instructions below carefully before you start working on the assignment, and before you make a submission.

- The accepted format for your submission is **pdf** only. More details below.
- You can either write your solutions in the tex file (then build to pdf) or by writing your solution by hand or using your preferred editor (then convert to pdf). However, you are encouraged to write your solutions in the tex file. If you decide not to write your answer in tex, it is your responsibility to make sure you write your name and ID on the submission file.
- If you use the tex file, make sure you edit line 28 to add your name and ID. Only write your solution and do not change anything else in the tex file. If you do, you will be penalized.
- All questions in this assignment use the university schema discussed in class (available on Brightspace under Resources → University_Toy_Database), unless otherwise stated.
- For SQL questions, upload a text file with your queries in the format shown in the file "template.txt" uploaded on culearn. An example submission is in the file "sample.txt". You will be penalized if the format is incorrect or there is no text file submission.

Q 1: (7 points)

Answer the following questions using the university schema discussed in class:
(a) The primary key for the *advisor* relation is *s_id*. Suppose a student can have more than one suprvisors. Would *s_id* still be a primary key in *advisor*? If yes, why? If not, what would be a suitable primary key? (3 marks)

No, s_id would not be a primary key, since there may be two (or more) tuples for a single student, corresponding to two (or more) advisors. The primary key should then consist of the two attributes s_id, i_id.

(b) The primary key for *prereq* is both attributes *course_id* and *prereq_id*. Why wouldn't only *course_id* work as primary key? (2 marks)

Because a course could have multiple prerequisites. In this case, we would expect multiple tuples with the same *course_id* and different *prereq_id*. Since primary key is unique, this will not be possible if only *course_id* is the primary key.

(c) Given the existing schema of *teaches*, two or more instructors can teach the same section. How can the primary key be changed to restrict a section to one instructor only? (2 marks)
The primary key would exclude *ID*. This means that there will be at most one tuple for each section. Hence, one instructor.

Q 2: (12 points)

Consider the following bank database schema:
branch(branch_name, branch_city, assets)
customer(ID, customer_name, customer_street, customer_city)
loan(loan_number, branch_name, amount)
borrower(ID, loan_number)
account(account_number, branch_name, balance)
depositor(ID, account_number)
Write an expression in relational algebra to find the following:

(a) Find the cities that host branches that have a loan that is greater than $50000. (3 marks)
$\Pi_{branch\_city}\,(branch \bowtie_{branch.branch\_name=loan.branch\_name} \sigma_{amount>50000}\,(loan))$

(b) Find the ID of each depositor who has an account with a balance greater than $50000 at the "Ottawa" branch. (3 marks)
$\Pi_{ID}(\sigma_{balance>10000 \wedge branch\_name="Ottawa"}(depositor \bowtie_{depositor.account\_number=account.account\_number} account))$

(c) Find the names of customers who have at least one loan amount that is greater than at least one account balance. (6 marks)
$\Pi_{customer\_name}(customer \bowtie_{customer.ID=depositor.ID}$
$(\sigma_{loan.amount>account.balance}((depositor \bowtie_{depositor.account\_number=account.account\_number} account) \bowtie_{depositor.ID=borrower.ID}$
$(borrower \bowtie_{borrower.loan\_number=loan.loan\_number} loan))))$.

Q 3: (33 points)

Using the university database schema discussed in class, write the SQL statements that do:
(a) Create a new course ("Aces of Databases") with ID ("COMP5118") in the Computer Science department ("Comp. Sci.") with 0 credit hours. (3 marks)

```
insert into course
values ('COMP5118', 'Aces of Databases', 'Comp. Sci.', 0);
```

(b) Create a section 'A' for this course in the Winter of 2020 with no known location or time, yet. (4 marks)

```
insert into section
values ('COMP5118', 'A', 'Winter', 2020, null, null, null)
```

(c) Enroll all students in the department into this course. (5 marks)

```
insert into takes(
select ID , 'COMP5118', 'A', 'Winter', 2020, null
from student
where dept_name = 'Comp. Sci.');
```

(d) One student with ID 12345 cannot take this course because of violating the prerequisite requirements (didn't pass COMP3005). Unregister this student from the new section. (3 marks)

```
delete from takes
where course_id= 'COMP5118' and sec_id = 'A' and
year = 2020 and semester = 'Winter' and ID = 12345;
```

(e) For each student who took a course at least twice, show the course_ID and the student ID. (5 marks)

```
select distinct course_id, ID
from takes
group by ID , course_id
having count(*) >= 2;
```

(f) Find the ID and name of instructors who never gave a grade 'A' in the courses they taught (note that instructors who never taught a course satisfy this condition). (5 marks)

```
select ID , name
from instructor
except (
select distinct instructor.ID , instructor.name
from instructor, teaches, takes
where instructor.ID = teaches.ID
and teaches.course_id = takes.course_id
and teaches.year = takes.year
and teaches.semester= takes.semester
and teaches.sec_id= takes.sec_id
and takes.grade = 'A');
```

(g) Rewrite the previous query so that you make sure that the instructor taught at least one course. (5 marks)

```
select distinct(instructor.ID), instructor.name
from instructor, teaches, takes
where instructor.ID =teaches.ID
and teaches.course_id=takes.course_id
and teaches.year = takes.year
and teaches.semester = takes.semester
and teaches.sec_id = takes.sec_id
and takes.grade is not null
except
select distinct instructor.ID , instructor.name
from instructor, teaches, takes
where instructor.ID = teaches.ID
and teaches.course_id = takes.course_id
and teaches.year = takes.year
and teaches.semester = takes.semester
and teaches.sec_id = takes.sec_id
and takes.grade= 'A';
```

without except

```
select distinct instructor.ID , instructor.name
from instructor, teaches, takes
where instructor.ID = teaches.ID
and teaches.course_id = takes.course_id
and teaches.year = takes.year
and teaches.semester = takes.semester
and teaches.sec_id = takes.sec_id
and takes.grade is not null
and instructor.ID not in (
select instructor.ID
from instructor, teaches, takes
where instructor.ID = teaches.ID
and teaches.course_id = takes.course_id
and teaches.year = takes.year
and teaches.semester = takes.semester
and teaches.sec_id = takes.sec_id
and takes.grade = 'A'
);
```

(h) Find the lowest, across all departments, of the per-department maximum salary. (3 marks)

```
select min(maxsalary)
```

```
from (select dept_name, max(salary) as maxsalary
from instructor
group by dept_name);
```

Q 4 (5 points)

Consider the following car insurance schema:
person(driver_id, name, address)
car(licence, model, year)
accident(report_number, date, location)
owns(driver_id, licence_plate)
participated(report_number, licence_plate, driver_id, damage_amount)
Write SQL queries to:

(a) Find the number of accidents involving a car belonging to a person named "Ahmed El-Roby". (3 marks)

```
select count (distinct report_number)
from participated, owns, person
where owns.driver_id = person.driver_id
and person.name = 'Ahmed El-Roby'
and owns.licence_plate = participated.licence_plate;
```

(b) Update the damage amount for the car with licence plate "DB007" in the accident with report number "AR2020" to $3000. (2 marks)

```
update participated
set damage_amount = 3000
where report_number = 'DB007' and
licence_plate = 'AR2020');
```