

Assignment #2

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 formats for your submission are: pdf, docx, txt, and java. 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 or docx). However, you are encouraged to write your solutions in the tex file (5% bonus). 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 (on culearn), 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.
- For programming questions, upload your .java file.
- Late submissions are allowed for 24 hours after the deadline above with a penalty of 10% of the total grade of the assignment. Submissions after more than 24 are not allowed.

Q 1:

(3 points)

Consider the following DDL statements:

```
create table takes
  (ID          varchar(5),
   course_id   varchar(8),
   sec_id      varchar(8),
   semester    varchar(6),
   year        numeric(4,0),
   grade       varchar(2),
   primary key (ID, course_id, sec_id, semester, year),
   foreign key (course_id, sec_id, semester, year) references section
     on delete cascade,
   foreign key (ID) references student
     on delete cascade
);

create table section
  (course_id   varchar(8),
   sec_id      varchar(8),
   semester    varchar(6)
     check (semester in ('Fall', 'Winter', 'Spring', 'Summer')),
   year        numeric(4,0)
     check (year > 1701 and year < 2100),
   building    varchar(15),
   room_number varchar(7),
```

```

    time_slot_id      varchar(4),
    primary key (course_id, sec_id, semester, year),
    foreign key (course_id) references course
        on delete cascade,
    foreign key (building, room_number) references classroom
        on delete set null
);

```

Now, consider the following SQL query:

```

select course_id, semester, year, sec_id, avg (tot_cred)
from takes natural join student
where year = 2017
group by course_id, semester, year, sec_id
having count (ID) >= 2;

```

Will appending **natural join** *section* in the **from** clause change the returned result? Explain why?

Q 2:

(2 points)

Write an SQL query using the university schema to find the names of each instructor who has never taught a course at the university. Do this using no subqueries and no set operations.

Q 3:

(2 points)

Rewrite the following query to replace the natural join with an inner join with **using** condition:

```

select *
from section natural join classroom;

```

Q 4:

(5 points)

Using the university schema, define a view *tot_credits* (*year*, *num_credits*), giving the total number of credits taken in each year. Then, explain why insertions would not be possible into this view.

Q 5:

(10 points)

Write a Java program that finds all prerequisites for a given course using JDBC. The program should:

- Takes a course id value as input using keyboard.
- Finds the prerequisites of this course through a SQL query.
- For each course returned, repeats the previous step until no new prerequisites can be found.
- Prints the results.

Don't forget to handle the case for cyclic prerequisites. For example, if course A is prerequisite to course B, course B is prerequisite to course C, and course C is prerequisite to course A, do not infinite loop.

Q 6:

(5 points)

Consider the following schema:

employee(*emp_name*, *street*, *city*)

works(*emp_name*, *company_name*, *salary*)

Write a function *avg_sal* that takes a company name as input and finds the average salary of employees in the company. Then, write a SQL query that uses this function to find companies whose employees earn (on average) higher salary than the company "Losers Inc."

Q 7:

(10 points)

Design a database for a hospital with a set of patients and a set of medical doctors. The database should capture each encounter between a doctor and a patient on each visit. Associate with each patient a log of the various tests and examinations conducted. Those tests can be ordered by a doctor, but this is not mandatory. The database should keep track of the dates of the tests as well as their results. Draw an ER-diagram for your design. Underline the primary key in the entity sets.

Q 8:

(6 points)

Reduce your ER-diagram to relations. Underline primary keys.

Q 9:

(5 points)

If any test must be ordered by only one doctor, what needs to be changed in your answer to Q7? Either draw a new ER-diagram, or describe the changes. How will the reduced relations change?