New submission for: EXAM 2, 2:30 PM - ONLY CLICK IF YOU'RE READY TO START YOUR EXAM AND THIS IS YOUR TIME SLOT

Database Systems — CSci 4380
Midterm Exam #2 Data Model

In this database, we will store information about a university offering all of its courses online. The database stores detailed information about the courses including which online sites are used as a resource for different components of courses. Additionally, it stores information about exam dates, office hours etc. For each attribute, example values are provided.

Note: Each class in the database may have zero to many of each of the following: instructors, sites for resources, office hours, class meetings, exams and students. Example date formatting: 2020/11/05.

```
-- All courses offered in the university
CREATE TABLE courses (
     classcode       varchar(40) PRIMARY KEY -- ex: CSCI-4380
     , coursename     varchar(200) -- ex: Database Systems
     , credits        int          -- ex: 4
     , department     varchar(40)  -- ex: Computer Science
) ;

-- Classes are offerings of a specific course in a specific semester,
-- year and section. We will assume no cross listed courses for simplicity.
CREATE TABLE classes(
     , crn            int   PRIMARY KEY
     , classcode      varchar(40)
     , semester       varchar(10) -- ex: Fall, Spring, Summer I, Summer II
     , year           int         -- ex: 2020, 2021
     , sectionno      int         -- ex: 1,2
     , FOREIGN KEY (classcode) REFERENCES courses (classcode)
     , UNIQUE (classcode, semester, year, sectionno)
) ;

-- All instructors in the university
CREATE TABLE instructors(
     , id             int   PRIMARY KEY
     , name           varchar(40)   -- ex: Sibel Adali
     , email          varchar(40)  -- ex: adalis@rpi.edu
     , onlineroom     varchar(40)  -- ex: rensselaer.webex.com/meet/adalis
     , note           text
) ;

-- Who teaches which course(s)
CREATE TABLE teaches(
     crn              int
     , instructorid   int
     , PRIMARY KEY (crn, instructorid)
     , FOREIGN KEY (crn) REFERENCES classes(crn)
       ON DELETE CASCADE ON UPDATE CASCADE
     , FOREIGN KEY (instructorid) REFERENCES instructors(id)
       ON DELETE CASCADE ON UPDATE CASCADE
) ;

-- When classes meet
CREATE TABLE classmeetings(
     crn        int
     , dayofweek  varchar(10) -- ex: Monday, Tuesday
     , starttime  time        -- ex: time '14:30'
     , duration   int         -- in minutes, ex: 150
     , note       text
     , PRIMARY KEY (crn, dayofweek, starttime)
     , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE ON UPDATE CASCADE
) ;
```

```
-- When classes have office hours
CREATE TABLE officehours(
     crn          int
     , dayofweek    varchar(10)  -- ex: Monday, Tuesday
     , starttime    time         -- ex: time '18:00'
     , endtime      time         -- ex: time '19:30'
     , PRIMARY KEY (crn, dayofweek, starttime)
     , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE
) ;

-- When classes have exams
CREATE TABLE exams(
     crn        int
     , examname   varchar(40) -- ex: Exam 1, Exam 2, Final Exam
     , examdate   date        -- ex: date '2020/11/02'
     , pointvalue int         -- ex: 12, 20
     , starttime  time        -- ex: time '14:30'
     , duration   int         -- in minutes, ex: 130
     , note       text
     , PRIMARY KEY (crn, examname)
     , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE ON UPDATE CASCADE
) ;

-- All sites that are used for different courses.
CREATE TABLE sites(
     , sitename     varchar(40) PRIMARY KEY --ex: slack, discord, teams, submitty
     , bestbrowser  varchar(40)  --ex: firefox, chrome
     , generalurl   varchar(100) --ex: webex.com
) ;

-- Which sites are used for which courses, example resourcetypes are
-- discussions, hw, videos, exams, coursenotes, meetings, officehours
CREATE TABLE resourcesites(
     rid              int  PRIMARY KEY
     , crn            int
     , resourcetype   varchar(100) -- ex: see above.
     , sitename       varchar(40)
     , resourceurl    varchar(100)
     , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE
     , FOREIGN KEY (sitename) REFERENCES sites(sitename) ON UPDATE CASCADE
) ;

-- All students in the database
CREATE TABLE students (
     studentid      int   PRIMARY KEY
     , email        varchar(100)
     , firstname    varchar(100)
     , lastname     varchar(100)
) ;

-- Who is enrolled in which class.
CREATE TABLE enrollment (
     crn             int
     , studentid     int
     , PRIMARY KEY (crn, studentid)
     , FOREIGN KEY (crn) REFERENCES classes(crn) ON DELETE CASCADE
     , FOREIGN KEY (studentid) REFERENCES students(studentid) ON DELETE CASCADE

) ;
```

Here is a shorthand of the schema:

```
courses(classcode, coursename, credits, department)
classes(crn, classcode, semester, year, sectionno)
instructors(id, name, email, onlineroom, note)
teaches(crn, instructorid)
classmeetings(crn, dayofweek, starttime, duration, note)
officehours(crn, dayofweek, starttime, endtime)
exams(crn, examname, examdate, pointvalue, starttime, duration, note)
sites(sitename, bestbrowser, generalurl)
resourcesites(rid, crn, resourcetype, sitename, resourceurl)
students(studentid, email, firstname, lastname)
enrollment(crn, studentid)
```

Question 1 (10 points). Return the crn and coursename for all classes offered in Fall 2020 (semester, year) that use both Piazza and Submitty (sitename) as resources and have at least one class meeting on Tuesday (sitename).

Write your answer below:

Press TAB to indent. Press ESC to advance from answer area.

```
SELECT DISTINCT
    cls.crn,
    cs.coursename
FROM
    courses cs INNER JOIN classes cls
    ON cs.classcode = cls.classcode
    INNER JOIN classmeetings cm
    ON cls.crn = cm.crn
WHERE
    lower(cls.semester) LIKE '%fall%' AND
    cls.year = 2020 AND
    lower(cm.dayofweek) LIKE '%tuesday%' AND
    EXISTS (
        SELECT
            1
        FROM
            resourcesites rs
        WHERE
```

Clear | Use Most Recent Submission

Question 2 (12 points). Return the id and name of all instructors in the database who taught a class that has at least one class meeting on Tuesday (dayofweek) and have no office hours on Tuesday (dayofweek).

Write your answer below:

Press TAB to indent. Press ESC to advance from answer area.

```
WITH
    t1 AS (
        SELECT
            cm.crn
        FROM
            classmeetings cm
        WHERE
            lower(cm.dayofweek) LIKE '%tuesday%'
        EXCEPT
        SELECT
            oh.crn
        FROM
            officehours oh
        WHERE
            lower(oh.dayofweek) LIKE '%tuesday'
    )
SELECT
    ins.id,
```

Clear | Use Most Recent Submission

Question 3 (12 points). Return the crn and coursename for all classes offered in Fall or Spring (semester) of 2020 (year) that satisfy at least one of the following two conditions: either (1) the class is using at least three different sites as resourcesites, or (2) the class has exactly two instructors.

Write your answer below:

Press TAB to indent. Press ESC to advance from answer area.

```
SELECT
    cls.crn,
    css.coursename
FROM
    courses css INNER JOIN classes cls
    ON css.classcode = cls.classcode
WHERE
    (
        lower(cls.semester) LIKE '%fall%' OR
        lower(cls.semester) LIKE '%spring%'
    ) AND
    cls.year = 2020 AND
    cls.crn IN (
        SELECT
            rs.crn
        FROM
            resourcesites rs
        GROUP BY
```

Clear | Use Most Recent Submission

Question 4 (10 points). For each professor in the database who is teaching at least one class, return the id and name of the professor, the number of courses they are teaching and the number of different sites they are using for discussions (resources.resourcetype).

Write your answer below:

Press TAB to indent. Press ESC to advance from answer area.

```
SELECT
    ins.id,
    ins.name,
    (
        SELECT COUNT(*)
        FROM teaches ts1
        WHERE ts1.instructorid = ins.id
    ) AS numcourses,
    (
        SELECT COUNT(*)
        FROM resourcesites rs
        WHERE
            rs.crn IN (
                SELECT ts2.crn FROM teaches ts2 WHERE ts2.instructorid = ins.id
            ) AND
            lower(rs.resourcetype) LIKE '%discussions%'
    ) AS numsites
FROM
```

Clear | Use Most Recent Submission

Question 5 (10 points). Return the crn, dayofweek and starttime of all office hours for a class with only one exam.

Write your answer below:

Press TAB to indent. Press ESC to advance from answer area.

```
SELECT
    oh.crn,
    oh.dayofweek,
    oh.starttime
FROM
    officehours oh
WHERE
    oh.crn IN (
        SELECT
            ex.crn
        FROM
            exams ex
        GROUP BY
            ex.crn
        HAVING
            COUNT(*) = 1
    );
```

Clear | Use Most Recent Submission

Question 6 (14 points). Find the dayofweek and starttime in Fall 2020 (semester, year) that has the highest number of students on Webex (sitename) at the same time in a class meeting (assuming all enrolled students join at the start time of their respective classes). Return the dayofweek, starttime and the total number of students.

It is possible to solve this question with a single SQL query (including subqueries). However, you can break this problem into a multi-step code, using multiple SQL queries, creating temporary tables and dropping them if you wish.

Write your answer below:

Press TAB to indent. Press ESC to advance from answer area.

```
WITH
    t1 AS (
        SELECT
            en.crn,
            COUNT(*) as studentnum
        FROM
            enrollment en
        WHERE
            en.crn IN (
                SELECT
                    cls.crn
                FROM
                    classes cls
                WHERE
                    lower(cls.semester) LIKE '%fall%' AND
                    cls.year = 2020 AND
                    EXISTS (
                        SELECT
```

Clear | Use Most Recent Submission

Question 7 (10 points). Enroll Baby Yoda (firstname='Baby' and lastname='Yoda') in section 1 (sectionno) of all classes in Spring 2021 (semester, year) that are using Signal (sitename) as a resourcesite.

Write your answer below:

Press TAB to indent. Press ESC to advance from answer area.

```
INSERT INTO
    enrollment
SELECT
    cls.crn
    std.studentid
FROM
    classes cls,
    students std
WHERE
    cls.sectionno = 1 AND
    lower(cls.semester) LIKE '%spring%' AND
    cls.year = 2021
    AND EXISTS (
        SELECT
            1
        FROM
            resourcesites rs
        WHERE
```

Clear | Use Most Recent Submission

Question 8 (12 points). Write a single transaction block (BEGIN;/COMMIT;) to reprimand professors for using TikTok (sitename) as a resourcesite in their classes.

To accomplish this, update the note attribute for the instructor to string 'inappropriate resource: TikTok' and cancel any classes using TikTok by removing all tuples for these classes from any table in the database.

You do not need to create any temporary tables to achieve this, but if you wish, you can create such a table to aid you in this query. Drop the table at the end of your transaction block.

Write your answer below:

Press TAB to indent. Press ESC to advance from answer area.

```
BEGIN;

UPDATE
    instructors
SET note='inappropriate resource: TikTok'
WHERE
    id IN (
        WITH
        t1 AS (
                SELECT
                    rs.crn
                FROM
                    resourcesites rs
                WHERE
                    lower(rs.sitename) LIKE '%tiktok%'
            )
        SELECT
            ts.instructorid
```

Clear | Use Most Recent Submission

Question 9 (10 points). You are given the following two tables with no tuples initially.

```
CREATE TABLE a(val INT) ;
CREATE TABLE b(val INT) ;

CREATE FUNCTION atrgf () RETURNS trigger AS $$
    BEGIN
        IF NEW.val > 3 THEN
            INSERT INTO b SELECT sum(val) FROM a;
   END IF ;
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER atrg BEFORE INSERT ON a
    FOR EACH ROW EXECUTE FUNCTION atrgf();

CREATE FUNCTION btrgf () RETURNS trigger AS $$
    BEGIN
        IF NEW.val - OLD.val > 5 THEN
           INSERT INTO a VALUES(NEW.val);
   END IF ;
        RETURN NEW;
    END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER btrg AFTER UPDATE ON b
    FOR EACH ROW EXECUTE FUNCTION btrgf();
```

Check all tuples in the database after the following transaction executes:

```
BEGIN ;
INSERT INTO a VALUES(4) ;
INSERT INTO a VALUES(2) ;
INSERT INTO a VALUES(5) ;
UPDATE b SET val = 14 WHERE val = 4 ;
INSERT INTO a VALUES(8) ;
UPDATE b SET val = val*10 WHERE val < 10 ;
COMMIT;
```

You may select many:

- [x] Table a (4)
- [x] Table a (2)
- [x] Table a (5)
- [x] Table a (8)
- [ ] Table a (0)
- [ ] Table a (40)
- [ ] Table a (20)
- [x] Table a (60)
- [ ] Table a (80)
- [ ] Table a (110)
- [ ] Table b (0)
- [x] Table b (NULL)
- [ ] Table b (4)
- [ ] Table b (2)
- [ ] Table b (6)
- [x] Table b (11)
- [x] Table b (19)
- [ ] Table b (20)
- [x] Table b (60)
- [ ] Table b (79)
- [ ] Table b (80)
- [ ] Table b (110)

Clear | Use Most Recent Submission

By clicking "Submit" you are confirming that you have read, understand, and agree to follow the Academic Integrity Policy.

Submit

94 / 100 **Total**

Autograding Total ***(Without Hidden Points)***

10 / 10 **Autograding Total** ***(With Hidden Points)***

84 / 90 **TA / Instructor Grading Total**

10 / 10 **Question 1** ***(Graded by: Gulati)***
- [x] 0 Full Credit

11 / 12 **Question 2** ***(Graded by: Adali)***
- [x] -1 Incorrect select (NO DISTINCT, incorrect attributes)

11 / 12 **Question 3** ***(Graded by: Gulati)***
- [x] -1 Incorrect select (NO DISTINCT, incorrect attributes)
- [x] 0 distinct for both subqueries

9 / 10 **Question 4** ***(Graded by: Adali)***
- [x] -1 Missing distinct in count

10 / 10 **Question 5** ***(Graded by: Arrington)***
- [x] 0 Full Credit

14 / 14 **Question 6** ***(Graded by: Adali)***
- [x] 0 Full Credit

9 / 10 **Question 7** ***(Graded by: Gulati)***
- [x] -1 Missing distinct in select

10 / 12 **Question 8** ***(Graded by: Arrington)***
- [x] -1 Incorrect insert/update syntax (use of joins etc. in update)
- [x] -1 Incorrect missing selections in the delete condition (subquery)

Grade inquiries closed on ***11/27/2020 @ 11:59 PM EST***