Recall this schema, which we have used many times in class.

Relations

Student(<u>sID</u>, surName, firstName, campus, email, cgpa)

Course(<u>dept, cNum</u>, name, breadth)

Offering(<u>oID</u>, dept, cNum, term, instructor)

Took(<u>sID, oID</u>, grade)

Integrity constraints

Offering[dept, cNum] $\subseteq$ Course[dept, cNum]

Took[sID] $\subseteq$ Student[sID]

Took[oID] $\subseteq$ Offering[oID]

Question 1. [5 MARKS]

Part (a) [1 MARK]

According to the schema, can a student take the same course more than once? Yes No.

Part (b) [2 MARKS]

Consider this constraint:

$$Ploot(instructor1, instructor2, term) :=$$

$$\Pi_{O1.cNum1, O2.cNum2, O1.term} \sigma_{\substack{O1.instructor < O2.instructor \\ \wedge \\ O1.dept = O2.dept = `CSC' \\ \wedge \\ O1.cNum = O2.cNum \\ \wedge \\ O1.term = O2.term}} [(\rho_{O1} Offering) \times (\rho_{O2} Offering)]$$

$$\sigma_{\substack{P1.instructor1 = P2.instructor1 \\ \wedge \\ P1.instructor2 = P2.instructor2 \\ \wedge \\ P1.term \neq P2.term}} [(\rho_{P1} Ploot) \times (\rho_{P2} Ploot)] = \emptyset$$

Define an instance of Offering that violates the constraint.

oID	dept	cNum	term	instructor

Part (c) [2 MARKS]

Write the following constraint using relational algebra: No CSC course may count towards the breadth requirement (that is, have breadth = True) unless it is a 100-level course.

Question 2. [8 MARKS]

Write a query in relational algebra to find the sID of every student who has taken each CSC course that has ever been offered. We started the first step for you.

– This student has taken this CSC course.
$Taken(sID, cNum) :=$

Question 3. [5 MARKS]

Suppose we want to find the cNum of the first CSC course that was ever taught, that is, the one with the minimum value for term. If there was a tie, we want to report the cNums of all the tied courses.

The following query attempts to solve this. It is syntactically correct, but doesn't always produce the right answer.

$$CSCterms(cNum, term) := \Pi_{cNum,term}\sigma_{dept='CSC'}\, Offering$$

$$NotFirst(cNum) := \Pi_{C2.cNum}\sigma_{C1.term<C2.term}[(\rho_{C1}CSCterms) \times (\rho_{C2}CSCterms)]$$

$$Answer(cNum) := (\Pi_{cNum}CSCterms) - NotFirst$$

Part (a) [2 MARKS]

Define an instance of Offering with 4 rows on which the query gives the wrong answer. For simplicity, use integers to represent the terms.

oID	dept	cNum	term	instructor

What relation *should be* produced in this case? cNum

What relation *is* produced in this case? cNum

Part (b) [3 MARKS]

On the query above, make the smallest change(s) that will correct it.

Question 4. [7 MARKS]

Consider this new schema for a music industry database:

Relations

Musician(<u>mID</u>, surName, firstName, birthdate)

Album(<u>aID</u>, title, mID, year)

RecordCompany(<u>cID</u>, name, president)

Produced(<u>aID, cID</u>)

Integrity constraints

Album[mID] $\subseteq$ Musician[mID]

Produced[aID] $\subseteq$ Album[aID]

Produced[cID] $\subseteq$ RecordCompany[cID]

Part (a) [1 MARK]

Suppose relation *Album* has 1,000 tuples. How many tuples could *Produced* have? Circle all that apply:

0 1 821 1,000 2,500

Part (b) [2 MARKS]

Which of the following constraints are enforced by the schema? Circle Yes or No for each.

Every musician has at least one album.	Yes	No
Every album has at least one record company that produced it.	Yes	No
Every record company has produced at least one album.	Yes	No
Every album has at most one record company that produced it.	Yes	No
Every album has at most one musician.	Yes	No

Part (c) [2 MARKS]

Suppose every album has one genre, and we add an attribute called genre to the Produced relation to keep track of it. Write a constraint in relational algebra to restrict the value of genre to either hip hop, pop, or country.

Part (d) [2 MARKS]

Why is it a bad idea to store this genre information in the Produced relation?

Question 5. [8 MARKS]

For this question, you will write SQL queries using a simplified version of the Instagram schema from Assignment 1. (We removed unnecessary pieces and renamed table `User` to `Account` because `user` is a reserved word in SQL.)

Relations

Account(<u>uID</u>, name, website, phone)

Follows(<u>follower, followed</u>)

Post(<u>pid</u>, uid, location, caption)

Hashtag(<u>pid, tag</u>)

Integrity constraints

Follows[follower] $\subseteq$ Account[uID]

Follows[followed] $\subseteq$ Account[uID]

Post[uID] $\subseteq$ Account[uID]

Hashtag[pID] $\subseteq$ Post[pID]

Part (a) [3 MARKS]

Write a query in SQL to find users that have more than 100 posts where the location is Toronto. For each one, give their uID and the total number of their Toronto posts. Show the users in non-increasing order by their number of Toronto posts.

Part (b) [1 MARK]

Write an SQL query to find the pid of posts that have no hashtags.

The following query is supposed to print the name, uID and number of followers for users who have fewer than three followers. It runs but does not always give the correct output.

```
SELECT name, uid, count(follower) AS num_followers

FROM Follows, Account

WHERE Follows.followed = Account.uid

GROUP BY Account.uid

HAVING count(follower) < 3;
```

Part (c) [2 MARKS]

Suppose that Follows and Account have these values. What will be the output of the query?

```
 follower | followed
----------+----------
        2 |        1
        3 |        2
        2 |        3
        4 |        2
        1 |        3
        4 |        3
```

```
 uid | name  | website
-----+-------+----------
   1 | user1 | website1
   2 | user2 | website2
   3 | user3 | website3
   4 | user4 | website4
```

Part (d) [1 MARK]

Generalizing to any dataset, explain what is wrong with the output of this query.

Part (e) [1 MARK]

Fix the query by making the smallest change that you can. Write your corrections directly on the query text above.

Question 6. [8 MARKS]

Suppose we have these tables:

```
Follows:
     a     |       b
-----------+----------------
 sina      | kanyewest
 sina      | RonConwayFacts
 diane     | LilaFontes
 diane     | swcarpentry
 diane     | mfeathers
 diane     | sina
 michelle  | sina
 michelle  | diane
 michelle  | Jeff
(9 rows)
```

```
Profile:
    id     |    name    | location
-----------+------------+----------
 alan      | catman     | Ottawa
 sina      | superman   |
 diane     | superwoman | Toronto
 michelle  | rockstar   | Montreal
(4 rows)
```

```
Tweets:
id   | userid |    content
-----+--------+----------------
 123 | alan   | hellow twitter
 125 | alan   | bye twitter
 126 | alan   | hellow twitter
 128 | alan   | bye twitter
 476 | sina   | hellow twitter
 553 | diane  | hellow twitter
(6 rows)
```

Show the result of running each of the following queries. If a table is produced, include the column names. If the query generates an error, explain.

```
SELECT count(*)
FROM Profile LEFT JOIN Follows
    ON a = id;
```

```
SELECT P.id, count(T.content) AS number
FROM Profile P JOIN Tweets t
    ON T.userid = P.id
    AND P.location = 'Toronto'
GROUP BY(p.id);
```

Here are the tables again, for easy reference:

Follows:

a	b
sina	kanyewest
sina	RonConwayFacts
diane	LilaFontes
diane	swcarpentry
diane	mfeathers
diane	sina
michelle	sina
michelle	diane
michelle	Jeff

(9 rows)

Profile:

id	name	location
alan	catman	Ottawa
sina	superman	
diane	superwoman	Toronto
michelle	rockstar	Montreal

(4 rows)

Tweets:

id	userid	content
123	alan	hellow twitter
125	alan	bye twitter
126	alan	hellow twitter
128	alan	bye twitter
476	sina	hellow twitter
553	diane	hellow twitter

(6 rows)

Show the result of running each of the following queries. If a table is produced, include the column names. If the query generates an error, explain.

```
SELECT id, count(b) AS followers
FROM Profile JOIN Follows
    ON a = id;
```

```
SELECT location
FROM Follows, Profile
WHERE id = a AND b = 'sina';
```