Question 1. [12 marks]

Consider this schema for Twitter, a social media platform where users post messages called "tweets".

Relations

User(userID, name, email)
A Twitter user.

Tweet(tweetID, userID, content, day)
The user with *userID* made a tweet containing *content* on *day*.

Follows(a, b)
User *a* follows user *b* on Twitter, which means that *a* has subscribed to *b*'s tweets.

Likes(who, what, d)
User *who* liked tweet *what* on day *d*.

Integrity Constraints

Tweet[userID] $\subseteq$ User[userID]

Follows[a] $\subseteq$ User[userID]

Follows[b] $\subseteq$ User[userID]

Likes[who] $\subseteq$ User[userID]

Likes[what] $\subseteq$ Tweet[tweetID]

Part (a) [2 marks]

Does the schema enforce this constraint: A user cannot like the same tweet twice? Circle one:

Solution:

$\boxed{\text{Yes}}$ No.

Since who and what together form a key for relation Likes, the same user-tweetID combination cannot occur more than once in relation Likes.

Part (b) [2 marks]

Does the schema enforce this constraint: You can't follow yourself? Circle one:

Solution:

Yes $\boxed{\text{No.}}$

This would enforce it:

$$\sigma_{a=b} Follows = \emptyset$$

Part (c) [1 mark]

Suppose relation *Likes* has 300 tuples. How many tuples could *Users* have? Circle all that apply:

Solution:

0 $\boxed{1}$ $\boxed{256}$ $\boxed{300}$ $\boxed{912}$

Part (d) [2 MARKS]

Suppose relation *User* has m tuples and relation *Tweet* has n tuples. What is the maximum number of tuples that relation *Likes* can have?

Solution:

$m \times n$

Explain how the schema imposes this limit:

Solution:

Each user-TweetID combination can only occur once, and the total possible number of combinations is $m \times n$.

Part (e) [3 MARKS]

Suppose we add the following constraint: Likes[who] $\subseteq$ Follows[b]. Make the smallest possible non-empty instance of relations *Likes* and *Follows* that violates this constraint:

Solution:

Likes:

who	what	d
miriam	T23	Jan 1, 2016

Follows:

a	b
drizzy	dianeh

Express this constraint in English:

Solution: You can't like any tweets unless you have follower(s).

What kind of constraint is it? Circle all that apply:

Solution:

$\boxed{\text{referential integrity constraint}}$ foreign-key constraint $\boxed{\text{integrity constraint}}$

Part (f) [2 marks]

Which of the following queries can be expressed using the same form of relational algebra that we used in class and on Assignment 1, that is: the operators $\Pi, \sigma, \bowtie, \bowtie_{condition}, \times, \cap, \cup, -, \rho$ and assignment? Circle all that apply.

Solution:

1. **Yes:** Find everyone who follows 6 or more people who have never liked a tweet.
2. **No:** Let's say user X is "upstream" of Y if either X follows Y, or X follows someone else who is upstream of Y. Find every user who is upstream of the person with userID 'Oprah'.
3. **Yes:** Find the second last tweet from the person with userID 'Oprah'.
4. **No:** Find the user who follows the most people.
5. **Yes:** Find the user who made the first tweet.

Question 2. [8 marks]

Here is the schema from Assignment 1. A few attributes and relations have been omitted for simplicity.

Relations

Product(<u>DIN</u>, manufacturer, name, form, schedule)
A tuple in this relation represents a drug product.

Price(<u>DIN</u>, price)
The price of a drug product.

Prescription(<u>RxID</u>, date, patient, drug, doctor)
A prescription for *drug* was written on *date* for *patient* by *doctor*. Attribute *patient* is the patient's OHIP number.

Filled(<u>RxID</u>, date, pharmacist)
Prescription *RxID* was filled by *pharmacist* on *date*. Attribute *pharmacist* is the pharmacist's OCP number.

Integrity constraints

Price[DIN] $\subseteq$ Product[DIN]

Prescription[drug] $\subseteq$ Product[DIN]

Filled[RxID] $\subseteq$ Prescription[RxID]

$\Pi_{\text{schedule}}\text{Product} \subseteq$ {"prescription", "narcotic", "OTC"}

Write a query in relational algebra to report the OHIP number of every patient who has had a prescription that (a) was for the most expensive drug product (or a product tied for most expensive) and (b) they never filled.

Use only the basic operators $\Pi, \sigma, \bowtie, \times, \cap, \cup, -, \rho$, and assignment.

Solution:

– DIN of a drug that is not the most expensive drug.

$NotMax(DIN) := \Pi_{P1.price<P2.price}(\rho_{P1}Price \times \rho_{P2}Price)$

– DIN of a drug that IS the most expensive drug.

$Max(DIN) := (\Pi_{DIN}Price) - (\Pi_{DIN}NotMax)$

– This prescription for this patient is for the most expensive drug, or for a drug that is
– tied for most expensive.

$MaxPrescription(RxID, patient) := \Pi_{RxID,patient}(\sigma_{drug=DIN}Prescription \times Max)$

– This patient has had a prescription for the/a most expensive drug that they never filled.

$Answer(patient) := \Pi_{patient}[(\Pi_{RxID}MaxPrescrtiption) - (\Pi_{RxID}Filled)]$

Continue your answer here if more space is needed.

Question 3. [6 MARKS]

Suppose we have implemented the Twitter schema from Question 1 in SQL, and the tables currently contain the following:

```
Profile:
 userid |     name     |    email
--------+--------------+--------------
 adele  | Adele Adkins |
 drizzy | Drake        |
 potus  | Barack Obama | potus@gov.us
 rjm    | Renee Miller | rjm@cs
```

```
Follows:
   a    |   b
--------+--------
 potus  | drizzy
 drizzy | rjm
```

```
Tweet:
 tweetid | userid | content  |    day
---------+--------+----------+------------
      15 | adele  | Hello    | 2016-10-16
      61 | adele  | It's me  | 2016-10-16
      33 | potus  | 6 weeks  | 2016-10-11
      28 | rjm    | in the 6 | 2016-10-10
```

```
Likes:
  who   | what |     d
--------+------+------------
 drizzy |   61 | 2016-10-18
 rjm    |   33 | 2016-10-17
 drizzy |   15 | 2016-10-16
 potus  |   15 | 2016-10-16
```

Show the output of each of the queries below. If a query will not run successfully, write "Illegal".

Solutions

```
----- (1)
SELECT who
FROM Likes JOIN Tweet ON what = tweetID
WHERE userID = 'adele';

  who
--------
 drizzy
 drizzy
 potus
(3 rows)


----- (2)
SELECT userID, count(tweetID), count(day)
FROM Tweet
GROUP BY userID;

 userid | count | count
--------+-------+-------
 rjm    |     1 |     1
 adele  |     2 |     2
 potus  |     1 |     1
(3 rows)
```

```
----- (3)
SELECT count(*) AS num1, count(email) AS num2
FROM Profile;

 num1 | num2
------+------
    4 |    2
(1 row)


----- (4)
SELECT name, content
FROM Profile NATURAL RIGHT JOIN Tweet;

     name     | content
--------------+----------
 Adele Adkins | Hello
 Adele Adkins | It's me
 Barack Obama | 6 weeks
 Renee Miller | in the 6
(4 rows)


----- (5)
(SELECT a AS userID
FROM Follows
WHERE b = 'drizzy')
   UNION ALL
(SELECT userID
FROM Tweet

 userid
--------
 potus
 potus
 rjm
(3 rows)


----- (6)
SELECT tweetID, count(who)
FROM Tweet, Likes
WHERE tweetID = what;

psql:questions.sql:33: ERROR:  column "tweet.tweetid" must appear in the GROUP BY clause or be use
LINE 1: SELECT tweetID, count(who)
```

Question 4. [4 MARKS]

Write a query to find the userID of everyone who has made more than one Tweet. Ensure that it would work on any instance of the database, not simply the one above.

Solution:
This approach would work just as well if we were restricting to people who have made more than 100 Tweets:

```
SELECT Profile.userid
FROM Profile JOIN Tweet ON Profile.userID = Tweet.userID
GROUP BY Profile.userID
HAVING count(*) > 1;
```

This qpproach would work for the question, but would not scale well to larger cutoffs!

```
SELECT DISTINCT Profile.userid
FROM Profile, Tweet t1, Tweet t2
WHERE t1.userID = t2.userID and t1.tweetID <> t2.tweetID and Profile.userID = T1.userID;
```