Question 1. [12 marks]

Twitter is a social media platform where users post messages called “tweets”. Users can “follow” other users, and can “like” a tweet. Consider this relational schema for Twitter data. Keys are underlined.

As this question considers relational algebra, assume all relations are sets containing no nulls.

\[
\begin{align*}
\text{User}(&\text{userID}, \text{name}, \text{email}) \\
\text{A Twitter user.} \\
\text{Tweet}(\text{tweetID}, &\text{userID}, \text{content}, \text{date}) \\
&\text{The user with userID made a tweet containing content on date.} \\
\text{Follows}(a, &b) \\
&\text{User a follows user b on Twitter, which means that} \\
&a \text{ has subscribed to b's tweets.} \\
\text{Likes}(who, &\text{tweetID, userID}) \\
&\text{User who liked tweet tweetID, userID.}
\end{align*}
\]

Part (a) [2 marks]

Does the scheme enforce this constraint: a user cannot like his or her own tweet? Circle one:

- Yes
- No.

This would enforce it: \(\sigma_{\text{who} = \text{userID}} \text{Likes} = \emptyset\)

Part (b) [2 marks]

Does the schema enforce this constraint: every userID in Tweet must have a name and email (recorded in User)? Circle one:

- Yes
- No.

Through fk from Tweet to User

Part (c) [2 marks]

Suppose relation Tweet has 100 tuples and \(\Pi_{\text{tweetID}}(\text{Tweet})\) has 10 tuples. How many tuples could Users have? Circle all that apply:

- 0
- 1
- 100
- 10,000
- 100,000
Part (d)  [4 marks]
Which of the following pairs of queries are equivalent? Circle each pair that returns the same results on all database instances.

1. $\Pi_{tweetID}(Likes \bowtie Tweet) = \Pi_{tweetID}(Tweet)$
   
   Solution: no - right side can have more values

2. $User = User \times User$
   
   Solution: no - right side has more attributes and tuples

3. $\Pi_{userID}(\sigma_{tweetID=1}(Tweet)) = \Pi_{userID}(\sigma_{tweetID=1}(Likes))$
   
   Solution: no - tweetID 1 may not be liked among other reasons

4. $\Pi_{name}(User \bowtie Tweet \bowtie Likes) = \Pi_{name}(User \bowtie Tweet \bowtie Likes \bowtie Follows)$
   
   Solution: yes (because it is the set of names returned, multiplicities don’t matter)

Part (e)  [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 assignment, and the operators $\Pi, \sigma, \bowtie_{\text{condition}}, \times, \cap, \cup, -, \rho$? Circle all that apply.

1. Find all users who have never tweeted.
   
   Solution: yes

2. Find dates on which every tweet that was posted was liked by at least two users.
   
   Solution: yes

3. Find the email of the most popular user (the user whose tweets are liked by the most people).
   
   Solution: no

4. Find the first date on which a tweet was made.
   
   Solution: yes

5. Find the users with the fewest followers.
   
   Solution: no
Question 2.  [8 marks]

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

### Relations

- **Product**(DIN, manufacturer, name, form, schedule)
  - A tuple in this relation represents a drug product.
- **Price**(DIN, price)
  - The price of a drug product.
- **Prescription**(RxID, 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**(RxID, date, pharmacist)
  - Prescription RxID was filled by pharmacist on date. Attribute pharmacist is the pharmacist’s OCP number.

### Integrity constraints

- Price[DIN] ⊆ Product[DIN]
- Prescription[drug] ⊆ Product[DIN]
- Filled[RxID] ⊆ Prescription[RxID]
- \( \Pi_{\text{schedule \ Product}} \subseteq \{ \text{“prescription”, “narcotic”, “OTC”} \} \)

For every drug product that has been prescribed, report its DIN, the date on which the first prescription for it was written, and the date on which the last prescription for it was written.

Use only the basic operators \( \Pi, \sigma, \bowtie, \times, \cap, \cup, \neg, \rho, \) and assignment.

**Solution:**

\[
P_1(\text{drug}_1, \text{date}_1) := \Pi_{\text{drug}, \text{date}}(\text{Prescription})
\]

\[
P_2(\text{drug}_2, \text{date}_2) := \Pi_{\text{drug}, \text{date}}(\text{Prescription})
\]

\[
\text{not first} := \Pi_{\text{drug}_1, \text{date}_1}(P_1 \bowtie_{\text{drug}_1=\text{drug}_2, \text{date}_1>\text{date}_2} P_2)
\]

\[
\text{not last} := \Pi_{\text{drug}_2, \text{date}_2}(P_1 \bowtie_{\text{drug}_1=\text{drug}_2, \text{date}_1>\text{date}_2} P_2)
\]

\[
\text{first} := P_1 - \text{not first}
\]

\[
\text{last} := P_2 - \text{not last}
\]

\[
\text{solution}(\text{drug}, \text{first}, \text{last}) := \Pi_{\text{drug}_1, \text{date}_1, \text{date}_2, \text{first} \bowtie_{\text{drug}_1=\text{drug}_2} \text{last}}
\]
Question 3.  [6 marks]

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

<table>
<thead>
<tr>
<th>User:</th>
<th>Follows:</th>
</tr>
</thead>
<tbody>
<tr>
<td>userid</td>
<td>name</td>
</tr>
<tr>
<td>--------+--------------------------+--------------</td>
<td></td>
</tr>
<tr>
<td>adele</td>
<td>Adele Laurie Blue Adkins</td>
</tr>
<tr>
<td>drizzy</td>
<td>Drake</td>
</tr>
<tr>
<td>potus</td>
<td>Barack Obama</td>
</tr>
<tr>
<td>rjm</td>
<td>Renee Miller</td>
</tr>
</tbody>
</table>

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

| Tweet: |
|--------+--------------------------+-------------------+----------+-------------+------------|
| tweetid | userid | content | date | who | tweetid | userid |
| --------+--------+----------+------------+--------+---------+---------|
| 15 | adele | Hello | 2016-10-16 | drizzy | 15 | adele |
| 61 | adele | It’s me | 2016-10-16 | drizzy | 61 | adele |
| 33 | potus | 6 weeks | 2016-10-11 | potus | 33 | potus |
| 28 | rjm | in the 6 | 2016-10-10 | potus | 61 | adele |

Show the output of each of the following queries.

Solutions

----- (1)
SELECT who
FROM Likes natural full join Tweet
WHERE tweetID < 30;

who
----------
drizzy
NULL
(2 rows)

----- (2)
SELECT t.userID, count(*), max(date)
FROM Tweet t, Likes l
WHERE t.userID = l.who
GROUP BY t.userID;

userid | count | max
--------+-------+----------
potus | 2 | 2016-10-11
(1 row)
----- (3)
SELECT count(*)
FROM User, Likes;

    count
-------
      16
(1 row)

----- (4)
SELECT DISTINCT date
FROM User NATURAL JOIN Tweet;

date
--------
2016-10-16
2016-10-10
2016-10-11
(3 rows)

----- (5)
SELECT T.TweetID, max(date)
FROM Tweet T, Likes L
WHERE T.userid = L.userid
GROUP BY T.TweetID
HAVING count(*) < 1;

tweetid | max
---------+-----
(0 rows)

----- (6)
(SELECT userID
FROM Likes)
    EXCEPT ALL
(SELECT userID
FROM Tweet);

    userid
-------
   adele
(1 row)
Question 4. [4 marks]

Write a query to find the name of users who have only liked tweets posted on the same date that they have made a tweet. Ensure that your query would work on any instance of the database, not simply the one above.

Solution:

Example

CREATE VIEW goodusers as ((SELECT userid from likes)
 except
 (SELECT u.userid
  FROM likes l, tweet t, user u
  WHERE who = u.userid and l.tweetid = t.tweetid and date not in
   (select date from tweet userTweet where userTweet.userid =
    who)));

SELECT name
FROM user U, goodusers G
WHERE u.userid = g.userid;