Database Systems — CSci 4380
Midterm Exam #2
October 31, 2019

```
 .-.
(o o) boo!
| O \
 \   \
  `~~~'
```

SOLUTIONS

Question 1 (42 points). Write the following queries using SQL using the data model below. **In each query in this question, do not use any subqueries (in select/from/where/having clauses) and do not use WITH statements. Use only SELECT FROM WHERE GROUP BY HAVING blocks and SET operations. You can use INNER/OUTER JOIN statements in the FROM clause. If you use subqueries, you will lose one point in per query.**

Users(userid, email, name, createdon,displayname, description, url, city, country)
Posts(postid, postdate, posttime, posttext, media, userid)
Likes(postid, userid, dateliked)
PostHashtags(postid, hashtag, rank)
Comments(commentid, postid, userid, commenttext, commentdate, commenttime, replyto_commentid)
Follows(userid, followed_userid, followdate)
Bookmarks(userid, postid, bookmarkdate)

(a) (10 points) Return the id and text of all posts in which the user who created the post also liked or commented on his/her own post.

Solutions:

```
SELECT
   p.postid
   , p.posttext
FROM
   posts p
   , likes l
WHERE
   p.postid = l.postid
   and p.userid = l.userid
UNION
SELECT
   p.postid
   , p.posttext
FROM
   posts p
   , comments c
WHERE
   p.postid = c.postid
   and p.userid = c.userid

or in a grossly inefficient query

SELECT DISTINCT
   p.postid
   , p.posttext
FROM
   posts p
   , likes l
   , comments c
WHERE
   (p.postid = l.postid
```

```
    and p.userid = l.userid)
   or (p.postid = c.postid
       and p.userid = c.userid)
```

(b) (10 points) Return the id and text of all posts that contain the hashtag `'#sismanlives'` and have received some comments containing words `'yacs'` and `'schedule'`.

Solutions:

```
SELECT DISTINCT
   p.postid
   , p.posttext
FROM
   posts p
   , posthashtags ph
   , comments c
WHERE
   p.postid = ph.postid
   AND ph.hashtag = '#sismanlives'
   AND p.postid = c.postid
   AND c.commenttext like '%yacs%'
   AND c.commenttext like '%schedule%';
```

(c) (10 points) Return all pairs of userids of users who liked the same post within 10 days of each other for at least 50 different posts.

Solutions:

```
SELECT
   l1.userid, l2.userid
FROM
   likes l1
   , likes l2
WHERE
   l1.userid <> l2.userid
   AND l1.postid = l2.postid
   AND l1.dateliked - l2.dateliked <= interval '10 days'
   AND l1.dateliked - l2.dateliked >= interval '0 days'
GROUP BY
   l1.userid, l2.userid
HAVING
   count(distinct l1.postid)>=50
```

(d) (12 points) Return all pairs of userids `u1,u2` of users such that `u1` does not follow `u2`, however `u1` has bookmarked at least 5 posts of `u2`.

Solutions:

```
SELECT
   b.userid as u1
   , p.userid as u2
FROM
   posts p
   , bookmarks b
WHERE
   p.postid = b.postid
```

```
   and p.userid <> b.userid --I will not penalize for missing this
                            --may be people can follow themselves
GROUP BY
   p.userid, b.userid
HAVING
   count(*)>= 5
EXCEPT
SELECT
   userid
   , followed_userid
FROM
   follows;
```

Question 2 (10*2=20 points). For each of the following, write a single SQL expression. You are allowed to use subqueries if needed.

(a) Delete all bookmarks for posts posted before 2012.

Solutions:

```
DELETE FROM
   bookmarks
WHERE
   postid in
        (SELECT postid
         FROM posts
         WHERE extract(year from postdate)<=2012);
```

(b) Update the tuples in the bookmarks table with a null value for `bookmarkdate`. For these tuples, set the `bookmarkdate` to the `postdate` of the post for that bookmark.

Solutions:

```
UPDATE
   bookmarks b
SET
   b.bookmarkdate = (SELECT p.postdate
                     FROM posts p
                     WHERE p.postid=b.postid)
WHERE
   b.bookmarkdate IS NULL ;
```

Question 3 (16 points). Write a single SQL query (using any SQL construct) to find the influencers in the database. For each user, the (influence) score is calculated as the total number of likes and comments they have received for all their posts. Return the id, name and score of users who have the top 100 score values in the database.

Solutions:

```
WITH l as (
SELECT
   p.userid
   , count(l.postid) as numl
FROM
   posts p
   left join likes l
   on p.postid = l.postid
GROUP BY
   p.userid
),
SELECT
   u.userid
   , u.name
   , count(c.commentid)+l.numl as iscore
FROM
   posts p
   join l l on l.userid = p.userid
   join users u on u.userid = l.userid
   left join comments c on p.userid = c.userid
GROUP BY
   u.userid
   , l.numl
ORDER BY
   iscore desc
LIMIT 100;
```

Question 4 (10 points). For each user, find the total number of likes this user received for his/her posts in each year the user has created a post. Return the user id, year and number of likes. Note that the user may have no likes in a given year. You can use any SQL construct for this query as well as multi-step procedural SQL.

Solutions: Note: this was a Halloween trick. This is one of the easiest queries in the exam!

```
SELECT
   p.userid
   , extract(year from p.postdate)
   , count(l.userid) as numlikes
FROM
   posts p
   left join likes l on p.postid = l.postid
GROUP BY
   p.userid
   , extract(year from p.postdate)
```

Question 5 (12 points). You are given the following data definitions and table contents.

```
CREATE TABLE USERS (userid INT PRIMARY KEY, name VARCHAR(100)) ;
CREATE TABLE POSTS (postid INT PRIMARY KEY, userid INT
      , FOREIGN KEY (userid) REFERENCES users(userid) ON DELETE CASCADE ON UPDATE SET NULL) ;
CREATE TABLE LIKES (postid INT, userid INT, PRIMARY KEY (postid, userid)
     , FOREIGN KEY (postid) REFERENCES posts(postid) ON DELETE CASCADE
     , FOREIGN KEY (userid) REFERENCES users(userid) ON DELETE CASCADE) ;
```

USERS

id	name
1	Rick
2	Morty
3	Jaguar
4	Beth

POSTS

postid	userid
11	1
12	1
13	2
14	3
15	3

LIKES

postid	userid
11	2
11	3
13	1
14	4
15	4

```
CREATE FUNCTION dostuff(idvar int) RETURNS INT AS $$
BEGIN
   INSERT INTO users(id) VALUES(idvar);
   UPDATE likes SET userid = idvar WHERE postid<12;
   RETURN 1 ;
END ; $$ LANGUAGE plpgsql ;
```

For each operation below, describe which rows from which tables are changed/deleted and why (or why not). Assume each operation operates on the table contents listed above (hence each part is independent).

Solutions:

(a) `DELETE FROM users WHERE name = 'Rick' ;`

Delete (1,Rick) from users, cascades to delete of (11,1) and (12,1) from posts, cascades to delete of (11,2), (11,3) due to posts and (13,1) due to users.

(b) `UPDATE posts SET postid = 24 WHERE postid = 12;`

Update (12,1) to (24,1). Nothing else changed as 12 was not in LIKES.

(c) `UPDATE users SET userid = 9 WHERE name = 'Morty';`

Update (2, Morty) to (9, Morty). Cascades to Posts where (13,2) is set to (13, Null), cascades to LIKES for (11,2) but there is no action, and as a result the whole transaction fails. No changes are made.

(d) `SELECT dostuff(5);`

If we treat the two operations as part of a transaction: insert into users (5,null), then update (11,2) and (11,3) to both (11,5). However this is not allowed as (postid,userid) is primary key. The whole transaction fails and no changes are made.

If we treat them as independent operations, then insert will succeed but update will fail. This is actually the default behavior. So, I will accept both solutions with sufficient explanation.

Use this page for scratch work only. Do not share your solutions or any drafts of your solutions with anyone.

Data model to be used in Exam #2

This is a data model loosely based on data stored in Instagram. Note that a post in this model can only have a single media, photo or video which is stored only as a text value for simplicity.

```
create table users (
     userid        int  primary key
     , email       varchar(100)
     , name        varchar(100)
     , createdon   date
     , displayname varchar(100)
     , description varchar(255)
     , url         varchar(100)
     , city        varchar(100)
     , country     varchar(100)
) ;

create table posts (
     postid        int primary key
     , postdate    date
     , posttime    time
     , posttext    text
     , media       varchar(10)
     , userid      int
       --user who created the post
     , foreign key (userid)
            references users(userid)
) ;

create table likes (
     postid        int
     , userid      int
     , dateliked   date
     , primary key (postid, userid)
     , foreign key (postid)
             references posts(postid)
     , foreign key (userid)
             references users(userid)
) ;

create table posthashtags(
     postid   int
     , hashtag   varchar(100)
     , rank      int
       --first, second, etc. hashtag for post
     , primary key (postid, hashtag)
     , foreign key (postid)
             references posts(postid)
);
```

```
create table comments (
     commentid      int primary key
     , postid       int
       --post being commented on
     , userid       int
       --user who is commenting
     , commenttext  varchar(100)
     , commentdate  date
     , commenttime  time
     , replyto_commentid  int
       --comment replies to another comment
     , foreign key (postid)
             references posts(postid)
     , foreign key (userid)
             references users(userid)
     , foreign key (replyto_commentid)
             references comments(commentid)
) ;

create table follows (
     userid            int
       --user who follows
     , followed_userid int
       --user who is followed
     , followdate      date
       --date the follow started
     , primary key (userid, followed_userid)
     , foreign key (userid)
            references users(userid)
) ;

create table bookmarks (
     userid int
     , postid int
     , bookmarkdate date
     , primary key(postid, userid)
     , foreign key (postid)
            references posts(postid)
     , foreign key (userid)
            references users(userid)
) ;
```