Assignment 1B: Recursive Queries

Due date: February 9, 2020, 11:55pm IST

General Instructions

1. Please complete this assignment *individually*.
2. You will submit just 1 file: query.sql.
3. Use PostgreSQL 12 for your homework. See **this link** for instructions on how to download and install it on your OS. The .sql files are run automatically using the psql command using the \i option, so please ensure that there are no syntax errors in the file. *If we are unable to run your file, you get an automatic reduction to 0 marks.*

   To understand how to run many queries at once from text file, a dummy query file **example.sql** is available. To run **example.sql** in PostgreSQL, type the following command in the terminal:

   sudo -u postgres psql *dbname*

   `\i /address/to/example.sql`

   This command will run all the queries listed in example.sql at once.
4. The format of the file should be as follows. You can have a preamble where you may create views if you like (please note that no procedures are allowed (this has to be pure SQL), and correspondingly have a cleanup section where these views are removed. One line should identify the query number (note the two hyphens before and after the query number), followed by the actual, syntactically correct SQL query. Leave a blank line after each query.

   - -PREAMBLE- -

   OPTIONAL DEFINITIONS

   - -1- -

   SQL QUERY

   - -2- -

   SQL QUERY

   - -3- -

   SQL QUERY

   - -CLEANUP- -

   CLEANUP EVERYTHING YOU CREATED HERE
5. Some of the queries below require an 'ORDER BY' clause. If you made an error in this clause, your answer may be evaluated as incorrect, giving you zero marks for that query.
6. The submission will be done on Moodle. No changes are allowed in attribute names or table names.
7. If unspecified, order in ascending order by column 1, then column 2 .. etc. In case of any doubts please ask on Piazza. The instructors ordering will be final and no queries will be entertained on the same.
8. There are no NULL values in the dataset, so you need not worry about that.
9. There is no data provided for these queries (except in the examples shown later). For testing, make your own dataset. The .sql file that you submit however, should contain only queries
10. If any query asks you to output top x rows and you are getting y rows, output $min(x, y)$ rows.

1 Dataset

As Saul Goodman once said,

With the increasing number of "guys he knows", he wants to know some things about the guys in his network, but isn't easy considering the size of the network. With the platform's database described below, help him out.

1. In this assignment, you'll work with a synthetically generated dataset. The schema of the same is described below, but we won't share the actual dataset that will be used for evaluation.
2. The database will include following three tables and you should use only these tables while writing solution of the queries. You can create temporary views while handling any SQL query but you should include SQL queries for creating and deleting these temporary views at the starting and end of your SQL file respectively. Note - you don't have to define these tables in the submission file, these will already be present will evaluation.
   (a) userdetails

userid : integer	username : text	place : text

   (b) friendlist

userid1 : integer	userid2 : integer

   (c) block

userid1 : integer	userid2 : integer

3. Keys:
   (a) `userid` is primary key for `userdetails` table
   (b) `(userid1, userid2)` is primary key for `friendlist` table
   (c) `(userid1, userid2)` is primary key for `block` table
4. `a` and `b` are friends if `(a, b)` $\in$ `friendlist` or `(b, a)` $\in$ `friendlist`.
5. `a` and `b` have blocked each other if `(a, b)` $\in$ `block` or `(b, a)` $\in$ `block`.
6. If `(a, b)` $\in$ `friendlist`, then `(b, a)` $\notin$ `friendlist` and vice-versa.
7. If `(a, b)` $\in$ `block`, then `(b, a)` $\notin$ `block` and vice-versa.
8. If either of `(a, b)` or `(b, a)` $\in$ `friendlist`, then `(a, b)`, `(b, a)` $\notin$ `block` and vice-versa.
9. Finally, it may happen that there is a user that is present only in `userdetails`, that is, has no friends, hasn't blocked anyone and isn't blocked by anyone.

2 Queries

Social Graph is a graph that represents social relations between entities. In short, it is a model or representation of a social network, where the word graph has been taken from graph theory. The social graph has been referred to as "the global mapping of everybody and how they're related".

Consider the social graph G formed by the `friendlist` and `userdetails` tables. The nodes for this graph will be the users, and there will exist an edge between them iff they are "friends" of each other. Two users are friends if the unordered tuple `(user1, user2)` is present in `friendlist` table. In the queries that follow, "components" refer to the usual components in a graph.

Mathematically, $G = (V, E)$ where:

- $V = \{$`userdetails.userid`$\}$
- $E = \{(u, v) | (u, v) \in$ `friendlist` or $(v, u) \in$ `friendlist`$; u, v \in$ `userdetails.userid`$\}$

1. Find the total number of components in G. **Output column: count**
2. Give the number of users in each component of G. Order by ascending order of number of users. Since output will be a single column, ties don't matter. **Output column: count**
3. Give the user id of the user that has blocked the most number of people in the same component. In case of ties, list all of them and order by ascending order of user id. We DO NOT mean "component-wise maximas", instead, we "global maximas". To clarify the query further, define $b(a) = |\{u : (a, u) \in$ `block` or $(u, a) \in$ `block` and $C(a) = C(u)\}|$, where $C(a)$ denotes component of node a and $|.|$ denotes cardinality of the set. Define $B = \{b(a) : a \in$ `userdetails.userid`$\}$. You need to list all the users u such that $b(u) = max(B)$. Since count will be the same for all, you just need to order according to userid. **Output columns: userid, count**
4. Given two user ids **userid1** and **userid2**, find the length of shortest path from **userid1 to userid2** in the graph G. If they are not in the same component, output -1. If both are already friends, output 1. If the shortest path goes like: `userid1` $\rightarrow a_1 \rightarrow a_2 \rightarrow \cdots \rightarrow a_n \rightarrow$ `userid2`, output $n + 1$. **Output column: length. Take userid1 = 1558, userid2 = 2826.**
5. Just like Facebook, in our platform too, you can't send someone a friend request if you've blocked them. Find the maximum number of friend requests **user1** can send to users belonging to the same component as him/her. **Output column: count. Take user1 = 704**
6. We define "potential number of friends of a person A" as people who live in the same city as person A, are not in the same component as A, who aren't friends with A, and aren't blocked by A. Give the user ids of top 10 people with most number of potential friends. Sort in descending order of potential friends and resolve all ties by ascending order of user ids. **Output column: userid**
7. Saul might know a guy, who knows a guy, who knows another guy, but he might not be the person with highest number of third degree connections. Person A's third degree connections are people connected to him with shortest path length of 3 in graph G. Give the user ids of top 10 users with the most number of third degree connections. Sort in descending order of connections and resolve all ties by ascending order of user ids. **Output column: userid**
8. Given 3 userids A, B and C, give the number of paths that exist between A and C that also pass through B. Note that A $\rightarrow a_1 \rightarrow B \rightarrow a_2 \rightarrow$ C and A $\rightarrow a_1 \rightarrow B \rightarrow a_3 \rightarrow$ C constitute two distinct paths in graph G, that is, in path from A to C (via B), if even a single node is different (may be in A$\rightarrow$B part or in B$\rightarrow$C part), then it counts as a different path. All the vertices in the path $A \rightarrow C$ should be distinct, that is, a path like $A \rightarrow \cdots a_1 \rightarrow \cdots B \rightarrow \cdots a_1 \rightarrow \cdots C$ is not valid. Return -1 if they don't belong to the same component. **Output column: count. Take A = 3552, B = 321, C = 1436**
9. Find the number of paths in Graph G from user A to B such that adjacent people in the path aren't from the same city. Return -1 if they don't belong to the same component, else return number of paths. **Output column: count. Take A = 3552, B = 321**

10. Find the number of paths in Graph G from user A to B, `(A, B)` $\notin$ `block`, such that any person in the path shouldn't be blocked by any other person in that path. Return -1 if they don't belong to the same component, else return number of paths. **Output column: count. Take A = 3552, B = 321**

3 Expected Results

3.1 Userdetails table

```
 userid | username |   place
--------+----------+-----------
      1 | Annea    | Jhansi
      2 | Malisha  | Indore
      3 | Chris    | Jammu
      4 | Robert   | Jhansi
      5 | Zamaya   | Jodhpur
      6 | Tom      | Cuttack
      7 | Naman    | Delhi
      8 | Keiron   | Jodhpur
      9 | Annea    | Jodhpur
     10 | Sudhir   | Goa
(10 rows)
```

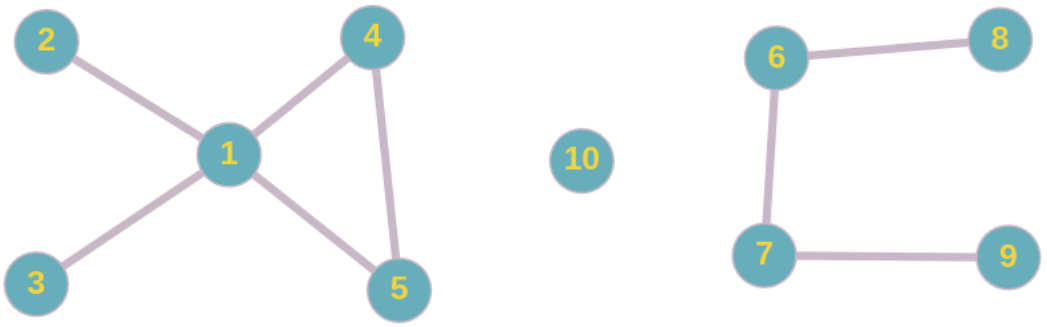
3.2 Friendlist table

```
 userid1 | userid2
---------+---------
       1 |       2
       1 |       3
       1 |       4
       1 |       5
       4 |       5
       6 |       8
       6 |       7
       7 |       9
(8 rows)
```

3.3 Block table

```
 userid1 | userid2
---------+---------
       2 |       3
       2 |       4
       2 |       6
       2 |       7
       2 |       8
       2 |       9
       8 |       7
(7 rows)
```

3.4 Graph



3.5 Results

Note: For some of the questions, we are using different values of user ids while giving the answer. The question has **different** values for the same. Take care while writing your solutions. You need to use the ids given in the question and any mistake in doing so can fetch you zero marks.

1.
```
 count
-------
     3
(1 row)
```

2.
```
 count
-------
     1
     4
     5
(2 rows)
```

3.
```
 userid | count
--------+-------
      2 |     2
(1 row)
```
Although user 2 has blocked 6 users, only 2 of them belong to the same component as 2.

4. The given answer is for userid1 = 2 and userid2 = 5.

```
 length
--------
      2
(1 row)
```

There are two paths: $2 \to 1 \to 4 \to 5$ and $2 \to 1 \to 5$, the second one being the shorter.

5. The given answer is for userid = 8.

```
 count
-------
     1
(1 row)
```

Nodes in 8's component are $\{6, 7, 8, 9\}$. 6 and 8 are already friends while 8 and 7 have blocked each other – can send only to 9.

6.
```
 userid
--------
      5
      8
      9
(3 rows)
```

8 and 9 are potential friends of 5, making 5's count two. 8 and 9 are in same component, keeping them at one each.

7.
```
 userid
--------
      8
      9
(2 rows)
```

No other user has any third degree connections.

8. The given answer is for $A = 2$, $B = 1$, $C = 5$.

```
 count
-------
     2
(1 row)
```

9. The given answer is for $A = 2$, $B = 5$.

```
 count
-------
     1
(1 row)
```

1 and 4 are from the same city, leading to rejection of that path.

10. The given answer is for $A = 2$, $B = 5$.

```
 count
-------
     1
(1 row)
```

2 has blocked 4, leading to rejection of that path.