Lab Overview

A lot of the time when querying databases, you will need to handle large amounts of data. The client has collected a lot of data and now wants to see what this data can tell us about their current LMS. In this lab, you are provided with two files containing Create table and Insert statements.

Rules

1. You must follow the rules of the honor code. The labs must be done in groups of exactly two people. No larger groups are allowed, and if you have extraordinary extenuating circumstances that force you to do the labs alone, you must obtain permission to do so from the course leader.
2. Presenting P+ assignments are optional for a higher grade if the given tasks are completed and passed.
3. This is a PSQL lab. No other programming languages, either embedded in the database or external to it, are allowed.
4. Please refrain from creating any functions since this lab is designed to assess query programming languages.
5. You are not allowed to hard-code anything except that which has been explicitly given to you in the problems. In particular, this means that constructs like limit 1 or similar artificial ways of reducing the output are forbidden.

Lab Presentation

Make sure you have the following ready to be presented to the TA:

- Your code should be up on the screen along with the terminal ready to execute the tasks.
- Motivations for how the solution for the requested queries are sensible. (Tasks may have several ways to solve them, but they have to be good enough for e.g. a client to accept.)

Before you start

Make sure you're saving your queries. You can use your preferred text editor or DataGrip by jetLab which has specific features for databases.
If you go with DataGrip you can use it for free with the student option.
Tip: *Take your time and make sure you've read the questions carefully as some are quite dense.*

OBS! The data from your previous lab **will not** be used for this or further labs. The reason for this is to formalize correct answers to the queries below, otherwise different databases would have different solutions, which could cause unfair correction.
See the postgres installation guide for instructions on moving files, removing, adding and changing your database.
NOTE! You are allowed to skip using the server and instead set up the database locally. In this case it is your responsibility that everything works as expected.

Querying the Database

Queries for the following questions are to be completed and should show the same result as the expected outcome. Lists and tables are expected to be alphabetical or numerical in order.

1. Display each post once with all of its tags and sort the posts alphabetically in terms of their title.
   ***Tip**: There are many ways to do this but one way is to use the PSQL function string_agg.*
   ***Note**: The order in which the tags appear does not matter*

Expected output format:

title	tags
A bouncing ball	#doginfluencer, #Dogadventures, #gymgirl, #digitalart, #explore, #gymrat, #happybirthday, #networking, #amazing, #picoftheday, #tattooart, #fitnessmodel, #fyp, #b2b, #yoga, #body, #yogainspiration, #productivity, #dogphotography
A bouncing kangaroo	#artgallery, #likeforlike, #dogsofinstagram, #dogmom, #socialmedia
About charity fundraiser	#modernart, #likeforlikes, #instatravel, #artistsoninstagram, #followforfollowback, #sweat, #homedecor, #cool, #photooftheday, #beautiful, #contemporaryart, #dogtraining, #doginfluencer, #fitnesslifestyle, #discount, #socialmedia, #instafashion, #fitnessmodel, #instalike
About fashion show	#love, #motivation
About fireworks display	#likeforlike, #like4follow, #happy
About interactive game	#happybirthday, #instadaily, #gymmotivation, #motivation, #cardio, #artdaily, #picsart, #dogtraining, #beautiful, #fashionista, #likeme
About life drawing	#like4likes, #digitalart
About live music	#dog, #fun, #contemporaryart
…	…

2. Rank the top 5 most popular posts of all time with the tag **'#leadership'** based on the amount of likes they've received.
   ***Note**:*

- *The list may be longer than 5 if some of the post have the same rank.*
- *Remember rule 5.*

***Tip**: Use the PSQL function rank, there is also an alternate version of this function called dense_rank with a slightly different format.*

Expected output format:

postid	title	rank
81048	Cranky	1
73002	Over charity fundraiser	2
82958	Trapped	3
…	…	…

3. Present a weekly financial report spanning from week 1 to 30 based on how many first-time subscriptions were started, how many were renewed and overall activity (measured in number of posts created) during each of those weeks.
   ***Tip**:*
   - *Use the PSQL function date_part*
   - *The first date in the transaction table for a specific user is their registration date*

   ***Important notes**:*
   - *Started indicates the first ever transaction for a user.*
   - *Renewed indicates every subsequent transaction of the subscription.*
   - *If you observe the create table statements you'll notice that TransactionID is a foreign key to UserID.*
   - *There are many different ways to solve this problem so please try on your own a bit. The following is simply a showcasing of some methods that could be used to solve the problem, note that this is not the only way:*
     - *Breaking down queries into smaller problems is often a good way to solve complex queries. We can, for instance, use CTEs to create a temporary table to query from: with starting_dates as (…subquery…). We can then query this table as normal.*
     - *We can also handle each column in a complex query independently, for instance:*
       select
       week,
       (select count(*) from table1) as sub1,
       (select count(*) from table2) as sub2,
       (select count(*) from table3) as sub3
       from generate_series(1,30) as week

- *What will this give us? How can we make sure that each row in the output shows data only for the specific week?*

Expected output format:

week	new_customers	kept_customers	activity
1	0	0	0
2	0	0	0
3	0	0	0
4	37	0	0
…	…	…	…
8	25	15	0
…	…	…	…

4. We want to see all users that have registered to the service during the month of January and display whether or not they have managed to add any friends. The first column states the name of the user, the second column tells us if this user has at least one friend added with true or false, and the third column displays when the user was registered. This should be sorted by name alphabetically.
   Tip:
   - *The aggregate function "every" may be useful here. It may also be helpful to read up on different types of joins.*
   - *You can reuse the method to fetch registration date from question 3*

   Note: *Depending on your setup the output might show t/f instead of true/false, both of which are fine.*

Expected output format:

name	has_friend	registration_date
Albin Gren	false	2024-01-26
Alf Vikström	true	2024-01-26
Camilla Edlund	true	2024-01-28
…	…	…

5. In this database there exists a unique chain of friends of the type “X is friend with Y and Y is friend with Z, but X is not friend with Z” for some combinations X, Y, Z. Use a recursive method to present a list of each friend in the friend chain. The chain starts with User ID 20 and User Name Anas.
   Tip: *It may be helpful to read up on different types of joins.*
   Note: *in this query you are allowed to directly reference an ID once in order to identify the friend chain. Using an ID from a different user should identify that friend chain instead. The “First” friend in the friend chain is “Anas”*

Expected output format:

name	user_id	friend_id
Anas	20	21
Harald	21	22
Martin	22	23
…	…	…
Toto	27	—

P+

Present a table with 2 Columns, where the first column contains all users in the LMS who have created one or more posts during the month of march, and the second column contains a boolean value (True or False). The boolean column should state True (t) if all of the users’ posts from that month have received in total at least 50 likes and should otherwise be False (f). For example, two posts during the month of march with 10 likes each will result in 20 likes total.

Tip: Pay careful attention to which authors should be true.

Expected output format:

name	received_likes
Carina Ström	true
Christer Brodin	false
Ingvar Norén	true
Magnus Hultman	true
…	…