Lab 2

Lab Overview

SQL, Structured Query Language, is a programming language used to communicate with relational databases. It's a declarative language in which you write queries to describe the output you want. In this lab, the client is looking to add some data and get an overview of the added data in a PostgreSQL database.

Rules

The installation guide provides students with the proper software needed to complete this lab.

1. 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. Both students in a group must be able to present all of the lab for the group to pass. Lab assistants do not record partial labs.
2. You must present correct and valid solutions to all the given tasks in order to pass the lab.
3. Presenting P+ assignments are optional for a higher grade if the given tasks are completed and passed.
4. This is a PSQL lab. No other programming languages, either embedded in the database or external to it, are allowed.
5. Please refrain from creating any functions since this lab is designed to assess query programming languages. Usage of built in PSQL functions is acceptable.
6. 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.
7. You must utilize nothing but a single top-level SELECT statement to answer every problem(although you are, of course, allowed to use any number of sub-selects required within that top level statement). You are specifically forbidden from referencing any temporary data structures like views or temporary tables from the solutions. CTE:s are not temporary tables and are encouraged as queries get longer.

Lab 2

Lab Presentation

Course related terminology is expected during the the lab presentation and make sure you have the following ready to be presented to the TA:

- Code for the database creation showing keys, domains etc. as part of lab 1.
- Simple executable select statements to show the contents of the database.
- All queries asked for in the tasks, ready to be executed.
- Motivations for how the solution for each requested query is sensible. (Not all tasks have one query as an answer, but they have to be good enough for e.g. a client to accept.)

Before you start

Make sure you're saving all your queries and insert statements. PostgreSQL uses the ".psql" fileformat. You can use your prefered text editing software.

Tasks

Prelude: BCNF

If your database is still not normalized you **must** do it now. You will not pass this lab otherwise. Please take some time here to make sure that it is in BCNF before you go on to the other tasks.

Part 1: Populating the database

Insert data into each table fulfilling the following requirements. It's a good idea to do the data manipulating actions lastly, when you are certain the relations won't need altering, because updating lots of data can be very tedious work. Check the relations thoroughly to make sure the data is inserted correctly.
Tip*: if you're having a hard time coming up with books here's a list*

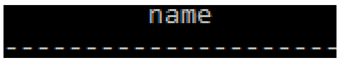
Lab 2

1. At least
   a. 15 book titles.
   b. 25 physical book copies in total. Make sure that:
      - *Some books can have more than one copy.*
      - ***At least** one book should have no copies.*
   c. 10 different students.
   d. 3 different admins.
   e. 5 borrowed books that have not been returned.
   f. 5 borrowed books that have been returned.
   g. 5 fines.

Part 2: Querying the database

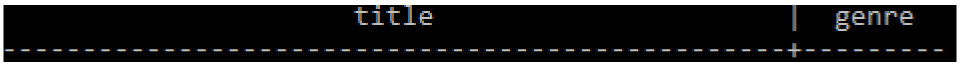
The following statements for querying a database are to be executed and should show a satisfactory result. Be sure to follow the output format!

1. Find and display all users' full names.
   Output format:



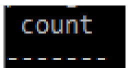
```
         name
----------------------
```

2. Display the title and genre of all books within a genre of your choice and order by the title.
   ***Important notes**: If a book has additional genres, those genres don't need to be listed.*
   Make sure to list one specific genre and not all genres.
   Output format:



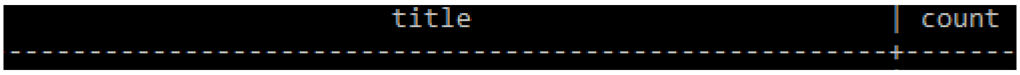
```
                       title                        |  genre
----------------------------------------------------+---------
```

3. Count all the books currently borrowed.
   Output format:



```
 count
-------
```

4. Count all the physical copies of all books (including books that have no copies).
   Output format:

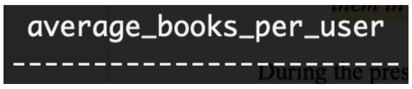


```
                          title                           | count
----------------------------------------------------------+-------
```

Lab 2

5. Present the average amount of physical books users have borrowed since the first of january, round the result to two decimals.
   Important notes:
   - *two books of the same title count as 2 books.*

   Output format:



```
 average_books_per_user
------------------------
```

P+

Using the recursive method, sum all **prime** numbers between 1 and 100.

Important notes:

- *You are specifically forbidden from referencing any tables in this solution.* *(In other words: don't create a table with manually inserted primes and then sum them in a query.)*

Tip:

- *There is more than one approach.*