## 1 Preliminaries

Before starting on this assignment, please be sure to read the General Instructions that are on Piazza (under Resources->General Resources). If you did Lab1, you should already know how to log in to the class PostgreSQL server. You'll get help on Lab2 in your Lab Section, not the Lectures, so *be sure to attend Lab Sections*.

## 2 Goal

The goal of the second assignment is to create a PostgreSQL data schema with 7 Newspaper Subscription tables that are very similar to the tables that you created in Lab1. The tables have the same names, attributes and data types as the tables of Lab1, and the same Primary Keys and Foreign Keys, but which has some additional UNIQUE constraints and restrictions on NULL that are described below.

After you create the data schema with the 7 tables, you must write five SQL statements that use those tables. Under Resources→Lab2, we have provided you with data that you can load into your tables so that you can test the results of your queries. Testing can prove that a query is wrong, but not that it is right, so be careful. We will not give you the results of these queries on the load data; you should be able to figure out the results on that data yourselves. You can also test your queries on your own data. In the "real world", you have to make and check your own tests.

Lab2 is due in two weeks, so you will have an opportunity to discuss the assignment during the Discussion Section in the first week of the assignment, and to discuss issues you have had in writing a solution to the assignment during the Discussion Section of the second week. Instructions for submitting the assignment appear at the end of this document.

## 3 Lab2 Description

### 3.1 Create PostgreSQL Schema Lab2

You will create a Lab2 schema to set apart the database tables created in this lab from the tables you will create in future, as well as from tables (and other objects) in the default (public) schema. We'll refer to the Newspaper Subscription database throughout the Lab Assignments, but your schemas will be called Lab1, Lab2, etc. In PostgreSQL, a database can have more than one schema; see here for more details on PostgreSQL schemas. You create the Lab2 schema as follows:

```
CREATE SCHEMA Lab2;
```

Now that you have created the schema, you want to set Lab2 to be your default schema when you use psql. If you do not set Lab2 as the default schema, then you will have to qualify your table names with the schema name (e.g., Lab2.Authors). To set the default schema, you modify your search path. (For more details, see here.)

```
ALTER ROLE username SET SEARCH_PATH to Lab2;
```

You will need to log out and log back in to the server for this default schema change to take effect. (Students often forget to do this.)

You do not have to include the CREATE SCHEMA or ALTER ROLE statements in your solution.

### 3.2 Create tables

You will create tables in schema Lab2 for SubscriptionKinds, Editions, Subscribers, Subscriptions, Holds, Articles, and ReadArticles. The attributes for these 7 tables are the same as for the tables of Lab1. Moreover, the data types for the attributes in these tables are the same as the ones specified for the tables of Lab1, and the Primary Keys and other Foreign Keys are also the same. You may use the *create_lab1.sql* file solution that we provided on Piazza for Lab2 as the basis for the *create_lab2.sql* file that you include in your Lab2 solution (although you don't have to include our comments). However, the tables must have additional constraints, which are described in the next section.

#### 3.2.1 Constraints

The following attributes cannot be NULL. All other attributes can be NULL … but remember that attributes in Primary Keys also cannot be NULL.

- In SubscriptionKinds: rate
- In Subscribers: subscriberName
- In Articles: articleAuthor

Also, the following must be unique for the specified table. That is, there cannot be identical rows in that table that have exactly the same (non-NULL) values for all of those attributes (composite unique constraint).

- In Subscribers: the 2 attributes subscriberName, subscriberAddress
- In Holds, the 3 attributes subscriberPhone, subscriptionStartDate and holdEndDate
- In Articles, the 3 attributes editionDate, articleAuthor and articlePage

For example:

- The first constraint says that there can't be two rows in Subscribers which have the same values for both subscriberName and subscriberAddress (if both of those attributes are not NULL). Think of this as saying that there can't be two subscribers who have the same non-NULL name and address. *A question that's not part of Lab2: Could any of these attributes be NULL?*
- The third constraint says that there can't be two rows in Articles which have the same values for all 3 attributes editionDate, articleAuthor and articlePage (if all 3 of these attributes are not NULL). So an author can't have more than one article in an edition that's on the same page, assuming that all of those attributes aren't NULL.

You will write a CREATE TABLE command for each of the 7 tables that has these additional constraints. Save the commands in the file *create_lab2.sql*

## 4 SQL Queries

Below are English descriptions of the five SQL queries that you need to write for this assignment, which you will include in files queryX.sql, where X is the number of the query, e.g., your SQL statement for Query 1 will be in the file query1.sql, and so forth. Follow the directions as given. You will lose points if you give extra tuples or attributes in your results, if you give attributes in with the wrong names or in the wrong order, or if you have missing or wrong results. You will also lose points if your queries are unnecessarily complex, even if they are correct. Grading is based on correctness of queries on all data, not just the load data that we have provided.

Remember the Referential Integrity constraints from Lab1, which you should retain for Lab2. For example, if a subscriber (identified by subscriberPhone) appears in a Subscriptions tuple, then there must be a tuple in Subscribers for that subscriber (i.e., with that subscriberPhone value).

Attributes should have their original names in the results of your queries, unless an alias is requested. And if a query asks that several attributes appear in the result, the first attribute mentioned should appear first, the second attribute mentioned should appear second, etc.

### 4.1 Query 1

The Editions table has attribute which specify the date of that edition, the number of articles in that edition and the number of pages in that edition. But database systems don't understand English; obvious English constraints about values (like article numbers and page numbers) aren't enforced unless databases and applications enforce them.

Write a SQL query which finds the editionDate of editions which have at least one article whose article number is more than the number of articles in that edition, and whose page number is more than the number of pages in that edition.

The attribute in your result should appear as theEditionDate. No duplicates should appear in your result.

### 4.2 Query 2

A subscription kind is identified by its mode and interval. For a subscription, paymentReceived indicates whether payment has been received for that subscription.

Write a SQL query which finds the phone and name for subscribers who have a subscription whose rate is more than 137.25 and for which payment has not been received.

The rows in your result should appear in alphabetical order by name. If two result rows have the same name, then the row with the bigger phone number should appear before the row with the smaller phone number.

No duplicates should appear in your result.

### 4.3 Query 3

Write a SQL query that outputs the article number, article author and article pages for the articles which have not been read for more than 20 minutes by any subscriber who has the string 'er' (with that capitalization) appearing anywhere in their name.

No duplicates should appear in your result.

### 4.4 Query 4

subscriptionInterval is an attribute in SubscriptionKinds. The end date of a subscription can be computed by adding its subscriptionStartDate to the subscriptionInterval for its subscription kind.

But when you add a DATE and an INTERVAL, the result is a TIMESTAMP (not a DATE). If you have a TIMESTAMP value myTimestamp, then one of the ways that you get the date from that TIMESTAMP is by writing DATE(myTimestamp).

Write a SQL query which finds all subscriptions where:

- the subscription starts on or before December 17, 2022,
- the subscription ends on or after October 3, 2023,
- the address of the subscriber who has that subscription isn't NULL,
- the subscription kind (which is identified by subscriptionMode and subscriptionInterval) is still being offered (stillOffered), and
- there has been at least one hold (in Holds) placed on the subscription.

Attributes which should appear in your result as subscriberPhone, subscriptionStartDate, subscriptionEndDate, subscriberName and subscriptionRate, where:

- subscriberPhone identifies the subscriber for the subscription,
- subscriptionStartDate is the start date of the subscription,
- subscriptionEndDate is the end date of the subscription,
- subscriberName is the name of the subscriber, and
- subscriptionRate is the rate for that kind of subscription.

No duplicates should appear in your result.

### 4.5 Query 5

For each article in the Articles table, articleAuthor gives the name of the author, and editionDate specifies the edition in which the article appears. There could be a subscriber who has the same name as an article author.

Write a SQL query that finds the subscriber name, and subscriber address and edition whenever that subscriber's name appears as the author of more than one article which is in that edition.

The attributes in your result should appear as theSubscriberName, theSubscriberAddress and theEditionDate. No duplicates should appear in your result.

## 5 Testing

While your solution is still a work in progress, it is a good idea to drop all objects from the database every time you run the script, so you can start fresh. Of course, dropping each object may be tedious, and sometimes there may be a particular order in which objects must be dropped. The following commands (which you can put at the top of *create_lab2.sql* if you want, but you don't have to), will drop your Lab2 schema (and all objects within it), and then create the (empty) schema again:

```
DROP SCHEMA Lab2 CASCADE;
CREATE SCHEMA Lab2;
```

Before you submit, login to your database via psql and execute your script. As you've learned already, the command to execute a script is: \i <filename>.

Under Resources→Lab2 on Piazza, we have posted a load script named *load_lab2.sql* that loads data into the 7 tables of the database. You can execute that script with the command:

*\i load_lab2.sql*

You can test your 5 queries using that data, but you will have to figure out on your own whether your query results are correct. We won't provide answers, and students should not share answers with other students. Also, your queries must be correct on any database instance, not just on the data that we provide. You may want to test your SQL statements on your own data as well.

## 6 Submitting

1. You can always get rid of duplicates by using DISTINCT in your SELECT. In CSE 180, we deduct points if students use DISTINCT and it wasn't necessary because even without DISTINCT, there couldn't be duplicates. We will also deduct if you were told not to eliminate duplicates, but you did.

2. Save your scripts for table creations and query statements as *create_lab2.sql* and *query1.sql* through *query5.sql* You may add informative comments inside your scripts if you want (the server interprets lines that start with two hyphens as comment lines).

3. Zip the file(s) to a single file with name Lab2_XXXXXXX.zip where XXXXXXX is your 7-digit student ID. For example, if a student's ID is 1234567, then the file that this student submits for Lab2 should be named Lab2_1234567.zip

   To generate the zip file you can use the Unix command:

   *zip Lab2_1234567 create_lab2.sql query1.sql query2.sql query3.sql query4.sql query5.sql*

   (Of course, you use your own student ID, not 1234567.)

4. You should already know how to transfer the files from the UNIX timeshare to your local machine before submitting to Canvas. If you are still not familiar with the process, use the instructions that we provided at the Lab1 assignment.

5. Lab2 is due by 11:59pm on Tuesday, October 31. **Late submissions will not be accepted; Canvas won't take them, nor will we.** Always check to make sure that your submission is on Canvas, and that you've submitted the correct file.

   But as the Syllabus tells you, Lab Assignment grades don't count towards your CSE 180 Course Score! Even though Lab Assignments will be graded, the most important feedback students receive from Lab Assignment grades is about their errors. So be sure to submit your Lab Assignments on-time, so that you'll receive that feedback.

   Yes, your CSE 180 Course Score won't be affected if you don't do any Lab Assignments, or if you do all your assignments using an LLM-based system such as ChatGPT. But those are very poor choices, since you won't be able to use LLM-based systems on CSE 180 Exams, or during job interviews.