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 6 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 but there are some UNIQUE constraints and some restrictions on NULLs.

After you create the data schema with the 6 tables, you will be required to write some SQL statements that use those tables. Under Resources→Lab2, we will soon provide 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.

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 ones you will create in future, as well as from tables (and other objects) in the default (public) schema. Note that the meaning of schema here is specific to PostgreSQL and different from the general meaning. See here for more details on PostgreSQL schemas. You create the Lab2 schema like this:

```
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.Customers). 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 the tables Movies, Theaters, TheaterSeats, Showings, Customers and Tickets. The attributes of the 6 tables are the same as the tables of Lab1. Data types for the attribute names in these tables are also the same as the ones specified for the tables of Lab1. The Primary Keys are also the same. You may use our Lab1 solution as a basis for Lab2, or you may us your own solution if it's correct. However, the tables must have the additional constraints described in the next section.

3.2.1 Constraints

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

- In Movies: name
- In Theaters: numSeats
- In TheaterSeats: brokenSeat

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 Theaters: the attribute address
- In Movies: the 2 attributes name and year
- In Customers: the 2 attributes name and address

For example, the third constraint says that there can't be two rows in Customer that have the same values for both name and address (if both name and address are not NULL). Think of this as saying that there can't be two different customers who have both the same name and address..

You will write a CREATE TABLE command for each of the 6 tables. Save the commands in the file create.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.

Remember the Referential Integrity constraints from Lab1, which should be retained for Lab2. For example, if a customerID appears in Tickets tuple, then there must be a tuple in Customers that has that same customerID.

4.1 Query 1

Find the ID and address for all the theaters that have a broken seat. No duplicates should appear in your answer.

4.2 Query 2

Find the name and year of all movies for which a customer named Donald Duck bought a ticket. No duplicates should appear in your answer.

4.3 Query 3

Find the ID, name, year and length for every movie which was longer than the 2011 movie Avengers. In your result, movies with the largest year should appear first; within each year, movies should be in alphabetized by name. No duplicates should appear in your answer.

4.4 Query 4

Find the ID and name of each customer whose name has the letter 'a' or 'A' anywhere in it, and who bought tickets to at least 2 different movies. Careful; a customer who bought 2 or more tickets to the same movie doesn't qualify. No duplicates should appear in your answer

4.5 Query 5

For each ticket for which all of the following are true:

a) the ticket was bought by a customer whose name starts with 'D' (capital D),
b) the ticket is for a showing whose price code isn't NULL, and
c) the ticket is on a date between June 1, 2019 and June 30, 2019 (including those dates), and
d) the ticket is for a theater that has more than 5 seats,

Output the ID, name and address of the customer, the address and number of seats of the theater, and the price code for the showing. The 6 attributes in your result should appear as custID, custName, custAddress, theaterAddress, theaterSeats and priceCode. 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.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 will soon provide a load script named *lab2_data_loading.sql* that loads data into the 6 tables of the database. You can execute that script with the command:

\i *lab2_data_loading.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. Save your scripts for table creations and query statements as create.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).

2. 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.sql query1.sql query2.sql query3.sql query4.sql query5.sql

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

3. 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 we provided at the Lab1 assignment.

4. Lab2 is due by 11:59pm on Sunday, February 2. Late submissions will not be accepted, and there will be no make-up Lab assignments.

5. You may lose credit if your queries are unnecessarily complex, even if they are correct. For example, if you use DISTINCT but it's not needed, you might lose half a point. Of course, you will also lose credit if DISTINCT is needed and you omit it.

6. Be sure to follow directions about Academic Integrity that are in the Syllabus. If you have any questions about those directions, please speak to the instructor as soon as possible.