1. Preliminaries

Before starting on this assignment, please be sure to read the PostgreSQL Info file (CSE182_S25_PSQL_Info.pdf) and the Docker Install Guide that are on Piazza.

2. Goal

The goal of the first assignment is to create a PostgreSQL data schema with 7 tables. That is all that is required in this assignment; don't do anything that's not required. The other Lab Assignments will be more challenging. In your Lab Sections, you may be given information about how to load data into a table and issue simple SQL queries, because that's fun, but loading data and issuing queries are **not** required in this assignment. (That will show up in the Lab2 assignment.)

3. Lab1 Description

3.1 Create PostgreSQL Schema Lab1

You will create a Lab1 schema to set apart the database tables created in this lab from tables you will create in future labs, 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 distinct from the general meaning of schema. See here for more details on PostgreSQL schemas. You create the Lab1 schema using the following command:

```
CREATE SCHEMA Lab1;
```

[PostgreSQL makes all identifiers lowercase, unless you put them in quotation marks, e.g.,, "Lab1". But in CSE 182, you don't have to bother using quotation marks for identifiers. We use capitals in assignments for readability, but it's okay (and equivalent) if you use lab1 as the schema name.]

Now that you have created the schema, you want to make Lab1 be the default schema when you use psql. If you do not set Lab1 as the default schema, then you will have to qualify your table names with the schema name (e.g., by writing Lab1.customer, rather than just customer). To set the default schema, you modify your search path as follows. (For more details, see here.)

```
ALTER ROLE cse182 SET SEARCH_PATH to Lab1;
```

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, and then are surprised that their tables aren't in the expected schema.) To see your current SEARCH_PATH, enter:

```
SHOW SEARCH_PATH;
```

If you forget to do the ALTER ROLE for Lab1 to modify your search path then your schema will be the default schema, PUBLIC.

3.2 Tables

You will be creating a (simplified) database for a pharmacy store chain. The data types and Referential Integrity for the attributes in these 7 tables are described in the next section. This schema might not provide everything that an actual database of such a company would include, but it's a decent start.

Important: To receive full credit, you must use the attribute names as given, and the attributes must be in the order given. Also, the data types and referential integrity must match the specifications given in the next section. Do not do more than you're asked to do in this assignment.

Customer(customerID, customerName, address, dateOfBirth, healthInsuranceName)

Pharmacy(pharmacyID, address, openTime, closeTime, numEmployees)

Drug(drugID, drugName, manufacturer, prescriptionRequired)

Supplier(supplierID, supplierName, rating)

Purchase(purchaseID, customerID, pharmacyID, purchaseTimestamp, totalPrice, creditCardType, creditCardNumber, expirationDate)

DrugsInPurchase(purchaseID, drugID, quantity, subtotal, discount)

OrderSupply(pharmacyID, supplierID, drugID, drugPrice, quantity, orderDate, status)

The underlined attribute (or attributes) identifies the Primary Key of each table. A table can only have one Primary Key, but that Primary Key may involve multiple attributes.

- The relation Customer specifies the customer's ID, their name, their address, their date of birth, and their health insurance.
- The relation Pharmacy specifies the ID of a specific branch of the pharmacy chain, its address, the hours of operation (assuming it is the same every day), and the number of employees.
- The relation Drug specifies the drug's ID, its name, its manufacturer (which is not the same as the supplier), and whether a prescription is needed to purchase it.
- The relation Supplier specifies the supplier's ID, name and a rating on a scale 1-10 according to how reliable the supplier has been.
- The relation Purchase records an ID for every purchase of a customer, the ID of the customer and the pharmacy that the purchase was made, a timestamp, the total amount of dollars paid, the credit card type, number, and its expiration date.
    - Any customerID that's in a Purchase row must appear as a customerID in the Customer table.
    - Any pharmacyID that's in a Purchase row must appear as a pharmacyID in the Pharmacy table.

- The relation DrugsInPurchase specifies the ID of the purchase, the ID of a drug in the purchase, its quantity, the subtotal amount of dollars for that drug, and whether there was a discount for this drug as a percentage.
    - Any purchaseID that's in a DrugsInPurchase row must appear as a purchaseID in the Purchase table.
    - Any drugID that's in a DrugsInPurchase row must appear as a drugID in the Drug table.
- The relation OrderSupply records an order made to a supplier for a specific drug, using the pharmacy's ID, the supplier's ID, the drug's ID, the price (a single unit of) the drug was bought, the quantity, the date that the order was placed, and its status (completed/in transit/…).
    - Any pharmacyID that's in a OrderSupply row must appear as a pharmacyID in the Pharmacy table.
    - Any supplierID that's in a OrderSupply row must appear as a supplierID in the Supplier table.
    - Any drugID that's in a OrderSupply row must appear as a drugID in the Drug table.

In this assignment, you'll just have to create tables with the correct table names, attributes, data types, Primary Keys and Referential Integrity. "Must appear as" means that there's a Referential Integrity requirement. **Be sure not to forget Primary Keys and Referential Integrity when you do Lab1!**

3.2.1 Data types

- For customerID, pharmacyID, drugID, supplierID, purchaseID, numEmployees, rating, creditCardNumber, quantity, and discount use *integer*.
- For customerName, healthInsuranceName, drugName, manufacturer, and supplierName use *character of variable length,* with maximum length 40.
- For address use *character of variable length,* with maximum length 50.
- For status use *character with fixed length 4*. We'll say more about the values of this attribute later.
- For dateOfBirth, expirationDate, and orderDate use the *date* type.
- For openTime and closeTime use the *time* type (without time zone) with no fractional digits for recording seconds.
- For purchaseTimestamp use the *timestamp* type.
- For totalPrice, subtotal, and drugPrice use *numeric,* with at most 6 decimal digits to the left of the decimal point and 2 decimal digits after it.
- For creditCardType use *character with fixed length 1*. We'll say more about the values of this attribute later.
- For prescriptionRequired use *boolean*.

You must write a CREATE TABLE statement for each of the 7 tables. Write the statements in the same order that the tables are listed above. **Use the data types, Primary Keys and Referential Integrity described above.** You will lose credit if you do anything beyond that, even if you think that it's sensible. Save your statements in the file create_lab1.sql

PostgreSQL maps all SQL identifiers (e.g., table names and attributes) to lowercase. That's okay in your CSE 182 assignments. You won't lose points for Lab1 because Customer appears in the database as customer. It is possible to specify specific case choices for an identifier by putting that identifier inside double-quote symbols, e.g., as "Customer". But then every time you refer to that identifier, you'll have to use the double-quotes. "CUSTOMER" is not the same identifier as "Customer", and neither of those is the same as Customer (written without double-quotes), which PostgreSQL maps to customer. We will use capitalization for readability, but we won't bother using double-quotes in our own Lectures, Lab Assignments and Exams.

4. Testing

While you're working on your solution, it is a good idea to drop all objects from the schema every time you run the create_lab1.sql script, so you can start fresh. Dropping each object in a schema may be tedious, and sometimes there may be a particular order in which objects must be dropped. (Why?) The following command, which you should put at the top of your create_lab1.sql script, will drop your Lab1 schema (and all the objects within it), and then create the (empty) schema again:

```
DROP SCHEMA Lab1 CASCADE;
CREATE SCHEMA Lab1;
```

The first statement will result in an error if the Lab1 Schema doesn't exist, but execution of your script will continue after that.

Before you submit your Lab1 solution, login to your database via psql and execute your create_lab1.sql script. As you'll learn in Lab Sections, the command to execute a script is: \i <filename> Verify that every table has been created by using the command: \d When you execute \d, the tables may be displayed in any order, not necessarily in the order in which you created them.

To verify that the attributes of each table are in the correct order, and that each attribute is assigned its correct data type use the following command: \d <table>.

We've supplied some load data that you can use to test your solution in the file load_lab1.sql. After you've created your tables, using the command : \i create_lab1.sql, you can load that data in psql by executing the command: \i load_lab1.sql. (Why will loading the data twice always result in errors?) If your solution fails on the load data, then it's likely that your solution has an error. But although testing can demonstrate that a program is buggy, testing cannot prove that a program is correct.

You do not have to develop your solution on unix.ucsc.edu, but please recognize that we'll run your solution on unix.ucsc.edu. If your solution fails on unix.uscs.edu, you'll receive a poor grade, even if you your solution worked in some other environment.

5. Submitting

1. Save your script as create_lab1.sql You may add informative comments to your scripts if you want. Put any other information for the Graders in a separate README file that you may submit.

2. Zip the file(s) to a single file with name Lab1_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 Lab1 should be named Lab1_1234567.zip

   If you have a README file (which is not required), you can use the Unix command:

   zip Lab1_1234567 create_lab1.sql README

   If you don't have a README file, to create the zip file you can use the Unix command:

   zip Lab1_1234567 create_lab1.sql

   (Of course, you should use **your own student ID**, not 1234567.) Submit a zip file, even if you only have one file.

Submit the zip file on Canvas under Assignment Lab1. Please be sure that you have access to Canvas for CSE 182. Registered students should automatically have access; students who are not registered in CSE 182 will not be submit solutions. You can replace your original solution, if you like, up to the Lab1 deadline. (Canvas will give the new file a slightly different name, but that' okay.) No students will be admitted to CSE 182 after the Lab1 due date.

The CSE 182 Teaching Assistants will help you set up PostgreSQL in a docker container and show you how to move and zip the necessary files in order to submit them during Lab Sections. Attend your Lab Section to ensure that you know how to handle this correctly!

Lab1 is due by 11:59pm on Tuesday, April 15. **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.

Students are strongly encouraged to do Lab Assignments on their own, perhaps using Generative AI systems (such as OpenAI's ChatGPT, Microsoft's Copilot, Meta's LLaMA, and Google's Bard) as learning aids, but not as substitutes for learning. These systems may provide incorrect answers, so they should be regarded as clever (somewhat unreliable) assistants which have terrific memories, not as perfect oracles.

Students will not be able to use LLM-based systems on CSE 182 Exams, or during job interviews. Lab Assignments scores count 20% towards Course Scores, and students who use LLM-based systems on Lab Assignments to supply solutions might receive perfect scores … but such students are mostly cheating themselves.