1 Goal

The goal of the second assignment is to create a PostgreSQL data schema with 7 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 they also have some 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 will 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. 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.

2 Lab2 Schema

2.1 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. 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 cse182 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.

2.2 Create tables

As in Lab1, you will create the tables Customer, Pharmacy, Drug, Supplier, Purchase, DrugsInPurchase, and OrderSupply. The attributes for these 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 which you include in your Lab2 solution (although you don't have to include our comments, and you're using Lab2 instead of Lab1 as the schema). However, your Lab2 tables must have additional constraints, which are described in the next section.

2.2.1 Constraints

The following attributes cannot be NULL. All other attributes can be NULL … but remember that attributes in Primary Keys also cannot be NULL, even though NOT NULL isn't specified for them.

- In Customer: customerName
- In Pharmacy: address
- In Drug: drugName
- In Supplier: supplierName

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 Customer: the 2 attributes customerName and address
- In Pharmacy: the attribute address
- In Supplier: the attribute supplierName
- In Purchase: the 3 attributes purchaseTimestamp, creditCardType, creditCardNumber

Explanations:

- The first constraint says that there can't be two rows in Customer which have the same values for both of the attributes customerName and address (if both of those attributes are not NULL).

  For example, there can't be two customers whose customerName is 'John Doe', and whose address is '123 Main St, New York, NY'. But there could be two customers whose customerName is 'John Doe' and whose address is NULL.

- The second constraint says that there can't be two rows in Pharmacy which have the same values for address (recall that address here isn't allowed to be NULL*).*

- The third constraint says that there can't be two rows in Supplier which have the same values for supplierName (recall that supplierName here isn't allowed to be NULL*).*

- The fourth constraint says that there can't be two rows in Purchase which have the same values for all three of the attributes purchaseTimestamp, creditCardType, and creditCardNumber. (if all three of those attributes are not NULL).

  This means that we disallow two purchases from the same credit card that occur at the exact same moment because this is most likely a duplicate transaction.

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

3 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 that the Referential Integrity constraints from Lab1 are still in effect here. For example, any purchaseID that is in a DrugsInPurchase row must appear as a purchaseID in the Purchase table.

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.

3.1 Query 1

Write a SQL query which finds the purchases for which the name of the customer starts with 'R' (with that capitalization), the pharmacy has exactly 12 employees, and the total price paid is greater than 40.00 dollars. The attributes in your result should be the name of the customer, the address of the pharmacy, and the total price, which should appear in your result as theCustomerName, thePharmacyAddress, and theTotalPrice.

No duplicates should appear in your result.

3.2 Query 2

A credit card is a Visa card if the type of that credit card (creditCardType) is 'V' (uppercase V).

Write a SQL query which finds all customers whose have made a purchase with a Visa credit card and who have never (in any pharmacy and using any credit card) made a purchase that exceeds 100.00 dollars in total price. The attributes in your result should be address and customerID. Tuples in your result should be in reverse alphabetical order by address; if two tuples have the same address, they should appear in increasing order of customerID.

No duplicates should appear in your result.

3.3 Query 3

Recall that a pharmacy has ordered a drug from a supplier if there is a tuple in OrderSupply for that pharmacy, supplier, and drug.

Write a SQL query which finds all suppliers, excluding 'McKesson', who received an order for the exact same drug by at least 2 different pharmacies. The attribute in your result should be the supplier's ID, which should appear as theSupplierID.

No duplicates should appear in your result.

3.4 Query 4

Note: If myTimestamp is a TIMESTAMP, then in PostgreSQL, DATE(myTimestamp) is the DATE value which is in that timestamp. Unfortunately, different relational database systems have different ways of extracting the DATE from a TIMESTAMP. (There are at least 3 ways to do that in PostgreSQL; this one is the simplest.)

Recall that a tuple in DrugsInPurchase identifies a specific drug contained in a purchase. Write a SQL query which finds all purchased drugs for which all of the following are true:

- The drug required a prescription. (Use prescriptionRequired.)
- The date of purchaseTimestamp is January 12, 2024 or later.
- The pharmacy address where it was purchased has 'en' appearing anywhere in it, with that capitalization.
- The quantity purchased is greater than or equal to 2.
- The credit card number used for the purchase is NULL.

The attributes in your result should be the drugID and the purchaseTimestamp, which should appear as theDrugID and thePurchaseTimestamp.

No duplicates should appear in your result.

3.5 Query 5

Several supply orders in the OrderSupply table might have the same order date. We want to find all the orders who have the earliest order date. An order has the earliest order date if there are no orders who have an earlier order date.

Write a SQL query which finds the pharmacyID and supplierID for all orders who have the earliest order date. The attributes in your result should appear as thePharmacyID and theSupplierID.

No duplicates should appear in your result.

4 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 will post 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 to the 5 queries when they are run on the load data, 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.

5 Submitting

1. 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 (lines that start with two hyphens are interpreted 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_lab2.sql query1.sql query2.sql query3.sql query4.sql query5.sql

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

3. Lab2 is due by 11:59pm on Tuesday, April 29. Late submissions will not be accepted, and there will be no make-up Lab assignments.

4. You can get always rid of duplicates by using DISTINCT in your SELECT. In CSE 182, we do not deduct points if students use DISTINCT and it wasn't necessary to use it. (Using DISTINCT is unnecessary in a query if the query couldn't have duplicates even without DISTINCT appearing.) However, we will deduct points if you weren't told to eliminate duplicates but you do eliminate them.

5. 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.