Lab Assignment 2 CSE 180 – Fall 2020 Due: Sunday, Nov 1, 11:59pm

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 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 with 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 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 Houses, Persons, Brokers, Offers, ForSaleHouses and SoldHouses. The attributes of the 6 tables are the same as the tables of Lab1 with one exception. Persons no longer has an address attribute; instead, it has a houseID attribute, identifying the house in which that person lives. For Lab2, you're given a revised_create.sql that corresponds to this attribute change; revised_create.sql also includes a new Referential Integrity constraint which requires that if a person lives in a house (specified by houseID in Persons) then there must be a corresponding houseID in Houses. Data types for the other 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 should use revised_create.sql file as the basis for your create.sql file for Lab2. However, the tables must have additional constraints described in the next section.

Houses(houseID, address, ownerID, mostRecentSaleDate)

Persons(personID, personName, **houseID**)

Brokers(brokerID, brokerLevel, companyName, soldCount)

Offers(houseID, offererID, offerDate, offerPrice, isACurrentOffer)

ForSaleHouses(houseID, forSaleDate, brokerID, forSalePrice, isStillForSale)

SoldHouses(houseID, forSaleDate, soldDate, buyerID, soldPrice)

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 Houses: address
- In Offers: offerPrice
- In SoldHouses: soldPrice

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 Houses: the attribute address
- In Persons: the 2 attributes personName and houseID
- In SoldHouses: the 2 attributes buyerID and soldDate

For example, the first constraint says that there can't be two rows in Houses that have the same address. (Note that you were also told just above that the address attribute in Houses can't be NULL.) And the third constraint says that there can't be two rows in SoldHouses that have the same values for both buyerID and soldDate (if both buyerID and soldDate are not NULL). Think of this as saying that there can't be two houses that were sold which were bought by the same buyer on the same date.

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. 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 should be retained for Lab2. For example, if a houseID appears in an Offers tuple, then there must be a tuple in Houses that has that same houseID. And Lab2 has an additional constraint: If a houseID appears in a Persons tuple, then there must be a tuple in Houses that has that same houseID.

You should also assume that every owner is a person, every broker is a person, and every offerer is a person, and every buyer is a person, even though we haven't specified those constraints yet.

4.1 Query 1

For each offer that is current, give the name of the offerer, the address of the house, and the mostRecentSaleDate (which is an attribute of Houses) for that house. Order your result in alphabetical order based on name of the offerer. If there are two tuples in your result that have the same offerer name, the tuple with the later value for most recent sales date should come first. No duplicates should appear in your result.

4.2 Query 2

For each person who has the string 'son' appearing anywhere in their name (lowercase letters), give the name of the person, the address of their house, and the name of the owner of that house. No duplicates should appear in your result.

4.3 Query 3

Find the ID and name of each broker whose company is 'Weathervane Group Realty', and who had at least one house for sale before October 1, 2020 that is not still for sale, and where their "for sale house" sold for one million or more No duplicates should appear in your result.

(Careful; the same house could be put up for sale more than once, perhaps with different brokers.)

4.4 Query 4

For each house that has at least one offer, find the highest offer price for that house. Your output should include the houseID, its address and the highest offer price for that house. The attributes in your result should be houseID, address and highOffer. No duplicates should appear in your result.

4.5 Query 5

For each sold house for which the following are all true:

a) The buyer's name starts with the letter 'S' (uppercase), and
b) the sold date was between February 10, 2020 and April 29, 2020 (including those dates), and
c) the price for which the house was sold was greater than its for sale price, and
d) there is at least one different house that is still for sale (isStillForSale) that has the same broker who put this house up for sale

Output the houseID, ownerID, buyerID, soldPrice, forSalePrice, brokerID and the companyName of the broker. The 7 attributes in your result should appear as theHouseID, theOwnerID, theBuyerID, soldPrice, forSalePrice, theBrokerID and theCompany. 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 have provided 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, November 1. Late submissions will <u>not</u> be accepted, and there will be no make-up Lab assignments.

5. You can get always 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 <u>not</u> to eliminate duplicates but you did.

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.