1.   Preliminaries

Before starting on this assignment, please be sure to read the General Instructions that are on Piazza (under Resources->General Resources), and also make sure you are able to log in to the class PostgreSQL server. You'll get help on this in your Lab Section, not the Lectures, so *be sure to attend Lab Sections*.

2.   Goal

The goal of the first assignment is to create a PostgreSQL data schema with 6 tables. That is all that is required in this assignment; don't do anything that's not required. The other Lab Assignments are much more difficult. 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

As we noted in the general instructions, 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; that's okay unless you put them in quotation marks, e.g.,, "Lab1". That's okay.; you don't have to bother using quotation marks for identifiers. We use capitals 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 username SET SEARCH_PATH to Lab1;
```

<u>You will need to log out and log back in to the server for this default schema change to take effect.</u> (Students **often forget** to do this, and then are surprised that their tables aren't in the expected schema.)

3.2 Tables

You'll be creating tables for a very simplified version of a Real Estate database schema, with tables for Houses, Persons, Brokers, Offers, ForSaleHouses and SoldHouses. The data types and Referential Integrity for the attributes in these 6 Real Estate tables are described in the next section. No, this schema doesn't provide everything that a real-world Real Estate database would have, 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.   Follow directions; <u>do not</u> do more than you're asked to do in this assignment.

Houses(<u>houseID</u>, address, ownerID, mostRecentSaleDate)

Persons(<u>personID</u>, personName, address)

Brokers(<u>brokerID</u>, brokerLevel, companyName, soldCount)

Offers(<u>houseID, offererID, offerDate</u>, offerPrice, isACurrentOffer)

ForSaleHouses(<u>houseID, forSaleDate</u>, brokerID, forSalePrice, isStillForSale)

SoldHouses(<u>houseID, forSaleDate</u>, soldDate, buyerID, soldPrice)

The underlined attribute (or attributes) identifies the <u>Primary Key</u> of each table.
- A house in Houses specifies the house's ID, address, owner, and the date that the house was most recently sold.
    - In practice, an owner would have to be a person, but you're not asked to specify that (yet).
- A person in Persons specifies the person's ID, name and address.
- A broker in Brokers specifies the broker's ID, level (which we'll explain later), company name, and count of the number of houses that broker has sold.
    - In practice, a broker would have to be a person, but you're not asked to specify that (yet).
- An offer is made by someone (the offerer) on a particular date to buy a house at a particular price. An offer in Offers specifies the house, the offerer, the date that the offer was made, the price that was offered, and a boolean indication as to whether the offer is still current. (An offerer can withdraw an offer, in which case it's no longer current.)
    - Any houseID that's in an Offers row must appear as a houseID in the Houses table.
    - In practice, a offerer would have to be a person, but you're not asked to specify that (yet).
- A house in ForSaleHouses is for sale. It has the houseID, the date that the house was put up for sale, the broker selling the house, the price that the house is being sold at, and a boolean indication as to whether the house is still up for sale. (A house is no longer for sale if it's been sold, or if its owner has taken it off the market.)
    - Any houseID that's in a ForSaleHouses row must appear as a houseID in the Houses table.
    - In practice, a broker would have to be a person, but you're not asked to specify that (yet).
- A house in SoldHouses has been sold. It has the houseID, the date that the house was put up for sale, the date that it was sold, the buyer and the price at which the house was sold.
    - Any (houseID, forSaleDate) that's in a SoldHouses row must appear as a (houseID, forSaleDate) in the ForSaleHouses table.
    - In practice, a buyer would have to be a person, but you're not asked to specify that (yet).

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

Sometimes an attribute (such as houseID, address and forSaleDate) appears in more than one table. Attributes that have the same attribute name might not have the same data type in all tables, but in our schema, they do.

- For houseID, ownerID, personID, offererID, brokerID, buyerID and soldCount, use *integer*.
- For brokerLevel, which classifies Brokers, use *character with fixed length 1*. (We'll explain values of brokerLevel in a later Lab Assignment.)
- For personName and companyName, use *character of variable length*, with maximum length 30.
- For address, use *character of variable length*, with maximum length 50.
- offerPrice, forSalePrice and soldPrice should be *numeric,* with at most 8 decimal digits to the left of the decimal point and 2 decimal digits after it.
- mostRecentSaleDate, offerDate, forSaleDate and soldDate should be of type *date*.
- The isACurrentOffer attribute, which indicates whether an offer is still current, and the isStillForSale attribute, which indicates whether a hold is still for sale, should be Boolean.

You must write a CREATE TABLE statement for each of the six tables in Section 3.2. 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. In particular, don't find ways to constrain owners, brokers, offerers and buyers so that they have to be persons. (Please wait; you'll be asked to do that in Lab3.) Save your statements in the file create.sql

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.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. The following command, which you should put at the top of your create.sql, 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;

Before you submit your Lab1 solution, login to your database via psql and execute your create.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

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

5. Submitting

1. Save your script as create.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 <u>not</u> required), you can use the Unix command:

        zip Lab1_1234567 create.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.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 180. Registered students should automatically have access; students who are not registered in CSE 180 will not have access. <u>No students will be admitted to CSE 180 after the Lab1 due date.</u>

If you are working on the UNIX timeshare and your zip file is located there, you will need to copy your file to your computer so that you can upload it to Canvas through your browser. For that, you will need an FTP (File Transfer Protocol) client to securely transfer files from the UNIX timeshare. A widely used secure FTP client is Filezilla. Installation instructions are found in the site of [FileZilla](#) (make sure you install the distribution suitable for your operating system). After opening the Filezilla client, you will need to set the host field to unix.ucsc.edu, the username to your CruzId and the password to your Blue password, while the port number should be set to 22 (the default port for remote login). By clicking the Quickconnect button, if your credentials are correct, you will connect and be able to see the contents of your remote Unix folder at the right pane (under the title "Remote site"), while the left pane (under the title "Local site") will display the contents of your local file system. With the mouse, you can drag the file from the Unix folder and drop it to the desired location at your computer. This will transfer the file to your local machine, without erasing it from its original remote location. Filezilla is only one of several options for an FTP client. If you are finding it difficult to install the necessary tools and successfully do file transfers, you should <u>promptly</u> ask for help in the Lab Sections; **do not** postpone this until the deadline date. Computers in UCSC Unix Labs also have pre-installed SSH and FTP clients (PuTTY and PSFTP).

Other approaches to copy files includes using SCP (Secure Copy) and using Cut-and-Paste, where you copy the contents of the file from the unix system, and then paste contents into a file on your computer. Cut-and-Paste may work with for small files, but that's a hack that does not work well for large files.

The CSE 180 Teaching Assistants, Sonali Suri and Jesh Bheemanpally, will discuss approaches to access unix remotely (SSH for Mac/Linux and PuTTY for Windows) and to move files to your computer (SCP for Mac/Linux and Filezilla for Windows/Mac/Linux) with you during Lab Sections. Attend your Lab Section to ensure that you know how to handle this correctly!

Lab1 is due by 11:59pm on Sunday, October 18. **Late submissions will not be accepted (Canvas won't**

take them), and there will be no make-up Lab assignments. Check to make sure that your submission is on Canvas, and that you've submitted the correct file. You will receive **no credit** if you accidentally submit the wrong file, even if you attempt to "prove" that you completed the correct file on time.