1 Preliminaries

Before starting on this assignment, please be sure to read the General Instructions that are on Piazza (under Resources->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 5 tables. This is all that is required in this assignment. 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 is **not** required in this assignment. (That will show up in the Lab2 assignment.)

3 Lab1 Description

3.1 Create PostgreSQL Schema Lab1

As we note on 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;
```

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.Customers, rather than just Customers). 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;
```

You may need to log out and log back in the server for this default schema change to take effect.

3.2 Tables

You will create tables in schema Lab1 for tables Persons, Houses, Landlords, Ownerships and Tenants as described below. Datatypes for the attribute in these tables are described in the next section.

Persons (<u>SSN</u>, Name, HouseId, ApartmentNumber, Salary)

Houses (<u>HouseID</u>, HouseAddress, ApartmentCount, Color)

Landlords (<u>LandlordID</u>, OwnerSSN, LandlordAddress)

Ownerships (<u>LandlordID, HouseID</u>, PurchaseDate, PropertyTax)

Tenants (<u>HouseID, ApartmentNumber</u>, LeaseTenantSSN, LeaseStartDate, LeaseExpirationDate, Rent, LastRentPaidDate, RentOverdue)

Important: To receive full credit, you must use the attribute names as given, and the attributes must be in the order given. Also, the datatypes must match the specifications given in the next section.

The underlined field (or fields) identifies the primary key of the table. The tables of the schema record information about persons and houses. Persons live in Houses (as Tenants in a house or apartment) and people also may own houses (as Landlords). The attribute SSN, which is the primary key of table Persons, is the Social Security number of the person.

In this assignment, you just have to create tables with the correct table names, attributes, datatypes and primary keys. But if you're curious about what's going on: Houses that have more than one apartment (indicated by ApartmentCount) are apartment houses; houses with only one apartment are single-dwelling homes. For a single-dwelling home, the tenant is in ApartmentNumber 1. Multiple persons may live together, but someone has to sign the lease (for the apartment or single-dwelling home), and that's indicated by LeaseTenantSSN. Landlords may own multiple houses.

3.2.1 Datatypes

- For SSN, OwnerSSN, TenantSSN, LeaseTenantSSN, ApartmentNumber and ApartmentCount and the HouseID and LandlordID attributes, use *integer*.
- For the name attribute of Persons, use *character* of fixed length 30.
- For HouseAddress, LandlordAddress and Color use *character* of variable length, with maximum length 40.
- The attributes for Rent, PropertyTax and Salary should be *decimal,* with at most 5 digits to the left of the decimal point and 2 decimal spaces after it.
- The attributes recording dates (i.e., PurchaseDate, LeaseStartDate and LeaseExpirationDate) should be of type *date*.
- For the RentOverdue attribute, which indicates whether the tenant has missed a rent payment, use *boolean*.

You will write a CREATE TABLE command for each of the five tables in Section 3.2. Write the commands in the order that the tables are listed above. Save the commands in the file create.sql You do not need any enumerated types in this assignment, just the datatypes mentioned above.

4 Testing

While you're working on your solution, it is a good idea to drop all objects from the database 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 datatype 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, to create the zip file 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 use your own student ID, not 1234567.)

3. Submit the zip file on Canvas under Assignment Lab1. If you are working on the UNIX timeshare and your zip file is located there, you will need to copy your file locally 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. As noted above, 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 promptly ask for help in the lab sections (do not postpone it for the day of the deadline). The computers at the Lab also have pre-installed SSH and FTP clients (Putty and PSFTP).

4. Lab1 is due by 11:59pm on Sunday, Jan 22. Late submissions will not be accepted, and there will be no make-up Lab assignments.