1. Preliminaries

Before starting on this assignment, please be sure to read the General Instructions that are on Canvas (under Files-->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 7 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, unless you put them in quotation marks, e.g.,, “Lab1”. But in CSE 180, 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 username 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;
```

3.2 Tables

You'll be creating tables for a very simplified version of a Highway Information database (which we'll call the Highway Info database), with tables for Highways, Exits, Interchanges, Cameras, Vehicles, Drivers and Photos. The data types and Referential Integrity for the attributes in these 7 Highway Info tables are described in the next section. No, the Highway Info database doesn't provide everything that a Highway Information 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; do not do more than you're asked to do in this assignment.

Highways(highwayNum, length, speedLimit)

Exits(highwayNum, exitNum, description, mileMarker, exitCity, exitState, isExitOpen)

Interchanges(highwayNum1, exitNum1, highwayNum2, exitNum2)

Cameras(cameraID, highwayNum, mileMarker, isCameraWorking)

Owners(ownerState, ownerLicenseID, name, address, startDate, expirationDate)

Vehicles(vehicleState, vehicleLicensePlate, ownerState, ownerLicenseID, year, color)

Photos(cameraID, vehicleState, vehicleLicensePlate, photoTimestamp)

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.

- A highway in **Highways** specifies the highway's number, length and speed limit.

- An exit in **Exits** describes an exit on a highway. Besides identifying the highway by giving the highway's number and direction, it also gives the exit number of the exit, a description of the exit, the mile marker of the exit (which is a distance on the highway), the city that the exit is in, and whether the exit is open.
    - Any highwayNum that appears in an exit row must appear as the highwayNum of a row in Highways.

- An interchange in Interchanges describes a place where two highways meet. An interchange specifies the exit on the first highway and the exit on the second highway where those two highways meet.
    - Any (highwayNum1, exitNum1) that appears in an interchange row must appear as the (highwayNum, exitNum) of a row in Exits.
    - Any (highwayNum2, exitNum2) that appears in an interchange row must appear as the (highwayNum, exitNum) of a row in Exits.

- A camera in Cameras specifies the ID of the camera, the highway on which the camera appears, the mile marker of the camera (which is a distance on the highway) and whether the camera is working.
  - Any highwayNum that appears in a camera row must appear as the highwayNum of a row in Highways.

- An owner in Owners specifies the state and license ID for that owner, as well as the name and address of the owner, and the start date and expiration date of the owner's license. (We're assuming that vehicle owners must be drivers.)

- A vehicle in Vehicles specifies the state and license plate of the vehicle, as well as the vehicle's owner (identified by the state and license ID of the owner), and the vehicle's year and color.
  - Any (ownerState, ownerLicenseID) that appears in a vehicle row must appear as the (ownerState, ownerLicenseID) of a row in Owners.

- A photo in Photos gives the ID of the camera which took the photo, as well as the timestamp when the photo was taken. It also gives the vehicle in the photo, identified by the state and license plate of the vehicle.
  - Any cameraID that appears in a photo row must appear as the cameraID of a row in Cameras.
  - Any (vehicleState, vehicleLicensePlate) that appears in a photo row must appear as the (vehicleState, vehicleLicensePlate) of a row in Vehicles.

[Although a real-world application would include a photograph attribute in the Photos table, we're not going to provides photographs in our Lab Assignments.]

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 highwayNum and mileMarker) 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 highwayNum, highwayNum1, highwayNum2, exitNum, exitNum1, exitNum2, cameraID, year and speedLimit, use *integer*.
- For exitState, vehicleState, ownerState and color, use *character with fixed length 2*. We'll explain the values of color later.
- For vehicleLicensePlate, use *character with fixed length 7*.
- For ownerLicenseID, use *character with fixed length 8*.
- For exitCity, use *character of variable length* with maximum length 20.
- For name, address and description, use *character of variable length*, with maximum length 60.
- mileMarker and length should be *numeric,* with at most 3 decimal digits to the left of the decimal point and 1 decimal digit after it.
- startDate and expirationDate should be of type *date*.
- photoTimestamp should be of type *timestamp*.
- isExitOpen and isCameraWorking should be *boolean*.

You must write a CREATE TABLE statement for each of the 7 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. Save your statements in the file create_lab1.sql

Note that PostgreSQL maps all SQL identifiers (e.g., table names and attributes) to lowercase. That's okay in your CSE 180 assignments. You won't lose points for Lab1 because Highways appears in the database as highways, and speedLimit appears as speedlimit. It is possible to specify case for identifiers by putting that identifier inside double-quote symbols, e.g., as "Highways". But then every time you refer to that identifier, you'll have to use the double-quotes. "HIGHWAYS" is not the same identifier as "Highways", and neither is the same as Highways, which PostgreSQL maps to lowercase, as highways. We use capitalization for readability (and suggest that you do the same), but we won't bother using double-quotes in our own Lectures and Lab Assignment solutions.

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. The following command, which you should put at the top of your create_lab1.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_lab1.sql script. As you'll learn in Lab Sections, the command to execute a script is: \i <filename> (You don't include the pointy brackets; <filename> means that you enter a specific filename.)

Verify that every table has been created by using the command: \d Note that when you execute \d, the tables that are displayed may appear 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 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 180. Enrolled students should automatically have access. 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's okay.) No students will be admitted to CSE 180 after the Lab1 due date.

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. To do 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 promptly 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 works well with for small files, but it's a hack that does not work well for large files.

The CSE 180 Teaching Assistants, Utkarsh Gupta, Tanuj Gupta and Gavin Cooke, 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 Tuesday, January 24. **Late submissions will not be accepted (Canvas won't take them, nor will we), and there will be no make-up Lab assignments.** Always 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.