Lab 1 : Introduction to SQL
CMSC 362
Marmorstein
Spring 2017
Due: Friday 02/10/2017 by 11:59:59pm

The purpose of this lab is to introduce you to the commands for creating an SQL table, populating it with data, and then querying it. In the first part of the lab, I will walk you through the creation of a simple table which stores information about missing animals. Then will ask you to create your own table which stores a list of "faculty evaluations". Finally, I will have you populate your database with some example data and execute a few queries against it.

Step 1. Setting Up
I have set up a postgresql database server on the CMSC lab server. I have created one database for each of you on that server. You also have an account which can access that database. The name of the database, your user name, and your password are all set to your last name (all lowercase). To connect to the server you can type:

psql -h torvalds -U lastName

This will dump you into a client program (sort of like an SQL interpreter) which lets you type SQL commands to be executed by the server. A better way to execute commands however is to put your SQL statements in a file and tell the server to execute them.

Create a folder named Lab1. In that folder create a file named "lab1.psql". All of your sql code will go in this file. At the top of your file *type two dashes followed by a space and then your name, the text "CMSC 362" and the words "Lab 1"*. Any line that starts with two dashes is considered a comment in SQL.

Save your file and quit. At the command prompt, type "*psql -h torvalds -U lastname -f lab1.psql*" to execute all of the commands in the file. Of course, since all you have in their right now is a comment, nothing will happen, but that's okay.

Step 2. Creating a Table

To create a new table in SQL, you use the "CREATE TABLE" command (surprised?). The syntax looks like this:

```
CREATE TABLE name (  attribute     type,
                     attribute     type,
                     attribute     type,
                     ... );
```

It is considered good style to line up your attributes on separate lines. If you don't use good style, it will still work. However, in this class you will **ALWAYS** line up your attributes so that I can read your code more easily.

The attribute is just a string like "first name". The type can be one of several values:

INTEGER	: stores integer values (surprise!)
CHAR(n)	: stores a string of n characters (*where n is a positive integer*)
VARCHAR(n)	: stores a null-terminated string of up to n characters
DOUBLE PRECISION	: a double precision floating point value
TIME	: stores a time value
DATE	: stores a date
INET	: stores a dotted-decimal IP address

There are many other data types you can store. For a complete list see
http://www.postgresql.org/docs/8.3/interactive/datatype.html

In lab1.psql add the following line to create a table:

```
CREATE TABLE lostandfound ( date_found     DATE,
                            kind_of_animal VARCHAR(20),
                            cage_number    INTEGER,
                            weight         DOUBLE PRECISION);
```

Now run "psql -h torvalds -U lastName -f lab1.psql" again.

If you try to run this command twice you will get an error the second time. Why? Because the table already exists! To solve this problem, you can add "DROP TABLE lostandfound;" to the top of your lab1.psql file directly beneath the comment. Of course, this will create errors if the table doesn't exist (if it's already been dropped or if it was never created). A solution is to use the special keyword IF EXISTS. Change the line to:

```
    DROP TABLE IF EXISTS lostandfound;
```

Step 2. Populating your database

You have now created a table which has four attributes. It is time to insert some data into the database.

We can populate the database using the INSERT INTO command.

Type:

INSERT INTO lostandfound (date_found, kind_of_animal, cage_number, weight) VALUES ('2008-12-04', 'dog', 3, 25.6);

There are a couple of things you should notice about this statement. First, we had to put single quotes around the value for the date and the string but not around the integer or floating point values. Second, we listed ALL of the attributes followed by the word VALUES and then listed the values IN THE SAME ORDER.

In SQL, you can list the attributes in any order you want as long as you make sure that the data values are ordered in the same way.

In fact, you can actually omit some attributes. The database will automatically populate them either with default values or with **NULL**s.

Step 3. Modifying the database

Let's add a default value to the cage_number attribute. Change your table so that the "cage_numer INTEGER" line looks like this:

cage_number   INTEGER   DEFAULT 1,

And run the psql command again.

Then add a NEW line:

INSERT INTO lostandfound   (kind_of_animal, date_found) VALUES ('cat', '2009-01-03');

As you will see in a minute, this will add a new entry with a cage value of 1 and a **NULL** value for the weight.

Step 4. Querying the database

It doesn't do us a lot of good to put data INTO the database if we can't get it out again. In order to QUERY data from the database we use the "SELECT" command. We will talk about the SELECT keyword in much more detail later, but for now all you need to know is that:

```
SELECT * FROM lostandfound;
```

 will give you a complete printout of the table.

Go ahead and add "SELECT * FROM lostandfound;" to the bottom of your lab1.psql file. Then rerun the psql command.

Step 5. On your own

I've held your hand through the first five steps. Now I want YOU to create a database table.

The schema for the table is:

evals(instructor_name : string,
       course_name : string,
       course_number : integer,
       student_id : string,
       helpfulness : integer,
       availability : integer,
       comment : string);

A. Use CREATE TABLE to set up the relation described above.
B. Populate it with at least 10 rows.
C. Execute a query that will show all of the information in the table.

You can add the SQL code for all of this directly to the bottom of your lab1.psql file.

Step 5. Submitting

Upload the lab1.psql file directly using the submit page at

```
http://narnia.homeunix.com/~robert/submit/
```

(you do not need to create a tarball of the Lab1 folder -- just submit the lab1.psql file directly).