CMPS182: General Information on lab assignments

Academic Integrity

No form of academic dishonesty will be tolerated. Incidents of academic dishonesty will be promptly reported according to the UCSC policy on academic integrity. Please see http://undergraduate.ucsc.edu/acd integrity/index.html if you need more details.

For Lab Assignments:

Policy for late lab assignments

- If your lab assignment is due by, say, 6pm on Jan 1, 2015, you will not be penalized if you submit your assignment before 6pm on Jan 1, 2015.
- Your final score for your lab assignment will be penalized by 50% if you submit by 6pm Jan 2, 2015.
- No lab assignments will be accepted after 6pm Jan 2, 2015. *There will not be make up lab assignments.*

PostgreSQL

- **PostgreSQL (lack of) backups:** PostgreSQL is not backed up, so anything you need long-term should be saved in the CATS file system. Some suggestions for general management of your database application are given below. CATS-Instructional Computing warns: *"Although system backups are performed, YOUR STUDENTS MUST BE SURE TO PERFORM THEIR OWN BACKUPS, as I will not restore anything from my own backups except in case of emergency and/or complete system failure."*

For the duration of the project, we suggest that you establish some kind of routine that includes reloading your database from the files created in this project part each time you want to get a "fresh" start with PostgreSQL. Remember to delete the contents of each relation (or destroy and recreate the relations) before reloading. Otherwise, unless there is a declared key (or you take APPEND out of your control file), PostgreSQL will happily append the new data to your old relation, causing your relation size to double, triple, quadruple, etc. To get rid of a table called R, issue the command:

```
drop table R;
```

If you want to get rid of all tuples in R without deleting the table itself, issue the command:

```
delete from R;
```

SQL on PostgreSQL:

PostgreSQL's implementation of SQL may differ somewhat from the SQL standard and also from the SQL that we cover in class based on the textbook. The PostgreSQL SQL compliance page provides more information.

General Information about PostgreSQL:

General information about PostgreSQL can be found here and postgreSQL documents can be found at here. In the frequently asked questions (FAQ), you will find links to nice tutorials under “How can I learn SQL?”.

Getting started

You may use the psql command line interface to interact with your data base. See Managing a Database for information on starting using psql. Script files are text files of psql commands which can be executed like a batch file using the the `\i` command in psql. The syntax is `\i filename` entered following the psql prompt, where file name is the complete (case sensitive) name of the script file you desire to run. To run your execution script `data.script` and save the script log in the file `data.log`, do the following:

`script data.log`	to start saving the script log
`psql -h [DATABASE SERVER HOSTNAME] -U [USERNAME]`	to run PostgreSQL's command line interface to log into your database
`\i data.script`	to import an execution script
`\q`	to exit pqsl (PostgreSQL's command line interface)
`exit`	to stop saving the script log

The execution script file you create can consist of most any series of commands which you could enter following the psql prompt. This includes all of the SQL commands which you will be using to create, modify and test your data base. Examples include the `CREATE TABLE` and `SELECT` commands. Just like when using the psql command line interface you must terminate each SQL command in your script file with a semicolon. `\i` and `\q` do not need semicolons. If you are recording your session using the script command described above into a script log then it is useful to start psql using the `-h` and `-U` options so that all commands included in your execution script file will be echoed to the console and thus to your script log file. Here is an example script file that creates a table (relation) named products:

```
     CREATE TABLE products (
       productID INT,
       name VARCHAR(80),
          price NUMERIC(10,2),
          retailPrice NUMERIC(10,2)
     );
```

Here is an example script file that loads four tuples into the table (relation) named products created using the previous script file:

```
     COPY products FROM stdin USING DELIMITERS '|';
     1419|American Greetings CreataCard Gold V4.0|21.49|25.24
     1424|Barbie(R) Nail Designer(TM)|20.74|25.99
     1427|Panzer Commander|21.99|30.24
     1431|Riven: The Sequel to Myst|31.99|40.24
     \.
```

This is the format you will use for the files that load data into your tables. The USING DELIMITERS '|' and the use of '|' as a delimiter is optional. The default delimiter is the tab character. The delimited data on each line must match the attributes and their types in your table in a one to one manner and in the order they were defined in your CREATE TABLE commands. The COPY data must be terminated with '\.' For testing, some students have found it convenient to have a separate script file to populate each of the tables. I also find it convenient have a single script file to create all of my tables and another one to drop all of my tables. Look for examples at the end of this document.

Sample Script Files

Note: These script files are for an example database that is not related to this project assignment. The example database records information on bars, customers, beers, and the associations among them.

createbeers.script

```
-- Sample Script file to Create a BEERS DB

-- print out the current time

SELECT timeofday();

-- Create Tables

CREATE TABLE Beers (
  name VARCHAR(30),
  manf VARCHAR(50)
);
CREATE TABLE Bars (
  name VARCHAR(30),
  addr VARCHAR(50),
  license VARCHAR(50)
);

CREATE TABLE Sells (
  bar VARCHAR(20),
  beer VARCHAR(30),
  price REAL
);

CREATE TABLE Drinkers (
  name VARCHAR(30),
  addr VARCHAR(50),
  phone CHAR(16)
);

CREATE TABLE Likes (
  drinker VARCHAR(30),
  beer VARCHAR(30)
);

CREATE TABLE Frequents (
  drinker VARCHAR(30),
  bar VARCHAR(30)
);

-- print out the current time

SELECT timeofday();
```

databeers.script

```
-- Sample Script file to Populate a BEERS DB

-- print out the current time

SELECT timeofday();

-- Populate the tables

COPY Beers FROM stdin USING DELIMITERS '|';
Coors|Adolph Coors
Coors Lite|Adolph Coors
Miller|Miller Brewing
```

```
Miller Lite|Miller Brewing
MGD|Miller Brewing
Bud|Anheuser-Busch
Bud Lite|Anheuser-Busch
Michelob|Anheuser-Busch
Anchor Steam|Anchor Brewing
\.

COPY Bars FROM stdin USING DELIMITERS '|';
Joe's|123 Any Street|B7462A
Sue's|456 My Way|C5473S
\.

COPY Sells FROM stdin USING DELIMITERS '|';
Joe's|Coors|2.50
Joe's|Bud|2.50
Joe's|Bud Lite|2.50
Joe's|Michelob|2.50
Joe's|Anchor Steam|3.50
Sue's|Coors|2.00
Sue's|Miller|2.00
\.

COPY Drinkers FROM stdin USING DELIMITERS '|';
Bill Jones|180 Saint St.|831-459-1812
Kelly Arthur|180 Alto Pl.|650-856-2002
Fred|1234 Fifth St.|831-426-1956
\.

COPY Likes FROM stdin USING DELIMITERS '|';
Bill Jones|Miller
Bill Jones|Michelob
Kelly Arthur|Anchor Steam
Fred|MGD \.

COPY Frequents FROM stdin USING DELIMITERS '|';
Bill Jones|Joe's
Bill Jones|Sue's
Kelly Arthur|Joe's
\.

-- print out the current time

SELECT timeofday();
```

querybeers.script

```
-- print out the current time
SELECT timeofday();
```

```
-- Execute some SELECT queries--
SELECT * FROM Bars;
SELECT * FROM Drinkers;

-- print out the current time
SELECT timeofday();
```

dropbeers.script

```
DROP TABLE Beers;
DROP TABLE Bars;
DROP TABLE Sells;
DROP TABLE Likes;
DROP TABLE Frequents;
DROP TABLE Drinkers;
```