General Information on Homeworks and Lab Assignments

CSE180, Winter 2020

Academic Integrity:

No form of academic dishonesty will be tolerated. You are encouraged to read the campus policies regarding academic integrity at https://www.ue.ucsc.edu/academic_misconduct. Violations may lead to penalties including (but not limited to) failing this course.

You are allowed to ask for some help when working on assignments, provided that you acknowledge the help that you received on the work that you turn in.

Points will be deducted if it appears that labor has been divided among multiple students; otherwise, there will be no penalty for small amounts of acknowledged assistance.

If you have any questions about these rules, please discuss them with the instructor immediately.

Gradiance homework assignments:

Non-lab homework assignments will be assigned via Gradiance, which has automated grading; see: http://www.gradiance.com/pub/stud-guide.html. Gradiance assignments may have different number of problems in them. You receive 3 points for each problem that you get right; you lose 1 point for each problem that you get wrong.

Go to http://www.gradiance.com/services, and use class code **FBCEFD41** to enroll in CSE 180 for Winter 2020. You can select your own User ID for Gradiance and Password for Gradiance; please remember your User ID. (Your cruzid is a good choice.) If you forget your Password, but remember your User ID, Gradiance can send your Password to the email address you used when you enrolled in Gradiance. Gradiance uses HTTP, rather than HTTPS, so be sure to use a password that you don't use for any other account!

If your Gradiance homework assignment is due by, say, 11:59pm on Friday, February 7, 2020, then you must submit (not just begin) your assignment by 11:59pm on Friday, February 7, 2020. *There will not be any make-up Gradiance assignments or extensions.*

You may redo a Gradiance assignment as often as you'd like before the deadline, waiting at least 10 minutes between submissions. The score on your **Final Submission** is the only one that counts, even if you received a higher score on an earlier submission.

(Ungraded) Practice Problem Sets:

Answer keys will not be provided. Some problems will be reviewed in class, but most Practice Problems won't be reviewed.

Lab Assignments:

The class will have a total of 4 lab assignments, as mentioned in the CSE 180 Syllabus. To work on the lab assignments, each student will be allotted an individual role (account) at the PostgreSQL server of the class and ownership to a blank database with the same name as the name of the role. The name of each student's account will be the same as the respective CruzID. The Teaching Assistants will send the password of each student's account individually through e-mail at the beginning of the term. You will be able to access the class server by issuing the following command at the shell prompt:

```
psql -h cse180-db.lt.ucsc.edu -U username
```

where `username` is your CruzID. If the command goes through, the server will prompt for your password. If you don't have PostgreSQL installed on your local machine, you can ssh to the UCSC Unix server unix.ucsc.edu with your CruzID and blue password, and then connect to our PostgreSQL server with the above command.

Once you are logged into the server and, you can change the password of your PostgreSQL account by issuing the following command at the server's command line interface:

```
ALTER ROLE username WITH PASSWORD 'newpassword';
```

A more secure way (hence the preferred way) to change the password of your PostgreSQL account is by using the command:

```
\password
```

and following prompts. If you change your password using \password, your new password is not stored in the command history or in the server log. That's why it's more secure than the ALTER ROLE command.

PostgreSQL Server

Use of schemas: For each lab assignment, you will create a schema to set apart the database tables created in the assignment from ones you created before and/or will create in the future, 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. See here for more details on PostgreSQL schemas. To create a schema for say, Lab1, you can use:

```
CREATE SCHEMA Lab1;
```

After you create the schema, you want to set it to be your default schema when you use psql. If you do not set it as the default schema, then you will have to qualify your table names with the schema name (e.g. Lab1.Products if you have a table with the name Products in the schema). To set the default schema, you modify your search path, as described in section 5.8.3 here:

```
ALTER ROLE username SET SEARCH_PATH TO Lab1;
```

Note: You need to log out and log in again, for this change to take effect. **Students often forget to do this.**

To verify your search path, use:

```
SHOW SEARCH_PATH;
```

PostgreSQL (lack of) backups: PostgreSQL is not backed up, so anything you need long-term should be saved in the Unix file system, not in PostgreSQL.

For the duration of the class, 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, you will get errors if the relation has a key, or you will have multiple copies of data if the relation does not have a key. To get rid of a table called R, at the line interface of the server issue the following 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;
```

If you want to drop a schema myschema, issue the command

```
DROP SCHEMA myschema CASCADE;
```

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 Conformance page provides more information.

General Information about PostgreSQL:
General information about PostgreSQL can be found here. This quarter, we're using PostgreSQL 11. PostgreSQL documents can be found here. In the wiki.postgresql.org Frequently Asked Questions (FAQ), you will find links to some reasonable tutorials under "How can I learn SQL?"

Getting started

You may use the psql command line interface to interact with your database. See psql PostgreSQL Client Application for information on 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 `filename` is the complete (case sensitive) name of the script file you wish to run. For example, if you only want to run an execution script named `myfile.sql`, you can do the following:

`psql -h cse180-db.lt.ucsc.edu -U username`	To run PostgreSQL's command line interface to log into your database
`\i myfile.sql`	To import the execution script myfile.sql
`\q`	To exit psql (PostgreSQL's command line interface)

Of course, you may run multiple scripts and commands within a psql session before you exit psql.

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 database. 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.

Here is an example script file that creates a table (relation) named products:

```
CREATE TABLE products (
 productID INT,
        name VARCHAR(80),
         price DECIMAL(10,2),
         retailPrice DECIMAL(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. It is also often convenient have a single script file to create all of my tables in an assignment, and another one to drop all of the tables (or drop the schema). Look for examples at the end of this document.

Sample Script Files

Note: These script files are for an example database that records information on bars, customers, beers, and the associations among them.

createbeers.sql

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

-- print out the current time

SELECT timeofday();

-- Create Tables

CREATE TABLE Beers (
beer VARCHAR(30),
manf VARCHAR(50),
PRIMARY KEY (beer)
);

CREATE TABLE Bars (
bar VARCHAR(30),
addr VARCHAR(50),
license VARCHAR(50),
PRIMARY KEY (bar)
);

CREATE TABLE Sells (
bar VARCHAR(20),
beer VARCHAR(30),    price REAL,
PRIMARY KEY (bar, beer),
FOREIGN KEY (bar) REFERENCES Bars,
FOREIGN KEY (beer) REFERENCES Beers
);
```

```sql
CREATE TABLE Drinkers (
drinker VARCHAR(30),
addr VARCHAR(50),
phone CHAR(16),
PRIMARY KEY (drinker)
);

CREATE TABLE Likes (
drinker VARCHAR(30),
beer VARCHAR(30),
PRIMARY KEY (drinker, beer),
FOREIGN KEY (drinker) REFERENCES Drinkers,
FOREIGN KEY (beer) REFERENCES Beers
);

CREATE TABLE Frequents (
drinker VARCHAR(30),
bar VARCHAR(30),
PRIMARY KEY (drinker, bar),
FOREIGN KEY (drinker) REFERENCES Drinkers,
FOREIGN KEY (bar) REFERENCES Bars
);

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

databeers.sql

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

-- print out current time; not required in your Lab Assignments

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 '|';
Joes|123 Any Street|B7462A
Sues|456 My Way|C5473S
\.

COPY Sells FROM stdin USING DELIMITERS '|';
Joes|Coors|2.50
Joes|Bud|2.50
Joes|Bud Lite|2.50
Joes|Michelob|2.50
Joes|Anchor Steam|3.50
Sues|Coors|2.00
Sues|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|Joes
Bill Jones|Sues
Kelly Arthur|Joes
\.

-- print out current time; not required in your Lab Assignments
SELECT timeofday();
```

querybeers.sql

```
-- 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.sql

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

Use of Canvas for CSE 180:

We use Canvas for submission of Lab Assignments and for grading of Lab Assignments and Exams. Login to Canvas at https://canvas.ucsc.edu using your CruzID and Gold password. CSE 180 should be one of the classes available. Info on Canvas is at http://its.ucsc.edu/canvas/index.html

To submit Lab Assignments, you'll make a zip file on unix that consists of the files required in the Lab, and you'll submit that zip file on Canvas. Info about how to transfer file to your computer so that you can submit on Canvas will appear in the Lab1 assignment file. You can also get help doing this in your Lab Section. Be sure to attend! If your lab assignment is due by, say, 11:59pm on Sunday, February 2, 2020, you must submit your assignment on Canvas before 11:59pm on Sunday, February 2, 2020. Please read the Lab assignments early and submit your solutions early; questions asked just before the deadline might not always be answered in a timely way. Solutions will not be graded until after the deadline, so you can revise your solution after submission. Only the last submission will be graded.

No lab assignments will be accepted after 11:59pm on the day of the deadline. *There will not be any extensions or make-up lab assignments.* **Please check that you've submitted the correct file on Canvas, and that the file submission is completed before 11:59pm on the due date.** Canvas will not accept the file after that.

Remember that you can help each other with concepts and bugs, but assignments must be done individually. Academic Integrity is taken very seriously. Violations may lead to penalties including (but not limited to) failing this course.

If you have any questions about your grades, please first ask your Teaching Assistant and then if you still have concerns, ask me. **Never** send questions or comments to the Readers who graded your homework. Unfortunately, Canvas shows the name of the Reader who graded, but the TAs and I are responsible for grading. Contact us, not the Readers if you disagree with a grade.

After the course ends, your course grade is determined by your scores on Exams, Labs and Gradiance, as described on the "Course Evaluation" slide that's in the Syllabus. You won't be able to do any additional work to improve your grade.