## PART I: Writing SQL Queries

Consider the following relation schemas:

> Student(*<u>sid</u>:* integer, *sname:* string, *major:* string, *standing:* string, *age:* integer)
> Class(*<u>name</u>:* string, *meets at:* string, *room:* string, *fid:* integer)
> Faculty(*<u>fid</u>*: integer, *fname:* string, *deptid:* integer)
> Enrolled(*<u>sid</u>:* integer, *<u>cname</u>:* string)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. A student's standing refers to a student's year, namely freshman FR, sophomore SO, junior JR, and senior SR. Write the following queries in SQL. No duplicates should be printed in any of the answers.

1. Find all Juniors (standing = JR) who are enrolled in a class taught by any faculty whose surname begins with the letter T. Print the students and faculty names.
2. For each level, print the level and the average age of students for that level.
3. For all levels except JR, print the level and the average age of students for that level.
4. For each faculty member that has taught classes only in room R128, print the faculty member's name and the total number of classes she or he has taught.

## PART II: Using PostgreSQL

1. Log into your account on our servers using <aid>@<aid>-db.qatar.cmu.local where *aid* is your andrew ID. Enter your Linux user account password when prompted.

2. You can either run a PostgreSQL Client from the command-line or use a web-based PostgreSQL Client such as phpPgAdmin (https://<aid>@<aid>-db.qatar.cmu.local) and Shell In A Box (https://<aid>-db.qatar.cmu.local:12320).

   We will continue the instructions on how to create and manipulate databases from the command-line.

3. Create a new empty database *Recitation 3*:

   ```
   createdb -U postgres -h <aid>-db.qatar.cmu.local Recitation3 ;
   ```

4. Connect to the database *Recitation 3*:

   ```
   psql -U postgres -h <aid>-db.qatar.cmu.local Recitation3 ;
   ```

5. Create 4 tables namely: Student, Faculty, Class, and Enrolled). We have written the SQL CREATE statements in a file under '/home/shared/Ex5.1/CreateTableScript-Ex5.1.txt'.

You can either copy each CREATE statement to the command prompt and hit enter or instruct Postgres to read all the CREATE statements from the file:

```
\i  ‘/home/shared/ Ex5.1/CreateTableScript-Ex5.1.txt’ ;
```

6. To check that the tables were created, use phpPgAdmin or type the command:

```
\c Recitation3 \dt ;
```

7. Populate the tables by inserting values into tuples. You can do this by writing INSERT statements on the command prompt. However, we have created 4 CSV (Comma Separated Values) files namely *student.csv*, *faculty.csv*, *class.csv*, and *enrolled.csv* under ‘/home/shared/Ex5.1’. The following command will copy the contents of each file to the corresponding table:

```
copy <table_name> from  ‘<filename>’  with CSV
copy student from  ‘/home/shared/ Ex5.1/Student.csv’  with CSV;
copy faculty from   ‘/home/shared/ Ex5.1/Faculty.csv’  with CSV;
copy class from   ‘/home/shared/ Ex5.1/Class.csv’  with CSV;
copy enrolled from   ‘/home/shared/ Ex5.1/Enrolled.csv’  with CSV;
```

8. Write your first query ever!

```
SELECT * FROM student
```

9. Type the queries from PART I and see the result sets.

10. Close the connection to the database: `\q`

## PART III: Writing Tuple Relational Calculus (TRC) and Domain Relational Calculus (DRC) Expressions

Consider the following relation schemas:

Sailor(*sid:* integer, *sname:* string, *rating:* integer, *age:* integer)
Boat(*bid:* integer, *bname:* string, *color:* string)
Reserves(*sid:* integer, *bid:* integer, *day:* date)

For each of the following, write-down the TRC and DRC expressions:

1. Find the names of sailors who have reserved at least two boats.
2. Find the names of sailors who have reserved all boats.