1 Using PostgreSQL

1. Make sure you have installed PostgreSQL (refer to online documentation if needed).

2. Access the terminal (e.g. on Windows, cmd or the psql terminal) (you can, if you wish, use whatever GUI available, but we will stick with the terminal for this recitation).

3. Create a new empty database called Recitation4:

   ```bash
   createdb -U postgres -h Recitation4 ;
   ```

4. Connect to the database Recitation4:

   ```bash
   psql -U postgres Recitation4 ;
   ```

5. Or, alternatively, you can login to Postgres and then access the database:

   ```bash
   psql -U postgres;
   \c Recitation4 # selects the Recitation4 database;
   ```

6. Under the database, create four tables; namely: Student, Faculty, Class, and Enrolled. You can either enter the SQL CREATE statements via the command-line or put them all in a file (aka a SQL script). For convenience, we have provided you with the file containing the SQL CREATE statements:

   ```
   https://web2.qatar.cmu.edu/~mhhammou/15415-s18/recitations/Recitation4.tgz
   ```

   Download, extract and keep note of the directory in which this is in.
7. Next, let’s import what’s in the txt file:

```
\i 'C:/Users/Tamim/Recitation4/CreateTableScript.txt'; # example
```

8. To check that the tables were created, we run the following command:

```
\dt;
```

9. Populate the tables by inserting tuples. Again, you may enter your SQL INSERT statements via the command-line. However, we shall show you how to import existing data into the tables. In the same tgz file above, there are four CSV (Comma Separated Values) files; namely: `student.csv`, `faculty.csv`, `class.csv`, and `enrolled.csv`, each of which should be imported into the respective table. To do so, use the copy command as exemplified below:

```
copy <table_name> from '<filename>' with CSV # EXAMPLE

copy student from 'C:/Users/Tamim/Recitation4/Student.csv' with CSV;
copy faculty from 'C:/Users/Tamim/Recitation4/Faculty.csv' with CSV;
copy class from 'C:/Users/Tamim/Recitation4/Class.csv' with CSV;
copy enrolled from 'C:/Users/Tamim/Recitation4/Enrolled.csv' with CSV;
```

10. Write your first query!

```
SELECT * from student;
```

11. Close the connection to the database:

```
\q;
```
2 Writing SQL Queries

Consider the following relation schemas:

\[
\begin{align*}
\text{Student} & \quad (\text{sid}: \text{integer}, \text{sname}: \text{string}, \text{major}: \text{string} \\
& \quad \text{standing}: \text{string}, \text{age}: \text{integer}) \\
\text{Class} & \quad (\text{name}: \text{string}, \text{meets\_at}: \text{string}, \text{room}: \text{string}, \text{fid}: \text{integer}) \\
\text{Faculty} & \quad (\text{fid}: \text{integer}, \text{fname}: \text{string}, \text{deptid}: \text{integer}) \\
\text{Enrolled} & \quad (\text{sid}: \text{integer}, \text{cname}: \text{string})
\end{align*}
\]

The meaning of these relations is straightforward. For example, Enrolled records student-class pairs such that the student is \textit{Enrolled} in the class. A student’s \textit{standing} refers to the student’s year, and can take on the values \texttt{FR} (Freshman), \texttt{SO} (Sophomore), \texttt{JR} (Junior), and \texttt{SR} (Senior).

Write SQL queries to achieve the following requirements. Note that no duplicates should be produced in any of the answers.

1. Find all Juniors who are enrolled in a class taught by any faculty whose surname begins with the letter \texttt{T}. Print the students and faculty names.
2. For all the standings except JR, print the standing and the average age of students in that group.

3. Find the names of all students who have a conflict; i.e., they are enrolled in two classes that meet at the same time.

4. Find the super hero student(s) i.e., the one(s) enrolled in the maximum number of classes. Print the student(s) name(s) and number of classes.