CS 3380 Lab Assignment 5

1 Directions

This assignment must be completed by **Sunday, October 5th at 11:59 PM**. You must submit a single `lab5.sql` text file that contains SQL commands based on the information that follows. You **should not** submit the `lab5data.sql` file. Late submissions, either for the files or the URL, will not be accepted.

2 Goals

- Creating SQL views
- Writing queries with sub-queries
- Using built-in SQL functions
- Writing queries with the SQL set operators
- Writing statements that have a `WITH` clause

3 Tasks

3.1 Download

Begin by downloading a SQL file needed for this assignment by executing the following commands in your terminal:

```
mkdir -p ~/cs3380/lab5
cd ~/cs3380/lab5
wget http://babbage.cs.missouri.edu/~klaricm/fs14/cs3380/lab5/lab5data.sql
```

Note that you might not be able to copy-paste the above commands. You may need to type them manually into your terminal.

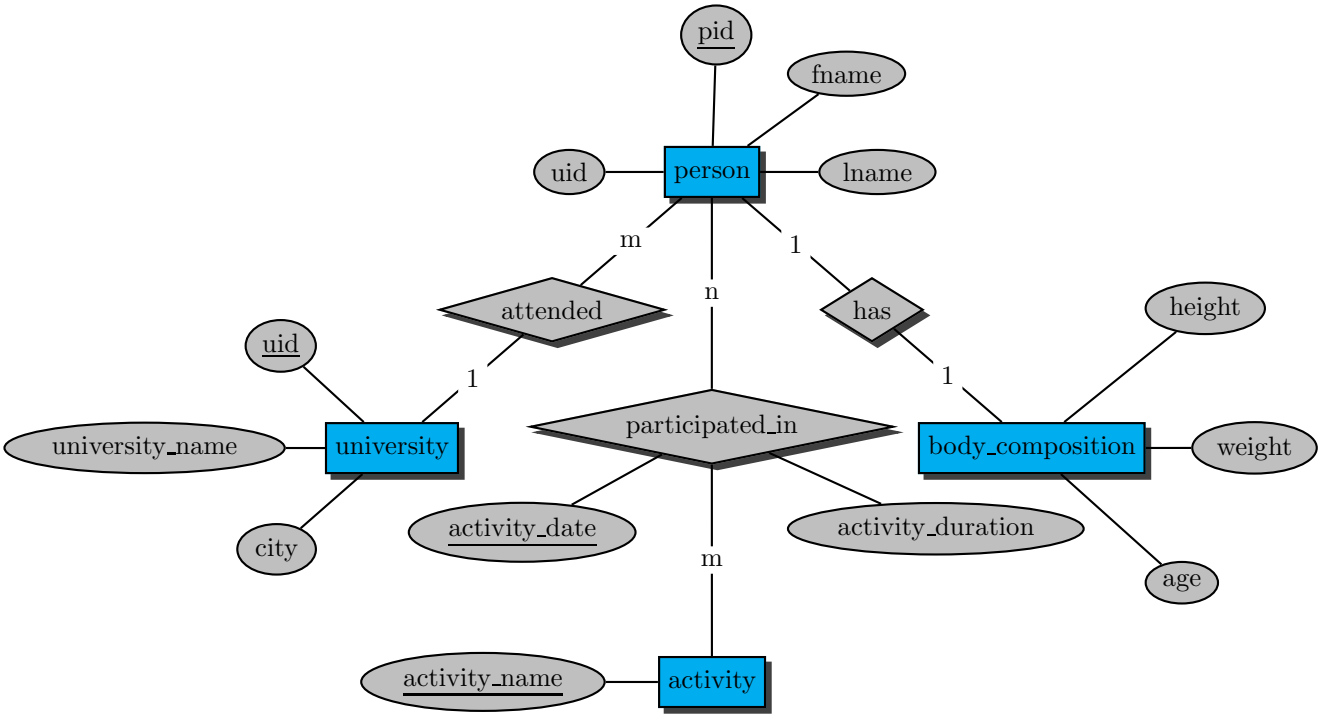
Next, run the `psql` command to login to your database. Then issue the command `\i lab5data.sql` to run the commands contained within the SQL file.

3.2 Inspect the Data

The `lab5data.sql` file will create five tables within the `lab5` schema. Recall, to write SQL queries that reference tables held within schemas simply qualifying the table name with the schema name. A simple example follows.

```
SELECT * FROM lab5.university;
```

An ERD for this dataset follows.



3.3 Implementation

Now, create and open the `lab5.sql` file in your editor in the `~/cs3380/lab5` directory. You are required to complete the following tasks.

Add
echo output #1...2...3 before each of your queries or you will have 5 points taken off! This is to help make it easier for the grader to read and interpret the output from your lab assignment.

For example:

Question 1:

```
\echo output #1
SELECT * FROM university;
```

Question 2:

```
\echo ouput #2
SELECT * FROM person;
```

Repeat this for all 10 questions.

3.3.1 Create a View: Part 1

Create a view that shows the person's id (pid), first name (fname) and last name (lname) for all people who have a body weight above 140. This view should be named "weight" (without the quotes). You must use an `INNER JOIN` in the views query.

3.3.2 Create a View: Part 2

Create a view that returns the first name (fname), last name (lname) and BMI for people with a weight above 150. This view should be named "BMI". You must use an `INNER JOIN` and you must reference the "weight" view created in 3.3.1. BMI is calculated as

$$703 \cdot \frac{weight}{height^2} \tag{1}$$

It is usually rounded to a whole number. For example, a person with a height of 71 inches and weight of 145 lbs would have a BMI of 20.2 which would be rounded to 20. Use an SQL function to achieve this rounded result.

3.3.3 Using `EXISTS`

Write a query that shows returns the name and city of the university that has no people in database that are associated with it. Your query **must** use `EXISTS` to achieve. *(2 rows)*

3.3.4 Using `IN`

Write a query that returns only the uid value for all universities in the city Columbia. Then use that query with an `IN` sub-query expression to retrieve the first and last names for all people that go to school in Columbia. *(4 rows)*

3.3.5 Using `NOT IN`

Write a query that returns all of the activities with records in the `participated_in` table. Then use that query with a `NOT IN` sub-query expression to retrieve the activities that are not played by any player in the database. *(2 rows)*

3.3.6 Using `UNION`

Write a query that returns the pid of all people listed in `participated_in` that participate in 'running'. Then modify your query to use `UNION` to return all people who run or play racquetball. You **must** use the `UNION` operator to accomplish this. You **cannot** use `OR`. *(5 Rows)*

3.3.7 Using `INTERSECTS`

Write a query that returns the first and last name of all people listed in `body_composition` table who are older than 30 years old. Then modify your query to use `INTERSECTS` to return all people who are older than 30 and are taller than 65 inches. You **must** use the `INTERSECTS` operator to accomplish this. You **cannot** use `AND`. *(3 rows)*

3.3.8 Using `ORDER BY`

Write a query that returns peoples first and last names weight, height, and age. Records should be ordered first by height in descending (Z-to-A order), then by weight in ascending order, and finally by the person's last name in ascending order. *(12 rows)*

3.3.9 Using a `WITH` clause: Part 1

First, write a query that returns the person's id (pid), first name (fname) and last name (lname) from all people who are from the people who go to the University of Missouri Columbia. Then, place that query in a WITH clause and use it as a common table expression (CTE) to combine the result with the `body_composition` table via an inner join to get the body compositions for people who attend the University of Missouri Columbia. *(2 rows)*

3.3.10 Using a `WITH` clause: Part 2

First, write a query that returns the person id (pid) of all people who are taller than 70 inches that do not attend the 'University of Missouri Columbia' (Hint: uid != 2). Then, place that query in a `WITH` clause and use it as a common table expression (CTE) to update all people in that set so that their university is now the 'University of Missouri Columbia'. In this query you're allowed to have a hard-coded uid = 2 for the university id.

Bonus: Modify your answer for the question in 3.3.10, so that it does not contain a hard-coded uid = '2' for the university id to update the person table. (4 rows)