Section 3 Worksheet

Part 1. Movies and Directors

```
CREATE TABLE Movie (
  movie_name VARCHAR(75),
  movie_id INT,
  director_id INT,
  year_released INT,
  budget INT,
  PRIMARY KEY(movie_id),
  FOREIGN KEY(director_id) REFERENCES Director(director_id)
);

CREATE TABLE Director (
  director_id INT,
  director_name VARCHAR(75),
  director_country VARCHAR(75),
  PRIMARY KEY(director_id)
);
```

1. **Find the id and name of all directors who have directed more than 20 movies.**
   - First, notice what the problem is asking of you. With SQL queries, the idea of how they are written is so that they read like a sentence. From the question, we know that we will [SELECT director_id and director_name].
   - We also see that we want the property of the count of movies associated with that director being > 20. Associating movies to a director lends itself naturally to categorization over director_id so a GROUP BY is probably needed. Also, the property we want to extract is over groups of movies associated with a director, so a HAVING clause is also needed.

```
SELECT D.director_id, D.director_name
  FROM Director AS D, Movie AS M
 WHERE D.director_id = M.director_id
 GROUP BY D.director_name, D.director_id
HAVING COUNT(*) > 20;
```

2. **For each director, find the corresponding movie that has the highest budget.**

- The trigger word here is "corresponding" which indicates a query that performs witnessing (i.e. a subquery of some sort will likely help).
- We can split this problem into sub-problems. First is to find the highest budget per director. Second is to find a movie that corresponds to that max. The first sub-problem is done in the virtual table MovieMaxBudget below. The second sub-problem should utilize what we have established. We should ask "how can I find a movie that matches the max budget (from the virtual table)?" We can just compare budgets of movies directed by director X with the max budget of director X.

```
-- This is a partial solution
WITH MovieMaxBudget AS
  (SELECT M.director_id AS director_id,
          max(M.budget) AS max_budget
     FROM Movie M
    GROUP BY M.director_id)
SELECT D.director_name, M.movie_name
  FROM Movie AS M, Director AS D, MovieMaxBudget AS MMB
 WHERE M.director_id = D.director_id AND
       D.director_id = MMB.director_id AND
       MMB.max_budget = M.budget;
```

- This question was phrased a bit poorly. A better question would be: "For each director, find [a] corresponding movie that has the highest budget."
- In this case, we do not care for which movie is picked. We can make a modification to the query above to generate 1 movie per director. We can do this by grouping on the director and arbitrarily picking a movie via an aggregate (e.g. max).

```
WITH MovieMaxBudget AS
  (SELECT M.director_id AS director_id,
          max(M.budget) AS max_budget
     FROM Movie M
    GROUP BY M.director_id)
SELECT D.director_name, max(M.movie_name)
  FROM Movie AS M, Director AS D, MovieMaxBudget AS MMB
 WHERE M.director_id = D.director_id AND
       D.director_id = MMB.director_id AND
       MMB.max_budget = M.budget;
 GROUP BY D.director_name;
```

Part 2: Classes and Instructors

```
CREATE TABLE Class (
  dept VARCHAR(6),
  number INTEGER,
  title VARCHAR(75),
  PRIMARY KEY (dept, number)
);

CREATE TABLE Instructor (
  username VARCHAR(8),
  fname VARCHAR(50),
  lname VARCHAR(50),
  started_on CHAR(10),
  PRIMARY KEY (username)
);

CREATE TABLE Teaches (
  username VARCHAR(8),
  dept VARCHAR(6),
  number INTEGER,
  PRIMARY KEY (username, dept, number),
  FOREIGN KEY (username) REFERENCES Instructor(username),
  FOREIGN KEY (dept, number) REFERENCES Class(dept, number)
);
```

1. **How many classes are being taught by at least one instructor?**
   - By the nature of our data, we know that any class that appears in Teaches must be taught by at least 1 teacher. Thus, if we categorize the tuples in Teaches by dept and number (the primary key of Class), we can get our answer by counting the number of groups.
   - The sticking point of this query is how to count the number of groups. One solution is to SELECT a constant (1) to correspond with a class group. Doing that, we can count the number of groups in a parent query.

```
SELECT count(*) AS class_count
  FROM (SELECT 1
          FROM Teaches
         GROUP BY dept, number ) X;
```

2. **Which instructors teach more than 1 class? Give the username, first name, and last name of these instructors.**

- There are few way of thinking about this query. One is that for each teacher we can see how many classes they teach. If you follow this thinking you can check the number of courses taught in the Teaches table in a subquery.

```
SELECT I.username, I.fname, I.lname
  FROM Instructor AS I
 WHERE 1 < (SELECT COUNT(*)
              FROM Teaches AS T
             WHERE T.username = I.username);
```

- As with most subquery unnesting problems, you should automatically think about if you can resolve the issue with a GROUP BY and HAVING clause. You'll notice in the above query that we are correlating the subquery on username. This lends itself nicely to a GROUP BY on username because we don't need to deal with information of tuples not with the specific instructor that we correlate on or group by.

```
SELECT I.username, I.fname, I.lname
  FROM Instructor AS I, Teaches AS T
 WHERE I.username = T.username
 GROUP BY I.username, I.fname, I.lname
HAVING COUNT(*) > 1;
```

3. **Which CSE courses do neither Dr. Levy (‘levy’) nor Dr. Wetherall (‘djw’) teach? Give the department, number, and title of these courses.**

- The framing of this question is a negated existential. This hints that a simple SELECT-FROM-WHERE query (monotonic) will not work.
- As a side note this problem can be formally translated into relational calculus/first-order logic (think CSE 311):
  - $T(c, t) := \text{t teaches c}$
  - $\neg\exists c.(T(c, \text{'levy'}) \vee T(c, \text{'djw'}))$
  - $\forall c.(\neg T(c, \text{'levy'}) \wedge \neg T(c, \text{'djw'}))$
- A gut reaction, if you think of filtering out tuples with levy or djw, might lead to the query below.

```
-- This query is wrong
SELECT C.dept, C.number, C.title
  FROM Class AS C, Teaches AS T
 WHERE C.dept = 'CSE' AND
       C.dept = T.dept AND
       C.number = T.number AND
       (T.username != 'levy' OR
        T.username != 'djw');
```

- The trick to this problem is that more than one instructor may teach a single course. For example, Dr. Levy might teach CSE 344 (so it shouldn’t be in the output), but another instructor may teach the course as well (e.g. Dan Suciu). With this example, the above query would incorrectly return CSE 344 because that specific tuple (Dan Suciu, …, CSE 344, …) passes.
- The negated existential can be translated to SQL via the NOT IN keywords.

```
-- Correct solution
SELECT *
  FROM Class AS C
 WHERE C.dept = 'CSE' AND
       C.number NOT IN (SELECT C.number
                          FROM Class AS C, Teaches AS T
                         WHERE C.dept = T.dept AND
                               C.number = T.number AND
                               (T.username = 'levy' OR
                                T.username = 'djw'));
```