CSE 344 Section 4 Worksheet

```
-- All classes, even ones that are not being taught this quarter.
-- Some classes may be taught by multiple instructors.
CREATE TABLE Class (
    dept      VARCHAR(50),  -- eg, "CSE" or "DATA
    coursenum INT,          -- eg, "344" or "514"
    title     VARCHAR(50),
    PRIMARY KEY (dept, coursenum));

-- All instructors, including ones that are not currently teaching.
-- Some instructors may teach more than one class in a single
quarter.
CREATE TABLE Instructor (
    username   VARCHAR(50) PRIMARY KEY,
    fname      VARCHAR(50),
    lname      VARCHAR(50),
    started_on CHAR(10));

-- This quarter's teaching assignments
CREATE TABLE Teaches(
    username  VARCHAR(50) REFERENCES Instructor,
    dept      VARCHAR(50),  -- FK into Class
    coursenum INT,          -- FK into Class
    FOREIGN KEY (dept, number) REFERENCES Class);
```

1. How many classes are currently 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 coursenum (the primary key), 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. The easy solution is to wrap the grouping query in a count(*) query.

```
SELECT COUNT(*)
  FROM (SELECT 1
          FROM Teaches
         GROUP BY dept, coursenum);
```

2. Which instructors currently teach more than 1 class? Give the username, first name, and last name of these instructors. Do NOT use a correlated subquery (although that is a good place to start).

There are a few ways 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 IN.username, IN.fname, IN.lname
  FROM Instructor IN
 WHERE 1 < (SELECT COUNT(*)
              FROM Teaches T
             WHERE T.username = IN.username);
```

This pattern lends itself nicely to a GROUP BY on username.

```
SELECT I.username, I.fname, I.lname
  FROM Instructor I, Teaches 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 (username ‘levy’) nor Dr. Wetherall (username ‘djw’) teach this quarter? 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 query) will not work.

A gut reaction, if you think of filtering out tuples with levy or djw, might lead to the query below.

This query is **wrong**! Imagine we have a course taught by levy. You can see that if we have a course taught by levy or djw with another instructor, that tuple will end up in the answer even though it shouldn’t

```
SELECT C.dept, C.coursenum, C.title
  FROM Class C, Teaches T
 WHERE C.dept = 'CSE'
   AND C.dept = T.dept
   AND C.coursenum = T.coursenum
   AND T.username <> 'levy'
   AND T.username <> 'djw';
```

The tricky part of this problem is that more than one instructor may teach a single course. But this problem can be solved with subqueries easily. A negated existential problem can be translated directly into SQL via the NOT IN keywords.

```
SELECT C.dept, C.coursenum, C.title
  FROM Class C
 WHERE C.dept = 'CSE'
       AND C.number NOT IN
             (SELECT T.coursenum
                FROM Teaches T
               WHERE (T.username = 'levy' OR T.username = 'djw')
                     AND T.dept = 'CSE');
```

Alternatively, you might take a different approach: to compute classes in CSE, then subtract those taught by levy or djw. This decorrelated version uses *set difference*.

```
SELECT C.dept, C.coursenum, C.title
  FROM Class C
 WHERE C.dept = 'CSE'
EXCEPT
SELECT C.dept, C.coursenum, C.title
  FROM Class C, Teaches T
 WHERE C.dept = 'CSE'
   AND C.dept = T.dept
   AND C.coursenum = T.coursenum
   AND (T.username = 'levy' OR T.username = 'djw');
```