Subqueries: Solutions

Schema

Student(<u>sID</u>, surName, firstName, campus, email, cgpa)   Offering[dept, cNum] ⊆ Course[dept, cNum]

Course(<u>dept, cNum</u>, name, breadth)   Took[sID] ⊆ Student[sID]

Offering(<u>oID</u>, dept, cNum, term, instructor)   Took[oID] ⊆ Offering[oID]

Took(<u>sID, oID</u>, grade)

Questions

1. What does this query do? (Recall that the || operator concatenates two strings.)

```
SELECT sid, dept || cnum as course, grade
FROM Took,
    (SELECT *
     FROM Offering
     WHERE instructor = 'Horton') Hoffering
WHERE Took.oid = Hoffering.oid;
```

   Solution: It finds information about students who took an offering taught by Horton. On our dataset, this is the output:

```
  sid  | course | grade
-------+--------+-------
 99132 | CSC343 |    79
 98000 | CSC343 |    82
 98000 | CSC263 |    78
 99999 | CSC343 |    89
   157 | CSC343 |    99
(5 rows)
```

2. What does this query do?

```
SELECT sid, surname
FROM Student
WHERE cgpa >
    (SELECT cgpa
     FROM Student
     WHERE sid = 99999);
```

   Solution: It finds information about students whose cgpa is higher than student 99999. On our dataset, this is the output:

```
  sid  |  surname
-------+------------
 99132 | Marchmount
```

```
    98000 | Fairgrieve
      157 | Lakemeyer
    (3 rows)
```

3. What does this query do?

```
SELECT sid, dept || cnum AS course, grade
FROM Took JOIN Offering ON Took.oid = Offering.oid
WHERE
    grade >= 80 AND
    (cnum, dept) IN (
            SELECT cnum, dept
            FROM Took JOIN Offering ON Took.oid = Offering.oid
                      JOIN Student ON Took.sid = Student.sid
            WHERE surname = 'Lakemeyer');
```

Solution: It finds information about students got an 80 or higher in a course that some Lakemeyer took. They did not have to take the course together.

4. (a) Suppose we have these relations: R(a, b) and S(b, c). What does this query do?

```
SELECT a
FROM R
WHERE b in (SELECT b FROM S);
```

Solution: It finds a values from R whose b occurs in S.

(b) Can we express this query without using subqueries?

Solution: You might think this query is equivalent:

```
SELECT a
FROM R, S
WHERE R.b = S.b
```

(Or we could do a natural join.) But they are not the same in all cases. If a tuple from R connects successfully with more than one tuple from S, this new query will yield duplicates that the original did not.

5. What does this query do?

```
SELECT instructor
FROM Offering Off1
WHERE NOT EXISTS (
    SELECT *
    FROM Offering
    WHERE
        oid <> Off1.oid AND
        instructor = Off1.instructor );
```

Solution: It finds instructors who have exactly one offering. On our dataset, this is the output:

```
 instructor
------------
```

```
Truta
Heap
Chechik
Davies
Johancsik
Reisman
Dow
Miller
Mendel
Richler
(10 rows)
```

6. What does this query do?
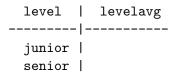
```
SELECT DISTINCT oid
FROM Took
WHERE EXISTS (
    SELECT *
    FROM Took t, Offering o
    WHERE
        t.oid = o.oid AND
        t.oid <> Took.oid AND
        o.dept = 'CSC' AND
        took.sid = t.sid )
ORDER BY oid;
```

Solution: It finds course offerings that include a student who has taken something else that is a CSC course. On our dataset, this is the output:

```
 oid
-----
   1
   3
   5
   6
   7
   8
   9
  11
  13
  14
  15
  16
  17
  21
  22
  26
  27
  28
  31
  34
```

```
    35
    38
    39
(23 rows)
```

7. Now let's write some queries! For each course, that is, each department and course number combination, find the instructor who has taught the most offerings of it. If there are ties, include them all. Report the course (eg "csc343"), instructor and the number of offerings of the course by that instructor.

   (a) First, create a view called Counts to hold, for each course, and each instructor who has taught it, their number of offerings.

   (b) Now solve the problem. Do not use any joins. (This will force you to use a subquery.)

8. Use EXISTS to find the surname and email address of students who have never taken a CSC course.

9. Use EXISTS to find every instructor who has given a grade of 100.

10. Let's say that a course has level "junior" if its cNum is between 100 and 299 inclusive, and has level "senior" if its cNum is between 300 and 499 inclusive. Report the average grade, across all departments and course offerings, for all junior courses and for all senior courses. Report your answer in a table that looks like this:

```
  level  |  levelavg
---------|-----------
  junior |
  senior |
```

Each average should be an average of the individual student grades, not an average of the course averages.