University of Toronto
CSC343, Winter 2017

SQL Exercises: Subqueries

Schema

Student(sID, surName, firstName, campus, email, cgpa)

Course(dept, cNum, name, breadth)

Offering(oID, dept, cNum, term, instructor)

Took(sID, oID, grade)

Offering[dept, cNum] $\subseteq$ Course[dept, cNum]

Took[sID] $\subseteq$ Student[sID]

Took[oID] $\subseteq$ Offering[oID]

Questions

1. What does this query do? (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;
```

2. What does this query do?

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

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');
```

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);
```

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

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 );
```

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;
```

7. Now let's write some queries! For each course 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. Suggestion: Use one or more views to hold intermediate step(s).

   Solution:

```
-- This intermediate result is helpful:
CREATE VIEW Counts as
SELECT dept || cnum as course, instructor, count(oid)
FROM Offering
GROUP BY cnum, dept, instructor;

-- Let's take a look at what this computes.
-- (Our dataset doesn't give this view a very good test.)
SELECT * from Counts;

 course | instructor | count
--------+------------+-------
 CSC148 | Miller     |     1
 CSC148 | Jepson     |     2
 EEB263 | Suzuki     |     1
 CSC343 | Mylopoulos |     2
 EEB216 | Suzuki     |     1
 ENG235 | Richler    |     1
 ENV200 | Suzuki     |     1
 EEB263 | Johancsik  |     1
 ENG235 | Percy      |     1
 HIS220 | Dow        |     1
 CSC343 | Horton     |     1
 CSC148 | Chechik    |     1
 EEB150 | Mendel     |     1
 CSC343 | Truta      |     1
 ENV320 | Suzuki     |     1
 ENG205 | Reisman    |     1
 HIS220 | Young      |     1
 ENG205 | Atwood     |     1
 CSC263 | Horton     |     2
 ENG110 | Atwood     |     1
 HIS296 | Young      |     1
 CSC207 | Gries      |     2
 ANT200 | Zorich     |     1
 ANT203 | Davies     |     1
 ENG110 | Percy      |     1
 ANT203 | Zorich     |     1
 CSC343 | Heap       |     1
 CSC320 | Jepson     |     2
 CSC207 | Craig      |     2
 CSC263 | Craig      |     1
(30 rows)

-- Now we can solve the problem using a subquery:
SELECT course, instructor, count
FROM Counts C1
WHERE count >= ALL (
      SELECT count
      FROM Counts C2
      WHERE C1.course = C2.course )
ORDER BY C1.course;

-- Here's another version:
SELECT course, instructor, count
FROM Counts C1
WHERE count = (
      SELECT max(count)
      FROM Counts C2
      WHERE C1.course = C2.course )
ORDER BY C1.course;

-- Here's what they both produce:

 course | instructor | count
--------+------------+-------
 ANT200 | Zorich     |     1
 ANT203 | Zorich     |     1
 ANT203 | Davies     |     1
 CSC148 | Jepson     |     2
 CSC207 | Craig      |     2
 CSC207 | Gries      |     2
 CSC263 | Horton     |     2
 CSC320 | Jepson     |     2
 CSC343 | Mylopoulos |     2
 EEB150 | Mendel     |     1
 EEB216 | Suzuki     |      1
 EEB263 | Suzuki     |      1
 EEB263 | Johancsik  |      1
 ENG110 | Atwood     |      1
 ENG110 | Percy      |      1
 ENG205 | Atwood     |      1
 ENG205 | Reisman    |      1
 ENG235 | Richler    |      1
 ENG235 | Percy      |      1
 ENV200 | Suzuki     |      1
 ENV320 | Suzuki     |      1
 HIS220 | Dow        |      1
 HIS220 | Young      |      1
 HIS296 | Young      |      1
(24 rows)
```

8. 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.

Solution:

```
CREATE VIEW Grades AS
SELECT cnum, dept, grade
FROM Offering natural join Took;

(SELECT 'junior' AS level, avg(grade) AS levelavg
FROM Grades
WHERE cnum >= 100 AND cnum <= 299)
   union
(SELECT 'senior' AS level, avg(grade) AS levelavg
FROM Grades
WHERE cnum >= 300 AND cnum <= 499);

 level  |      levelavg
--------+---------------------
 junior | 75.0952380952380952
 senior | 77.5000000000000000
(2 rows)
```