Subqueries: Solutions

Schema

Student(sID, surName, firstName, campus, email, cgpa)  Offering(dept, cNum) ⊆ Course(dept, cNum)
Course(dept, cNum, name, breadth)                      Took(sID) ⊆ Student(sID)
Offering(oID, dept, cNum, term, instructor)           Took(oID) ⊆ Offering(oID)
Took(sID, oID, grade)

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') Hoffer;
WHERE Took.oid = Hoffer.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
```

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:
3. What does this query do?

```sql
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. On our dataset, this is the output:

```
sid | course | grade
-----+--------+-------
157  | CSC343  | 99
99999| CSC343  | 89
98000| CSC343  | 82
. . . rows omitted
99132| ANT203  | 82
98000| ENG235  | 92
(24 rows)
```

Output:

XXX

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

```sql
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:

```sql
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:

<table>
<thead>
<tr>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truta</td>
</tr>
<tr>
<td>Heap</td>
</tr>
<tr>
<td>Chechik</td>
</tr>
<tr>
<td>Davies</td>
</tr>
<tr>
<td>Johancsik</td>
</tr>
<tr>
<td>Reisman</td>
</tr>
<tr>
<td>Dow</td>
</tr>
<tr>
<td>Miller</td>
</tr>
<tr>
<td>Mendel</td>
</tr>
<tr>
<td>Richler</td>
</tr>
</tbody>
</table>

(10 rows)

6. 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:

CREATE VIEW Counts as
SELECT cnum || dept 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 query a very good test.)
SELECT * from Counts;

<table>
<thead>
<tr>
<th>course</th>
<th>instructor</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>148CSC</td>
<td>Miller</td>
<td>1</td>
</tr>
<tr>
<td>263EEB</td>
<td>Suzuki</td>
<td>1</td>
</tr>
<tr>
<td>216EEB</td>
<td>Suzuki</td>
<td>1</td>
</tr>
<tr>
<td>235ENG</td>
<td>Richler</td>
<td>1</td>
</tr>
<tr>
<td>205ENG</td>
<td>Atwood</td>
<td>1</td>
</tr>
<tr>
<td>110ENG</td>
<td>Percy</td>
<td>1</td>
</tr>
<tr>
<td>148CSC</td>
<td>Jepson</td>
<td>2</td>
</tr>
</tbody>
</table>

SELECT course, instructor, count
FROM Counts C1
WHERE count >= ALL (
7. 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:

<table>
<thead>
<tr>
<th>level</th>
<th>levelavg</th>
</tr>
</thead>
<tbody>
<tr>
<td>junior</td>
<td></td>
</tr>
<tr>
<td>senior</td>
<td></td>
</tr>
</tbody>
</table>

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

Solution:

```sql
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 'junior' AS level, avg(grade) AS levelavg
FROM Grades
WHERE cnum >= 300 AND cnum <= 499);
```

Output:

<table>
<thead>
<tr>
<th>level</th>
<th>levelavg</th>
</tr>
</thead>
<tbody>
<tr>
<td>junior</td>
<td>75.0952380952380952</td>
</tr>
<tr>
<td>junior</td>
<td>77.5000000000000000</td>
</tr>
</tbody>
</table>
(2 rows)