

CDF IDs:

Names:

Subqueries: Solutions

Schema

Student(<u>sID</u> , surName, firstName, campus, email, cgpa)	Offering[dept, cNum] \subseteq Course[dept, cNum]
Course(<u>dept</u> , cNum, name, breadth)	Took[sID] \subseteq Student[sID]
Offering(<u>oID</u> , dept, cNum, term, instructor)	Took[oID] \subseteq Offering[oID]
Took(<u>sID</u> , <u>oID</u> , 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') Hofferings
WHERE Took.oid = Hofferings.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. 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?

```
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. 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;

```

course	instructor	count
148CSC	Miller	1
263EEB	Suzuki	1
216EEB	Suzuki	1
235ENG	Richler	1
205ENG	Atwood	1
110ENG	Percy	1
148CSC	Jepson	2

```

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

```

```

SELECT count
FROM Counts C2
WHERE C1.course = C2.course )
ORDER BY C1.course;

```

== OR ==

```

SELECT course, instructor, count
FROM Counts C1
WHERE count = (
    SELECT max(count)
    FROM Counts C2
    WHERE C1.course = C2.course )
ORDER BY C1.course;

```

course	instructor	count
110ENG	Percy	1
110ENG	Atwood	1
148CSC	Jepson	2
150EEB	Mendel	1
200ANT	Zorich	1

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:

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

```

Output:

level	levelavg
junior	75.0952380952380952
junior	77.5000000000000000

(2 rows)