Solution to Homework 5

Problem 1. a. List all courses that are taught by professors who belong to the EE or MGT department.

Answer:

- *Algebra*:

$$\pi_{\texttt{CrsCode,Semester}}(\sigma_{\texttt{DeptId='EE' OR DeptId='MGT'}}(\textsc{Professor}) \bowtie_{\texttt{Id=ProfId}} \textsc{Teaching})$$

- *SQL*:

```
SELECT   T.CrsCode, T.Semester
FROM     TEACHING T,  PROFESSOR P
WHERE    T.ProfId=P.Id
         AND (P.DeptId = 'EE' OR P.DeptId = 'EE')
```

b. List the names of all students who took courses *both* in spring 1997 and fall 1998.

Answer:

- *Algebra*: Let SEMS be a relation with `Semester` as the only column and ⟨'S1997'⟩ and ⟨'F1998'⟩ as the only tuples. Then the answer is:

$$\pi_{\texttt{Name}}(\textsc{Student} \bowtie_{\texttt{Id=StudId}} \pi_{\texttt{StudId,Semester}}(\textsc{Transcript})/\textsc{Sems})$$

- *SQL*:

```
SELECT   S.Name
FROM     STUDENT S
WHERE    NOT EXISTS (
              (SELECT * FROM SEMS)
              EXCEPT
              (SELECT T.Semester
               FROM TEACHING T
               WHERE T.StudId = S.Id ) )
```

Note that if the query required only the students who took courses in either of the two semesters, then the solution would not have required the division operator and NOT EXISTS and it would have been much simpler.

c. List the names of all students who took courses from at least two professors in different departments.

Answer:

- *Algebra*: We construct the requisite expression in two stages. First, we construct STUDDEPT(StudId, DeptId) as

$$\pi_{\texttt{StudId,DeptId}}(\textsc{Transcript} \bowtie \textsc{Teaching} \bowtie_{\texttt{ProfId=Id}} \textsc{Professor})$$

Tuples in this relation represent instances when a given student took a course from a professor in the given department. Constructing the final answer is now easy:

$$\pi_{\texttt{Name}}(\textsc{Student} \bowtie_{\texttt{Id=StudId}} \sigma_{\texttt{DeptId}\neq\texttt{DeptId2}}(\textsc{StudDept} \bowtie \textsc{StudDept}[\texttt{StudId},\texttt{DeptId2}]))$$

- *SQL*: The SQL solution is also in two stages, for clarity. First, we create an auxiliary view:

```
CREATE VIEW   StudDept(StudId, DeptId) AS
    SELECT    R.StudId, P.DeptId
    FROM      Transcript R, Teaching T, Professor P
    WHERE     R.CrsCode = T.Crscode
              AND R.Semester = T.Semester
              AND T.ProfId = P.Id

SELECT    S.Name
FROM      Student S, StudDept SD1, StudDept SD2
WHERE     S.Id = SD1.StudId AND SD1.StudId = SD2.StudId
          AND SD1.DeptId ≠ SD2.DeptId
```

d. List all courses that are offered by department MGT and that have been taken by all students.

Answer:

- *Algebra*:

$$\pi_{\texttt{CrsCode},\texttt{StudId}}(\sigma_{\texttt{DeptId}=\texttt{'MGT'}}\textsc{Course} \bowtie \textsc{Transcript}) \;/\; \pi_{\texttt{StudId}}(\textsc{Student})$$

- *SQL*:

```
SELECT    C.CrsCode
FROM      Course C
WHERE     C.DeptId = 'MGT' AND
          NOT EXISTS (
               (SELECT   S.StudId
                FROM     Student S)
                EXCEPT
               (SELECT   T.StudId
                FROM     Transcript T
                WHERE    C.CrsCode = T.CrsCode) )
```

* e. Find every department that has a professor who taught all courses ever offered by that department.

Answer:

- *Algebra*: The algebraic solution is trickier than SQL's. Let ProfCrsAux(ProfId, CrsCode) be defined as follows:

$$\pi_{\texttt{ProfId},\texttt{CrsCode}}(\textsc{Professor} \bowtie_{\texttt{DeptId}\neq\texttt{DeptId}} \textsc{Course})$$

A tuple, $\langle p, c \rangle$ here means that c is a course that is offered by a department *other than* the one where p works. Next, consider PROFCRS(ProfId, CrsCode), which is defined as follows:

$$\text{PROFCRSAUX} \cup \pi_{\texttt{ProfId,CrsCode}}(\text{TEACHING})$$

A tuple, $\langle p, c \rangle$ in this relation means that c is a course that was either taught by p or it is offered by a department *other than* the one where p works.

Finally, the expression for the requisite query can be constructed as follows:

$$\text{PROFCRS} \;/\; \pi_{\texttt{CrsCode}}(\text{COURSE})$$

Note that constructing PROFCRSAUX was essential: The expression

$$\pi_{\texttt{ProfId,CrsCode}}(\text{TEACHING}) \;/\; \pi_{\texttt{CrsCode}}(\text{COURSE})$$

is *not* correct, because it finds every professor who have taught every course in every department rather than the courses that are offered by the department where that professor works.

The solution given above uses the division operator. There is a more straightforward solution where one would first find professors who didn't teach one of the courses offered by the professor's department, and then take the complement.

- *SQL*:

```
SELECT   P.DeptId
FROM     PROFESSOR P
WHERE    EXISTS (
         (SELECT  P2.Id
          FROM    PROFESSOR P2
          WHERE   NOT EXISTS (
                       (SELECT   C.CrsCode
                        FROM     COURSE C
                        WHERE    C.DeptId = P2.DeptId )
                       EXCEPT
                       (SELECT   T.CrsCode
                        FROM     TEACHING T
                        WHERE    T.ProfId = P2.Id ) )
                  AND  P.Id = P2.Id
```

In fact, the following simplified query is also correct:

```
SELECT   P.DeptId
FROM     PROFESSOR P
WHERE    NOT EXISTS (
              (SELECT   C.CrsCode
               FROM     COURSE C
               WHERE    C.DeptId = P.DeptId )
```

Problem 2.

a. Find the average age of students who received an A for *some* course.

Answer:
Note: the following is *incorrect*, because some students' ages might be counted more than once:

```
SELECT    AVG(S.Age)
FROM      STUDENT S, TRANSCRIPT T
WHERE     S.Id =T.StudId AND T.Grade = 'A'
```

The correct solution requires a subquery with a DISTINCT clause:

```
SELECT    AVG(S.Age)
FROM      STUDENT S
WHERE     S.Id  IN
     (SELECT DISTINCT  S.Id
      FROM     STUDENT S, TRANSCRIPT T
      WHERE    S.Id =T.StudId AND T.Grade = 'A' )
```

b. Find the minimum age among straight A students *per course*.

Answer:

```
SELECT    T.CrsCode, MIN(S.Age)
FROM      STUDENT S, TRANSCRIPT T
WHERE     S.Id = T.StudId AND
          NOT EXISTS (SELECT T.Grade
                      FROM TRANSCRIPT T
                      WHERE S.Id = T.StudId AND T.Grade ≠
GROUP BY T.CrsCode
```

c. Find the minimum age among straight A students per course among the students who have taken CS305 or MAT123. (Hint: a HAVING clause might help.)

Answer:
Add the following HAVING clause to the above query:

```
HAVING    S.CrsCode IN ('CS305','MAT123')
```

d. Raise by 10% the salary of every professor who is younger than 40 and had taught MAT123 in the spring 1997 or fall 1997 semester. (*Hint*: Try a nested subquery in the WHERE clause of the UPDATE statement.)

Answer:

```
UPDATE   PROFESSOR
SET      Salary = Salary * 1.1
WHERE    Id IN
             (SELECT   P.Id
              FROM     PROFESSOR P, TEACHING T
              WHERE    P.Id = T.ProfId
                  AND P.Age < 40
                  AND T.CrsCode = 'MAT123'
                  AND (T.Semester = 'S1997'
                          OR T.Semester = 'F1997') )
```

e. Find the professors whose salary is at least 10% higher than the average salary of all professors.

Answer:

```
CREATE VIEW   AVGSAL(Avgsal) AS
     SELECT    AVG(P.Salary)
     FROM      PROFESSOR P
```

The actual query that uses the above view:

```
SELECT    P.Id
FROM      PROFESSOR P, AVGSAL A
WHERE     P.Salary > 1.1 * A.Avgsal
```

f. Find all professors whose salary is at least 10% higher than the average salary of all professors *in their departments*.

Answer:

```
CREATE VIEW   DEPTAVGSAL(DeptId, Avgsal) AS
     SELECT    P.DeptId, AVG(P.Salary)
     FROM      PROFESSOR P
     GROUP BY P.DeptId
```

The actual query, which uses the above view:

```
SELECT    P.Id
FROM      PROFESSOR P, DEPTAVGSAL D
WHERE     P.DeptId = D.DeptId AND P.Salary > 1.1 * D.Avgsal
```