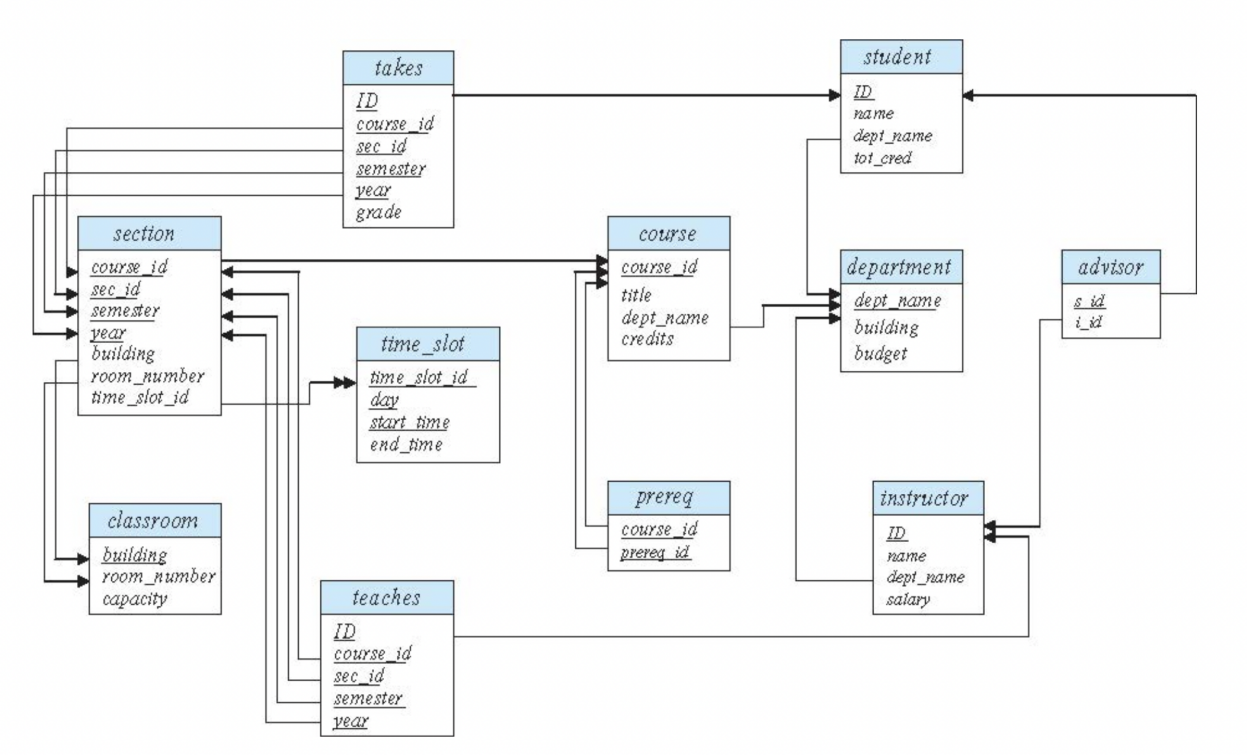
[Year-Dream] SQL PJT_Solution



3. 다음 쿼리들을 모두 SQL문으로 작성해봅시다.
   (각 쿼리들은 주석으로 번호를 표시하고, 최종적으로 하나의 파일에 기록해주세요)
   (단, 쿼리의 결과가 없을 수도 있습니다. 에러가 아니고, 만족하는 데이터가 없는 것입니다.)

   a. **Computer Science 학부 과목을 한 과목이라도 수강하고 있는 모든 학생의 ID와 이름을 출력하세요.**
   (단, 중복을 제거하여 출력해주세요)

   ▼ 설명

   현재는 가장 최신학기를 기준으로합니다. (2010년 가을학기) takes 테이블을 기준으로 course 테이블과 student 테이블을 left JOIN합니다. takes와 student는 ID를 공유하므로, ID를 기준으로 JOIN을 수행합니다. 이렇게 조인한 테이블에 다시 course 테이블을 left JOIN합니다. 이 때는 course_id를 공유하므로 course_id를 기준으로 JOIN합니다. 모두 JOIN한 테이블을 기준으로 최신학기에 컴퓨터공학 과목을 듣고 있는 학생의 ID와 이름을 출력합니다.

이 때, 각 테이블을 중간에 rename하는 이유는, on의 조건에서 ID와 course_id를 각 테이블에서 구분하기 위함입니다.

```
# a.
select distinct S.ID, S.name as a
from takes as T
left join student as S
  on S.ID = T.ID
left join course as C
  on C.course_id = T.course_id
where C.dept_name = 'Comp. Sci.' and T.semester = 'Fall' and T.year = 2010;
```

b. “CS-001”를 id로 가지고, “Weekly Seminar”가 title인 1학점짜리 과목을 새로 신설해보세요.

▼ 설명

course 테이블에 column 순서대로 tuple을 생성하여 추가합니다. CS-001 과목이니, Computer Science 학부 과목으로 추정할 수 있습니다.

```
# b.
insert into course values ('CS-001', 'Weekly Seminar', 'Comp. Sci.', 1);
```

c. Computer Science 학부생 전원을 sec_id가 1인 2007년도 가을학기 섹션에 등록시켜주세요.
(HINT. University ERD도 참고해보세요)

▼ 설명

가장 어려운 케이스였을텐데, 전체적인 스텝을 2가지로 구분합니다. 먼저 Computer Science 학부 전원을 찾는 것이 첫번째이고, 해당 subquery의 결과(table)을 insert 하는 것이 그 다음으로 진행됩니다. sec_id가 1인 section에 등록을 하기 위해서는 새로운 section이 필요합니다. 임의의 section을 하나 만들기 위해서 임의의 course를 지정하여 추가합니다. 다른 것들은 정의되지 않았기 때문에, null값으로 지정합니다. (course_id는 PK이므로 null일 수 없어 임의로 지정하였습니다. null value로 지정이 가능하다면, null로 지정해도 상관없습니다.)
이제 해당 섹션으로 등록(takes)하는 과정은 dept_name이 “Comp. Sci.”인 모든 학생의 정보를 가져와서 insert절을 수행합니다.

```
##takes: (ID, course_id, sec_id, semester, year, grade)
##section: (course_id, sec_id, semester, year, building, room_number, time_slot_id)
# c.
insert into section values ('CS-001', 1, 'Fall', 2007, null, null, null);
insert into takes
  select ID, 'CS-001', 1, 'Fall', 2007, null
    from student
    where dept_name = 'Comp. Sci.';
```

d. 각 학부별로 가장 많은 연봉을 가진 instructor중에서 가장 연봉이 낮은 instructor의 name과 dept_name을 찾아주세요.

▼ 설명

전형적인 subquery를 사용하는 문제입니다. 학부별로 가장 많은 연봉을 가진 instructor를 먼저 구한 뒤, 그 중에 가장 낮은 사람의 name과 dept_name을 select합니다. 각 학부별 연봉을 구하기 위해서 group by를 사용하고, 최대 연봉을 구하기 위해서 max 함수를 사용합니다. 그리고 subquery의 결과인 result 테이블에서 가장 작은 값을 뽑아서, 그 때의 name과 dept_name을 출력합니다.

```
# d.
select name, dept_name, min(max_salary) as d
from (select dept_name, max(salary) as max_salary
    from instructor
    group by dept_name) as result;
```

e. 2008년도 가을학기에 수업을 진행하는 모든 instructor를 찾아주세요.

▼ 설명

수업을 진행하는 모든 강사를 찾기 위해 instructor 테이블과 teached 테이블을 사용합니다. 두 테이블을 inner join하면 가르치고 있는 모든 강사를 찾을 수 있고, 그 중에 2008년 가을학기 조건을 가진 데이터를 찾습니다.

```
# e.
select distinct name as e
from instructor inner join teaches using (ID)
where year = 2008 and semester = 'Fall';
```

f. 2007년도 기준 물리학부에 재학중인 모든 학생을 찾아주세요.

▼ 설명

쿼리 e와 비슷한 예시입니다. 물리학부에 재학중인 정보를 찾기 위해 student 테이블과 takes 테이블을 사용합니다.

```
# f.
select distinct name as f
from student inner join takes using (ID)
where year = 2007 and dept_name = 'Physics';
```

e. 컴퓨터공학 강사 Lee가 담당하고 있는 학생들 중 다른 학부 학생을 모두 찾아주세요.

▼ 설명

쿼리 a와 비슷한 예시입니다. advisor 테이블을 기준으로 student와 instructor 테이블을 JOIN합니다. 조인한 최종 테이블에서 컴퓨터공학 소속인 Lee라는 이름의 강사와, 컴퓨터공학 소속이 아닌 학생들을 찾아서 이름을 출력해줍니다.

```
# g.
select distinct S.name as student_name
from advisor as A
left join student as S
  on S.ID = s_ID
left join instructor as I
  on i_ID = I.ID
where I.name = 'Lee' and I.dept_name = 'Comp. Sci.' and S.dept_name <> 'Comp. Sci.';
```

f. 이 때까지 한번도 과목을 수강한 적이 없는 모든 학생의 ID와 name을 출력해주세요.

▼ 설명

과목의 수강 기준은 takes 테이블에 있습니다. 이 때까지는 최신을 기준으로 하기 때문에, 2010년 가을을 기준으로 하고, 현재 student 테이블에 있지만 2010년 이전에 들었던 기록에 없는 학생들을 찾아줍니다.

```
# h.
select distinct ID, name as h
from student
where ID not in (
  select ID
    from takes
    where year < 2010);
```