CS1555 Recitation 6 Solution

Objective: To practice more SQL queries on Postgres.

Consider the following relation schemas:

Student (SID, Name, Class, Major, SSN)
Student_Dir (SID, Address, Phone)
FK: (SID) → Student (SID)
Course (Course_No, Name, Course_Level)
Course_taken (Course_No, SID, Term, Grade)
FK: (Course_No) → Course (Course_No); (SID) → Student (SID)

Write a SQL query for each of the queries below:

1. List the student ID and course number for every student who took a course in Fall 18 but has not received a grade yet.

```
select ct.sid, ct.course_no
from course_taken ct
where ct.term = 'Fall 18'
        and ct.grade is null;
```

2. List the SIDs and names of all the students and the number of courses they have taken.

```
select s.sid, s.name, count(distinct course_no) as num_courses
from student s, course_taken ct
where s.sid = ct.sid
group by s.sid, s.name;
```

3. List the SIDs, names and GPAs of the students whose GPAs are greater than 3.7. List them in the descending order of the GPAs.

```
select s.SID, s.name, avg(grade) as GPA
from course_taken ct join student s on ct.sid = s.sid
group by s.sid, s.name
having avg(grade) > 3.7
order by GPA desc;
```

4. List the SIDs and names of the students who have not taken the course “Operating Systems”.

--Solution 1: using set difference
```
select sid, name
from student
where sid not in (select sid from course_taken ct, course c
        where ct.course_no = c.course_no
            and c.name = 'Operating Systems');
```

--Solution 2: equivalently, you can use the "exists" operator as follows:
```
select s.sid, s.name
from student s
where not exists (select * from course_taken ct, course c
        where ct.course_no = c.course_no
            and c.name = 'Operating Systems'
            and ct.sid = s.sid);
```

--Solution 3: using outer join
```
select s.sid, s.name
from student s left outer join (select sid, course_no
        from course_taken ct natural join course
        where name = 'Operating Systems') os_taking
on s.sid = os_taking.sid
where course_no is null;
```

5. Find the SID(s) of the student(s) who has(have) the highest GPA.

--Solution 1: following the intuition: the students that has highest GPA are those whose GPA is greater than or equal to all of the students. The condition "avg(ct2.grade) is not null" is used to avoid the issue with null comparison (any comparison with one operand being null returns false regardless of the other operand). Note that multiple tuples will be returned if there are multiple students whose GPA are the highest.

```
select ct1.SID
from course_taken ct1
group by (ct1.sid)
having avg(ct1.grade) >= all (select avg(ct2.grade)
                        from course_taken ct2
                        group by (ct2.sid)
                        having avg(ct2.grade) is not null);
```

--Solution 2: following the reasoning as we did in the recitation of Relational Algebra:

```
select ct1.sid
from course_taken ct1
group by ct1.sid
having avg(ct1.grade) = (
        select max(GPA) from (
                select avg(ct2.grade) as GPA
                from course_taken ct2
                group by ct2.sid
        ) highest_GPA
);
```

6. Find the top 3 students with the highest GPAs.

   --Note that if all the grades of a student is null, the average (GPA) will be null. Ordering by GPA, those with null GPA will appear first. Therefore, a condition "avg(grade) is not null" needs to be specified in order to eliminate those tuples with null GPA in the result set.

```
select sid, avg(grade) as GPA
from course_taken
group by sid
having avg(grade) is not null
order by GPA desc
limit 3;
```

7. List the letter grade and the corresponding count in the Course_Taken table. Use the following rules to map a number grade to a letter grade:
   a. A grade > 3.5 is counted towards ‘A’;
   b. A grade > 2.5 is counted towards ‘B’;
   c. All other grades including NULL are counted towards ‘C’.

```
select case
     when grade > 3.5 then 'A'
     when grade > 2.5 then 'B'
     else 'C'
     end as letter_grade, count(*)
from course_taken
group by letter_grade;
```