CS1555 Recitation 7 Solution

Objective:

1. To practice more SQL queries on PostgreSQL.
2. To practice Views

PART 1:

Before we start:

- Download the SQL script studentdb.sql through an sFTP client (such as FileZilla) from the machine "class3.cs.pitt.edu" at the directory:
  - `/afs/pitt.edu/home/r/a/raa88/public/studentdb.sql`

---

1. Assuming there is another table for outreach students who want to major in certificates:

```
create table student_outreach (
        sid integer not null,
        name     varchar(15) not null,
        class    integer,
        major    varchar (10),
        ssn      varchar (16) not null,
        constraint pk_stud_bad primary key(sid)
);
```

Insert the following student in the outreach table:

```
insert into student_outreach values ('130', 'Zach', 1,'CS',
'abcd');
```

List all the students in your organization?

```
(select *
from student)
union (
select *
from student_outreach);
```

2. For each course a student from 'CS' major has repeated, list the studentID and course number.

```
select s.sid, ct.course_no, count(*)
from course_taken ct join student s on ct.sid = s.sid
where major = 'CS'
group by s.sid, ct.course_no
having count(*) >1 ;
```

---

3. List the SIDs and names of the students who have not taken the course "Web Applications".

```
--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 = 'Web Applications');
```

```
--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 = 'Web Applications'
                      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 = 'Web Applications') wa_taking
                      on s.sid = wa_taking.sid
where course_no is null;
```

---

4. 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, we specify a condition "avg(grade) is
not null" in order to eliminate those tuples with null GPA to
appear in the result set.
```

```
select sid, avg(grade) as GPA
from course_taken
group by sid
having avg(grade) is not null
order by avg(grade) DESC
fetch first 3 rows only;
```

---

5. Find the SID and GPA of the top 1 student whose GPA is greater than the student whose SID is 123.

```
select ct1.sid, avg(ct1.grade) as GPA
from course_taken ct1
group by ct1.sid
having avg(ct1.grade) > (
        select avg(ct2.grade) from course_taken ct2 where ct2.sid = '123')
order by avg(ct1.grade) DESC
fetch first 1 rows only;
```

---

6. Rank the students (student ID and name) based on their GPA. Can we do something simpler?

```
select sid,        name,
       (1 + (select count(*)
             from (select s.sid, s.name, avg(grade) as gpa
                   from course_taken ct
                        join student s on ct.sid = s.sid
                   where grade is not null
                   group by s.sid, s.name
                   having avg(grade) > i.gpa
                   order by gpa) e)
          ) as rank
from (select s.sid, s.name, avg(grade) as gpa
      from course_taken ct
           join student s on ct.sid = s.sid
      where grade is not null
      group by s.sid, s.name
      order by gpa) i
order by rank;
```

```
-- Simplify
create or replace view student_gpa as
select s.sid, s.name, avg(grade) as gpa
from course_taken ct
     join student s on ct.sid = s.sid
where grade is not null
group by s.sid, s.name
order by gpa;

-- Now the query
select i.sid, i.name,
       (1 + (select count(*)
             from student_gpa e
             where e.gpa > i.gpa)
          ) as rank
from student_gpa i
order by rank;
```

Is there another way to do that?

```
-- Using Rank()

Select sid, name, RANK() OVER (
                  Order by gpa desc
                  ) AS rank
From student_gpa
```

PART 2:

---

1. Create a view called student_courses that lists the SIDs, student names, number of courses in the Course_taken table.

```
create or replace view student_courses as
select s.sid, s.name, count(course_no) as num_courses
from student s, course_taken ct
where s.sid = ct.sid
group by s.sid, s.name;
```

2. Create a materialized view called mv_student_courses that lists the SIDs, student names, number of courses in the Course_taken table.

```
drop materialized view if exists mv_student_courses;
create materialized view mv_student_courses as
select s.sid, s.name, count(course_no) as num_courses
from student s, course_taken ct
where s.sid = ct.sid
group by s.sid, s.name;
```

3. Execute the following commands. Compare the query results and time used of the two select statements.

```
insert into course_taken (course_no, sid, term, grade)
values ('CS1555', '129','Fall 19', null);

--REFRESH MATERIALIZED VIEW mv_student_courses;
select * from mv_student_courses;
select * from student_courses;
```

- The result from the materialized view is incorrect because the materialized view was not refreshed after the insert statement.
- The result from the view is correct because what a normal view does is rewriting the query. It does not store a snapshot of the query result like the materialized view.
- The running time of the materialized view is shorter, because it does not need to rewrite the query and run the rewritten query on the original course_taken table.