CSE 4310 DATABASE MANAGEMENT SYSTEMS II LAB

LAB 01[Introduction]

Submitted By

Namisa Najah Raisa

ID : 210042112

BSc in Software Engineering
Dept of Computer Science and Engineering

January 17, 2024

January 17, 2024

Contents

1 Queries **3**

1 Queries

1. Find the names of all the instructors from the 'Biology' department.

```sql
select name
from instructor
where dept_name='Biology';
```

2. Show the Course ID and the Title of all the courses registered for by the student with ID '73492'.

```sql
select co.course_id,co.title
from course co,section sec,takes ta
where co.course_id=sec.course_id
and sec.course_id=ta.course_id
and ta.ID='73492';
```

3. Find the names and department names of all the students who have taken a course offered by the 'Comp. Sci.' department.

```sql
select st.name,st.dept_name
from student st,takes ta,course co
where st.ID=ta.ID
and ta.course_id=co.course_id
and co.dept_name='Comp.␣Sci.';
```

4. Find the names of the students who take 'CS-101' course in 'Spring, 2018'.

```sql
select st.name
from student st,takes ta
where st.ID=ta.ID
and ta.course_id='CS-101'
and ta.semester='Spring'
and ta.year='2018';
```

5. Find the names of students who have taken the highest number of courses with a specific prefix 'CS'.

```sql
select st.name, course_count
from student st
join(
select ta.ID, count(ta.course_id) as course_count
from takes ta
join course co on ta.course_id = co.course_id
where co.course_id like 'CS-%'
group by ta.ID
) counts on st.ID=counts.ID
where course_count=(
select max(course_count)
from
(select count(ta.course_id) as course_count
from takes ta
join course co on ta.course_id = co.course_id
where co.course_id like 'CS-%'
group by ta.ID
) count_subquery
);
```

6. Find the names of students who have taken courses taught by at least three different instructors.

```sql
select st.name
from student st, takes ta, section sec, teaches teach
where st.ID = ta.ID
and ta.course_id = sec.course_id
and ta.sec_id = sec.sec_id
and sec.course_id = teach.course_id
and sec.sec_id = teach.sec_id
and sec.semester = teach.semester
and sec.year = teach.year
group by st.name
having count(distinct teach.ID)>=3;
```

7. Find the course name and section having the minimum number of enrollments. Do not include the sections that do not have any students enrolled.

```sql
--Major problems with the approach
select co.title as course_name,sec.course_id,sec.sec_id
    as section_id,count(t.ID) as enrollments_count
from course co
join section sec on co.course_id = sec.course_id
left join takes t on sec.course_id = t.course_id
and sec.sec_id = t.sec_id
and sec.semester = t.semester
and sec.year = t.year
where t.ID is not null
group by co.title,sec.course_id,sec.sec_id
having count(t.ID) =
(select min(enrollments_count)
from
(select count(takes.ID) as enrollments_count
from section
left join takes on section.course_id = takes.course_id
and section.sec_id = takes.sec_id
and section.semester = takes.semester
and section.year = takes.year
where takes.ID is not null
group by section.course_id,section.sec_id) as
    enrollment_counts);
```

8. Find the name of the instructor, dept_name, and count of students he/she advising. If an instructor is not advising any student, show 0.

```sql
select ins.name, ins.dept_name, count(st.ID) as
    num_students
from instructor ins
left join teaches teach on ins.ID = teach.ID
left join section sec on teach.course_id =
    sec.course_id and teach.sec_id = sec.sec_id
left join takes tk on sec.course_id = tk.course_id and
    sec.sec_id = tk.sec_id and sec.semester =
    tk.semester and sec.year = tk.year
left join student st on ins.ID = st.ID
group by ins.name, ins.dept_name;
```

9. Find the name and department of the students who take more courses than the average number of courses taken by a student.

```sql
select name,dept_name
from student
where ID in
(select ID
from takes
group by ID
having count(*)>
(select avg(course_count)from
(select ID,count(*) as course_count
from takes
group by ID
) as course_counts)
);
```

10. Insert each instructor as a student with total credit set to 0 in the same department they are teaching.

```sql
insert into student
(select ins.ID,ins.name,ins.dept_name,0
from instructor ins
where ins.id!='76543');
```

11. Remove all the newly added students from the previous query.

```sql
delete from student
where ID in
(select ID from instructor where id!='76543');
```

12. Update the 'tot_cred' for each student based on the credits taken.

```sql
update student st
set tot_cred = (
select sum(co.credits)
from takes ta
join section sec on ta.course_id = sec.course_id and
    ta.sec_id = sec.sec_id
join course co on sec.course_id = co.course_id
where ta.ID = st.ID
);
```

13. Update the salary of each instructor to 10000 times the number of course sections they have taught.

```sql
update instructor ins
set salary = ins.salary*10000*(
select count(distinct sec.sec_id)
from teaches teach
join section sec
on teach.course_id = sec.course_id
and teach.sec_id = sec.sec_id
where teach.ID = ins.ID
);
```

14. Grades are mapped to a grade point as follows: A:10, B:8, C:6, D:4, and F:0. Create a table to store these mappings,and write a query to find the Credit Point Information (CPI) of each student, using this table. Make sure students who have not got a non-null grade in any course are displayed with a CPI of null.

```sql
--creating table
create table Grade_Points
(grade char(1),
points numeric(3,1));

--inserting records
insert into Grade_Points values ('A', 10);
insert into Grade_Points values ('B', 8);
insert into Grade_Points values ('C', 6);
insert into Grade_Points values ('D', 4);
insert into Grade_Points values ('F', 0);



--query
select st.name,sum(co.credits *
    gp.points)/nullif(sum(co.credits), 0) as cpi
from takes ta
join section sec on ta.course_id = sec.course_id and
    ta.sec_id = sec.sec_id
```

```sql
join course co on sec.course_id = co.course_id
join Grade_Points gp on ta.grade = gp.grade
join student st on ta.ID = st.ID
group by st.name;
```

___The End___