HINDUSTANCOLLEGEOFSCIENCE&TECHNOLOGY
FARAHMATHURA

(AffiliatedtoAKTU,LUCKNOW)

Session
2021-2022

Database Management System
Practical file

Submitted by

Himanshu Gupta

(1900640100020)

Department of Computer Science & Engineering

HINDUSTANCOLLEGEOFSCIENCE&TECHNOLOGY
FARAHMATHURA
(AffiliatedtoAKTU,LUCKNOW)

Basic SQL

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

   select *from tab

   select *from INSTRUCTOR

NAME
cliick

2. Find the names of courses in Computer science department which have 3 credits

   select name from INSTRUCTOR where DEPT_NAME='Biology'

TITLE
Robotics
Image Processing
Database System Concep

3. For the student with ID 12345 (or any other value), show all course_id and title of all courses registered for by the student.

select k. course_id,k.title from takes t,course k

where t.course_id=k.course_id and id=123

COURSE_ID	TITLE
CS-101	Intro. to Computer Science
CS-190	Game Design
CS-315	Robotics
CS-347	Database System Concepts

4. As above, but show the total number of credits for such courses (taken by that student). Don't display the tot_creds value from the student table, you should use SQL aggregation on courses taken by the student.

select COURSE_ID, SUM(CREDITS)

from course

group by COURSE_ID;

COURSE_ID	SUM(CREDITS)
BIO-101	4
BIO-301	4
BIO-399	3
CS-101	4
CS-190	4
CS-315	3
CS-319	3
CS-347	3
EE-181	3
FIN-201	3
More than 10 rows available. Increase rows selector t	

5. As above, but display the total credits for each of the students, along with the ID of the student; don't bother about the name of the student. (Don't bother about students who have not registered for any course, they can be omitted)

```
select ID, sum(credits)
   -> from takes natural join course
   -> group by ID;
```

ID	sum(credits)
98988	8
00128	7
12345	14
45678	11
54321	8
76543	7
98765	7
76653	3
23121	3
19991	3
55739	3
44553	4

6. Find the names of all students who have taken any Comp. Sci. course ever (there should be no duplicate names)

select NAME from student where DEPT_NAME='Comp. Sci.'

..... NAME
Zhang
Shankar
Williams
Brown

7. Display the IDs of all instructors who have never taught a couse (Notesad1) Oracle uses the keyword minus in place of except; (2) interpret "taught" as "taught or is scheduled to teach")

select ID from INSTRUCTOR

minus

select id from TEACHES

ID
33456
58583
76543

8. As above, but display the names of the instructors also, not just the IDs.

select NAME,DEPT_NAME from INSTRUCTOR

NAME	DEPT_NAME
Srinivasan	Comp. Sci.
Wu	Finance
Mozart	Music
Einstein	Physics
El Said	History
Gold	Physics
Katz	Comp. Sci.
Califieri	History
Singh	Finance
Crick	Biology
Brandt	Comp. Sci.
Kim	Elec. Eng.

9. You need to create a movie database. Create three tables, one for actors(AID, name), one for movies(MID, title) and one for actor_role(MID, AID, rolename). Use appropriate data types for each of the attributes, and add appropriate primary/foreign key constraints.

```
create table actors (
10.     AID varchar(20),
```

```
    name varchar(50),
    primary key (AID)
  );
  create table movies (
    MID varchar(20),
    title varchar(50),
    primary key (MID)
  );
  create table actor_role (
    MID varchar(20),
    AID varchar(20),
    rolename varchar(30),
    primary key (MID,AID,rolename),
    foreign key (MID) references movies(MID),
    foreign key (AID) references actors(AID)
);
```

10. Insert data to the above tables (approx 3 to 6 rows in each table), including data for actor "Charlie Chaplin", and for yourself (using your roll number as ID).

```
delete from actor_role;
delete from actors;
delete from movies;
insert into actors values ('01','Charlie Chaplin');
insert into movies values ('M1','City Lights');
insert into actor_role values ('M1','01','Tramp');
insert into actors values ('02','Stephen Chow');
insert into movies values ('M2','Kung Fu Hustle');
insert into actor_role values ('M2','02','Kung Fu Hustle');
insert into actors values ('03','Jay Chou');
insert into movies values ('M3','Initial D');
insert into actor_role values ('M3','03','Fujiwara Takumi');
insert into actors values ('04','Kan Wu');
```

11.Write a query to list all movies in which actor "Charlie Chaplin"

has acted, along with the number of roles he had in that movie.

```
select name, title, count(rolename)
    -> from actor_role natural join movies natural join actors
    -> where name = 'Charlie Chaplin' group by MID;
+-----------------+-------------+-----------------+
| name            | title       | count(rolename) |
+-----------------+-------------+-----------------+
| Charlie Chaplin | City Lights |               1 |
+-----------------+-------------+-----------------+
```

12.Write a query to list all actors who have not acted in any movie

```
select name from actors
    -> where name not in (
    ->   select distinct name
    ->   from actor_role natural join actors);
+--------+
| name   |
+--------+
| Kan Wu |
+--------+
```

13. List names of actors, along with titles of movies they have acted in. If they have not acted in any movie, show the movie title as null. (Do not use SQL outerjoin syntax here, write it from scratch.)

```
14.
15.    select * from (
16.        -> (select name, title
17.        -> from actors, movies, actor_role
18.        -> where actors.AID = actor_role.AID and movies.MID = actor_role.MID)
19.        -> union
```

```
20.        -> (select name as name, null as title from
21.        -> (select name from actors
22.        -> where name not in
23.        -> (select distinct name
24.        -> from actor_role natural join actors)
25.        -> ) as alias1)
26.        -> ) as alias2;
27.    +-----------------+----------------+
28.    | name            | title          |
29.    +-----------------+----------------+
30.    | Charlie Chaplin | City Lights    |
31.    | Stephen Chow    | Kung Fu Hustle |
32.    | Jay Chou        | Initial D      |
33.    | Kan Wu          | NULL           |
34.+-----------------+----------------+
```

Intermediate SQL

Using the university schema that you have write the following queries. In some cases you need to insert extra data to show the effect of a particular feature -- this is indicated with the question. You should then show not only the query, but also the insert statements to add the required extra data.

1. Find the maximum and minimum enrollment across all sections, considering only sections that had some enrollment, don't worry about those that had no students taking that section

```
2. select max(enrollment), min(enrollment)
3.     -> from (select sec_id, semester, year, count(distinct id) as enrollment
4.     -> from takes
```

```
5.     -> group by sec_id, semester, year) as alias1;
6. +-----------------+-----------------+
7. | max(enrollment) | min(enrollment) |
8. +-----------------+-----------------+
9. |               7 |               1 |
10.+-----------------+-----------------+
```

2.Find all sections that had the maximum enrollment (along with the enrollment), using a subquery.

```
with T(sec_id, semester, year, enrollment) as (select sec_id, semester, year,
count(distinct id)from takes group by sec_id, semester, year)
    -> select T.sec_id, T.semester, T.year, T.enrollment from T, (select
max(enrollment) as ma, min(enrollment) from T) as tmp
    -> where T.enrollment = tmp.ma;
+--------+----------+------+------------+
| sec_id | semester | year | enrollment |
+--------+----------+------+------------+
| 1      | Fall     | 2009 |          7 |
+--------+----------+------+------------+
```

3. As in in Q1, but now also include sections with no students taking them; the enrollment for such sections should be treated as 0. Do this in two different ways (and create require data for testing)

```
delete from course where course_id = 'CS-001';
delete from section where sec_id = '1' and semester = 'Fall' and year = '2010';
insert into course(course_id) values ('CS-001');
insert into section(course_id, sec_id, semester, year) values ('CS-
001','1','Fall','2010');
```

a. Using a scalar subquery

```
select distinct sec_id, semester, year, (
    ->   select count(distinct id) from takes
    ->   where (
    ->     takes.sec_id, takes.semester, takes.year
    ->   ) = (
```

```
    ->     section.sec_id, section.semester, section.year))
    -> as cnt from section;
    +--------+----------+------+------+
    | sec_id | semester | year | cnt  |
    +--------+----------+------+------+
    | 1      | Fall     | 2010 |    0 |
    | 1      | Fall     | 2009 |    7 |
    | 1      | Spring   | 2010 |    6 |
    | 1      | Summer   | 2009 |    1 |
    | 1      | Summer   | 2010 |    1 |
    | 1      | Spring   | 2009 |    1 |
    | 2      | Spring   | 2009 |    2 |
    | 2      | Spring   | 2010 |    1 |
    +--------+----------+------+------+
```

b. Using aggregation on a left outer join (use the SQL natural left outer join syntax)

```
select distinct sec_id, semester, year, ifnull(cnt,0)
    -> from section left outer join (
    ->   select sec_id, semester, year, count(distinct id) as cnt
    ->   from takes group by sec_id, semester, year) as T
    -> using (sec_id, semester, year);
+--------+----------+------+---------------+
| sec_id | semester | year | ifnull(cnt,0) |
+--------+----------+------+---------------+
| 1      | Fall     | 2010 |             0 |
| 1      | Fall     | 2009 |             7 |
| 1      | Spring   | 2010 |             6 |
| 1      | Summer   | 2009 |             1 |
| 1      | Summer   | 2010 |             1 |
| 1      | Spring   | 2009 |             1 |
| 2      | Spring   | 2009 |             2 |
```

```
| 2      | Spring   | 2010 |             1 |
+--------+----------+------+---------------+
```

5. Find all courses whose identifier starts with the string "CS-1"

```
select course_id, title
    -> from section natural join course
   -> where course_id like 'CS-1%';
    +-----------+----------------------------+
    | course_id | title                      |
    +-----------+----------------------------+
    | CS-101    | Intro. to Computer Science |
    | CS-101    | Intro. to Computer Science |
    | CS-190    | Game Design                |
    | CS-190    | Game Design                |
    +-----------+----------------------------+
```

6. Find instructors who have taught all the above courses

   a. Using the "not exists ... except ..." structure

```
        Select distinct ID, name from (
    ->   select * from teaches natural join instructor)
    -> as T where not exists (
    ->   select cs_course.course_id from (
    ->     select course_id from course
    ->     where course_id like 'CS-1%')
    ->   as cs_course where cs_course.course_id not in (
    ->     select course_id from (
    ->       select * from teaches natural join instructor)
 ->     as S where S.name = T.name));
```

   b.Using matching of counts which we covered in class (don't forget the distinct clause!).

```
select distinct course_id
  ->   from teaches natural join instructor
  ->   where course_id like 'CS-1%')
  -> select distinct ID, name from (
  ->   select * from teaches natural join instructor) as T
  -> where ((select count(course_id) from S)=(
  ->   select count(distinct course_id)
  ->   from teaches natural join instructor
  ->   where name = T.name and course_id like 'CS-1%'
  -> ));
```

7. Insert each instructor as a student, with tot_creds = 0, in the same department

```
insert into student
  -> select ID, name, dept_name, '0'
  -> from instructor;
```

8. Now delete all the newly added "students" above (note: already existing students who happened to have tot_creds = 0 should not get deleted)

```
delete from student
  -> where (ID, name, dept_name) in (
  ->   select ID, name, dept_name
  ->   from instructor);
```

9. Some of you may have noticed that the tot_creds value for students did not match the credits from courses they have taken. Write and execute query to update tot_creds based on the credits passed, to bring the

database back to consistency. (This query is provided in the book/slides.)

```
update student as S set tot_cred = (
    ->   select sum(credits)
    ->   from takes natural join course
    ->   where S.ID = takes.ID and takes.grade is not null);
```

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

```
update instructor as I set salary = 10000 * (
    ->   select count(distinct sec_id, semester, year)
    ->   from teaches as T where I.ID = T.ID);
```

11. Create your own query: define what you want to do in English, then write the query in SQL. Make it as difficult as you wish, the harder the better.

```
select count(S.name) from student as S,(
    ->   select * from takes natural join course) as C
    -> where S.ID = C.ID and C.dept_name = 'Comp. Sci.';
    +---------------+
    | count(S.name) |
    +---------------+
    |            15 |
    +---------------+
```