Database – First Hand in

SQL

1.

a. Find the names of all students who have taken at least one Comp. Sci. course; make sure there are no duplicate names in the result.

```
select distinct NAME from STUDENT s, TAKES t, COURSE c where s.ID = t.ID and t.COURSE_ID = c.COURSE_ID and c.DEPT_NAME = 'Comp. Sci.';
```

Name
Zhang
Brown
Bourikas
Shankar
Levy
Williams

b. Find the IDs and names of all students who have not taken any course offering before Spring 2009.

```
select distinct s.ID, s.NAME from STUDENT s, TAKES t WHERE s.ID = t.ID and (t.YEAR <> '2009' or t.SEMESTER = 'Summer' or t.SEMESTER = 'Fall');
```

ID	NAME
98988	Tanaka
54321	Williams
19991	Brandt
23121	Chavez
44553	Peltier
98765	Bourikas
76543	Brown
00128	Zhang
12345	Shankar
45678	Levy
55739	Sanchez

c. For each department, find the maximum salary of instructors in that department. You may assume that every department has at least one instructor.

```
Select DEPT_NAME, max(SALARY) as SALARY from INSTRUCTOR group by DEPT_NAME;
```

DEPT_NAME	SALARY
Elec. Eng.	80000
Physics	95000
Comp. Sci.	92000
Finance	90000
Biology	72000
Music	40000
History	62000

d. Find the lowest, across all departments, of the per-department maximum salary computed by the preceding query.

```
select min(SALARY) as SALARY from (select DEPT_NAME, max(SALARY) as SALARY from INSTRUCTOR
group by DEPT_NAME);
```

SALARY
40000

2.

a. Create a new course "CS-001", titled "Weekly Seminar", with 0 credits.

There is a constraint on course that makes it impossible to insert a course with 0 credits. I have therefore sat it to 4.

```
insert into course(COURSE_ID, TITLE, CREDITS) values ('CS-001', 'WEEKLY SEMINAR', '4');
```

b. Create a section of this course in Autumn 2009, with *section id* of 1.

```
insert into section(COURSE_ID, SEC_ID, SEMESTER, YEAR) values ('CS-001', '1', 'Fall', '2009');
```

c. Enroll every student in the Comp. Sci. department in the above section.

```
insert into takes(ID, COURSE_ID, SEC_ID, SEMESTER, YEAR)
select distinct s.ID, 'CS-001', '1', 'Fall', '2009'
from STUDENT s
where s.DEPT_NAME = 'Comp. Sci.';
```

d. Delete enrollments in the above section where the student's name is Chavez.

```
delete from takes t where t.SEC_ID = '1' and t.ID in
(select id from student s where s.NAME = 'Chavez');
```

e. Delete the course CS-001. What will happen if you run this delete statement without first deleting offerings (sections) of this course?

```
delete from course where course_id = 'CS-001';
```

If the course is deleted before the section, the section will automatically be deleted.

f. Delete all *takes* tuples corresponding to any section of any course with the word "database" as a part of the title; ignore case when matching the word with the title.

```
delete from TAKES t where t.SEC_ID in
(select s.SEC_ID from SECTION s
join COURSE on s.COURSE_ID = COURSE.COURSE_ID where lower (COURSE.TITLE like '%database') )
```

3.

a. Write your query using an outer join and then

```
select * from EMPLOYEE e
left outer join MANAGES m on e.EMPLOYEE = m.EMPLOYEE_NAME where m.EMPLOYEE_NAME is
null or m.EMPLOYEE_NAME = ' ';
```

b. Write it again using no outer join at all.

```
select * from EMPLOYEE e where e.EMPLOYEE_NAME not in
(select m.EMPLOYEE_NAME from MANAGES m)
```

4.

Show how to define a view tot_credits (year, num credits), giving the total number of credits taken by students in each year.

```
CREATE OR REPLACE VIEW tot_credits(YEAR, NUMCREDITS) AS
SELECT T.YEAR, SUM(C.CREDITS) FROM STUDENT S
  JOIN TAKES T ON T.ID = S.ID
  JOIN SECTION S ON T.COURSE_ID = S.COURSE_ID AND T.SEC_ID = S.SEC_ID AND T.SEMESTER =
S.SEMESTER AND T.YEAR = S.YEAR
  JOIN COURSE C ON S.COURSE_ID = C.COURSE_ID
  WHERE t.Grade is not null or t.grade != 'F'
  GROUP BY T.YEAR;
```

JDBC

5.

The code finds all mgr with the ename of “dog”, and prints out the mname.

PL/SQL and triggers

6.

Write a PL/SQL Procedure `getInstructorInfo(id)` that fetches all information about an instructor (in the University DB)

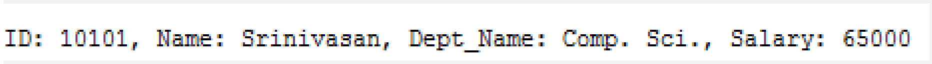
```
create or replace PROCEDURE getInstructorInfo(inst_id VARCHAR2) IS

BEGIN

FOR AROW IN (SELECT ID, NAME, DEPT_NAME, SALARY FROM INSTRUCTOR WHERE ID = inst_ID)
LOOP
SYS.DBMS_OUTPUT.PUT_LINE('Emp ID: ' || arow.id || ', Emp name: ' || arow.name || ', Emp
Dept_Name: ' || arow.dept_name ||', Emp Salary: ' || arow.salary);
END LOOP;
END getInstructorInfo;
```

This can be called with the command:

Call getinstructorinfo(10101) which will output:



```
ID: 10101, Name: Srinivasan, Dept_Name: Comp. Sci., Salary: 65000
```

7.

Write a Trigger `limitSalary` that handles situations when instructors are added or updated.
It should check that the salary for the affected instructor(s) is/are below an upper limit of 100000.
If this is not the case it should adjust the salary for these instructors so that it equals the highest paid instructor.

```
create or replace TRIGGER limitSalary
BEFORE INSERT OR UPDATE ON INSTRUCTOR FOR EACH ROW
BEGIN
  IF (:new.SALARY > 100000) THEN
   SELECT MAX(i.SALARY) into :new.SALARY from INSTRUCTOR i;
  END IF;
END;
```