Lab1

```
insert into employee values(000,'Aarav','Mumbai');
insert into employee values(001,'Dhruv','Delhi');
insert into employee values(002,'Gautam','Bangalore');
insert into employee values(003,'Bob','Mangalore');
insert into employee values(004,'Sam','Manipal');

select emp_name from employee;
select * from employee where emp_address = 'Manipal';

alter table employee add(Salary Number(6,2);
//ALTER TABLE employee
//MODIFY salary NUMBER(6,2);


update employee set salary = 5000.0 where emp_no = 000;
update employee set salary = 6000.0 where emp_no= 001;
update employee set salary = 7000.0 where emp_no = 002;
update employee set salary = 8000.0 where emp_no = 003;
update employee set salary = 9000.0 where emp_no = 004;


desc employee;

DELETE FROM EMPLOYEE WHERE emp_address ='Mangalore';

rename employee to employee1;
Drop table employee1;
```

LAB 2

1

```
CREATE TABLE Employee (
   empNo NUMBER(3) PRIMARY KEY,
   empName VARCHAR(20) NOT NULL,
   gender VARCHAR(6) CHECK (gender IN ('M', 'F')) NOT NULL,   salary NUMBER(7,2) NOT
NULL,
   address VARCHAR(30) NOT NULL,
   dNo NUMBER(3)
);
```

2

```
CREATE TABLE Department (
   deptNo NUMBER(3) PRIMARY KEY,
   deptName VARCHAR(20) unique,
   location VARCHAR(30)
);

ALTER TABLE Employee
ADD FOREIGN KEY (dNo) REFERENCES Department(deptNo);

//ALTER TABLE Employee
ADD CONSTRAINT fk_dept FOREIGN KEY (dNo) REFERENCES Department(deptNo);



insert into department values(1,'Management','Mumbai');
insert into department values(2,'Finance', 'Delhi');
insert into department values(3,'Tech', 'Goa');

insert into employee values(000,'Aarav','M',50000.00,'Mumbai',1);
insert into employee values(001,'Dhruv','M',60000.00,'Delhi',2);
insert into employee values(002,'Gautam','F',70000.00,'Goa',3);


insert into employee values(003,'Bob','Male',50000.00,'Mumbai',1);
insert into department values(3,'HR','Chennai');


DELETE FROM Department WHERE deptNo = 1;

SELECT constraint_name
FROM user_constraints
WHERE table_name = 'EMPLOYEE'
AND constraint_type = 'R';

ALTER TABLE Employee DROP CONSTRAINT SYS_C00143846;

//ALTER TABLE Employee DROP CONSTRAINT fk_dept;



ALTER TABLE Employee ADD CONSTRAINT fk_dept FOREIGN KEY (dNo) REFERENCES
Department(deptNo) ON DELETE CASCADE;
```

```
ALTER TABLE Employee MODIFY salary NUMBER(7, 2) DEFAULT 10000 CONSTRAINT
salary_default CHECK (salary >= 0);


INSERT INTO Department (DeptNo, DeptName, Location) VALUES (4, 'HR', 'Sikkim');
INSERT INTO Employee (empNo, empName, gender, address, dNo) VALUES (004, 'Disha', 'M',
'Sikkim', 4);
SELECT * FROM Employee WHERE empNo = 4;
```

University Database Schema:

Student (ID, name,dept-name, tot-cred)
Instructor(ID, name, dept-name, salary)
Course (Course-id, title,dept-name, credits)
Takes (ID, course-id, sec-id, semester, year, grade)
Classroom (building, room-number, capacity)
Department (dept-name, building, budget)
Section (course-id, section-id, semester, year, building, room-number, time-slot-id)
Teaches (id, course-id, section-id, semester, year)
Advisor(s-id, i-id)
Time-slot (time-slot-id, day, start-time, end-time)
Prereq (course-id, prereq-id)

```
create table classroom
        (building               varchar(15),
         room_numbervarchar(7),
         capacity               numeric(4,0),
         primary key (building, room_number)
        );

create table department
        (dept_name              varchar(20),
         building               varchar(15),
         budget                 numeric(12,2) check (budget > 0),
         primary key (dept_name)
        );

create table course
        (course_id              varchar(8),
         title          varchar(50),
         dept_name              varchar(20),
         credits        numeric(2,0) check (credits > 0),
         primary key (course_id),
```

```
        foreign key (dept_name) references department
                on delete set null
        );

create table instructor
        (ID                     varchar(5),
        name                    varchar(20) not null,
        dept_name               varchar(20),
        salary numeric(8,2) check (salary > 29000),
        primary key (ID),
        foreign key (dept_name) references department
                on delete set null
        );

create table section
        (course_id              varchar(8),
       sec_id           varchar(8),
        semester                varchar(6)
        check (semester in ('Fall', 'Winter', 'Spring', 'Summer')),
        year    numeric(4,0) check (year > 1701 and year < 2100),
        building                varchar(15),
        room_numbervarchar(7),
        time_slot_id    varchar(4),
        primary key (course_id, sec_id, semester, year),
        foreign key (course_id) references course
                on delete cascade,
        foreign key (building, room_number) references classroom
                on delete set null
        );

create table teaches
        (ID                     varchar(5),
        course_id               varchar(8),
        sec_id          varchar(8),
        semester                varchar(6),
        year                    numeric(4,0),
        primary key (ID, course_id, sec_id, semester, year),
        foreign key (course_id,sec_id, semester, year) references section on delete cascade,
        foreign key (ID) references instructor on delete cascade
        );

create table student
        (ID                     varchar(5),
        name                    varchar(20) not null,
```

```
        dept_name           varchar(20),
        tot_cred            numeric(3,0) check (tot_cred >= 0),
        primary key (ID),
        foreign key (dept_name) references department
                on delete set null
        );

create table takes
        (ID                 varchar(5),
        course_id           varchar(8),
        sec_id      varchar(8),
        semester            varchar(6),
        year                numeric(4,0),
        grade       varchar(2),
        primary key (ID, course_id, sec_id, semester, year),
        foreign key (course_id,sec_id, semester, year) references section on delete cascade,
        foreign key (ID) references student on delete cascade
        );

create table advisor
        (s_ID               varchar(5),
        i_ID                varchar(5),
        primary key (s_ID),
        foreign key (i_ID) references instructor (ID)
                on delete set null,
        foreign key (s_ID) references student (ID)
                on delete cascade
        );

create table time_slot
        (time_slot_id       varchar(4),
        day                 varchar(1),
        start_hr            numeric(2) check (start_hr >= 0 and start_hr < 24),
        start_min           numeric(2) check (start_min >= 0 and start_min < 60),
        end_hr              numeric(2) check (end_hr >= 0 and end_hr < 24),
        end_min             numeric(2) check (end_min >= 0 and end_min < 60),
        primary key (time_slot_id, day, start_hr, start_min)
        );

create table prereq
        (course_id          varchar(8),
        prereq_id           varchar(8),
        primary key (course_id, prereq_id),
        foreign key (course_id) references course
```

```
            on delete cascade,
        foreign key (prereq_id) references course
        );
```

9.

```
9.  select name,dept_name from student;
10. select name from instructor where dept_name = 'Comp. Sci.';
11. select title from course where dept_name = 'Comp. Sci.' and credits = 3;
12. SELECT course_id, title
FROM course
WHERE course_id IN (
   SELECT course_id
   FROM takes
   WHERE ID = '12345'
);
Or
select t.course_id, c.title from course c,takes t where t.id = 12345 and t.course_id = c.course_id;
13. SELECT name
FROM instructor
WHERE salary BETWEEN 40000 AND 90000;

14.
SELECT id
FROM instructor
WHERE id NOT IN (SELECT id FROM teaches);
Or

select i.id from instructor i where i.id not in (select id from teaches);
15. SELECT S.name AS student_name,
      C.title AS course_name,
      T.year
FROM student S, takes T, section Sec, course C
WHERE S.ID = T.ID
  AND T.course_id = Sec.course_id
  AND T.sec_id = Sec.sec_id
  AND T.semester = Sec.semester
  AND T.year = Sec.year
  AND Sec.course_id = C.course_id
  AND Sec.room_number = ‘303’;
```

16.select s.name, t.course_id, c.title c_name from student s, takes t, course c where s.id = t.id
and t.course_id = c.course_id
and t.year = 2015;
;
17.
select i.name inst_salary from instructor i where i.salary > some (select j.salary from instructor j where dept_name = 'Comp. Sci.');

University Database Schema:

Student (ID, name,dept-name, tot-cred)
Instructor(ID, name, dept-name, salary)
Course (Course-id, title,dept-name, credits)
Takes (ID, course-id, sec-id, semester, year, grade)
Classroom (building, room-number, capacity)
Department (dept-name, building, budget)
Section (course-id, section-id, semester, year, building, room-number, time-slot-id)
Teaches (id, course-id, section-id, semester, year)
Advisor(s-id, i-id)
Time-slot (time-slot-id, day, start-time, end-time)
Prereq (course-id, prereq-id)

String Operations (Use %, _, LIKE):
18. Find the names of all instructors whose department name includes the substring ‘ch’.
Built-in Functions:
19. List the student names along with the length of the student names.
20. List the department names and 3 characters from 3 rdposition of each department name
21. List the instructor names in upper case.
22. Replace NULL with value1(say 0) for a column in any of the table
23. Display the salary and salary/3 rounded to nearest hundred from Instructo

18. SELECT name
FROM instructor
WHERE dept_name LIKE '%ch%';

19.SELECT name, LENGTH(name)
FROM student;

20.
SELECT dept_name, SUBSTR(dept_name, 3, 3)
FROM department;

21.
SELECT UPPER(name)
FROM instructor;

22.
```
SELECT ID, name, NVL(tot_cred, 0)
FROM student;
```

23.
```
SELECT salary, ROUND(salary / 3, -2)
FROM instructor;
```

24.
```
CREATE TABLE employee (
   emp_id NUMBER PRIMARY KEY,
   emp_name VARCHAR2(50)
);

ALTER TABLE employee
ADD dob DATE;

insert into employee values(1,'Aarav', to_date('30062005','DDMMYYYY'));
insert into employee values(2,'Dhruv', to_date('30071979','DDMMYYYY'));
insert into employee values(3,'Gautam', to_date('30061947','DDMMYYYY'));
insert into employee values(4,'Disha', to_date('29022020','DDMMYYYY'));



select emp_name, to_char(dob,'DD-MON-YYYY') f1, to_char(dob,'DD-MON-YY') f2,
to_char(dob,'DD-MM-YY') f3
from employee;
```

25.
```
select emp_name, to_char(dob, 'YEAR') f1, initcap(to_char(dob,'YEAR')) f2,
lower(to_char(dob,'YEAR')) f3 from employee;
```

Additional
```
alter table employee add salary number(7,2);
```

1.
```
alter table employee add constraint sal_check check(salary > 5000);
```

2.
```
SELECT TO_CHAR(SYSDATE, 'Q') AS quarter_of_year
FROM DUAL;
```

3.

```
SELECT FLOOR(3600 / 3600) AS hours,
       FLOOR(MOD(3600, 3600) / 60) AS minutes,
       MOD(3600, 60) AS seconds
FROM DUAL;
```

4.

```
SELECT TO_CHAR(SYSDATE, 'IW') AS week_of_year
FROM DUAL;
```

5.

```
SELECT DISTINCT dept_name
FROM instructor;
```

6.

```
SELECT i.name, t.course_id
FROM instructor i, teaches t
WHERE i.ID = t.ID;
```

7.

```
SELECT i.name AS instructor_name, t.course_id
FROM instructor i, teaches t
WHERE i.ID = t.ID;
```

8.

```
SELECT s.name AS student_name,
       s.dept_name,
       i.name AS advisor_name,
       (SELECT COUNT(*)
        FROM takes t
        WHERE t.ID = s.ID) AS num_courses
FROM student s, advisor a, instructor i
WHERE s.ID = a.s_ID
  AND a.i_ID = i.ID;
```

```
SELECT s.name AS student_name,
       s.dept_name,
       i.name AS advisor_name,
       COUNT(t.course_id) AS num_courses
FROM student s, advisor a, instructor i, takes t
WHERE s.ID = a.s_ID
  AND a.i_ID = i.ID
```

```
  AND s.ID = t.ID
GROUP BY s.ID, s.name, s.dept_name, i.name;
```

Additional DBS Lab 2
Q3

```
 select floor(seconds/3600) hours, floor((seconds-(3600*(floor(seconds/3600))))/60) minutes,
seconds-(floor(seconds/3600))*3600-(floor((seconds-(3600*(floor(seconds/3600))))/60))*60
seconds from time;

     HOURS    MINUTES    SECONDS
---------- ---------- ----------
        64         30         23

SQL> select * from time;

   SECONDS
----------
    232223

SQL> desc time;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SECONDS
```

LAB3

UNION (Use union all to retain duplicates):
1.Find courses that ran in Fall 2009 or in Spring 2010

```
select course_id from section where semester ='Fall' and year = 2009
union
select course_id from section where semester = 'Spring' and year = 2010;
```

INTERSECT (Use intersect all to retain duplicates):
2. Find courses that ran in Fall 2009 and in spring 2010

```
select course_id from section where semester ='Fall' and year = 2009
intersect
select course_id from section where semester = 'Spring' and year = 2010;
```

MINUS:
3. Find courses that ran in Fall 2009 but not in Spring 2010

```
select course_id from section where semester ='Fall' and year = 2009
minus
select course_id from section where semester = 'Spring' and year = 2010;
```

Null values
4. Find the name of the course for which none of the students registered.
Nested Subqueries
select title from course where course_id not in (select distinct course_id from takes);

Set Membership (in / not in):
5. Find courses offered in Fall 2009 and in Spring 2010.
select distinct course_id from section where semester = 'Fall' and year = 2009 and course_id in (select distinct
course_id from section where semester = 'Spring' and year = 2010);

6. Find the total number of students who have taken course taught by the instructor with ID 10101.
select count(student.ID) from student, takes
where student.id = takes.id
and takes.course_id in
(select course_id from teaches where teaches.id = 10101);

7. Find courses offered in Fall 2009 but not in Spring 2010.
select distinct course_id from section where semester = 'Fall' and year = 2009
and
course_id not in (select distinct course_id from section where semester = 'Spring' and year = 2010);

8. Find the names of all students whose name is same as the instructor's name.
select distinct name from student where student.name
in
(select distinct instructor.name from instructor);

Set Comparison (>=some/all)
9. Find names of instructors with salary greater than that of some (at least one) instructor in the Biology department.
SELECT name FROM instructor
WHERE salary > SOME (SELECT salary FROM instructor WHERE dept_name = 'Biology');

10. Find the names of all instructors whose salary is greater than the salary of all instructors in the Biology department.
SELECT name FROM instructor
WHERE salary > ALL (SELECT salary FROM instructor WHERE dept_name = 'Biology');

11. Find the departments that have the highest average salary.
select dept_name from instructor
group by dept_name having avg(salary) >=

```
all
(select avg(salary) from instructor group by dept_name);
```

12. Find the names of those departments whose budget is lesser than the average salary of all instructors.

```
select dept_name from department where budget <
all
(select avg(salary) from instructor);
```

Test for Empty Relations (exists/ not exists)

13. Find all courses taught in both the Fall 2009 semester and in the Spring 2010 semester

```
select
select distinct course_id from section s1
where semester = 'Fall' and year = 2009
and exists
(select * from section s2 where s1.course_id = s2.course_id and
s2.semester = 'Spring' and s2.year = 2010);
```

14. Find all students who have taken all courses offered in the Biology department.

```
SELECT DISTINCT S.ID, S.name
FROM student S
WHERE NOT EXISTS (
   (SELECT course_id FROM course WHERE dept_name = 'Biology')
   MINUS
   (SELECT T.course_id FROM takes T WHERE S.ID = T.ID)
);
```

Test for Absence of Duplicate Tuples

15. Find all courses that were offered at most once in 2009

```
select distinct course_id from section where year = 2009
group by course_id having count(*) <=1;
```

or

```
SELECT DISTINCT S1.course_id
FROM section S1
WHERE year = 2009
AND NOT EXISTS (
   SELECT * FROM section S2
   WHERE S1.course_id = S2.course_id
```

```
   AND S1.sec_id <> S2.sec_id
   AND S2.year = 2009
);
```

16. Find all the students who have opted at least two courses offered by CSE department.

```
SELECT DISTINCT s.name, s.id
FROM student s
WHERE s.id IN (
   SELECT t.id
   FROM takes t, course c
   WHERE t.course_id = c.course_id
   AND c.dept_name = 'Comp. Sci.'
   GROUP BY t.id
   HAVING COUNT(DISTINCT t.course_id) >= 2
);
```

or

```
SELECT ID, name
FROM student
WHERE ID IN (
   SELECT ID
   FROM takes
   WHERE course_id IN (SELECT course_id FROM course WHERE dept_name = 'Comp. Sci.')
   GROUP BY ID
   HAVING COUNT(DISTINCT course_id) >= 2
);
```

Subqueries in the From Clause

17. Find the average instructors salary of those departments where the average salary is greater than 42000

```
select dept_name, avg_salary
from
(select dept_name, avg(salary)  avg_salary
from instructor group by dept_name)  dept_avg_salary
where avg_salary>42000;
```

18.

```
CREATE VIEW all_courses AS
SELECT course_id, building, room_number
FROM section
WHERE semester = 'Fall' AND year = 2009
```

```
AND course_id IN (SELECT course_id FROM course WHERE dept_name = 'Physics');
```

19.
```
SELECT * FROM all_courses;
```

20.
```
create view department_total_salary as
select dept_name, sum(salary) total_salary
from instructor group by dept_name;
```

Additional
1.
```
SELECT DISTINCT dept_name FROM instructor;
```

2.
```
SELECT DISTINCT dept_name FROM instructor;
```

3.
```
SELECT DISTINCT I.name, T.course_id
FROM instructor I
LEFT JOIN teaches T ON I.ID = T.ID;
```

4.
```
SELECT DISTINCT I.name, T.course_id
FROM instructor I
LEFT JOIN teaches T ON I.ID = T.ID;
```