LAB 8 :

Q1). Based on the University Database Schema in Lab 2, write a procedure which takes the dept_name as input parameter and lists all the instructors associated with the department as well as list all the courses offered by the department. Also, write an anonymous block with the procedure call.

```
Create or replace procedure dept_count(in dep_name instructor.dept_name
%type) is
2 declare instruct_count integer;
3 begin
4 select count(*) into instruc_count
5 from instructor
6 where instructor.dept_name= dept_count.dep_name;
7 dbms_output.put_line(instruct_count);
8 end;
9 /

Declare
2 begin
3 dept_count('Comp. Sci.');
4 end;
5 /
```

Q2). Based on the University Database Schema in Lab 2, write a Pl/Sql block of code that lists the most popular course (highest number of students take it) for each of the departments. It should make use of a procedure course_popular which finds the most popular course in the given department.

```
create or replace procedure course_popular(in dep_name course.dept_name
%type, out course_name course.title%type) is
2 begin
3 with counts as (select takes.course_id,count(id) as students
4 from takes group by takes.course_id
5 having takes.course_id in (select course_id from course where
dept_name=dep_name)),
6 max_count as (select max(students) as ms from counts)
7 select course_id into course_name from counts,max_count where students =
ms;
8 end;
9 /

declare id course.course_id%type;
2 begin
3 course_popular('Comp. Sci.',id);
4 end;
```

```
5 /
```

Functions:

Q3). Write a function to return the Square of a given number and call it from an anonymous block.

```
create or replace function square_num(a number)
2 return number as
3 sqr number;
4 begin
5 sqr:= a*a;
6 return sqr;
7 end;
8 /

declare
2 begin
3 dbms_output.put_line(square_num(8));
4 end;
5 /
```

Q4). Based on the University Database Schema in Lab 2, write a Pl/Sql block of code that lists the highest paid Instructor in each of the Department. It should make use of a function department_highest which returns the highest paid Instructor for the given branch.

```
create or replace function highest_paid(d_name varchar)
2 return varchar as
3 instruc_name varchar(20);
4 begin
5 select name into instruc_name from instructor
6 natural join (select dept_name, max(salary) as max_sal from instructor
group by dept_name)
7 where dept_name=d_name and salary=max_sal;
8 return instruc_name;
9 end;
10 /


declare
2 begin
3 dbms_output.put_line(highest_paid('Comp. Sci.'));
4 end;
5 /
```

Triggers –

Q1). Based on the University database Schema in Lab 2, write a row trigger that records along with the time any change made in the Takes (ID, course-id, sec-id, semester, year, grade) table in log_change_Takes (Time_Of_Change, ID, courseid,sec-id, semester, year, grade).

```
create table log_change_takes(
2 time_of_change timestamp,
3 id varchar(5),
4 course_id varchar(10),
5 sec_id varchar(10),
6 semester varchar(7),
7 year numeric(4,0),
8 grade varchar(2));
```

```
create or replace trigger log_change_takes
2 before insert or update
3 or delete on takes
4 for each row
5 begin
6 case
7 when inserting then
8 insert into log_change_takes values
(current_timestamp,:new.id,:new.course_id,:new.sec_id,:new.semester,:new.yea
r,:new.grade);
9 when updating then
10 insert into log_change_takes values
(current_timestamp,:new.id,:new.course_id,:new.sec_id,:new.semester,:new.yea
r,:new.grade);
11 when deleting then
12 insert into log_change_takes values
(current_timestamp,:new.id,:new.course_id,:new.sec_id,:new.semester,:new.yea
r,:new.grade);
13 end case;
14 end;
15 /
```

Q2) Based on the University database schema in Lab: 2, write a row trigger to insert the existing values of the Instructor (ID, name, dept-name, salary) table into a new table Old_ Data_Instructor (ID, name, dept-name, salary) when the salary table is updated.

```
create table Old_Data_Instructor(
2 ID varchar(5),
3 name varchar(20),
4 dept_name varchar(25),
```

```
5 salary numeric(6,2));

create or replace trigger instructor_trigger
2 before update on instructor
3 for each row
4 begin
5 insert into Old_Data_Instructor
values(:OLD.ID,:OLD.name,:OLD.dept_name,:OLD.salary);
6 end;
7 /
```