LAB 8

LAB EXERCISE

PROCEDURES

1). 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 ins_and_courses(d_name
department.dept_name%type) is
    cursor c1 is select name from instructor where dept_name =
d_name;
    cursor c2 is select title from course where dept_name =
d_name;
    begin
        dbms_output.put_line('Instructors : ');
        for info in c1
            loop
                dbms_output.put(info.name || ', ');
            end loop;
        dbms_output.put_line('');
        dbms_output.put_line('Courses : ');
        for info in c2
            loop
                dbms_output.put(info.title || ', ');
            end loop;
        dbms_output.put_line('');
    end;
    /

declare
```

```
    d_name department.dept_name%type;
begin
    d_name := '&department_name';
    ins_and_courses(d_name);
end;
/
```

2). 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(d_name
department.dept_name%type) is
    c_name course.title%type;
    cursor c1 is select title from (course natural join takes)
where dept_name = d_name group by course_id,title
    having count() = (select max(freq) from (select count()as
freq from (course natural join takes) where dept_name = d_name
group by course_id,title));
    begin
        dbms_output.put('Most popular courses in  '||d_name||'
: ');
        for info in c1
            loop
                dbms_output.put(info.title || ', ');
            end loop;
        dbms_output.put_line('');
    end;
    /

declare
    cursor c1 is select distinct dept_name from department;
begin
    for info in c1
        loop
```

```
            course_popular(info.dept_name);
        end loop;
end;
/
```

FUNCTIONS

3). 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)
return number as
sqr number;
begin
sqr:= a*a;
return sqr;
end;
/
```

```
declare
begin
dbms_output.put_line(square_num(8));
end;
/
```

4). 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)
return varchar as
instruc_name varchar(20);
begin
select name into instruc_name from instructor
```

```
natural join (select dept_name, max(salary) as max_sal from instructor group by dept_name)
where dept_name=d_name and salary=max_sal;
return instruc_name;
end;
/
```

```
declare
begin
dbms_output.put_line(highest_paid('Comp. Sci.'));
end;
/
```

<u>TRIGGERS</u>

1). 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(
time_of_change timestamp,
id varchar(5),
course_id varchar(10),
sec_id varchar(10),
semester varchar(7),
year numeric(4,0),
grade varchar(2));
```

```
create or replace trigger log_change_takes
before insert or update
or delete on takes
for each row
begin
case
```

```
    when inserting then
          insert into log_change_takes values (current_timestamp,:new.id,:new.course_id,:new.sec_id,:new.semester,:new.year,:new.grade);
    when updating then
            insert into log_change_takes values (current_timestamp,:new.id,:new.course_id,:new.sec_id,:new.semester,:new.year,:new.grade);
   when deleting then
               insert into log_change_takes values (current_timestamp,:new.id,:new.course_id,:new.sec_id,:new.semester,:new.year,:new.grade);
 end case;
 end;
 /
```

2). 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(
ID varchar(5),
name varchar(20),
dept_name varchar(25),
salary numeric(6,2));
```

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