PL/SQL Basics:

Assignment Harshit Kushmakar| 16896

Subprograms

Practice 1

• Create a procedure called USER_QUERY_EMP that accepts three parameters. Parameter

p_myeno is of IN parameter mode which provides the empno value. The other two

parameters p_myjob and p_mysal are of OUT mode. The procedure retrieves the salary

and job of an employee with the provided employee number and assigns those to the

two OUT parameters respectively. The procedure should handle the error if the empno

does not exist in the EMP table by displaying an appropriate message. Use bind variables

for the two OUT Parameters.

• Compile the code, invoke the procedure, and display the salary and job title for

employee number 7839. Do the same for employee number 7123.

```
set serveroutput on

create or replace procedure USER_QUERY_EMP_Pratyush( p_myeno in employee_pratyush.id%type,
                                        p_myjob out employee_pratyush.job%type,
                                        p_mysal out employee_pratyush.salary%type)

as
```

```
begin
   select job, salary into p_myjob, p_mysal from employee_pratyush where
id=p_myeno;
   exception
      when no_data_found
         then DBMS_OUTPUT.PUT_LINE('Invalid Employee Id');
end;


declare
   p_myeno Employee_Pratyush.id%type:=&emp_id;
   p_myjob Employee_Pratyush.job%type;
   p_mysal Employee_Pratyush.salary%type:=-1;
begin
   USER_QUERY_EMP_Pratyush(p_myeno, p_myjob, p_mysal);
   if p_mysal>0
      then DBMS_OUTPUT.PUT_LINE('Salary: '||p_mysal||' Job Title: '||p_myjob);
   end if;
end;
/


select * from employee_pratyush;
--insert into employee_pratyush values(7839,'XYZ', 8000, 'SDE', 24,2);
--ALTER table employee_pratyush modify id number(20);


--ALTER table employee_pratyush add COMM number(20);
```

--update employee_pratyush set comm=100 where id<=8;

Practice 2

• Create a function named USER_ANNUAL_COMP that has a parameter p_eno for passing

on the values of an employee number of the employee. The function calculates and

returns the annual compensation of the employee by using the following formula.

annual_compensation = (p_sal+p_comm)*12

If the salary or commission value is NULL then zero should be substituted for it.

• Give a call to USER_ANNUAL_COMP from a SELECT statement, against the EMP table.

```
create or replace function USER_ANNUAL_COMP_Kushmakar(p_eno
employee_Kushmakar.id%type)

return number

is

   annual_compensation number;

   p_sal employee_Kushmakar.salary%type;

   p_comm employee_Kushmakar.comm%type;

begin

   select salary,comm into p_sal,p_comm from employee_Kushmakar where
id=p_eno;

   if p_sal is null

      then p_sal := 0;
   end if;
   if p_comm is null
      then p_comm :=0;
   end if;
      annual_compensation:=(p_sal+p_comm)*12;
      return annual_compensation;
end;
/

declare
   p_myeno employee_Kushmakar.id%type:=&emp_id;
   annual_comp number;
begin
   select USER_ANNUAL_COMP_Kushmakar(p_myeno) into annual_comp from
employee_Kushmakar where id=p_myeno;
   DBMS_OUTPUT.PUT_LINE(annual_comp);
end;

select * from dept_Kushmakar;
```

Practice 3

• Create a function named USER_VALID_DEPTNO that has a single parameter p_dno to

accept a department number and returns a BOOLEAN value. The function returns TRUE

if the department number exists in the DEPT table else it returns FALSE.

• Create a procedure named SHOW_STRENGTH that accepts department number in a

single parameter p_deptno from user. The procedure gives a call to

USER_VALID_DEPTNO. If the function returns TRUE then the procedure finds out how

many employees are there in the department from the EMP table and displays the same

on the screen. If the function returns FALSE then the procedure displays an appropriate

error message.

• Give call to SHOW_STRENGTH by passing on department number 10. Do the same for

department number 76.

```
--part1

create or replace function user_valid_deptno_Kushmakar (p_dno dept_Kushmakar.depid%type)

return boolean

as

   is_dep boolean:=true;

   cnt number :=0;

begin

   select count(*) into cnt from dept_Kushmakar where depid=p_dno;

   if cnt>0

      then return true;
   else
      return false;
   end if;
end;
/

declare
   p_dno dept_Kushmakar.depid%type:=&d_id;
begin
   if user_valid_deptno_Kushmakar(p_dno)
      then
      dbms_output.put_line('Department Id is valid');
   else
      dbms_output.put_line('Department Id is not valid');
   end if;
end;

--part2
create or replace procedure SHOW_STRENGTH_Kushmakar (p_deptno
dept_Kushmakar.depid%type)
as
   cnt number;
begin
   if user_valid_deptno_Kushmakar(p_deptno)
   then
   select count(*) into cnt from employee_Kushmakar where depid=p_deptno;

   DBMS_OUTPUT.PUT_LINE('Number of employee with Department Id
'||p_deptno||' is '||cnt);

   else

   DBMS_OUTPUT.PUT_LINE('Department Id is not valid');

   end if;

end;


declare

   p_dno dept_Kushmakar.depid%type:=&d_id;

begin

   SHOW_STRENGTH_Kushmakar(p_dno);

end;

/

--part3

declare


begin

   SHOW_STRENGTH_Kushmakar('76');

end;

/


ALTER table employee_Kushmakar add join_date date;

update employee_Kushmakar set comm=100 where id<=8;
```

Practice 4

• Create a procedure named SHOW_RECORDS that accepts a single parameter

p_join_date. The procedure determines and displays on the screen, the details of the

employees who have joined after p_join_date, in the following format.

Employees Joined after ddth, Month yyyy

EMPLOYEE NAME JOB SALARY DEPARTMENT

____

XXXXXXXX XXXXX99,999 99

XXXXXXXX XXXXX99,999 99

2/3

The procedure should display appropriate message if there is no employee who joined after

p_join_date .

• Give a call to SHOW_RECORDS from an anonymous PL/SQL block

```
create or replace

procedure SHOW_RECORDS_Kushmakar(p_join_date date)

as

 cursor c1 is select * from Employee_Kushmakar where join_date>p_join_date;

 e1 Employee_Kushmakar%rowtype;

 cnt number:=0;
```

```
begin
 select count(*) into cnt from Employee_Kushmakar where
join_date>p_join_date;
 if cnt>0
 then
 DBMS_output.put_line('Employees Joined after '||p_join_date);
 open c1;
 loop
 fetch c1 into e1;
 DBMS_output.put_line(e1.name||' '||e1.job||' '||e1.salary||'
'||e1.depid);
 exit when c1%notfound;
 end loop;
 close c1;
 else
 DBMS_output.put_line('NO Employee Joined after '||p_join_date);
 end if;
end;

declare
begin
 SHOW_RECORDS_Kushmakar('07-DEC-22');
end;
```

Practice 5

• Create a procedure named ADD_EMPLOYEE to hire an employee. Parameters to the

procedure are job, mgr, hiredate, salary, commission and deptno. Validate the following:

a. Employee number is not taken as a parameter but is auto generated by using a

SEQUENCE.

b. Job is either 'CLERK' or 'ANALYST' or 'SALESMAN'. The input value can be entered in

any case (upper or lower or initcap).

c. Mgr is an existing employee.

d. Hiredate is less than system date.

e. Salary must be greater than 800

f. Commission is not null if the job is SALESMAN. For any other job, commission may be

null.

g. Deptno must exist in the DEPT table.

Insert the record if the above validations are met and display a message '1 row

inserted'. If the row is not inserted generate an exception and handle it by displaying an

appropriate message.

• Give a call to ADD_EMPLOYEE through an anonymous PL/SQL block.

Practice 6

• Create a function named FIND_SAL_GRADE which accepts salary of an employee finds

the corresponding salary grade from SALGRADE table and returns the grade. The

function should raise an exception if the salary value does not fit in any of the salary

ranges specified in the salgrade table.

• Create a procedure CALL_FIND_SAL_GRADE that does not accept any parameter. The

procedure gives call to FIND_SAL_GRADE for each record in the emp table by passing on

the salary value from the current record. The procedure displays the corresponding

employee number, employee name and the salary grade returned by FIND_SAL_GRADE,

on the screen. The procedure should handle error thrown by the function by displaying

an appropriate message.

• Give a call to CALL_FIND_SAL_GRADE through an anonymous PL/SQL block

```
create or replace function FIND_SAL_GRADE_Kushmakar(e_salary

employee_Kushmakar.salary%TYPE)

return salgrade.grade%type

as

e_grade salgrade.grade%type:='0';

begin
```

```
 select grade into e_grade from SALGRADE where e_salary>min_sal and
e_salary<max_sal;
 exception
 when no_data_found
 then
 DBMS_OUTPUT.PUT_LINE('Salary range not matched');
 return e_grade;
end;

--
create or replace procedure CALL_FIND_SAL_GRADE_Kushmakar
as
e_salgrade varchar2(1);
e Employee_Kushmakar%rowtype;
begin
 select * into e from Employee_Kushmakar where id=1;
 e_salgrade:=FIND_SAL_GRADE_Kushmakar(e.salary);
 DBMS_output.put_line(e.id||' '||e.name||' '||e_salgrade);
end;

--
declare
begin
 CALL_FIND_SAL_GRADE_Kushmakar;
```

```
end;
```