set serveroutput on;

**1.Write a PL/SQL procedure for mini_statement that acceepts an account number and**
**returns that last 5 transactions for that account ordered by Trans_date descending. Display the**
**transaction date, type, amount, and description using DBMS_OUTPUT.**
**TABLE NAME: BANK_TRANSACTIONS**
**COLUMNS: TRANS_ID,ACCOUNT_NO,TRANS_DATE,TRANS_TYPE, AMOUNT,DESCRIPTION**

```
create or replace PROCEDURE shubham_bulk_bank_m(acc_no IN NUMBER) IS
  TYPE e_list IS TABLE OF shubham_bank_transaction%ROWTYPE;
  emo_rec e_list;
BEGIN
  SELECT * BULK COLLECT INTO emo_rec
  FROM shubham_bank_transaction
  WHERE account_no = acc_no
  ORDER BY trans_date DESC
  FETCH FIRST 5 ROWS ONLY;
  FOR i IN 1..emo_rec.COUNT LOOP
   dbms_output.put_line(emo_rec(i).Trans_date||'  '||emo_rec(i).trans_type||'
'||emo_rec(i).amount||'   '||emo_rec(i).description);
  END LOOP;
END;/
begin
shubham_bank(100);
end;
```

**2.Write a procedure that takes a schema name and prints all procedure names in that schema.**

```
CREATE OR REPLACE PROCEDURE list_schema_procedures(p_schema_name IN
VARCHAR2) IS
 v_count NUMBER := 0;
BEGIN
 FOR rec IN (
 SELECT object_name
 FROM all_objects
 WHERE object_type = 'PROCEDURE'
 AND owner = UPPER(p_schema_name)
 ORDER BY object_name
 ) LOOP
 DBMS_OUTPUT.PUT_LINE(v_count||'  '||rec.object_name);
 v_count:=v_count+1;
```

```
 END LOOP;
END;
/
begin
list_schema_procedures('customdev');
end;
```

**3.Develop a function that returns the nth highest salary using only PLSQL , not Rownum or Rank().**

```
set SERVEROUTPUT  on;
create or replace function shubham_highest_sal(n in number) return number is
salary number:=0;
cursor e_cursor is select sal from employee2 group by sal order by sal desc fetch next n
rows only;
begin
open e_cursor;
loop
fetch e_cursor into salary;
exit when e_cursor%notfound;
end loop;
close e_cursor;
return salary;
end;
/
begin
dbms_output.put_line(shubham_highest_sal(3));
end;
```

**4.Write a procedure that takes a schema name and prints all Table names in that schema.**

```
create or replace PROCEDURE shubham_table(p_schema in varchar2) is
begin
for i in ( select table_name from all_tables
```

```
 where owner =upper(p_schema)
 order by table_name ) loop
 dbms_output.put_line(i.table_name);
 end loop;
end;
/
begin
shubham_table('customdev');
end;
```

**5.Create a package inside minimum no  of 3 procedure and 3 functions after that, do select, update ,delete operation  on the tables(use minimum 4 tables.**

```
create or replace PACKAGE shubham_3p_3f is
procedure shubham_p_select(eid in number) ;
procedure shubham_p_select_all;
procedure shubham_p_update(eid in number,ename in varchar2);
function shubham_f_delete(eid in number) return varchar2;
function shubham_f_update(eid in number,inc_sal in number) return varchar2 ;
function shubham_f_total return number;
end shubham_3p_3f;
/


create or replace PACKAGE body shubham_3p_3f is


--shubham_p_select -> PROCEDURE
procedure shubham_p_select(eid in number) is
begin
for i in (select * from employee2 where empno=eid) loop
dbms_output.put_line(i.ename||'  '||i.sal);
end loop;
end shubham_p_select;

--shubham_p_select_all -> PROCEDURE
procedure shubham_p_select_all is
begin
for i in (select * from employee ) loop
dbms_output.put_line(i.empid||' '||i.emp_city);
end loop;
end shubham_p_select_all;

--shubham_p_update -> PROCEDURE
procedure shubham_p_update(eid in number,ename in varchar2)is
begin
update emp set emp_name= ename where emp_id=eid;
dbms_output.put_line('Changed');
end shubham_p_update;
```

```
--shubham_f_delete -> function
function shubham_f_delete(eid in number) return varchar2 is
begin
delete from employee2 where empno=eid;
return 'Deleted Single Record';
end shubham_f_delete;



--shubham_f_update -> function
function shubham_f_update(eid in number,inc_sal in number) return varchar2 is
ename  varchar2(100);
begin
update shubham_geeks set score=score+inc_sal where id=eid;
select name into ename from shubham_geeks where id=eid;
return ename;
end shubham_f_update;
--shubham_f_total -> function
function shubham_f_total return number is
total number:=0;
begin
select sum(sal) into total from employee2;
return total;
end shubham_f_total;
end shubham_3p_3f;
/
begin
shubham_3p_3f.shubham_p_select_all;
--shubham_3p_3f.shubham_p_select(7369);
end;
```

**6.create procedure to fetch the how many tables used inside procedure ,package and Functions .print all the tables name .**

```
CREATE OR REPLACE PROCEDURE GET_TABLE_NAME121(P_NAME IN VARCHAR) --
P_NAME =procedure name,function name ,package name
IS
TYPE ARRay IS TABLE OF VARCHAR2(100);
TABLE_NAME ARRay;
v_count number:=0;
BEGIN
select count(*) into v_count from all_dependencies WHERE
        name = upper(P_NAME)
      AND referenced_owner = upper('CUSTOMDEV')
      AND referenced_type = upper('table');
SELECT  REFERENCED_NAME bulk collect into TABLE_NAME
FROM ALL_DEPENDENCIES
```

```
WHERE
NAME=upper(P_NAME)  and REFERENCED_OWNER=upper('CUSTOMDEV') and
REFERENCED_TYPE=upper('table');
FOR i IN 1..TABLE_NAME.COUNT LOOP
DBMS_OUTPUT.PUT_LINE('TABLE  -> '||TABLE_NAME(i));
END LOOP;
DBMS_OUTPUT.PUT_LINE('Total No. Of Table Present in : '||P_NAME ||' -> ' ||v_count);
END;
 /
 begin
 GET_TABLE_NAME121('shubham_3p_3f');
 end;
```

**7.Write a function that returns the number of working days between two dates, excluding weekends.**

```
create or replace function shubham_date(s_date in date,end_date  in date) return
number is
curr_date date:=s_date;
w_date number:=0;
begin
while curr_date<end_date loop
if to_char(curr_date,'D') not in ('1','7') then --  1->sunday 7->saturday
w_date:=w_date+1;
end if;
curr_date:=curr_date+1;
end loop;
return w_date;
end;
/
begin
   dbms_output.put_line(shubham_date('25-04-25','30-4-25'));
end;
/
```

**8.-> 3 tables is there, i want to know the common columns.**

```
create or replace procedure shubham_common_row(t1 in varchar2,t2 in VARCHAR2,t3
in VARCHAR2) is
v_count number:=0;
begin
select COUNT(DISTINCT table_name) into v_count from user_tab_cols where
table_name IN (
   UPPER('trainingemp'),
   UPPER('employee2'),
   UPPER('dept')
);
for i in
(SELECT column_name
FROM user_tab_cols
WHERE table_name IN (
```

```
   UPPER(t1),
   UPPER(t2),
   UPPER(t3)
)
GROUP BY column_name
HAVING COUNT(DISTINCT table_name)=v_count) loop
dbms_output.put_line(i.column_name);
end loop;
end shubham_common_row;
/
begin
shubham_common_row('trainingemp','employee2','dept');
end;
```

```
set serveroutput on;
/
--1.Write a PL/SQL block to:
--1.1)Calculate bonus:
--30% of salary if rating = 'A' and experience ≥ 10.
--20% if (rating = 'B' or 'C') and experience ≥ 5.
--10% otherwise.
--1.2)Determine promotion eligibility:
--rating = 'A' and bonus ≥ 25,000 → Senior Management.
--rating = 'B' and bonus ≥ 20,000 → Mid-Level Management.
--Otherwise → Not Eligible.
--1.3)Decide appraisal:
--Outstanding if bonus ≥ 20,000 and experience ≥ 7.
--Good if either condition is true.
--Needs Improvement otherwise.
--Display employee_id, bonus, promotion eligibility, and appraisal decision.
declare
rating char(1):=Upper('&rating');
experiance number:=&experiance;
bonus number:=&bonus;
v_sal number:=0;
appraisal varchar2(100);
promotion varchar2(100);
eid employees.employee_id%type;
cursor emp_cursor is select employee_id from employees;
```

```
begin
for i in emp_cursor loop
eid:=i.employee_id;
if rating='A' and experiance>=10 then
v_sal:=bonus*1.3;
ELSIF (rating='B' or rating ='C') and experiance >=5 then
v_sal:=bonus*1.2;
else
v_sal:=bonus*1.1;
end if;
if experiance>=7 and bonus>=20000 then
appraisal:='Outstanding';
ELSIF experiance>=4 and bonus >=10000 then
appraisal:='Good';
else
appraisal:='Needs Improvement';
end if;
if rating='A' and bonus>=25000 then
promotion:='Senior Management ';
ELSIF rating='B' and bonus >=20000 then
promotion:=' Mid-Level Management ';
else
promotion:=' Not Eligible ';
end if;
dbms_output.put_line('Bonus : '||Bonus||' ,Appraisal : '|| appraisal||' ,promotion :
'||promotion ||' ,Employee_id : '||eid ||' '||' rating : '||rating);
end loop;
end;
/
```

--**2.Reverse a number using a WHILE loop and check if it's a palindrome.**

```
declare
v_number number:=&v_number;
v_digit number:=0;
v_sum number:=0;
v_temp number:=0;
begin
v_temp:=v_number;
while v_number>0 loop
v_digit:=mod(v_number,10);
v_number:=trunc(v_number/10);
v_sum:=v_sum*10+v_digit;
end loop;
dbms_output.put_line('Reverse Number : '||v_sum||' '||'Actual Number : '||v_temp);
if v_temp=v_sum then
dbms_output.put_line('Pelindrom Number');
```

```
else
dbms_output.put_line('Not A Pelindrom ');
end if;
end;
```
----------Student using cursors----------

**--3.Write a PL/SQL block using cursors and loop to:**
**--3.1)Calculate the average marks for each student based on their subject**
**--marks.**
**--3.2)Determine if the student has passed or failed:**
**--Pass: Average marks ≥ 50.**
**Fail: Average marks < 50.**
**3.3)Display the student name, average marks, and their pass/fail status**.

```
select * from STUDENT_MOD;
create or replace procedure shubham_student is
v_average number;
status varchar(100);
cursor stu_cursor is select sname,sub1,sub2,sub3,sub4,sub5 from shubhamstudent;
type emp_rec is record(name varchar2(20),v_sub1 number,v_sub2 number,v_sub3
number,v_sub4 number,v_sub5 number);
employ emp_rec;
begin
open stu_cursor;
loop
fetch stu_cursor into employ;
exit when stu_cursor%notfound;
v_average:=(employ.v_sub1+employ.v_sub2+employ.v_sub3+employ.v_sub4+employ.v_
sub5)/5;
dbms_output.put_line(v_average||' ' || employ.name);
end loop;
dbms_output.put_line(v_average);
for i in (select * from employees) loop
if v_average>=50 then
status:='Pass';
else
status:='Fail';
end if;
end loop;
--dbms_output.put_line(status||v_average);
dbms_output.put_line(employ.name||' '||v_average||' '||status);
end;
/
begin
shubham_student();
end;
/
```

**5.The package should have a function that checks whether a given**
**--customer ID exists in the CUSTOMERS table. The function returns TRUE if**

**--the customer ID exists,otherwise FALSE.**

```
create or replace package shubham_fun_cust is
function isCheck(cid in number) return boolean;
end;
/
create or replace package body shubham_fun_cust is
function isCheck(cid in number) return boolean is
v_count number:=0;
begin
select count(*) into v_count from customers where id=cid;
if v_count=0 then
return false;
else
return true;
end if;
end;
end;
/
begin
if shubham_fun_cust.isCheck(6) then
dbms_output.put_line('Customer Present');
else
dbms_output.put_line('Customer not Present');
end if;
end;
```

**6.creating a procedure that retrieves data from two tables using cursors**
**--and processes the data. The first cursor selects all employees from the**
**--EMPLOYEES table, and the second cursor selects all departments from**
**--the DEPARTMENTS table. Write a procedure that loops through the**
**--employees and prints out their department name.**

```
create or replace procedure shubham_retrieves is
emp_rec employees%rowtype;
dept_rec DEPARTMENTS%rowtype;
cursor emp_cursor is select * from employees;
cursor dept_cursor is select * from DEPARTMENTS;
begin
for i in (select * from employees e join departments d on
e.department_id(+)=d.department_id) loop
dbms_output.put_line(i.department_name);
end loop;
end;
/
begin
shubham_retrieves();
end;
```

7.**Write a PL/SQL procedure that takes an employee's ID and performance**

**--rating as inputs. The procedure should check if the employee ID exists in**
**--the employees table and raise an error if it does not.If the employee exists,**
**--update the salary based on the performance rating ('Excellent' for a 10%**
**--increase, 'Good' for a 5% increase, and no change for other ratings).**

```
create or replace procedure shubham_rating(emp_id in number,rating in VARCHAR2) is
invalide_employee_id exception;
v_count number:=0;
begin
select count(*) into v_count from employees where employee_id=emp_id;
if v_count=0 then
raise invalide_employee_id;
else
if rating='EXCELLENCE' THEN
update employees set salary=salary*1.1 WHERE EMPLOYEE_ID=EMP_ID;
dbms_output.put_line('Salary Updated 10% ');
ELSIF rating='GOOD' THEN
update employees set salary=salary*1.05 WHERE EMPLOYEE_ID=EMP_ID;
dbms_output.put_line('Salary Updated 5% ');
ELSE
update employees set salary=salary WHERE EMPLOYEE_ID=EMP_ID;
END IF;
END IF;
EXCEPTION
WHEN invalide_employee_id THEN
dbms_output.put_line('EMPLOYEE NOT FOUND ');
WHEN OTHERS THEN
dbms_output.put_line('ERROR ');
end;
/
declare
rating varchar2(100):=upper('&rating');
begin
shubham_rating(101,rating);
end;
/
rollback;
/
select * from employees;
/
```

**8.Write a PL/SQL block to fetch and display the details of employees who earn above the average salary of all employees.**

```
declare
v_average number:=0;
begin
SELECT
   AVG(salary)
```

```
INTO v_average
FROM
  employees;
dbms_output.put_line(v_average);
FOR i IN (
  SELECT
    *
  FROM
    employees
  WHERE
    salary > v_average
) LOOP
  dbms_output.put_line(i.employee_id
                || ' '
                || i.salary
                || ' '
                || i.first_name
                || ' '
                || i.last_name);
END LOOP;

end;
/
```

**4.1.Calculate the total bill for each patient based on the services they have**
**--received.**
**--2.Determine which department generated the highest revenue for the**
**--hospital.**
**--3.Display:**
**--Patient details (ID, name).**
**--Service-wise costs.**
**--The total bill for each patient.**

```
DECLARE
  CURSOR patiat_cur IS
  SELECT
    p.patient_id,
    p.patient_name,
    s.service_id,
    s.service_name,
    s.cost,
    s.department
  FROM
      patients p
    JOIN patient_services ps ON p.patient_id = ps.patient_id
    JOIN services       s ON ps.service_id = s.service_id;
```

```
  total_cost NUMBER;
BEGIN
  FOR i IN patiat_cur LOOP
    SELECT
      SUM(s.cost)
    INTO total_cost
    FROM
         patients p
      JOIN patient_services ps ON p.patient_id = ps.patient_id
      JOIN services       s ON ps.service_id = s.service_id
    WHERE
      p.patient_id = i.patient_id;
    dbms_output.put_line(i.patient_id
                || ' '
                || i.patient_name
                || ' '
                || ''
                || i.service_name
                || ' '
                || i.cost
                || ' '
                || total_cost
                || ' '
                || i.department);
  END LOOP;
END;
set serveroutput on;
----------------------------Cursor Using rowtype ---------------------------------------
DECLARE
  v_emp employee2%rowtype;
  CURSOR e_cursor IS
  SELECT
    *
  FROM
    employee2;

BEGIN
  OPEN e_cursor;
  LOOP
    FETCH e_cursor INTO v_emp;
    EXIT WHEN e_cursor%notfound;
    dbms_output.put_line(v_emp.ename
                || ' '
                || v_emp.sal
                || ' '
                || v_emp.mgr
                || ' '
```

```
                    || v_emp.empno);
  END LOOP;
  CLOSE e_cursor;
END;
/
-----------------------------Cursor Using %type Attribute ---------------------------------------
DECLARE
  e_name employee2.ename%TYPE;
  CURSOR e_couser IS
  SELECT
    ename
  FROM
    employee2;
BEGIN
  OPEN e_couser;
  LOOP
    FETCH e_couser INTO e_name;
    EXIT WHEN e_couser%notfound;
    dbms_output.put_line(e_name);
  END LOOP;
  CLOSE e_couser;
END;
/
-----------------------------Cursor Using TYPE RECORD Attribute basic loop--------------------
DECLARE
  TYPE e_list1 IS RECORD (
      e_name employee2.ename%TYPE,
      e_sal  employee2.sal%TYPE
  );
  e_list e_list1;
  CURSOR e_cursor IS
  SELECT
    ename,
    sal
  FROM
    employee2;
BEGIN
  OPEN e_cursor;
  LOOP
    FETCH e_cursor INTO e_list;
    EXIT WHEN e_cursor%notfound;
    dbms_output.put_line(e_list.e_name
                    || ' '
                    || e_list.e_sal);
  END LOOP;
  CLOSE e_cursor;
END;
```

```
------------------------------Cursor Using TYPE RECORD Attribute while loop--------------------
SET SERVEROUTPUT ON;
DECLARE
  TYPE e_list1 IS RECORD (
       e_name employee2.ename%TYPE,
       e_sal  employee2.sal%TYPE
  );
  e_list e_list1;
  CURSOR e_cursor IS
  SELECT
    ename,
    sal
  FROM
    employee2;
BEGIN
  OPEN e_cursor;
  FETCH e_cursor INTO e_list;
  WHILE e_cursor%found LOOP
    dbms_output.put_line(e_list.e_name
                || ' '
                || e_list.e_sal);
    FETCH e_cursor INTO e_list;
  END LOOP;
  CLOSE e_cursor;
END;
/
------------------------------Cursor Using ----------------------------------------
DECLARE
  CURSOR c_emps IS
  SELECT
    *
  FROM
    employees;
  v_emps c_emps%rowtype;
  size1  NUMBER := 0;
BEGIN
  SELECT
    COUNT(*)
  INTO size1
  FROM
    employee2;
  OPEN c_emps;
  FOR i IN 1..size1 LOOP
    FETCH c_emps INTO v_emps;
    dbms_output.put_line(v_emps.employee_id
                || ' '
                || v_emps.first_name
```

```
                || ' '
                || v_emps.last_name);

  END LOOP;
  CLOSE c_emps;
END;
/
```
**--An EMPLOYEES table with columns EMPLOYEE_ID, FIRST_NAME, LAST_NAME, DEPARTMENT_ID, and SALARY.**
**--Write a PL/SQL block using a cursor to fetch and display the details of employees working in a**
**--specific department, provided as an input. The details should include EMPLOYEE_ID, FIRST_NAME, LAST_NAME, and DEPARTMENT_ID**
```
set serveroutput on;
DECLARE
  deptid employees.department_id%TYPE := &deptid;
  TYPE emp_rec IS RECORD (
       employeeid employees.employee_id%TYPE,
       firstname  employees.first_name%TYPE,
       lastname   employees.last_name%TYPE,
       salary    employees.salary%TYPE
  );
  elist  emp_rec;
  CURSOR employ_list IS
  SELECT
    employee_id,
    first_name,
    last_name,
    salary
  FROM
    employees;--where DEPARTMENT_ID=deptid
BEGIN
  OPEN employ_list;
  LOOP
    FETCH employ_list INTO elist;
    EXIT WHEN employ_list%notfound;
    dbms_output.put_line(elist.firstname
                || ' '
                || elist.salary);
  END LOOP;
  CLOSE employ_list;
END;
/
```
**--An EMPLOYEES table with columns EMPLOYEE_ID, FIRST_NAME, LAST_NAME, and SALARY.**
**--Write a PL/SQL block that uses a cursor to fetch and display details of employees whose salaries are in the top 10% of all employees' salaries.**

```
set SERVEROUTPUT on;
DECLARE
  v_limit NUMBER;
  CURSOR ecursor IS
  SELECT
    employee_id,
    first_name,
    last_name,
    salary
  FROM
    (
      SELECT
        *
      FROM
        employees
      ORDER BY
        salary DESC
    )
  WHERE
    ROWNUM <= 10;
  emp_rec ecursor%rowtype;
BEGIN
  SELECT
    floor(count(*) * 0.10)
  INTO v_limit
  FROM
    employees;
  IF v_limit = 0 THEN
    v_limit := 1;
  END IF;
  OPEN ecursor;
  LOOP
    FETCH ecursor INTO emp_rec;
    EXIT WHEN ecursor%notfound;
    dbms_output.put_line(emp_rec.employee_id);
  END LOOP;
  CLOSE ecursor;
END;
/
DECLARE
  v_limit NUMBER;
  CURSOR ecursor IS
  SELECT
    employee_id,
    first_name,
    last_name,
    salary
```

```
  FROM
    (
      SELECT
        *
      FROM
        employees
      ORDER BY
        salary DESC
    )
  WHERE
    ROWNUM <= 10;
  emp_rec ecursor%rowtype;
BEGIN
  -- Calculate top 10% count
  SELECT
    floor(count(*) * 0.10)
  INTO v_limit
  FROM
    employees;
  IF v_limit = 0 THEN
    v_limit := 1;
  END IF;
  FOR emp_rec IN (
    SELECT
      employee_id,
      first_name,
      last_name,
      salary
    FROM
      (
        SELECT
          *
        FROM
          employees
        ORDER BY
          salary DESC
      )
    WHERE
      ROWNUM <= v_limit
  ) LOOP
    dbms_output.put_line('ID: '
                         || emp_rec.employee_id
                         || ', Name: '
                         || emp_rec.first_name
                         || ' '
                         || emp_rec.last_name
                         || ', Salary: '
```

```
                || emp_rec.salary);
  END LOOP;
END;
/
```
**--Write a PL/SQL block using a cursor to calculate the total salary expenditure for each department.**
**--Display the DEPARTMENT_NAME along with the total salary expenditure in the format: "Department:**
**--[DEPARTMENT_NAME], Total Salary Expenditure: [TOTAL_SALARY]".**
```
DECLARE
  CURSOR ecursor IS
  SELECT
    d.dept_name,
    SUM(e.salary) AS total_sal
  FROM
      department d
    JOIN employees e ON e.department_id = d.department_id
  GROUP BY
    d.dept_name;

  emp_rec ecursor%rowtype;
BEGIN
  OPEN ecursor;
  LOOP
    FETCH ecursor INTO emp_rec;
    EXIT WHEN ecursor%notfound;
    dbms_output.put_line('Department: '
                || emp_rec.dept_name
                || ' , Total Salary Expenditure:'
                || emp_rec.total_sal);
  END LOOP;
  CLOSE ecursor;
END;
/
```
**--Write a PL/SQL block using two open cursors to fetch:**
**--Employee details (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY).**
**--Department details (DEPARTMENT_ID, DEPARTMENT_NAME) at the same time.**
**--Display the results in the format: "Employee ID: [EMPLOYEE_ID], Name: [FIRST_NAME] [LAST_NAME], Department: [DEPARTMENT_NAME], Salary: [SALARY]".**
```
DECLARE
  dtable department%rowtype;
  etable employees%rowtype;
  CURSOR ecursor IS
  SELECT
    *
  FROM
```

```
    employees;
  CURSOR dcursor IS
  SELECT
    *
  FROM
    department;
BEGIN
  OPEN ecursor;
  OPEN dcursor;
  LOOP
    FETCH ecursor INTO etable;
    FETCH dcursor INTO dtable;
    EXIT WHEN
      ecursor%notfound
      AND dcursor%notfound;
    dbms_output.put_line('Employee ID: '
                || etable.employee_id
                || ' , Name: '
                || etable.first_name
                || ' '
                || etable.last_name
                || ', Department: '
                || dtable.dept_name
                || ', Salary: '
                || etable.salary);
  END LOOP;
  CLOSE dcursor;
  CLOSE ecursor;
END;
/
```

**--Write a PL/SQL block to fetch employee and department details using a cursor and a record, and display the results in the following format:**
**--"Employee ID: [EMPLOYEE_ID], Name: [FIRST_NAME] [LAST_NAME], Department: [DEPARTMENT_NAME], Salary: [SALARY]".**

```
DECLARE
  CURSOR employ_list IS
  SELECT
    d.dept_name,
    e.employee_id,
    e.first_name,
    e.last_name,
    e.salary
  FROM
       employees e
    JOIN department d ON d.department_id = e.department_id;
  TYPE emp_rec IS RECORD (
       deptname   department.dept_name%TYPE,
```

```
        employeeid employees.employee_id%TYPE,
        firstname  employees.first_name%TYPE,
        lastname   employees.last_name%TYPE,
        salary     employees.salary%TYPE
    );
    elist emp_rec;
BEGIN
    OPEN employ_list;
    FETCH employ_list INTO elist;
    LOOP
      FETCH employ_list INTO elist;
      EXIT WHEN employ_list%notfound;
      dbms_output.put_line('Employee ID: '
                    || elist.employeeid
                    || ' Name: '
                    || elist.firstname
                    || ' '
                    || elist.lastname
                    || ' Department: '
                    || elist.deptname
                    || ' Salary: '
                    || elist.salary);
    END LOOP;
    CLOSE employ_list;
END;
/
```

**--. How many steps are associated when working with an explicit cursor?**

**--There are four working phase of cursor 1-> Declaration 2-> Open 3->fetch 4->close**

```
set SERVEROUTPUT on;
declare
type e_name is varray(50) of employee2.ename%type;
emp_ename e_name:=e_name();
idx number:=1;
begin
for i in (select * from employee2) loop
emp_ename.extend;
emp_ename(idx):=i.job;
idx:=idx+1;
end loop;
for x in 1..emp_ename.count() loop
dbms_output.put_line(emp_ename(x)||' '||emp_ename.count());
end loop;
end;
/
declare
type e_name is varray(50) of employee2%rowtype;
emp_ename e_name:=e_name();
```

```
begin
select * BULK COLLECT into emp_ename from employee2;
for x in 1..emp_ename.count() loop
dbms_output.put_line(emp_ename(x).ename||' '||emp_ename(x).sal);
end loop;
end;
/
declare
type e_list is record(
v_ename employee2.ename%type,
v_sal employee2.sal%type
);
emp e_list;
begin
select ename,sal into emp from employee2 where empno =7369;
dbms_output.put_line(emp.v_ename||' '||emp.v_sal);
end;
/

DECLARE
 TYPE e_list IS TABLE OF employees.first_name%TYPE INDEX BY PLS_INTEGER;
 emps e_list;
 idx PLS_INTEGER;
BEGIN
 emps(100) := 'Bob';
 emps(120) := 'Sue';
  emps(130) := 'Suea';
  idx:=emps.first;
--  FOR i IN emps.first()..emps.last() LOOP
-- if emps.exists(i) then
--   dbms_output.put_line(emps(i));
--  end if;
--  END LOOP;
while idx is not null loop
dbms_output.put_line(emps(idx));
idx:=emps.next(idx);
end loop;
END;
/
declare
type e_name is varray(50) of employee2.ename%type;
emp_ename e_name:=e_name();
idx number:=1;
size1 number:=0;
begin
select count(*) into size1 from employee2;
for i in (select * from employee2) loop
```

```
emp_ename.extend;
emp_ename(idx):=i.job;
idx:=idx+1;
end loop;
for x in 1..emp_ename.count() loop
dbms_output.put_line(emp_ename(x)||' '||emp_ename.count());
end loop;
dbms_output.put_line(size1);
end;
/
declare
type e_name is varray(50) of employee2.ename%type;
emp_ename e_name:=e_name();
idx number:=1;
size1 number:=0;
begin
select count(*) into size1 from employee2;
for i in 1..size1 loop
emp_Ename.extend;
select ename into emp_Ename(idx) from employee2 ;
idx:=idx+1;
END LOOP;
for x in 1..emp_ename.count() loop
dbms_output.put_line(emp_ename(x)||' '||emp_ename.count());
end loop;
dbms_output.put_line(size1);
end;
```

**/--writing a procedure that will:**
**--Accept a department ID as input.**
**--Retrieve all employees in that department.**
**--Update their salary by applying a 10% increment.**
**--Print out the updated employee details.**

```
CREATE OR REPLACE PROCEDURE updatesalaryindept (
   p_dept_id IN VARCHAR
) IS
   CURSOR emp_cursor IS
   SELECT
      employee_id,
      first_name,
      salary
   FROM
      employees
   WHERE
      department_id = p_dept_id;
   v_empno employees.employee_id%TYPE;
   v_ename employees.first_name%TYPE;
   v_sal   employees.salary%TYPE;
```

```
BEGIN
  FOR r_emp IN emp_cursor LOOP
  -- Retrieve current employee details
    v_empno := r_emp.employee_id;
    v_ename := r_emp.first_name;
    v_sal := r_emp.salary;
    UPDATE employees
    SET
      salary = v_sal * 0.1
    WHERE
      employee_id = v_empno;
    dbms_output.put_line('Updated salary for '
                || v_ename
                || ' (Emp No: '
                || v_empno
                || ') to: '
                || v_sal * 1.1);
  END LOOP;
--  COMMIT;
END updatesalaryindept;
/
BEGIN
  updatesalaryindept('D1');
END;
```

**Write a PL/SQL procedure to insert a new employee into the emp table. The procedure should:**
**--Accept employee details as parameters.**
**--If an employee already exists (based on empno), raise an exception and print an appropriate message**.

```
CREATE OR REPLACE PROCEDURE insertemployee (
  p_empno   IN NUMBER,
  p_ename   IN VARCHAR2,
  p_job     IN VARCHAR2,
  p_mgr     IN NUMBER,
  p_hiredate IN DATE,
  p_sal     IN NUMBER,
  p_comm    IN NUMBER,
  p_deptno  IN NUMBER
) IS
BEGIN
  IF EXISTS (
    SELECT
      *
    FROM
      employee2
    WHERE
      empno = p_empno
```

```
  ) THEN
    dbms_output.put_line('Error: Employee with empno '
                || p_empno
                || ' already exists.');
  ELSE
    INSERT INTO emp (
      empno,
      ename,
      job,
      mgr,
      hiredate,
      sal,
      comm,
      deptno
    ) VALUES ( p_empno,
          p_ename,
          p_job,
          p_mgr,
          p_hiredate,
          p_sal,
          p_comm,
          p_deptno );
--  COMMIT;
    dbms_output.put_line('Employee '
                || p_ename
                || ' inserted successfully.');
  END IF;
EXCEPTION
  WHEN OTHERS THEN
    dbms_output.put_line('Unexpected error: ' || sqlerrm);
END insertemployee;
BEGIN
  insertemployee();
END;
```

**3. Create a stored procedure that:**
**--Displays employees earning more than $50,000.**
**--Increases the salary of employees in the department no 20 by 10%.**
**--Finds and displays the most recently joined employee.**

```
CREATE OR REPLACE PROCEDURE storing IS
--  elist employees%rowtype;
  elist1 employees%rowtype;
  hdate employees.hire_date%TYPE;
  sal number;
BEGIN
  SELECT
    *
  INTO elist
```

```
  FROM
    employees
  WHERE
    salary >= 50000;
  SELECT
    employees.*,
    salary * 1.20
  INTO elist1,sal
  FROM
    employees
  WHERE
    department_id = 'D1';
  SELECT
    MAX(hire_date)
  INTO hdate
  FROM
    employees;
--   dbms_output.put_line(elist.first_name);
  dbms_output.put_line(hdate);
END storing;
BEGIN
  storing;
END;
```

**--4. Write a PL/SQL procedure to delete employees whose salary is below a specified threshold(like below 1000 sal).**

```
--Display the number of employees deleted.
CREATE OR REPLACE PROCEDURE delete_employees (
  threshold IN NUMBER
) AS
  v_count NUMBER := 0;
BEGIN
  SELECT
    COUNT(*)
  INTO v_count
  FROM
    employees
  WHERE
    salary < threshold;
  DELETE FROM employees
  WHERE
    salary < threshold;
  dbms_output.put_line(v_count
                || ' employees '
                || threshold);
--   COMMIT;
END;
BEGIN
```

```
  delete_employees(1000);
END;
```
--**Write a PL/SQL procedure to retrieve the top N highest-paid employees.(parameter pass the N value).**
```
CREATE OR REPLACE PROCEDURE total_sal_dept (
  n IN NUMBER
) IS
  emp_rec employees%rowtype;
  CURSOR ecursor IS
  SELECT  *   FROM employees
  WHERE
    salary IS NOT NULL
    AND ROWNUM <= n;
BEGIN
  OPEN ecursor;
  LOOP
    FETCH ecursor INTO emp_rec;
    EXIT WHEN ecursor%notfound;
    dbms_output.put_line(emp_rec.first_name);
  END LOOP;
END;
/
BEGIN
  total_sal_dept(5);
END;
/
```

**6.Write a procedure to calculate and display the total salary for each department.**
```
CREATE OR REPLACE PROCEDURE total_sal_dept (
  dept IN VARCHAR
) IS
  total NUMBER := 0;
BEGIN
  SELECT
    SUM(e.salary)
  INTO total
  FROM
       employees e
    JOIN department d ON e.department_id = d.department_id
  WHERE
    d.department_id = dept
  GROUP BY
    e.department_id;
  dbms_output.put_line(total);
EXCEPTION
  WHEN no_data_found THEN
```

```
    dbms_output.put_line('No Data Found');
END total_sal_dept;
BEGIN
  total_sal_dept(dept => 'D2');
END;
/
```
--7. **Create a procedure that takes a VARRAY of department IDs as input and displays the names of employees working in those departments**.
```
CREATE OR REPLACE TYPE e_list1 IS
  TABLE OF VARCHAR2(100);
/
CREATE OR REPLACE PROCEDURE employee_details (
  departid IN e_list1
) AS
BEGIN
  FOR i IN 1..departid.count LOOP
    dbms_output.put_line('Department ID: ' || departid(i));
    FOR x IN (
      SELECT
        first_name,last_name
      FROM employees
      WHERE
        department_id = departid(i)
    ) LOOP
      dbms_output.put_line(' - '|| x.first_name|| ' ‘ || x.last_name);
    END LOOP;
    dbms_output.put_line('Total Employees in Department ' || departid(i));
  END LOOP;
END;
/
DECLARE
  deptid e_list1;
BEGIN
  deptid := e_list1('D1', 'D2', 'D3');
  employee_details(departid => deptid);
END;
/
set serveroutput on;
select * from employees;
Create Or Replace Procedure Total_Sal_Dept(N In Number) Is
emp_rec employees%rowtype;
cursor ecursor is select *  from employees where salary is not null and rownum<=n;
begin
open ecursor;
loop
fetch ecursor into emp_rec;
exit when ecursor%notfound;
```

```
dbms_output.put_line(emp_rec.first_name);
 end loop;
end;
/
begin
total_sal_dept(6);
end;
/
create or replace procedure storing11 is
elist employees%rowtype;
elist1 employees%rowtype;
hdate employees.hire_date%type;
inc employees.salary%type;
begin
for i in (select * from employees) loop
select * into elist from employees where salary >=50000;
--select employees.*,salary*1.20 as increamet into elist1,inc from employees where
department_id='D1';
--select max(hire_date) into hdate from employees;
dbms_output.put_line(elist.first_name);
end loop;
select max(hire_date) into hdate from employees;
dbms_output.put_line(hdate);
end;
/
begin
storing11();
end;
/
CREATE OR REPLACE PROCEDURE UpdateSalaryInDept(p_dept_id IN varchar) IS
  CURSOR emp_cursor IS
    SELECT employee_id, first_name, salary
    FROM employees
    WHERE department_id = p_dept_id;
  v_empno employees.employee_id%TYPE;
  v_ename employees.first_name%TYPE;
  v_sal employees.salary%TYPE;
BEGIN
  FOR r_emp IN emp_cursor LOOP
    v_empno := r_emp.employee_id;
    v_ename := r_emp.first_name;
    v_sal := r_emp.salary;
    UPDATE employees
    SET salary = v_sal * 0.1
    WHERE employee_id = v_empno;
    DBMS_OUTPUT.PUT_LINE('Updated salary for ' || v_ename || ' (Emp No: ' || v_empno ||
') to: ' || v_sal * 1.1);
```

```
 END LOOP;
--  COMMIT;
END UpdateSalaryInDept;
 /
 begin
 UpdateSalaryInDept('D1');
 end;
 /
set serveroutput on;
create or replace procedure storing is
elist employees%rowtype;
elist1 employees%rowtype;
hdate employees.hire_date%type;
esal employees.salary%type;
cursor e_cursor is select * from employees where salary >=50000;
cursor e_cursor1 is select employees.*,salary*1.20 as increamet  from employees
where department_id='D1';
begin
open e_cursor ;
open e_cursor1;
loop
fetch e_cursor into elist;
--fetch e_cursor1 into elist1,esal;
exit when e_cursor%notfound and e_cursor1%notfound;
dbms_output.put_line(elist.first_name);
end loop;
close e_cursor;
close e_cursor1;
select max(hire_date) into hdate from employees;
dbms_output.put_line(hdate);
end storing;
/
begin
storing();
end;
/
CREATE OR REPLACE PROCEDURE delete_employees(p_threshold IN NUMBER) AS
   v_count NUMBER := 0;
BEGIN
   SELECT COUNT(*) INTO v_count
   FROM employees
   WHERE salary < p_threshold;
   DELETE FROM employees
   WHERE salary < p_threshold;
   DBMS_OUTPUT.PUT_LINE(v_count || ' employees ' || p_threshold);
--    COMMIT;
END;
```

```
/
begin
delete_employees(2000);
end;
/
set serveroutput on;
create or replace procedure employee_details(departid in deptid) as
begin
for i in 1..departid.count loop
dbms_output.put_line(departid(i));
for x in (select * from employees where department_id=departid(i)) loop
dbms_output.put_line(x.first_name||' '||x.last_name);
end loop;
end loop;
end employee_details;
/
declare
type e_list is varray(50) of employees.department_id%type ;
deptid e_list := e_list('D1','D2','D3','D4');
BEGIN
   employee_details(deptid);
end;
/
select * from trainingemp;
SELECT * FROM demoemp;
CREATE TABLE shubham_book_log (
   log_id INT  PRIMARY KEY,
   book_id INT,
   log_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   action VARCHAR(50)
);
desc shubham_books;
create or replace trigger shubham_trigger_insert
before insert on shubham_books
for each row
begin
--insert into shubham_book_log(log_id,book_id,log_date,action)
values(1,:new.book_id,sysdate,'book added');
delete from shubham_book_log where log_id=:new.book_id;
end;
/
insert into shubham_books (book_id,title,author,added_date)
values(1,'Sql','Sharath',sysdate);
select * from shubham_spring;
delete from shubham_books;
set serveroutput on;
declare
```

```
type coll_table is table of employee2%rowtype ;
emp coll_table;
v_count number:=0;
begin
select count(*) into v_count from employee2;
select * bulk collect into emp from employee2;
--for i in 1..v_count loop
for i in emp.first..emp.last loop
dbms_output.put_line('Employee Name : '||emp(i).ename);
dbms_output.put_line('Salary : '||emp(i).sal);
end loop;
dbms_output.put_line('Total No of Data : '||v_count);
end;
/
DECLARE
  TYPE salary IS TABLE OF NUMBER INDEX BY VARCHAR2(20);
  salary_list salary;
BEGIN
 null;
END;
/
declare
type array is table of employee2%rowtype;
emp_rec array;
begin
select sal bulk collect into emp_rec from employee2;
--dbms_output.put_line('Salary : '||emp.sal);
forall i in emp_rec.first..emp_rec.last
update employee2 set sal=emp_rec(i).sal*1.1 where empno=emp_rec(i).empno ;
dbms_output.put_line('Updated');
end;
/
select * from employee2;
rollback;
/
declare
CURSOR emp_cursor is select * from employee2;
type emp_table is table of employee2%rowtype;
emp1 emp_table;
r_list employee2%rowtype;
begin
open emp_cursor;
loop
fetch emp_cursor into r_list;
exit when emp_cursor%notfound;
--emp1.extend;
--emp1(emp1.first):=r_list;
```

```
dbms_output.put_line('Salary : '||r_list.sal);
end loop;
close emp_cursor;
end;
/
declare
type coll_table is table of employee2%rowtype ;
emp coll_table;
v_count number:=0;
begin
select count(*) into v_count from employee2;
select * bulk collect into emp from employee2;
loop
exit when emp%notfound;
dbms_output.put_line('Employee Name : '||emp.ename);
dbms_output.put_line('Salary : '||emp.sal);
end loop;
dbms_output.put_line('Total No of Data : '||v_count);
end;
SET SERVEROUTPUT ON;
CREATE OR REPLACE PROCEDURE list_schema_procedures(p_schema_name IN
VARCHAR2) IS
  v_count NUMBER := 0;
BEGIN
  FOR rec IN (
    SELECT object_name
    FROM all_objects
    WHERE object_type = 'PROCEDURE'
     AND owner = UPPER(p_schema_name)
    ORDER BY object_name
  ) LOOP
    DBMS_OUTPUT.PUT_LINE('Procedure: ' || rec.object_name);
  END LOOP;
END;
/
EXEC list_schema_procedures('customdev');
create view shubham_view as select * from employee2;
select * from shubham_view;
select * from all_dependencies /*where owner='CUSTOMDEV'*/;
select object_name from all_objects where owner ='CUSTOMDEV' and
object_type='PROCEDURE' order by object_name;
/
set SERVEROUTPUT on;
create or replace procedure adjust_salary(p_emp_id in number,p_adjustment in out
number) is
v_current_salary number;
Begin
```

```
select salary into v_current_salary from employees where employee_id = p_emp_id;
p_adjustment :=v_current_salary + p_adjustment;

update employees set salary = p_adjustment where employee_id = p_emp_id ;

end adjust_salary;
/
declare
total number(20):=1000;
begin
adjust_salary(101,p_adjustment=>total);
dbms_output.put_line(total);
end;
/
rollback;
select * from employees_pl;
commit;
/
create or replace procedure accept_inp(dept_id in number) is
emp_rec employees%rowtype;
cursor cus is select f_name, sal from employees_pl where dept_id = dept_id;
begin
update employees_pl set sal =sal+(0.1 * sal) where dept_id = dept_id;
for i in cus loop
dbms_output.put_line(i.f_name || ' ' || i.sal);
end loop;
end;
/
begin
accept_inp(10);
end;
/
rollback;
/
select * from employee2;
SELECT *
FROM employee2
PIVOT (
   SUM(sal)
   FOR job IN ('clerk')
) AS pivot_shubham;
set serveroutput on;

create or replace function shubham_record(eid in  number) return
employee2%rowtype is
e_rec employee2%rowtype;
```

```
begin
select * into e_rec from employee2 where empno=eid;
return e_rec;
end;
/
declare
e_list employee2%rowtype;
begin
e_list:=shubham_record(7369);
dbms_output.put_line(e_list.ename||'  '||e_list.sal);
end;
/
create or replace function shubham_record1(n in number) return  employee2%rowtype
is
e_rec employee2%rowtype;
begin
select * into e_rec from employee2 offset n-1 rows fetch next 1 rows only;
return e_rec;
end;
/
declare
e_list employee2%rowtype;
begin
e_list:=shubham_record1(3);
dbms_output.put_line(e_list.ename||'  '||e_list.sal);
end;
/
SELECT  name as tableName, type
     FROM all_source
     WHERE type IN ('PROCEDURE', 'FUNCTION', 'PACKAGE', 'PACKAGE BODY') and
owner='CUSTOMDEV' order by name;
     /
     SELECT  REFERENCED_NAME
FROM ALL_DEPENDENCIES
WHERE
NAME=upper('package_sharath2')  and REFERENCED_OWNER=upper('CUSTOMDEV')
and REFERENCED_TYPE=upper('table') ;

--Can you explain how to use a nested table of records in PL/SQL? ->  syntax type
type_name is table of datatype

DECLARE
 TYPE e_list IS TABLE OF employee2.ename%type index by PLS_INTEGER;
 emps e_list;
 idx  number := 1;
BEGIN
 FOR x IN 7000 .. 8000 LOOP
```

```
 begin
  SELECT ename INTO emps(idx)
  FROM   employee2
  WHERE  empno = x;
  idx := idx + 1;
  EXCEPTION
  when no_data_found then
 null;
  end;
 END LOOP;
 emps.delete(3);
 FOR i IN 1..emps.count() LOOP
  IF emps.exists(i) THEN
    dbms_output.put_line(emps(i));
  END IF;
 END LOOP;
END;
/

DECLARE
 TYPE e_list IS TABLE OF employees.first_name%TYPE INDEX BY PLS_INTEGER;
 emps e_list;
BEGIN
 emps(100) := 'Bob';
 emps(120) := 'Sue';
 FOR i IN emps.first()..emps.last() LOOP
  if emps.exists(i) then
  dbms_output.put_line(emps(i));
end if;
 END LOOP;
END;

/
DECLARE
 TYPE e_list IS TABLE OF employee2.ename%type index by PLS_INTEGER;
 emps e_list;
 idx  number := 1;
BEGIN
 FOR x IN (select * from employee2) LOOP
 begin
  emps(idx):=x.ename;
  idx := idx + 1;
  EXCEPTION
  when no_data_found then
 null;
  end;
 END LOOP;
```

```
 emps.delete(3);
 FOR i IN 1..emps.count() LOOP
  IF emps.exists(i) THEN
    dbms_output.put_line(emps(i));
  END IF;
 END LOOP;
END;
/
DECLARE
 TYPE e_list IS TABLE OF employee2.ename%type;
 emps e_list := e_list();
 idx  number := 1;
BEGIN
 FOR x IN 7000 .. 8000 LOOP
 begin
  emps.extend;
  SELECT job INTO emps(idx)
  FROM   employee2
  WHERE  empno = x;
  idx := idx + 1;
  EXCEPTION
  when no_data_found then
  emps.trim;
  end;
 END LOOP;
 emps.delete(3);
 FOR i IN 1..emps.count() LOOP
  IF emps.exists(i) THEN
    dbms_output.put_line(emps(i));
  END IF;
 END LOOP;
END;
set serveroutput on;
declare
type e_list is varray(10) of varchar(50);
emp e_list;
begin
emp:=e_list('alice','bob','prince','smith');
for i in 1..emp.count() loop
dbms_output.put_line(emp(i));
end loop;
end;
/
declare
type e_list is varray(10) of char(1);
v_name e_list;
begin
```

```
v_name :=e_list('S','H','U','B','H','A','M');
for i in 1..v_name.count() loop
dbms_output.put_line(v_name(i));
end loop;
end;
/
DECLARE
TYPE E_LIST IS VARRAY(20) OF VARCHAR(50);
EMP E_LIST:=E_LIST();
IDX NUMBER :=1;
BEGIN
FOR I IN 7000..8000 LOOP
EMP.EXTEND;
BEGIN
SELECT JOB INTO EMP(IDX) FROM EMPLOYEE2 WHERE EMPNO=I;
IDX:=IDX+1;
EXCEPTION
WHEN NO_DATA_FOUND THEN
EMP.TRIM;
END;
END LOOP;
FOR X IN 1..EMP.COUNT() LOOP
dbms_output.PUT_LINE(EMP(X));
END LOOP;
dbms_output.PUT_LINE(EMP.COUNT());
END;
/
------------------------------Associate Array-----------------------------------
declare
type e_list is TABLE of employee2.job%type INDEX by pls_integer;
emp e_list;
idx number:=1;
begin
for i in 7000..8000 loop
begin
select job into emp(idx)
from employee2 where empno=i;
idx:=idx+1;
EXCEPTION
when no_data_found then
dbms_output.PUT_LINE('sqlerrm');
end;
end loop;
FOR X IN 1..EMP.COUNT() LOOP
dbms_output.PUT_LINE(EMP(X));
END LOOP;
end;
```

```
/-------------------------------------------------Error using for loop----------
declare
type e_list is table of varchar(50) INDEX by PLS_INTEGER;
emp e_list;
begin
emp(100):='alice';
emp(120):='Bob';
for i in emp.first()..emp.last() loop
dbms_output.put_line(emp(i));
end loop;
end;

------------------------------using while loop----------------------------------------

declare
type e_list is table of varchar(50) INDEX by PLS_INTEGER;
emp e_list;
idx PLS_INTEGER;
begin
emp(100):='alice';
emp(120):='Bob';
idx:=emp.first;
while idx is not null loop
 dbms_output.put_line(emp(idx));
 idx:=emp.next(idx);
end loop;
end;
/
set serveroutput on;
--------------Traverse using for loop----------------------
declare
type Traverse is varray(10) of VARCHAR(20);
emp Traverse;
begin
emp:=Traverse('Alice','Bob','Smith','Charli');
for i in 1..emp.count() loop
dbms_output.put_line(emp(i));
end loop;
end;
/
----------------Traverse using while loop-------------------------

declare
type Traverse is varray(10) of VARCHAR(20);
emp Traverse;
i number :=1;
begin
```

```
emp:=Traverse('Alice','Bob','Smith','Charli');
while i<=emp.count loop
dbms_output.put_line(emp(i));
i:=i+1;
end loop;

end;
/
```

---

```
declare
type arr is varray(20) of number ;
v_sum number:=1;
sumArray arr;
begin
sumArray:=arr(1,51,5,8,5,4);
for i in 1..sumArray.count loop
v_sum:=v_sum+sumArray(i);
end loop;
dbms_output.put_line(v_sum);
end;
/
```

---------- sum of salary of employee2 table----------

```
DECLARE
TYPE E_LIST IS VARRAY(20) OF employee2.sal%type;
EMP E_LIST:=E_LIST();
IDX NUMBER :=1;
total_sal number:=0;
BEGIN
FOR I IN 7000..8000 LOOP
EMP.EXTEND;
BEGIN
SELECT sal INTO EMP(IDX) FROM EMPLOYEE2 WHERE EMPNO=I;
--total_sal:=total_sal+emp(idx);
select sum(sal) into total_sal from employee2 ;
IDX:=IDX+1;
EXCEPTION
WHEN NO_DATA_FOUND THEN
EMP.TRIM;
END;
END LOOP;
FOR X IN 1..EMP.COUNT() LOOP
dbms_output.PUT_LINE(EMP(X));
END LOOP;
```

```
dbms_output.PUT_LINE('total '||total_sal);
END;
/
--------------------------------------------------------------------------------
DECLARE
 TYPE e_list IS TABLE OF employee2.ename%type;
 emps e_list := e_list();
 idx  number := 1;
BEGIN
 FOR x IN 7000 .. 8000 LOOP
 begin
  emps.extend;
  SELECT ename INTO emps(idx)
  FROM   employee2
  WHERE  empno = x;
  idx := idx + 1;
  EXCEPTION
  when no_data_found then
  emps.trim;
  end;
 END LOOP;
 emps.delete(3);
 FOR i IN 1..emps.count() LOOP
  IF emps.exists(i) THEN
    dbms_output.put_line(emps(i));
  END IF;
 END LOOP;
END;
/
--3.Write a PL/SQL block to: Create a VARRAY of size 10. Initialize it with 6 elements.
Display the current size and the maximum capacity of the array.

declare
type e_list is varray(10) of employee2.ename%type;
emp e_list;

begin
emp :=e_list('Amit','Rohit','Shubham','Rajendra','Sharath','Pankaj');

dbms_output.put_line('current size of Array is : '||' '||emp.count());
dbms_output.put_line('maximum capacity of the array is : '||' '||emp.limit());

end;

--.Create a PL/SQL block that uses a VARRAY to store 5 employee salaries. Iterate
through the array using a loop to calculate and display the total salary. using for in loop
without cursor
```

```
declare
type e_sal is varray(5) of employee2.sal%type;
emp_sal e_sal:=e_sal();
total_sal employee2.sal%type;
idx PLS_INTEGER:=1;
begin
for i in (select sal from employee2 where rownum<=5) loop
emp_sal.extend;
emp_sal(idx):=i.sal;
idx:=idx+1;
end loop;

for x in 1..emp_sal.count() loop
dbms_output.put_line(emp_sal(x));
end loop;
end;
/
```

--.Create a PL/SQL block that uses a VARRAY to store 5 employee salaries. Iterate through the array using a loop to calculate and display the total salary. using for in loop with cursor

```
declare
type e_sal is varray(5) of employee2.sal%type;
emp_sal e_sal:=e_sal();
total_sal employee2.sal%type:=0;
idx PLS_INTEGER:=1;
cursor e_cursor is select sal from employee2 where rownum<=5;
begin
for i in e_cursor loop
emp_sal.extend;
emp_sal(idx):=i.sal;
idx:=idx+1;
total_sal:=total_sal+i.sal;
end loop;
for x in 1..emp_sal.count() loop
dbms_output.put_line(emp_sal(x));
end loop;
dbms_output.put_line('total_sal'||' '||total_sal);
end;
/
SELECT
  *
FROM
```

```
   trainingemp;

set SERVEROUTPUT on;

--------------procedure  using in ------------


CREATE OR REPLACE PROCEDURE shubhamprocedure (
   emp_id IN NUMBER
) IS
   v_sal      NUMBER;
   v_annualsal NUMBER;
BEGIN
   SELECT
      sal,
      sal * 12
   INTO
      v_sal,
      v_annualsal
   FROM
      trainingemp
   WHERE
      empno = emp_id;

   dbms_output.put_line('salari'
                        || ' '
                        || v_sal);
   dbms_output.put_line('Annual salari'
                        || ' '
                        || v_annualsal);
EXCEPTION
   WHEN no_data_found THEN
      dbms_output.put_line('No employee found');
END shubhamprocedure;
/
--declare
--emp_id number;
BEGIN
   shubhamprocedure(emp_id => 7499);
END;
/

--------------procedure using in,out -----------------------


CREATE OR REPLACE PROCEDURE shubhamprodecure (
   emp_id IN NUMBER,
```

```
  v_sal  OUT NUMBER
) IS
BEGIN
  SELECT
    sal
  INTO v_sal
  FROM
    trainingemp
  WHERE
    empno = emp_id;

EXCEPTION
  WHEN no_data_found THEN
    dbms_output.put_line('No employee found');
END shubhamprodecure;
/
--exec shubhamprodecure(emp_id=>7369,v_sal);

DECLARE
  v1_sal NUMBER;
BEGIN
  shubhamprodecure(
    emp_id => 7369,
    v_sal  => v1_sal
  );
  dbms_output.put_line(v1_sal);
END;
/

-----------------
```

```
CREATE TABLE employee2
  AS
    (
      SELECT
        empno,
        ename,
        job,
        hiredate,
        sal,
        mgr,
        deptno,
```

```
            comm
          FROM
            trainingemp
--          WHERE
--             empno = NULL
      );

SELECT
  *
FROM
  employee2;
  commit;


SELECT
  *
FROM
  trainingemp;

DROP TABLE employee2;

DELETE FROM employee2;

CREATE SEQUENCE s4 START WITH 1 INCREMENT BY 1 MAXVALUE 1000 CYCLE CACHE
24;

DROP SEQUENCE s5;

s4.nextval;

INSERT INTO employee2 (
  empno,
  sal
) VALUES ( s4.NEXTVAL,
       s5.NEXTVAL );

CREATE SEQUENCE s5 START WITH 1 INCREMENT BY 10 MAXVALUE 1000 CYCLE
NOCACHE;

CREATE VIEW v31 AS
  SELECT
    *
  FROM
    trainingemp
  WHERE
    empno = 7369;
```

```
SELECT
  *
FROM
  v31;

SELECT
  ename,
  empno,
  sal,
  sal * 12 AS annul_sal
FROM
  v31;

CREATE UNIQUE INDEX shindex ON
  trainingemp (
    ename
  );

DROP INDEX shindex;

SELECT
  ename,
  empno,
  sal
FROM
  trainingemp
ORDER BY
  empno DESC
--offset (select count(*) from TRAININGEMP) -5 rows
OFFSET 0 ROWS FETCH NEXT 5 ROWS ONLY;

SELECT
  *
FROM
  trainingemp
--order by empno
OFFSET (
  SELECT
    COUNT(*)
  FROM
    trainingemp
) - 6 ROWS;

SELECT
  *
FROM
  trainingemp
```

```
--order by empno
OFFSET (
  SELECT
    COUNT(*)
  FROM
    trainingemp
) - 6 ROWS FETCH NEXT 6 ROWS ONLY;

SELECT
  *
FROM
  (
    SELECT
      trainingemp.*,
      ROWNUM AS rownum1
    FROM
      trainingemp
  )
WHERE
  rownum1 >= 6;

SELECT
  substr(ename, -1, 1)
FROM
  trainingemp;

SELECT
  substr(ename,
      length(ename),
      1)
FROM
  trainingemp;

SELECT
  *
FROM
  trainingemp
OFFSET (
  SELECT
    COUNT(*)
  FROM
    trainingemp
) - 10 ROWS FETCH NEXT 10 ROWS ONLY;

set serveroutput on;
DECLARE
 TYPE e_list IS TABLE OF employees.first_name%TYPE INDEX BY PLS_INTEGER;
```

```
 emps e_list;
BEGIN
 FOR x IN 100 .. 110 LOOP
 begin
  SELECT first_name
  INTO   emps(x)
  FROM   employees
  WHERE  employee_id = x ;
  EXCEPTION
  when no_data_found then
   dbms_output.put_line('emps(i)');
  end;
 END LOOP;
 FOR i IN emps.first()..emps.last() LOOP
  dbms_output.put_line(emps(i));
 END LOOP;
END;
/
DECLARE
 TYPE e_list IS TABLE OF employees.first_name%TYPE INDEX BY PLS_INTEGER;
 emps e_list;
BEGIN
 FOR x IN 1 .. 110 LOOP
  BEGIN
   SELECT first_name
   INTO   emps(x)
   FROM   employees
   WHERE  employee_id = x;
  EXCEPTION
   WHEN NO_DATA_FOUND THEN
     emps(x) := '';
  END;
 END LOOP;
 FOR i IN emps.FIRST .. emps.LAST LOOP
  IF emps.EXISTS(i) THEN
   DBMS_OUTPUT.PUT_LINE(emps(i));
  END IF;
 END LOOP;
END;
/
select * from employees;

select * from shubhamEmployee;
declare
emp_list employee2%rowtype;
--idx PLS_INTEGER:=1;
begin
```

```
for i in 7000..8000 loop
begin
select * into emp_list from employee2 where empno=1;
dbms_output.put_line(emp_list.ename);
EXCEPTION
WHEN NO_DATA_FOUND THEN
dbms_output.put_line('emp_list.ename');
end;
end loop;
end;
/
set serveroutput on;
-----------------------------------------% Atribute-----------------------------------------------------
------------

DECLARE
  v_name trainingemp.ename%TYPE;
BEGIN
  SELECT
    ename
  INTO v_name
  FROM
    trainingemp
  WHERE
    empno = 7521;

  dbms_output.put_line(v_name);
END;

variable g_name varchar(255);

SELECT
  ename
INTO :g_name
FROM
  trainingemp
WHERE
  empno = 7369;

PRINT g_name;

SELECT
  *
FROM
  shubhamlogin;
```

```
 ------------------------------------Using Local Variable-----------------------------------------
----------


DECLARE
   v_principle NUMBER := 20000;
   v_n       NUMBER := 2;
   v_r       NUMBER := 7;
   v_amount    NUMBER;
BEGIN
   v_amount := v_principle * ( ( ( 1 + v_r ) * * v_n - 1 ) / v_r );

   dbms_output.put_line((v_amount));
END;

VARIABLE amount number;

DECLARE
   v_principle NUMBER := &v_principle;
   v_n       NUMBER := &v_n;
   v_r       NUMBER := &v_r;
   v_r1       NUMBER;
BEGIN
   v_r1 := ( ( 1 + v_r ) * * v_n );
--v_r1:=power(1+v_r,v_n);
   :amount := ceil(v_principle *(v_r * v_r1) /(v_r1 - 1));

   dbms_output.put_line(ceil(:amount));
END;
/

PRINT amount;

 ------------------------------------------------------bind-----------------------------------------

VARIABLE v_amount NUMBER;

DECLARE
   v_principle NUMBER := 20000;
   v_n       NUMBER := 2;
   v_r       NUMBER := 7 / 100; -- Assuming 7% interest, convert to decimal
   v_rl      NUMBER;
BEGIN
   v_rl := power(1 + v_r, v_n);
```

```
   :v_amount := v_principle * ( v_r * v_rl ) / ( v_rl - 1 );
   dbms_output.put_line(:v_amount);
END;
/

PRINT v_amount;
 ------------------------------------------------


 variable v_a refcursor;

BEGIN
   OPEN :v_a FOR SELECT
                *
             FROM
               trainingemp;

END;
/

print v_a;
 ------------------------------------------------Counting-----------------------

DECLARE
   v_countion NUMBER := &v_counting;
BEGIN
   dbms_output.put_line(v_countion * 1);
   dbms_output.put_line(v_countion * 2);
   dbms_output.put_line(v_countion * 3);
   dbms_output.put_line(v_countion * 4);
   dbms_output.put_line(v_countion * 5);
   dbms_output.put_line(v_countion * 6);
   dbms_output.put_line(v_countion * 7);
   dbms_output.put_line(v_countion * 8);
   dbms_output.put_line(v_countion * 9);
   dbms_output.put_line(v_countion * 10);
END;

 ---------------------------------------insert----------------------------------------
-- begin
-- insert into shubham_user (teamid) values(1);
--
---- commit
-- end;

 ----------------------------update-----------------------------------------------------
BEGIN
```

```
  UPDATE shubham_user
  SET
    teamid = 3
  WHERE
    userid = 2;
 --commit
  ROLLBACK;
END;

SELECT
  *
FROM
  shubham_user;
-------------------------Delete-------------------------------------------------------

BEGIN
  DELETE shubham_user
  WHERE
    userid = 1;
--rollback;
END;
 ----------------------------counting using for loop----------------------------------------

DECLARE
  v_counting NUMBER := &v_counting;
  v_i      NUMBER := 1;
BEGIN
  FOR v_i IN 1..10 LOOP
    dbms_output.put_line(v_counting * v_i);
  END LOOP;
END;

 ----------------------------counting using while loop ----------------------------------------

DECLARE
  v_c NUMBER := &v_c;
  v_i NUMBER;
BEGIN
  v_i := 1;
  WHILE v_i < 10 LOOP
    dbms_output.put_line(v_c * v_i);
    v_i := v_i + 1;
  END LOOP;

END;
----------------------Tuesday Control Statement  ------------------------------------------------
--------------------------------
```

```
set serveroutput on;

DECLARE
  v_salary      NUMBER := &v_salary;
  v_year_service NUMBER := &v_year_service;
  v_bonus       NUMBER;
BEGIN
  IF
    v_salary < 30000
    AND v_year_service >= 10
  THEN
    v_bonus := 15;
  ELSIF
    v_salary BETWEEN 30000 AND 50000
    AND v_year_service BETWEEN 5 AND 10
  THEN
    v_bonus := 10;
  ELSIF
    v_salary > 50000
    AND v_year_service < 5
  THEN
    v_bonus := 5;
  ELSE
    dbms_output.put_line('No bonus');
  END IF;

  dbms_output.put_line('bonus is '
                || v_bonus
                || ' % of salary');
END;
```

```
------------------------Fibonacci Series---------------------------
DECLARE
  v_start  NUMBER;
  v_end    NUMBER;
  v_sum    NUMBER;
  v_number NUMBER;
BEGIN
  v_start := 0;
  v_end := 1;
  v_number := 1;
  LOOP
    v_sum := v_start + v_end;
    dbms_output.put_line(v_start);
```

```
    v_start := v_end;
    v_end := v_sum;
    v_number := v_number + 1;
    EXIT WHEN v_number > 5;
  END LOOP;

END;



------------------Fibonacci Series using for loop--------------
DECLARE
  v_start  NUMBER;
  v_end    NUMBER;
  v_sum    NUMBER;
  v_number NUMBER;
BEGIN
  v_start := 0;
  v_end := 1;
  FOR v_number IN 1..5 LOOP
    v_sum := v_start + v_end;
    dbms_output.put_line(v_start);
    v_start := v_end;
    v_end := v_sum;
  END LOOP;

END;



-----------------total Salary in in a single variable ----------------------

DECLARE
  v_count      NUMBER;
  v_total_sal  NUMBER := 0;
  v_countre    NUMBER := 1;
  v_current_sal trainingemp.sal%TYPE;
BEGIN
  SELECT
    COUNT(*)
  INTO v_count
  FROM
    trainingemp;

  WHILE v_countre <= v_count LOOP
    SELECT
      sal
```

```
    INTO v_current_sal
    FROM
      (
        SELECT
          sal,
          ROW_NUMBER()
          OVER(
            ORDER BY
              empno
          ) AS rw
        FROM
          trainingemp
      )
    WHERE
      rw = v_countre;

    v_total_sal := v_total_sal + v_current_sal;
    v_countre := v_countre + 1;
  END LOOP;

  dbms_output.put_line(v_total_sal);
END;

----------for Loop----------

DECLARE
  v_count      NUMBER;
  v_total_sal  NUMBER := 0;
  v_countre    NUMBER;
  v_current_sal trainingemp.sal%TYPE;
BEGIN
  SELECT
    COUNT(*)
  INTO v_count
  FROM
    trainingemp;

  FOR v_countre IN 1..v_count LOOP
    SELECT
      sal
    INTO v_current_sal
    FROM
      (
        SELECT
          sal,
          ROW_NUMBER()
```

```
            OVER(
               ORDER BY
                  empno
            ) AS rn
          FROM
            trainingemp
        )
      WHERE
        rn = v_countre;

      v_total_sal := v_total_sal + v_current_sal;
   END LOOP;

   dbms_output.put_line(v_total_sal);
END;
```

----------Dynamic value ----------

```
DECLARE
   v_char  CHAR(1) := upper('&v_char');
   v_grade VARCHAR(10);
BEGIN
   CASE
      WHEN v_char = 'A' THEN
         v_grade := 'Excellence';
      WHEN v_char = 'B' THEN
         v_grade := 'Very Good';
      WHEN v_char = 'C' THEN
         v_grade := 'fair';
      ELSE
         v_grade := 'fail';
   END CASE;

   dbms_output.put_line(v_grade);
END;
```

----------What is the wrong usage for this code ->default value must in single quote not in double quote  ----------

```
DECLARE
   v_temp VARCHAR2(50) NOT NULL DEFAULT 'TEMP';
BEGIN
   dbms_output.put_line(v_temp);
END;
```

```
--What will be the output of this code   1->15 and 2->40
DECLARE
  v_num NUMBER := 20;
BEGIN
  v_num := 40;
  DECLARE
    v_num NUMBER := 15;
  BEGIN
    dbms_output.put_line('1 -> ' || v_num);
    v_num := 30;
  END;

  dbms_output.put_line('2 -> ' || v_num);
END;
/
--Give an example using %type attribute and Bind variable
variable annual_sal number;
declare
v_ename  trainingemp.ename%type;
v_sal  trainingemp.sal%type;
begin
select ename,sal,sal*12 into v_ename,v_sal,:annual_sal from trainingemp where
empno =7369;
 dbms_output.put_line('v_ename ->' || v_ename);
 dbms_output.put_line('v_sal ->' || v_sal);
  dbms_output.put_line('annual_sal -> ' || :annual_sal );
end;


--.If you execute dbms_output.put_line and you cannot see the output, which one can
be the reason -> not execute a set serveroutput on;

--Write a PL/SQL block that evaluates a student's score and assigns a grade based on
the following conditions:
--Score ≥ 90: Grade A
--Score ≥ 80: Grade B
--Score ≥ 70: Grade C
--Score ≥ 60: Grade D
--Score < 60: Fail
--, use if then elseif else blocks

declare
v_grade VARCHAR(20);
v_score number:=&v_score;
begin
if v_score>=90 then
```

```
v_grade:='A';
elsif v_score >=80 then
v_grade:='b';
elsif v_score >=70 then
v_grade:='c';
elsif v_score >=60 then
v_grade:='d';
else
V_GRADE:='Fail';
END IF;
dbms_output.put_line('Grade -> ' || v_grade );
end;
/
```

```
--
--Write a PL/SQL block to calculate an employee's bonus based on their salary using if then else block:
--Salary > 10,000: Bonus is 20% of salary
--Salary between 5,000 and 10,000: Bonus is 10% of salary
--Salary < 5,000: Bonus is 5% of salary


declare
v_salary number:=&v_salary;
v_bonus number;
begin
if v_salary >10000 then
v_bonus:=20;
elsif v_salary between 5000 and 10000 then
v_bonus:=10;
else
v_bonus:=5;
end if;
dbms_output.put_line('Bobus is  ' || v_bonus || ' % of salary' );
end;
```

```
--Write a PL/SQL block to determine whether a given year is a leap year.

declare
v_year number:=&v_year;
begin
```

```
if mod(v_year,4)=0 and (mod(v_year,400)=0 or mod(v_year,100)!=0) then
dbms_output.put_line('Leap year');
else
dbms_output.put_line('Not a Leap year');
end if;
end;
```

--Write a PL/SQL block to display the description of an employee's job role based on their job code:
--MGR: Manager
--DEV: Developer
--TST: Tester
--HR: Human Resources
--Any other code: Unknown Role (use case expressions)

```
declare
v_role char(3):=upper('&v_role');---ex -> mgr,dev,tst,hr
v_fullrole varchar(20);
begin
case
when v_role='MGR' then
v_fullrole:='Manager';
when v_role='DEV' then
v_fullrole:='Developer';
when v_role='TST' then
v_fullrole:='Tester';
when v_role='HR' then
v_fullrole:='Human Resources';
else
v_fullrole:='Unknown Role';
end case;
dbms_output.put_line(v_role ||': ' || v_fullrole  );
end;
```

--. Write a PL/SQL block to calculate a salary increment percentage based on an employee's job role:
--MGR: 20% Increment
--DEV: 15% Increment
--TST: 10% Increment
--HR: 8% Increment
--Any other role: 5% Increment
--(Case statements)

```
declare
```

```
v_role char(3):=upper('&v_role');--     ex -> mgr,dev,tst,hr
v_increment number;
begin
case
when  v_role='MGR' then
v_increment:=20;
when v_role='DEV' then
v_increment:=15;
when v_role='TST' then
v_increment:=10;
when v_role='HR' then
v_increment:=8;
else
v_increment:=5;
end case;
dbms_output.put_line(v_role||': ' || v_increment || '% Increment' );
end;
/
----------------------------------
create or replace procedure shrst(in_empno in trainingemp.empno%type,in_percent in
number)
is
begin
update employee2 set
sal=sal+sal*in_percent/100
where empno=in_empno;
end shrst;
/
begin
shrst(7369,10);
end;
rollback;
select * from employee2;

set serveroutput on;
create or replace procedure shubham_exception(dept_id in number) is
emp_rec employee2%rowtype;
begin
for i in (select * from employee2 where deptno=dept_id) loop
--select * into emp_rec from employee2 where deptno=dept_id;
--dbms_output.put_line(emp_rec.ename||' '||emp_rec.sal);
dbms_output.put_line(i.ename||' '||i.sal);
end loop;
dbms_output.put_line('No Data Found');
EXCEPTION
when no_data_found then
dbms_output.put_line('No Data Found');
```

```
end shubham_exception;

/
begin
shubham_exception(1);
end;
/
select * from employee2;
/
set SERVEROUTPUT on;
declare
dept number:=&dept;
emp_rec employee2.ename%type;
begin
select ename into emp_rec from employee2 where deptno=dept;
EXCEPTION
when No_data_found then
dbms_output.put_line('No_data_found ');
when too_many_rows then
dbms_output.put_line('To Many Rows ');
when others then
dbms_output.put_line('Error ');

end;
```

## Function

```
desc employees;
set serveroutput on;

CREATE OR REPLACE FUNCTION totalcount RETURN NUMBER IS
  total NUMBER := 0;
BEGIN
  SELECT
    COUNT(*)
  INTO total
  FROM
    employee2;

  RETURN total;
END;
/

BEGIN
  dbms_output.put_line(totalcount());
```

```
END;
/
```

**--. Write a PL/SQL function that returns the name of the department with the highest total salary**

```
SELECT
  *
FROM
  employees;

CREATE OR REPLACE FUNCTION dname RETURN VARCHAR IS
  department_name VARCHAR(50);
  max_sal       employees.salary%TYPE;
BEGIN
  SELECT
    MAX(salary)
  INTO max_sal
  FROM
    employees;

  SELECT
    d.dept_name
  INTO department_name
  FROM
       employees e
    JOIN department d ON e.department_id = d.department_id
  WHERE
    e.salary >= max_sal;

  RETURN department_name;
END;
/

BEGIN
  dbms_output.put_line(dname());
END;
/
```

**--. Write a function to find the manager name for a given employee ID. If the employee has no manager, return 'No Manager'**

```
CREATE OR REPLACE FUNCTION mgrname (
  empno IN NUMBER
) RETURN VARCHAR IS
  total NUMBER := 0;
  ename VARCHAR(50) := 'NOT fOUND';
BEGIN
```

```
  SELECT
    COUNT(e.firstname)
  INTO total
  FROM
       shubham_user e
    JOIN shubham_project p ON p.managerid = e.userid
  WHERE
    e.userid = empno;

  IF total > 0 THEN
    SELECT
      e.firstname
    INTO ename
    FROM
         shubham_user e
      JOIN shubham_project p ON p.managerid = e.userid
    WHERE
      e.userid = empno
    GROUP BY
      e.firstname;

    RETURN ename;
  ELSE
    RETURN ename;
  END IF;

END;
/

BEGIN
  dbms_output.put_line(mgrname(5));
END;
/
```

**--Write a function to check if an employee belongs to a specific department.**
**--Return TRUE if the employee belongs to the department, else return FALSE.**

```
CREATE OR REPLACE FUNCTION is_employee_in_department (
  emp_id  IN NUMBER,
  dept_id IN VARCHAR
) RETURN BOOLEAN IS
  v_emp_dept_id employees.department_id%TYPE;
BEGIN
  SELECT
    department_id
  INTO v_emp_dept_id
  FROM
    employees
```

```
  WHERE
    employee_id = emp_id;
  IF v_emp_dept_id = dept_id THEN
    RETURN TRUE;
  ELSE
    RETURN FALSE;
  END IF;
EXCEPTION
  WHEN no_data_found THEN
    RETURN FALSE;
  WHEN OTHERS THEN
    RETURN FALSE;
END;
/

DECLARE
  emp_id  NUMBER;
  dept_id VARCHAR(20);
BEGIN
  IF is_employee_in_department(
    emp_id  => 21,
    dept_id => 'D1'
  ) THEN
    dbms_output.put_line('Department found');
  ELSE
    dbms_output.put_line('Department not found');
  END IF;
END;
/

SELECT
  *
FROM
  employees;

--. Which part of the following code is incorrect? missing return keyword
/
CREATE OR REPLACE FUNCTION get_avg_sal (
  p_dept_id IN NUMBER
) RETURN NUMBER AS
  v_avg_sal NUMBER;
BEGIN
  SELECT
    AVG(salary)
  INTO v_avg_sal
  FROM
    employees
```

```
  WHERE
    department_id = p_dept_id;

  RETURN v_avg_sal;
END get_avg_sal;
/
```

**--Write a function to find the average salary of employees reporting to a specific manager**

```
CREATE OR REPLACE FUNCTION avg_salary (
  mgrname IN VARCHAR2
) RETURN NUMBER IS
  average_sal NUMBER := 0;
BEGIN
  SELECT
    AVG(salary)
  INTO average_sal
  FROM
    employee2
  WHERE
   mgr_name=mgrname;
  RETURN average_sal;
END avg_salary;
/

BEGIN
  avg_salary();
END;
/

SELECT
  *
FROM
  employee2;
```

--**Write a function to return the total number of employees who were hired after a specific date.**

```
CREATE OR REPLACE FUNCTION hdcount (
  hdate IN DATE
) RETURN NUMBER IS
  total NUMBER := 0;
BEGIN
  SELECT
    COUNT(*)
  INTO total
```

```
  FROM
    employees
  WHERE
    hire_date > hdate;

  RETURN total;
END hdcount;
/

BEGIN
  dbms_output.put_line(hdcount('18-04-25'));
END;
/

--select ename from employee2 where empno in (select mgr from employee2);
set serveroutput on;
```
**--Write a function that returns the number of working days between two dates, excluding weekends.**

```
create or replace function shubham_date(s_date in date,end_date  in date) return
number is
curr_date date:=s_date;
w_date number:=0;
begin
while curr_date<end_date loop
if to_char(curr_date,'D') not in ('1','7') then --  1->sunday 7->saturday
w_date:=w_date+1;
end if;
curr_date:=curr_date+1;
end loop;
return w_date;
end;
/
begin
  dbms_output.put_line(shubham_date('02-04-25','30-4-25'));
end;

set serveroutput on;

declare
type emp_rec_all is record(
empno employee2.empno%type,
ename employee2.ename%type,
sal employee2.sal%type
);
emp_rec emp_rec_all;
begin
```

```
select empno,ename,sal into emp_rec from employee2 where empno=7369;
dbms_output.put_line(emp_rec.empno||'  '||emp_rec.ename||'  '||emp_rec.sal);
end;
/
```

---

```
DECLARE
emp_id employee2.empno%type:=&emp_id;
employee employee2%rowtype;
begin
select * into employee from employee2 where empno=emp_id ;
dbms_output.put_line('empno : '||employee.empno||' ,ename : '||employee.ename||'
,sal : '||employee.sal||' ,mgrno : '||employee.mgr);
end;
/
```

---

```
declare
type address_rec is record(
street varchar(50),
city varchar(50),
zip number
);
type emp_rec_all is record(
empno employee2.empno%type,
ename employee2.ename%type,
sal employee2.sal%type,
Address address_rec,
emp1 employee2%rowtype
);
emp emp_rec_all;
begin
emp.Address.street:='Dhawari';
emp.Address.city:='Satna';
emp.Address.zip:=485001;
select empno,ename,sal into emp.empno,emp.ename,emp.sal from employee2 where
empno=7369;
select * into emp.emp1 from employee2 where empno=7369;
--emp.ename:='Sharath';
--emp.empno:=7369;
--emp.sal:=20000;
dbms_output.put_line(emp.Address.street||' '||' '||emp.Address.city||'
'||emp.Address.zip||' '||emp.ename||' '||emp.empno||' '||emp.sal||' emp1 sal :
'||emp.emp1.sal);
end;
```