Functions:

- A standalone function is created using the **CREATE FUNCTION** statement. The simplified syntax for the **CREATE OR REPLACE PROCEDURE** statement is as follows –
- Syntax:
```
CREATE [OR REPLACE] FUNCTION function_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
RETURN return_datatype
{IS | AS}
BEGIN
  < function_body >
END [function_name];
```
- Calling a function:
```
Function_name();
```

Procedures:

- A procedure is created with the **CREATE OR REPLACE PROCEDURE** statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows –
- Syntax:
```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter_name [IN | OUT | IN OUT] type [, ...])]
{IS | AS}
BEGIN
 < procedure_body >
END procedure_name;
```
- The procedure can also be called from another PL/SQL block –

```
BEGIN
  Procedure_name
END;
/
```

IN

An IN parameter lets you pass a value to the subprogram. **It is a read-only parameter**. Inside the subprogram, an IN parameter acts like a constant. It cannot be assigned a value. You can pass a constant, literal, initialized variable, or expression as an IN parameter. You can also initialize it to a default value; however, in that case, it is omitted from the subprogram call. **It is the default mode of parameter passing. Parameters are passed by reference**

OUT

An OUT parameter returns a value to the calling program. Inside the subprogram, an OUT parameter acts like a variable. You can change its value and reference the value after assigning it. **The actual parameter must be variable and it is passed by value**.

IN OUT

An **IN OUT** parameter passes an initial value to a subprogram and returns an updated value to the caller. It can be assigned a value and the value can be read.

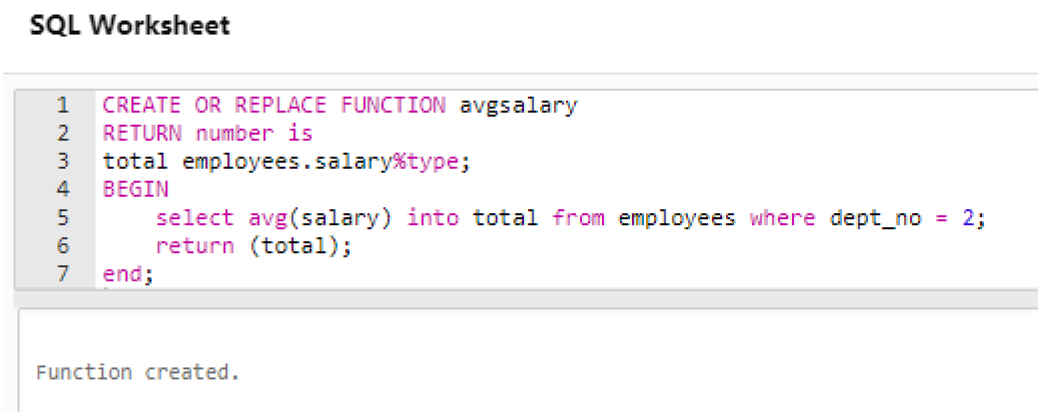
The actual parameter corresponding to an IN OUT formal parameter must be a variable, not a constant or an expression. Formal parameter must be assigned a value. **Actual parameter is passed by value.**

1. **Write a function to calculate average salary of a department and return average salary.**

Query:

```
CREATE OR REPLACE FUNCTION avgsalary

RETURN number is

total employees.salary%type;

BEGIN

   select avg(salary) into total from employees where dept_no = 2;

   return (total);

end;
```

Output:



SQL Worksheet

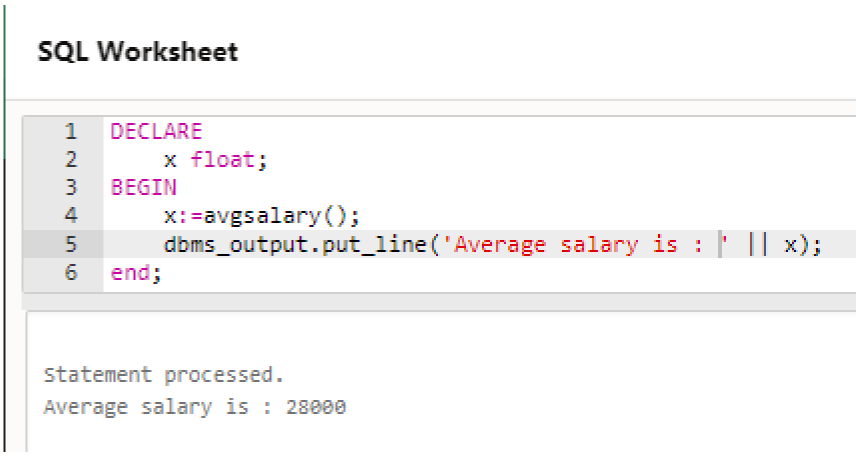
```
CREATE OR REPLACE FUNCTION avgsalary
RETURN number is
total employees.salary%type;
BEGIN
    select avg(salary) into total from employees where dept_no = 2;
    return (total);
end;
```

```
Function created.
```

Query:

```
DECLARE

   x float;

BEGIN

   x:=avgsalary();

   dbms_output.put_line('Average salary is : ' || x);

end;
```

Output:



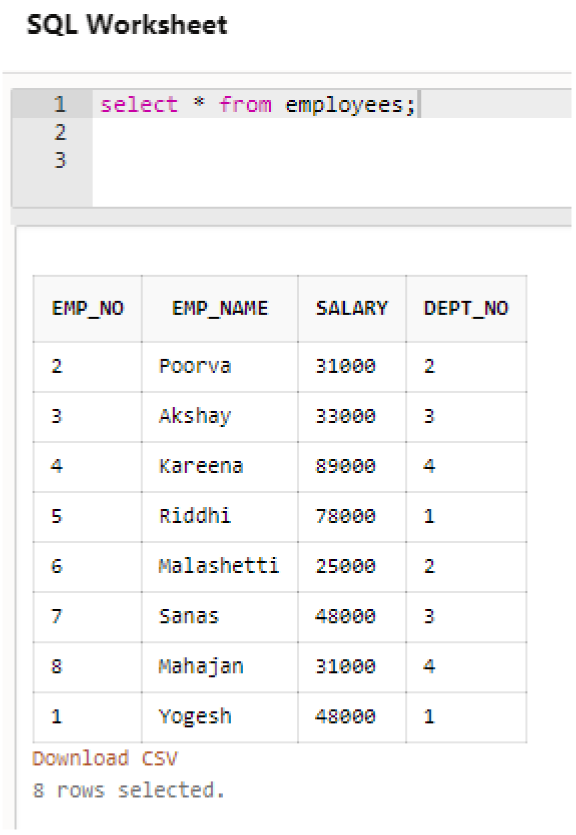
SQL Worksheet

```
DECLARE
    x float;
BEGIN
    x:=avgsalary();
    dbms_output.put_line('Average salary is : ' || x);
end;
```

```
Statement processed.
Average salary is : 28000
```

2. Write a procedure to delete an employee record where employee number is a parameter to a Procedure.

select * from employees;



SQL Worksheet

```
select * from employees;
```

EMP_NO	EMP_NAME	SALARY	DEPT_NO
2	Poorva	31000	2
3	Akshay	33000	3
4	Kareena	89000	4
5	Riddhi	78000	1
6	Malashetti	25000	2
7	Sanas	48000	3
8	Mahajan	31000	4
1	Yogesh	48000	1

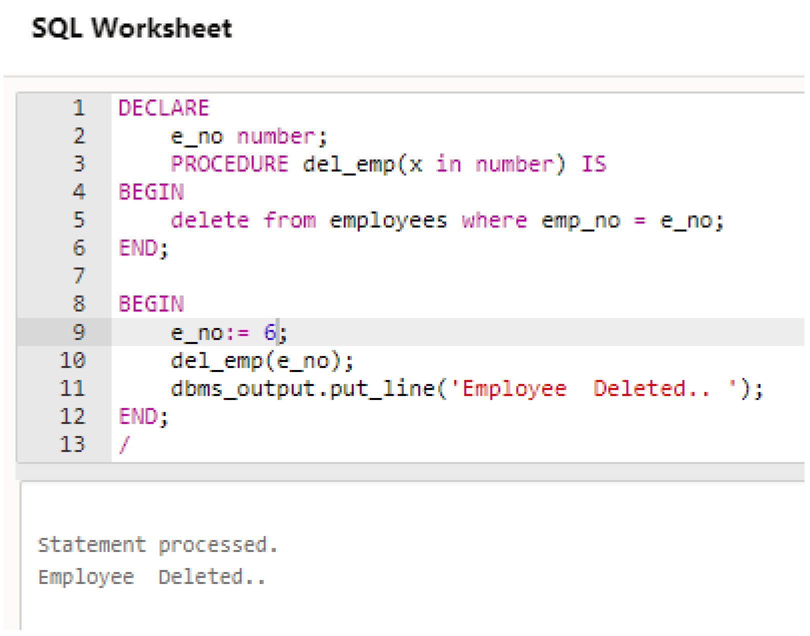
```
Download CSV
8 rows selected.
```

<u>Query</u>:

DECLARE

```
   e_no number;
   PROCEDURE del_emp(x in number) IS
BEGIN
   delete from employees where emp_no = e_no;
END;

BEGIN
   e_no:= 6;
   del_emp(e_no);
   dbms_output.put_line('Employee  Deleted.. ');
END;
/
```

Output:

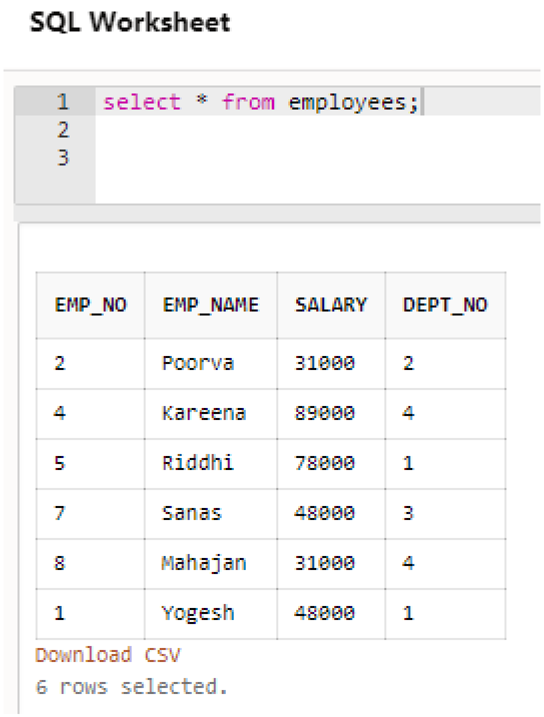


SQL Worksheet

```
DECLARE
    e_no number;
    PROCEDURE del_emp(x in number) IS
BEGIN
    delete from employees where emp_no = e_no;
END;

BEGIN
    e_no:= 6;
    del_emp(e_no);
    dbms_output.put_line('Employee  Deleted.. ');
END;
/
```

```
Statement processed.
Employee  Deleted..
```

select * from employees;



SQL Worksheet

```
select * from employees;
```

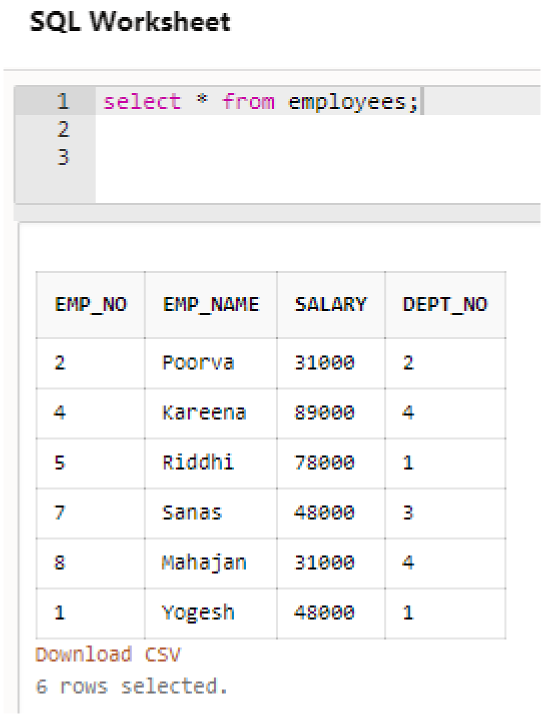
EMP_NO	EMP_NAME	SALARY	DEPT_NO
2	Poorva	31000	2
4	Kareena	89000	4
5	Riddhi	78000	1
7	Sanas	48000	3
8	Mahajan	31000	4
1	Yogesh	48000	1

Download CSV

6 rows selected.

3. Write a procedure to get a salary of an employee. Employee name is passed as a parameter to the procedure.

select * from employees;



SQL Worksheet

```
select * from employees;
```

EMP_NO	EMP_NAME	SALARY	DEPT_NO
2	Poorva	31000	2
4	Kareena	89000	4
5	Riddhi	78000	1
7	Sanas	48000	3
8	Mahajan	31000	4
1	Yogesh	48000	1

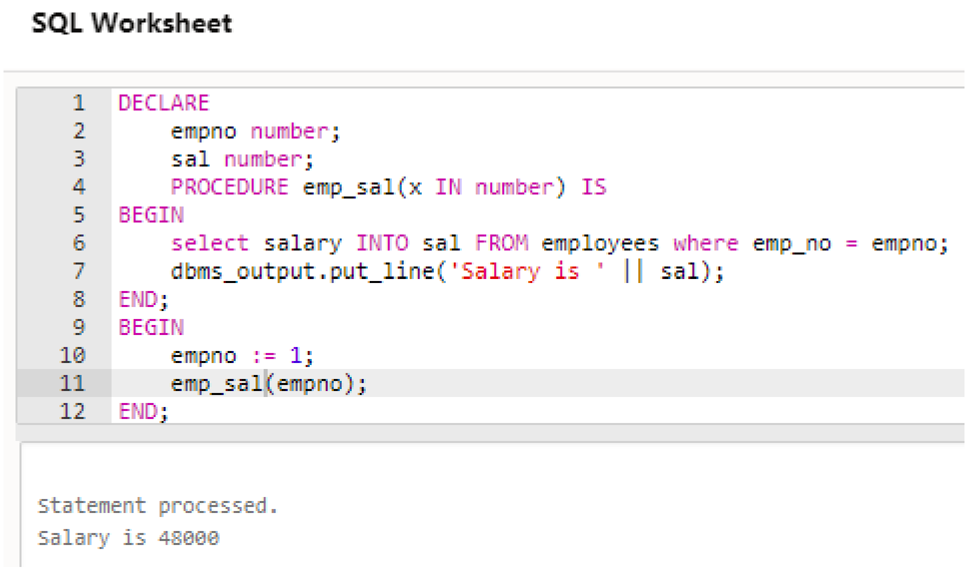
Download CSV

6 rows selected.

Query:

```
DECLARE
    empno number;
    sal number;
    PROCEDURE emp_sal(x IN number) IS
BEGIN
    select salary INTO sal FROM employees where emp_no = empno;
    dbms_output.put_line('Salary is ' || sal);
END;
BEGIN
    empno := 1;
    emp_sal(empno);
END;
```

Output:



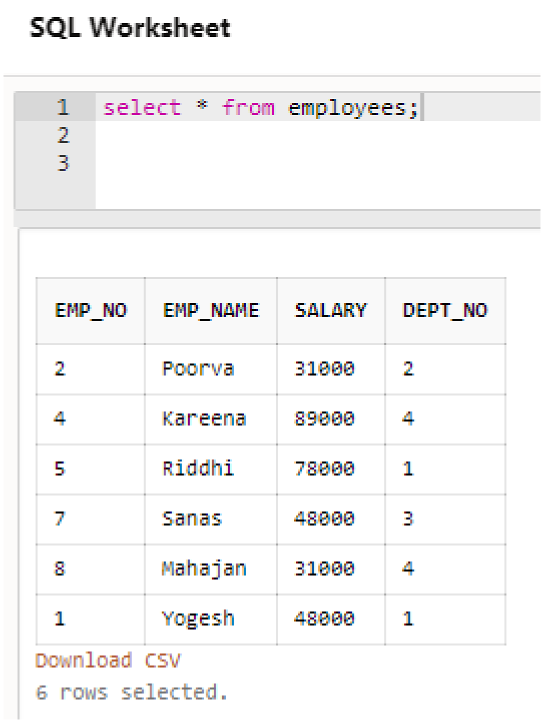
SQL Worksheet

```
DECLARE
    empno number;
    sal number;
    PROCEDURE emp_sal(x IN number) IS
BEGIN
    select salary INTO sal FROM employees where emp_no = empno;
    dbms_output.put_line('Salary is ' || sal);
END;
BEGIN
    empno := 1;
    emp_sal(empno);
END;
```

```
Statement processed.
Salary is 48000
```

4. Write a function to insert into emp table and throw an exception if corresponding dept_no is not present.

select * from employees;



SQL Worksheet

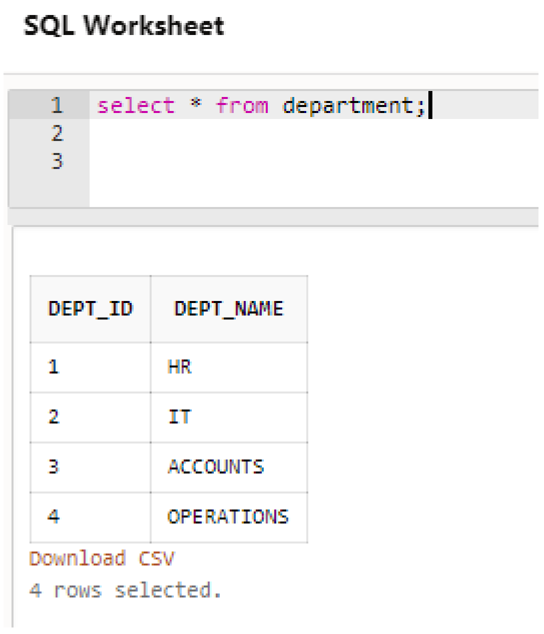
```
select * from employees;
```

EMP_NO	EMP_NAME	SALARY	DEPT_NO
2	Poorva	31000	2
4	Kareena	89000	4
5	Riddhi	78000	1
7	Sanas	48000	3
8	Mahajan	31000	4
1	Yogesh	48000	1

Download CSV

6 rows selected.

select * from department;



SQL Worksheet

```
select * from department;
```

DEPT_ID	DEPT_NAME
1	HR
2	IT
3	ACCOUNTS
4	OPERATIONS

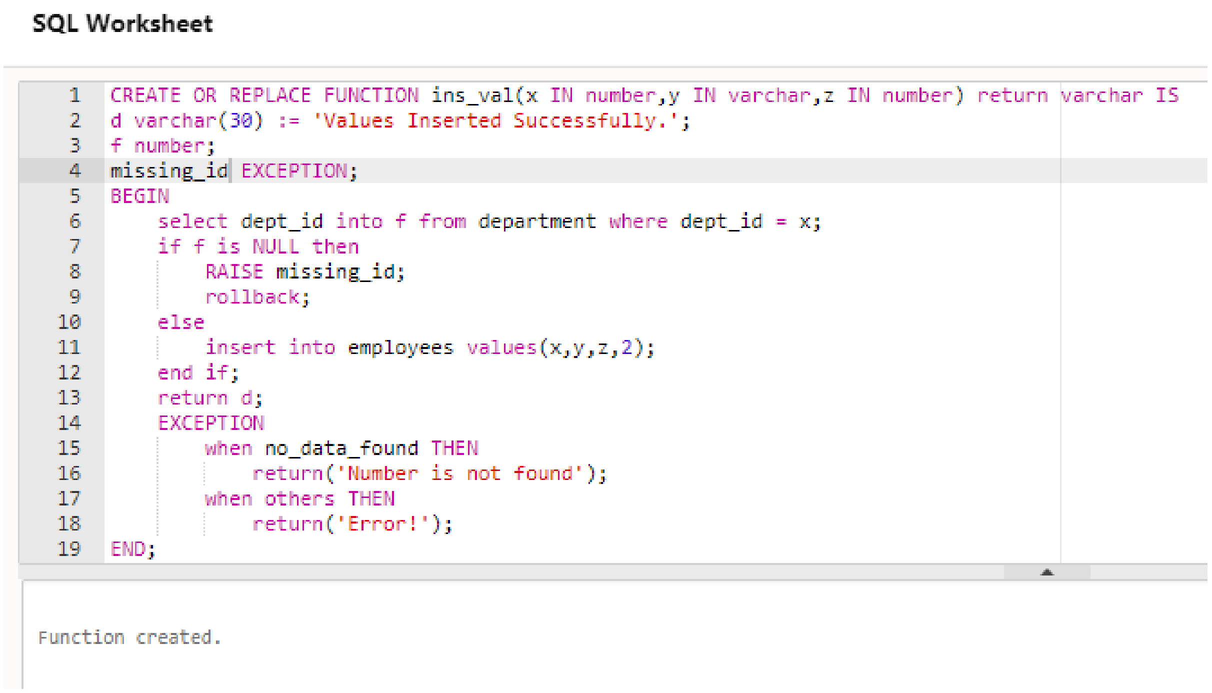
Download CSV

4 rows selected.

<u>Query</u>:

```
CREATE OR REPLACE FUNCTION ins_val(x IN number,y IN varchar,z IN number) return varchar IS

d varchar(30) := 'Values Inserted Successfully.';

f number;

missing_id EXCEPTION;

BEGIN

   select dept_id into f from department where dept_id = x;

   if f is NULL then

      RAISE missing_id;

      rollback;

   else

      insert into employees values(x,y,z,2);

   end if;

   return d;

   EXCEPTION

      when no_data_found THEN

         return('Number is not found');

      when others THEN

         return('Error!');

END;
```

<u>Output</u>:


SQL Worksheet

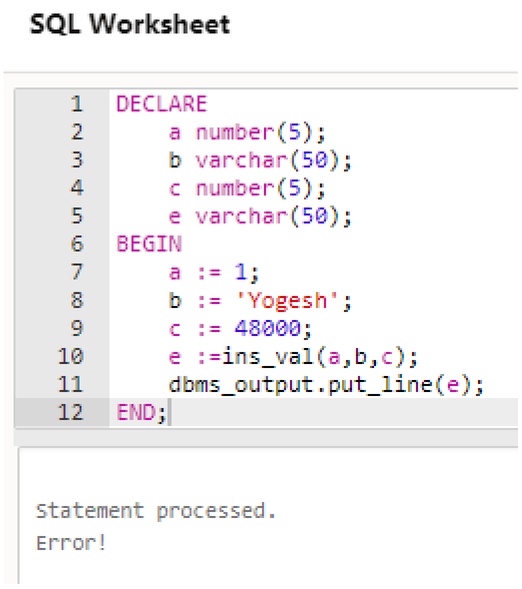
```
CREATE OR REPLACE FUNCTION ins_val(x IN number,y IN varchar,z IN number) return varchar IS
d varchar(30) := 'Values Inserted Successfully.';
f number;
missing_id EXCEPTION;
BEGIN
    select dept_id into f from department where dept_id = x;
    if f is NULL then
        RAISE missing_id;
        rollback;
    else
        insert into employees values(x,y,z,2);
    end if;
    return d;
    EXCEPTION
        when no_data_found THEN
            return('Number is not found');
        when others THEN
            return('Error!');
END;
```

```
Function created.
```

Query:

```
DECLARE
  a number(5);
  b varchar(50);
  c number(5);
  e varchar(50);
BEGIN
  a := 1;
  b := 'Yogesh';
  c := 48000;
  e :=ins_val(a,b,c);
  dbms_output.put_line(e);
END;
```

Output:



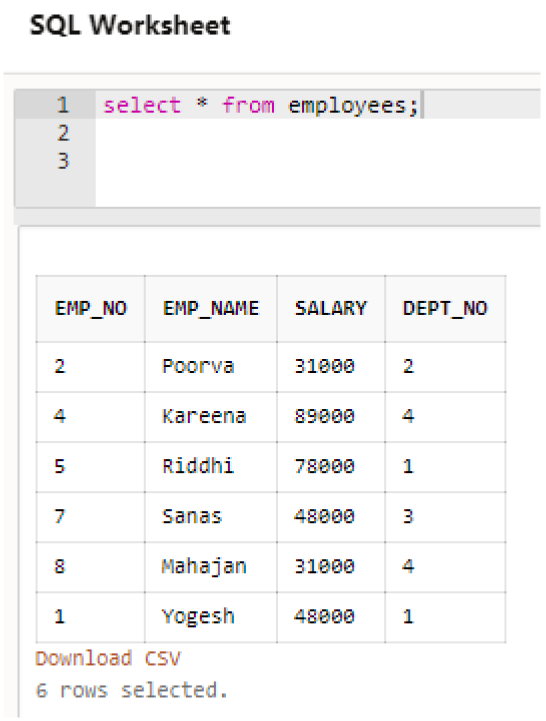
SQL Worksheet

```
DECLARE
    a number(5);
    b varchar(50);
    c number(5);
    e varchar(50);
BEGIN
    a := 1;
    b := 'Yogesh';
    c := 48000;
    e :=ins_val(a,b,c);
    dbms_output.put_line(e);
END;
```

```
Statement processed.
Error!
```

5.Create a procedure to raise a salary with parameters empno and raise as parameter. If salary is missing raise an exception "Missing salary". If record is not present then raise "No Data Found" exception, otherwise increase the salary of an employee by specified value.

select * from employees;



SQL Worksheet

```
select * from employees;
```

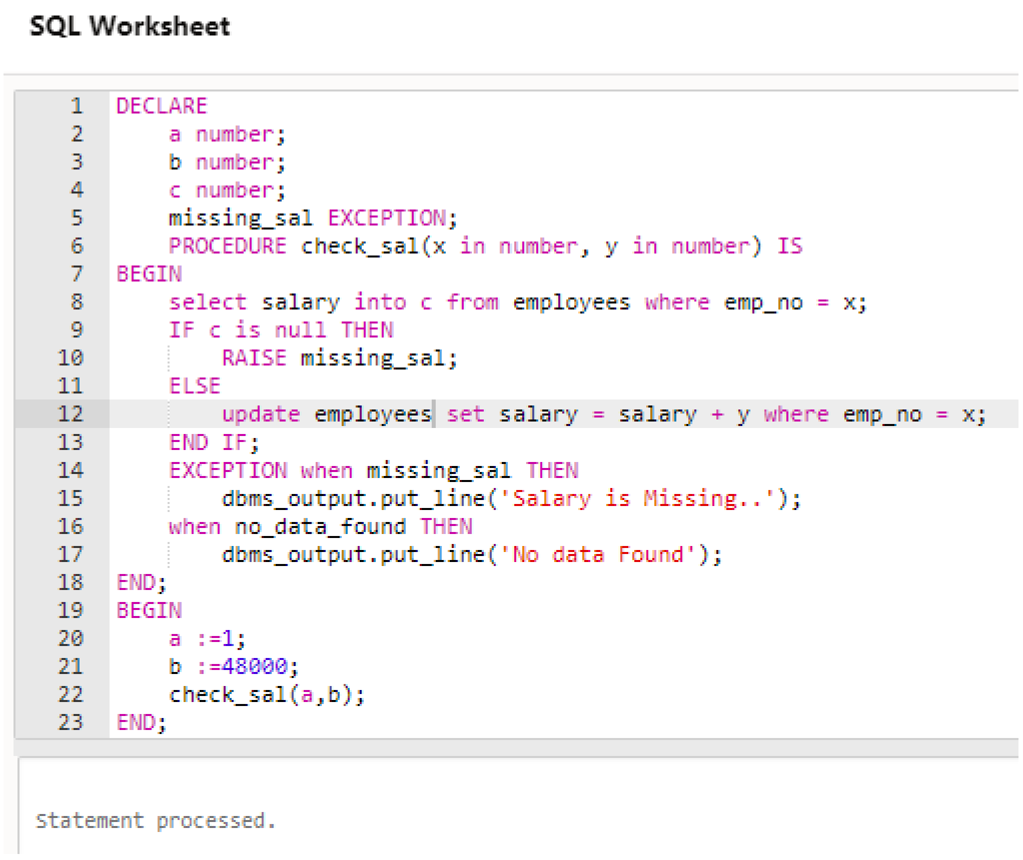
EMP_NO	EMP_NAME	SALARY	DEPT_NO
2	Poorva	31000	2
4	Kareena	89000	4
5	Riddhi	78000	1
7	Sanas	48000	3
8	Mahajan	31000	4
1	Yogesh	48000	1

```
Download CSV
6 rows selected.
```

Query:

```
DECLARE
  a number;
  b number;
  c number;
  missing_sal EXCEPTION;
  PROCEDURE check_sal(x in number, y in number) IS
BEGIN
  select salary into c from employees where emp_no = x;
  IF c is null THEN
    RAISE missing_sal;
  ELSE
    update employees set salary = salary + y where emp_no = x;
    END IF;

    EXCEPTION when missing_sal THEN

        dbms_output.put_line('Salary is Missing..');

    when no_data_found THEN

        dbms_output.put_line('No data Found');

END;

BEGIN

    a :=1;

    b :=48000;

    check_sal(a,b);

END;
```

Output:

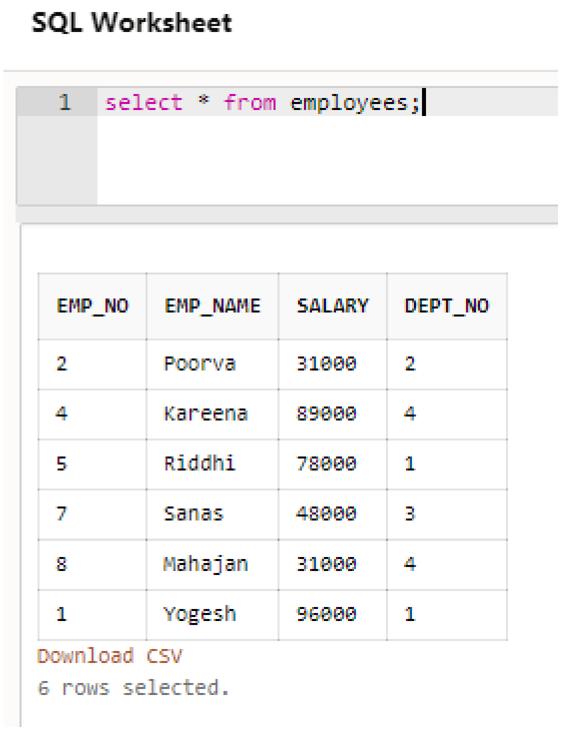


SQL Worksheet

```
DECLARE
    a number;
    b number;
    c number;
    missing_sal EXCEPTION;
    PROCEDURE check_sal(x in number, y in number) IS
BEGIN
    select salary into c from employees where emp_no = x;
    IF c is null THEN
        RAISE missing_sal;
    ELSE
        update employees set salary = salary + y where emp_no = x;
    END IF;
    EXCEPTION when missing_sal THEN
        dbms_output.put_line('Salary is Missing..');
    when no_data_found THEN
        dbms_output.put_line('No data Found');
END;
BEGIN
    a :=1;
    b :=48000;
    check_sal(a,b);
END;
```

```
Statement processed.
```

select * from employees;


SQL Worksheet

```
select * from employees;
```

EMP_NO	EMP_NAME	SALARY	DEPT_NO
2	Poorva	31000	2
4	Kareena	89000	4
5	Riddhi	78000	1
7	Sanas	48000	3
8	Mahajan	31000	4
1	Yogesh	96000	1

```
Download CSV
6 rows selected.
```

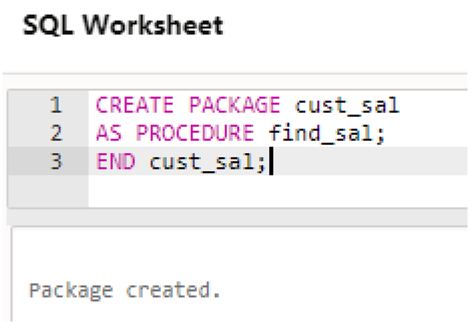
6.Create a package according to following criteria.

a) It should contain a cursor to give 10% raise to all employees earning less than 10000

Query:

```
CREATE PACKAGE cust_sal
AS PROCEDURE find_sal;
END cust_sal;
```

Output:



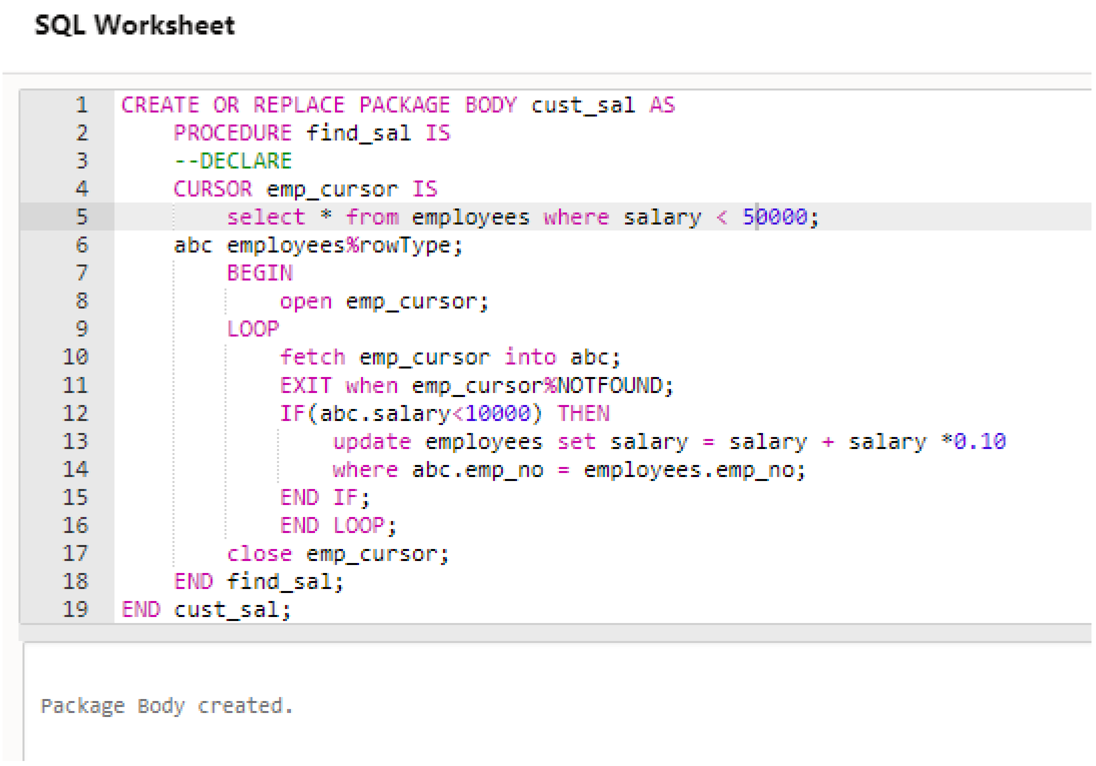
```
CREATE PACKAGE cust_sal
AS PROCEDURE find_sal;
END cust_sal;
```

```
Package created.
```

Query:

```
CREATE OR REPLACE PACKAGE BODY cust_sal AS
   PROCEDURE find_sal IS
   --DECLARE
   CURSOR emp_cursor IS
      select * from employees where salary < 50000;
   abc employees%rowType;
      BEGIN
         open emp_cursor;
      LOOP
         fetch emp_cursor into abc;
         EXIT when emp_cursor%NOTFOUND;
         IF(abc.salary<10000) THEN
            update employees set salary = salary + salary *0.10
            where abc.emp_no = employees.emp_no;
         END IF;
         END LOOP;
      close emp_cursor;
   END find_sal;
END cust_sal;
```

Output:



SQL Worksheet

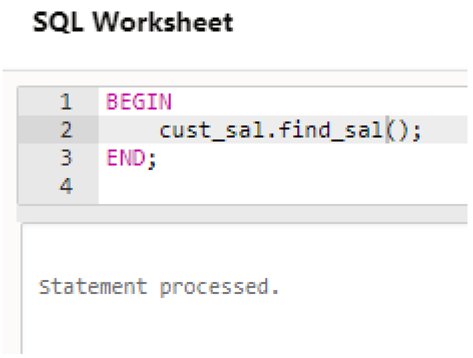
```
CREATE OR REPLACE PACKAGE BODY cust_sal AS
    PROCEDURE find_sal IS
    --DECLARE
    CURSOR emp_cursor IS
        select * from employees where salary < 50000;
    abc employees%rowType;
        BEGIN
            open emp_cursor;
        LOOP
            fetch emp_cursor into abc;
            EXIT when emp_cursor%NOTFOUND;
            IF(abc.salary<10000) THEN
                update employees set salary = salary + salary *0.10
                where abc.emp_no = employees.emp_no;
            END IF;
            END LOOP;
        close emp_cursor;
    END find_sal;
END cust_sal;
```

```
Package Body created.
```

Query:

```
BEGIN
   cust_sal.find_sal();
END;
```

Output:

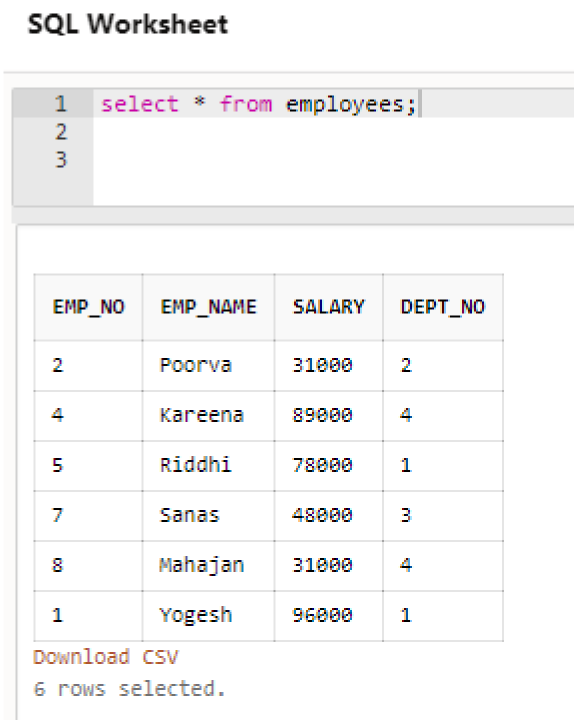


SQL Worksheet

```
BEGIN
    cust_sal.find_sal();
END;
```

```
Statement processed.
```

select * from employees;



SQL Worksheet

```
select * from employees;
```

EMP_NO	EMP_NAME	SALARY	DEPT_NO
2	Poorva	31000	2
4	Kareena	89000	4
5	Riddhi	78000	1
7	Sanas	48000	3
8	Mahajan	31000	4
1	Yogesh	96000	1

Download CSV

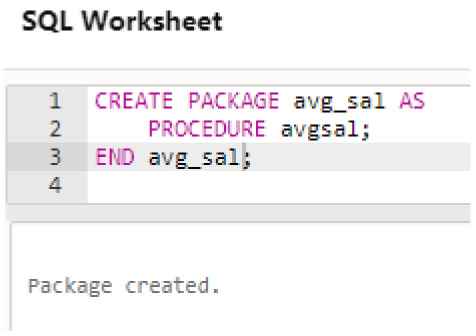
```
6 rows selected.
```

b) It should contain a function to calculate average salary of the department

Query:

```
CREATE PACKAGE avg_sal AS
   PROCEDURE avgsal;
END avg_sal;
```

Output:



SQL Worksheet

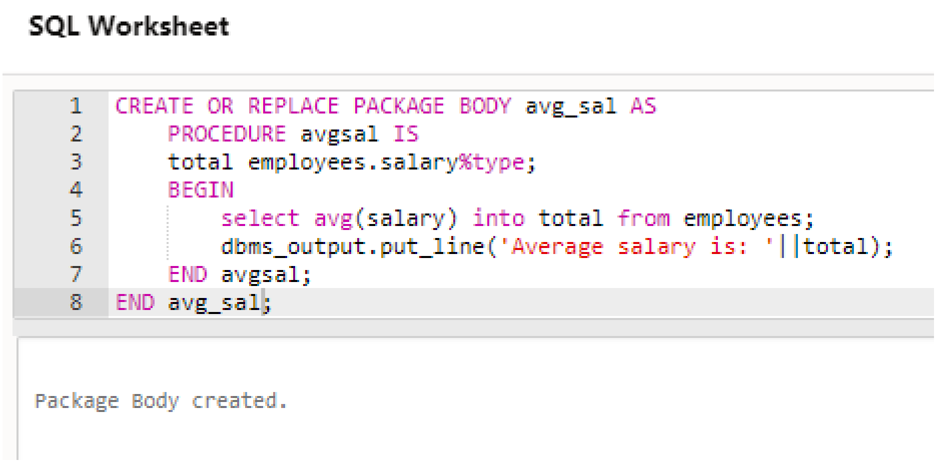
```
CREATE PACKAGE avg_sal AS
     PROCEDURE avgsal;
END avg_sal;
```

```
Package created.
```

Query:

```
CREATE OR REPLACE PACKAGE BODY avg_sal AS
    PROCEDURE avgsal IS
    total employees.salary%type;
    BEGIN
        select avg(salary) into total from employees;
        dbms_output.put_line('Average salary is: '||total);
    END avgsal;
END avg_sal;
```

Output:


SQL Worksheet

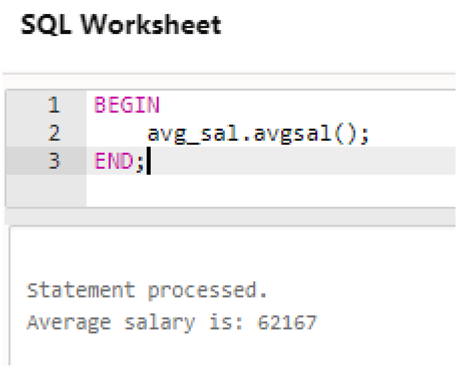
```
CREATE OR REPLACE PACKAGE BODY avg_sal AS
    PROCEDURE avgsal IS
    total employees.salary%type;
    BEGIN
        select avg(salary) into total from employees;
        dbms_output.put_line('Average salary is: '||total);
    END avgsal;
END avg_sal;
```

```
Package Body created.
```

Query:

```
BEGIN
    avg_sal.avgsal();
END;
```

Output:


SQL Worksheet

```
BEGIN
    avg_sal.avgsal();
END;
```

```
Statement processed.
Average salary is: 62167
```