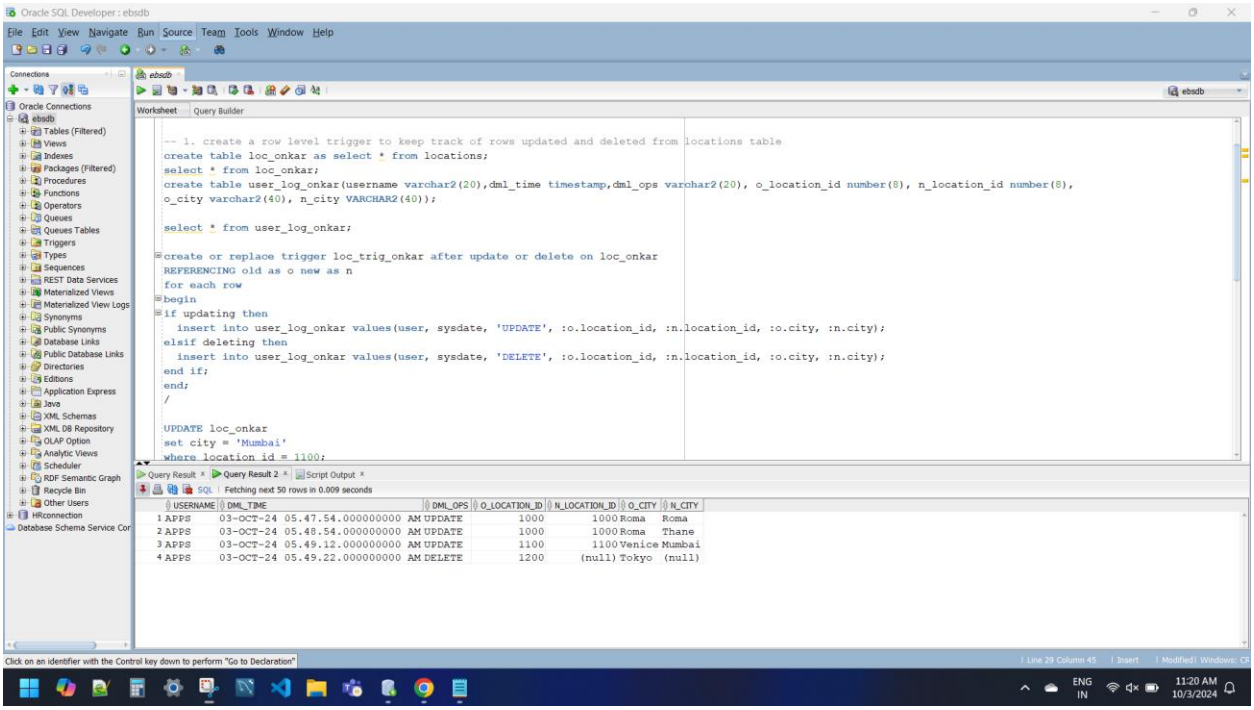
```
--PLSQL Assessment

set serveroutput on;


-- 1. create a row level trigger to keep track of rows updated and deleted from locations
table

create table loc_onkar as select * from locations;

select * from loc_onkar;

create table user_log_onkar(username varchar2(20),dml_time timestamp,dml_ops
varchar2(20), o_location_id number(8), n_location_id number(8),

o_city varchar2(40), n_city VARCHAR2(40));


select * from user_log_onkar;


create or replace trigger loc_trig_onkar after update or delete on loc_onkar

REFERENCING old as o new as n

for each row

begin

if updating then

  insert into user_log_onkar values(user, sysdate,
'UPDATE', :o.location_id, :n.location_id, :o.city, :n.city);

elsif deleting then

  insert into user_log_onkar values(user, sysdate,
'DELETE', :o.location_id, :n.location_id, :o.city, :n.city);

end if;

end;

/
```

UPDATE loc_onkar

set city = 'Mumbai'

where location_id = 1100;

delete from loc_onkar where location_id = 1200;



-- 2. write a plsql program to make sum of 10,20,30,40,50 using loop

declare

type num_nest is table of number(8,2);

l_num num_nest;

t_num number(8,2) := 0;

begin

l_num := num_nest(10,20,30,40,50);

for i in 1..l_num.count loop

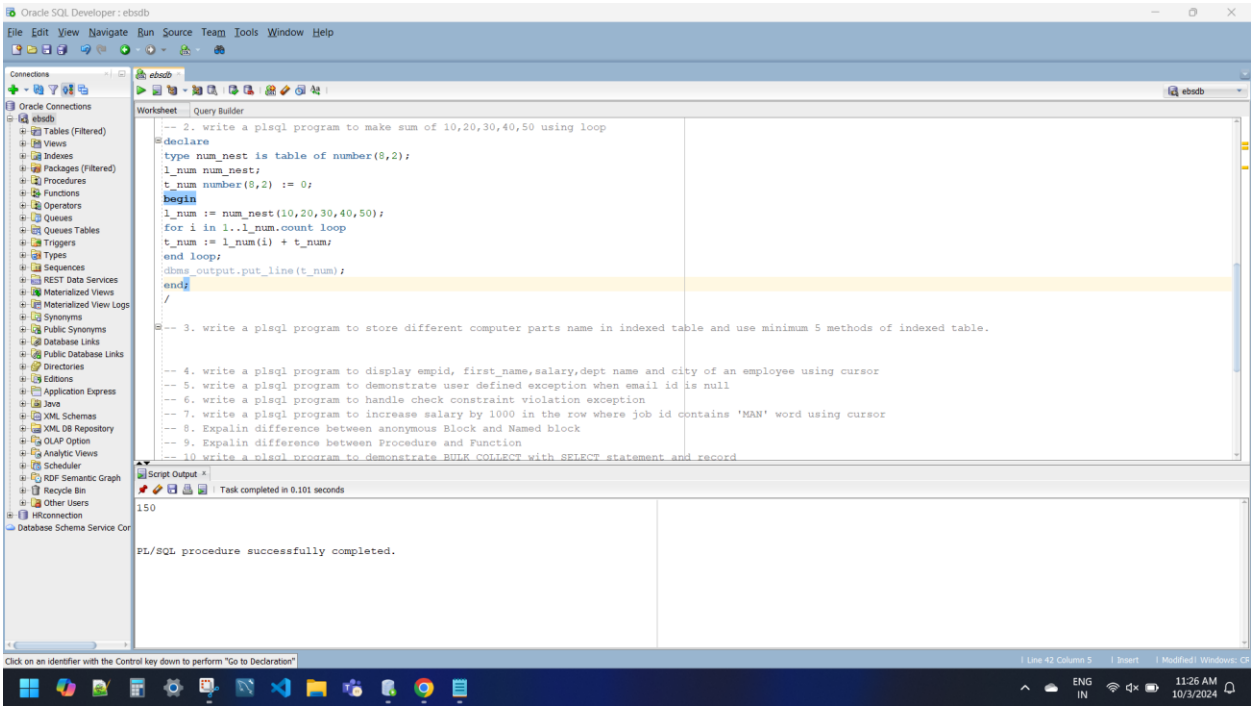
t_num := l_num(i) + t_num;

end loop;

dbms_output.put_line(t_num);

end;

/



-- 3. write a plsql program to store different computer parts name in indexed table and use minimum 5 methods of indexed table.

declare

type comp_parts_idx is table of varchar2(40)

index by PLS_INTEGER;

c_idx comp_parts_idx;

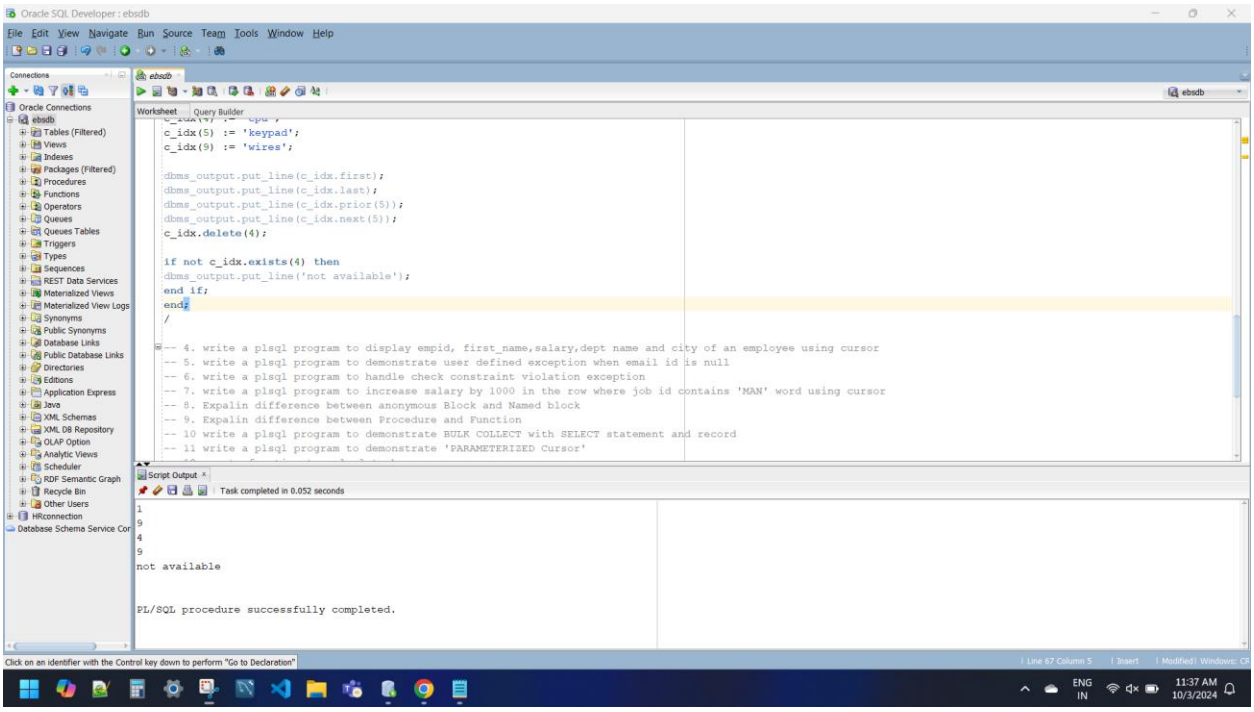
begin

c_idx(1) := 'mouse';

c_idx(2) := 'monitor';

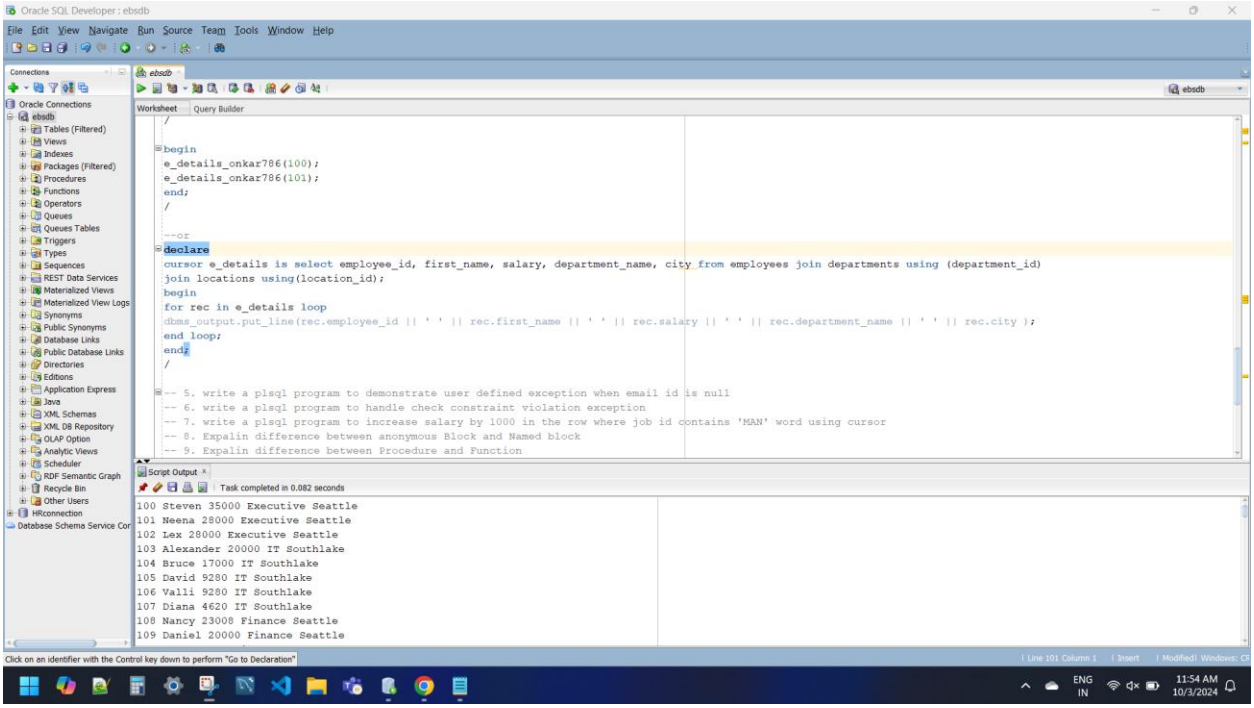
c_idx(3) := 'ram';

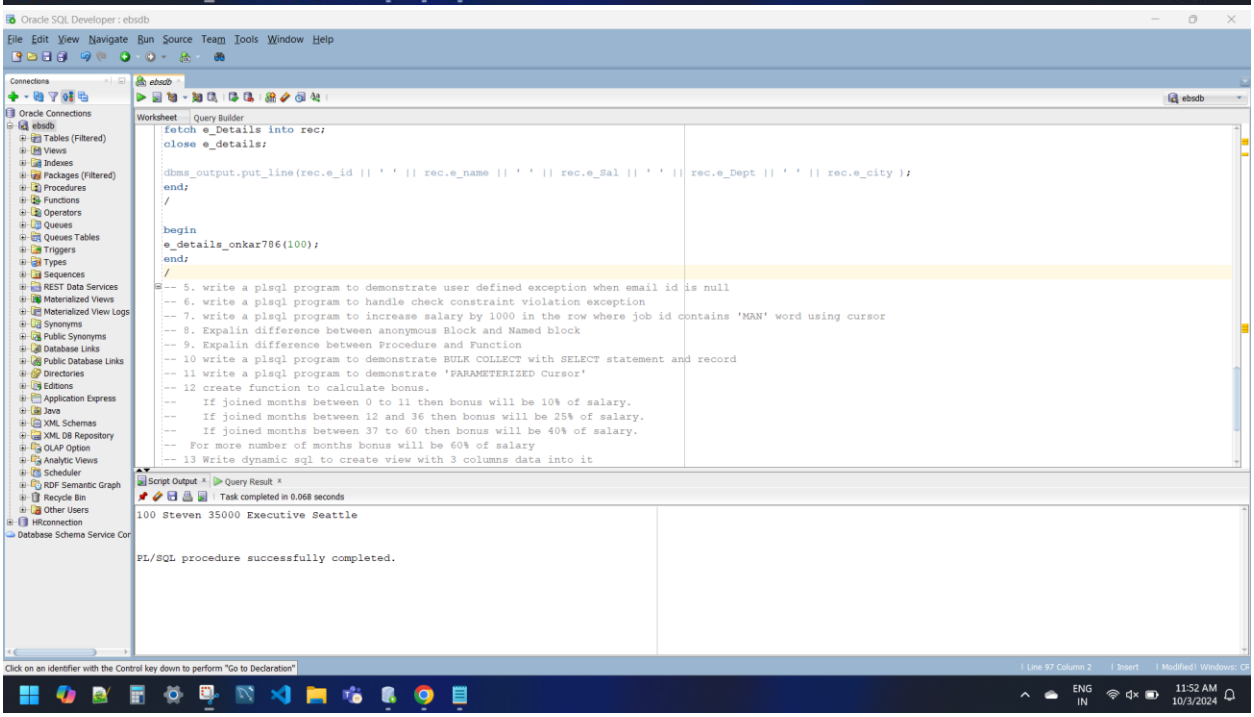
```
c_idx(4) := 'cpu';

c_idx(5) := 'keypad';

c_idx(9) := 'wires';


dbms_output.put_line(c_idx.first);

dbms_output.put_line(c_idx.last);

dbms_output.put_line(c_idx.prior(5));

dbms_output.put_line(c_idx.next(5));

c_idx.delete(4);


if not c_idx.exists(4) then

dbms_output.put_line('not available');

end if;

end;

/
```



```
-- 4. write a plsql program to display empid, first_name,salary,dept name and city of an employee using cursor

select * from employees;

select * from departments;

select * from locations;

create or replace procedure e_details_onkar786(e_id in number) is

cursor e_details is select employee_id, first_name, salary, department_name, city from employees join departments using (department_id)

join locations using(location_id) where employee_id = e_id;

type e_rec is record (

e_id employees.employee_id%type,

e_name employees.first_name%type,

e_Sal employees.salary%type,

e_Dept departments.department_name%type,

e_City locations.city%type

);

rec e_rec;

begin

open e_details;

fetch e_Details into rec;

close e_details;


dbms_output.put_line(rec.e_id || ' ' || rec.e_name || ' ' || rec.e_Sal || ' ' || rec.e_Dept || ' ' || rec.e_city );

end;

/
```

```
begin

e_details_onkar786(100);

e_details_onkar786(101);

end;

/


--or

declare

cursor e_details is select employee_id, first_name, salary, department_name, city from
employees join departments using (department_id)

join locations using(location_id);

begin

for rec in e_details loop

dbms_output.put_line(rec.employee_id || ' ' || rec.first_name || ' ' || rec.salary || ' ' ||
rec.department_name || ' ' || rec.city );

end loop;

end;

/
```





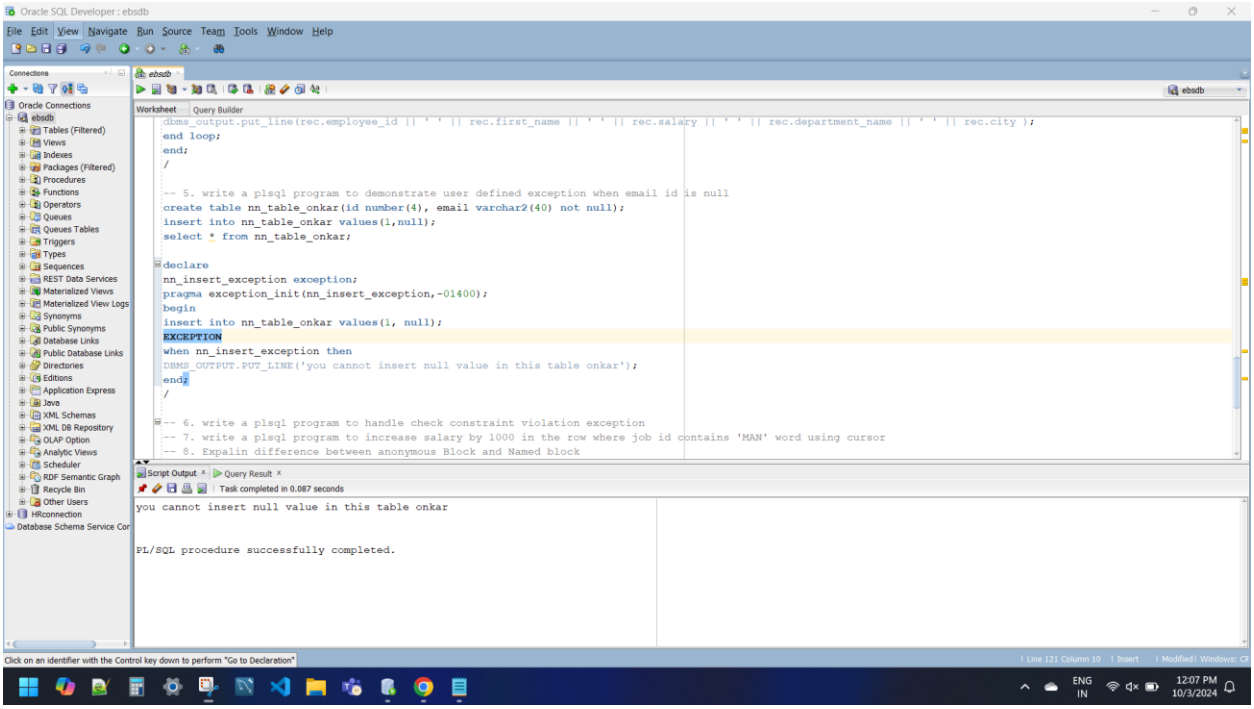
-- 5. write a plsql program to demonstrate user defined exception when email id is null
//////////////

create table nn_table_onkar(id number(4), email varchar2(40) not null);

insert into nn_table_onkar values(1,null);

select * from nn_table_onkar;

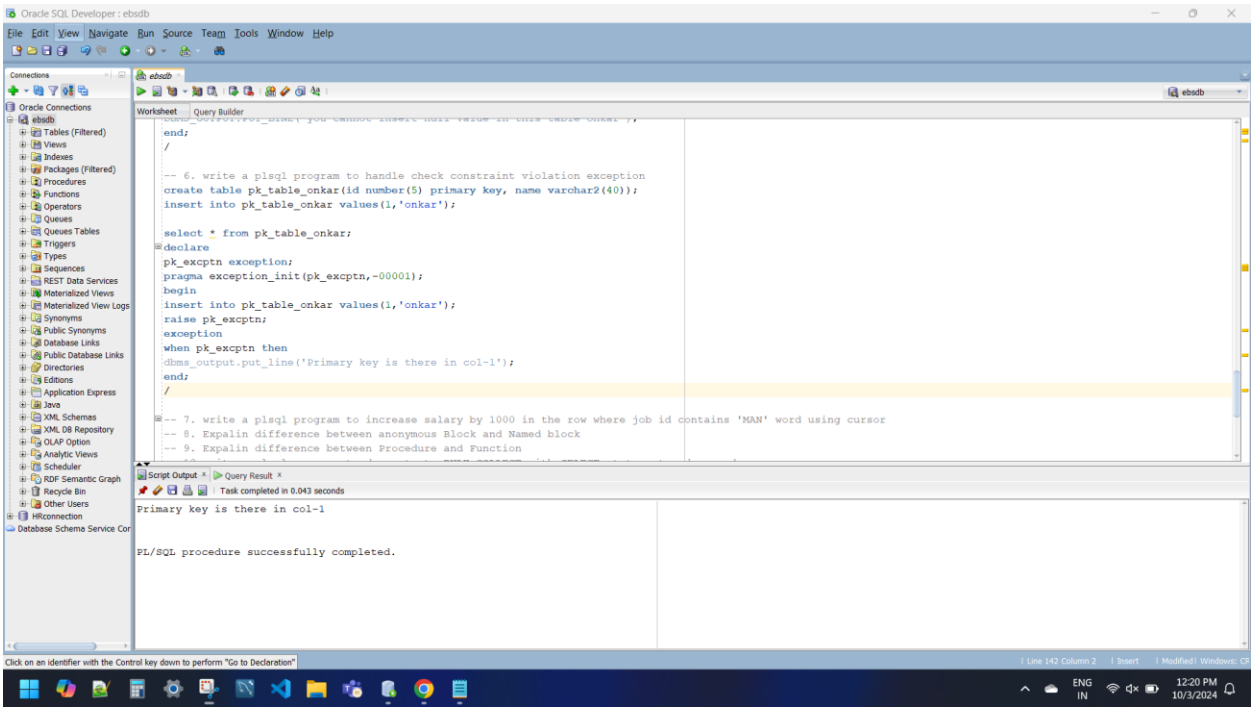
```
declare
nn_insert_exception exception;
pragma exception_init(nn_insert_exception,-01400);
begin
insert into nn_table_onkar values(1, null);
EXCEPTION
when nn_insert_exception then
DBMS_OUTPUT.PUT_LINE('you cannot insert null value in this table onkar');
end;
/
```



-- 6. write a plsql program to handle check constraint violation exception

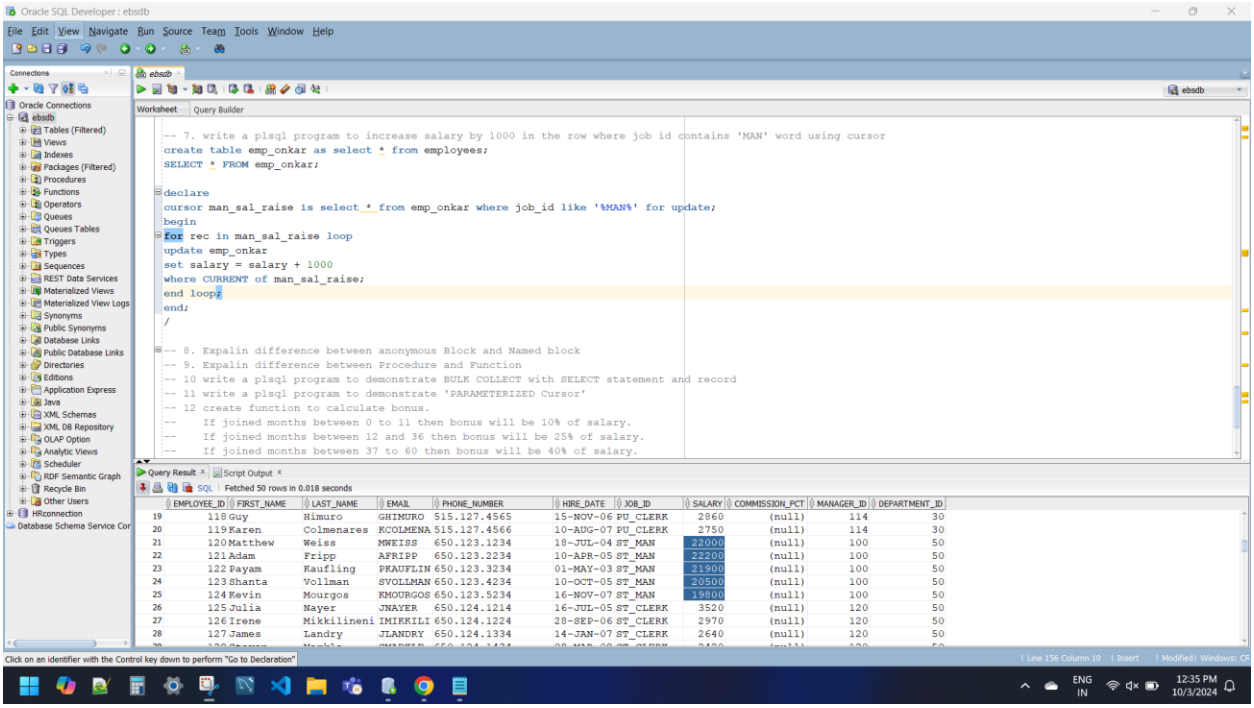
create table pk_table_onkar(id number(5) primary key, name varchar2(40));

```
insert into pk_table_onkar values(1,'onkar');


select * from pk_table_onkar;

declare

pk_excptn exception;

pragma exception_init(pk_excptn,-00001);

begin

insert into pk_table_onkar values(2,'onkar');

insert into pk_table_onkar values(1,'onkar');

raise pk_excptn;

exception

when pk_excptn then

dbms_output.put_line('Primary key is there in col-1');

end;

/
```



-- 7. write a plsql program to increase salary by 1000 in the row where job id contains 'MAN' word using cursor

```
create table emp_onkar as select * from employees;

SELECT * FROM emp_onkar;


declare

cursor man_sal_raise is select * from emp_onkar where job_id like '%MAN%' for update;

begin

for rec in man_sal_raise loop

update emp_onkar

set salary = salary + 1000

where CURRENT of man_sal_raise;

end loop;

end;

/
```



-- 8. Expalin difference between anonymous Block and Named block

--Anonymous Block:

--An anonymous block is a PL/SQL block without a name.

--It is generally used for temporary operations or operations that are not reused.

--It does not get stored in the database.

--It compiles again and again.

--Named Block:

--A named block is a PL/SQL block with a name, such as a procedure or a function.

--Named blocks are stored in the database and can be reused multiple times.

--Named blocks can accept parameters and return values.

--Named blocks can be invoked explicitly by their name.

--Compiles only once.

-- 9. Expalin difference between Procedure and Function

--Procedure:

--A procedure is a named PL/SQL block that performs a specific task or set of actions.

--It may or may not return a value.

--Procedures can have input/output parameters but don't return a value explicitly.

--Return keyword is not necessary to use.

--In header return keyword is not mention.

--Function:

--A function is a named PL/SQL block that must return value.

--It is typically used for calculations or operations that result in a specific value.

--Functions can also have input parameters, but they return a value using the RETURN keyword.

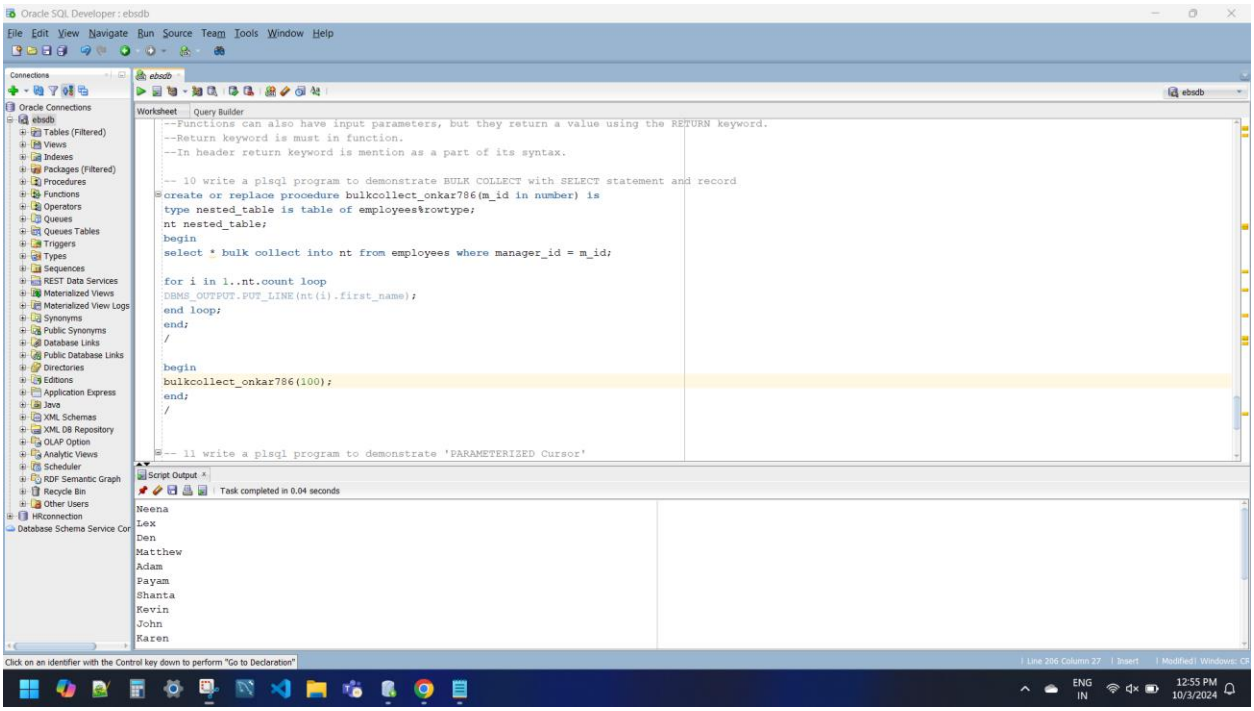
--Return keyword is must in function.

--In header return keyword is mention as a part of its syntax.

-- 10 write a plsql program to demonstrate BULK COLLECT with SELECT statement and record

```
create or replace procedure bulkcollect_onkar786(m_id in number) is

type nested_table is table of employees%rowtype;

nt nested_table;

begin

select * bulk collect into nt from employees where manager_id = m_id;
```

```
for i in 1..nt.count loop

DBMS_OUTPUT.PUT_LINE(nt(i).first_name);

end loop;

end;

/


begin

bulkcollect_onkar786(100);

end;

/
```




```
-- 11 write a plsql program to demonstrate 'PARAMETERIZED Cursor'

declare

cursor e_details(m_id in number) is select * from employees WHERE manager_id = m_id;
```

rec employees%rowtype;

begin

open e_details(&m_id);

loop

exit when e_details%notfound;

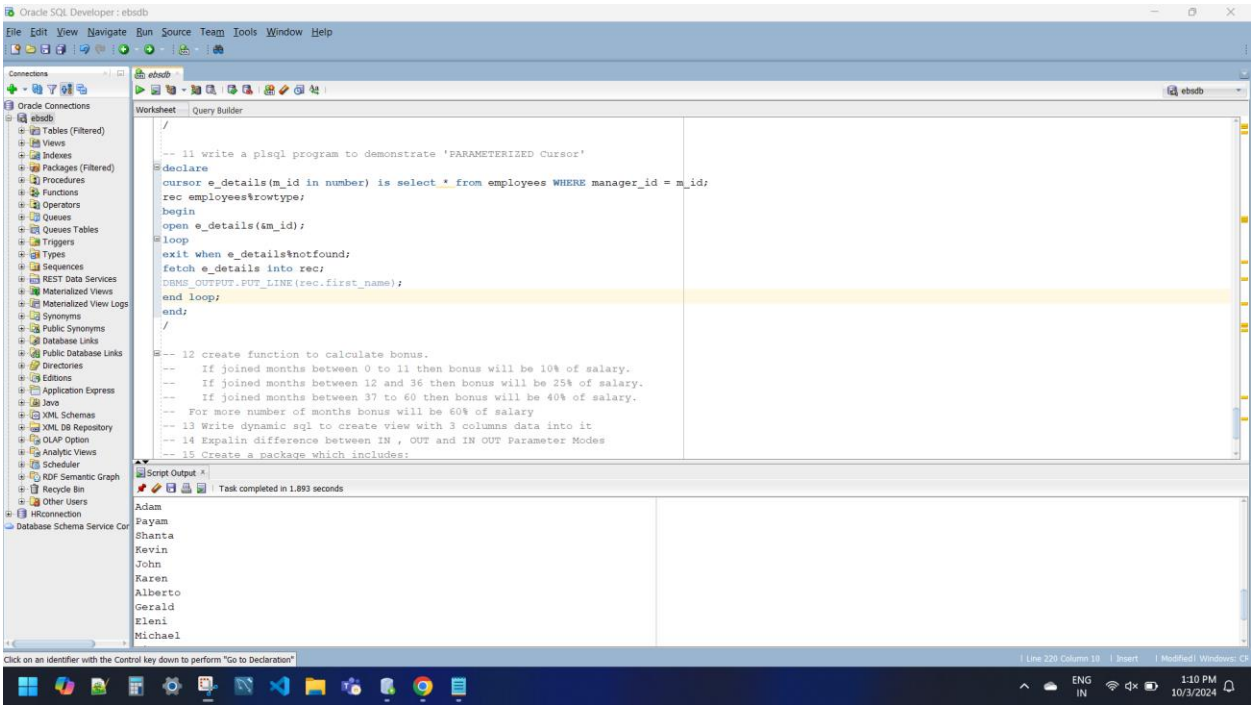
fetch e_details into rec;

DBMS_OUTPUT.PUT_LINE(rec.first_name);

end loop;

end;

/



-- 12 create function to calculate bonus.

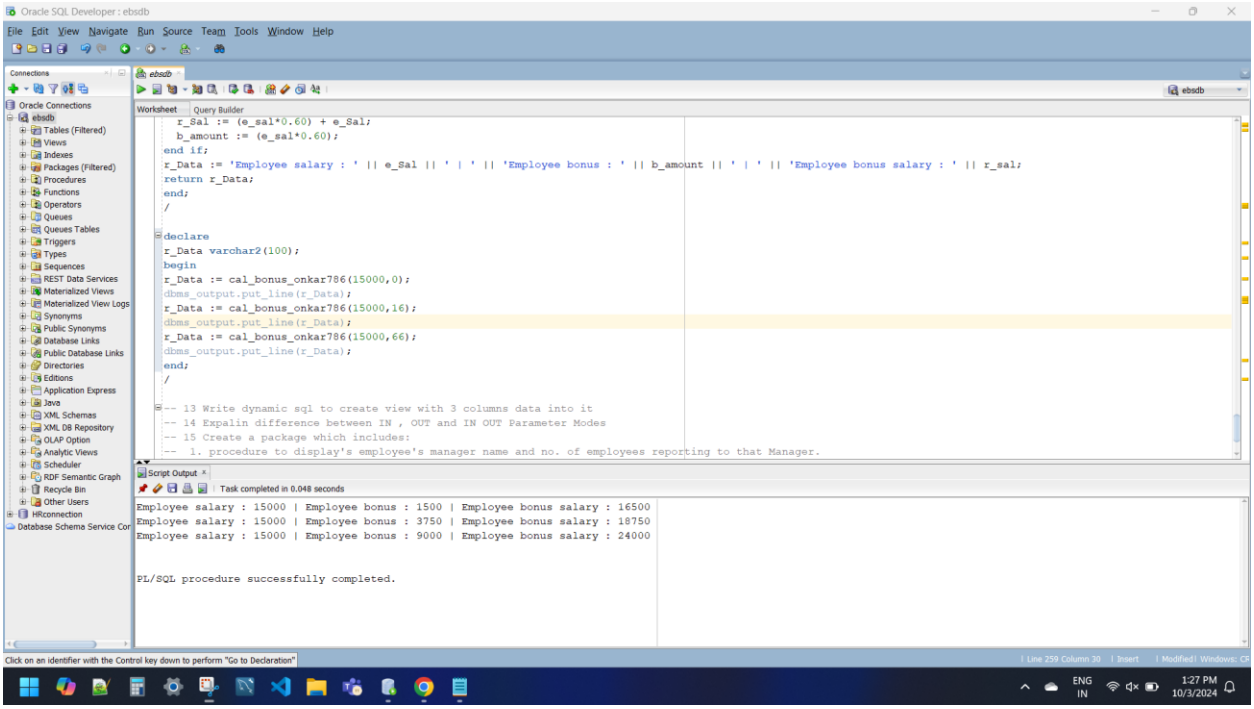
-- If joined months between 0 to 11 then bonus will be 10% of salary.

-- If joined months between 12 and 36 then bonus will be 25% of salary.

-- If joined months between 37 to 60 then bonus will be 40% of salary.

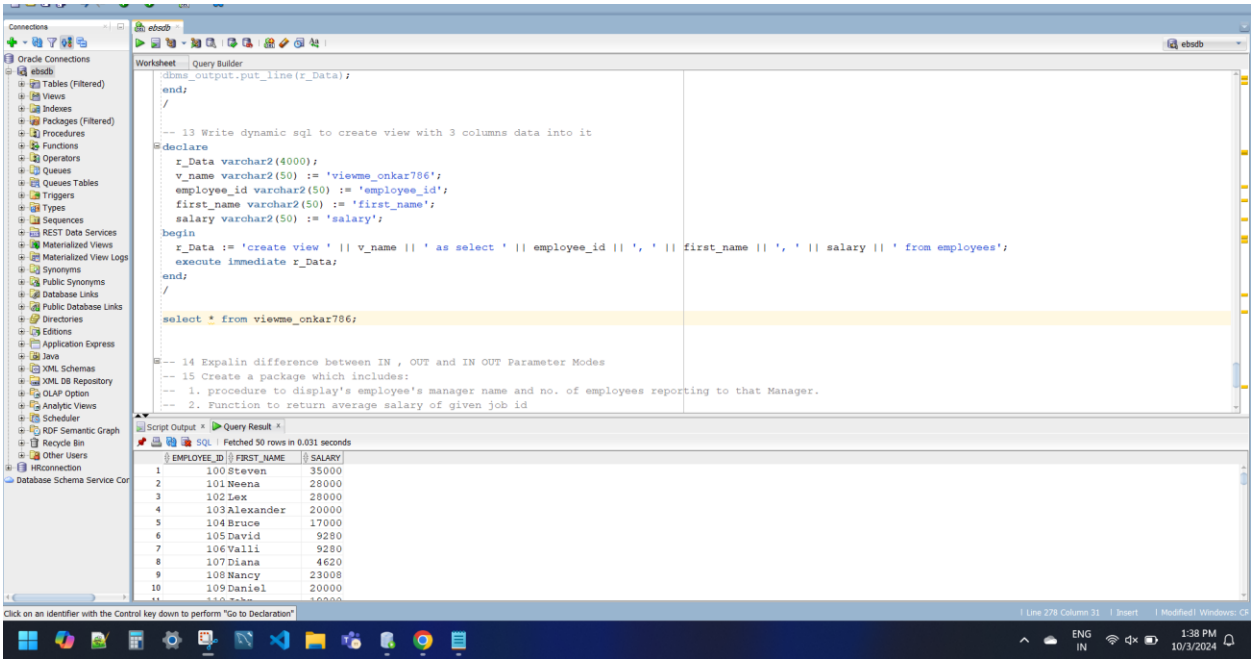
```
--        For more number of months bonus will be 60% of salary


create or replace function cal_bonus_onkar786(e_sal number, joined_months number)
return varchar2 is

r_sal number(10,2);

b_amount number(10,2);

r_Data varchar2(100);

begin

if joined_months between 0 and 11 then

  b_amount := (e_sal*0.10);

  r_Sal := (e_sal*0.10) + e_Sal;

elsif joined_months between 12 and 36 then

  b_amount := (e_sal*0.25);

  r_Sal := (e_sal*0.25) + e_Sal;

elsif joined_months between 37 and 60 then

  r_Sal := (e_sal*0.40) + e_Sal;

  b_amount := (e_sal*0.40);

else

  r_Sal := (e_sal*0.60) + e_Sal;

  b_amount := (e_sal*0.60);

end if;

r_Data := 'Employee salary : ' || e_Sal || ' | ' || 'Employee bonus : ' || b_amount || ' | ' ||
'Employee bonus salary : ' || r_sal;

return r_Data;

end;

/
```

```
declare
r_Data varchar2(100);
begin
r_Data := cal_bonus_onkar786(15000,0);
dbms_output.put_line(r_Data);
r_Data := cal_bonus_onkar786(15000,16);
dbms_output.put_line(r_Data);
r_Data := cal_bonus_onkar786(15000,66);
dbms_output.put_line(r_Data);
end;
/
```




```
-- 13 Write dynamic sql to create view with 3 columns data into it
declare
```

```
  r_Data varchar2(4000);

  v_name varchar2(50) := 'viewme_onkar786';

  employee_id varchar2(50) := 'employee_id';

  first_name varchar2(50) := 'first_name';

  salary varchar2(50) := 'salary';

begin

  r_Data := 'create view ' || v_name || ' as select ' || employee_id || ', ' || first_name || ', ' ||
salary || ' from employees';

  execute immediate r_Data;

end;

/
```




```
select * from viewme_onkar786;
```

```
-- 14 Expalin difference between IN , OUT and IN OUT Parameter Modes

--IN Parameter Mode:
```

--The IN parameter is the default mode.

--The calling program passes the argument to the procedure or function.

--The procedure or function can only read the value but cannot modify it.

--IN parameters act like constants inside the procedure or function, and trying to modify them will result in an error.

--Can set deafult value.

--OUT Parameter Mode:

--An OUT parameter is initially treated as uninitialized within the procedure or function.

--The calling program does not pass a value instead, it receives the value after the procedure or function execution.

--The procedure or function must assign a value to the OUT parameter before exiting.

--It is not default mode.

--Cannot set deafult value.

--IN OUT Parameter Mode:

--It is not default mode.

--Cannot set deafult value.

--The calling program passes a value to the IN OUT parameter.

--The procedure or function can both read and modify the value.

--The modified value is then returned to the calling program when the procedure or function finishes.

-- 15 Create a package which includes:

```
--	1. procedure to display's employee's manager name and no. of employees reporting to that Manager.
--	2. Function to return average salary of given job id
--	3. Procedure to rename any table
--rename emp_onkar to emp_onkar786;

create or replace package pkg_table_onkar is
procedure m_Details(e_id in number, m_name out varchar2, c_emp out number);
function avg_sal(j_id varchar2) return number;
procedure rename_table(t_name in varchar2, mt_name in varchar2);
end pkg_table_onkar;
/

CREATE OR REPLACE
PACKAGE BODY PKG_TABLE_ONKAR AS

 procedure m_Details(e_id in number, m_name out varchar2, c_emp out number) AS
 r_emp number(4);
 BEGIN
  select manager_id into r_emp from employees where employee_id = e_id;
  select first_name into m_name from employees where employee_id = r_emp;
  select count(*) "no of emp" into c_emp from employees group by manager_id;
 END m_Details;

 function avg_sal(j_id varchar2) return number AS
 r_Data number(12,2);
```

```
  BEGIN
    select avg(salary) into r_data from employees group by job_id having job_id = j_id;
    return r_data;
  END avg_sal;

  procedure rename_table(t_name in varchar2, mt_name in varchar2) AS
  r_data varchar2(100);
  BEGIN
    r_data := 'rename table ' || t_name || ' to ' || mt_name;
    execute IMMEDIATE r_data;
  END rename_table;

END PKG_TABLE_ONKAR;
/

declare
m_name varchar2(40);
c_emp number(10,2);
begin
pkg_table_onkar.m_details(125,m_name=>m_name,c_emp=>c_emp);
dbms_output.put_line(m_name || ' ' || c_emp);
end;
/

declare
R_dATA NUMBER(10,2);
```

```
begin
r_Data := pkg_table_onkar.avg_sal('IT_PROG');
dbms_output.put_line(r_data);
end;
/

begin
pkg_table_onkar.rename_table('emp_onkar4056','emphello_onkar');
end;
/
```

Oracle SQL Developer : ebsdb

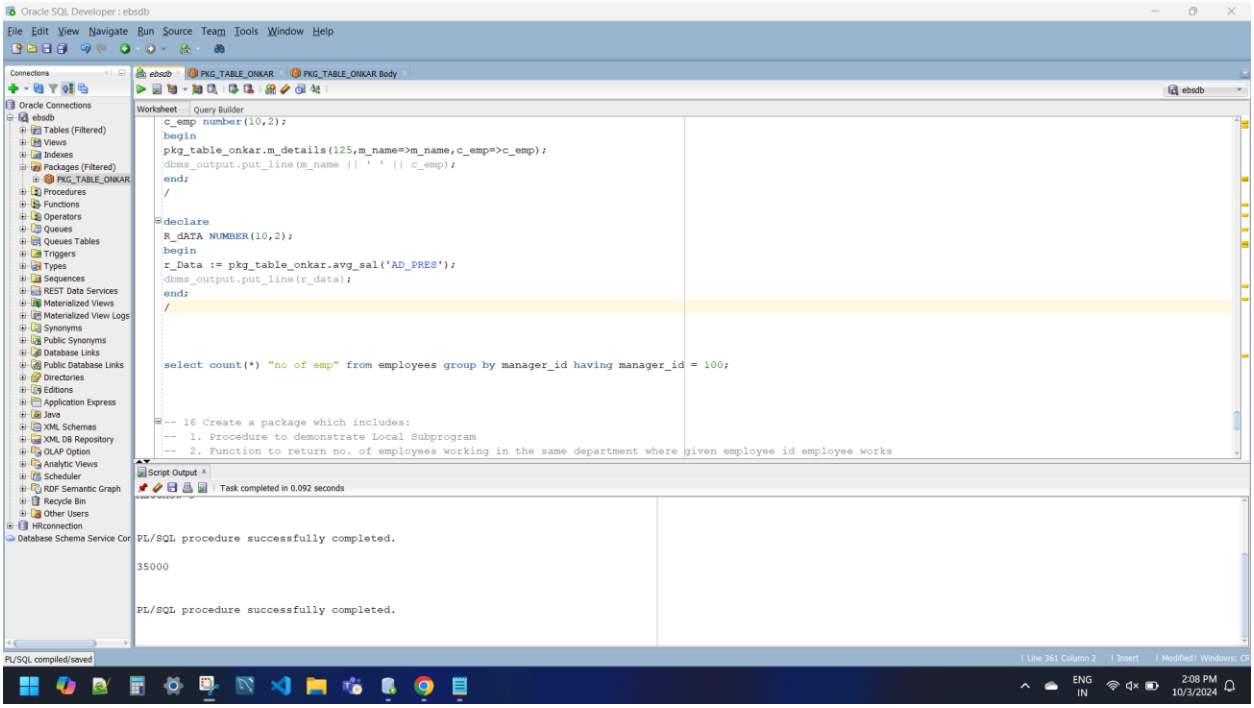
```
dbms_output.put_line(m_name || ' ' || c_emp);
end;
/

declare
R_dATA NUMBER(10,2);
begin
r_Data := pkg_table_onkar.avg_sal('IT_PROG');
dbms_output.put_line(r_data);
end;
/

begin
pkg_table_onkar.rename_table('emp_onkar4056','emphello_onkar');
end;
/

select * from emp_onkar786;


-- 16 Create a package which includes:
--  1. Procedure to demonstrate Local Subprogram
--  2. Function to return no. of employees working in the same department where given employee id employee works
--  3. procedure to update phone number in this way [11]-[23]-(24)-[3599] using IN OUT Mode parameters
```

```
done


PL/SQL procedure successfully completed.
```

Oracle SQL Developer : ebsdb

```
    return r_data;
  END avg_sal;

  procedure rename_table(t_name in varchar2, mt_name in varchar2) AS
  r_data varchar2(100);
  BEGIN
    r_data := 'rename table ' || t_name || ' to ' || mt_name;
    execute IMMEDIATE r_data;
  END rename_table;

END PKG_TABLE_ONKAR;
/

declare
m_name varchar2(40);
c_emp number(10,2);
begin
pkg_table_onkar.m_details(101,m_name=>m_name,c_emp=>c_emp);
dbms_output.put_line(m_name || ' ' || c_emp);
end;
/


select count(*) "no of emp" from employees group by manager_id having manager_id = 100;
```

```
Steven 14


PL/SQL procedure successfully completed.
```



-- 16 Create a package which includes:

-- 1. Procedure to demonstrate Local Subprogram

-- 2. Function to return no. of employees working in the same department where given employee id employee works

-- 3. procedure to update phone number in this way [11]-[23]-(24)-[3599] using IN OUT Mode parameters

```
create or replace package last_question_onkar is

procedure localsub_cal_tax(e_Sal in number);

function no_of_emp_dept(e_id number) return number;

procedure modify_phone_no(p_no in out varchar2);

end;

/
```

CREATE OR REPLACE

```
PACKAGE BODY LAST_QUESTION_ONKAR AS

 procedure localsub_cal_tax(e_Sal in number) AS
 r_data number(10,2);
 function tax(e_Sal number) return number is
 r_sal number(10,2);
 begin
 if e_sal >0 and e_sal<10000 then
   r_Sal := (e_sal*0.10)+e_sal;
 elsif e_sal >10001 and e_sal<20000 then
   r_Sal := (e_sal*0.20)+e_sal;
 elsif e_sal >20001 and e_sal<30000 then
   r_Sal := (e_sal*0.30)+e_sal;
 else
   r_Sal := (e_sal*0.50)+e_sal;
 end if;
 return r_sal;
 end tax;
 BEGIN
  DBMS_OUTPUT.PUT_LINE('employee_salary is :' || e_Sal );
  r_data := tax(e_Sal);
  DBMS_OUTPUT.PUT_LINE('employee_salary after tax is :' || r_data );
 END localsub_cal_tax;


 function no_of_emp_dept(e_id number) return number AS
 r_data number(10,2);
```

```
  d_id number(4);

  BEGIN

  select department_id into d_id from employees where employee_id = e_id;

  select count(*) into r_Data from employees group by department_id having department_id
= d_id;

  return r_Data;

  END no_of_emp_dept;


  procedure modify_phone_no(p_no in out varchar2) AS

  BEGIN

   p_no := '[' || substr(p_no,1,2) || ']-[' || substr(p_no,3,2) || ']-(' || substr(p_no,5,2) || ')-[' ||
substr(p_no,7,4) || ']';

  END modify_phone_no;


END LAST_QUESTION_ONKAR;

/


begin

last_question_onkar.localsub_cal_tax(15000);

end;

/


declare

r_Data number(10,2);

begin

r_Data := last_question_onkar.no_of_emp_dept(102);
```

```
dbms_output.put_line(r_Data);
end;
/

declare
p_no varchar2(100) := &phone_no;
begin
last_question_onkar.modify_phone_no(p_no=>p_no);
dbms_output.put_line(p_no);
end;
/
```

Oracle SQL Developer : ebsdb

```
begin
last_question_onkar.localsub_cal_tax(15000);
end;
/

declare
r_Data number(10,2);
begin
r_Data := last_question_onkar.no_of_emp_dept(102);
dbms_output.put_line(r_Data);
end;
/

declare
p_no varchar2(100) := &phone_no;
begin
last_question_onkar.modify_phone_no(p_no=>p_no);
dbms_output.put_line(p_no);
end;
/
```

Task completed in 3.225 seconds

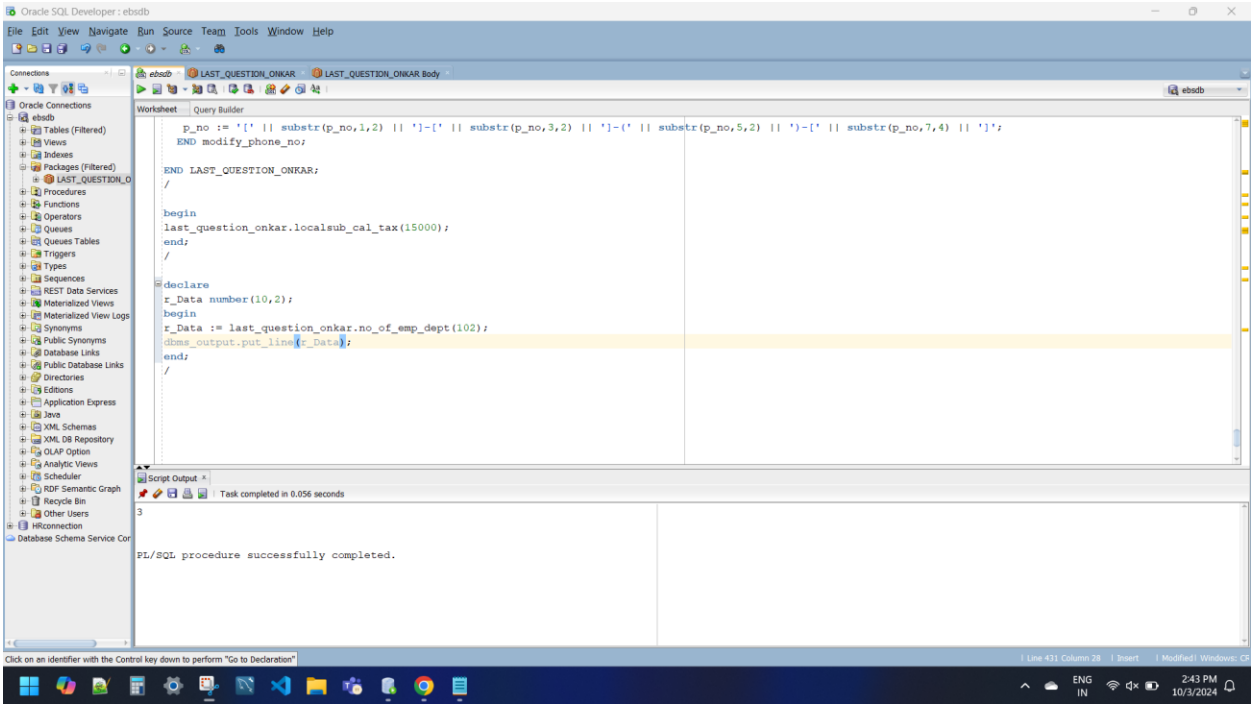
```
begin
last_question_onkar.modify_phone_no(p_no=>p_no);
dbms_output.put_line(p_no);
end;
[93]-[24]-(46)-[4800]


PL/SQL procedure successfully completed.
```

Oracle SQL Developer : ebsdb

```
    select count(*) into r_Data from employees group by department_id having department_id = d_id;
    return r_Data;
    END no_of_emp_dept;

    procedure modify_phone_no(p_no in out varchar2) AS
    BEGIN
     p_no := '[' || substr(p_no,1,2) || ']-[' || substr(p_no,3,2) || ']-(' || substr(p_no,5,2) || ')-[' || substr(p_no,7,4) || ']';
    END modify_phone_no;

  END LAST_QUESTION_ONKAR;
  /

declare

begin
last_question_onkar.localsub_cal_tax(15000);
end;
/
```

Task completed in 0.055 seconds

```
employee_salary is :15000
employee_salary after tax is :18000


PL/SQL procedure successfully completed.
```


Oracle SQL Developer : ebsdb
File Edit View Navigate Run Source Team Tools Window Help
ebsdb
LAST_QUESTION_ONKAR
LAST_QUESTION_ONKAR Body
Worksheet
Query Builder
p_no := '[' || substr(p_no,1,2) || ']-[' || substr(p_no,3,2) || ']-(' || substr(p_no,5,2) || ')-[' || substr(p_no,7,4) || ']';
END modify_phone_no;
END LAST_QUESTION_ONKAR;
/
begin
last_question_onkar.localsub_cal_tax(15000);
end;
/
declare
r_Data number(10,2);
begin
r_Data := last_question_onkar.no_of_emp_dept(102);
dbms_output.put_line(r_Data);
end;
/
Script Output
Task completed in 0.056 seconds
3
PL/SQL procedure successfully completed.
Click on an identifier with the Control key down to perform "Go to Declaration"
Line 431 Column 28
Insert
Modified
Windows: CR
ENG
IN
2:43 PM
10/3/2024