-- Dynamic SQL

--1. write a plsql program to demostrate Execute immediate with RENAME statement

select * from emp_onkar;

declare

r_Data varchar2(100);

t_name varchar2(40) := 'emp_onkar';

c_name varchar2(40) := 'FIRST_NAME';

n_name varchar2(40) := 'e_name';

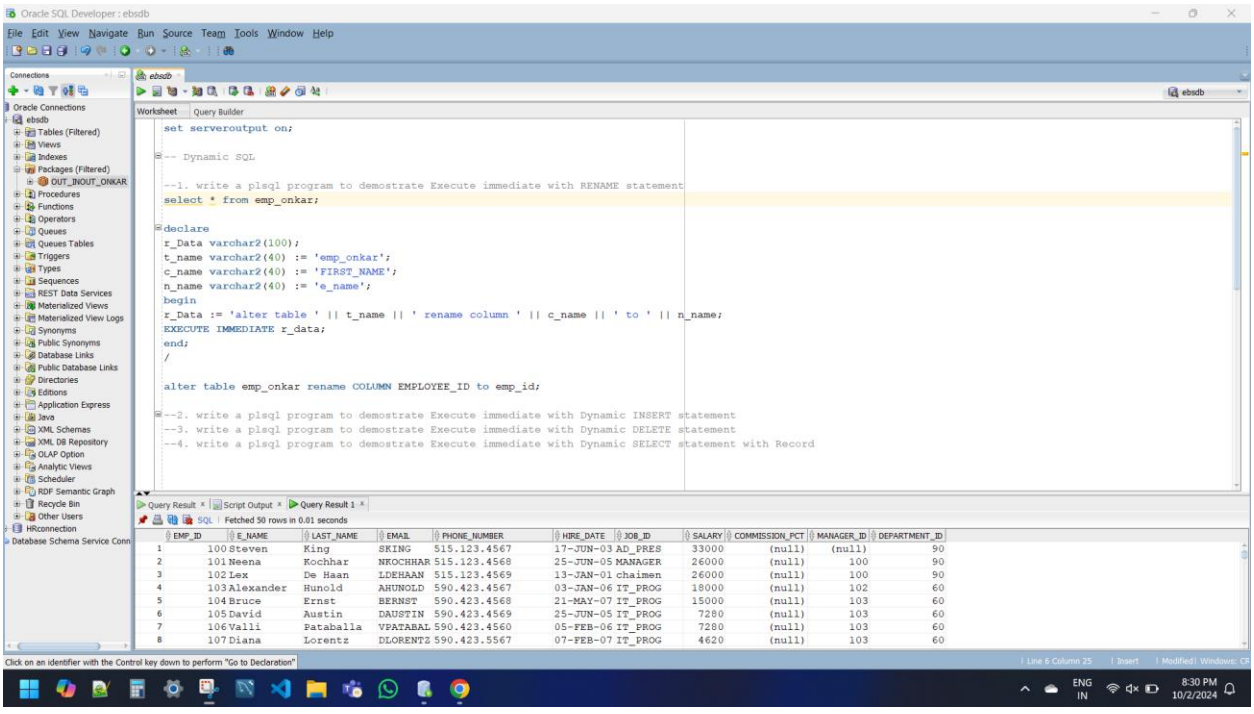
begin

r_Data := 'alter table ' || t_name || ' rename column ' || c_name || ' to ' || n_name;

EXECUTE IMMEDIATE r_data;

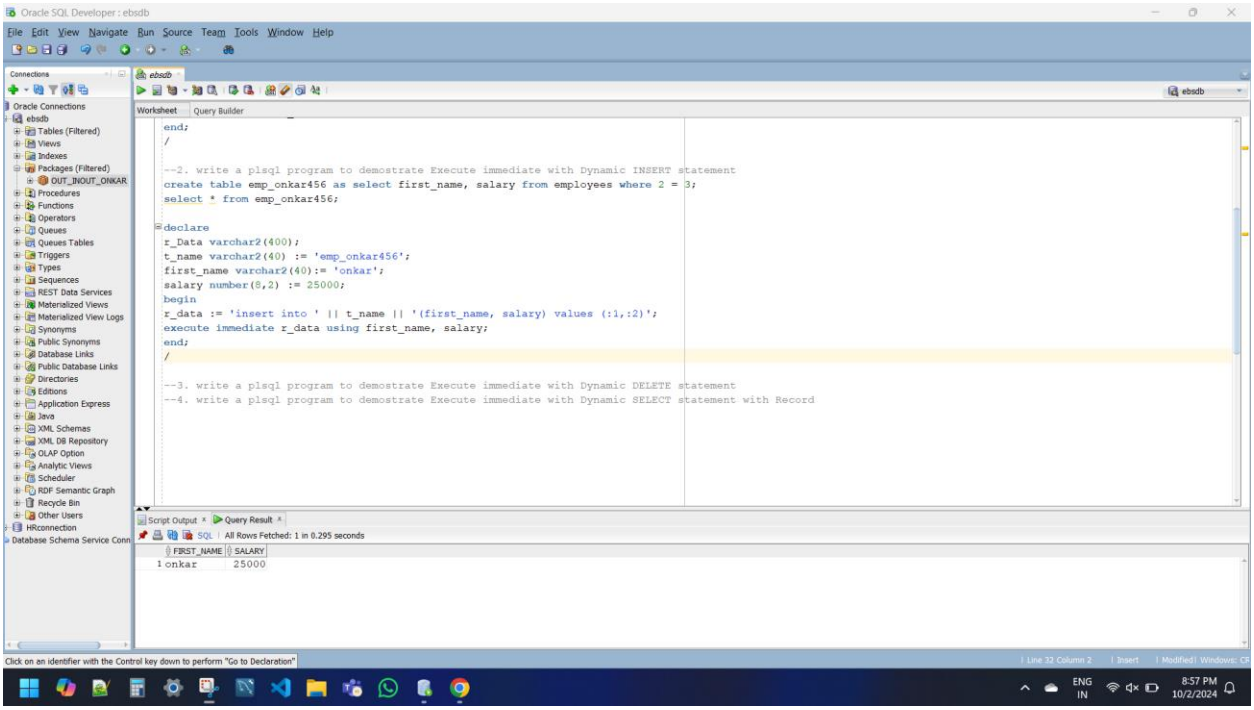
end;

/



--2. write a plsql program to demostrate Execute immediate with Dynamic INSERT statement

```
create table emp_onkar456 as select first_name, salary from employees where 2 = 3;

select * from emp_onkar456;


declare

r_Data varchar2(400);

t_name varchar2(40) := 'emp_onkar456';

first_name varchar2(40):= 'onkar';

salary number(8,2) := 25000;

begin

r_data := 'insert into ' || t_name || '(first_name, salary) values (:1,:2)';

execute immediate r_data using first_name, salary;

end;

/
```



--3. write a plsql program to demostrate Execute immediate with Dynamic DELETE statement

declare

r_Data varchar2(400);

t_name varchar2(40) := 'emp_onkar456';

f_name varchar2(40):= 'onkar';

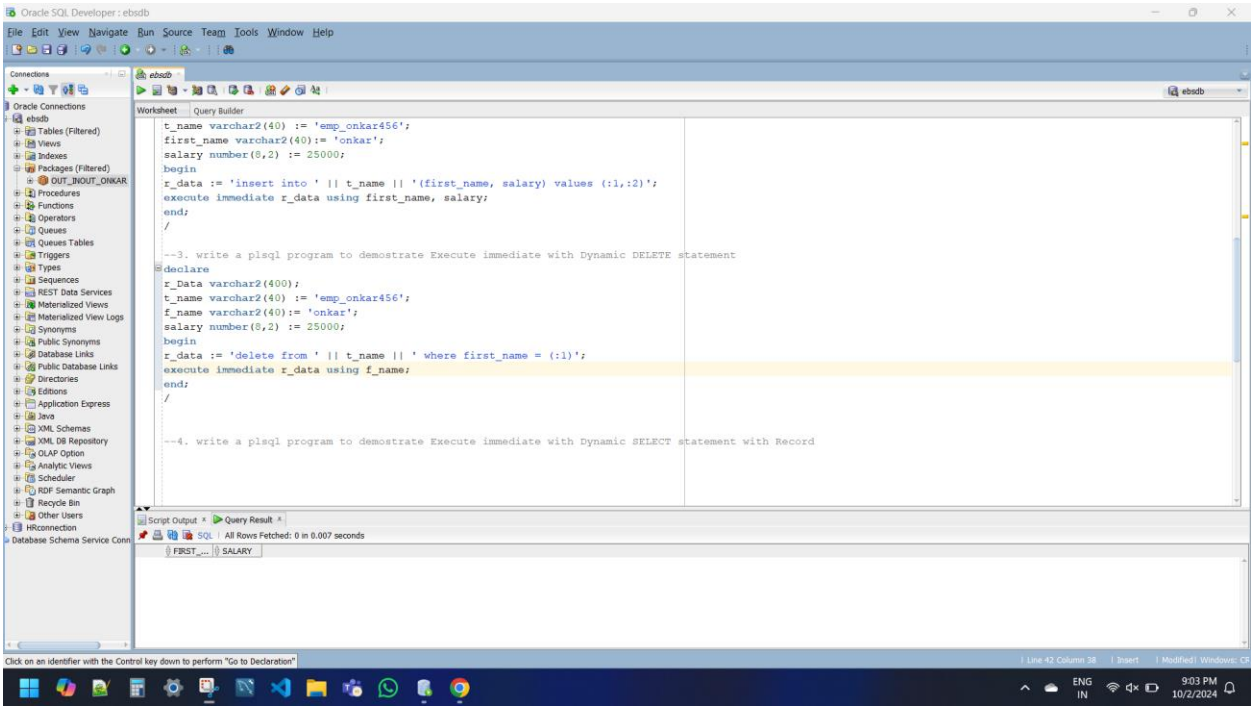
begin

r_data := 'delete from ' || t_name || ' where first_name = (:1)';

execute immediate r_data using f_name;

end;

/

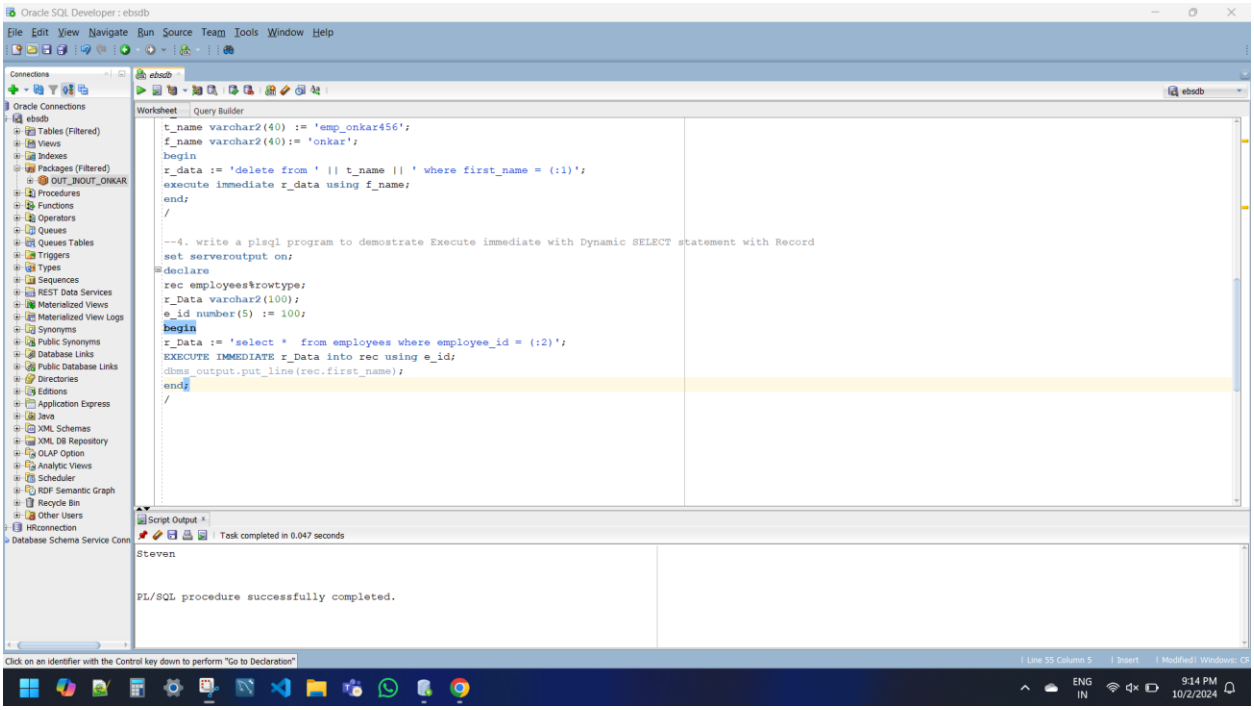


--4. write a plsql program to demostrate Execute immediate with Dynamic SELECT statement with Record

```
set serveroutput on;

declare

rec employees%rowtype;

r_Data varchar2(100);

e_id number(5) := 100;

begin

r_Data := 'select *  from employees where employee_id = (:2)';

EXECUTE IMMEDIATE r_Data into rec using e_id;

dbms_output.put_line(rec.first_name);

end;

/
```

```
set serveroutput on;
```

dbms_output.put_line();



-- Design Consideration with PLSQL Code

--1. write a plsql program to demonstrate standarding exception

select * from emp_onkar456;

alter table emp_onkar456 modify FIRST_NAME varchar2(50) not null;

create or replace package std_exception_onkar is

nn_insert_val exception;

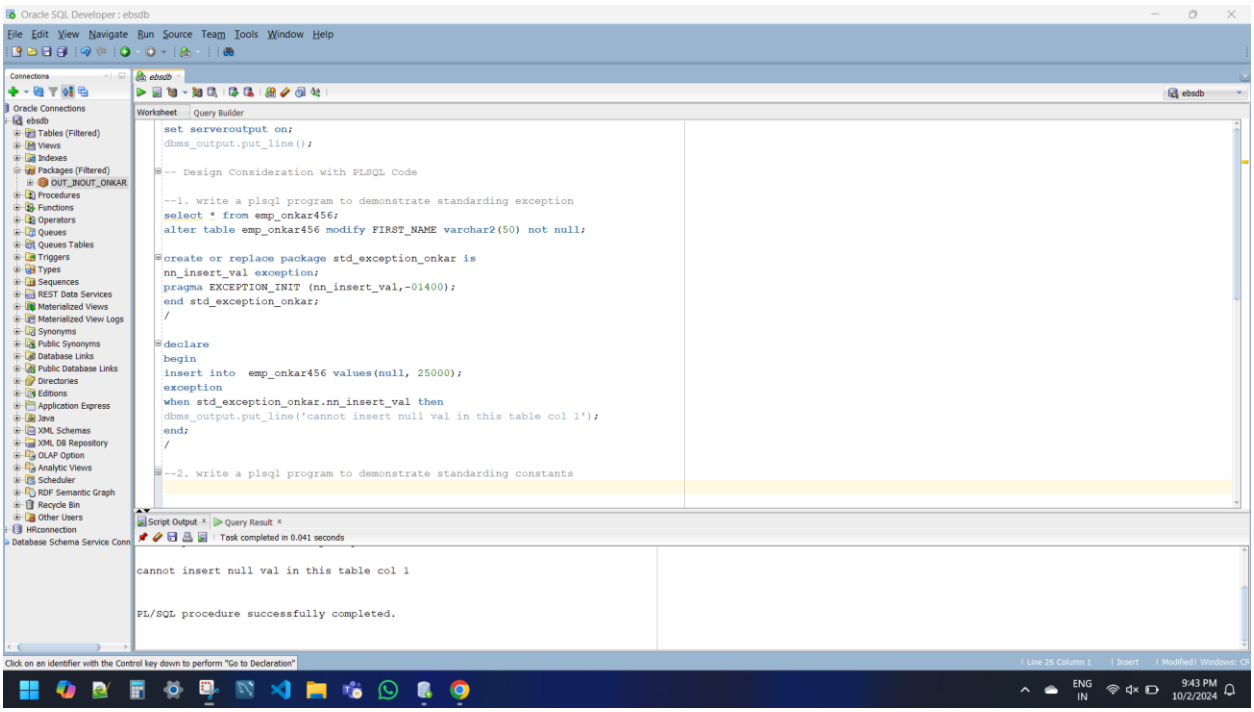
pragma EXCEPTION_INIT (nn_insert_val,-01400);

end std_exception_onkar;

/

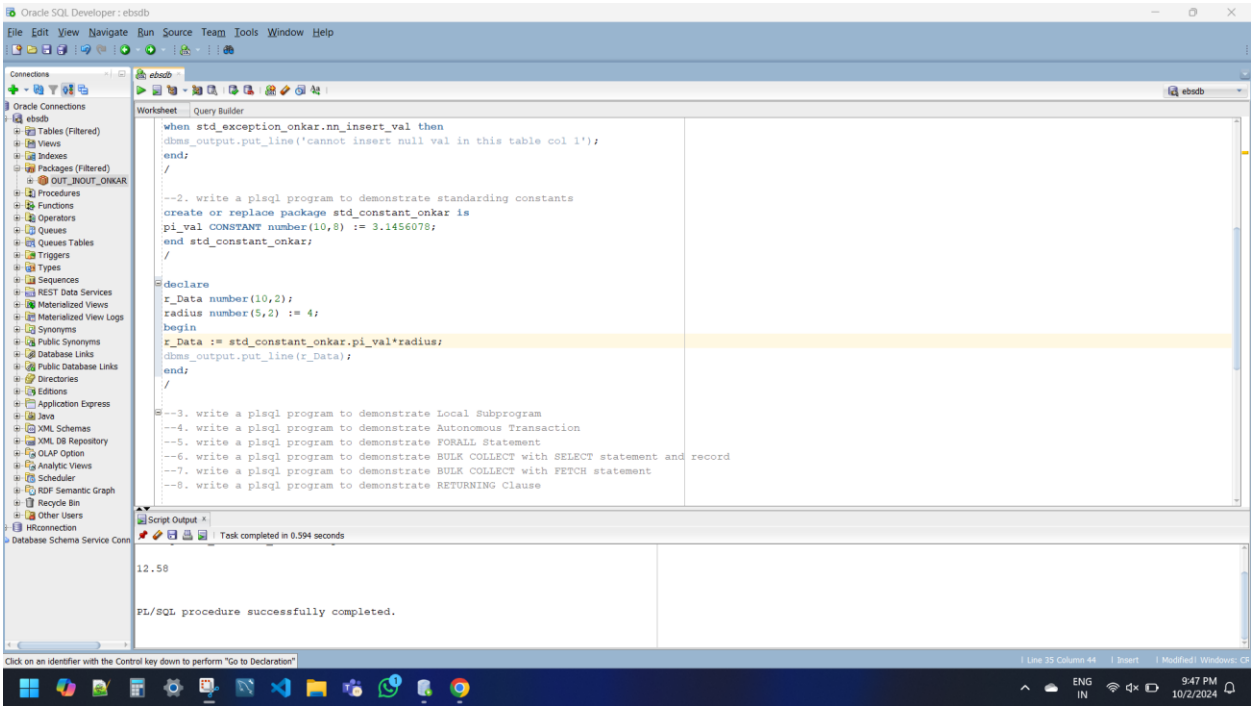
declare

```
begin
insert into  emp_onkar456 values(null, 25000);
exception
when std_exception_onkar.nn_insert_val then
dbms_output.put_line('cannot insert null val in this table col 1');
end;
/
```



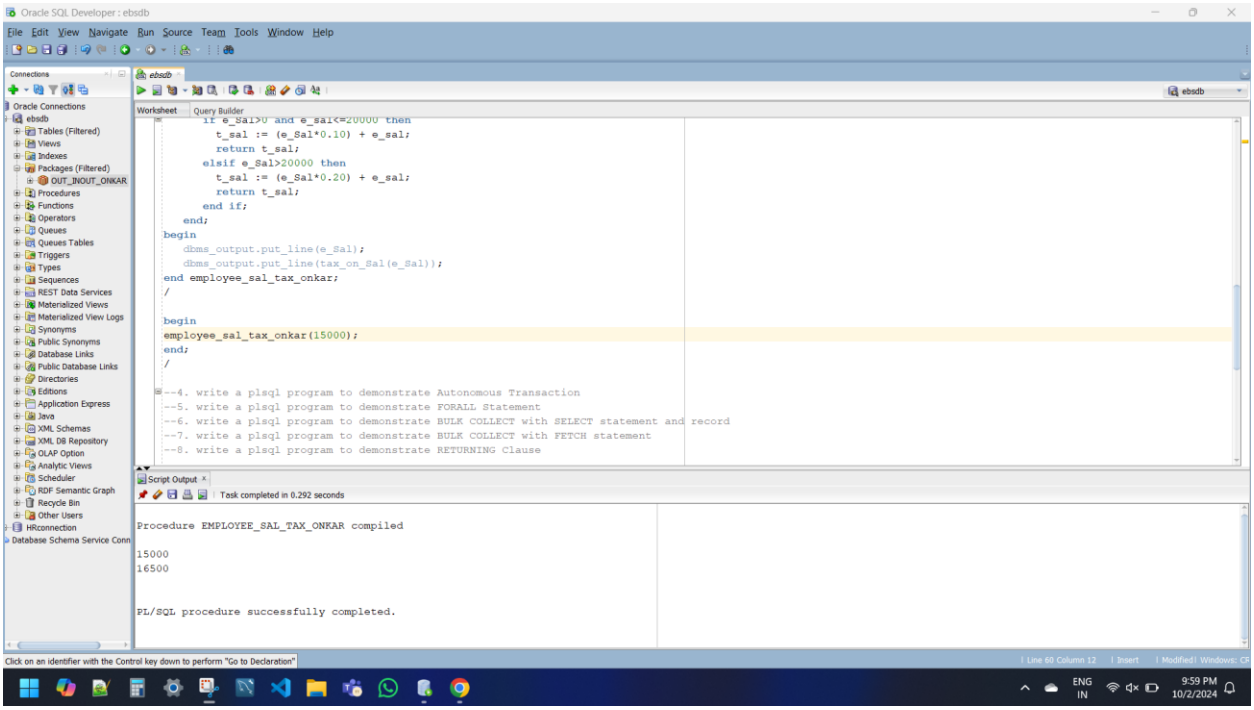

```
--2. write a plsql program to demonstrate standarding constants
create or replace package std_constant_onkar is
pi_val CONSTANT number(10,8) := 3.1456078;
end std_constant_onkar;
/

declare
```

```
r_Data number(10,2);
radius number(5,2) := 4;
begin
r_Data := std_constant_onkar.pi_val*radius;
dbms_output.put_line(r_Data);
end;
/
```




```
--3. write a plsql program to demonstrate Local Subprogram
create or replace procedure employee_sal_tax_onkar(e_Sal in number) is
t_Sal number(10,2);
function tax_on_Sal(e_Sal number) return number is
  begin
    if e_Sal>0 and e_sal<=20000 then
      t_sal := (e_Sal*0.10) + e_sal;
```

```
      return t_sal;
    elsif e_Sal>20000 then
      t_sal := (e_Sal*0.20) + e_sal;
      return t_sal;
    end if;
  end;
begin
  dbms_output.put_line(e_Sal);
  dbms_output.put_line(tax_on_Sal(e_Sal));
end employee_sal_tax_onkar;
/

begin
employee_sal_tax_onkar(15000);
end;
/
```



--4. write a plsql program to demonstrate Autonomous Transaction

create table marathon_onkar(distance number(10));

create table sprint_onkar(distance number(10));

select * from marathon_onkar;

select * from sprint_onkar;

create or replace function sprint_func(s_dis number) return varchar2 is

r_Data varchar2(40);

pragma autonomous_transaction;

begin

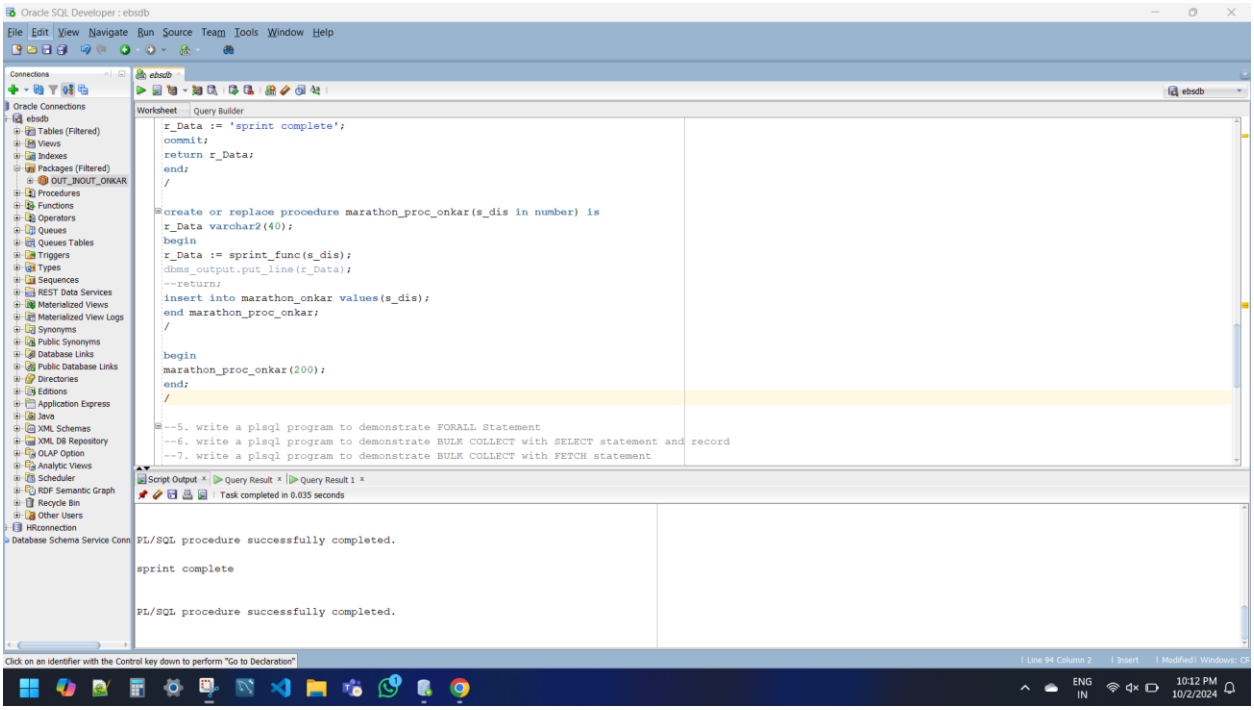
insert into sprint_onkar values(s_dis);

r_Data := 'sprint complete';

commit;

return r_Data;

```
end;
/

create or replace procedure marathon_proc_onkar(s_dis in number) is
r_Data varchar2(40);
begin
r_Data := sprint_func(s_dis);
dbms_output.put_line(r_Data);
--return;
insert into marathon_onkar values(s_dis);
end marathon_proc_onkar;
/

begin
marathon_proc_onkar(200);
end;
/
```



--5. write a plsql program to demonstrate FORALL Statement

select * from emp_onkar;

create or replace procedure forall_onkar(e_Sal in number) is

type nes_table is table of number;

nt nes_table;

begin

nt := nes_table(100,102,103,104);

forall i in 1..nt.count

update emp_onkar

set salary = e_Sal

where emp_id = nt(i);

end;

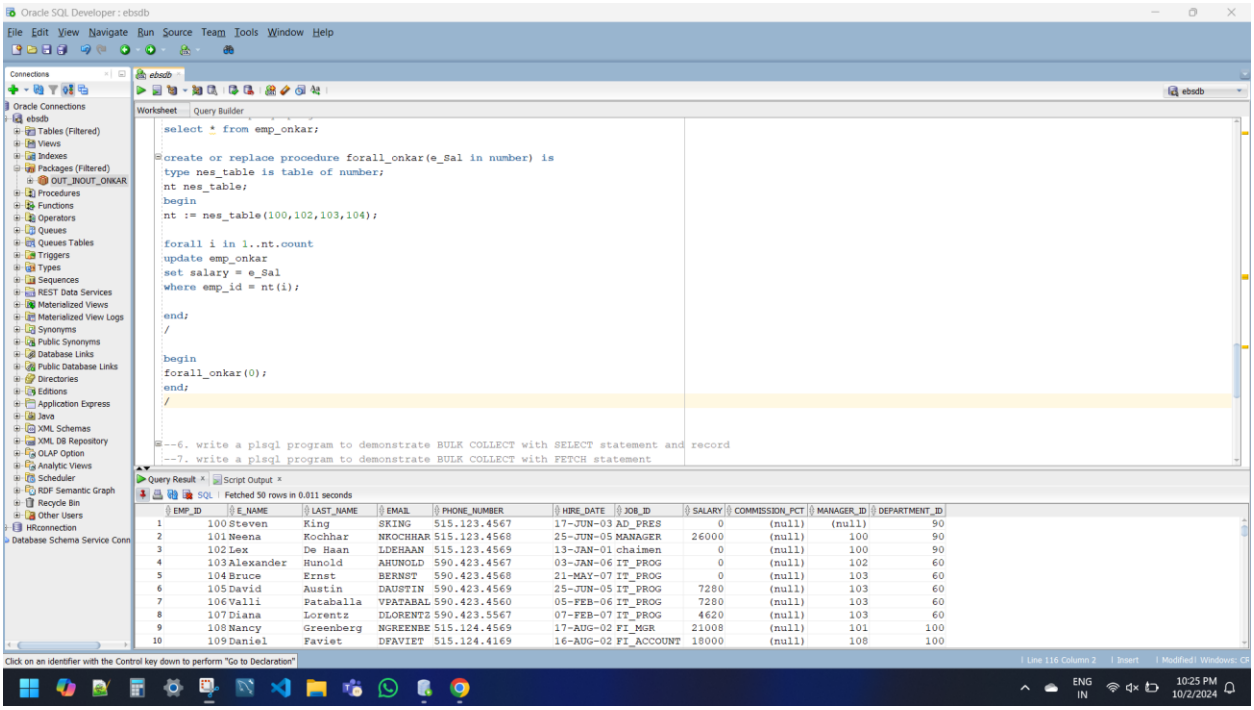
/

begin

forall_onkar(0);

end;

/



--6. write a plsql program to demonstrate BULK COLLECT with SELECT statement and record

create or replace procedure bulcollect(m_id in number) is

--rec employees%rowtype;

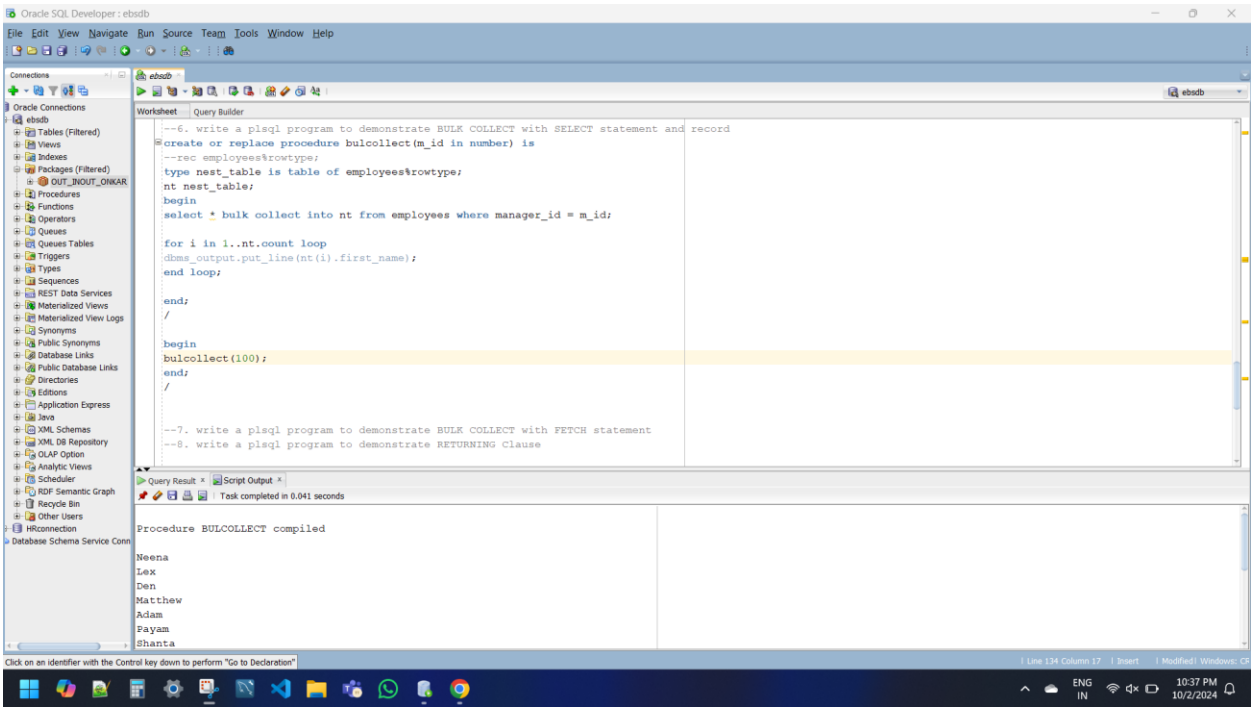
type nest_table is table of employees%rowtype;

nt nest_table;

begin

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

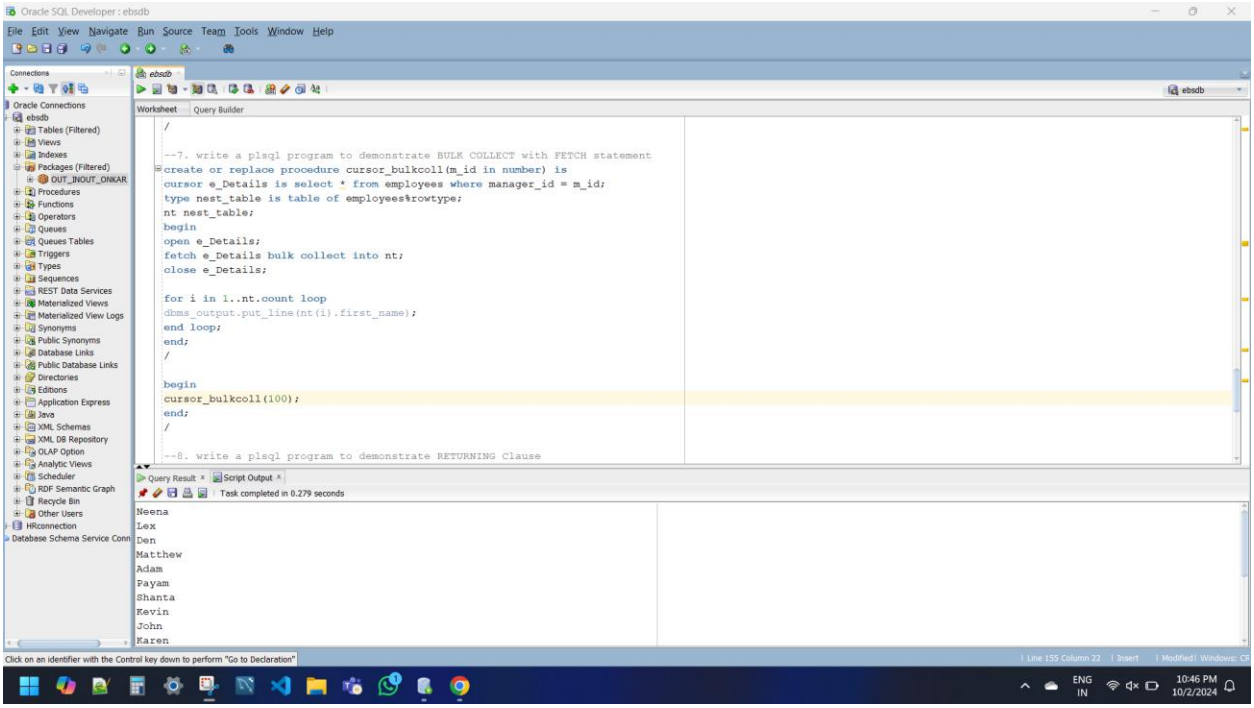
```
for i in 1..nt.count loop
dbms_output.put_line(nt(i).first_name);
end loop;


end;
/


begin
bulcollect(100);
end;
/
```



--7. write a plsql program to demonstrate BULK COLLECT with FETCH statement

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

```
cursor e_Details is select * from employees where manager_id = m_id;
type nest_table is table of employees%rowtype;
nt nest_table;
begin
open e_Details;
fetch e_Details bulk collect into nt;
close e_Details;

for i in 1..nt.count loop
dbms_output.put_line(nt(i).first_name);
end loop;
end;
/

begin
cursor_bulkcoll(100);
end;
/
```



--8. write a plsql program to demonstrate RETURNING Clause

select * from emp_onkar;

create or replace procedure returning_clause(e_id in number) is

f_name varchar2(40);

begin

update emp_onkar

set salary = 0

where emp_id = e_id returning e_name into f_name;

dbms_output.put_line(f_name);

end;

/

begin

returning_clause(100);

end;

/