

-- Dynamic SQL

--1. write a plsql program to demonstrate 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;

/

The screenshot displays the Oracle SQL Developer interface. The main window shows a PL/SQL script with the following content:

```
set serveroutput on;

-- Dynamic SQL

--1. write a plsql program to demonstrate 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;
/

alter table emp_onkar rename COLUMN EMPLOYEE_ID to emp_id;
```

Below the script, the 'Query Result' window shows the output of the first query, which is a list of employees from the emp\_onkar table. The results are as follows:

EMP_ID	E_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	Steven	King	SKING	515.123.4567	17-JUN-03	AD_PRES	33000	(null)	(null)	90
2	Neena	Kochhar	NKOCHHAR	515.123.4568	25-JUN-05	MANAGER	26000	(null)	100	90
3	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-01	chaimen	26000	(null)	100	90
4	Alexander	Rumold	ARUMOLD	590.423.4567	03-JAN-06	IT_PROG	19000	(null)	102	60
5	Bruce	Ernst	BERNST	590.423.4568	21-MAY-07	IT_PROG	15000	(null)	103	60
6	David	Austin	DAUSTIN	590.423.4569	25-JUN-05	IT_PROG	7280	(null)	103	60
7	Valli	Pataballa	VPATABAL	590.423.4560	05-FEB-06	IT_PROG	7280	(null)	103	60
8	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-07	IT_PROG	4620	(null)	103	60

--2. write a plsql program to demonstrate 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;

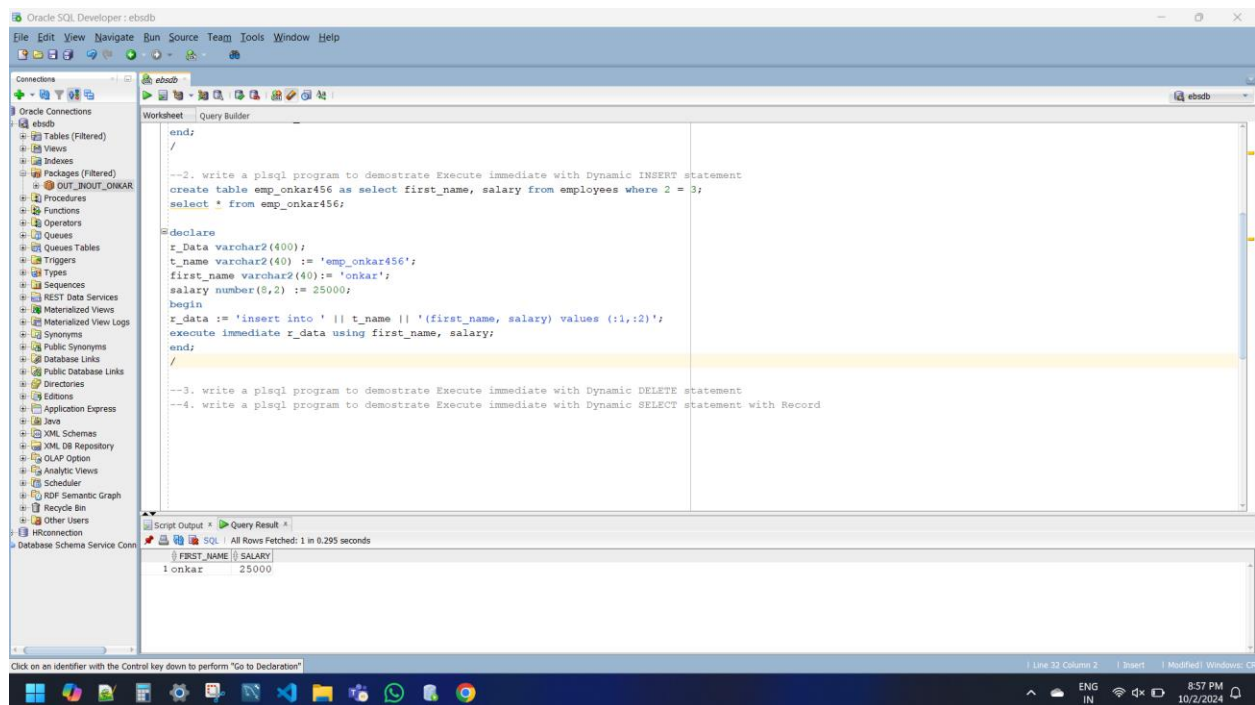
begin

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 demonstrate Execute immediate with Dynamic DELETE statement

declare

r\_Data varchar2(400);

t\_name varchar2(40) := 'emp\_onkar456';

f\_name varchar2(40):= 'onkar';

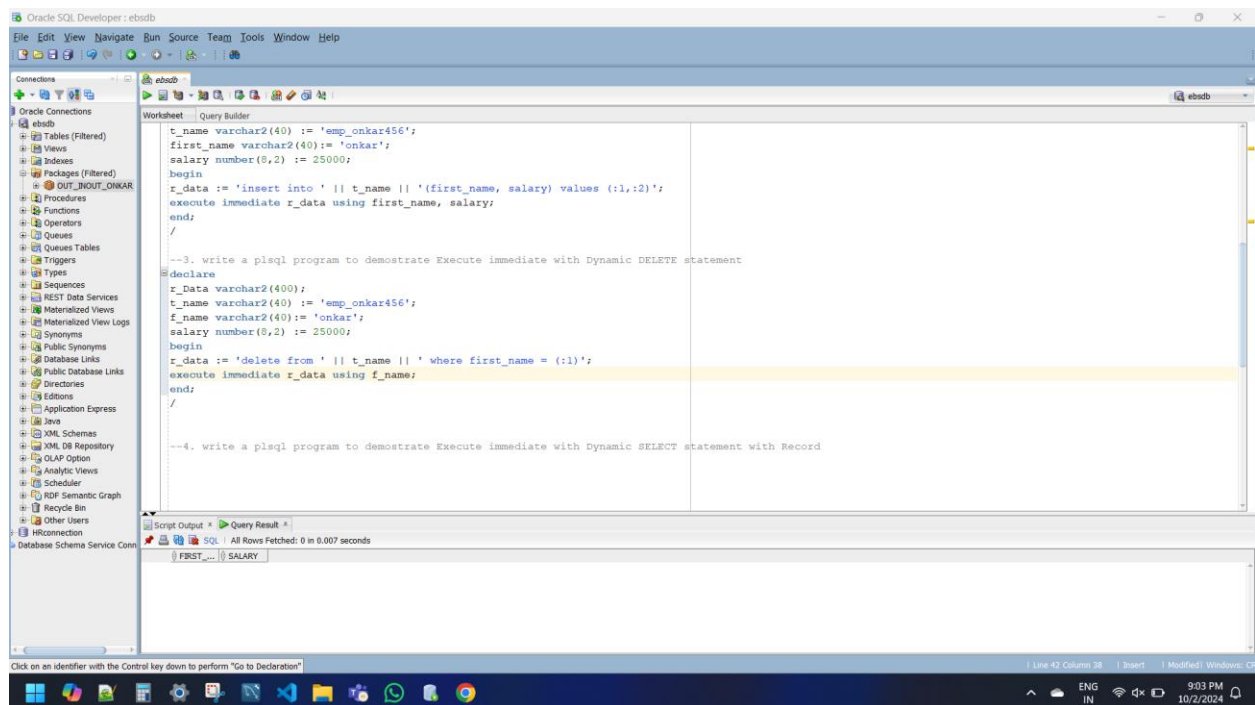
begin

r\_data := 'delete from ' || t\_name || ' where first\_name = (:1)';

execute immediate r\_data using f\_name;

end;

/



--4. write a plsql program to demonstrate 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;

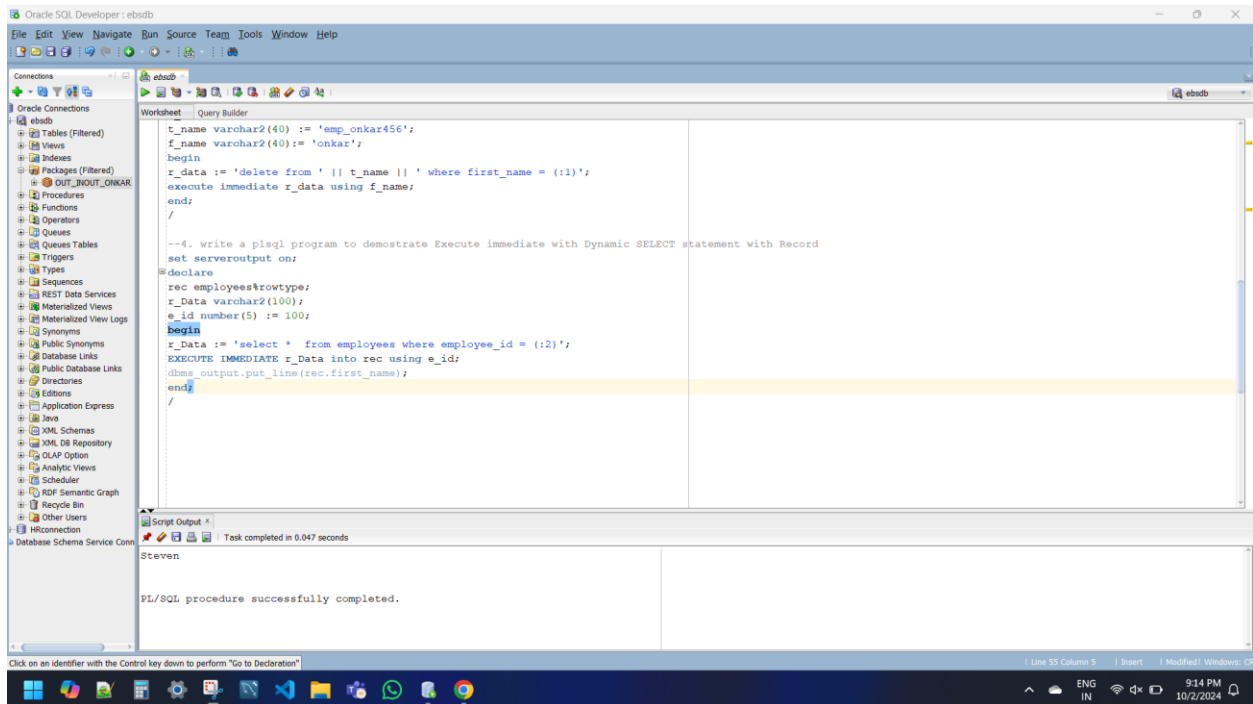
dbms\_output.put\_line(rec.first\_name);

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);

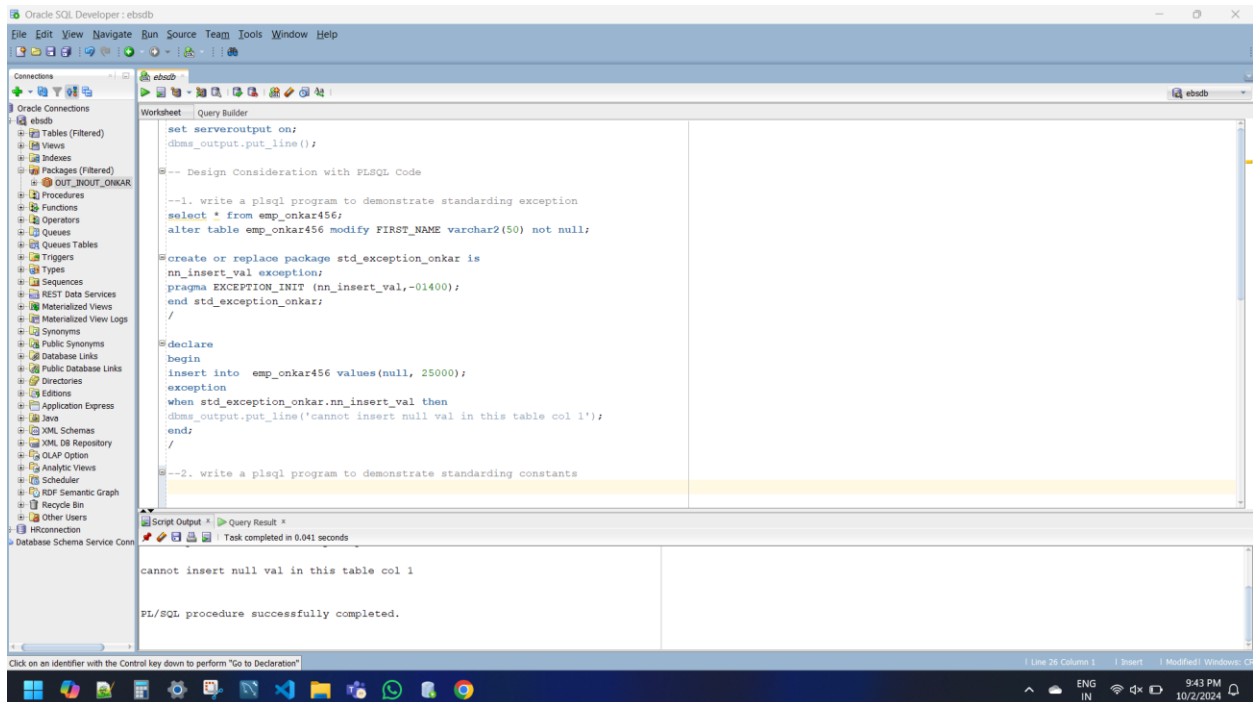
exception

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

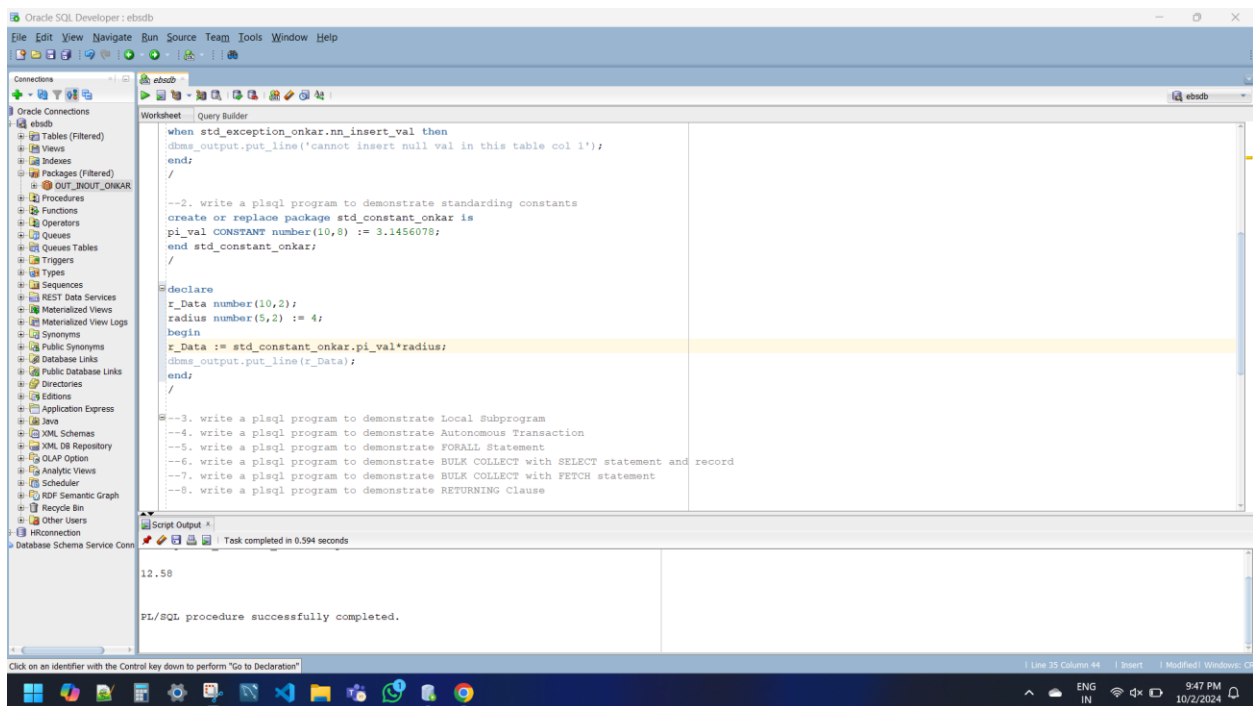
r_Data := std_constant_onkar.pi_val*radius;

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;

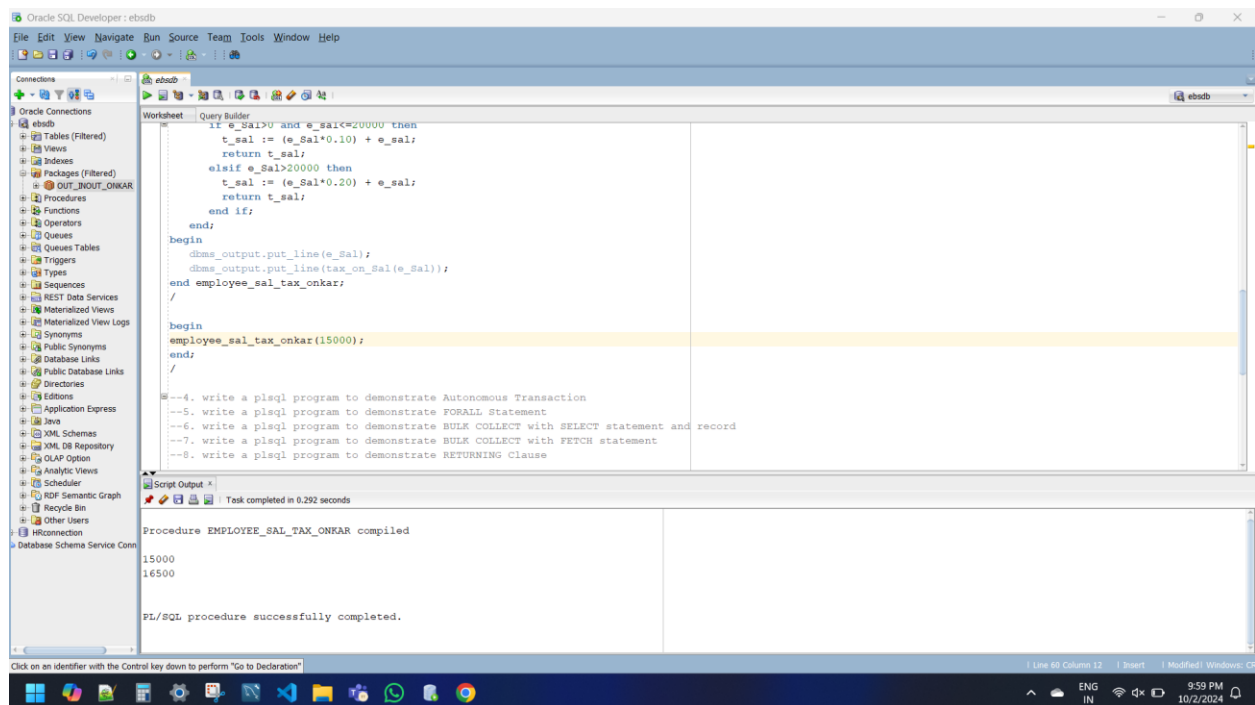
begin
    dbms_output.put_line(e_Sal);
    dbms_output.put_line(tax_on_Sal(e_Sal));
end employee_sal_tax_onkar;

/
```

```
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;
```

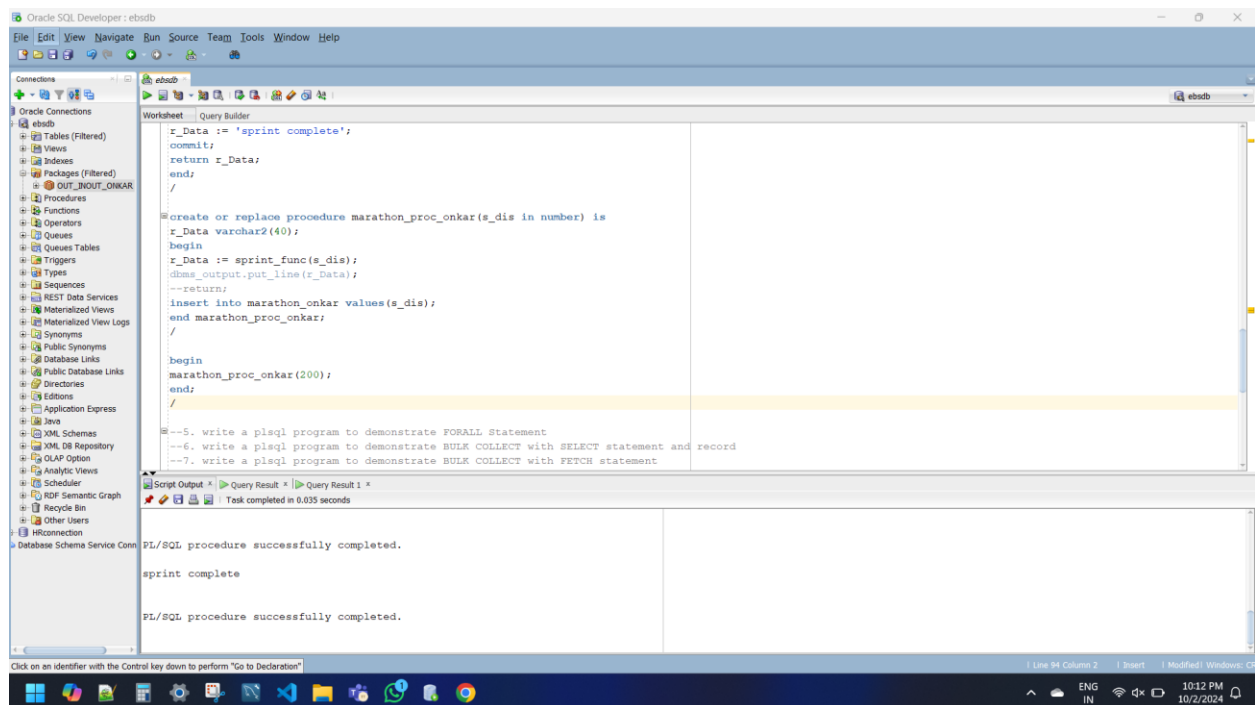
```
/
```

```
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;

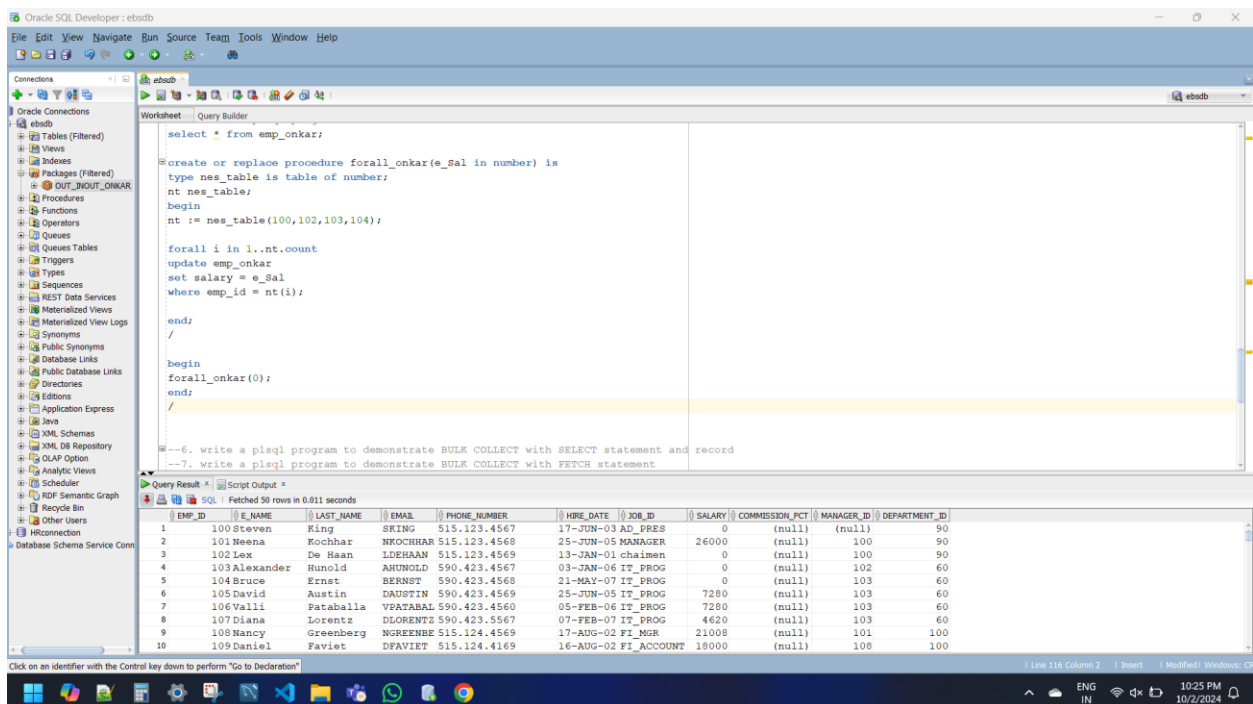
/

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;
```

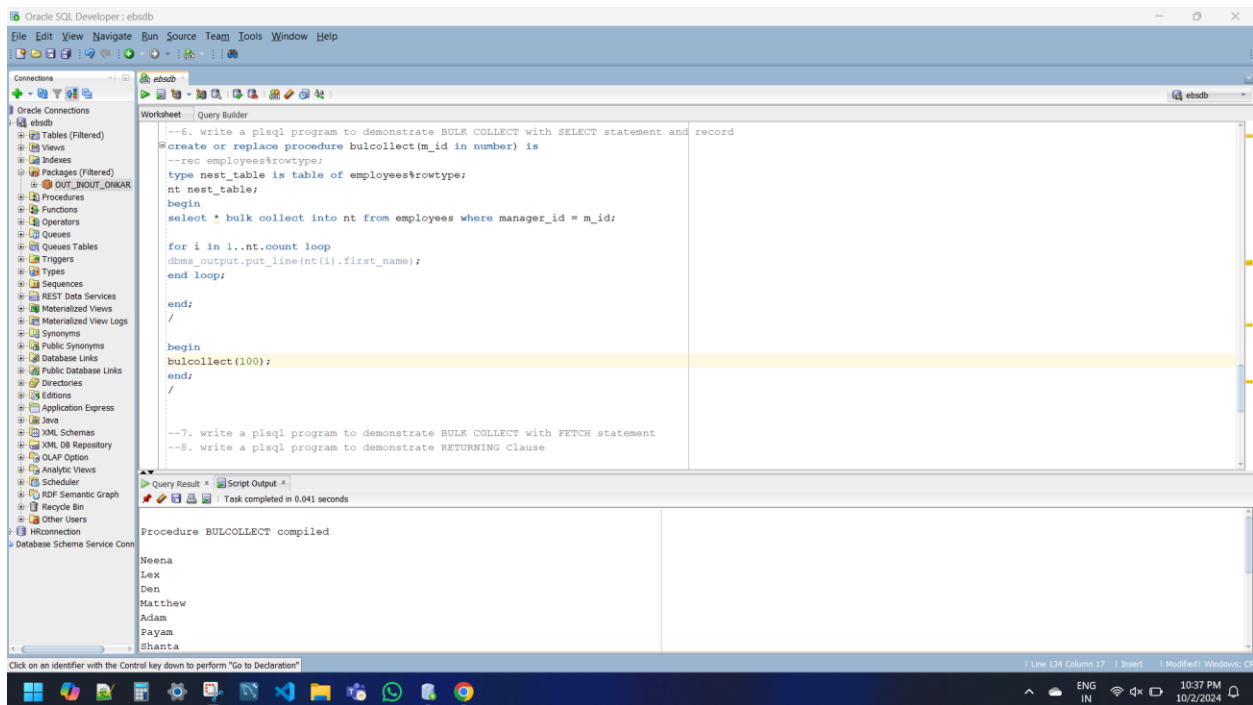
```
/
```

```
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;
```

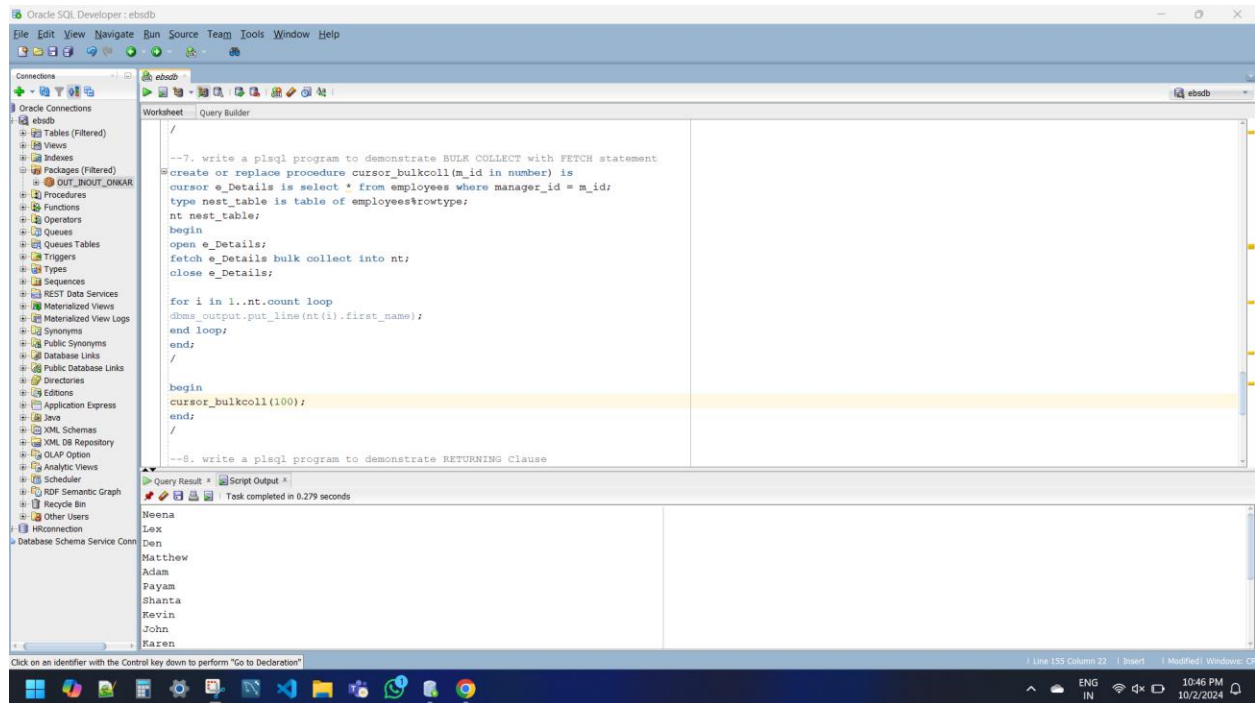
```
/
```

```
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;

/

