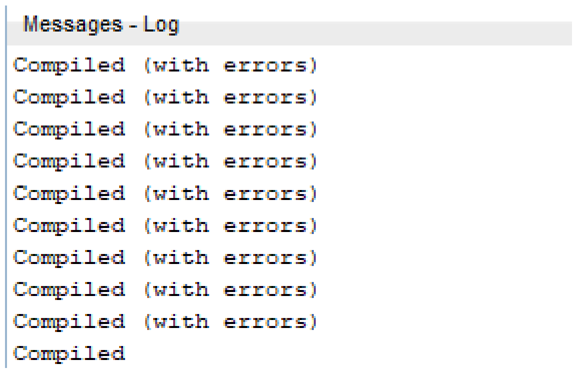
--- packages in plsqql

syntax:

```
create or replace package emppack as

--- stataements:

end emppack;
```

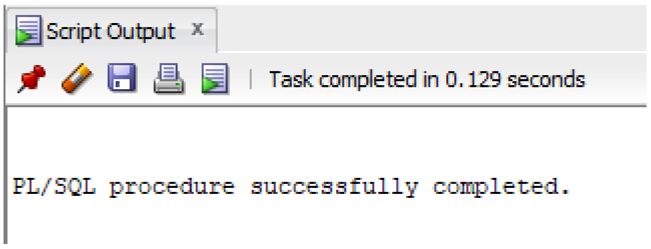
--- package emp

```
create or replace package emppack as

v_sal_inc_rate number:=0.05;
cursor c_emp is select * from emp_info;
procedure inc_sal;
function get_avg_sal(dept_id int) return number;

end emppack;
CREATE OR REPLACE
PACKAGE BODY EMPPACK AS

  procedure inc_sal AS
  BEGIN
    for r1 in c_emp loop
    update emp_info set salary=salary + salary*v_sal_inc_rate;
    end loop;
  END inc_sal;

  function get_avg_sal(dept_id int) return number AS
  v_avg_sal number:=0;
  BEGIN
    select avg(salary) into v_avg_sal from emp_info where department_id = dept_id;
    return v_avg_sal;
  END get_avg_sal;

END EMPPACK;
```



```
Compiled (with errors)
Compiled (with errors)
Compiled (with errors)
Compiled (with errors)
Compiled (with errors)
Compiled (with errors)
Compiled (with errors)
Compiled (with errors)
Compiled (with errors)
Compiled
```

--- to implement package
exec emppack.inc_sal;



```
PL/SQL procedure successfully completed.
```

```
begin
dbms_output.put_line('Increment and Average salaries are: ');
dbms_output.put_line(emppack.get_avg_sal(30));
dbms_output.put_line(emppack.v_sal_inc_rate);
end;
```

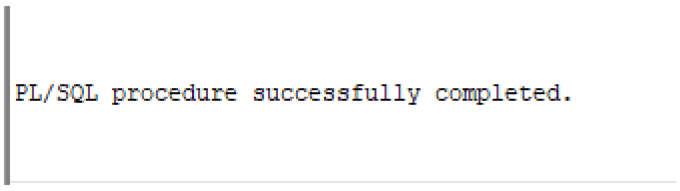
```
PL/SQL procedure successfully completed.
```

To drop the package
Goto package -> right click on package-> select drop package-> select ok.

--- visibiity of packages

```
create package emp_pack2 as

v_sal_inc_rate number:=1000;
cursor c_emp is select * from emp_info;

procedure inc_sal;
function get_avg_sal(dept_id int) return number;

end emp_pack2;


CREATE OR REPLACE
PACKAGE BODY EMP_PACK2 AS

 v_sal_inc int:=500;

  procedure print_test as
  begin
  dbms_output.put_line('Test: '||v_sal_inc);
  end;

  procedure inc_sal AS
  BEGIN
```

```
    for r1 in c_emp loop
    update emp_info set salary=salary+salary*v_sal_inc_rate where
employee_id=r1.employee_id;
    end loop;
  END inc_sal;

  function get_avg_sal(dept_id int) return number AS
  v_avg_sal number:=0;
  BEGIN
   print_test;
    select avg(salary) into v_avg_sal from emp_info where department_id=dept_id;
    return v_avg_sal;
  END get_avg_sal;

END EMP_PACK2;
begin
dbms_output.put_line(emp_pack2.get_avg_sal(50));
end;
```

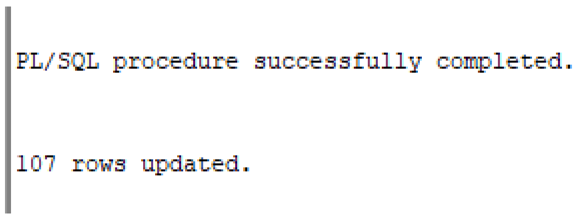


```
PL/SQL procedure successfully completed.
```

```
create table log_info (log_source varchar(20),log_msg varchar(20),log_date date);
Table created


  begin
  insert into log_info values('EMP_PKG','Package Initialized',sysdate);
Package initialized in table


exec dbms_output.put_line(emp_pack2.get_avg_sal(80));
```

PL/SQL Triggers:

```
CREATE OR REPLACE TRIGGER FIRST_TRIGGER
BEFORE INSERT OR UPDATE ON EMP_INFO
REFERENCING OLD AS XX NEW AS YY
BEGIN
  NULL;
END;
--- triggers

update emp_info set salary=salary+1000;
```



```
PL/SQL procedure successfully completed.


107 rows updated.
```

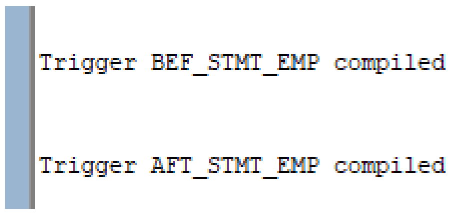
Statement level and Row Level Triggers:

--- triggers statement before

```
create trigger bef_stmt_emp before insert or update on emp_info
begin
dbms_output.put_line('Before Statement Trigger is Fired!...');
end;
```

--- triggers statement after

```
create trigger aft_stmt_emp after insert or update on emp_info
begin
dbms_output.put_line('After Statement Trigger is Fired!...');
end;
```



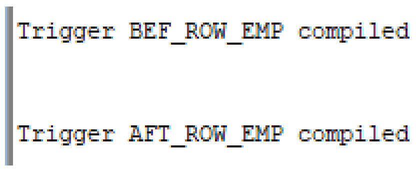
```
Trigger BEF_STMT_EMP compiled


Trigger AFT_STMT_EMP compiled
```

--- triggers row before

```
create trigger bef_row_emp before insert or update on emp_info
for each row
begin
dbms_output.put_line('Before Row Trigger is Fired!...');
end;
```

--- triggers row after

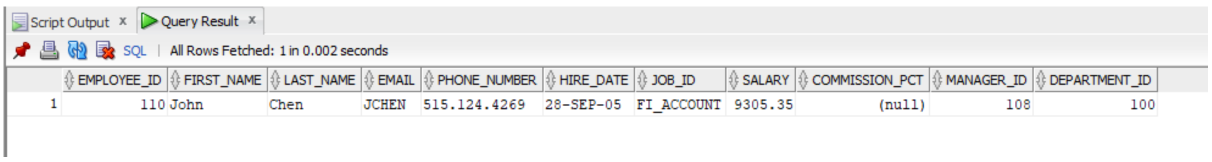
```
create trigger aft_row_emp after insert or update on emp_info
for each row
begin
dbms_output.put_line('After Row Trigger is Fired!...');
end;
```



```
Trigger BEF_ROW_EMP compiled


Trigger AFT_ROW_EMP compiled
```

update emp_info set salary=salary+100 where employee_id=110;

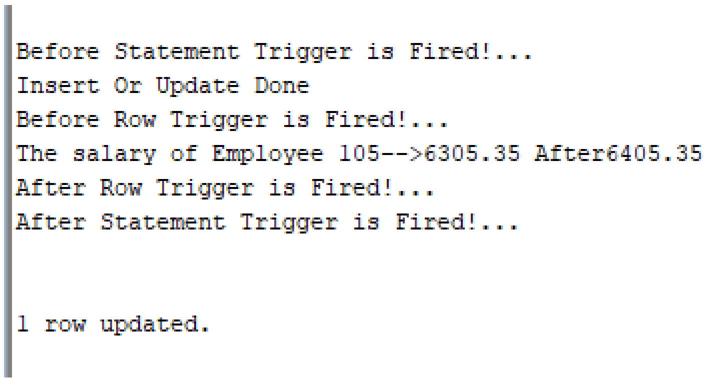
select * from emp_info where employee_id=110;



	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	110	John	Chen	JCHEN	515.124.4269	28-SEP-05	FI_ACCOUNT	9305.35	(null)	108	100

NEW and OLD Qualifiers in Triggers:

create or replace NONEDITIONABLE trigger bef_row_emp before insert or update on emp_info
for each row
begin
dbms_output.put_line('Before Row Trigger is Fired!...');
dbms_output.put_line('The salary of Employee '||:old.employee_id||'-->'||:old.salary||' After'||:new.salary);
end;



```
Before Statement Trigger is Fired!...
Insert Or Update Done
Before Row Trigger is Fired!...
The salary of Employee 105-->6305.35 After6405.35
After Row Trigger is Fired!...
After Statement Trigger is Fired!...


1 row updated.
```

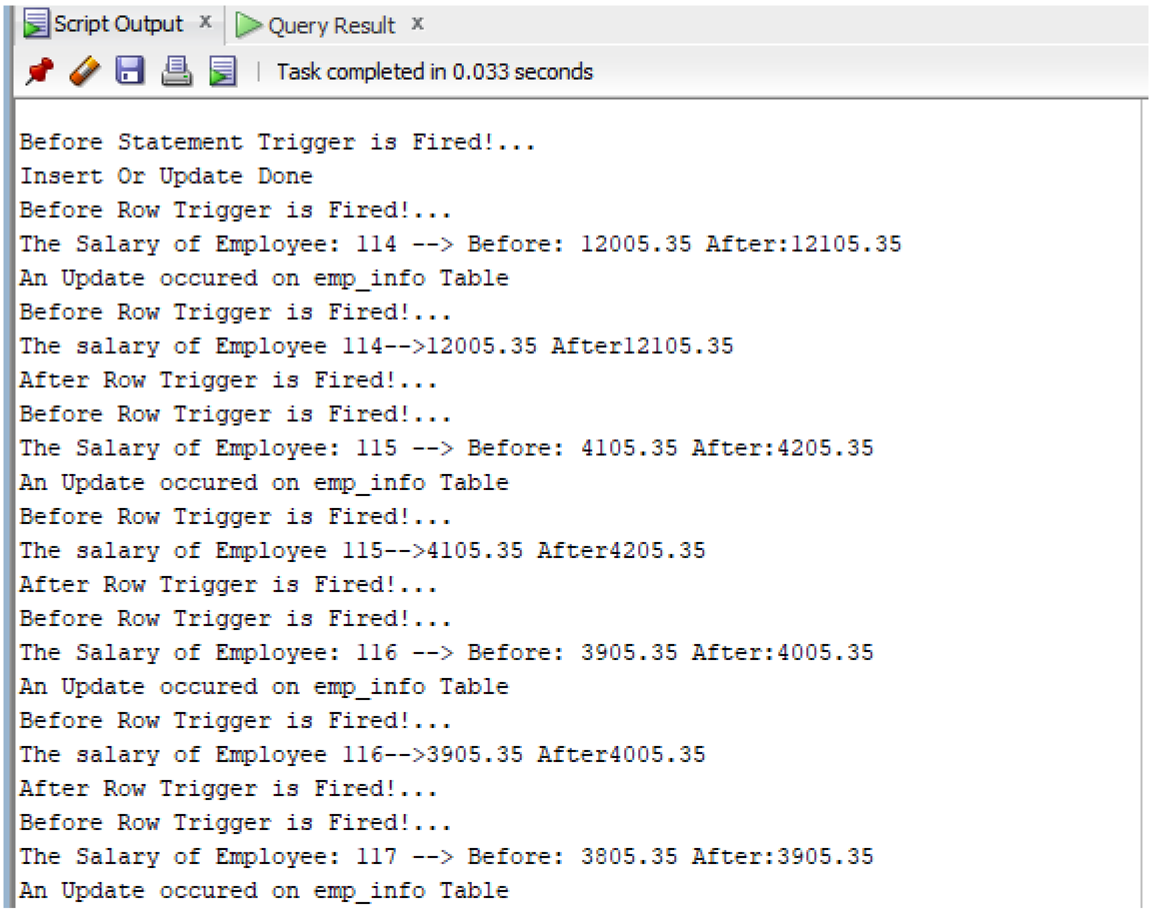
SET SERVEROUTPUT ON;

--- update emp_info set salary=salary+100 where employee_id=105;

create or replace trigger bef_row_emp_cpy
before insert or update or delete on emp_info referencing old as o new as n
for each row
begin
dbms_output.put_line('Before Row Trigger is Fired!...');
dbms_output.put_line('The Salary of Employee: '||:o.employee_id||' --> Before: '||:o.salary||' After:'||:n.salary);

if inserting then

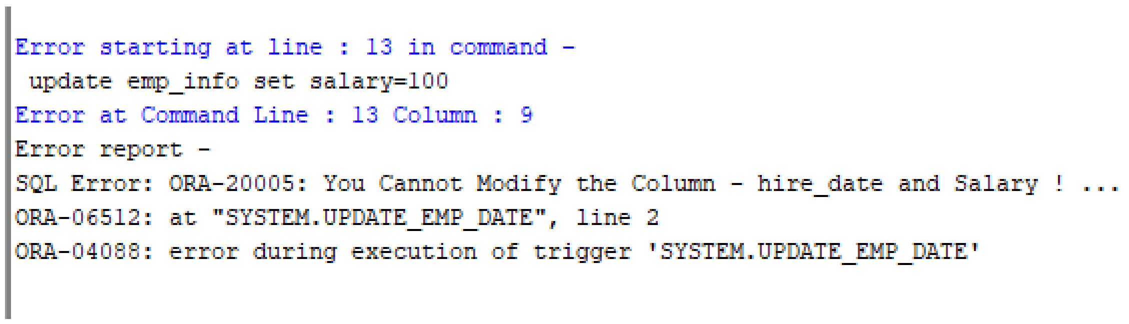
```
dbms_output.put_line('An Insert occured on emp_info Table');
elsif updating then
dbms_output.put_line('An Update occured on emp_info Table');
elsif deleting then
dbms_output.put_line('An Delete occured on emp_info Table');
end if;
end;
```



```
Before Statement Trigger is Fired!...
Insert Or Update Done
Before Row Trigger is Fired!...
The Salary of Employee: 114 --> Before: 12005.35 After:12105.35
An Update occured on emp_info Table
Before Row Trigger is Fired!...
The salary of Employee 114-->12005.35 After12105.35
After Row Trigger is Fired!...
Before Row Trigger is Fired!...
The Salary of Employee: 115 --> Before: 4105.35 After:4205.35
An Update occured on emp_info Table
Before Row Trigger is Fired!...
The salary of Employee 115-->4105.35 After4205.35
After Row Trigger is Fired!...
Before Row Trigger is Fired!...
The Salary of Employee: 116 --> Before: 3905.35 After:4005.35
An Update occured on emp_info Table
Before Row Trigger is Fired!...
The salary of Employee 116-->3905.35 After4005.35
After Row Trigger is Fired!...
Before Row Trigger is Fired!...
The Salary of Employee: 117 --> Before: 3805.35 After:3905.35
An Update occured on emp_info Table
```

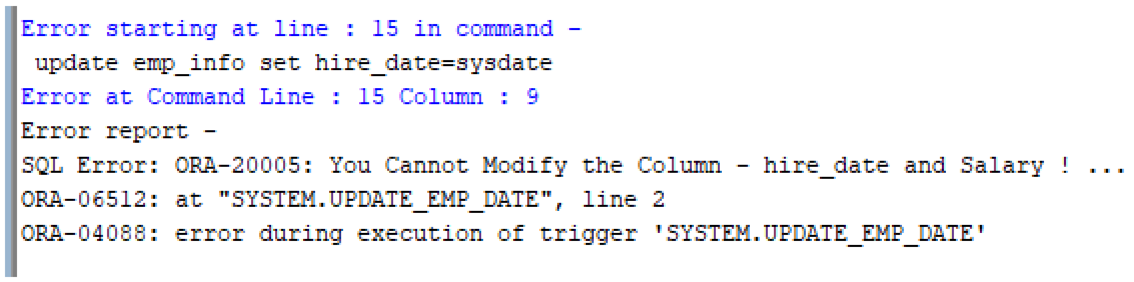
For this each row and column is updated by triggers condition;

```
--- update event on triggers

create or replace trigger update_emp_date before update of hire_date,salary on emp_info
for each row
begin
raise_application_error(-20005,'You Cannot Modify the Column - hire_date and Salary ! ...');
End;

 update emp_info set salary=100;
```



```
Error starting at line : 13 in command -
 update emp_info set salary=100
Error at Command Line : 13 Column : 9
Error report -
SQL Error: ORA-20005: You Cannot Modify the Column - hire_date and Salary ! ...
ORA-06512: at "SYSTEM.UPDATE_EMP_DATE", line 2
ORA-04088: error during execution of trigger 'SYSTEM.UPDATE_EMP_DATE'
```

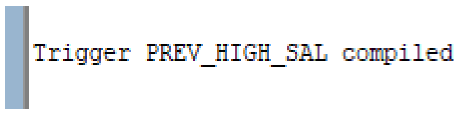
update emp_info set hire_date=sysdate;



```
Error starting at line : 15 in command -
 update emp_info set hire_date=sysdate
Error at Command Line : 15 Column : 9
Error report -
SQL Error: ORA-20005: You Cannot Modify the Column - hire_date and Salary ! ...
ORA-06512: at "SYSTEM.UPDATE_EMP_DATE", line 2
ORA-04088: error during execution of trigger 'SYSTEM.UPDATE_EMP_DATE'
```

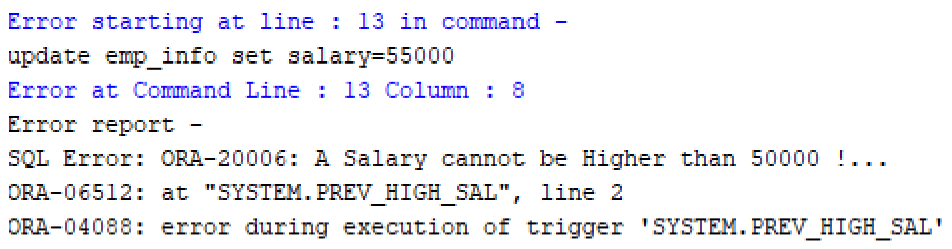
--- to use when clause in triggers

```
create trigger prev_high_sal before insert or update or delete of salary on emp_info
for each row
when(new.salary>50000)
begin
raise_application_error(-20006,'A Salary cannot be Higher than 50000 !...');
end;
```



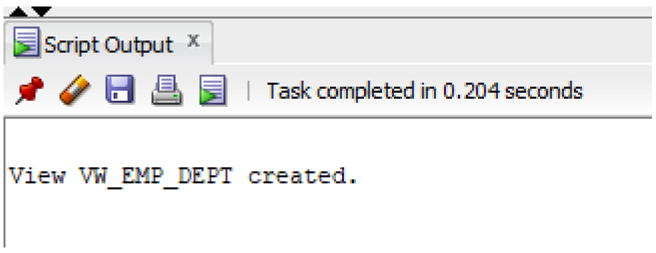
```
Trigger PREV_HIGH_SAL compiled
```

update emp_info set salary=55000;



```
Error starting at line : 13 in command -
update emp_info set salary=55000
Error at Command Line : 13 Column : 8
Error report -
SQL Error: ORA-20006: A Salary cannot be Higher than 50000 !...
ORA-06512: at "SYSTEM.PREV_HIGH_SAL", line 2
ORA-04088: error during execution of trigger 'SYSTEM.PREV_HIGH_SAL'
```

```
create view vw_emp_dept as
select upper(department_name) dname,min(salary) min_sal from emp_info join dept
using(department_id) group by department_name;
```



```
View VW_EMP_DEPT created.
```

Data manipulation cannot be done in views

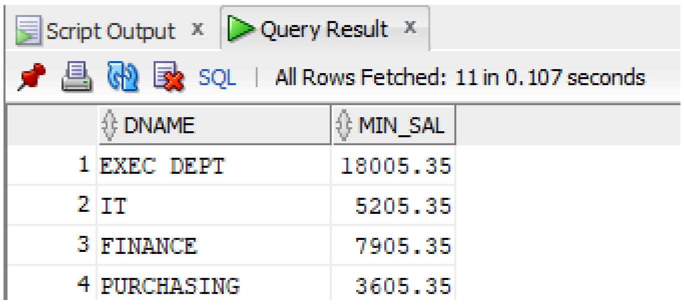
```
create or replace trigger emp_vw instead of insert or update or delete on vw_emp_dept
for each row
declare
v_dept_id pls_integer;
begin
if inserting then
select max(department_id)+10 into v_dept_id from dept;
insert into dept values(v_dept_id,:new.dname,null,null);
elsif deleting then
delete from dept where upper(department_name)=upper(:old.dname);
elsif updating('dname') then
update dept set department_name=:new.dname where
upper(department_name)=upper(:old.dname);
else
raise_application_error(-20007,'You cannot update data!....');
end if;
end;
```

```
Trigger EMP_VW compiled
```

```
update vw_emp_dept set dname='EXEC DEPT' where upper(dname)='EXECUTIVE';
```

```
1 row updated.
```

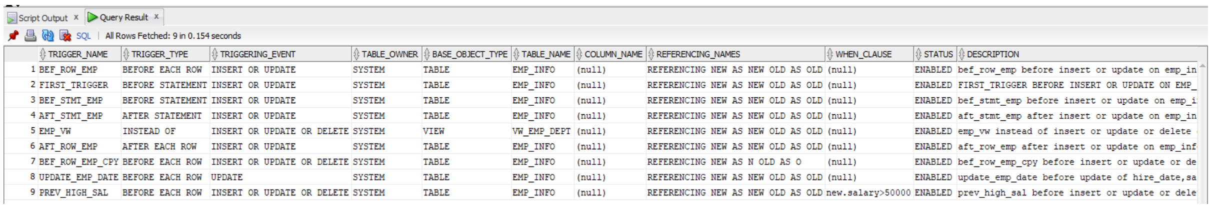
```
select * from vw_emp_dept;
```



	DNAME	MIN_SAL
1	EXEC DEPT	18005.35
2	IT	5205.35
3	FINANCE	7905.35
4	PURCHASING	3605.35

Like this dml ops are done;

```
select * from user_triggers;
```



	TRIGGER_NAME	TRIGGER_TYPE	TRIGGERING_EVENT	TABLE_OWNER	BASE_OBJECT_TYPE	TABLE_NAME	COLUMN_NAME	REFERENCING_NAMES	WHEN_CLAUSE	STATUS	DESCRIPTION
1	BEF_ROW_EMP	BEFORE EACH ROW	INSERT OR UPDATE	SYSTEM	TABLE	EMP_INFO	(null)	REFERENCING NEW AS NEW OLD AS OLD	(null)	ENABLED	bef_row_emp before insert or update on emp_in
2	FIRST_TRIGGER	BEFORE STATEMENT	INSERT OR UPDATE	SYSTEM	TABLE	EMP_INFO	(null)	REFERENCING NEW AS NEW OLD AS OLD	(null)	ENABLED	FIRST_TRIGGER BEFORE INSERT OR UPDATE ON EMP_
3	BEF_STMT_EMP	BEFORE STATEMENT	INSERT OR UPDATE	SYSTEM	TABLE	EMP_INFO	(null)	REFERENCING NEW AS NEW OLD AS OLD	(null)	ENABLED	bef_stmt_emp before insert or update on emp_i
4	AFT_STMT_EMP	AFTER STATEMENT	INSERT OR UPDATE	SYSTEM	TABLE	EMP_INFO	(null)	REFERENCING NEW AS NEW OLD AS OLD	(null)	ENABLED	aft_stmt_emp after insert or update on emp_in
5	EMP_VW	INSTEAD OF	INSERT OR UPDATE OR DELETE	SYSTEM	VIEW	VW_EMP_DEPT	(null)	REFERENCING NEW AS NEW OLD AS OLD	(null)	ENABLED	emp_vw instead of insert or update or delete
6	AFT_ROW_EMP	AFTER EACH ROW	INSERT OR UPDATE	SYSTEM	TABLE	EMP_INFO	(null)	REFERENCING NEW AS NEW OLD AS OLD	(null)	ENABLED	aft_row_emp after insert or update on emp_inf
7	BEF_ROW_EMP_CPY	BEFORE EACH ROW	INSERT OR UPDATE OR DELETE	SYSTEM	TABLE	EMP_INFO	(null)	REFERENCING NEW AS N OLD AS O	(null)	ENABLED	bef_row_emp_cpy before insert or update or de
8	UPDATE_EMP_DATE	BEFORE EACH ROW	UPDATE	SYSTEM	TABLE	EMP_INFO	(null)	REFERENCING NEW AS NEW OLD AS OLD	(null)	ENABLED	update_emp_date before update of hire_date,sa
9	PREV_HIGH_SAL	BEFORE EACH ROW	INSERT OR UPDATE OR DELETE	SYSTEM	TABLE	EMP_INFO	(null)	REFERENCING NEW AS NEW OLD AS OLD	new.salary>50000	ENABLED	prev_high_sal before insert or update or dele

To drop the triggers

Drop trigger trig_name;

Compound triggers:
To do multiple operations in single trigger name;

--- compund triggers

create trigger tr_emp_dept
for insert or update or delete on emp_info
compound trigger
v_dml_type varchar(30);

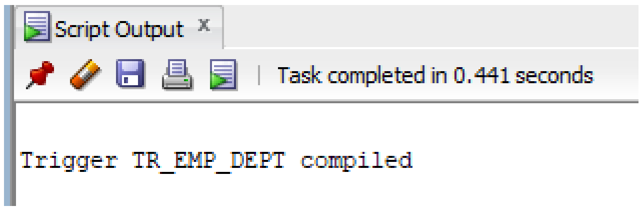
before statement is
begin
if inserting then
v_dml_type:='Insert';
elsif updating then
v_dml_type:='Update';
elsif deleting then
v_dml_type:='Delete';
end if;
dbms_output.put_line('Before Statement section is executed: '||v_dml_type||' event');
end before statement;

before each row is
x number;
begin
dbms_output.put_line('Before Row section is executed: '||v_dml_type||' event');
end before each row;

after each row is
begin
dbms_output.put_line('After Row section is executed: '||v_dml_type||' event');
end after each row;

after statement is
begin

```
dbms_output.put_line('After Statement section is executed: '||v_dml_type||' event');
end after statement;
end;
```



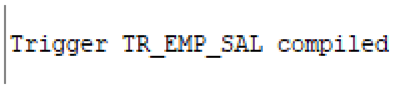
Script Output

Task completed in 0.441 seconds

```
Trigger TR_EMP_DEPT compiled
```

```
--- compund triggers

create trigger tr_emp_sal
for insert or update or delete on emp_info
compound trigger
type t_avg_sal is table of emp_info.salary%type index by pls_integer;
avg_dept_sal t_avg_sal;

before statement is
begin
for avg_sal in (select avg(salary) SALARY,nvl(department_id,999) DEPARTMENT_ID from
emp_info group by department_id) loop
avg_dept_sal(avg_sal.department_id):=avg_sal.salary;
end loop;
end before statement;

/*before each row is
x number;
begin
dbms_output.put_line('Before Row section is executed: '||v_dml_type||' event');
end before each row;*/


after each row is
v_int number:=15;

begin

if :new.salary>avg_dept_sal(:new.department_id)*v_int/100 then
raise_application_error(-20005,'A raise cannot be '||v_int||' Percent higher than its
department average');
end if;
end after each row;


after statement is
begin
dbms_output.put_line('All the changes are done successfully!...');
end after statement;
```

end;



```
Trigger TR_EMP_SAL compiled
```

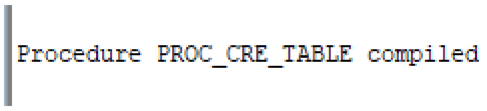
It will do all operations done as given.

Dynamic SQL

Immediate execution Statement;

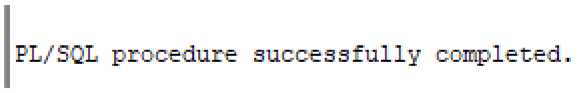
--- dynamic sql

```
create procedure proc_cre_table(tname in varchar,tcol in varchar) is
begin
execute immediate 'create table '||tname||' ('||tcol||')';
end;
```



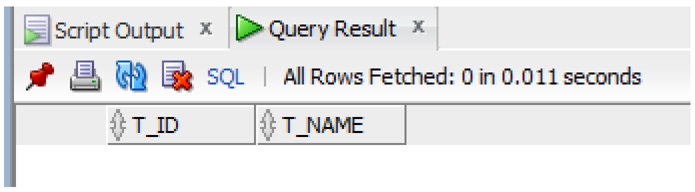
```
Procedure PROC_CRE_TABLE compiled
```

exec proc_cre_table('proc_table','t_id number primary key,t_name varchar(30)');



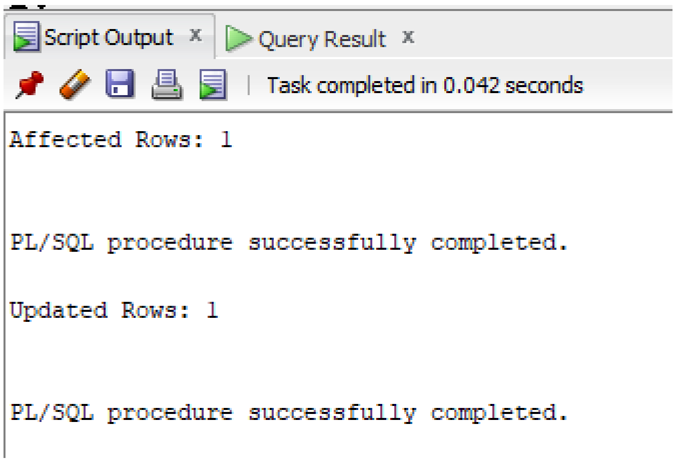
```
PL/SQL procedure successfully completed.
```

select * from proc_table;



```
All Rows Fetched: 0 in 0.011 seconds
        T_ID    T_NAME
```

```
declare
v_aff_rows number;
begin
v_aff_rows :=ins_val(102,'Pradeep');
dbms_output.put_line('Updated Rows: '||v_aff_rows);
end;
```

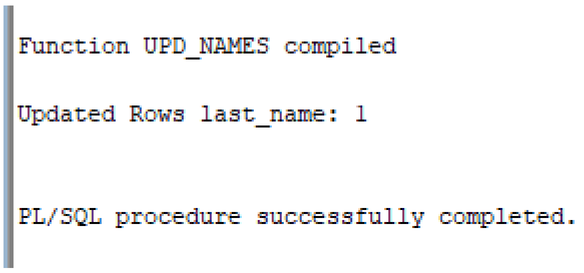


```
Affected Rows: 1


PL/SQL procedure successfully completed.

Updated Rows: 1


PL/SQL procedure successfully completed.
```

alter table names add (lname varchar(20));

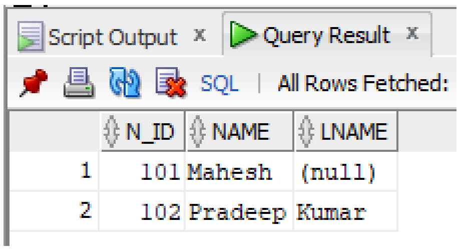
Added one more column

```
declare
v_aff_rows number;
begin
v_aff_rows :=upd_names(102,'Kumar');
dbms_output.put_line('Updated Rows last_name: '||v_aff_rows);
end;
```



```
Function UPD_NAMES compiled

Updated Rows last_name: 1


PL/SQL procedure successfully completed.
```

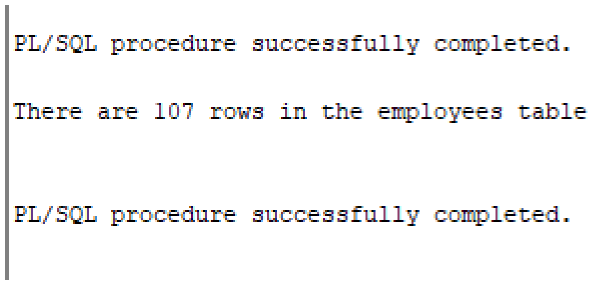
Select * from names;



	N_ID	NAME	LNAME
1	101	Mahesh	(null)
2	102	Pradeep	Kumar

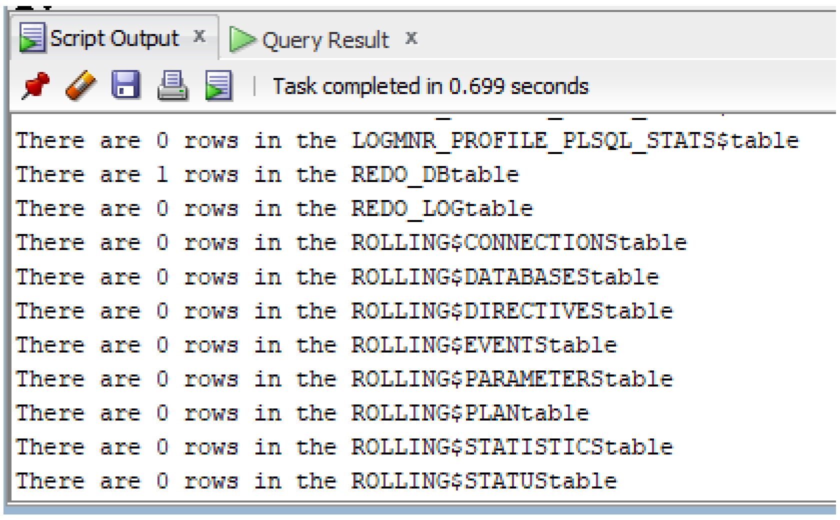
Like wise all the operations are done

```
--- into clause

create function count_row(tname in varchar) return pls_integer is
v_count number;
begin
execute immediate 'select count(*) from '||tname into v_count;
return v_count;
End;

begin
dbms_output.put_line('There are '||count_row('emp_info')||' rows in the employees table');
end;
```



```
PL/SQL procedure successfully completed.

There are 107 rows in the employees table


PL/SQL procedure successfully completed.
```

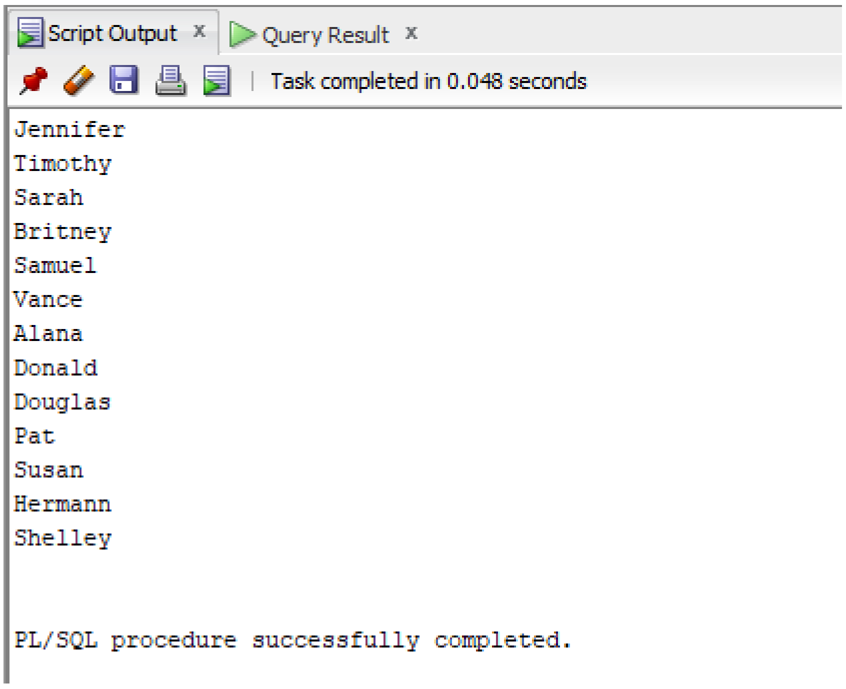
```
— to get how many tables in schema
declare
tabl_name varchar(50);
begin
for r in (select table_name from user_tables) loop
dbms_output.put_line('There are '||count_row(r.table_name)||' rows in the
'||r.table_name||'table');
end loop;
end;
```



```
There are 0 rows in the LOGMNR_PROFILE_PLSQL_STATS$table
There are 1 rows in the REDO_DBtable
There are 0 rows in the REDO_LOGtable
There are 0 rows in the ROLLING$CONNECTIONStable
There are 0 rows in the ROLLING$DATABASEStable
There are 0 rows in the ROLLING$DIRECTIVEStable
There are 0 rows in the ROLLING$EVENTStable
There are 0 rows in the ROLLING$PARAMETERStable
There are 0 rows in the ROLLING$PLANtable
There are 0 rows in the ROLLING$STATISTICStable
There are 0 rows in the ROLLING$STATUStable
```

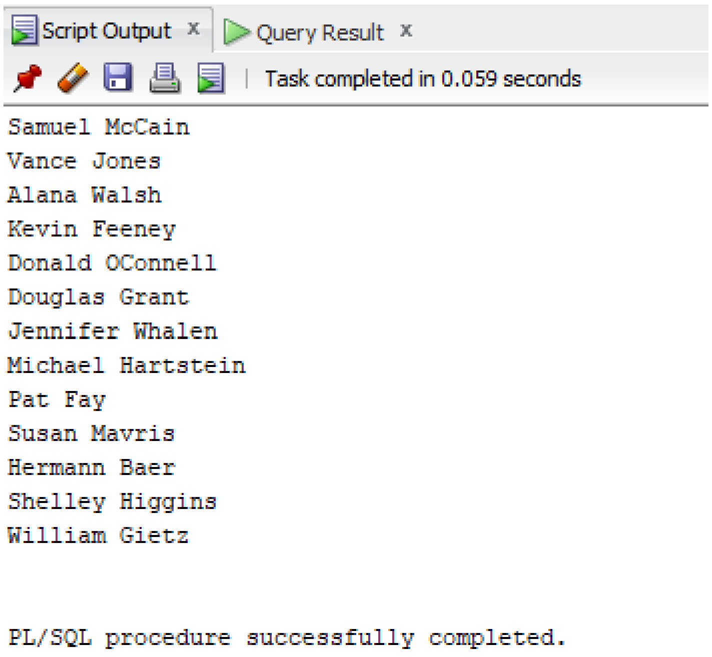
```
--- bulk collection into

declare
type t_name is table of varchar(20);
```

```
name t_name;
begin
execute immediate 'select distinct first_name from emp_info' bulk collect into name;
for i in 1..name.count loop
dbms_output.put_line(name(i));
end loop;
end;
```



```
Jennifer
Timothy
Sarah
Britney
Samuel
Vance
Alana
Donald
Douglas
Pat
Susan
Hermann
Shelley


PL/SQL procedure successfully completed.
```

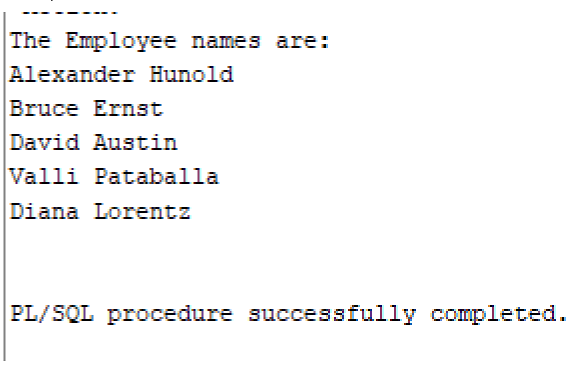
```
--- block in plsql

declare
v_dy_name varchar(1000);
begin

v_dy_name:=q'[
begin
dbms_output.put_line('The Employee names are: ');
for i in (select * from emp_info) loop
dbms_output.put_line(i.first_name||' '||i.last_name);
end loop;
end;
]';

execute immediate v_dy_name;
end;
```



```
Samuel McCain
Vance Jones
Alana Walsh
Kevin Feeney
Donald OConnell
Douglas Grant
Jennifer Whalen
Michael Hartstein
Pat Fay
Susan Mavris
Hermann Baer
Shelley Higgins
William Gietz


PL/SQL procedure successfully completed.
```

--- block using keyword in plsql

```
declare
v_dy_name varchar(1000);
v_dept_id number:=60;
begin

v_dy_name:=q'[
begin
dbms_output.put_line('The Employee names are: ');
for i in (select * from emp_info where department_id=:1) loop
dbms_output.put_line(i.first_name||' '||i.last_name);
end loop;
end;
]';

execute immediate v_dy_name using v_dept_id;
end;
```



```
---------
The Employee names are:
Alexander Hunold
Bruce Ernst
David Austin
Valli Pataballa
Diana Lorentz


PL/SQL procedure successfully completed.
```

```
declare
  type emp_cur_type  is ref cursor;
  emp_cursor      emp_cur_type;
  emp_record      emp_info%rowtype;
begin
  open emp_cursor for 'SELECT * FROM emp_info WHERE job_id = :job' using 'IT_PROG';
    fetch emp_cursor into emp_record;
    dbms_output.put_line(emp_record.first_name||emp_record.last_name);
  close emp_cursor;
end;
```

```
AlexanderHunold


PL/SQL procedure successfully completed.
```

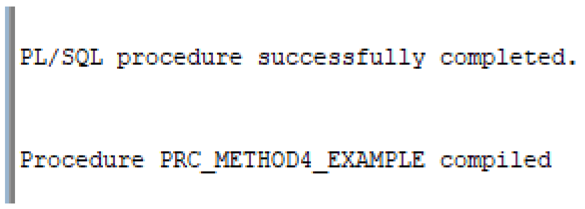
```
declare
  type emp_cur_type  is ref cursor;
  emp_cursor      emp_cur_type;
  emp_record      emp_info%rowtype;
  v_table_name    varchar(20);
begin
  v_table_name := 'emp_info';
  open emp_cursor for 'SELECT * FROM '||v_table_name||' WHERE job_id = :job' using
'IT_PROG';
  loop
    fetch emp_cursor into emp_record;
    exit when emp_cursor%notfound;
    dbms_output.put_line(emp_record.first_name||emp_record.last_name);
  end loop;
  close emp_cursor;
end;
```

```
AlexanderHunold
BruceErnst
DavidAustin
ValliPataballa
DianaLorentz


PL/SQL procedure successfully completed.
```

DBMS Package :

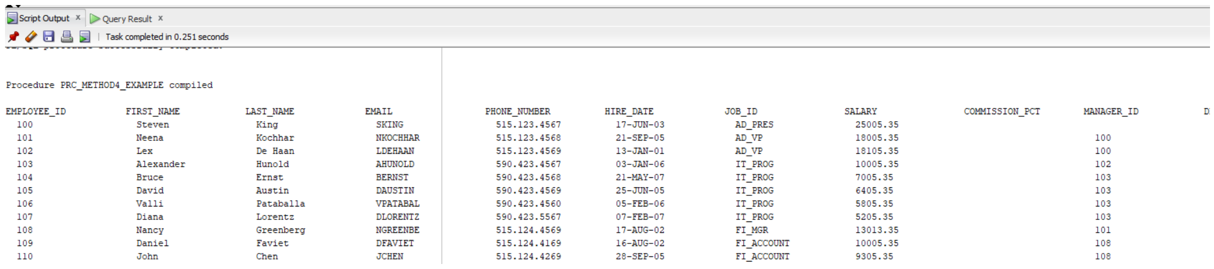
```
create or replace procedure prc_method4_example (p_table_name in varchar2) is
    type t_columns is table of user_tab_columns%rowtype index by pls_integer;
    v_columns              t_columns;
    v_columns_with_commas   varchar2(32767);
    v_number_value          number;
    v_string_value          varchar2(32767);
    v_date_value            date;
    v_output_string         varchar2(32767);
    cur_dynamic             integer;
begin
```

```
   select * bulk collect into v_columns from user_tab_columns where table_name =
upper(p_table_name);
   v_columns_with_commas:=v_columns(1).column_name;
   for i in 2..v_columns.count loop
      v_columns_with_commas:=v_columns_with_commas||','||v_columns(i).column_name;
   end loop;
   cur_dynamic := dbms_sql.open_cursor;
   dbms_sql.parse(cur_dynamic,'SELECT '||v_columns_with_commas||' FROM
'||p_table_name,dbms_sql.native);
    for idx in 1..v_columns.count loop
      if v_columns(idx).data_type = 'NUMBER' then
         dbms_sql.define_column(cur_dynamic,idx,1);
      elsif v_columns(idx).data_type in ('VARCHAR2','VARCHAR','CHAR') then
         dbms_sql.define_column(cur_dynamic,idx,'dummy text',v_columns(idx).char_length);
      elsif v_columns(idx).data_type = 'DATE' then
         dbms_sql.define_column(cur_dynamic,idx,sysdate);
      end if;
      v_output_string:=v_output_string||'  '||rpad(v_columns(idx).column_name,20);
    end loop;
    dbms_output.put_line(v_output_string);
    v_number_value:=dbms_sql.execute(cur_dynamic);
    while dbms_sql.fetch_rows(cur_dynamic) > 0 loop
      v_output_string:=null;
      for t in 1..v_columns.count loop
         if v_columns(t).data_type = 'NUMBER' then
            dbms_sql.column_value(cur_dynamic,t,v_number_value);
            v_output_string := v_output_string||'  '||rpad(nvl(to_char(v_number_value),' '),20);
         elsif v_columns(t).data_type in ('VARCHAR2','VARCHAR','CHAR') then
            dbms_sql.column_value(cur_dynamic,t,v_string_value);
            v_output_string := v_output_string||'  '||rpad(nvl(to_char(v_string_value),' '),20);
         elsif v_columns(t).data_type = 'DATE' then
            dbms_sql.column_value(cur_dynamic,t,v_date_value);
            v_output_string := v_output_string||'  '||rpad(nvl(to_char(v_date_value),' '),20);
         end if;
      end loop;
      dbms_output.put_line(v_output_string);
    end loop;
end;
```



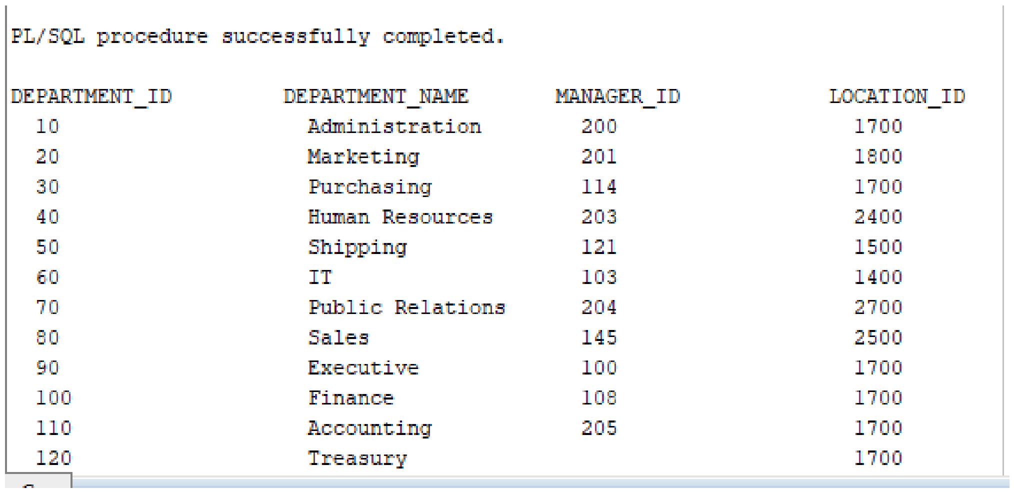
```
PL/SQL procedure successfully completed.


Procedure PRC_METHOD4_EXAMPLE compiled
```

```
EXEC prc_method4_example('emp_info');
```



```
Procedure PRC_METHOD4_EXAMPLE compiled

EMPLOYEE_ID    FIRST_NAME    LAST_NAME    EMAIL       PHONE_NUMBER   HIRE_DATE   JOB_ID       SALARY     COMMISSION_PCT   MANAGER_ID   D
  100            Steven        King         SKING       515.123.4567   17-JUN-03   AD_PRES      25005.35
  101            Neena         Kochhar      NKOCHHAR    515.123.4568   21-SEP-05   AD_VP        18005.35                    100
  102            Lex           De Haan      LDEHAAN     515.123.4569   13-JAN-01   AD_VP        18105.35                    100
  103            Alexander     Hunold       AHUNOLD     590.423.4567   03-JAN-06   IT_PROG      10005.35                    102
  104            Bruce         Ernst        BERNST      590.423.4568   21-MAY-07   IT_PROG      7005.35                     103
  105            David         Austin       DAUSTIN     590.423.4569   25-JUN-05   IT_PROG      6405.35                     103
  106            Valli         Pataballa    VPATABAL    590.423.4560   05-FEB-06   IT_PROG      5805.35                     103
  107            Diana         Lorentz      DLORENTZ    590.423.5567   07-FEB-07   IT_PROG      5205.35                     103
  108            Nancy         Greenberg    NGREENBE    515.124.4569   17-AUG-02   FI_MGR       13013.35                    101
  109            Daniel        Faviet       DFAVIET     515.124.4169   16-AUG-02   FI_ACCOUNT   10005.35                    108
  110            John          Chen         JCHEN       515.124.4269   28-SEP-05   FI_ACCOUNT   9305.35                     108
```

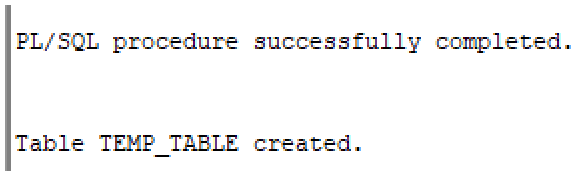
EXEC prc_method4_example('dept');



```
PL/SQL procedure successfully completed.

DEPARTMENT_ID          DEPARTMENT_NAME       MANAGER_ID            LOCATION_ID
  10                     Administration        200                   1700
  20                     Marketing             201                   1800
  30                     Purchasing            114                   1700
  40                     Human Resources       203                   2400
  50                     Shipping              121                   1500
  60                     IT                    103                   1400
  70                     Public Relations      204                   2700
  80                     Sales                 145                   2500
  90                     Executive             100                   1700
  100                    Finance               108                   1700
  110                    Accounting            205                   1700
  120                    Treasury                                    1700
```

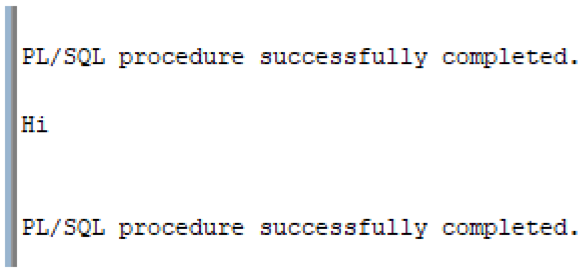
Oracle Supplied Packages:

create table temp_table(id number generated always as identity, text varchar2(1000));



```
PL/SQL procedure successfully completed.


Table TEMP_TABLE created.
```

exec dbms_output.enable;
exec dbms_output.put_line('Hi');



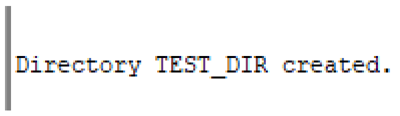
```
PL/SQL procedure successfully completed.

Hi


PL/SQL procedure successfully completed.
```

```
declare
   v_buffer varchar2(1000);
   v_status integer;
begin
   dbms_output.put('...');
   dbms_output.put_line('Hello');
   dbms_output.put_line('How are you');
```

```
    for i in 1..10 loop
        dbms_output.get_line(v_buffer,v_status);
        if v_status = 0 then
            insert into temp_table(text) values (v_buffer);
        end if;
    end loop;
end;
```
 It does not show the output

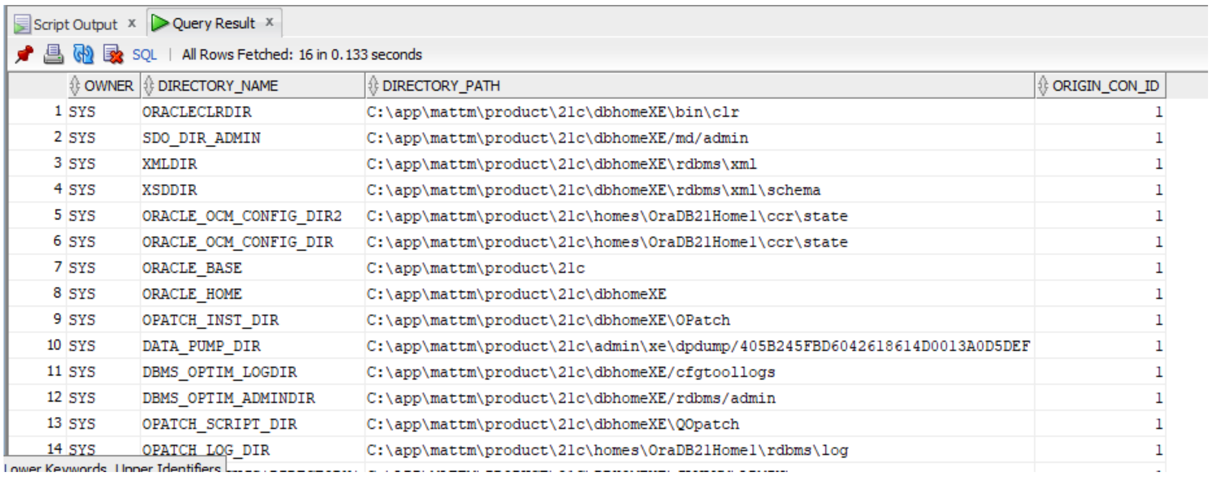
Either serveroutput on and enable buffer to view the output in dbms package.

UTL File packages
create directory test_dir as 'C:\My Folder';



```
Directory TEST_DIR created.
```

select * from ALL_DIRECTORIES;



All Rows Fetched: 16 in 0.133 seconds

	OWNER	DIRECTORY_NAME	DIRECTORY_PATH	ORIGIN_CON_ID
1	SYS	ORACLECLRDIR	C:\app\mattm\product\21c\dbhomeXE\bin\clr	1
2	SYS	SDO_DIR_ADMIN	C:\app\mattm\product\21c\dbhomeXE/md/admin	1
3	SYS	XMLDIR	C:\app\mattm\product\21c\dbhomeXE\rdbms\xml	1
4	SYS	XSDDIR	C:\app\mattm\product\21c\dbhomeXE\rdbms\xml\schema	1
5	SYS	ORACLE_OCM_CONFIG_DIR2	C:\app\mattm\product\21c\homes\OraDB21Home1\ccr\state	1
6	SYS	ORACLE_OCM_CONFIG_DIR	C:\app\mattm\product\21c\homes\OraDB21Home1\ccr\state	1
7	SYS	ORACLE_BASE	C:\app\mattm\product\21c	1
8	SYS	ORACLE_HOME	C:\app\mattm\product\21c\dbhomeXE	1
9	SYS	OPATCH_INST_DIR	C:\app\mattm\product\21c\dbhomeXE\OPatch	1
10	SYS	DATA_PUMP_DIR	C:\app\mattm\product\21c\admin\xe\dpdump/405B245FBD6042618614D0013A0D5DEF	1
11	SYS	DBMS_OPTIM_LOGDIR	C:\app\mattm\product\21c\dbhomeXE/cfgtoollogs	1
12	SYS	DBMS_OPTIM_ADMINDIR	C:\app\mattm\product\21c\dbhomeXE/rdbms/admin	1
13	SYS	OPATCH_SCRIPT_DIR	C:\app\mattm\product\21c\dbhomeXE\QOpatch	1
14	SYS	OPATCH_LOG_DIR	C:\app\mattm\product\21c\homes\OraDB21Home1\rdbms\log	1

All directories in the oracle file.

UTL Mail Packages:

```
--Sending an email with the least number of parameters
begin
    utl_mail.send(
                sender      => 'somebody@somedomain.com',
                recipients => 'oraclemaster@outlook.com',
                subject     => 'Example 1: Test Email Subject',
                message     => 'This is a test email from someone.'
               );
end;
/
```

```
--Sending an email with specific names to the sender and recipients
begin
   utl_mail.send(
               sender     => 'Some Person <somebody@somedomain.com>',
               recipients => 'Oracle Masters <oraclemaster@outlook.com>',
               subject    => 'Example 2: Test Email Subject',
               message    => 'This is a test email from someone.'
              );
end;
```