PROCEDURES

An Oracle stored procedure is a program stored in an Oracle database.

It is a compiled block of code which is stored as an object within the database.

It may or may not return any value or might return more than one value. Procedures can return data in OUT and IN OUT parameters.

We often refer to an Oracle stored procedure as a procedure.

Table Used:

Shipment2:

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

SHIPMENT_ID	CUST_ID	WEIGHT	TRUCK_ID	DESTINATION	SHIP_DATE
100	100	500	100	london	
101	101	100	102	paris	
102	101	300	103	london	
103	101	10	102	panamacity	12/12/2003
104	101	20	101	losangles	
105	102	200	102	rome	
106	100	50	101	siouxcity	9/18/2003
107	104	500	100	manhattan	
108	103	50	103	sanfransico	
109	104	25	101	sanfransico	
110	102	200	103	london	10/11/1998
111	103	100	101	london	9/9/1999
112	104	500	100	london	6/18/1988
113	104	200	100	london	10/11/1998
114	104	50	103	manhattan	9/9/1999
115	100	75	103	losangles	6/18/1988
116	101	55	102	baltimore	10/11/1998
117	103	45	101	paris	5/29/2003
118	103	45	100	rome	9/17/2002
119	103	45	102	losangles	7/1/2002
120	104	45	102	london	
121	100	150	102	siouxcity	

Num:

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

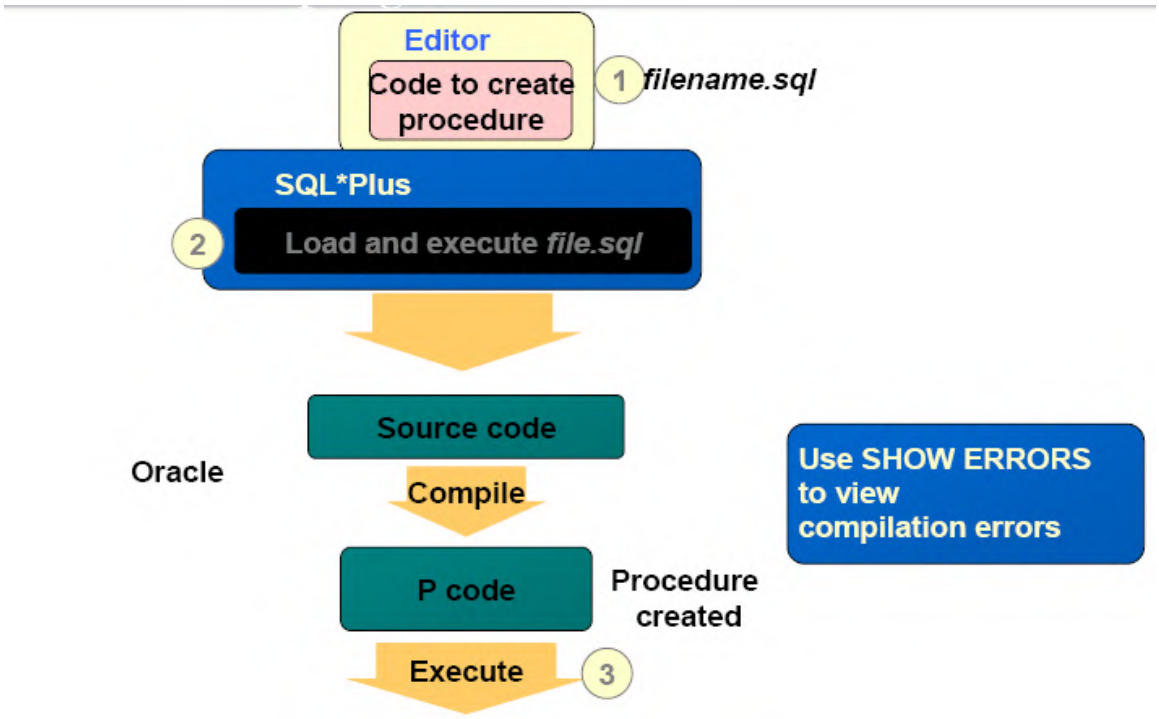
A	B
10	4
20	20
3	1
3	35
33	
55	4
2	1
100	200

Emp:

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	SEX	HEALTH2	ENERGY	NEW_COL
7001	Gilly	CRIC	100	3/31/2008 7:43:06 PM	25000	20	60	M			
8001	1.1	ABCG	10	12/4/2004	1000	40	10	F			
8000	Jspdemo	MGR	10	4/14/1998	5555	50	10	M			
7666	Ponting	TL	7000	2/14/2008 7:46:19 PM	20000	23	10	M			
123	Rahul	CLERK	7001	4/12/2007	10000	10	60	M			
7002	Test	TESTING	200	4/18/2008 4:33:20 PM	25000	20	60	M			
7521	Ward	SALESMAN	7698	4/4/2008	1250	1250	30	M			
7943	John	CLERK	7943	12/10/1983	2000	2000	50	M			
7934	Miller	CLERK	7782	1/23/1982	1300	1300	10	M			
7902	Ford	ANALYST	7566	12/3/1981	3000	3000	20	M			
7900	James	CLERK	7698	12/3/1981	769.5	769.5	30	M			
7876	Adams	CLERK	7788	3/14/1982	6000	891	30	M			
7844	Turner	SALESMAN	7698	9/8/1981	1500	1500	30	M			
7839	King	PRESIDENT	78	2/20/1988	5000	5000	20	M			
7788	Scott	ANALYST	7566	11/19/1996	3000	3000	20	M			
7782	Clark	MANAGER	7839	6/9/1981	2450	2450	10	M			
7698	Blake	MANAGER	7839	5/1/1981	2850	2850	30	M			
7654	Martin	SALESMAN	7698	9/28/1981	1250	1250	30	M			
7566	Jones	SECURITY	7839	1/23/1982	2975	2975	20	M			

DEVELOPING A PROCEDURE



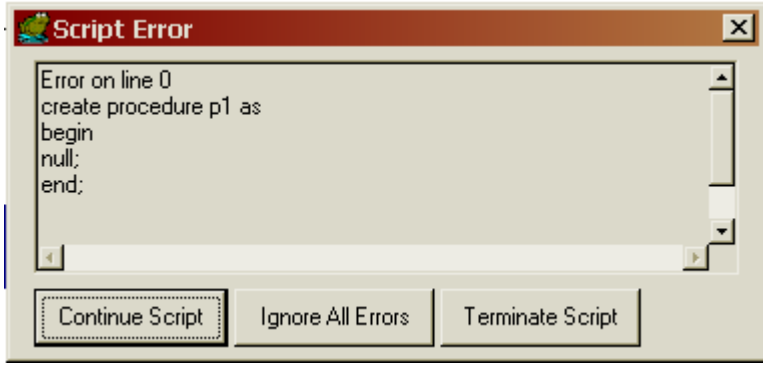
Writing a Procedure

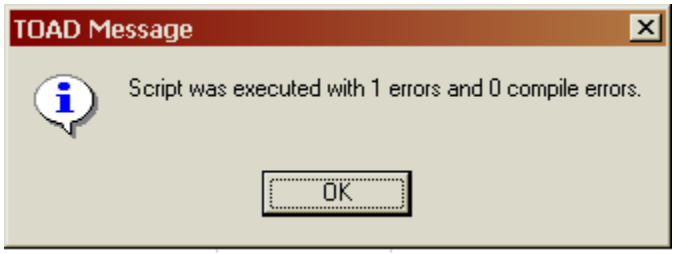
Method 1:

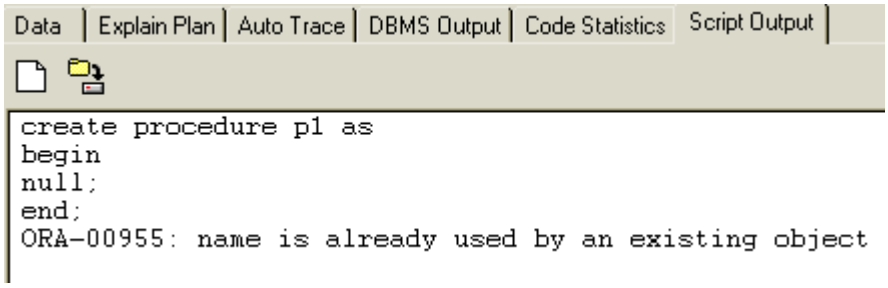
```
Create procedure Procedure_name
AS -- (or) IS

Begin
       -- body
End;
```

This Syntax is for creating a procedure.

Note:
If any procedure with the same name already exists, then it will raise an error.







Method 2:

```
create or replace procedure Procedure_name
AS -- (or) IS

Begin
       -- body
End;
```

Note:
If you use keyword **REPLACE** the exiting procedure will be replaced by new one.

Compiling a Procedure

Method 1:

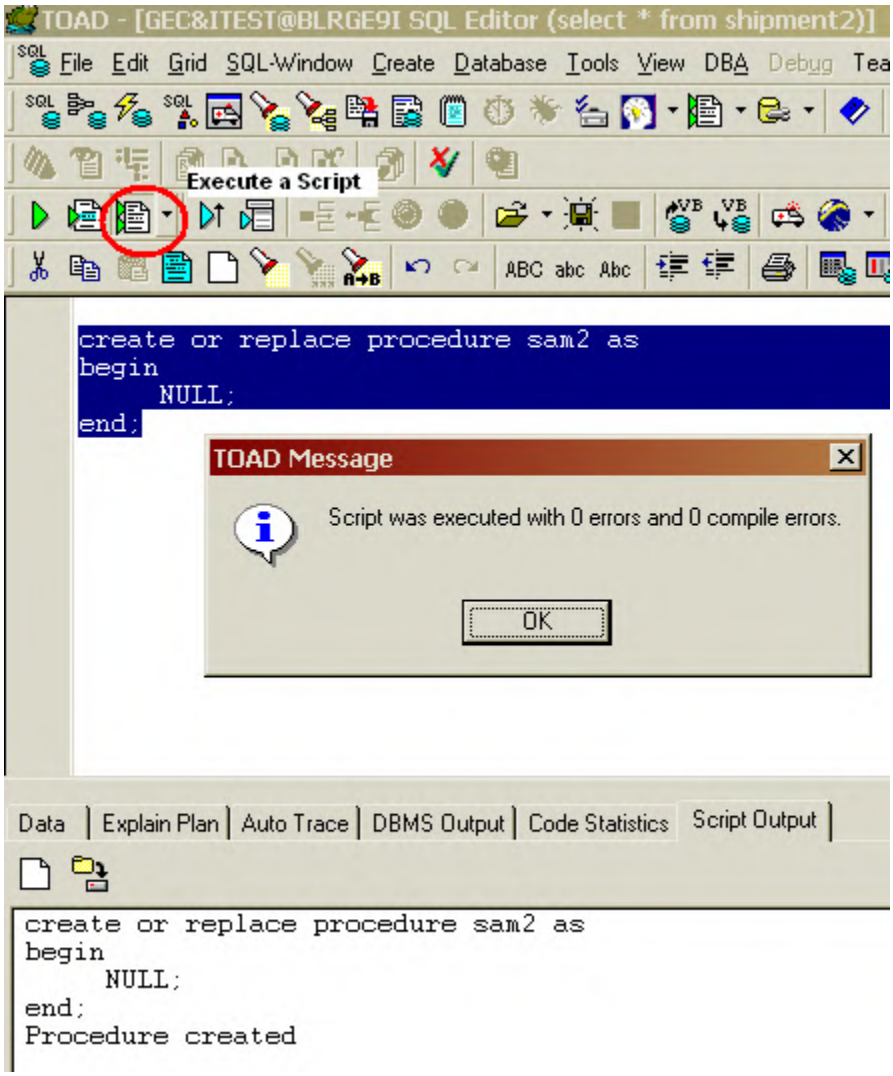
In SQL Plus:

```
SQL> Procedure
      Body

      End;

     •
SQL> Run;
```

Method 2:

```
SQL> Procedure
      Body;
      End;
      /
```

In Toad:



Executing a Procedure

Method 1:

```
Exec Procedure_name;
Exec Procedure_name(Parameter);
```

Method 2:

```
Begin
      Procedure_name;
      Procedure_name(Parameter);
End;
```

Simple Programs

For writing a procedure I am using TOAD EDITOR ….

-- Simple Procedure

```
Create or Replace Procedure Sample1
As
Begin
       Null;
End;

-- Executing Procedure Sample1
Begin
   sample1;
End;
```

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

```
Begin
   sample1;
End;
PL/SQL procedure successfully completed
```

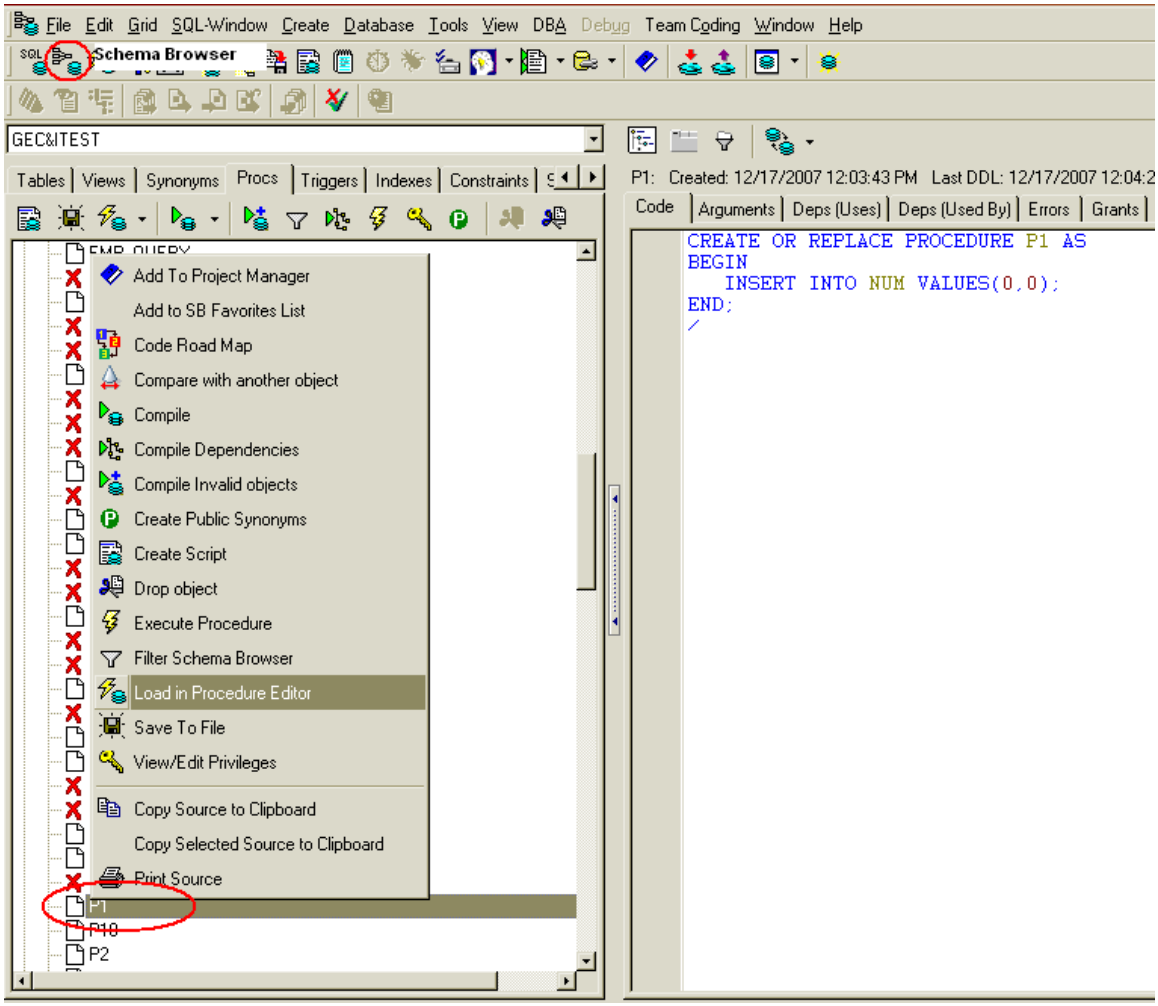
-- Simple Procedure performing DML operations

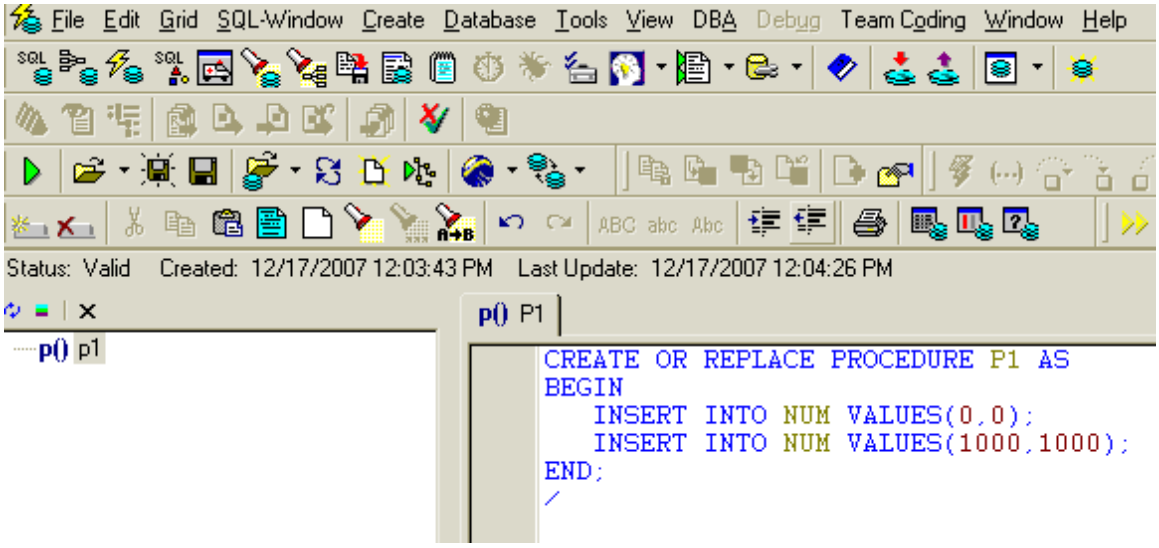
```
Create or Replace Procedure Sample2
As
Begin
       Insert into num values(122,120);
       Delete from num where a=b;
       Update num set b=Null where a=3;
End;

-- Executing Procedure Sample2

Begin
        sample2;
End;
```

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

A	B
10	4
122	120
3	
3	
33	
55	4
2	1
100	200

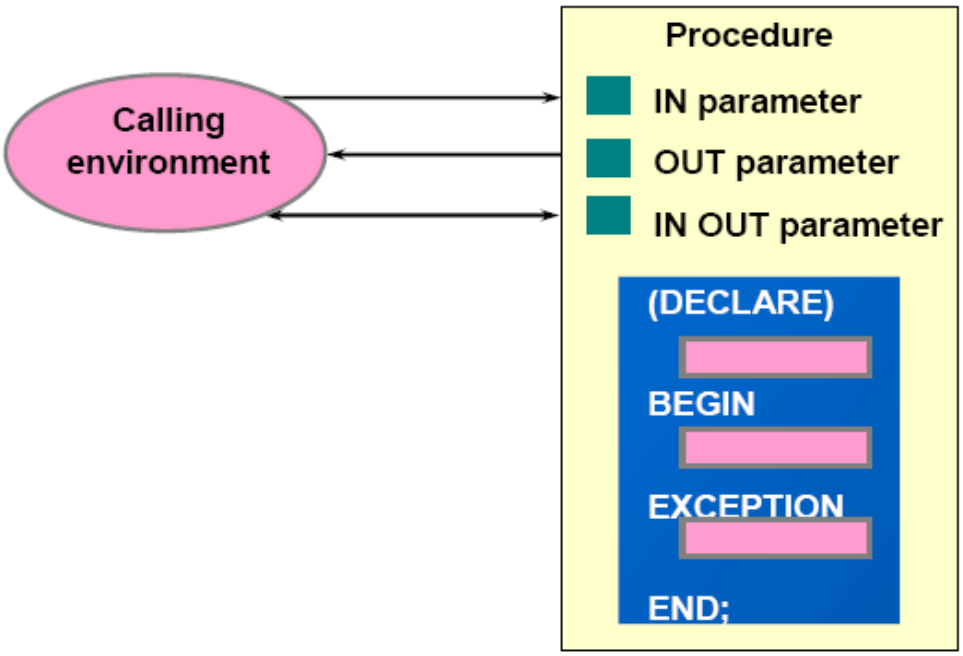
Edit a Procedure





Procedures with Parameters

Procedure Parameter Mode:



There are three types of parameter:

i. **IN**

ii. **OUT**

iii. **IN OUT**

An **IN** parameter is used an input only. An IN parameter cannot be changed by the called program.

An **OUT** parameter is initially NULL. The program assigns the parameter a value and that value is returned to the calling program.

An **IN OUT** parameter may or may not have an initial value. That initial value may or may not be modified by the called program. Any changes made to the parameter are returned to the calling program.

IN	OUT	IN OUT
Default mode	Must be specified	Must be specified
Value is passed into subprogram	Returned to calling environment	Passed into subprogram; returned to calling environment
Formal parameter acts as a constant	Uninitialized variable	Initialized variable
Actual parameter can be a literal, expression, constant, or initialized variable	Must be a variable	Must be a variable
Can be assigned a default value	Cannot be assigned a default value	Cannot be assigned a default value

-- Simple Procedure Using a Parameter

```
Create or Replace Procedure Sample3(x int, y int)
As
Begin
       insert into num values(x,y);
End;

-- Executing Procedure Sample2

Begin
       sample3 (999, 99);

End;
```

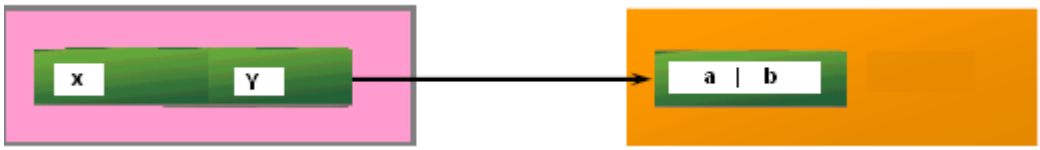
Note:

When the type of parameter is not specified then by default it will take it as "IN"

Data | Explain Plan

A	B
10	4
122	120
999	99
3	
3	
33	
55	4
2	1
100	200

IN Parameter



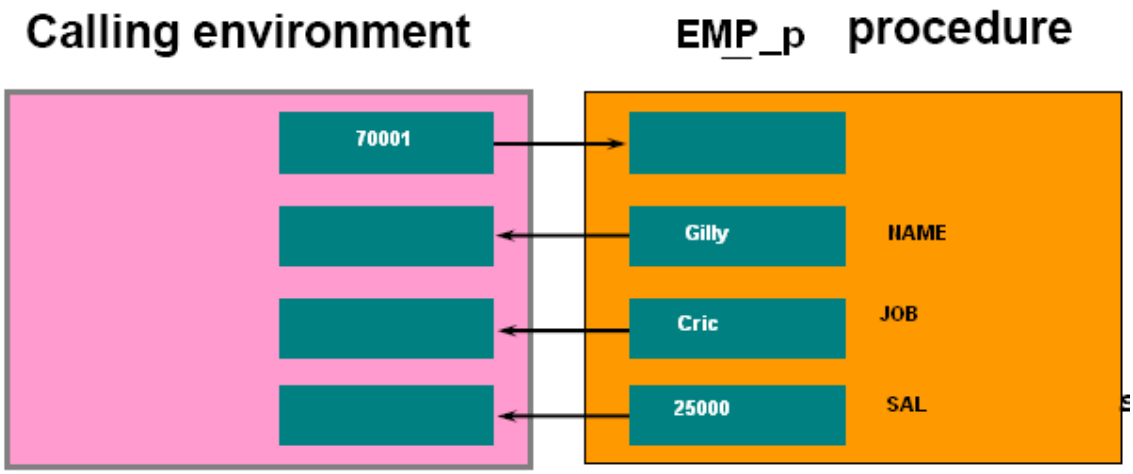
-- Sample procedure using IN parameter

```
CREATE OR REPLACE PROCEDURE inc_salary
(desg IN emp.job%TYPE)
IS
BEGIN
UPDATE emp
SET sal = sal * 1.10
WHERE job = desg;
END inc_salary;

-- Executing

Begin
       inc_salary('CLERK');
End;
```

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL
7001	Gilly	CRIC	100	3/31/2008 7:43:06 PM	25000
8001	1.1	ABCG	10	12/4/2004	1000
8000	Jspdemo	MGR	10	4/14/1998	5555
7666	Ponting	TL	7000	2/14/2008 7:46:19 PM	20000
123	Rahul	CLERK	7001	4/12/2007	11000
7002	Test	TESTIN	200	4/18/2008 4:33:20 PM	25000
7521	Ward	SALESM	7698	4/4/2008	1250
7943	John	CLERK	7943	12/10/1983	2200
7934	Miller	CLERK	7782	1/23/1982	1430
7902	Ford	ANALYS	7566	12/3/1981	3000
7900	James	CLERK	7698	12/3/1981	846.45
7876	Adams	CLERK	7788	3/14/1982	6600
7844	Turner	SALESM	7698	9/8/1981	1500
7839	King	PRESID	78	2/20/1988	5000
7788	Scott	ANALYS	7566	11/19/1996	3000
7782	Clark	MANAGI	7839	6/9/1981	2450
7698	Blake	MANAGI	7839	5/1/1981	2850
7654	Martin	SALESM	7698	9/28/1981	1250
7566	Jones	SECURI	7839	1/23/1982	2975

OUT Parameter



An OUT parameter is the opposite of the IN parameter, but I suppose you already had that figured out. Use the OUT parameter to pass a value back from the program to the calling PL/SQL block. An OUT parameter is like the return value for a function, but it appears in the parameter list and you can, of course, have as many OUT parameters as you like.

There are several consequences of these rules concerning OUT parameters:

- You cannot assign an OUT parameter's value to another variable or even use it in a re-assignment to itself. An OUT parameter can be found only on the left side of an assignment operation.
- You also cannot provide a default value to an OUT parameter. You can only assign a value to an OUT parameter inside the body of the module.
- Any assignments made to OUT parameters are rolled back when an exception is raised in the program. Because the value for an OUT parameter is not actually assigned until a program completes successfully, any intermediate assignments to OUT parameters are therefore ignored. Unless an exception handler traps the exception and then assigns a value to the OUT parameter, no assignment is made to that parameter. The variable will retain the same value it had before the program was called.
- An OUT actual parameter has to be a variable. It cannot be a constant, literal, or expression, since these formats do not provide a receptacle in which PL/SQL can place the OUT going value.

Steps to Execute Procedure

1. Create Procedure
2. Declare BIND Variables
3. execute the procedure
4. View the result by PRINT keyword.

Method 1:

-- Sample Procedure using OUT Parameter

```
create or replace procedure sun2(no IN int, name OUT emp.ename%TYPE, sa
OUT int)
IS
Begin
    select ename, sal into name,sa from emp where empno = no;
end;
```

For execution do the following steps:

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> variable b_name char(10);
SQL> variable b_sal number;
SQL> execute sun2(7001,:b_name,:b_sal);

PL/SQL procedure successfully completed.

SQL> print :b_name;

B_NAME
--------------------------------
Gilly

SQL> print :b_sal;

     B_SAL
----------
     25000

SQL> |
```

--Calling above Procedure in Anonymous block

```
declare
   b_name char(20);
   b_sal number;

begin
   sun2(1007,b_name,b_sal);
end;


SQL>PRINT e_name;

SQL> PRINT  e_sal
```

--Above program to display the values within the anonymous block

```
declare
   d_name char(20);
   d_sal number;

begin
sun2(1007,e_name,e_sal);
dbms_output.put_line('name ::'||b_name);
dbms_output.put_line( 'salary :' || b_sal);
end;
```

INOUT Parameter

-- Sample Procedure Using INOUT Parameter

```
create or replace procedure sample6(idno IN OUT emp.empno%TYPE)
as
Begin
         select empno into idno from emp where empno = idno;
end;
```

Steps to Execute:

Oracle SQL*Plus

File Edit Search Options Help

```
SQL> ed
Wrote file afiedt.buf

  1  create or replace procedure sample6(idno IN OUT emp.empno%TYPE)
  2  as
  3  Begin
  4        insert into emp(empno) values(idno);
  5* end;
SQL> /

Procedure created.

SQL> variable empno number;
SQL> begin
  2  :empno := 0001;
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL> print empno;

     EMPNO
----------
         1

SQL> execute sample6(:empno);
```

Method of Passing Parameters

Positional:

List actual parameters in the same order as formal parameters.

Named:

List actual parameters in arbitrary order by associating each with its corresponding formal Parameter.

Combination:

List some of the actual parameters as positional and some as named.

-- Sample Procedure using DEFAULT parameters

```
CREATE OR REPLACE PROCEDURE P9
( no IN  emp6.empno%TYPE DEFAULT 1,
name IN OUT  emp6.empname%TYPE,
sal out emp6.empsalary%TYPE
) IS
e_no number;
e_name char(20);
e_sal number;
BEGIN e_no :=no;
e_name := name;
e_sal := sal;
END;


-- Calling the procedure


declare
      t_no number := 2;
      t_name char(20) := 'dinesh';
      t_sal number;
Begin
      p9(
            no => t_no,
            name => t_name,
            sal => t_sal
        );
End;
```

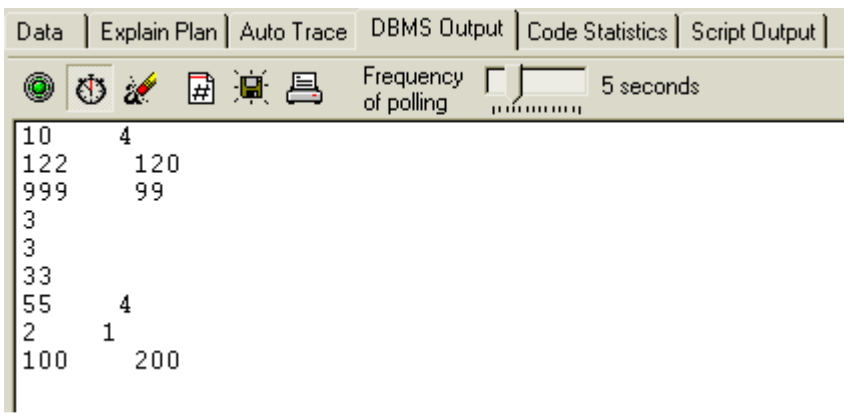
NOTE: IN OUT Parameter should not have a DEFAULT VALUE

Procedure with Cursors

-- Sample procedure using cursors…

```
create or replace procedure p10 as
      a1 num.a%TYPE;
      b1 num.b%TYPE;
      cursor c1 is
      select a,b from num;
begin
      open c1;
      loop
            fetch c1 into a1,b1;
            exit when c1%NOTFOUND;
            dbms_output.put_line(a1||'     ' || b1);
      end loop;
end;
```

```
-- Calling a Procedure.

begin
       p10;
end;
```



Drop Procedure

Drop Procedure Procedure_name;

Eg:

Drop Procedure P10;