Procedure is basic program unit which is created to perform specific task. A procedure compiled in database for future use is called stored procedure.

Definition:

A stored procedure is a set of [Structured Query Language (SQL)](https://searchsqlserver.techtarget.com/definition/SQL) statements with an assigned name, which are stored in a [relational database management system](https://searchdatamanagement.techtarget.com/definition/RDBMS-relational-database-management-system) as a group, so that it can be reused and shared by multiple programs.

Creating a Procedure

A procedure is created with the **CREATE OR REPLACE PROCEDURE**statement. The simplified syntax for the CREATE OR REPLACE PROCEDURE statement is as follows –

CREATE [OR REPLACE] PROCEDURE procedure\_name

[(parameter\_name [IN | OUT | IN OUT] type [, ...])]

{IS | AS}

BEGIN

< procedure\_body >

END procedure\_name;

Where,

* *procedure-name* specifies the name of the procedure.
* [OR REPLACE] option allows the modification of an existing procedure.
* The optional parameter list contains name, mode and types of the parameters. **IN** represents the value that will be passed from outside and OUT represents the parameter that will be used to return a value outside of the procedure.
* *procedure-body* contains the executable part.
* The AS keyword is used instead of the IS keyword for creating a standalone procedure.

### **Example**

The following example creates a simple procedure that displays the string 'Hello World!' on the screen when executed.

CREATE OR REPLACE PROCEDURE greetings

AS

BEGIN

dbms\_output.put\_line('Hello World!');

END;

/

**EXECUTION:**

SQL> set serveroutput on size 30000;

SQL> EXECUTE greetings;

Hello World!

PL/SQL procedure successfully completed.

**Passing Parameter**

create table new\_employees ( employee\_id number, emp\_name varchar2(200));

insert into new\_employees values ('1', 'TEST1');

insert into new\_employees values ('2', 'TEST2');

insert into new\_employees values ('3', 'TEST3');

insert into new\_employees values ('4', 'TEST3');

insert into new\_employees values ('5', 'TEST4');

insert into new\_employees values ('6', 'TEST5');

insert into new\_employees values ('7', 'TEST6');

COMMIT;

SELECT \* FROM new\_employees;

CREATE OR REPLACE PROCEDURE remove\_emp (emp\_id NUMBER) AS

BEGIN

DELETE FROM new\_employees

WHERE new\_employees.employee\_id = remove\_emp.emp\_id;

COMMIT;

END;

/

Sql> execute remove\_emp (‘7’);

Sql> SELECT \* FROM new\_employees;

**EXAMPLE: 2**

create table new\_staff

(

staff\_id number,

staff\_name varchar2(200),

staff\_address varchar2(200),

phone number(20),

gender char(1),

Job varchar2(200),

Salary number

);

insert into new\_staff values ('1','test1','address1','234567','M','CLERK','5000');

insert into new\_staff values ('2','test1','address2','345678','M','MANAGER','25000');

insert into new\_staff values ('3','test1','address3','456901','M','ACCOUNTANT','15000');

COMMIT;

SELECT \* FROM new\_staff;

SELECT \* FROM staff\_detail;

SELECT \* FROM staff\_more\_detail;

SELECT \* FROM salary\_detail;

Create table staff\_detail

( staff\_id number,

Staff\_name varchar2(200)

);

Create table staff\_more\_detail

( staff\_id number,

Staff\_address varchar2(200),

Phone number,

Gender char(1)

);

Create table salary\_detail

(

Staff\_id number,

Job varchar2(200),

Salary number

);

CREATE OR REPLACE PROCEDURE insertEMPDetails as

BEGIN

BEGIN

FOR det IN (SELECT \* FROM new\_staff) LOOP

insert into staff\_detail values (det.Staff\_Id, DET.STAFF\_NAME);

INSERT INTO STAFF\_MORE\_DETAIL

VALUES

(DET.STAFF\_ID, DET.STAFF\_ADDRESS, DET.PHONE, DET.GENDER);

INSERT INTO salary\_detail VALUES (DET.STAFF\_ID, DET.JOB, DET.SALARY);

COMMIT;

END LOOP;

END;

END insertEMPDetails;

/

**EXECUTION:**

Sql> execute insertEMPDetails;

**Example 3:**

CREATE TABLE DBUSER (

USER\_ID NUMBER (5) NOT NULL,

USERNAME VARCHAR2 (20) NOT NULL,

CREATED\_BY VARCHAR2 (20) NOT NULL,

CREATED\_DATE DATE NOT NULL,

PRIMARY KEY ( USER\_ID )

)

A stored procedure, accept 4 IN parameters and insert it into table “DBUSER”.

CREATE OR REPLACE PROCEDURE insertDBUSER(

p\_userid IN DBUSER.USER\_ID%TYPE,

p\_username IN DBUSER.USERNAME%TYPE,

p\_createdby IN DBUSER.CREATED\_BY%TYPE,

p\_date IN DBUSER.CREATED\_DATE%TYPE)

IS

BEGIN

INSERT INTO DBUSER ("USER\_ID", "USERNAME", "CREATED\_BY", "CREATED\_DATE")

VALUES (p\_userid, p\_username,p\_createdby, p\_date);

COMMIT;

END;

/

**Sql> execute** insertDBUSER(1001,'mkyong','system',SYSDATE);

A record is inserted into DBUSER table via insertDBUSER store procedure.

**Procedure Example:**

**Sql>** set serveroutput on size 30000;

Sql> <paste the below script>

DECLARE

a number;

b number;

c number;

PROCEDURE findMin(x IN number, y IN number, z OUT number) IS

BEGIN

IF x < y THEN

z:= x;

ELSE

z:= y;

END IF;

END;

BEGIN

a:= 23;

b:= 45;

findMin(a, b, c);

dbms\_output.put\_line(' Minimum of (23, 45) : ' || c);

END;

/

When the above code is executed at the SQL prompt, it produces the following result −

Minimum of (23, 45) : 23

PL/SQL procedure successfully completed.