SUB BLOCKS:

A SUB BLOCK IS A NAMED BLOCK OF CODE THAT IS DIRECTLY SAVED ON THE DB SERVER AND IT CAN BE EXECUTED WHEN AND WHERE IT IS REQUIRED. WE HAVE FOUR TYPES OF SUB BLOCKS IN ORACLE.

1.STORED PROCEDURES

2.STORED FUNCTIONS

3.PACKAGES

4.TRIGGERS

STORED PROCEDURES:

A STORED PROCEDURE IS A DATABASE OBJECT WHICH CONTAINS PRECOMPILED QUERIES. STORED PROCEDURES ARE A BLOCK OF CODE DESIGNED TO PERFORM A TASK WHENEVER WE CALLED AND MAY BE OR MAY NOT BE RETURN A VALUE.

WHY WE NEED STORED PROCEDURE:

WHENEVER WE WANT TO EXECUTE A SQL QUERY FROM AN APPLICATION THE SQL QUERY WILL BE FIRST PARSED (I.E. COMPLIED) FOR EXECUTION WHERE THE PROCESS OF PARSING IS TIME CONSUMING BECAUSE PARSING OCCURS EACH AND EVERY TIME, WE EXECUTE THE QUERY OR STATEMENT.

TO OVERCOME THE ABOVE PROBLEM, WE WRITE SQL STATEMENTS OR QUERY UNDER STORED PROCEDURE AND EXECUTE, BECAUSE A STORED PROCEDURE IS A PRE-COMPLIED BLOCK OF CODE WITHOUT PARSING THE STATEMENTS GETS EXECUTED WHENEVER THE PROCEDURES ARE CALLED WHICH CAN INCREASE THE PERFORMANCE OF AN APPLICATION.

ADVANTAGES OF STORED PROCEDURE:

- **AS THERE IS NO UNNECESSARY COMPILATION OF QUERIES, THIS WILL REDUCE BURDEN ON DATABASE.**
- **APPLICATION PERFORMANCE WILL BE IMPROVED**
- **USER WILL GET QUICK RESPONSE**
- **CODE REUSABILITY & SECURITY.**

PROCEDURE SYNTAX:

```
      CREATE OR REPLACE PROCEDURE <PROCEDURE_NAME>
               [ PARAMETER NAME [MODE TYPE] DATATYPE,....]
                         IS
      <VARIABLE DECLARATION>;
      BEGIN
       <EXEC STATEMENTS>;

      [ EXCEPTION BLOCK
            <EXEC-STATEMENTS>;]
      END;
```

TO EXECUTE THE PROCEDURE:

SYNTAX1:

```
EXECUTE / EXEC <PROCEDURE_NAME>;
```

SYNTAX2:(ANONYMOUS BLOCK)

```
   BEGIN
    <PROCEDURE_NAME>;
   END;
```

EXAMPLES ON PROCEDURE WITHOUT PARAMATERS:

EX1:

```
CREATE OR REPLACE PROCEDURE MY_PROC
   IS
 BEGIN
  DBMS_OUTPUT.PUT_LINE ('WELCOME TO PROCEDURES....');
 END MY_PROC;
```

<u>TO EXECUTE THE PROCEDURE:</u>

SYNTAX1:

```
EX: EXEC MY_PROC;
```

SYNTAX2:

```
EX: BEGIN
     MY_PROC;
     END;
```

EX2: WRITE A PROCEDURE TO DISPLAY SUM OF TWO NUMBERS.

```
CREATE OR REPLACE PROCEDURE ADD_PROC
                IS
A NUMBER: =10;
B NUMBER: =20;
BEGIN
 DBMS_OUTPUT.PUT_LINE ('SUM OF TWO NUMBERS = '||(A+B));
END ADD_PROC;
```

<u>EXAMPLES ON PROCEDURES WITH PARAMETERS:</u>

EX3:

```
  CREATE OR REPLACE PROCEDURE ADD_PROC (A NUMBER, B
NUMBER)
                          IS
  BEGIN
   DBMS_OUTPUT.PUT_LINE ('SUM OF TWO NUMBERS = '||(A+B));
  END ADD_PROC;
```

<u>TO EXECUTE ABOVE PROCEDURE:</u>

```
EXEC ADD_PROC (10,60);
EXEC ADD_PROC (&A, &B);
```

EX4: WRITE A PROCEDURE TO ACCEPT EMPLOYEE NUMBER AND DISPLAY CORRESPONDING EMPLOYEE NET SALARY.

```
CREATE OR REPLACE PROCEDURE EMP_PROC (TEMPNO
EMP.EMPNO%TYPE)
                        IS
TSAL EMP.SAL%TYPE;
TCOMM EMP.COMM%TYPE;
NETSAL NUMBER;
COMM_NULL EXCEPTION;
BEGIN
 SELECT SAL, COMM INTO TSAL, TCOMM FROM EMP WHERE
EMPNO=TEMPNO;
 IF TCOMM IS NULL THEN
     RAISE COMM_NULL;
 END IF;
 NETSAL: =TSAL+TCOMM;
 DBMS_OUTPUT.PUT_LINE ('GIVEN EMPLOYEE NET SALARY =
'||NETSAL);
EXCEPTION
 WHEN COMM_NULL THEN
   RAISE_APPLICATION_ERROR (-20001,'GIVEN EMPLOYEE IS NOT
GETTING COMMISSION.');
 WHEN NO_DATA_FOUND THEN
   RAISE_APPLICATION_ERROR (-20002, 'SUCH EMPLOYEE
NUMBER IS NOT EXIST.');
END EMP_PROC;
```

<u>PROCEDURES RETURN VALUES THROUGH PARAMETER MODES:</u>

- THERE ARE THREE TYPES OF PARAMETERS MODES.

IN -> IT ACCEPTS INPUT INTO STORED PROCEDURE(DEFAULT)

OUT -> IT RETURNS OUTPUT THROUGH STORED PROCEDURE

IN OUT -> BOTH ACCEPTING AND ALSO RETURN.

<u>EX. ON “IN” PARAMETERS</u>:

EX5:

```
  CREATE OR REPLACE PROCEDURE ADD_PROC (A IN NUMBER, B
IN NUMBER)
                      IS
  BEGIN
   DBMS_OUTPUT.PUT_LINE ('SUM OF TWO NUMBERS = '||(A+B));
  END ADD_PROC;


EXEC ADD_PROC (90,30);
```

EX6:

CREATE A SP TO INPUT EMPNO AND DISPLAY THAT EMPLOYEE NAME, SAL FROM EMP TABLE?

```
SQL> CREATE OR REPLACE PROCEDURE SP1(P_EMPNO IN
NUMBER)
  IS
  V_ENAME VARCHAR2(10);
  V_SAL NUMBER (10);
  BEGIN
  SELECT ENAME, SAL INTO V_ENAME, V_SAL FROM EMP WHERE
EMPNO=P_EMPNO;
 DBMS_OUTPUT.PUT_LINE(V_ENAME||','||V_SAL);
  END;
  /
PROCEDURE CREATED.


SQL> EXECUTE SP1(7788);
SCOTT,3000
```

<u>EX ON “OUT” PARAMETERS:</u>

<u>EX7:</u>

```
SQL> CREATE OR REPLACE PROCEDURE SP2(X IN NUMBER, Y OUT NUMBER)
 IS
 BEGIN
 Y: =X*X*X;
 END;
  /
PROCEDURE CREATED.
SQL> EXECUTE SP2(5);
ERROR AT LINE 1:
ORA-06550: LINE 1, COLUMN 7:
PLS-00306: WRONG NUMBER OR TYPES OF ARGUMENTS IN CALL TO 'SP2'
```

<u>NOTE: TO OVERCOME THE ABOVE PROBLEM THEN WE FOLLOW THE FOLLOWING 3 STEPS,</u>

<u>STEP1: DECLARE REFERENCED /BIND VARIABLE FOR "OUT" PARAMETERS IN SP:</u>

<u>SYNTAX:</u>

```
        VAR[IABLE] <REF.VARIABLE NAME> <DT>[SIZE];
```

<u>STEP2: TO ADD A REFERENCED /BIND VARIABLE TO A SP:</u>

<u>SYNTAX:</u>

```
EXECUTE <PNAME> (VALUE1, VALUE2, ……:<REF.VARIABLE NAME>….);
```

<u>STEP3: PRINT REFERENCED VARIABLES:</u>

<u>SYNTAX:</u>

```
        PRINT <REF.VARIABLE NAME>;
```

<u>EXECUTION PLAN OF "OUT" PARAMETERS IN SP:</u>

```
SQL> VAR RY NUMBER;

SQL> EXECUTE SP2(5,:RY);

PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.

SQL> PRINT RY;


       RY
----------
      125
```

<u>EX8:</u>

CREATE A SP TO INPUT EMPNO AS A "IN" PARAMETER AND RETURNS THAT EMPLOYEE PROVIDENT FUND, PROFESSIONAL TAX AT 10%,20% ON BASIC SALARY BY USING "OUT" PARAMETERS?

```
SQL> CREATE OR REPLACE PROCEDURE SP3(P_EMPNO IN
NUMBER, PF OUT NUMBER, PT OUT NUMBER)
  IS
  V_SAL NUMBER (10);
  BEGIN
  SELECT SAL INTO V_SAL FROM EMP WHERE EMPNO=P_EMPNO;
  PF: = V_SAL*0.1;
  PT: = V_SAL*0.2;
  END;
  /
PROCEDURE CREATED.
SQL> VAR RPF NUMBER;
SQL> VAR RPT NUMBER;
SQL> EXECUTE SP3(7788,:RPF,:RPT);
PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.
SQL> PRINT RPF RPT;
```

EX9:

```
CREATE OR REPLACE PROCEDURE ADD_PROC (A IN NUMBER, B IN
NUMBER, C OUT NUMBER)
 IS
 BEGIN
  C: =A+B;
 END ADD_PROC;
```

OUTPUT:

```
VAR R NUMBER;
EXECUTE ADD_PROC (10,20,:R);
PRINT R;
```

<u>EX. ON "IN OUT" PARAMETERS:</u>

EX10:

```
SQL> CREATE OR REPLACE PROCEDURE SP4(X IN OUT NUMBER)
  AS
  BEGIN
  X: = X*X;
  END;
  /
PROCEDURE CREATED.
SQL> EXECUTE SP4(5);
ERROR AT LINE 1:
ORA-06550: LINE 1, COLUMN 11:
PLS-00363: EXPRESSION '5' CANNOT BE USED AS AN
ASSIGNMENT TARGET
```

<u>NOTE: TO OVERCOME THE ABOVE PROBLEM THEN WE FOLLOW THE FOLLOWING 4 STEPS,</u>

<u>STEP1: DECLARE REFERENCED VARIABLE FOR "OUT" PARAMETERS IN SP:</u>

SYNTAX:

```
        VAR[IABLE] <REF.VARIABLE NAME> <DT>[SIZE];
```

<u>STEP2: ASSIGN A VALUE TO REFERENCED VARIABLE:</u>

SYNTAX:

```
        EXECUTE <REF.VARIABLE NAME> := <VALUE>;
```

<u>STEP3: TO ADD A REFERENCED VARIABLE TO A SP:</u>

SYNTAX:

```
        EXECUTE <PNAME> (:<REF.VARIABLE NAME>……);
```

<u>STEP4: PRINT REFERENCED VARIABLES:</u>

SYNTAX:

```
        PRINT <REF.VARIABLE NAME>;
```

OUTPUT:

```
SQL> VAR RX NUMBER;
SQL> EXECUTE :RX := 10;
SQL> EXECUTE SP4(:RX);
SQL> PRINT RX;
```

NOTE: ALL PROCEDURES NAMES ARE STORED IN USER_OBJECTS.
SELECT OBJECT_NAME FROM USER_OBJECTS;

EX:

```
SELECT OBJECT_NAME FROM USER_OBJECTS WHERE
OBJECT_TYPE='PROCEDURE';
```

NOTE: PROCEDURE BODIES ARE STORED IN USER_SOURCE.

EX:

```
SELECT TEXT FROM USER_SOURCE WHERE NAME='EMP_PROC';
```

<u>DROPPING PROCEDURES:</u>

SYNTAX:

```
SQL> DROP PROCEDURE <PROCEDURE_NAME>;
```

EX: DROP PROCEDURE MY_PROC;