SUB BLOCKS:

A sub block is a named block of code that is directly saved on the 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.

Notes:

* A procedure may or may not returns a value.

* A procedure has 2 parts. Header & Body. Header part contains name of procedure, parameters/variables passed to procedure and Body contains declaration section, execution & exception Section.

Procedure syntax:

Create or Replace Procedure <Procedure_name>

[(argument mode datatype,....)]

is/as

<variable declaration>;

Begin

<exec statements>;

[Exception Block

<exec-statements>;]

end;

To execute the procedure:

syntax1:

Execute / Exec <procedure_Name>

syntax2:(Anonymous Block)

2. Begin

<procedure_name>;

End;

Examples on Procedure without paramaters:

Example1:

```
Create or Replace Procedure My_Proc
            Is
Begin
 dbms_output.put_line('Welcome to Procedures....');
End My_Proc;
```

To execute the Procedure:

syntax1:

ex: Exec My_Proc;

syntax2:

ex: Begin

my_proc;

End;

Ex: 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;
```

Examples on Procedures with Parameters:Ex:

```
 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;
```

to execute above procedure:

```
Exec Add_Proc(10,60);
Exec Add_Proc(&a,&b);
```

Ex: 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;
```

Procedures Return values through Parameter Modes:

There are three types of parameters modes.

IN --> it accepts input into Sub-Program(default).It stores the input value given by user.

OUT --> it returns output through sub-program.Genarally a procedure does not return a value but if we wanna return a value at that time we will use this out parameter.

IN OUT --> BOTH.

Ex. on IN:

```
 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);
```

Ex on OUT :

```
Create or replace procedure add_Proc(A in Number, B in Number, C out Number)
                    is
Begin
 C:=A+B;
End add_proc;
```

Here C is a return type that returns a value,to hold/catch the return value we need to use a referenced variable.

to execute above procedure two methods:

1. on Sql prompt:

```
sql> Var res Number;
```

- **to declare variables on sql prompt those variables are called as Global Variables/Bind variables/sql buffer variables/reference variable.**
- **bind variables are represented with ':' operator.**

```
sql> exec add_proc(20,50,:res);
sql> Print :res;
```

2. method: using pl/sql block.

```
Declare
 result Number;
Begin
 add_proc(&a,&b,result);
 dbms_output.put_line('Sum of two Numbers = '||Result);
End;
```

- **Above pl/sql block is called as Calling Procedure.**

Ex. on IN OUT:

```
Create or Replace Procedure Add_proc(a in out number)
                Is
Begin
a:=a*a*a;
```

End add_proc;

sql> variable a Number;

To assign value into Bind variable(a);

sql> Exec :a :=10;

sql> Exec Add_proc(:a);

Ex:

```
Create or replace procedure add_Proc(A in Number, B in Number, C out Number)
                        is
Begin
 C:=A+B;
End add_proc;
```

- All Procedures Names are stored in User_objects.

select object_name from user_objects;

select object_name from user_objects where object_type='PROCEDURE';

- Procedure Bodies are stored in User_Source.

select text from user_source where name='EMP_PROC';

Dropping Procedures:

syntax:

sql> Drop Procedure <procedure_name>;

ex: Drop Procedure my_proc;