Group A

Assignment No 6

Title of the Assignment: Named PL/SQL Block: PL/SQL Stored Procedure and Stored Function.

Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and 900 category is first class, if marks 899 and 825 category is Higher Second Class.Write a PL/SQL block to use procedure created with above requirement.

Stud_Marks(name, total_marks)

Result(Roll,Name, Class)

Objective:

1. To understand the difference between procedure and function
2. To understand commands related to procedure and function

Outcome: Students will be able to learn and understand various stored procedure and Stored Function

Theory:

Procedures and Functions are the subprograms which can be created and saved in the database as database objects. They can be called or referred inside the other blocks also.

❖ **Parameter:**

The parameter is variable or placeholder of any valid PL/SQL data-type through which the PL/SQL subprogram exchange the values with the main code. This parameter allows to give input to the subprograms and to extract from these subprograms.

- These parameters should be defined along with the subprograms at the time of creation.
- These parameters are included n the calling statement of these subprograms to interact the values with the subprograms.
- The datatype of the parameter in the subprogram and the calling statement should be same.
- The size of the datatype should not mention at the time of parameter declaration, as the size is dynamic for this type.

- Based on their purpose parameters are classified as
    1. IN Parameter
    2. OUT Parameter
    3. IN OUT Parameter

1. IN Parameter:
    - This parameter is used for giving input to the subprograms.
    - It is a read-only variable inside the subprograms. Their values cannot be changed inside the subprogram.
    - In the calling statement, these parameters can be a variable or a literal value or an expression, for example, it could be the arithmetic expression like '5*8' or 'a/b' where 'a' and 'b' are variables.
    - By default, the parameters are of IN type

2. OUT Parameter:
    - This parameter is used for getting output from the subprograms.
    - It is a read-write variable inside the subprograms. Their values can be changed inside the subprograms.
    - In the calling statement, these parameters should always be a variable to hold the value from the current subprograms.

3. IN OUT Parameter:
    - This parameter is used for both giving input and for getting output from the subprograms.
    - It is a read-write variable inside the subprograms. Their values can be changed inside the subprograms.
    - In the calling statement, these parameters should always be a variable to hold the value from the subprograms.

These parameter types should be mentioned at the time of creating the subprograms.

❖ RETURN

RETURN is the keyword that instructs the compiler to switch the control from the subprogram to the calling statement. In subprogram RETURN simply means that the control needs to exit from the subprogram. Once the controller finds RETURN keyword in the subprogram, the code after this will be skipped.

Normally, parent or main block will call the subprograms, and then the control will shift from those parent blocks to the called subprograms. RETURN in the subprogram will return the control back to their parent block. In the case of functions RETURN statement also returns the value. The datatype of this value is always mentioned at the time of function declaration. The datatype can be of any valid PL/SQL data type.

❖ Procedure in PL/SQL

A Procedure is a subprogram unit that consists of a group of PL/SQL statements. Each procedure in Oracle has its own unique name by which it can be referred. This subprogram unit is stored as a database object. Below are the characteristics of this subprogram unit.

- Procedures are standalone blocks of a program that can be stored in the database.
- Call to these procedures can be made by referring to their name, to execute the PL/SQL statements.
- It is mainly used to execute a process in PL/SQL.
- It can have nested blocks, or it can be defined & nested inside the other blocks or packages.
- It contains declaration part (optional), execution part, exception handling part (optional).
- The values can be passed into the procedure or fetched from the procedure through parameters.
- These parameters should be included in the calling statement.
- Procedure can have a RETURN statement to return the control to the calling block, but it cannot return any values through the RETURN statement.
- Procedures cannot be called directly from SELECT statements. They can be called from another block or through EXEC keyword.

Syntax:

```
CREATE OR REPLACE PROCEDURE
<procedure_name>
(
<parameter IN/OUT <data type>
…
…
)
[ IS | AS ]
<Declaration part>
BEGIN
<Execution part>
EXCEPTION
<Exception handling part>
            END;
```

❖ PROCEDURE

CREATE PROCEDURE instructs the compiler to create new procedure. Keyword 'OR REPLACE' instructs the compile to replace the existing procedure (if any) with the current one.

- Procedure name should be unique.
- Keyword 'IS' will be used, when the procedure is nested into some other blocks. If the procedure is standalone then 'AS' will be used. Other than this coding standard, both have the same meaning.

Procedures: Example

The below example creates a procedure ‘employer_details’ which gives the details of the employee.

```
CREATE OR REPLACE PROCEDURE employer_details
IS
CURSOR emp_cur IS
SELECT first_name, last_name, salary FROM emp_tbl;
emp_rec emp_cur%rowtype;
BEGIN
FOR emp_rec in sales_cur
LOOP
dbms_output.put_line(emp_cur.first_name || ' ' ||emp_cur.last_name
|| ' ' ||emp_cur.salary);
END LOOP;
END;
/
```

❖ FUNCTION:

A function is a standalone PL/SQL subprogram. Like PL/SQL procedure, functions have a unique name by which it can be referred. These are stored as PL/SQL database objects. Below are some of the characteristics of functions.

- Functions are a standalone block that is mainly used for calculation purpose.
- Function use RETURN keyword to return the value, and the datatype of this is defined at the time of creation.
- A Function should either return a value or raise the exception, i.e. return is mandatory in functions
- Function with no DML statements can be directly called in SELECT query whereas the function with DML operation can only be called from other PL/SQL blocks.
- It can have nested blocks, or it can be defined and nested inside the other blocks or packages.
- It contains declaration part (optional), execution part, exception handling part (optional).
- The values can be passed into the function or fetched from the procedure through the parameters.
- These parameters should be included in the calling statement.
- Function can also return the value through OUT parameters other than using RETURN.
- Since it will always return the value, in calling statement it always accompanies with assignment operator to populate the variables.

Syntax:

```
CREATE OR REPLACE FUNCTION
<procedure_name>
        (
        <parameter1 IN/OUT <datatype>
        ..
        .
        )
        RETURN <datatype>
[ IS | AS ]
        <declaration_part>
BEGIN
        <execution part>
EXCEPTION
        <exception handling part>
END;
```

CREATE FUNCTION instructs the compiler to create a new function. Keyword 'OR REPLACE' instructs the compiler to replace the existing function (if any) with the current one.

- The Function name should be unique.
- RETURN datatype should be mentioned.
- Keyword 'IS' will be used, when the procedure is nested into some other blocks. If the procedure is standalone then 'AS' will be used. Other than this coding standard, both have the same meaning

Function: Example

Let’s create a function called "employer_details_func' similar to the one created in stored proc

```
1> CREATE OR REPLACE FUNCTION employer_details_func
2> RETURN VARCHAR(20);
3> IS
5> emp_name VARCHAR(20);
6> BEGIN
7> SELECT first_name INTO emp_name
8> FROM emp_tbl WHERE empID = '100';
9> RETURN emp_name;
10> END;
11> /
```

Similarities between Procedure and Function

- Both can be called from other PL/SQL blocks.
- If the exception raised in the subprogram is not handled in the subprogram exception handling section, then it will propagate to the calling block.
- Both can have as many parameters as required.
- Both are treated as database objects in PL/SQL.

Program

```
MariaDB [(none)]> use stud;
Database changed
MariaDB [stud]> create table stud_marks(rollno int(2),name varchar(12),total_marks
int(12));
Query OK, 0 rows affected (0.06 sec)
MariaDB [stud]> insert into stud_marks
values(1,"Ravi",933),(2,"sagar",450),(3,"sarita",1300),(4,"avi",250),(5,"raj",675);
Query OK, 5 rows affected (0.02 sec)
Records: 5 Duplicates: 0 Warnings: 0

MariaDB [stud]> select * from stud_marks;
+--------+--------+-------------+
| rollno | name | total_marks |
+--------+--------+-------------+
| 1 | Ravi | 933 |
| 2 | sagar | 450 |
| 3 | sarita | 1300 |
| 4 | avi | 250 |
| 5 | raj | 675 |
+--------+--------+-------------+
5 rows in set (0.00 sec)

MariaDB [stud]> create table new_stud_marks(roll_no int(2),name char(10),grade
char(10));
Query OK, 0 rows affected (0.04 sec)

delimeter //
create procedure proc_grade1()
 begin
 declare i int;
 declare n int;
 declare rollno1 int;
 declare name1 varchar(20);
 declare class1 varchar(40);
 declare total1 int;
 declare s1 int;
 declare s2 int;
 declare s3 int;
 declare s4 int;
 declare s5 int;
 select count(*)into n from marks;
 set i=0;
 disp:loop
 set i=i+1;
 select rollno into rollno1 from marks where rollno=i;
 select name into name1 from marks where rollno=i;
 select sub1 into s1 from marks where rollno=i;
 select sub2 into s2 from marks where rollno=i;
 select sub3 into s3 from marks where rollno=i;
 select sub4 into s4 from marks where rollno=i;
 select sub5 into s5 from marks where rollno=i;
 set total1=s1+s2+s3+s4+s5;
 if total1<=1500 and total1>990 then
 set class1='distinction';
 else
 if total1<=989 and total1>900 then
 set class1='first class';
 else
 if total1<899 and total1>825 then
 set class1='higher class';
 else
 set class1='pass class';
 end if;
 end if;
 end if;
 insert into stud_marks values(rollno1,name1,total1);
 insert into results values(rollno1,name1,class1);
 if i=n then
 leave disp;
 end if;
 end loop;
 end
 //
```

```
mysql> create table stud_marks(rolllno int,name varchar(90),class varchar(90));
Query OK, 0 rows affected (0.29 sec)
mysql> create table results(rollno int,name varchar(90),class varchar (90));
Query OK, 0 rows affected (0.27 sec)
mysql> alter table stud_marks drop class;
Query OK, 0 rows affected (0.28 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> alter table stud_marks add total int;
Query OK, 0 rows affected (0.26 sec)
Records: 0  Duplicates: 0  Warnings: 0
 create table marks(rollno int,name varchar(99),sub1 int,sub2 int,sub3 int,sub4 int,sub5 int);
Query OK, 0 rows affected (0.34 sec)
mysql> insert into marks values(1,'komal',254,123,345,234,444);
Query OK, 1 row affected (0.06 sec)
mysql> select * from marks;
+--------+--------+------+------+------+------+------+
| rollno | name   | sub1 | sub2 | sub3 | sub4 | sub5 |
+--------+--------+------+------+------+------+------+
|      1 | komal  |  254 |  123 |  345 |  234 |  444 |
|      2 | shital |  154 |  223 |  325 |  134 |  144 |
|      3 | raj    |  154 |  523 |  245 |  244 |  414 |
+--------+--------+------+------+------+------+------+
3 rows in set (0.01 sec)

//////////////////////output//////////////////////////////////////
mysql> delimiter ;
mysql> call proc_grades();
```

```
Query OK, 1 row affected (0.97 sec)

mysql> select * from stud_marks;
+---------+--------+-------+
| rolllno | name   | total |
+---------+--------+-------+
|       1 | komal  |  NULL |
|       2 | shital |  NULL |
|       3 | raj    |  NULL |
+---------+--------+-------+
3 rows in set (0.00 sec)

mysql> select * from results;
+--------+--------+------------+
| rollno | name   | class      |
+--------+--------+------------+
|      1 | komal  | pass class |
|      2 | shital | pass class |
|      3 | raj    | pass class |
+--------+--------+------------+
3 rows in set (0.00 sec
```

Conclusion: Performed implementation of procedures and functions in MYSQL successfully.

Viva Questions:

- Explain a for loop with example?
- Explain if-else loop with example?
- Explain In parameter with example?
- Explain while loop with example?
- Explain out parameter with example?

Date:	
Marks obtained:	
Sign of course coordinator:	
Name of course Coordinator:	