DBMS Assignment A7

```
mysql> create table stu_marks(name varchar(20),total_marks integer(10));
Query OK, 0 rows affected, 1 warning (2.18 sec)

mysql> create table result(roll_no int(8),name varchar(20),grade varchar(15));
Query OK, 0 rows affected, 1 warning (1.14 sec)

mysql> delimiter &&
mysql> create procedure proc_grade(IN roll_no integer,IN name varchar(20),IN total_marks
integer(10))
    -> begin
    -> declare grade varchar(20)
    -> ;
    -> IF total_marks>=990 and total_marks<=1500 THEN
    -> set grade:='Distinction';
    -> ELSEIF total_marks>=900 and total_marks<=989 THEN
    -> set grade:='First Class';
    -> ELSEIF total_marks>=825 and total_marks<=899 THEN
    -> set grade:='Higher Second';
    -> END IF;
    -> insert into stu_marks values(name,total_marks);
    -> insert into result values(roll_no,name,grade);
    -> end;
    -> &&
Query OK, 0 rows affected, 1 warning (0.36 sec)

mysql> call proc_grade(1,'Dhruvil',1449);
Query OK, 1 row affected (0.63 sec)

mysql> call proc_grade(2,'Soham',990);
Query OK, 1 row affected (0.27 sec)
```

```
mysql> call proc_grade(3,'Gaurav',1080);
Query OK, 1 row affected (0.15 sec)

mysql> select * from result ;
+---------+----------+-------------+
| roll_no | name     | grade       |
+---------+----------+-------------+
|       1 | Dhruvil  | Distinction |
|       2 | Soham    | Distinction |
|       3 | Gaurav   | Distinction |
+---------+----------+-------------+
3 rows in set (0.05 sec)

mysql> call proc_grade(4,'Aabha',980);
Query OK, 1 row affected (0.29 sec)

mysql> select * from result ;
+---------+----------+-------------+
| roll_no | name     | grade       |
+---------+----------+-------------+
|       1 | Dhruvil  | Distinction |
|       2 | Soham    | Distinction |
|       3 | Gaurav   | Distinction |
|       4 | Aabha    | First Class |
+---------+----------+-------------+
4 rows in set (0.03 sec)

mysql> select * from stu_marks;
```

```
+----------+-------------+
| name     | total_marks |
+----------+-------------+
| Dhruvil  |        1449 |
| Soham    |         990 |
| Gaurav   |        1080 |
| Aabha    |         980 |
+----------+-------------+
```

4 rows in set (0.06 sec)

MES College of Engineering Pune-01

Department of Computer Engineering

Name of Student: Dhruvil Shah	**Class: TE Comp 1**
Semester/Year: 5th/2020	**Roll No: 047**
Date of Performance:	**Date of Submission:**
Examined By:	**Experiment No: Part A-07**

GROUP: A ASSIGNMENT NO: 07

AIM: Write a PL/SQL block of code for Stored Procedure and Stored Function.

PROBLEM STATEMENT:

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 and900 category is first class, if marks 899 and 825 category is Higher Second Class Write a PL/SQL block for using procedure created with above requirement.

Stud_Marks(name, total_marks)

Result(Roll,Name, Class)

OBJECTIVES:

- To learn stored procedure in PL/SQL.
- To learn stored functions in PL/SQL.

PRE - REQUISITES:

Interactive SQL commands, PL/SQL programming, use of oracle 11g database Editor.

APPARATUS:

- Operating System recommended: 64-bit Open source Linux or its derivative
- Front End :- Oracle Editor
- Back end: Oracle 11g

SYNTAX:

Stored Procedure:

- CREATE [OR REPLACE] PROCEDURE procedure_name
- [(parameter_name [IN | OUT | IN OUT] type [, ...])]
- {IS | AS}
- Variable declarations;
- Constant declarations;
- 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.
- Mode: IN, OUT, IN OUT
    - IN (Default): A parameter of IN mode cannot be assigned a value.
    - OUT: Explicitly specify mode. Parameter must be assigned a value before returning.
    - IN OUT: A parameter values can be accessed and assigned a new value.

Stored Function:

- CREATE [OR REPLACE] FUNCTION function_name
- [(parameter_name [IN | OUT | IN OUT] type [, ...])]
- RETURN return_datatype
- {IS | AS}
- BEGIN
- < function_body >
- END [function_name];

Where

- *function-name* specifies the name of the function.
- [OR REPLACE] option allows the modification of an existing function.

- 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.
- The function must contain a **return** statement and RETURN datatype must not include size specification..
- The *RETURN* clause specifies the data type you are going to return from the function.
- *function-body* contains the executable part.
- The AS keyword is used instead of the IS keyword for creating a standalone function.

CONCLUSION:

QUESTIONS:

1. What is a Stored Procedure?
2. Describe the use of %ROWTYPE and %TYPE in SQL?
3. Explain IN, OUT, IN-OUT mode in stored procedure.
4. What is a Stored Function?
5. What is difference between stored functions and stored procedures?

Q1 What is stored procedure?

Ans A stored procedure is prepared SQL code that you can save, so the code can be reused over and over again. So if you have an SQL query that you write over and over again, save it as stored procedure and then just call to execute it. You can also pass parameter to stored procedure, so that the stored procedure can act based on the parameter values that is passed

Syntax:

```
CREATE PROCEDURE procedure_name AS sql_statement GO;
```

Q2 Describe the use of %ROWTYPE and %TYPE in SQL?

Ans %ROWTYPE:

The %ROWTYPE attribute lets you declare a record that represents either a full or partial row of a database table or view. For every column of the full or partial row, the record has a field with the same name and data type. If the structure of the row changes, then the structure of the record changes accordingly.

%TYPE:

The %TYPE attribute lets you declare a data item of the same type as previously declared variable or column. If the declaration of the referenced item changes, then the declaration of the referenced item changes accordingly. The %TYPE attribute is particularly useful when declaring variables to hold database values.

Q3 Explain IN, OUT , IN-OUT mode in stored procedure.

Ans Values passed to or oracle function can be either IN, OUT or IN-OUT which decides how variables can be used within the procedure.

IN: A variable passed in this mode is of read only nature. This is to say, the value cannot be changed and its scope is restricted within the procedure.

OUT: In this mode a variable is write only and can be passed back to the calling program. It cannot be read inside the procedure and needs to be assigned a value.

INOUT: This procedure has features of both IN and OUT mode. The procedure can also read the variables value and can also change it to pass it to the calling function

Q4 What is Stored function ?

Ans A Stored function in MySQL is a set of SQL Statements that perform some task / operation and return a single value. It is one of the types of stored programs in MySQL. When you will create a stored function, makes sure that you have a CREATE ROUTINE database privilege

Syntax

DELIMITER $$

```
CREATE FUNCTION fun_name (fun_parameters(s))
RETURN DATATYPE
[NOT] ( characteristics)
fun_ body ;
```

Q5 What is the difference between Stored functions and Stored procedure ?

Ans

	Stored Function		Stored Procedure
1	A function has a return type and returns value	1	A procedure does not value a return type. But it returns values using OUT parameters.
2	A function does not allow output parameters	2	A procedure allows both input and output parameters.
3	You cannot manage transaction inside a function	3	You can manage transactions inside a function
4	You cannot cannot call stored procedures from a function	4	You can call function from a stored procedure