**5. 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 scoredby**

**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)**

**Note: Instructor will frame the problem statement for writing stored procedure and Function in**

**line with above statement.**

mysql> use Abhi;

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

mysql> create table marks(roll\_no int,name varchar(20),total\_marks varchar(20));

Query OK, 0 rows affected (0.67 sec)

mysql> create table result(roll\_no int,name varchar(20),class varchar(20));

Query OK, 0 rows affected (0.41 sec)

mysql> insert into marks values('1','Abhi','1400');

Query OK, 1 row affected (0.04 sec)

mysql> insert into marks values('2','piyush','980');

Query OK, 1 row affected (0.08 sec)

mysql> insert into marks values('3','hitesh','880');

Query OK, 1 row affected (0.08 sec)

mysql> insert into marks values('4','ashley','820');

Query OK, 1 row affected (0.08 sec)

mysql> insert into marks values('5','partik','740');

Query OK, 1 row affected (0.03 sec)

mysql> insert into marks values('6','patil','640');

Query OK, 1 row affected (0.08 sec)

mysql> delimiter //

mysql> create procedure proc\_result(in marks int,out class

char(20))

-> begin

-> if(marks<1500&&marks>990)

-> then

-> set class='Distincton';

-> end if;

-> if(marks<989&&marks>890)

-> then

-> set class='First Class';

-> end if;

-> if(marks<889&&marks>825)

-> then

-> set class='Higher Second Class';

-> end if;

-> if(marks<824&&marks>750)

-> then

-> set class='Second Class';

->end if;

->if(marks<749&&marks>650)

-> then

-> set class='Passed';

-> end if;

-> if(marks<649)

-> then

-> set class='Fail';

-> end if;

-> end;

-> //

Query OK, 0 rows affected (0.00 sec)

delimiter //

create function final\_results(R1 int)

returns int deterministic

begin

declare fmarks int;

declare grade varchar(20);

declare stud\_name varchar(20);

select marks.total\_marks,marks.name into fmarks,stud\_name from marks where marks.roll\_no=R1;

call proc\_result(fmarks,grade);

insert into result values(R1,stud\_name,grade);

return R1;

end;

//

Query OK, 0 rows affected (0.00 sec)

mysql> select final\_result3(2);

-> //

+------------------+

| final\_result3(2) |

+------------------+|

2 |

+------------------+

1 row in set (0.05 sec)

mysql> select final\_result3(3);//

+------------------+

| final\_result3(3) |

+------------------+

|

3 |

+------------------+

1 row in set (0.04 sec)

mysql> select final\_result3(4);//

+------------------+

| final\_result3(4) |

+------------------+

|

4 |

1 row in set (0.12 sec)

mysql> select final\_result3(5);//

+------------------+

| final\_result3(5) |

+------------------+

|

5 |

+------------------+

1 row in set (0.05 sec)

mysql> select \* from result;

-> //

+---------+--------+---------------------+

| roll\_no | name

| class

|+---------+--------+---------------------+

| 1 | NULL | Distincton |

| 1 | Abhi | Distincton |

| 1 | Abhi | Distincton |

| 2 | piyush | First Class | 3 | hitesh | Higher Second Class |

| 4 | ashley | Second Class |

| 5 | partik | Passed |

|

+---------+--------+---------------------+

7 rows in set (0.00 sec)

**Theory: PL/SQL Stored Procedures and Functions for Student Grade Categorization**

**Introduction to PL/SQL:**

PL/SQL (Procedural Language/Structured Query Language) is an extension of SQL that allows for procedural programming capabilities in Oracle databases. It enables developers to write complex database interactions using constructs such as loops, conditionals, and modular code structures. Two fundamental components of PL/SQL are stored procedures and stored functions.

**Stored Procedures:**

A stored procedure is a named block of PL/SQL code that performs a specific task and can accept parameters. Procedures can be invoked explicitly and do not return a value directly; instead, they can modify the database state or perform calculations that are reflected in output parameters or database changes.

**Key Features of Stored Procedures:**

Parameterization: They can accept input parameters and provide output parameters, allowing for dynamic execution based on the provided data.

Modularity: Procedures promote code reusability by encapsulating frequently used logic, making it easier to maintain and manage code.

Performance: Since procedures are compiled and stored in the database, they can execute more quickly than executing multiple individual SQL statements.

Example in Context: In our application, we create a stored procedure proc\_result to categorize student marks into distinct classes based on specified ranges. This allows for efficient processing of student data.

**Stored Functions:**

A stored function is similar to a procedure, but it is designed to return a single value. Functions can be called from SQL statements as part of expressions, making them very versatile. Like procedures, they can also accept parameters.

**Key Features of Stored Functions:**

Return Value: Functions must return a value, which can be used directly in SQL queries.

Usage in SQL: Functions can be called in SQL statements, enabling them to be integrated seamlessly into data retrieval processes.

Side Effects: While functions can modify database state, they are ideally used for calculations that do not alter the database context.

Example in Context: The stored function final\_results retrieves a student's marks and name based on their roll number, calls the proc\_result procedure to categorize the marks, and returns the roll number after inserting the classification into the results table. This ensures that the logic for determining the class is encapsulated within the procedure while the function manages the data retrieval and insertion process.