Experiment No. 6

Group: - 1 from T4 Batch

0077 – Aakash Joshi

2039 – Akanksha Lokhande

Aim: -

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 and900 category is first class, if marks899and 825 category is Higher Second Class.

Write a PL/SQLblock 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.

Date: - 1/9/2023

Theory: -

In the realm of database management and application development, PL/SQL (Procedural Language/Structured Query Language) holds a prominent role, enabling the creation of powerful, reusable, and modular code blocks. One essential aspect of PL/SQL is the creation of stored procedures and functions, which encapsulate business logic and enable efficient data processing. In this theoretical overview, we delve into the creation and use of PL/SQL stored procedures and functions, illustrated through a practical scenario involving student categorization.

Stored Procedures: The Power of Procedural Logic

Stored procedures in PL/SQL are a collection of SQL and procedural statements that can be executed as a single unit. They offer a multitude of benefits, including code encapsulation, modularity, reusability, and enhanced security. In the context of the student categorization problem, a stored procedure named `proc_Grade` can be designed to categorize students based

on their examination marks. This procedure will accept input parameters, such as the student's name and total marks, and return the corresponding class category.

Stored Functions: Reusable Logic Blocks

Stored functions are like stored procedures but have a distinct purpose: they return a single value. In our theoretical scenario, a stored function could be employed to calculate and return the student's class category based on their total marks. This function let's call it `fn_CategorizeStudent`, encapsulates the categorization logic and can be used in SQL statements to fetch student categories easily.

The Practical Scenario: Stud_Marks and Result Tables

The heart of our scenario is the interaction between two database tables: `Stud_Marks` and `Result`. The `Stud_Marks` table stores student information, including their names and total marks. The `Result` table is meant to capture the results of the categorization process. Using stored procedures and functions, we can seamlessly categorize students based on their marks and populate the `Result` table with the relevant data.

Realizing the Solution

To implement this solution, we begin by creating the `proc_Grade` stored procedure, which takes the student's name and total marks as input and determines their class category. The logic checks the marks against predefined criteria and assigns the category accordingly. Then, a stored function, `fn_CategorizeStudent`, encapsulates the categorization logic for reusability and clarity.

The PL/SQL block orchestrates the process. It retrieves student data from the `Stud_Marks` table, applies the `fn_CategorizeStudent` function to determine the category, and inserts the results into the `Result` table.

Conclusion: -

The theoretical framework described above embodies the essence of PL/SQL in database applications. Stored procedures and functions empower developers to encapsulate business logic, promote reusability, and enhance the efficiency of data processing. By applying this framework

to the student categorization problem, we showcase the practical value of PL/SQL in real-world scenarios, where data manipulation and categorization are fundamental tasks.

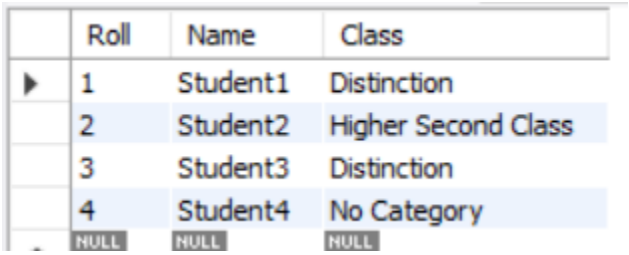
Code: -

```
-- Create a stored procedure to categorize student marks
DELIMITER //

CREATE PROCEDURE CategorizeStudentMarks3()
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE student_roll INT;
    DECLARE student_name VARCHAR(255);
    DECLARE student_marks INT;
    DECLARE student_category VARCHAR(255);

    DECLARE student_cursor CURSOR FOR
        SELECT Roll, Name, Total_Marks FROM Stud_Marks;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN student_cursor;

    categorize_loop: LOOP
        FETCH student_cursor INTO student_roll, student_name, student_marks;
        IF done = 1 THEN
```

```
            LEAVE categorize_loop;
        END IF;

        IF student_marks <= 1500 AND student_marks >= 990 THEN
            SET student_category = 'Distinction';
        ELSEIF student_marks >= 900 AND student_marks <= 989 THEN
            SET student_category = 'First Class';
        ELSEIF student_marks >= 825 AND student_marks <= 899 THEN
            SET student_category = 'Higher Second Class';
        ELSE
            SET student_category = 'No Category';
        END IF;

        INSERT INTO Result VALUES (student_roll, student_name, student_category);
    END LOOP;

    CLOSE student_cursor;
END;

//

DELIMITER ;

-- Call the stored procedure to categorize students and populate the Result table
CALL CategorizeStudentMarks3();
```

SELECT * FROM result;

Output: -



Roll	Name	Class
1	Student1	Distinction
2	Student2	Higher Second Class
3	Student3	Distinction
4	Student4	No Category
NULL	NULL	NULL