Problem Statement 3 (Unnamed Block)
Employee(emp_id, dept_idemp_name, DoJ, salary, commission,job_title)
Salary_Increment(emp_id, new_salary)
Consider the schema given above. Write a PLSQL Unnamed Block of code to increase the salary of employee
115 based on the following conditions:
Accept emp_id from user. If experience of employee is more than 10 years, increase salary by 20%. If experience
is greater than 5 years, increase salary by 10% Otherwise 5%. (Hint: Find the years of experience from Date of
Joining (DoJ)). Store the incremented salary in Salary_Increment table.
Also handle the exception by named exception handler or user defined exception handler.

Step 1: Create the Database and Tables
Create Database: Start by creating the database where we will store our tables.
Create Tables: Define Employee and Salary_Increment tables based on the schema provided.

```
-- Step 1: Create Database

CREATE DATABASE EmployeeDB;
USE EmployeeDB;


-- Step 2: Create Employee Table

CREATE TABLE Employee (
    emp_id INT PRIMARY KEY,
    dept_id INT,
    emp_name VARCHAR(50),
    DoJ DATE,
    salary DECIMAL(10, 2),
    commission DECIMAL(10, 2),
    job_title VARCHAR(50)
);


-- Step 3: Create Salary_Increment Table

CREATE TABLE Salary_Increment (
    emp_id INT PRIMARY KEY,
    new_salary DECIMAL(10, 2),
    FOREIGN KEY (emp_id) REFERENCES Employee(emp_id)
);
```

Step 2: Insert Sample Data into Employee Table
Populate the Employee table with some sample data to use for testing.

```
-- Insert sample data

INSERT INTO Employee (emp_id, dept_id, emp_name, DoJ, salary, commission,
job_title)
```

```
VALUES (115, 2, 'John Doe', '2010-06-15', 50000, 5000, 'Manager'),
       (116, 3, 'Jane Smith', '2018-08-20', 40000, 4000, 'Analyst');
```

Step 3: Write a MySQL Procedure to Perform Salary Increment Based on Experience
Since MySQL does not support PL/SQL's anonymous blocks directly, we’ll create a stored procedure to handle the salary increment and insert the result into the Salary_Increment table.

```
DELIMITER //

CREATE PROCEDURE IncrementSalary(IN p_emp_id INT)
BEGIN
    DECLARE v_salary DECIMAL(10, 2);
    DECLARE v_new_salary DECIMAL(10, 2);
    DECLARE v_doj DATE;
    DECLARE v_experience INT;
    DECLARE v_increment_rate DECIMAL(3, 2);

    -- Retrieve employee details and calculate years of experience
    SELECT salary, DoJ INTO v_salary, v_doj
    FROM Employee
    WHERE emp_id = p_emp_id;

    -- Calculate experience in years
    SET v_experience = TIMESTAMPDIFF(YEAR, v_doj, CURDATE());

    -- Determine increment rate based on experience
    IF v_experience > 10 THEN
        SET v_increment_rate = 0.20;
    ELSEIF v_experience > 5 THEN
        SET v_increment_rate = 0.10;
    ELSE
        SET v_increment_rate = 0.05;
    END IF;

    -- Calculate new salary
    SET v_new_salary = v_salary * (1 + v_increment_rate);

    -- Insert or update the new salary in Salary_Increment table
    INSERT INTO Salary_Increment (emp_id, new_salary)
    VALUES (p_emp_id, v_new_salary)
    ON DUPLICATE KEY UPDATE new_salary = v_new_salary;

    -- Display confirmation message
    SELECT CONCAT('Salary incremented successfully for employee ID: ', p_emp_id, '.
New Salary: ', v_new_salary) AS Result;

END //

DELIMITER ;
```

Step 4: Execute the Procedure
To test the procedure, pass an emp_id as input to IncrementSalary.

```
-- Call the procedure with a specific emp_id


CALL IncrementSalary(115);
```

Step 5: Verify the Result in Salary_Increment Table
To confirm the procedure's effect, query the Salary_Increment table.

```
-- Check updated salary in the Salary_Increment table

SELECT * FROM Salary_Increment;
```