```
-- create database procedures;
-- use procedures;

/*
create table Borrower(Roll_no INT primary key , Name varchar(30),DateOfIssue date,
NameOfBook varchar(30), Status char(1)
);

create table Fine(
Roll_no INT, Date date, Amt Decimal(10,2)
);


insert into Borrower(Roll_no, Name, DateOfIssue, NameOfBook, Status) values
(101, 'Shritej', '2025-07-01','Data Structures','1'),
(102, 'Atharva Patil', '2025-07-10', 'Database Systems', '1'),
(103, 'Kunjal Patil', '2025-06-25', 'Operating Systems', '1'),
(104, 'Chaitanya Pawar', '2025-07-20', 'Computer Networks', '1'),
(105, 'Harshad Kavade', '2025-06-01', 'Software Engg.', '1'),
(106, 'Ayush Mahadik', '2025-07-15', 'Machine Learning', 'R'),
(107, 'Pranav Shewale', '2025-07-28', 'AI Basics', '1');


DELIMITER $$
```

```
CREATE PROCEDURE CalculateFine(
    IN in_rollno INT,
    IN in_bookname VARCHAR(30)
)
BEGIN
    DECLARE v_issue_date DATE;
    DECLARE v_days INT;
    DECLARE v_fine DECIMAL(10,2) DEFAULT 0;
    DECLARE v_status CHAR(1);

    -- 🔹 Exception handler for unexpected SQL errors
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT 'Error: Unexpected SQL exception occurred.' AS message;
        ROLLBACK;
    END;

    -- 🔹 (1) User Input Validation (Error Handling)
    IF in_rollno IS NULL OR in_bookname IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Error: Roll number or Book name cannot be NULL.';
    END IF;

    -- 🔹 (2) Check if borrower record exists
    IF (SELECT COUNT(*) FROM Borrower WHERE Roll_no = in_rollno AND NameOfBook =
in_bookname) = 0 THEN
```

```
    SELECT 'Error: No record found for given Roll No or Book Name.' AS message;

ELSE
    -- 🔹 (3) Fetch status and issue date
    SELECT Status, DateOfIssue INTO v_status, v_issue_date
    FROM Borrower
    WHERE Roll_no = in_rollno AND NameOfBook = in_bookname;

    -- 🔹 (4) Check if already returned
    IF v_status = 'R' THEN
        SELECT 'Error: This book has already been returned.' AS message;

    ELSE
        -- 🔹 (5) Calculate days and fine
        SET v_days = DATEDIFF(CURDATE(), v_issue_date);

        IF v_days > 30 THEN
            SET v_fine = v_days * 50;
        ELSEIF v_days >= 15 THEN
            SET v_fine = v_days * 5;
        ELSE
            SET v_fine = 0;
        END IF;

        -- 🔹 (6) Update status to Returned
        UPDATE Borrower
```

```
        SET Status = 'R'
        WHERE Roll_no = in_rollno AND NameOfBook = in_bookname;

        -- 🔹 (7) Record fine if applicable
        IF v_fine > 0 THEN
          INSERT INTO Fine VALUES (in_rollno, CURDATE(), v_fine);
          SELECT CONCAT('Fine recorded: Rs ', v_fine) AS message;
        ELSE
          SELECT 'No fine applicable.' AS message;
        END IF;
      END IF;
    END IF;
END$$

DELIMITER ;

*/

call CalculateFine(102,'Database Systems');
```

Section	Present?	Explanation
✅ **Control Structures**	✔	You used IF…ELSEIF…ELSE multiple times for:• Checking record existence• Checking if already returned• Calculating fines (3 conditions: >30, 15–30, <15 days).

Section	Present?	Explanation
✅ **Error Handling (Manual checks)**	✔	You manually handle user mistakes:• “No record found for given Roll No or Book Name.”• “This book has already been returned.”These are **user errors**, handled via logic, not SQL exceptions.
✅ **Exception Handling (SQL errors)**	✔	You used:DECLARE EXIT HANDLER FOR SQLEXCEPTIONto handle runtime SQL errors (like constraint violations, invalid updates).
✅ **Transaction safety**	✔	You used ROLLBACK in the handler, which ensures data consistency if an exception occurs.
✅ **Descriptive output**	✔	You used SELECT 'Error: ...' and CONCAT('Fine recorded...'), so user gets clear messages.
✅ **Fine calculation logic**	✔	Uses correct control flow and formulas (except if college expects exact 5/day for 15–30 and 50/day for >30, not cumulative — you already match that pattern).
✅ **Data modification**	✔	Updates status and inserts fine correctly.