DBMSL ASSIGNMENT – 4

Roll No. : 31446

Unnamed PL/SQL code block : Use of Control structure and Exception handling is mandatory.
Suggested Problem statement:
Consider tables:

- **Borrower(Roll_no, Name, DateofIssue, NameofBook, Status)**
- **Fine(Roll_no, Date, Amt)**
- **Accept Roll_no & NameofBook from user.**
- **Check the number of days (from date of issue).**
- **If days are between 15 to 30 then fine amount will be Rs. 5 per day.**
- **If no. of days > 30, per day fine will be Rs. 50 per day & for days less than 30, Rs. 5 per day.**
- **After submitting the book, status will change from I to R. If condition of fine is true, then details will be stored into fine table.**
- **Also handles the exception by named exception handler or user define exception handler**

OR

Write PL/SQL code block to calculate the area of a circle for a value of radius varying from 5 to 9. Store the radius and the corresponding values of calculated area in an empty table named areas, consisting of two columns, radius and area.

1) CREATE TABLES

```
CREATE TABLE Borrower (
      Roll_no INT,
      Name VARCHAR(50),
      DateofIssue DATE,
      NameofBook VARCHAR(100),
      Status CHAR(1) CHECK (Status IN ('I', 'R'))
);

DESC Borrower;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| Roll_no     | int          | YES  |     | NULL    |       |
| Name        | varchar(50)  | YES  |     | NULL    |       |
| DateofIssue | date         | YES  |     | NULL    |       |
| NameofBook  | varchar(100) | YES  |     | NULL    |       |
| Status      | char(1)      | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

CREATE TABLE Fine (
      Roll_no INT,
      Date DATE,
      Amt INT
);

DESC Fine;
+---------+------+------+-----+---------+-------+
| Field   | Type | Null | Key | Default | Extra |
+---------+------+------+-----+---------+-------+
| Roll_no | int  | YES  |     | NULL    |       |
| Date    | date | YES  |     | NULL    |       |
| Amt     | int  | YES  |     | NULL    |       |
+---------+------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

2) INSERT INTO SAMPLE RECORDS

```
INSERT INTO Borrower VALUES
(1, 'Anjali', '2025-07-10', 'Data Science', 'I'),
(2, 'Ravi', '2025-07-20', 'Python for Beginner', 'I'),
(3, 'Suman', '2025-07-30', 'AI Basics', 'I');

SELECT * FROM Borrower;
+---------+--------+-------------+---------------------+--------+
| Roll_no | Name   | DateofIssue | NameofBook          | Status |
+---------+--------+-------------+---------------------+--------+
|       1 | Anjali | 2025-07-10  | Data Science        | I      |
|       2 | Ravi   | 2025-07-20  | Python for Beginner | I      |
|       3 | Suman  | 2025-07-30  | AI Basics           | I      |
+---------+--------+-------------+---------------------+--------+
3 rows in set (0.00 sec)
```

3) PL/SQL Block

```
DELIMITER $$

CREATE PROCEDURE ReturnBook(
    IN in_Roll_no INT,
    IN in_Book VARCHAR(100)
)
BEGIN
    DECLARE v_DateofIssue DATE;
    DECLARE v_Days INT DEFAULT 0;
    DECLARE v_fine INT DEFAULT 0;
    DECLARE v_Status CHAR(1);

    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        SELECT 'Book not issued or already returned.' AS Message;
    END;

    SELECT DateofIssue, Status INTO v_DateofIssue, v_Status
    FROM Borrower
    WHERE Roll_no = in_Roll_no AND NameofBook = in_Book;

    IF v_DateofIssue IS NULL OR v_Status = 'R' THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Book not
issued or already returned.';
    END IF;

    SET v_Days = DATEDIFF(CURDATE(), v_DateofIssue);
    IF v_Days > 30 THEN
        SET v_fine = (30 - 15) * 5 + (v_Days - 30) * 50;
    ELSEIF v_Days > 15 THEN
        SET v_fine = (v_Days - 15) * 5;
    ELSE
        SET v_fine = 0;
    END IF;

    UPDATE Borrower
    SET Status = 'R'
    WHERE Roll_no = in_Roll_no AND NameofBook = in_Book;

    IF v_fine > 0 THEN
        INSERT INTO Fine (Roll_no, Date, Amt)
        VALUES (in_Roll_no, CURDATE(), v_fine);
    END IF;

    IF v_fine = 0 THEN
        SELECT 'Book returned successfully with NO FINE!' AS Message;
    ELSE
        SELECT CONCAT('Book returned late with fine: Rs.', v_fine) AS Message;
    END IF;

END $$

DELIMITER ;
```

4) EXECUTE THE PROCEDURE

Test Case 1:
Book Returned Within 15 Days
(No Fine)

```
CALL ReturnBook(3, 'AI Basics');
+----------------------------------------+
| Message                                |
+----------------------------------------+
| Book returned successfully with NO FINE! |
+----------------------------------------+
1 row in set (0.18 sec)
```

Test Case 2:
Book Returned Between 16 to 30 Days
(Fine Rs.5 per Day)

```
CALL ReturnBook(2, 'Python for Beginner');
+-----------------------------------+
| Message                           |
+-----------------------------------+
| Book returned late with fine: Rs.10 |
+-----------------------------------+
1 row in set (0.32 sec)
```

Test Case 3:
Book Returned After More Than 30 Days
(Fine Rs. 5 per Day + Rs. 50 per Day accordingly)

```
CALL ReturnBook(1, 'Data Science');
+-----------------------------------+
| Message                           |
+-----------------------------------+
| Book returned late with fine: Rs.60 |
+-----------------------------------+
1 row in set (1.06 sec)
```

Error Test Case 1:
Book Already Returned

```
CALL ReturnBook(3, 'AI Basics');
+-------------------------------------+
| Message                             |
+-------------------------------------+
| Book not issued or already returned. |
+-------------------------------------+
1 row in set (0.00 sec)
```

Error Test Case 2:
Book Not Found for Given Roll_no and Book Name

```
CALL ReturnBook(99, 'Unknown Book');
+-------------------------------------+
| Message                             |
+-------------------------------------+
| Book not issued or already returned. |
+-------------------------------------+
1 row in set (0.00 sec)
```

```
SELECT * FROM Borrower;
+---------+--------+-------------+---------------------+--------+
| Roll_no | Name   | DateofIssue | NameofBook          | Status |
+---------+--------+-------------+---------------------+--------+
|       1 | Anjali | 2025-07-10  | Data Science        | R      |
|       2 | Ravi   | 2025-07-20  | Python for Beginner | R      |
|       3 | Suman  | 2025-07-30  | AI Basics           | R      |
+---------+--------+-------------+---------------------+--------+
3 rows in set (0.00 sec)
```

1) CREATE TABLES

```
CREATE TABLE areas(
    radius INT NOT NULL,
    area DOUBLE
);

DESC areas;
```

```
+--------+--------+------+-----+---------+-------+
| Field  | Type   | Null | Key | Default | Extra |
+--------+--------+------+-----+---------+-------+
| radius | int    | NO   |     | NULL    |       |
| area   | double | YES  |     | NULL    |       |
+--------+--------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

2) PL/SQL Block

```
DELIMITER $$
CREATE PROCEDURE CalculateAreas(
    IN start_radius INT,
    IN end_radius INT
)
BEGIN
    DECLARE r INT;
    DECLARE a DOUBLE;
    DELETE FROM areas;
    SET r = start_radius;
    area_loop: WHILE r <= end_radius DO
        SET a = PI() * r * r;
        INSERT INTO areas(radius, area) VALUES(r,a);
        SET r = r + 1;
    END WHILE area_loop;
    SELECT * FROM areas;

END $$

DELIMITER ;
```

4) EXECUTE THE PROCEDURE

```
CALL CalculateAreas(5,9);
```

```
+--------+--------------------+
| radius | area               |
+--------+--------------------+
|      5 |  78.53981633974483 |
|      6 | 113.09733552923255 |
|      7 | 153.93804002589985 |
|      8 | 201.06192982974676 |
|      9 | 254.46900494077323 |
+--------+--------------------+
5 rows in set (0.80 sec)
```