DATABASE SYSTEM-6350

ASSIGNMENT 3

MACQUARIE University

FACULTY OF SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

COMP2350/COMP6350:

Database Systems Session 2, 2023

Assignment Three: Procedural Programming

Unit Code	COMP6350	Assignment#	3
Student ID Number	47943319	Student Name	NIYATI NIYATI
Tutor's Name	Rafiullah (Rafi) Khan	Workshop Date/Time	11:00am-1:00pm(FRI)

TASK 1:

Code: To update the Member table to allow logging fine fees for overdue Below mentioned procedure can be used for the same:

```
DELIMITER //

DROP PROCEDURE IF EXISTS UpdateFineFeesForOverdue ;

CREATE PROCEDURE UpdateFineFeesForOverdue()
BEGIN
    DECLARE member_id_val INT;
    DECLARE return_due_date_val DATE;
    DECLARE date_returned_val DATE;
    DECLARE fine_fee_val DECIMAL(10, 2);
    DECLARE days_overdue INT;

    DECLARE cur CURSOR FOR
        SELECT MemberID, ReturnDueDate, DateReturned
        FROM Borrowedby
        WHERE DateReturned > ReturnDueDate;

    IF NOT EXISTS (
        SELECT * FROM information_schema.COLUMNS
        WHERE TABLE_NAME = 'Member' AND COLUMN_NAME = 'FineFee'
    ) THEN
        ALTER TABLE Member
        ADD COLUMN FineFee DECIMAL(10, 2) DEFAULT 0;
    END IF;

    OPEN cur;

    read_loop: LOOP
    FETCH cur INTO member_id_val, return_due_date_val, date_returned_val;
    IF date_returned_val IS NOT NULL THEN
        SET days_overdue = DATEDIFF(date_returned_val, return_due_date_val);
        SET fine_fee_val = days_overdue * 2; -- Assuming a fine of $2 per overdue day

        -- Debugging statement to check the values
        SELECT member_id_val, days_overdue, fine_fee_val;
```

```
        -- Update Member
        UPDATE Member
        SET FineFee = FineFee + fine_fee_val
        WHERE MemberID = member_id_val;

        -- Debugging statement to verify the update
        SELECT * FROM Member WHERE MemberID = member_id_val;
    END IF;

    END LOOP read_loop;
 CLOSE cur;
END //

DELIMITER ;
```

```
1   DELIMITER //
2
3   DROP PROCEDURE IF EXISTS UpdateFineFeesForOverdue ;
4
5   CREATE PROCEDURE UpdateFineFeesForOverdue()
6   BEGIN
7       DECLARE member_id_val INT;
8       DECLARE return_due_date_val DATE;
9       DECLARE date_returned_val DATE;
10      DECLARE fine_fee_val DECIMAL(10, 2);
11      DECLARE days_overdue INT;
12
13      DECLARE cur CURSOR FOR
14          SELECT MemberID, ReturnDueDate, DateReturned
15          FROM Borrowedby
16          WHERE DateReturned > ReturnDueDate;
```

Output

Action Output

#	Time	Action	Message
359	16:01:07	DROP PROCEDURE IF EXISTS UpdateFineFeesForOverdue	0 row(s) affected
360	16:01:07	CREATE PROCEDURE UpdateFineFeesForOverdue() BEGIN DECLARE memb...	0 row(s) affected
361	16:02:02	CALL UpdateFineFeesForOverdue()	Error Code: 1329. No data - zero rows fetched, selected
362	16:02:42	DROP PROCEDURE IF EXISTS UpdateFineFeesForOverdue ; CREATE PROCE...	0 row(s) affected

1. First in the procedure built I have declarated the variables for member_id_val, return_due_date_val, date_returned_val, fine_fee_val, and days_overdue.
2. Then sets up a cursor cur to fetch the MemberID, ReturnDueDate, and DateReturned from the Borrowedby table where the DateReturned is greater than the ReturnDueDate.
3. Then the procedure checks whether the FineFee column exists in the Member table and adds the column with default settings if it does not exist.
4. It then opens the cursor and initializes a LOOP to iterate through the results.

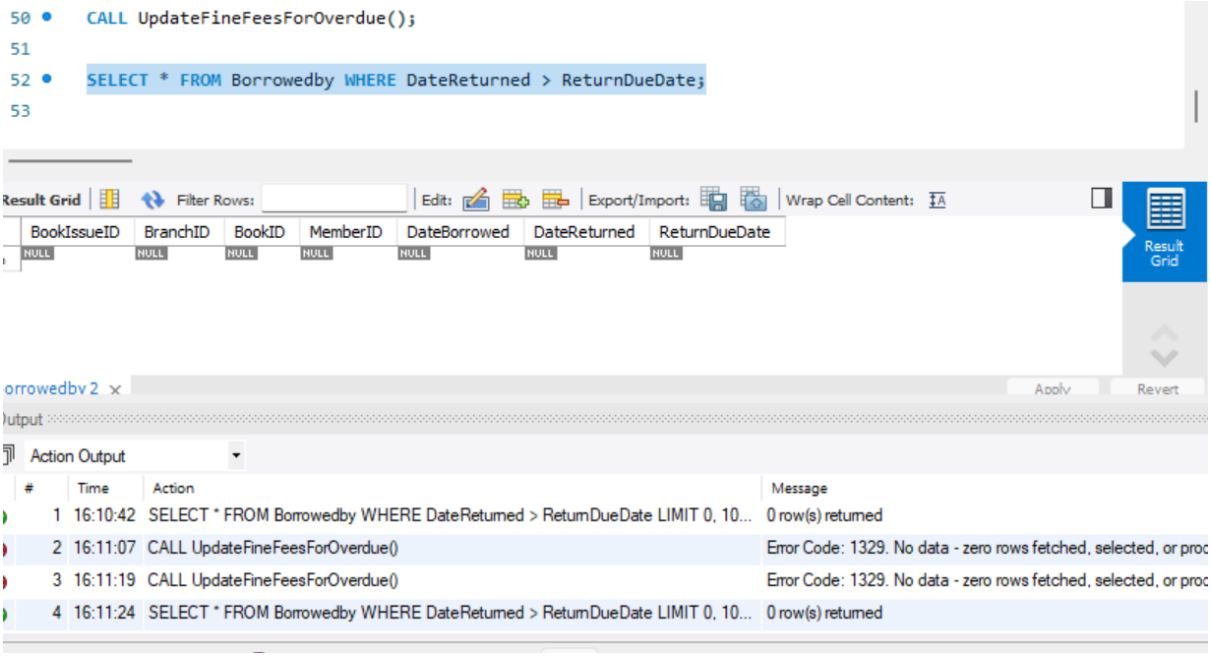
5. Within the loop, it calculates the days_overdue and the fine_fee_val based on the overdue days, and updates the FineFee for the respective MemberID.
6. I have also added debug statements to check the values and the update within the loop.
7. Finally, at end I have closed the cursor.

To verify the correctness of the above operations. :

CALL UpdateFineFeesForOverdue(); --call the procedure

SELECT * FROM Borrowedby --then to verify I have selected the overdue cases by using select and where code

WHERE DateReturned > ReturnDueDate;



As there is no data where overdue exists therefore called procedure doesn't changes any row.

For this purpose, we need to insert the data.

Code:

```
INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('17', 'Josh', 'Rousehill', 'north', 'NSW', '2025-10-31');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('18', 'Ram', 'Stringer rd', 'North', 'NSW', '2025-09-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('19', 'shyam', 'Beaumount hill', 'North', 'NSW', '2025-11-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('20', 'sita', 'Penant hill', 'North', 'NSW', '2025-03-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('21', 'greece', 'macquaire ', 'south', 'NSW', '2025-09-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('22', 'friend', 'brisbane', 'brisbane', 'QLD', '2025-09-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('23', 'carl', 'northwest', 'northwest', 'NSW', '2025-09-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('24', 'niyati', 'baukham hills', 'north', 'NSW', '2025-10-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('25', 'rolls', 'kellyville', 'north', 'NSW', '2025-11-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('26', 'royce', 'mardsen park', 'north ryde', 'NSW', '2025-11-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('27', 'jonathan', 'bella vista', 'north', 'NSW', '2025-11-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('28', 'william', 'chatswood', 'south', 'NSW', '2025-05-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('29', 'stella', 'sydney city', 'city', 'NSW', '2025-12-30');

INSERT INTO `6350_assignment3`.`member` (`MemberID`, `MemberName`, `MemberAddress`, `MemberSuburb`, `MemberState`, `MemberExpDate`) VALUES ('30', 'rose', 'castle hill', 'north', 'NSW', '2025-09-30');
```

```
INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('16', 'the friends', '1', '12.22');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('17', 'sheep', '1', '12.33');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('18', 'data science', '1', '15.43');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('19', 'big data', '1', '16.08');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('20', 'machine learning', '2', '190.03');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('21', 'god', '3', '145.93');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('22', 'behaviour', '2', '104.39');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('23', 'once upon a time', '2', '20.55');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('24', 'yes day', '3', '34.55');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('25', 'greenhouse academy', '3', '92.30');

INSERT INTO `6350_assignment3`.`Book` (`BookID`, `BookTitle`, `PublisherID`, `Price`) VALUES ('26', 'two tales', '2', '100.11');



INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('1', '16', '3', '2');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('1', '17', '3', '2');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('1', '18', '3', '1');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('1', '19', '3', '1');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('1', '20', '3', '1');
```

```
INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('2', '21', '3', '2');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('2', '22', '3', '1');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('2', '23', '3', '1');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('2', '24', '3', '1');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('2', '25', '3', '1');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('3', '26', '3', '1');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('3', '16', '3', '1');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('3', '17', '3', '1');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('3', '18', '3', '0');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('3', '19', '3', '0');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('3', '20', '3', '0');

INSERT INTO `6350_assignment3`.`Holding` (`BranchID`, `BookID`, `InStock`, `OnLoan`) VALUES ('3', '21', '3', '0');


INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('18', '1', '16', '17', '2022-09-15', '2022-10-20', '2022-10-15');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('19', '1', '17', '18', '2022-09-16', '2022-10-14', '2022-10-16');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('20', '1', '18', '19', '2022-10-21', '2022-11-30', '2022-11-21');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('21', '2', '21', '20', '2021-08-15', '2021-09-30', '2021-09-15');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('22', '2', '22', '21', '2022-10-11', '2022-11-11', '2022-11-11');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('23', '2', '23', '22', '2022-07-12', '2022-08-10', '2022-08-12');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('24', '3', '26', '23', '2022-10-21', '2022-11-20', '2022-11-21');
```

```
INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('25', '3', '16', '17', '2023-03-21', '2023-04-30', '2023-04-21');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('26', '3', '17', '25', '2020-08-15', '2020-09-15', '2020-09-15');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('27', '1', '19', '26', '2020-01-03', '2020-01-29', '2020-02-03');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('28', '1', '20', '19', '2023-05-03', '2023-06-10', '2023-06-03');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('29', '1', '16', '17', '2023-04-16', '2023-05-26', '2023-05-16');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('30', '2', '24', '29', '2023-03-10', '2023-04-10', '2023-04-10');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('31', '2', '25', '30', '2021-02-18', '2023-02-08', '2021-03-28');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('32', '2', '21', '17', '2023-02-18', '2023-03-08', '2023-02-28');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('33', '3', '18', '18', '2022-10-13', '2022-11-13', '2022-11-13');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('34', '3', '19', '19', '2021-10-13', '2021-11-15', '2021-11-13');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('35', '3', '20', '22', '2022-11-13', '2022-12-13', '2022-12-13');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('36', '3', '21', '20', '2022-10-13', '2022-11-17', '2022-11-13');

INSERT INTO `6350_assignment3`.`Borrowedby` (`BookIssueID`, `BranchID`, `BookID`, `MemberID`, `DateBorrowed`, `DateReturned`, `ReturnDueDate`) VALUES ('37', '1', '17', '21', '2020-12-13', '2021-01-10', '2021-01-13');
```

NOW,again call the procedure to show fine fee overdue- The result can be seen in different tabs

```
1 •  CALL `UpdateFineFeesForOverdue`;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MemberID	MemberStatus	MemberName	MemberAddress	MemberSuburb	MemberState	MemberExpDate	MemberPhone	FineFee
17	REGULAR	Josh	Rousehill	north	NSW	2025-10-31	NULL	266.00

Result 21 | Result 22 × | Result 23 | Result 24 | Result 25 | Result 26 | Result 27 | Result 28 | Result 29 | Result 30

```
1 •  CALL `UpdateFineFeesForOverdue`;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MemberID	MemberStatus	MemberName	MemberAddress	MemberSuburb	MemberState	MemberExpDate	MemberPhone	FineFee
19	REGULAR	shyam	Beaumount hill	North	NSW	2025-11-30	NULL	162.00

Result 21 | Result 22 | Result 23 | Result 24 × | Result 25 | Result 26 | Result 27 | Result 28 | Result 29 | Result 30

```
1 •  CALL `UpdateFineFeesForOverdue`;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

MemberID	MemberStatus	MemberName	MemberAddress	MemberSuburb	MemberState	MemberExpDate	MemberPhone	FineFee
20	REGULAR	sita	Penant hill	North	NSW	2025-03-30	NULL	182.00

Result 21 | Result 22 | Result 23 | Result 24 | Result 25 | Result 26 × | Result 27 | Result 28 | Result 29 | Result 30

TASK 2:

Write a trigger to implement the BR8 business rule listed on page 2. Exceptions must be handled by error handlers.

Code:

```
DELIMITER //

DROP TRIGGER IF EXISTS update_member_status_trigger;

CREATE TRIGGER update_member_status_trigger

BEFORE UPDATE ON Member

FOR EACH ROW

BEGIN

   DECLARE returned_count INT;

   IF NEW.FineFee = 0 THEN

      SET returned_count = (SELECT COUNT(*) FROM Borrowedby WHERE MemberID = NEW.MemberID AND
DateReturned IS NOT NULL);

      IF returned_count > 0 THEN

         SET NEW.MemberStatus = 'REGULAR';

      ELSE

         SIGNAL SQLSTATE '45000'

         SET MESSAGE_TEXT = 'No records of book returned back by the Member.';

      END IF;

   END IF;

END//


DELIMITER;
```

```
DELIMITER //
DROP TRIGGER IF EXISTS update_member_status_trigger;
CREATE TRIGGER update_member_status_trigger
BEFORE UPDATE ON Member
FOR EACH ROW
BEGIN
    DECLARE returned_count INT;
    IF NEW.FineFee = 0 THEN
        SET returned_count = (SELECT COUNT(*) FROM Borrowedby WHERE MemberID = NEW.MemberID AND DateReturn
        IF returned_count > 0 THEN
            SET NEW.MemberStatus = 'REGULAR';
        ELSE
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'No records of book returned back by the Member.';
        END IF;
    END IF;
END//
```

Output

Action Output

#	Time	Action	Message
1	12:05:58	DROP TRIGGER IF EXISTS update_member_status_trigger; CREATE TRIGGER upd...	0 row(s) affected

1. First ,drops the trigger update_member_status_trigger if it already exists to avoid conflicts.
2. It creates the trigger, specifying that it should execute BEFORE UPDATE ON Member for each row.
3. Within the trigger block, it declares the variable returned_count to store the count of returned books by the member.
4. then checks whether the FineFee for the updated member is 0.
5. If the FineFee is 0, the trigger counts the number of records in the Borrowedby table where the MemberID matches the NEW.MemberID and the DateReturned is not NULL.
6. If records are found, it sets the MemberStatus of the updated row to 'REGULAR'.
7. If no records are found, the trigger raises a custom exception with the message 'No records of book returned back by the Member.

Task 3

Write a stored procedure to list the members that currently have an overdue item and their (individual) membership has been suspended twice in the past three years. End these members' membership by seeing their MemberStatus to "TERMINATED". Error handler must be implemented to handle exceptions.

for this procedure I have increased member status from char (9) to more so that the 'terminated' word can be adjusted.

```
DELIMITER //


DROP PROCEDURE IF EXISTS EndMembershipIfSuspendedTwice;


CREATE PROCEDURE EndMembershipIfSuspendedTwice()
BEGIN
  DECLARE suspendedCount INT;


  DECLARE memberCursor CURSOR FOR
  SELECT m.MemberID
  FROM Member m
  WHERE m.MemberID IN (
    SELECT b.MemberID
    FROM Borrowedby b
    WHERE b.DateReturned > b.ReturnDueDate
  )
  AND m.MemberID IN (
    SELECT MemberID
    FROM (
      SELECT MemberID, COUNT(*) as count_returned_date
      FROM Borrowedby
      WHERE DateReturned > ReturnDueDate AND DateReturned >= DATE_SUB(CURDATE(), INTERVAL 3 YEAR)
```

```
        GROUP BY MemberID
      ) as returns
      WHERE returns.count_returned_date > 2
    );

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET @done = TRUE;

    OPEN memberCursor;

    memberLoop: LOOP
      FETCH memberCursor INTO suspendedCount;
      IF @done THEN
        LEAVE memberLoop;
      END IF;

      UPDATE Member
      SET MemberStatus = 'TERMINATED'
      WHERE MemberID = suspendedCount;
    END LOOP;

    CLOSE memberCursor;
END //

DELIMITER ;
```

1. First we drop the procedure EndMembershipIfSuspendedTwice if it already exists to avoid conflicts.
2. Inside the stored procedure, it declares the variable suspendedCount to store the count of suspended members.

3. It declares a cursor member Cursor that selects the MemberID of members who have returned books late and have been suspended more than twice in the past three years.
4. A CONTINUE HANDLER is defined to set the variable @done to TRUE if no more rows are found in the cursor.
5. The cursor is opened, and a loop memberLoop is initiated to fetch the MemberID into the suspendedCount variable.
6. If the @done variable is set to TRUE, the loop is exited.
7. For each fetched MemberID, the procedure updates the MemberStatus to 'TERMINATED'.
8. Finally, the cursor is closed.

```
DELIMITER //
DROP PROCEDURE IF EXISTS EndMembershipIfSuspendedTwice;
CREATE PROCEDURE EndMembershipIfSuspendedTwice()
BEGIN
    DECLARE suspendedCount INT;

    DECLARE memberCursor CURSOR FOR
    SELECT m.MemberID
    FROM Member m
    WHERE m.MemberID IN (
        SELECT b.MemberID
        FROM Borrowedby b
        WHERE b.DateReturned > b.ReturnDueDate
```

Output

Action Output

#	Time	Action	Message
1	13:26:58	DROP PROCEDURE IF EXISTS EndMembershipIfSuspendedTwice; CREATE PROC...	0 row(s) affected

TASK 4

Task2 :

case 1-Updating fine fee to zero and member has returned all the books i.e no outstanding books left

member id 20 earlier used to have fine which when updated to 0 ,1 row is affected and then trigger works therefore suspended changes to

```
UPDATE member
SET FineFee = 0
WHERE MemberID = 20;
SELECT * FROM member
WHERE MemberID = 20;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

MemberID	MemberStatus	MemberName	MemberAddress	MemberSuburb	MemberState	MemberExpDate	MemberPhone	FineFee
20	REGULAR	sita	Penant hill	North	NSW	2025-03-30	NULL	0.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Form Editor

member 1 | Apply | Revert

Output

Action Output

#	Time	Action	Message
9	11:37:44	DROP TRIGGER IF EXISTS update_member_status_trigger; CREATE TRIGGER u...	0 row(s) affected
10	11:37:51	UPDATE member SET FineFee = 0 WHERE MemberID = 20	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
11	11:38:16	SELECT * FROM member WHERE MemberID = 20 LIMIT 0, 1000	1 row(s) returned

CASE2: As in the given data member ID 6 showed suspended membership but the returned due date was null that shows book has not been returned therefore in clearing the fine also the status doesn't changes and shows record not found for the book returned. Error code:1644

```
-- 2. Update the FineFee to 0
UPDATE Member
SET FineFee = 0
WHERE MemberID =6;

-- Check if the trigger has updated the MemberStatus to 'REGULAR'
SELECT *
FROM Member
WHERE MemberID = 6;
```

Context

Output

Action Output

#	Time	Action	Message
1	12:05:58	DROP TRIGGER IF EXISTS update_member_status_trigger; CREATE TRIGGER upd...	0 row(s) affected
2	12:07:06	UPDATE Member SET FineFee = 0 WHERE MemberID =6	Error Code: 1644. No records of book returned back by the Member.

2. Terminated task

Case 1: when there is no data of suspended as the returned date is not mentioned in the data provided to populate there will be no result if we call the procedure.

```
CALL EndMembershipIfSuspendedTwice ;
```

Output

Action Output

#	Time	Action	Message
24	13:33:40	CALL EndMembershipIfSuspendedTwice	0 row(s) affected

```
SELECT * FROM member
WHERE MemberStatus='TERMINATED';
```

Result Grid

MemberID	MemberStatus	MemberName	MemberAddress	MemberSuburb	MemberState	MemberExpDate	MemberPhone
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Case2: when data is inserted in which 17,19,20 were having fine out of which member id:20 has paid the fine as assumed in the previous trigger so status is changed to regular. But others having more than twice suspended are converted into terminated.

```
SELECT * FROM `member`
WHERE MemberStatus='TERMINATED';
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

MemberID	MemberStatus	MemberName	MemberAddress	MemberSuburb	MemberState	MemberExpDate	MemberPhone	FineFee
17	TERMINATED	Josh	Rousehill	north	NSW	2025-10-31	NULL	448.00
19	TERMINATED	shyam	Beaumount hill	North	NSW	2025-11-30	NULL	252.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Report:Present the schema for the tables you created.

I have not created any additional table but my member table consists a new column fine fee due to call of procedure in task 1 .

```
Branch->BranchID(PK),BranchSuburb,BranchState


Member->MemberID(PK),MemberStatus,MemberName,MemberAddress,
MemberSuburb, MemberState, MemberExpDate, MemberPhone

Publisher->PublisherID (PK), PublisherName,PublisherAddress

Book->BookID (PK), BookTitle,PublisherID, PublishedYear,Price


Author->AuthorID(PK),AuthorName,AuthorAddress,Authoredby,BookID(PK,FK),
AuthorID(PK,FK)

Holding->BranchID(PK,FK), BookID(PK,FK), InStock,OnLoan

Borrowed by->BookIssueID(PK),BranchID(PK,FK),BookID(PK,FK),MemberID(PK,FK),
  DateBorrowed,DateReturned,ReturnDueDate
```