Unit Code	COMP2350	Assignment#	3
Student ID	46293396	Student Name	Thi Ngoc Trinh Nguyen
Tutor's Name	Rohitranjan Vasantkumar (Rohit) Gupta	Workshop Date / Time	Thursday, 1:00 – 3:00

Part A: Database Programming and Implemention

Part B: Report Writing

Table Schema I created:

1. Branch (BranchID [PK], BranchSuburb, BranchState)

2. Member (MemberID [PK], MemberStatus, MemberName, MemberAddress, MemberSuburb, MemberState, MemberExpDate, MemberPhone)

3. Publisher (PublisherID [PK], PublisherName, PublisherAddress)

4. Book (BookID [PK], BookTitle, PublisherID [FK], PublishedYear, Price)

5. Author (AuthorID [PK], AuthorName, AuthorAddress)

6. Authoredby (BookID [FK], AuthorID [FK])

7. Holding (BranchID [FK], BookID [FK], InStock, OnLoan)

8. Borrowedby (BookIssueID [PK], BranchID [FK], BookID [FK], MemberID [FK], DateBorrowed, DateReturned, ReturnDueDate)

Task1:

First we will check the member table:

Select * from member;

We have the data as following table:

MemberID	MemberStatus	MemberName	MemberAddress	MemberSuburb	MemberState	MemberExpDate	MemberPhone
1	REGULAR	Joe	4 Nowhere St	Here	NSW	2021-09-30	0434567811
2	REGULAR	Pablo	10 Somewhere St	There	ACT	2022-09-30	0412345678
3	REGULAR	Chen	23/9 Faraway Cl	Far	QLD	2020-11-30	0412346578
4	REGULAR	Zhang	Dunno St	North	NSW	2020-12-31	
5	REGULAR	Saleem	44 Magnolia St	South	SA	2020-09-30	1234567811
6	SUSPENDED	Homer	Middle of Nowhere	North Ryde	NSW	2020-09-30	1234555811
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Now we add Log Fee colume due to the task: **Update the Member table to allow logging fine fees for overdue.**

The SQL script for it:

ALTER TABLE Member ADD FineFees DECIMAL(10, 2) DEFAULT 0.00;

Then Select * from member; again you will see new column added

MemberID	MemberStatus	MemberName	MemberAddress	MemberSuburb	MemberState	MemberExpDate	MemberPhone	FineFees
1	REGULAR	Joe	4 Nowhere St	Here	NSW	2021-09-30	0434567811	0.00
2	REGULAR	Pablo	10 Somewhere St	There	ACT	2022-09-30	0412345678	0.00
3	REGULAR	Chen	23/9 Faraway Cl	Far	QLD	2020-11-30	0412346578	0.00
4	REGULAR	Zhang	Dunno St	North	NSW	2020-12-31		0.00
5	REGULAR	Saleem	44 Magnolia St	South	SA	2020-09-30	1234567811	0.00
6	SUSPENDED	Homer	Middle of Nowhere	North Ryde	NSW	2020-09-30	1234555811	0.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Or better way you can use DESC Member; to show the table structure:

Field	Type	Null	Key	Default	Extra
MemberID	int	NO	PRI	NULL	
MemberStatus	char(9)	YES		REGULAR	
MemberName	varchar(255)	NO		NULL	
MemberAddress	varchar(255)	NO		NULL	
MemberSuburb	varchar(25)	NO		NULL	
MemberState	char(3)	NO		NULL	
MemberExpDate	date	YES		NULL	
MemberPhone	varchar(10)	YES		NULL	
FineFees	decimal(10,2)	YES		0.00	

As you can see the new field was added in the member table.

However, since it is default so every fee is 0.00.

According to BR6 and BR7:

BR6. If a member has an outstanding fee* and it has reached $30, his/her membership will be suspended.
BR7. If a member has an overdue item, his/her fine fee will be increased $2/day passing the expiraUon date and the membership will be suspended.

So here is the additional script to update the fee so it match the rules:
(in real life this should me run daily)

```
UPDATE Member m
JOIN (
    SELECT bb.MemberID,
        COUNT(*) * 2 AS DailyFine -- $2 for each overdue book
    FROM Borrowedby bb
    WHERE bb.DateReturned IS NULL AND bb.ReturnDueDate < CURDATE()
    GROUP BY bb.MemberID
) AS OverdueFees ON m.MemberID = OverdueFees.MemberID
SET m.FineFees = m.FineFees + OverdueFees.DailyFine;
```

The above script will update the fee. After run that script you can run Select * from member to check:

MemberID	MemberStatus	MemberName	MemberAddress	MemberSuburb	MemberState	MemberExpDate	MemberPhone	FineFees
1	REGULAR	Joe	4 Nowhere St	Here	NSW	2021-09-30	0434567811	2.00
2	REGULAR	Pablo	10 Somewhere St	There	ACT	2022-09-30	0412345678	4.00
3	REGULAR	Chen	23/9 Faraway Cl	Far	QLD	2020-11-30	0412346578	0.00
4	REGULAR	Zhang	Dunno St	North	NSW	2020-12-31		0.00
5	REGULAR	Saleem	44 Magnolia St	South	SA	2020-09-30	1234567811	0.00
6	SUSPENDED	Homer	Middle of Nowhere	North Ryde	NSW	2020-09-30	1234555811	0.00
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Then you need to suspend member with over $30:

```
UPDATE Member
SET MemberStatus = 'SUSPENDED'
WHERE FineFees >= 30 AND MemberStatus = 'REGULAR';
```

since nobody has exceed it according the previous table so nothing change here. But in real life you might run it daily and it will update it.

Task 2:

We need a trigger for BR8.

BR8 said: When a suspended member clears their fine (i.e, paid all the outstanding fees) and has no or has returned all overdue items, reset the member's membership status to "REGULAR

To determine if a member has overdue items, we'd check the Borrowedby table's DateReturned and ReturnDueDate columns.

The conditions to check is:

- Check if the fine has been cleared (fee is 0.00).
- Check if the member has no overdue books.
- If both conditions are met, reset the member status to 'REGULAR'.

The script for it is:

```
DELIMITER //

CREATE TRIGGER BR8_Trigger
BEFORE UPDATE ON Member
FOR EACH ROW
BEGIN
    DECLARE hasOverdue INT DEFAULT 0;

    -- Error Handler
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
        -- Return an error message
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'An error occurred while updating the Member Status.';
    END;

    -- Check if there are overdue items for the member
    SELECT COUNT(*)
    INTO hasOverdue
    FROM Borrowedby
```

```
    WHERE MemberID = NEW.MemberID AND DateReturned IS NULL AND
ReturnDueDate < CURDATE();

    -- If the FineFees are 0.00 and there are no overdue items, set the MemberStatus to
REGULAR
    IF NEW.FineFees = 0.00 AND hasOverdue = 0 THEN
        SET NEW.MemberStatus = 'REGULAR';
    END IF;

END //

DELIMITER ;
```

The DELIMITER is for wrap the code because in side it also have the semi colon “;”.

Then you create Trigger and activate when there is an update for member table (each row)

There is also error handler for SQLEXCEPTION

After all of that we start to check the overdue item with the script:

```
SELECT COUNT(*)
INTO hasOverdue
FROM Borrowedby
WHERE MemberID = NEW.MemberID AND DateReturned IS NULL AND ReturnDueDate
< CURDATE();
```

This SQL query counts the number of overdue items for the updated member. It checks for borrowed items that haven't been returned (DateReturned IS NULL) and whose return due date has passed (ReturnDueDate < CURDATE()). The result is stored in the hasOverdue variable.

Now we update the member status:

```
  IF NEW.FineFees = 0.00 AND hasOverdue = 0 THEN
        SET NEW.MemberStatus = 'REGULAR';
    END IF;
```

This block checks two conditions:

The fine fees for the updated member are 0 (NEW.FineFees = 0.00).
The member has no overdue items (hasOverdue = 0).
If both conditions are true, the member's status is updated to REGULAR.

Task 3

we need: 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 sefng their
MemberStatus to "TERMINATED". And also handle the exception

I notice there is no way to know who has been suspended twice in last three years so we need to create new table for it

```
CREATE TABLE MemberHistory (
   HistoryID INT NOT NULL AUTO_INCREMENT,
   MemberID INT NOT NULL,
   OldStatus VARCHAR(255) NOT NULL,
   NewStatus VARCHAR(255) NOT NULL,
   ChangeDate DATE NOT NULL,
   PRIMARY KEY (HistoryID),


   CONSTRAINT member_fk_3 FOREIGN KEY (MemberID) REFERENCES Member (MemberID) ON DELETE CASCADE
);
```

This to store data for user to know the status change they have.

Here is the script for the above task:

```
DELIMITER //

CREATE PROCEDURE TerminateOverdueMembers()
BEGIN
    DECLARE currentDate DATE;
    SET currentDate = CURDATE();

    -- Drop temporary tables if they exist
    DROP TEMPORARY TABLE IF EXISTS TempOverdueMembers;
    DROP TEMPORARY TABLE IF EXISTS TempSuspendedMembers;

    -- Find members who have overdue items
    CREATE TEMPORARY TABLE TempOverdueMembers AS
    SELECT DISTINCT MemberID
    FROM Borrowedby
    WHERE DateReturned IS NULL
    AND currentDate > ReturnDueDate;

    -- Find members who were suspended twice or more in the last three years using MemberHistory
    CREATE TEMPORARY TABLE TempSuspendedMembers AS
    SELECT MemberID
    FROM MemberHistory
    WHERE NewStatus = 'SUSPENDED'
    AND ChangeDate BETWEEN DATE_SUB(currentDate, INTERVAL 3 YEAR) AND currentDate
    GROUP BY MemberID
    HAVING COUNT(MemberID) >= 2;

    -- Update the status of the members who meet both criteria to 'TERMINATED'
    UPDATE Member
    SET MemberStatus = 'TERMINATED'
    WHERE MemberID IN (
        SELECT o.MemberID
        FROM TempOverdueMembers o
```

```
        JOIN TempSuspendedMembers s ON o.MemberID = s.MemberID
    );

    -- Drop temporary tables
    DROP TEMPORARY TABLE IF EXISTS TempOverdueMembers;
    DROP TEMPORARY TABLE IF EXISTS TempSuspendedMembers;

END //

DELIMITER ;
```

Again the DELIMITER is to wrap the code so it allow “;” in the code block inside.

First you create the procedure.

Then you handle the exception, If any SQL exception occurs while the procedure is running, the changes made inside this procedure will be reverted using the ROLLBACK command. This ensures that if part of the procedure fails, any changes made before that point aren't saved to the database.

Then you Start Transaction; To allow sequence of one or more SQL operations executed as a single logical unit of work. The benefit is that you can ensure that a series of operations are all successful before making permanent changes to the database. If any operation within the transaction fails, you can revert all the changes using the ROLLBACK command.

Now you Update Members Status:

```
UPDATE Member
    SET MemberStatus = 'TERMINATED'
    WHERE MemberID IN (
        SELECT o.MemberID
        FROM TempOverdueMembers o
        JOIN TempSuspendedMembers s ON o.MemberID = s.MemberID
    );
```

The above code updates the MemberStatus to 'TERMINATED' for members that meet specific criteria:
- Have overdue items.
- Have a 'SUSPENDED' status.
- Have been suspended twice or more in the past three years.
The logic for each of these criteria is nested within the WHERE clause.

Then commit to close transaction

Task4:

After creating the Trigger for task2 and Procedure for task 3. It's time to test it:

Task2-Test (Testing the Trigger from Task 2):
Test Plan:

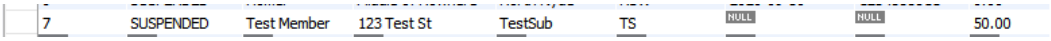
1) Scenario 1 (Overdue Item with Fine Reset):

- Description: A member who has an overdue book but has paid their fine fees.
- Expected Result: Their MemberStatus should not be set to 'REGULAR' after updating the fine fees to 0 because he have overdue book
- Test Data:
  MemberID: 7
  FineFees: 50
  Overdue books: 1

Testing:
First insert that member and see the current data:

INSERT INTO Member (MemberID, MemberStatus, MemberName, MemberAddress, MemberSuburb, MemberState, MemberExpDate, MemberPhone, FineFees)
VALUES ('7', 'REGULAR', 'TestMember1', '123 Test St', 'TestTown', 'NSW', '2024-12-31', '1234567890', 50.00);
select * from member

We will have



7	SUSPENDED	Test Member	123 Test St	TestSub	TS	NULL	NULL	50.00

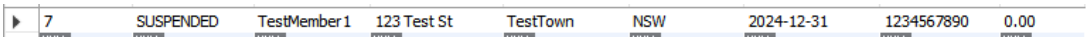
Then we add borrow so this user have overdue item:

INSERT INTO Borrowedby (BranchID, BookID, MemberID, DateBorrowed, DateReturned, ReturnDueDate)
VALUES ('1', '1', '7', '2023-01-01', NULL, '2023-01-14');

Now update the fine fees to see if it getting back status to regular

UPDATE Member SET FineFees = 0 WHERE MemberID = 7;

We have



7	SUSPENDED	TestMember1	123 Test St	TestTown	NSW	2024-12-31	1234567890	0.00

The Member Status unchange due to overdue book

2) Scenario 2 (No Overdue Item with Fine Reset):

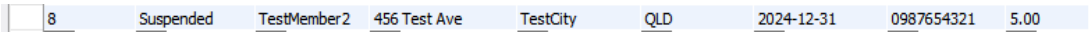
- Description: A member who doesn't have any overdue books and has paid their fine fees.
- Expected Result: Their MemberStatus remains 'REGULAR' after updating the fine fees to 0.
- Test Data:
  MemberID: 8
  FineFees: 5
  Overdue books: 0

  Insert Data and check

  INSERT INTO Member (MemberID, MemberStatus, MemberName, MemberAddress, MemberSuburb, MemberState, MemberExpDate, MemberPhone, FineFees)
  VALUES ('8', Suspended, 'TestMember2', '456 Test Ave', 'TestCity', 'QLD', '2024-12-31', '0987654321', 5.00);

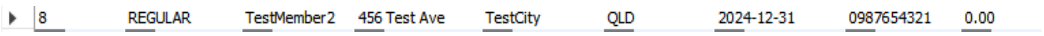
Select * from member

  Now we have:



Now update the fee. (no need added the overdue book)

Now it got back to regular:



Task3-Test (Testing the Stored Procedure from Task 3):
Test Plan:

1) Scenario 1 (Member with Overdue Items, Suspended Twice in Last 3 Years):

- Description: A member who has overdue items and has been suspended twice in the last three years.
- Expected Result: Their MemberStatus should be set to 'TERMINATED'.
- Test Data:
  MemberID: 10
  MemberStatus: 'SUSPENDED'
  Overdue books: 2
  Suspensions in Last 3 Years: 2

  First insert needed data:

  Insert member data

  INSERT INTO Member (MemberID, MemberStatus, MemberName, MemberAddress, MemberSuburb, MemberState, MemberExpDate, MemberPhone, FineFees)
  VALUES (10, 'SUSPENDED', 'TestMember3', '789 Test Road', 'TestVille', 'VIC', '2024-12-31', '1122334455', 10.00);

  .Insert data indicating this member has borrowed 2 books and both are overdue.

INSERT INTO Borrowedby (BranchID, BookID, MemberID, DateBorrowed, DateReturned, ReturnDueDate)
VALUES ('1', '2', '10', '2023-01-01', NULL, '2023-01-14');

INSERT INTO Borrowedby (BranchID, BookID, MemberID, DateBorrowed, DateReturned, ReturnDueDate)

VALUES ('1', '3', '10', '2023-01-02', NULL, '2023-01-15');

Insert history suspended:

INSERT INTO MemberHistory (MemberID, OldStatus, NewStatus, ChangeDate)
VALUES (10, 'REGULAR', 'SUSPENDED', '2021-02-01');

INSERT INTO MemberHistory (MemberID, OldStatus, NewStatus, ChangeDate)
VALUES (10, 'REGULAR', 'SUSPENDED', '2022-01-01');

The member should be:

10	SUSPENDED	TestMember3	789 Test Road	TestVille	VIC	2024-12-31	1122334455	10.00

Call the procedure and her should be:

10	TERMINATED	TestMember3	789 Test Road	TestVille	VIC	2024-12-31	1122334455	10.00

Remeber that you must

```
ALTER TABLE Member MODIFY MemberStatus CHAR(20);
```

Becasuse terminated has 10 character and default creation has 9 for member status:

```
CREATE TABLE Member (
  MemberID INT NOT NULL,
  MemberStatus char(9) DEFAULT 'REGULAR',
  MemberName varchar(255) NOT NULL,
  MemberAddress varchar(255) NOT NULL,
  MemberSuburb varchar(25) NOT NULL,
  MemberState char(3) NOT NULL,
  MemberExpDate DATE,
  MemberPhone varchar(10),
  PRIMARY KEY (`MemberID`)
);
```

2) Scenario 2 (Member without Overdue Items, Suspended Twice in Last 3 Years):

- Description: A member who doesn't have overdue items but has been suspended twice in the last three years.
- Expected Result: Their MemberStatus should remain 'SUSPENDED' or any previous state.
- Test Data:
  MemberID: 11
  MemberStatus: 'SUSPENDED'
  Overdue books: 0
  Suspensions in Last 3 Years: 2

Insert the member data for testing

```
INSERT INTO Member (MemberID, MemberStatus, MemberName, MemberAddress, MemberSuburb, MemberState, MemberExpDate, MemberPhone, FineFees)
VALUES (11, 'SUSPENDED', 'TestMember4', '321 Test Blvd', 'TestBorough', 'SA', '2024-12-31', '6655443322', 0.00);
```

Insert data in MemberHistory to indicate the member was suspended twice in the last three years.

```
INSERT INTO MemberHistory (MemberID, OldStatus, NewStatus, ChangeDate)
VALUES (11, 'REGULAR', 'SUSPENDED', '2021-02-15');

INSERT INTO MemberHistory (MemberID, OldStatus, NewStatus, ChangeDate)
VALUES (11, 'REGULAR', 'SUSPENDED', '2022-02-15');
```

The data should be

▶	11	SUSPENDED	TestMember4	321 Test Blvd	TestBorough	SA	2024-12-31	6655443322	0.00

And call the procedure it should remain

▶	11	SUSPENDED	TestMember4	321 Test Blvd	TestBorough	SA	2024-12-31	6655443322	0.00

Because it has no overdue item

3) Scenario 3 (Member with Overdue Items, Suspended Only Once in Last 3 Years):

- Description: A member who has overdue items but has been suspended only once in the last three years.
- Expected Result: Their MemberStatus should remain 'SUSPENDED' or any previous state.
- Test Data:
  MemberID: 12
  MemberStatus: 'SUSPENDED'
  Overdue books: 2
  Suspensions in Last 3 Years: 1

Insert the member data for testing

```
INSERT INTO Member (MemberID, MemberStatus, MemberName, MemberAddress, MemberSuburb, MemberState, MemberExpDate, MemberPhone, FineFees)
VALUES (12, 'SUSPENDED', 'TestMember5', '654 Test Lane', 'TestDistrict', 'TAS', '2024-12-31', '7766554433', 10.00);
```

Insert data indicating this member has borrowed 2 books and both are overdue.

```
INSERT INTO Borrowedby
(BranchID,BookID,MemberID,DateBorrowed,DateReturned,ReturnDueDate)
VALUES ('1', '1','12','2023-01-03',NULL,'2023-01-16');

INSERT INTO Borrowedby
(BranchID,BookID,MemberID,DateBorrowed,DateReturned,ReturnDueDate)
VALUES ('1', '3','12','2023-01-05',NULL,'2023-01-18');
```

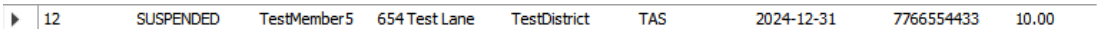
Insert data in MemberHistory to indicate the member was suspended once in the last three years.

```
INSERT INTO MemberHistory (MemberID, OldStatus, NewStatus, ChangeDate)
VALUES (12, 'REGULAR', 'SUSPENDED', '2022-03-01');
```

The data should be:

▶	12	SUSPENDED	TestMember5	654 Test Lane	TestDistrict	TAS	2024-12-31	7766554433	10.00

After call procedure it should be:



▶	12	SUSPENDED	TestMember5	654 Test Lane	TestDistrict	TAS	2024-12-31	7766554433	10.00

Because it only suspended once.