Create a trigger named trg_mem_balance that will maintain the correct value in the membership balance in the MEMBERSHIP table when videos are returned late. The trigger should execute as an AFTER trigger when the due date or return date attributes are updated in the DETAILRENTAL table. The trigger should satisfy the following conditions:

a. Calculate the value of the late fee prior to the update that triggered this execution of the trigger. The value of the late fee is the days late multiplied by the daily late fee. If the previous value of the late fee was null, then treat it as zero (0).

b. Calculate the value of the late fee after the update that triggered this execution of the trigger. If the value of the late fee is now null, then treat it as zero (0).

c. Subtract the prior value of the late fee from the current value of the late fee to determine the change in late fee for this video rental.

d. If the amount calculated in Part c is not zero (0), then update the membership balance by the amount calculated for the membership associated with this rental.

Coronel, Carlos; Morris, Steven. Database Systems: Design, Implementation, & Management (Page 436). Course Technology. Kindle Edition.

You must submit the following: A9.sql before the due date and time on Blackboard. Also submit a stapled printed copy in class on the due date. A late assignment will be assessed a penalty of 10% of the assigned points per calendar day up to 7 days. After 7 days no late assignment will be accepted.

```
USE [CIS31030]
GO
/****** Object:  Trigger [dbo].[trg_mem_balance ]
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author: TIM MAHAN
-- Create date: 11/18/16
-- Description:  A9
-- =============================================
ALTER TRIGGER [dbo].[trg_mem_balance ]
   ON  [dbo].[DETAILRENTAL]
   AFTER UPDATE
AS
BEGIN
      -- SET NOCOUNT ON added to prevent extra result sets from
      -- interfering with SELECT statements.
      SET NOCOUNT ON;
    -- Insert statements for trigger here
      --UPDATE
```

```
     IF (EXISTS(SELECT * FROM INSERTED) AND EXISTS(SELECT * FROM
DELETED))
BEGIN
--PART A
DECLARE PARTA CURSOR FOR
SELECT RENT_NUM, VID_NUM,
ISNULL(DATEDIFF(DD,Detail_ReturnDate,Detail_DueDate) *
Detail_DailyLateFee,0) AS OLDLATEFEE
FROM DELETED

--PART B
DECLARE PARTB CURSOR FOR
SELECT ISNULL(DATEDIFF(DD,Detail_ReturnDate,Detail_DueDate) *
Detail_DailyLateFee,0) AS NEWLATEFEE
FROM INSERTED

DECLARE @OLDLATEFEE DECIMAL
DECLARE @NEWLATEFEE DECIMAL
DECLARE @RENT_NUM INT
DECLARE @VID_NUM INT
DECLARE @CHANGELATEFEE DECIMAL

--PART C
OPEN PARTA
FETCH NEXT FROM PARTA
INTO @RENT_NUM, @VID_NUM, @OLDLATEFEE
OPEN PARTB
FETCH NEXT FROM PARTB
INTO @NEWLATEFEE
SET @CHANGELATEFEE = @NEWLATEFEE - @OLDLATEFEE

WHILE(@@FETCH_STATUS = 0)
BEGIN
FETCH NEXT FROM PARTA
INTO @RENT_NUM, @VID_NUM, @OLDLATEFEE
FETCH NEXT FROM PARTB
INTO @NEWLATEFEE
SET @CHANGELATEFEE = @NEWLATEFEE - @OLDLATEFEE

--PART D
UPDATE MEMBERSHIP
SET MEM_BALANCE = MEM_BALANCE + @CHANGELATEFEE
WHERE Mem_Num = (SELECT M.Mem_Num
                      FROM MEMBERSHIP M INNER JOIN RENTAL R
                      ON M.MEM_NUM = R.Mem_Num
                      INNER JOIN DETAILRENTAL DL ON DL.RENT_NUM =
R.RENT_NUM
                      WHERE R.Rent_Num = @RENT_NUM AND DL.Vid_Num =
@VID_NUM)

FETCH NEXT FROM PARTB
FETCH NEXT FROM PARTA
```

```
END

CLOSE PARTA
DEALLOCATE PARTA
CLOSE PARTB
DEALLOCATE PARTB
END
END
```