```
--TRIGGER EXAMPLE FEB 8 2017-- similar to A9
--1. Find a set of transactions for testing purposes
SELECT *
FROM ORDER_LINE
WHERE ORDER_NUM=51610 OR ITEM_NUM='FD11'
--2. Number of transactions involving the affected items
SELECT ITEM_NUM, COUNT(*) AS[ITEM COUNT]
FROM ORDER_LINE
WHERE ORDER_NUM= 51610 OR ITEM_NUM='FD11'
GROUP BY ITEM_NUM
--3. Inventory level for the affected items
SELECT ITEM_NUM, ON_HAND
FROM ITEM
WHERE ITEM_NUM IN ('FD11', 'KL78','TR40')
--4. This statement will reset all the number ordered to 10 so it is easy to check them
in testing
UPDATE ORDER_LINE
SET NUM_ORDERED=10
WHERE ORDER_NUM=51610 OR ITEM_NUM='FD11'



--5. This statement will add 10 to each order.
--this means the inventory for each item involved will be reduced by the same amount
UPDATE ORDER_LINE
SET NUM_ORDERED=NUM_ORDERED+10
WHERE ORDER_NUM=51610 OR ITEM_NUM='FD11'
--6.Same as above but reerse the effect from above
UPDATE ORDER_LINE
SET NUM_ORDERED=NUM_ORDERED-10
WHERE ORDER_NUM=51610 OR ITEM_NUM='FD11'
GO
CREATE TRIGGER [dbo].[UPDATE_ON_HAND]
ON [dbo].[ORDER_LINE]
AFTER INSERT, DELETE, UPDATE
AS
BEGIN
DECLARE @ITEM_NUM CHAR(4)
DECLARE @TOTAL INT
--INSERT CASE
IF(EXISTS(SELECT* FROM INSERTED)
       AND NOT EXISTS
       (SELECT * FROM DELETED))
BEGIN
--LOGIC TO HANDLE RECORDS IN INSERTED
DECLARE INSERTED_CURSOR CURSOR FOR
SELECT ITEM_NUM, SUM(NUM_ORDERED) AS [TOTAL]
FROM INSERTED
GROUP BY ITEM_NUM

OPEN INSERTED_CURSOR
FETCH NEXT FROM INSERTED_CURSOR
         INTO @ITEM_NUM, @TOTAL
WHILE(@@FETCH_STATUS=0)
BEGIN
UPDATE ITEM
SET ON_HAND=ON_HAND-@TOTAL
WHERE ITEM_NUM=@ITEM_NUM
```

```
FETCH NEXT FROM INSERTED_CURSOR
       INTO @ITEM_NUM, @TOTAL
END
CLOSE INSERTED_CURSOR
DEALLOCATE INSERTED_CURSOR
END
--7.DELETE CASE
IF(EXISTS (SELECT * FROM DELETED) AND NOT EXISTS(SELECT* FROM INSERTED))
BEGIN









--LOGIC TO HANDLE RECORDS IN INSERT
DECLARE DELETED_CURSOR CURSOR FOR
SELECT ITEM_NUM, SUM(NUM_ORDERED) AS [TOTAL]
FROM DELETED
GROUP BY ITEM_NUM

OPEN DELETED_CURSOR
FETCH NEXT FROM DELTETED_CURSOR
         INTO @ITEM_NUM, @TOTAL

WHILE(@@FETCH_STATUS=0)
BEGIN
UPDATE ITEM
SET ON_HAND=ON_HAND+@TOTAL
WHERE ITEM_NUM=@ITEM_NUM
FETCH NEXT FROM DELETED_CURSOR
       INTO @ITEM_NUM, @TOTAL
END
CLOSE DELETED_CURSOR
DEALLOCATE DELETED_CURSOR
END
--8. UPDATE CASE
IF(EXISTS(SELECT* FROM DELETED) AND EXISTS(SELECT * FROM INSERTED))
BEGIN
--LOGIC TO HANDLE RECORDS IN AN UPDATE CASE
DECLARE UPDATE_CURSOR CURSOR FOR
SELECT I.ITEM_NUM, SUM(I.NUM_ORDERED-D.NUM_ORDERED) AS [TOTAL]
FROM DELETED D INNER JOIN INSERTED I ON D.ORDER_NUM=I.ORDER_NUM AND D.ITEM_NUM=I.ITEM_NUM
GROUP BY I.ITEM_NUM

OPEN UPDATE_CURSOR
FETCH NEXT FROM UPDATE_CURSOR
       INTO @ITEM_NUM, @TOTAL
WHILE (@@FETCH_STATUS=0)
BEGIN
UPDATE ITEM
```

```
SET ON_HAND=ON_HAND-@TOTAL
WHERE ITEM_NUM=@ITEM_NUM
FETCH NEXT FROM UPDATE_CURSOR
	INTO @ITEM_NUM, @TOTAL
END
CLOSE UPDATE_CURSOR
DEALLOCATE UPDATE_CURSOR
END
END
--A9
DROP TRIGGER A9PRAC
CREATE TRIGGER A9PRAC
ON DETAILRENTAL
AFTER UPDATE
AS
BEGIN




--SET NOCOUNT ON added to prevent extra result sets from interfering with SELECT
statements
SET NOCOUNT ON;
DECLARE @RENT_NUM INT
DECLARE @PREVDAYSLATE INT
DECLARE @NEWDAYSLATE INT
DECLARE @PREVLATEFEE DECIMAL (5,2)
DECLARE @NEWLATEFEE DECIMAL (5,2)
DECLARE @DAILYLATEFEE DECIMAL (5,2)
--cursor to return both the old date difference and the new date difference
DECLARE DETAILCURSOR CURSOR FOR
SELECT I.RENT_NUM, DATEDIFF(DAY, D.DETAIL_DUEDATE, D.DETAIL_RETURNDATE) AS
	[PREVDAYSLATE], DATEDIFF(DAY, I.DETAIL_DUEDATE, I.DETAIL_RETURNDATE) AS
	[NEWDAYSLATE], I.DETAIL_DAILYLATEFEE
FROM INSERTED I INNER JOIN DELETED D ON I.RENT_NUM=D.RENT_NUM AND I.VID_NUM=D.VID_NUM

OPEN DETAILCURSOR
FETCH NEXT FROM DETAILCURSOR INTO @RENT_NUM, @PREVDAYSLATE, @NEWDAYSLATE, @DAILYLATEFEE
WHILE (@@FETCH_STATUS=0)
BEGIN
--If the date difference is null, treat it as 0.
IF @PREVDAYSLATE<0
	SET @PREVLATEFEE=0
ELSE
	SET @PREVLATEFEE=ISNULL(@PREVDAYSLATE,0)*@DAILYLATEFEE
IF @NEWDAYSLATE<0
	SET @NEWLATEFEE=0
ELSE
	SET @NEWLATEFEE=ISNULL(@NEWDAYSLATE,0)*@DAILYLATEFEE
--if the old late fee is different from the new late fee, find the difference; the
difference could be positive or negative.
IF((@NEWLATEFEE-@PREVLATEFEE)<>0)
BEGIN
UPDATE MEMBERSHIP
SET MEM_BALANCE = MEM_BALANCE+(@NEWLATEFEE-@PREVLATEFEE)
WHERE MEM_NUM=(SELECT MEM_NUM
		FROM RENTAL
```

```
            WHERE RENT_NUM=@RENT_NUM
            )
END
FETCH NEXT
FROM DETAILCURSOR INTO @RENT_NUM,@PREVDAYSLATE,@NEWDAYSLATE,@DAILYLATEFEE
END
CLOSE DETAILCURSOR
DEALLOCATE DETAILCURSOR
END
--test Code
--see what is in the detail table
SELECT *
FROM DETAILRENTAL
--PICK A SUBSET OF DETAIL ROWS FOR TESTING
SELECT *
FROM DETAILRENTAL
WHERE RENT_NUM IN (1004,1005)
--SET THE LATE FEE TO ONE DOLLAR SO IT IS EASY TO SEE THE EFFECT OF CHANGE
UPDATE DETAILRENTAL
SET DETAIL_DAILYLATEFEE=1
WHERE RENT_NUM IN (1004,1005)
--SET THE DUE DATES AND RETURN DATES SO THAT THEY ARE EASY TO MANIPULATE FOR TEST
PURPOSES
UPDATE DETAILRENTAL
SET DETAIL_DUEDATE='3-10-2013'
WHERE RENT_NUM IN (1004, 1005)
--FIND OUT WHO THE CUSTOMERS ARE FOR THOSE RENTALS
SELECT *
FROM MEMBERSHIP
WHERE MEM_NUM IN (SELECT MEM_NUM
                  FROM RENTAL
                  WHERE RENT_NUM IN (SELECT RENT_NUM
                                     FROM DETAILRENTAL
                                     WHERE RENT_NUM IN (1004,1005)
                                     )
                  )
--SET THE BALANCE OF THOSE CUSTOMERS TO A VALUE EASY TO TRACK
UPDATE MEMBERSHIP
SET MEM_BALANCE=10
WHERE MEM_NUM IN (110,111)
--STATEMENT TO CHECK THE RESULTS
SELECT *
FROM DETAILRENTAL
WHERE RENT_NUM IN (1004,1005)

SELECT *
FROM MEMBERSHIP
WHERE MEM_NUM IN (110,111)
--NOW THE UPDATE STATEMENTS
--INCREASE THE RETURN DATE TO CAUSE A LATE FEE
UPDATE DETAILRENTAL
SET DETAIL_RETURNDATE ='3-13-2013'
WHERE RENT_NUM IN (1004,1005)
--CHANGE THE DUE DATES BACK TO BE THE SAME AS THE DUE DATES
UPDATE DETAILRENTAL
SET DETAIL_RETURNDATE='3-10-2013'
WHERE RENT_NUM IN (1004,1005)
--CHANGE THE RETURN DATES TO LATE AGAIN
```

```
UPDATE DETAILRENTAL
SET DETAIL_RETURNDATE='3-13-2013'
WHERE RENT_NUM IN (1004,1005)
--THEN CHANGE THEM TO BEFORE THE DUE DATES
UPDATE DETAILRENTAL
SET DETAIL_RETURNDATE ='3-5-2013'
WHERE RENT_NUM IN (1004,1005)
--THEN CHANGE THE TO NULL
UPDATE DETAILRENTAL
SET DETAIL_RETURNDATE=NULL
WHERE RENT_NUM IN (1004,1005)
```

```
--A10 PART 2
--CREATING THE TABLES FOR THE DATA WAREHOUSE
--THESE DROP TABLE STATEMENTS CAN BE USED  TO DROP ALL THE TABLES SO THEY CAN BE
RECREATED IN CASE THE TABLES NEED TO BE RECREATED
DROP TABLE FACT
DROP TABLE TIMEDIM
DROP TABLE PILOTDIM
DROP TABLE PLANEDIM
DROP TABLE STAGING

CREATE TABLE TimeDim
(
	TIMEKEY INT IDENTITY NOT NULL,
	CHAR_DATE DATETIME,
	YEAR DATE,
	MONTH CHAR(15),
	QUARTER DECIMAL(2,0)
);
CREATE TABLE PilotDim
(
	PILOTKEY INT IDENTITY NOT NULL,
	EMP_NUM INT,
	EMP_FNAME CHAR(15),
	EMP_LNAME CHAR(20),
	EMP_TITLE CHAR (20),
	EMP_HIRE_DATE DATE
);
CREATE TABLE PlaneDim
(
	PLANEKEY INT IDENTITY NOT NULL,
	AC_NUMBER  NVARCHAR(5),
	MOD_CODE NVARCHAR(10),
	MOD_NAME NVARCHAR(20),
	MOD_CHG_MILE REAL
);
CREATE TABLE Fact
(
```

```
	TIMEKEY INT NOT NULL,
	PILOTKEY INT NOT NULL  ,
	PLANEKEY INT NOT NULL  ,
	CHAR_FUEL_GALLONS float,
	CHAR_DISTANCE REAL,
	REVENUE DECIMAL(10,2)
);
CREATE TABLE STAGING
(
	TIMEKEY INT,
	PILOTKEY INT ,
	PLANEKEY INT ,
	CHAR_FUEL_GALLONS FLOAT,
	CHAR_DISTANCE REAL,
	REVENUE DECIMAL(10,2),
	EMP_NUM INT,
	AC_NUMBER NVARCHAR(5),
	MOD_CODE NVARCHAR(10),
	CHAR_DATE datetime
	);

ALTER TABLE TIMEDIM
	ADD CONSTRAINT PK_TIMEDIM PRIMARY KEY (TIMEKEY)
ALTER TABLE PILOTDIM
	ADD CONSTRAINT PK_PILOTDIM PRIMARY KEY (PILOTKEY)
ALTER TABLE PLANEDIM
	ADD CONSTRAINT PK_PLANEDIM PRIMARY KEY (PLANEKEY)
ALTER TABLE FACT
ADD
	CONSTRAINT FK_FACT_TIMEDIM FOREIGN KEY (TIMEKEY) REFERENCES TIMEDIM,
	CONSTRAINT FK_FACT_PILOTDIM FOREIGN KEY (PILOTKEY) REFERENCES PILOTDIM,
	CONSTRAINT FK_FACT_PLANEDIM FOREIGN KEY (PLANEKEY) REFERENCES PLANEDIM,
	CONSTRAINT PK_FACT_TIMEDIM PRIMARY KEY (TIMEKEY, PILOTKEY, PLANEKEY)
--PART 3 A10
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- =============================================
-- Author:		Nga Nguyen
-- Create date: 4-6-17, Due 4-12-17
-- Description:	A10P3 STORED PROCEDURE
-- =============================================
CREATE PROCEDURE A10P3
AS
BEGIN
DECLARE @TIMEDIM INT;
DECLARE @PILOTDIM INT;
DECLARE @PLANEDIM INT;
DECLARE @FACT INT;

SET @TIMEDIM = (SELECT COUNT (*) FROM TIMEDIM);
SET @PILOTDIM = (SELECT COUNT (*) FROM PILOTDIM);
SET @PLANEDIM = (SELECT COUNT (*) FROM PLANEDIM);
SET @FACT = (SELECT COUNT (*) FROM FACT);

IF ((@TIMEDIM +@PILOTDIM+@PLANEDIM+@FACT) >0)
BEGIN
```

```
ALTER TABLE FACT
DROP CONSTRAINT FK_FACT_TIMEDIM,
       CONSTRAINT FK_FACT_PILOTDIM ,
       CONSTRAINT FK_FACT_PLANEDIM ,
       CONSTRAINT PK_FACT_TIMEDIM ;
ALTER TABLE TIMEDIM
       DROP CONSTRAINT PK_TIMEDIM;
ALTER TABLE PLANEDIM
       DROP CONSTRAINT PK_PLANEDIM ;
ALTER TABLE PILOTDIM
       DROP CONSTRAINT PK_PILOTDIM;

TRUNCATE TABLE STAGING;
TRUNCATE TABLE FACT;
TRUNCATE TABLE TIMEDIM;
TRUNCATE TABLE PLANEDIM;
TRUNCATE TABLE PILOTDIM;



ALTER TABLE TIMEDIM
       ADD CONSTRAINT PK_TIMEDIM PRIMARY KEY (TIMEKEY)
ALTER TABLE PLANEDIM
       ADD CONSTRAINT PK_PLANEDIM PRIMARY KEY (PLANEKEY)
ALTER TABLE PILOTDIM
       ADD CONSTRAINT PK_PILOTDIM PRIMARY KEY (PILOTKEY)
ALTER TABLE FACT
       ADD
       CONSTRAINT FK_FACT_TIMEDIM FOREIGN KEY (TIMEKEY) REFERENCES TIMEDIM,
       CONSTRAINT FK_FACT_PILOTDIM FOREIGN KEY (PILOTKEY) REFERENCES PILOTDIM,
       CONSTRAINT FK_FACT_PLANEDIM FOREIGN KEY (PLANEKEY) REFERENCES PLANEDIM,
       CONSTRAINT PK_FACT_TIMEDIM PRIMARY KEY (TIMEKEY, PILOTKEY, PLANEKEY)
END
--WILL POPULATE THE TIME DIMENSION
INSERT INTO TIMEDIM(CHAR_DATE, YEAR, MONTH, QUARTER)
SELECT CHAR_DATE, YEAR (CHAR_DATE), MONTH (CHAR_DATE), DATEPART (QUARTER, CHAR_DATE)
FROM CHARTER
GROUP BY CHAR_DATE;
--POPULATE PILOT DIMENSION
INSERT INTO PILOTDIM (EMP_NUM, EMP_FNAME, EMP_LNAME, EMP_HIRE_DATE)
SELECT EMP_NUM, EMP_FNAME, EMP_LNAME, EMP_HIRE_DATE
FROM EMPLOYEE
--POPULATE PLANE DIMENSION
INSERT INTO PLANEDIM (AC_NUMBER, MOD_CODE, MOD_NAME, MOD_CHG_MILE)
SELECT A.AC_NUMBER, M.MOD_CODE, M.MOD_NAME, M.MOD_CHG_MILE
FROM AIRCRAFT A INNER JOIN MODEL M ON A.MOD_CODE=M.MOD_CODE
--THE DIMENSIONS ARE POPULATED NOW.
INSERT INTO STAGING (CHAR_DATE, EMP_NUM, AC_NUMBER, CHAR_DISTANCE,
CHAR_FUEL_GALLONS,REVENUE, PILOTKEY, PLANEKEY,TIMEKEY )
SELECT C.CHAR_DATE, E.EMP_NUM, C.AC_NUMBER, C.CHAR_DISTANCE, C.CHAR_FUEL_GALLONS,
(M.MOD_CHG_MILE*C.CHAR_FUEL_GALLONS) AS [REVENUE], PD.PILOTKEY,PLD.PLANEKEY,TD.TIMEKEY
FROM CHARTER C INNER JOIN CREW CR ON C.CHAR_TRIP=CR.CHAR_TRIP INNER JOIN EMPLOYEE E ON
CR.EMP_NUM=E.EMP_NUM
       INNER JOIN AIRCRAFT A ON C.AC_NUMBER=A.AC_NUMBER INNER JOIN MODEL M ON
A.MOD_CODE=M.MOD_CODE
       INNER JOIN PILOTDIM PD ON E.EMP_NUM=PD.EMP_NUM INNER JOIN PLANEDIM PLD ON
C.AC_NUMBER=PLD.AC_NUMBER
```

```
       INNER JOIN TIMEDIM TD ON C.CHAR_DATE=TD.CHAR_DATE
INSERT INTO FACT(PILOTKEY,PLANEKEY,TIMEKEY,CHAR_DISTANCE,CHAR_FUEL_GALLONS, REVENUE)
SELECT PILOTKEY, PLANEKEY, TIMEKEY, CHAR_DISTANCE, CHAR_FUEL_GALLONS, REVENUE
FROM STAGING
TRUNCATE TABLE STAGING;
END
GO
EXECUTE A10P3;
--A10P4
--QUERY 1
SELECT CONCAT (PD.EMP_LNAME,', ', PD.EMP_FNAME) AS [NAME], F.CHAR_DISTANCE AS [TOTAL
MILES]
FROM FACT F INNER JOIN PILOTDIM PD ON F.PILOTKEY=PD.PILOTKEY
WHERE F.CHAR_DISTANCE = (SELECT MAX (CHAR_DISTANCE)
                                                    FROM FACT);
--QUERY 2
SELECT PD.MOD_NAME, TD.MONTH, SUM (F.REVENUE) AS [TOTAL REVENUE]
FROM FACT F INNER JOIN PLANEDIM PD ON F.PLANEKEY=PD.PLANEKEY INNER JOIN TIMEDIM TD ON
F.TIMEKEY=TD.TIMEKEY
GROUP BY PD.MOD_NAME, TD.MONTH
ORDER BY PD.MOD_NAME ASC, TD.MONTH ASC;
--A11 SOLUTIONS
--70
ALTER TABLE VIDEO
ADD VID_STATUS CHAR(4) NOT NULL DEFAULT 'IN' CHECK (VID_STATUS IN ('IN', 'OUT', 'LOST'))

--71
UPDATE  VIDEO
SET VID_STATUS = 'OUT'
WHERE   VID_NUM IN (SELECT VID_NUM FROM DETAILRENTAL WHERE DETAIL_RETURNDATE IS NULL)
--72
ALTER   TABLE PRICE
ADD PRICE_RENTDAYS INT NOT NULL DEFAULT 3
--73
UPDATE  PRICE
SET     PRICE_RENTDAYS = 5
WHERE   PRICE_CODE = 1 OR PRICE_CODE = 3

UPDATE  PRICE
SET     PRICE_RENTDAYS = 7
WHERE   PRICE_CODE = 4
```

```sql
--78
ALTER PROCEDURE [dbo].[PRC_NEW_DETAIL]
  @VID_NUM_TEMP INT,
  @RENT_NUM INT
AS
BEGIN
  DECLARE @STATUS_TEMP     VARCHAR(4)
  DECLARE @RENT_FEE        MONEY
  DECLARE @LATE_FEE        MONEY
  DECLARE @RENT_DAYS       INT
  DECLARE @DUE_DATE        DATETIME
  DECLARE @VID_NUM_COUNT   INT


  SELECT    @VID_NUM_COUNT =Count(*)
  FROM      VIDEO
  WHERE     VID_NUM =@VID_NUM_TEMP

  IF (@VID_NUM_COUNT = 0)
  BEGIN
        PRINT 'No Video with number ' + @VID_NUM_TEMP + ' was found.'
  END
  IF NOT EXISTS(SELECT * FROM RENTAL WHERE RENT_NUM = @RENT_NUM)
  BEGIN
        PRINT 'Rent number ' + @RENT_NUM + ' was found.'
  END
  ELSE
  BEGIN
    SELECT  @STATUS_TEMP=VID_STATUS
    FROM    VIDEO
    WHERE   VID_NUM = @VID_NUM_TEMP;

    IF @STATUS_TEMP <> 'IN'
        PRINT 'The video is not currently IN.  Video return must be entered before it can be rented again.'
    ELSE
    BEGIN
        SELECT  @RENT_FEE=P.PRICE_RENTFEE, @LATE_FEE=P.PRICE_DAILYLATEFEE
        FROM    VIDEO V INNER JOIN MOVIE M ON M.MOVIE_NUM = V.MOVIE_NUM
                INNER JOIN PRICE P ON P.PRICE_CODE = M.PRICE_CODE
        WHERE   VID_NUM = @VID_NUM_TEMP;

        SET @DUE_DATE = DATEADD(DAY, 3, GETDATE())

        INSERT INTO DETAILRENTAL(RENT_NUM, VID_NUM, DETAIL_FEE, DETAIL_DUEDATE, DETAIL_DAILYLATEFEE)
        VALUES (@RENT_NUM, @VID_NUM_TEMP, @RENT_FEE, @DUE_DATE, @LATE_FEE);

        UPDATE VIDEO
        SET VID_STATUS = 'OUT'
        WHERE VID_NUM = @VID_NUM_TEMP;
    END
  END
END
```

```sql
--79
ALTER PROCEDURE PRC_RETURN_VIDEO
    @VID_NUM_TEMP INT
AS
BEGIN
    DECLARE @VID_NUM_COUNT INT
    DECLARE @RENTALS_COUNT INT
    SELECT  @VID_NUM_COUNT=Count(*)
    FROM    VIDEO
    WHERE   VID_NUM = @VID_NUM_TEMP;

    IF @VID_NUM_COUNT = 0
        PRINT 'Video number ' + CONVERT(VARCHAR(10), @VID_NUM_TEMP) + ' not found.'
    ELSE
    BEGIN
        SELECT  @RENTALS_COUNT=Count(*)
        FROM    DETAILRENTAL
        WHERE   VID_NUM = @VID_NUM_TEMP AND DETAIL_RETURNDATE IS NULL

        IF @RENTALS_COUNT > 1
          PRINT 'ERROR: Video has multiple outstanding Rentals.'
        ELSE
        BEGIN
            UPDATE  VIDEO
            SET VID_STATUS = 'IN'
            WHERE   VID_NUM = @VID_NUM_TEMP;

            IF @RENTALS_COUNT = 0
                PRINT 'No rentals found.  Video is available for rental.'
            ELSE
            BEGIN
                UPDATE DETAILRENTAL
                SET DETAIL_RETURNDATE = CONVERT(DATETIME, CONVERT(CHAR(8), GETDATE()))
                WHERE VID_NUM = @VID_NUM_TEMP AND DETAIL_RETURNDATE IS NULL
                PRINT 'Video successfully returned and available for rental.'
            END
        END
    END
END
```