**1.**

***--Creating function to check if the message contains 'LHL' and takes in the message as a parameter, returns INT as the result is either 0 or 1***

CREATE FUNCTION checkMessage(@MESSAGE varchar(255))

RETURNS INT

AS

BEGIN

***--Declaring a result variable and saving the result of the CHARINDEX select. If 'LHL' is not found it will return 0 else it will return a value larger than 0.***

DECLARE @result INT

SET @result = (SELECT CHARINDEX('LHL', @MESSAGE))

***--Checking if the result is larger than 0. If so then return 1 else return 0 since it does not exist***

IF @result > 0

BEGIN

RETURN 1

END

RETURN 0

END

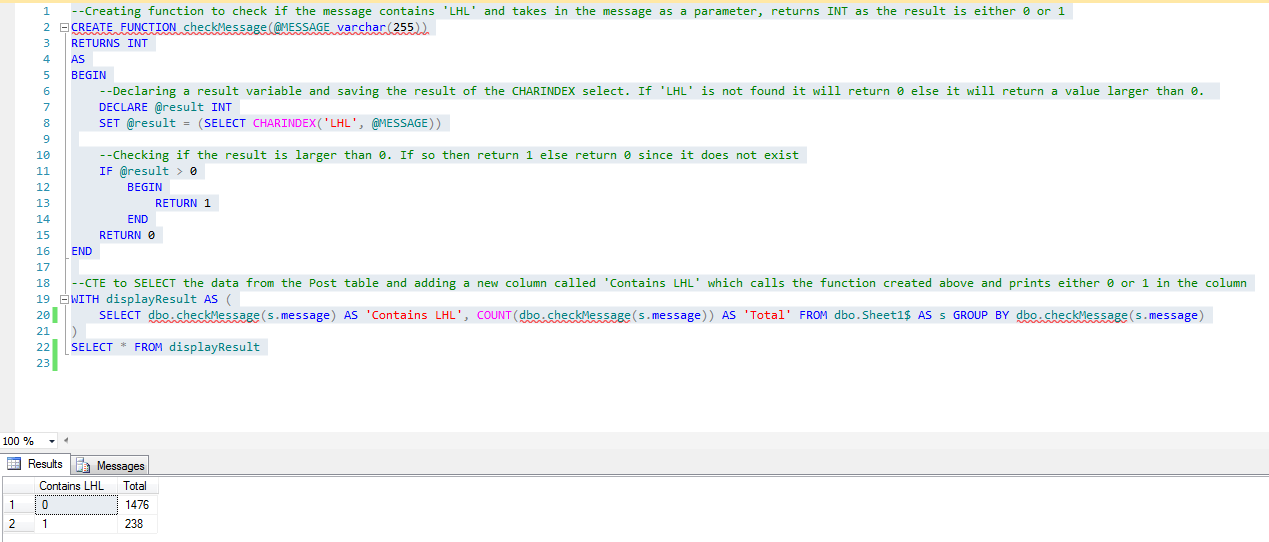
***--CTE to SELECT the data from the Post table and adding a new column called 'Contains LHL' which calls the function created above and prints either 0 or 1 in the column***

WITH displayResult AS (

SELECT dbo.checkMessage(s.message) AS 'Contains LHL', COUNT(dbo.checkMessage(s.message)) AS 'Total' FROM dbo.Sheet1$ AS s GROUP BY dbo.checkMessage(s.message)

)

SELECT \* FROM displayResult



**2.**

**a.**

CREATE FUNCTION Cupid\_Score(@Age INT, @Weight FLOAT, @Height FLOAT, @Smoker VARCHAR(50), @Salary INT)

RETURNS INT

AS

BEGIN

***--Declaring a variable which holds the score that will be set after going through***

***CASE WHEN***

DECLARE @AgeScore INT, @WeightHeightScore INT, @SmokerScore INT, @SalaryScore INT, @totalScore INT, @weightHeightRatio FLOAT

***--Calculate the Weight/Height Ratio and use this value in the CASE WHEN***

SET @WeightHeightRatio = @Weight/@Height

SET @AgeScore =

(

CASE

WHEN @Age BETWEEN 20 AND 30 THEN 4

WHEN @Age BETWEEN 31 AND 40 THEN 3

WHEN @Age BETWEEN 41 AND 50 THEN 2

WHEN @Age >50 THEN 1

END

)

SET @WeightHeightScore =

(

CASE

WHEN @WeightHeightRatio BETWEEN 20 AND 25 THEN 1

WHEN @WeightHeightRatio BETWEEN 25 AND 30 THEN 3

WHEN @WeightHeightRatio BETWEEN 30 AND 35 THEN 4

WHEN @WeightHeightRatio BETWEEN 35 AND 40 THEN 2

END

)

SET @SmokerScore =

(

CASE

WHEN @Smoker = 'Yes' THEN 0

WHEN @Smoker = 'No' THEN 2

END

)

SET @SalaryScore =

(

CASE

WHEN @Salary < 50000 THEN 1

WHEN @Salary BETWEEN 50000 AND 60000 THEN 2

WHEN @Salary BETWEEN 60000 AND 70000 THEN 3

WHEN @Salary > 70000 THEN 4

END

)

*--Now add up the scores and return the total score*

SET @totalScore = @AgeScore+@WeightHeightScore+@SmokerScore+@SalaryScore

RETURN @totalScore

END

**b.**

***--Create Member\_Profile table. use AS for Cupid column and call the Cupid\_Score function to use the other columns as the parameter values and calculate the cupid score***

CREATE TABLE Member\_Profile(

MemberID INT PRIMARY KEY,

Age INT,

Weight FLOAT,

Height FLOAT,

Smoker VARCHAR(50),

Salary INT,

Cupid AS (dbo.Cupid\_Score(Age,Weight,Height,Smoker,Salary))

)

**c.**

***--Insert data***

INSERT INTO Member\_Profile(MemberID,Age,Weight,Height,Smoker,Salary) VALUES(1,26,65,1.80,'Yes',55000)

INSERT INTO Member\_Profile(MemberID,Age,Weight,Height,Smoker,Salary) VALUES(2,50,65,1.88,'No',77000)

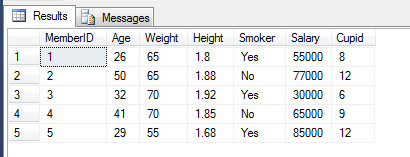
INSERT INTO Member\_Profile(MemberID,Age,Weight,Height,Smoker,Salary) VALUES(3,32,70,1.92,'Yes',30000)

INSERT INTO Member\_Profile(MemberID,Age,Weight,Height,Smoker,Salary) VALUES(4,41,70,1.85,'No',65000)

INSERT INTO Member\_Profile(MemberID,Age,Weight,Height,Smoker,Salary) VALUES(5,29,,1.68,'Yes',85000)

***--Display table***

SELECT \* FROM Member\_Profile



**3.**

*Creating table*

CREATE TABLE QuestionAnswers(

Username VARCHAR(50),

Question\_ID VARCHAR(50),

Answer VARCHAR(5)

)

INSERT INTO QuestionAnswers VALUES('Tom','Q001','D')

INSERT INTO QuestionAnswers VALUES('Wendy','Q009','A')

INSERT INTO QuestionAnswers VALUES('Eddy','Q089','C')

INSERT INTO QuestionAnswers VALUES('David','Q001','C')

INSERT INTO QuestionAnswers VALUES('Eve','Q001','D')

INSERT INTO QuestionAnswers VALUES('Paul','Q001','A')

INSERT INTO QuestionAnswers VALUES('Sam','Q001','B')

***Creating Trigger***

CREATE TRIGGER John\_Instead\_Insert ON QuestionAnswers

INSTEAD OF INSERT

AS

BEGIN

***--Checking if username that is being inserted is John, if so then do not add and instead raiserror***

IF(SELECT Username FROM inserted) = 'John'

BEGIN

RAISERROR ('You have not paid up your fee',10,1)

END

ELSE

***--If it's not John inserting then check if this person has previously inserted an answer for the same question, if so then UPDATE their existing Answer with the new one***

BEGIN

IF(EXISTS(SELECT \* FROM QuestionAnswers JOIN inserted ON QuestionAnswers.Username=inserted.Username AND QuestionAnswers.Question\_ID=inserted.Question\_ID))

UPDATE QuestionAnswers SET QuestionAnswers.Answer = inserted.Answer FROM inserted JOIN QuestionAnswers ON QuestionAnswers.Username=inserted.Username AND QuestionAnswers.Question\_ID=inserted.Question\_ID

ELSE

***--Else if user hasn't inserted an answer for this question before, then create a new row entry***

INSERT INTO QuestionAnswers SELECT Username, Question\_ID, Answer FROM inserted

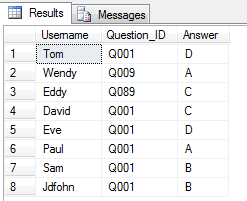
END

END

***Tests***

***--Checks if another user can insert***

INSERT INTO QuestionAnswers VALUES('Jdfohn','Q001','B')



***--Checks if RAISERROR prints when John attempts to insert***

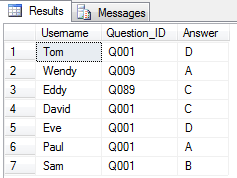
INSERT INTO QuestionAnswers VALUES('John','Q001','B')



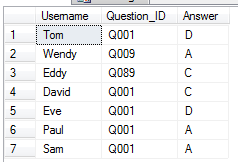
***--Checks if a question that has been answered already updates instead of adding new row***

INSERT INTO QuestionAnswers VALUES('Sam','Q001','A')

***Before***



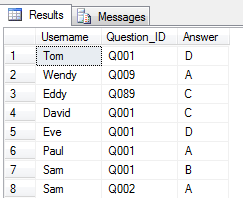
***After***



Q001 for Sam has been updated to A and no new row is created.

***--Checks if inserting a new answer from existing user creates new row***

INSERT INTO QuestionAnswers VALUES('Sam','Q002','A')



**4.**

CREATE TABLE QuestionAnswersQ4(

Username VARCHAR(50),

Question\_ID VARCHAR(50),

Answer VARCHAR(5)

)

INSERT INTO QuestionAnswersQ4 VALUES('Tom','Q001','D')

INSERT INTO QuestionAnswersQ4 VALUES('Wendy','Q009','A')

INSERT INTO QuestionAnswersQ4 VALUES('Eddy','Q089','C')

INSERT INTO QuestionAnswersQ4 VALUES('David','Q001','C')

INSERT INTO QuestionAnswersQ4 VALUES('Eve','Q001','D')

INSERT INTO QuestionAnswersQ4 VALUES('Paul','Q001','A')

INSERT INTO QuestionAnswersQ4 VALUES('Sam','Q001','B')

***--Creating audit table to record insert and deletes that happen to QuestionAnswersQ4 table***

CREATE TABLE audittable(

Username VARCHAR(50),

Question\_ID VARCHAR(50),

Answer VARCHAR(5),

Date\_Time DATETIME,

Type VARCHAR(50)

)

***Trigger***

CREATE TRIGGER TrackInsertDeletes ON QuestionAnswersQ4

AFTER INSERT, DELETE

AS

BEGIN

***--Check if there were any rows inserted, if so then add the insert into the audit table. Else if it was a delete, then add a delete row into the audit table.***

IF EXISTS(SELECT \* FROM inserted)

INSERT INTO audittable SELECT Username, Question\_ID, Answer, getDate(), 'Insert' FROM inserted

ELSE

INSERT INTO audittable SELECT Username, Question\_ID, Answer, getDate(), 'Delete' FROM deleted

END

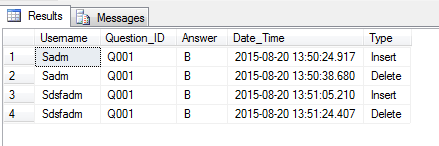
***Tests***

INSERT INTO QuestionAnswersQ4 VALUES('Sadm','Q001','B')

DELETE FROM QuestionAnswersQ4 WHERE Username='Sadm'

INSERT INTO QuestionAnswersQ4 VALUES('Sdsfadm','Q001','B')

DELETE FROM QuestionAnswersQ4 WHERE Username='Sdsfadm'



**5.**

CREATE PROCEDURE ExtractDate

AS

BEGIN

***--Declaring a table to store only the post dates and using substring to remove unwanted information in the created\_time column from the POST table and then inserting it into @post\_dates***

DECLARE @post\_dates TABLE (date DATETIME)

INSERT INTO @post\_dates SELECT SUBSTRING(created\_time,1,19) FROM dbo.Sheet1$

***--Declaring another table to store the times seperated***

DECLARE @post\_createdtimes TABLE (Year VARCHAR(50), Month VARCHAR(50), Day VARCHAR(50), Time VARCHAR(50))

INSERT INTO @post\_createdtimes SELECT DATENAME(year, date) AS 'Year', DATENAME(month, date) AS 'Month', DATENAME(dw, date) AS 'Day', DATEPART(hour, dateadd(hour, datediff(hour, 0, dateadd(mi, 30, date)), 0)) AS Time FROM @post\_dates

***--Total post group by YEAR***

SELECT Year, COUNT(Year) AS 'Total Posts' FROM @post\_createdtimes GROUP BY Year

***--Total post group by MONTH***

SELECT Month, COUNT(Month) AS 'Total Posts' FROM @post\_createdtimes GROUP BY Month

***--Total post group by DAY***

SELECT Day, COUNT(Day) AS 'Total Posts' FROM @post\_createdtimes GROUP BY Day

***--Total post group by TIME***

SELECT Time+':00' AS 'Time', COUNT(Time) AS 'Total Posts' FROM @post\_createdtimes GROUP BY Time

END

GO

EXECUTE ExtractDate

