Frequently Asked Questions:

PLEASE SEE the LAB6 Output Example Posted in the Lab Section on the Class Webpage !!

Note that
1) None of Employee table tuples will be deleted by the event of Delete on Dnumber of Department.
On the event of Delete Dnumber of Department, none of Dno will be deleted. It will be updated to default 1.

Q:
I know that Microsoft SQL server does not have FOR EACH ROW like Oracle, but I was wondering if there was an equivalent to it in SQL Server?

A: I explained it in class, Instead of FOR EACH ROW, which is a standard, MS SQL Server uses DELETED, INSERTED Tables for all the Old tuples and New tuples. See the Example Codes for SQL Server Specific Syntax in the lab section of the class webpage.
The Lecture notes are based on Trigger Standard that is implemented in most of Database Servers: Oracle, DB2, Teradata, Postgres. If you are using MS SQL Server, MS Trigger is out of the Standard trigger, their Trigger implementation is different than the Standard, they don’t have ROW trigger. There is Examples of MS Trigger syntax posted on the Class webpage.
You can find the syntax detail in SQL online documentation as I showed in class. You can start it from Program file - SQL Server 2014 - documentation or google it, it will come out then search with the key word trigger. There are all the detail syntax and examples for your trigger Lab.

Q:
In part 3 in Lab6, UPDATE Department Dnumber 4 to 99 means we have to update Dnumber from 4 to 99 in a loop??

A: NOOOOOOOOOOOO !!!!!
It means Submit update statement to change dnumber 4 to 99 to activate your update trigger as follow:

**UPDATE DEPARTMENT SET DNUMBER = 99 WHERE DNUMBER = 4;**

PLEASE SEE The LAB6 Example Output in the lab section

Q: Do we need to follow the testing procedure specified in part 5 in the Lab for every trigger?
A: NO! The testing part 5 in the lab is just a Suggestion for testing your triggers to avoid any possible transaction related errors between your triggers and any system triggers like table mutating errors or any circular trigger problems. You don't need to follow it unless you need it to avoid errors.

Q: I had deleted all constraints and creating new ones as in the Lab1, but I got an error from SQL server.

ALTER TABLE Employee
ADD CONSTRAINT EmpDeptFk FOREIGN KEY (Dno) REFERENCES Department(Dnumber)
ON Update Cascade
ON Delete SET Default

ALTER TABLE Department
ADD CONSTRAINT DeptMgrFk FOREIGN KEY (MGR_SSN) REFERENCES Employee(SSN)
ON Delete SET Default ON Update Cascade

A: Lab6 does not ask you to add those constraints in DDL. It asks you to implement the meaning of those Referential Integrity Constraints through triggers. You have to write triggers to do those actions. So, not to conflict with any system triggers that are doing the same action behind in case when you create those Constraints in your DDL, I asked to remove all those constraints from your tables, do not add those constraints. Before creating any trigger for that, I recommended that you drop all those constraints from the tables if those were already added before.

Q: I just wanted to ask you if this behavior was correct. I was looking over my lab again today, and decided to run some test cases. The test case I ran was:

**Description:** Update a Department with a DNUMBER that is not in the Department table.

Expected Result: No record should be inserted into the audit table, since no record in the Department table was updated.

Result: Record was inserted into Audit table.

FAIL

This was my trigger before modification:

It is very messy, but got the job done. I noticed that this was being called when I executed the statement:

```
UPDATE DEPARTMENT SET DNUMBER = 120 WHERE DNUMBER = 100
```

Even when there were no tuples in the Department table with a DNUMBER of 100.
I added the following logic around the 'UPDATE' clause in the Trigger to check for a corresponding value in the DELETED, and INSERTED tables (as this would ensure that something was actually being updated).

The trigger now looks like this:

```sql
IF @OLDDNUMBER IS NOT NULL AND @NEWDNUMBER IS NOT NULL
BEGIN
    UPDATE [dbo].[EMPLOYEE]
    SET [DNO] = @NEWDNUMBER
    WHERE DNO = @OLDDNUMBER
EXEC SP_Audit_Dept @date_of_change, @old_Dname, @new_Dname, @OLDDNUMBER, @NEWDNUMBER, @old_Mgrssn, @new_Mgrssn
END
```

Is this the best way to ensure the trigger only calls the SP when it is necessary?

A: Yes, you can add condition checks to make the trigger work only for the precise situation. More condition checks, the better to prevent unexpected side effects! You can add Where clause conditions in each Select from Deleted/Inserted as well if needed.