**Points**: 10

**Reference: Chapter 15, Murach**

**Goal**: Write a script that will do the following:

1. Write a trigger that will record additions, changes and deletions to the table Person.StateProvince in AdventureWorks2017
2. Write a stored procedure that will determine how active the use of the trigger has been.

**Requirements**: A single script must do the following:

1. Create a new database called AvLogs
2. Switch context to AvLogs
3. In the AvLogs database, create a table to record the history of actions on the table AdventureWorks2017.Person.StateProvince, and its name will be dbo.StateProvinceLog. The structure of the table duplicates AdventureWorks2017.Person.StateProvince with the following exceptions:
   1. There must also be three additional columns as follows:

ChangeType nchar(1) - - single letter to indicate type of change

ChangeDate datetime - - date and time change was made

UserName nvarchar(30) - - User name that made the change

* 1. Do not carry forward the rowguid column
  2. You do not need to add a primary key, nor be concerned with defining foreign keys.
  3. You can save yourself some keyboarding by right clicking on the AdventureWorks2017.Person.StateProvince in the object Object Explorer panel and selecting the menu choice **Script as->Create To->New Query Editor Window.** You can use that to edit/copy/paste as appropriate to capture the original column definitions that would need to be duplicated. Note, when recreating a table definition this way, you will notice that the columns Name and IsOnlyStateProvinceFlag don’t use the usual data types. Instead, they use user defined types. By looking at the columns in object explorer you will see the actual underlying data types. If you look in the “Types” folder for AdventureWorks2017, you see the customer type definition. More information about user defined types is at <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-type-transact-sql?view=sql-server-ver15> Murach gives some mention using a table type on pages 478-479

1. Switch context to AdventureWorks2017
2. Create a trigger named stPersonLog that will be “fired” anytime there is an INSERT, UPDATE or DELETE on AdventureWorks2017.Person.StateProvince that will do the following:
   1. It will copy all the information from either the inserted or deleted table as appropriate to the AvLogs.dbo.StateProvinceLog table.
   2. The trigger must note the type of change with a single letter of either I, D, or U (insert, delete, update), the time of the change and the user name associated with the change (note additional columns above)
   3. This should be accomplished with only a single trigger (i.e. not one separate trigger each for INSERT, UPDATE, DELETE).
3. Write three DML statements as follows:
   1. One that adds the province code “BA” with the country region code of “UM” and the Territory ID for the Northwest part of the United States. The full name of the province is “Bailey” Fill in the remainder of the columns based on values at your discretion. However, for rowguid column, you will need the value generated from the function NEWID()
   2. One that changes the full name of the BA province to “Buckey”
   3. One that deletes the BA province.
   4. Write a SELECT statement that shows the full contents of the table AvLogs.dbo.StateProvinceLog.
4. Switch context to AvLogs
5. Create a stored procedure name procLogActivity that will take a parameter named @checkDate that will be of type date. It will also have an OUTPUT parameter named @activeCount
6. The stored procedure will determine how active the trigger has been by counting the number of log entries since the date entered as the parameter. @activeCount will be assigned that number. The return value for the stored procedure should be zero if there were no errors, one otherwise.
7. Write a short snippet of code that will exercise the procLogActivity that should look as follows:

DECLARE @ReturnCount int

DECLARE @ReturnValue int

DECLARE @curDate date

SELECT @curDate = GETDATE()

EXEC @ReturnValue = procLogActivity

@checkDate = @curDate

,@activeCount = @ReturnCount OUTPUT

SELECT @ReturnValue, @ReturnCount

1. Enter the commands to remove the trigger (See Murach and <https://docs.microsoft.com/en-us/sql/t-sql/statements/drop-trigger-transact-sql?view=sql-server-ver15> )
2. Enter the commands to drop the database AvLogs
3. Run your script at least two times to make sure it runs clean.