Assignment G

- Submit only one sql file for this assignment. Name the Script **MIS-421-AssignmentG-LastName-FirstName.sql**.
- All scripts need to be included in this file.
- You must also include all setup and test scripts in the file.
- All triggers and stored procedures need to be created in your individual database to get credits.
- Make sure your tblSaleItem table has a composite primary key comprising SaleID and SaleItemID
- Your tblSale table's primary key is SaleID and it is an identity field.

Triggers:

Download from Canvas and run "Assignment G Preparationg.sql" in your individual database to prepare for the exercise. On the tblMembership table, create a user-defined INSTEAD OF UPDATE trigger. Name the trigger utrInsteadOfUpdatePayment. The following is the logic flow for the trigger:

When a person or company renews a membership by paying the annual membership fee, the organization's treasurer updates the MembershipFeePaidDate field for that membership with the new payment date. This update causes the `utrInsteadOfUpdatePayment` trigger to fire.

- The trigger retrieves the new MembershipFeePaidDate from the dbms's `inserted` table.
- The trigger retrieves the prior MembershipFeePaidDate and MembershipNumber from the dbms's `deleted` table.
- **Business Rules** (the curly brackets in the following message are placeholder, which indicates that you should get the values in your trigger and place the actual values there):
  - If prior payment date is null, the trigger updates the MembershipFeePaidDate with the new payment date and set isCurrentMember field to be true. Print a message in the following format:

    Membership Number {actual number}: payment date is {actual new payment date}; no prior payment date

  - If prior payment date is not null and prior payment date is earlier than the new payment date (@priorDate < @newDate), then the trigger updates the MembershipFeePaidDate with the new payment date and set isCurrentMember field to be true. Print a message in the following format:

    Membership Number {actual number}: payment date is {actual new payment date}; prior payment date is {actual prior payment date}

  - If prior payment date is not null and prior payment date is later than the new payment date (@priorDate > @newDate). No update, just print the following message.

    Membership Number {actual number}: new payment date {actual new payment date} is earlier than prior payment date {actual prior payment date}; no change made

- Test the trigger with each case mentioned above:
  Include the test scripts in MIS-421-AssignmentG-LastName-FirstName.sql.

Stored Procedures:

This stored procedure depends on the tables you created for your Assignment D. Make sure you correct the tables based on the feedback, if any, I provided for your Assignment D.
In MIS-421-AssignmentG-LastName-FirstName.sql, for your individual database write a stored procedure named uspCustomerPurchaseTransaction that includes input parameters of SaleID, CustomerID, EmployeeID, ItemID, and ItemPrice. The stored procedure also needs to include an OUTPUT parameter called status and another OUTPUT parameter called message in addition to the input parameters.

- At the beginning of the stored procedure, print out the values of the parameters. You can use the following code:

```
PRINT 'INPUT VALUES'
PRINT '------------------------------------'
PRINT 'SALE ID: ' + CAST(@saleID as varchar(10));
PRINT 'Employee ID: ' + CAST(@employeeID as varchar(10));
PRINT 'Customer ID: '  + CAST(@customerID as varchar(10));
PRINT 'Item ID: ' + CAST(@itemID as varchar(10));
PRINT 'Item Price: ' + CAst(@itemPrice as varchar(50));
PRINT '------------------------------------'
```

- Your stored procedure needs to perform the following tasks:
  1. Check if the ItemID and ItemPrice match the values in tblItem table.
  2. If they don't match, set the status output variable value to -1, and an appropriate message to the message output variable, rollback transaction, and return (do not proceed).
  3. Check if the sale already exists in the tblSale table:
     - If not, record the sale in the table (you only need to provide values for SaleID, CustomerID, and EmployeeID). If the insertion to the tblSale table fails (use a try…catch block), in the catch block set the status output variable value to -2 and an appropriate message to the message output variable, rollback transaction, print a message to the Message window, and return (do not proceed).
       NOTE: If your SaleID is an identity field as it should be in your database, you need to set the identity insert on/off to complete the insertion.
  4. Insert the sale item to the tblSaleItem table:
     - Declare a local variable @saleItemID. Set the value of @saleItemID to be 1+'the maximum SaltelteмID for the sale':

```
Select @saleItemID=max(SaleItemID) from tblSaleItem where SaleID=@saleID
IF @saleItemID is null
      SET @saleItemID = 1;
ELSE
      SET @saleItemID = @saleItemID + 1;
```

     - If the insertion to the tblSaleItem table fails (use a try…catch block), in the catch block rollback all transactions, set the status output variable value to -3 and an appropriate

message to the message output variable, rollback transaction, print a message to the Message window, and return.
    - If the insertion to the tblSaleItem table succeeds, commit the transactions and set the status output variable value to 1, indicating all transactions are successful and print a message to the Message window.
- In MIS-421-AssignmentG-LastName-FirstName.sql create three testing scripts to test that your stored procedure works:
    - One script testing the input values for ItemID and ItemPrice do not match the values in the tblItem table
    - One script testing a SaleID exists in the tblSale table and all statements are successfully executed
    - One script testing a SaleID does not exist in the tblSale table and all statements are successfully executed
- Submit MIS-421-AssignmentG-LastName-FirstName.sql to Canvas