Lab 5

Connection values:

Server Type = Database Engine
Server Name = boyce.coe.neu.edu
Authentication = SQL Server Authentication
Login = INFO6210
Password = NEUHusky!

```
/*
   SQL variables start with either @ or @@.
   @ indicates a local variable, which is in effect in the current
   scope.
   @@ indicates a global variable, which is in effect for all
   scopes of the current connection.
*/
```

-- A simple example of Stored Procedure

```
-- Set the database context
USE “The name of a database you have created.” ;
```

--Create a stored procedure with INPUT and OUTPUT parameters

```
/* A parameter has a data type, such as INT (integer).
   If a parameter will return a value, we specify the OUTPUT keyword.
   If we have only a single SQL statement after IF and/or ELSE,
   we don’t have to use BEGIN .... END, but if we have multiple
   statements, we have to put them in the BEGIN .... END block. */

CREATE PROCEDURE MyFirstProcedure
   @InNumber INT,
   @OutNumber INT OUTPUT
AS
BEGIN
   IF @InNumber < 0
         SET @OutNumber = 0;
   ELSE
      BEGIN
         SET @OutNumber=@InNumber + 1;
      END
   PRINT @OutNumber;
END

-- The statements highlighted in yellow must be executed together

-- Declare variables
DECLARE @MyInput INT;
DECLARE @MyOutput INT;

-- Initilize variable
SET @MyInput = 3;

-- Execute the procedure
EXEC MyFirstProcedure @MyInput, @MyOutput OUTPUT;

-- See result
SELECT @MyOutput;

-- Drop the procedure so that you can recreate it
DROP PROC MyFirstProcedure;
```

-- Use TRY and CATCH for error handling in a Stored Procedure

**/*
TRY contains regular SQL statements we execute to accomplish a task.
CATCH contains SQL statements used to handle the error if an error has Occurred.
*/**

```
USE “The name of a database you have created.” ;

GO


-- The statements highlighted in yellow must be executed together

BEGIN TRY
    BEGIN TRANSACTION;

    DELETE FROM AdventureWorks2008R2.Production.Product
        WHERE ProductID = 980;

    -- If the delete operation succeeds, commit the transaction.
    COMMIT TRANSACTION;
END TRY

BEGIN CATCH

    PRINT 'UNABLE TO DELETE PRODUCT!';

    -- Roll back any active or uncommittable transactions
    IF XACT_STATE() <> 0
    BEGIN
        ROLLBACK TRANSACTION;
    END;

END CATCH;
```

-- Simple examples of Functions

```
USE “The name of a database you have created.” ;

-- Create a scalar function
-- FUNCTION accepts Argument(s)
-- In this example, @Country is the argument.
-- FUNCTION uses the RETURN statement to return the value

CREATE FUNCTION whichContinent
(@Country nvarchar(15))
RETURNS varchar(30)
AS
BEGIN
      DECLARE @ReturnC varchar(30);

      SELECT @ReturnC = CASE @Country
                  when 'Argentina' then 'South America'
                  when 'Belgium' then 'Europe'
                  when 'Brazil' then 'South America'
                  when 'Canada' then 'North America'
                  when 'Denmark' then 'Europe'
                  when 'Finland' then 'Europe'
                  when 'France' then 'Europe'
                ELSE 'Unknown'
                END;

      RETURN @returnC;
END

-- Execute the new function

SELECT dbo.whichContinent('Canada');
```

```
USE “The name of a database you have created.” ;


-- Create a table-valued function

CREATE FUNCTION dbo.GetDateRange
(@StartDate date, @NumberOfDays int)
RETURNS @DateList TABLE (Position int, DateValue date)
AS BEGIN
    DECLARE @Counter int = 0;
    WHILE (@Counter < @NumberOfDays)
    BEGIN
        INSERT INTO @DateList
            VALUES(@Counter + 1,
                   DATEADD(day,@Counter,@StartDate));
        SET @Counter += 1;
    END
    RETURN;
END
GO

-- Execute the new function

SELECT * FROM dbo.GetDateRange('2009-12-31',14);
```

```
USE “The name of a database you have created.” ;

-- Create a table-valued function

CREATE FUNCTION GetLastOrdersForCustomer
(@CustomerID int, @NumberOfOrders int)
RETURNS TABLE
AS
RETURN (SELECT TOP(@NumberOfOrders)
               SalesOrderID,
               OrderDate,
               PurchaseOrderNumber
        FROM AdventureWorks2008R2.Sales.SalesOrderHeader
        WHERE CustomerID = @CustomerID
        ORDER BY OrderDate DESC, SalesOrderID DESC
        );
GO

-- Execute the new function

SELECT * FROM GetLastOrdersForCustomer(17288,2);
```

-- A simple example of WHILE Statement

```
/*
   We need to make sure that we have a way to stop the WHILE loop.
   Otherwise, we'll have an endless WHILE loop which may run forever.
   We use the variable @counter to determine when to terminate
   the WHILE loop in this example.
   We use CAST to convert an integer to character(s) so that we
   can concatenate the integer with other characters.
*/

DECLARE @counter INT;
SET @counter = 0;
WHILE @counter <> 5
   BEGIN
      SET @counter = @counter + 1;
      PRINT 'The counter : ' + CAST(@counter AS CHAR);
   END;
```

Lab 5 Questions

Note: 1.5 points for each question.

```
-- Lab 5-1

/* Using the SQL PIVOT command, rewrite the following query to present the same data
   in a horizontal format, as listed below. Please use AdventureWorks2008R2 for this
   question. */

select datepart(yy, OrderDate) Year,
       SalesPersonID,
          FirstName,
          LastName,
       cast(sum(TotalDue) as int) as TotalSales
from Sales.SalesOrderHeader sh
join Person.Person p
on sh.SalesPersonID = p.BusinessEntityID
where year(OrderDate) in (2006, 2007) and SalesPersonID between 275 and 278
group by SalesPersonID,  datepart(yy, OrderDate), FirstName, LastName
having sum(TotalDue) > 1500000;
```

Year	275 Michael Blythe	276 Linda Mitchell	277 Jillian Carson	278 Garrett Vargas
2006	3470329	3653237	4286905	
2007	4425590	4626812	4104090	1554236

```
-- Lab 5-2

/* Using data from AdventureWorks2008R2, create a function that accepts
   a customer id and returns the full name (last name + first name)
   of the customer. */


-- Lab 5-3

 /* Given the following tables, there is a university rule
    preventing a student from enrolling in a new class if there is
    an unpaid fine. Please write a table-level CHECK constraint
    to implement the rule. */

create table Course
(CourseID int primary key,
 CourseName varchar(50),
 InstructorID int,
 AcademicYear int,
 Semester smallint);

create table Student
(StudentID int primary key,
 LastName varchar (50),
 FirstName varchar (50),
 Email varchar(30),
 PhoneNumber varchar (20));

create table Enrollment
(CourseID int references Course(CourseID),
 StudentID int references Student(StudentID),
```

```
 RegisterDate date,
 primary key (CourseID, StudentID));

create table Fine
(StudentID int references Student(StudentID),
 IssueDate date,
 Amount money,
 PaidDate date
 primary key (StudentID, IssueDate));


-- Lab 5-4

/* Given the following tables, there is a $4 shipping fee
   for each ordered product.

   For example, if an order contains:
   Product 1, quantity 2
   Product 2, quantity 2

   then the order quantity is 4 and the shipping fee is $16.
   If the order value is greater than 600, then the shipping fee is $2
   for each ordered product. The discounted shipping fee
   will be $8,

   Use UnitPrice * Quantity of all products contained in an order to calculate
   the order value. ShippingFee and Tax are not included in the order value.

   Write a trigger to calculate the shipping fee for an order. Save
   the shipping fee in the ShippingFee column of the SalesOrder table. */

create table Customer
(CustomerID int primary key,
 LastName varchar(50),
 FirstName varchar(50),
 Membership varchar(10));

create table SalesOrder
(OrderID int primary key,
 CustomerID int references Customer(CustomerID),
 OrderDate date,
 ShippingFee money,
 Tax as OrderValue * 0.08,
 OrderValue money);

create table OrderDetail
(OrderID int references SalesOrder(OrderID),
 ProductID int,
 Quantity int,
 UnitPrice money
 primary key(OrderID, ProductID));
```

Useful Links

Create a Stored Procedure

http://msdn.microsoft.com/en-us/library/ms345415.aspx

Create a Function

http://msdn.microsoft.com/en-us/library/ms186755.aspx

Use TRY and CATCH for Error Handling

http://msdn.microsoft.com/en-us/library/ms175976.aspx

XACT_STATE

http://msdn.microsoft.com/en-us/library/ms189797.aspx

DATEADD

http://msdn.microsoft.com/en-us/library/ms186819.aspx

DATEPART

https://docs.microsoft.com/en-us/sql/t-sql/functions/datepart-transact-sql

CROSS APPLY vs INNER JOIN

https://stackoverflow.com/questions/1139160/when-should-i-use-cross-apply-over-inner-join