

Lab 4

Connection values:

Server Type = Database Engine

Server Name = boyce.coe.neu.edu

Authentication = SQL Server Authentication

Login = INF06210

Password = NEUHusky!

-- Create a database and some tables in the new database.

```
CREATE DATABASE "Use your name for the database name";  
GO
```

```
USE "Use your name for the database name";
```

```
CREATE TABLE dbo.Customers  
(  
    CustomerID varchar(5) NOT NULL PRIMARY KEY ,  
    Name varchar(40) NOT NULL  
);
```

```
CREATE TABLE dbo.Orders  
(  
    OrderID int IDENTITY NOT NULL PRIMARY KEY,  
    CustomerID varchar(5) NOT NULL  
        REFERENCES Customers(CustomerID),  
    OrderDate datetime DEFAULT Current_Timestamp  
);
```

```
CREATE TABLE dbo.Products  
(  
    ProductID int IDENTITY NOT NULL PRIMARY KEY,  
    Name varchar(40) NOT NULL,  
    UnitPrice money NOT NULL  
);
```

```
CREATE TABLE dbo.OrderItems  
(  
    OrderID int NOT NULL  
        REFERENCES dbo.Orders(OrderID),  
    ProductID int NOT NULL  
        REFERENCES dbo.Products(ProductID),  
    UnitPrice money NOT NULL,  
    Quantity int NOT NULL  
        CONSTRAINT PKOrderItem PRIMARY KEY CLUSTERED  
            (OrderID, ProductID)  
);
```

-- Put some data in the database

-- INSERT sample records

```
INSERT dbo.Customers  
VALUES ('ABC', 'Bob''s Pretty Good Garage');
```

```
INSERT dbo.Orders (CustomerID)  
VALUES ('ABC');
```

```
INSERT dbo.Products  
VALUES ('Widget', 5.55),  
       ('Thingamajig', 8.88)
```

```
INSERT dbo.OrderItems  
VALUES (1, 1, 5.55, 3);
```

/*

**If you create a table without specifying constraints,
You can use ALTER TABLE to add a constraint**

*/

-- Create a table without specifying constraints.

```
CREATE TABLE TBL3 (pk3 int);
```

-- Add the NOT NULL constraint

```
ALTER TABLE tbl3 ALTER COLUMN pk3 int not null;
```

-- Add the Primary Key constraint.

```
ALTER TABLE tbl3 ADD CONSTRAINT key3 PRIMARY KEY (pk3);
```

-- Add the Foreign Key constraint.

-- Create the parent table first.

```
CREATE TABLE TBL1 (pk1 int PRIMARY KEY);
```

```
ALTER TABLE tbl3 ADD CONSTRAINT R3 FOREIGN KEY (pk3)  
REFERENCES tbl1(pk1)
```

-- Must DROP the child table before dropping the parent table.

```
DROP TABLE TBL3;
```

```
DROP TABLE TBL1;
```

-- A simple example of WHILE Statement

```
/*
    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.
*/

/*
    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.
    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
END
```

-- Use a Nested Loop to populate your table.

-- Create a test table.

```
CREATE TABLE PART (Part_Id int, Category_Id int,  
    Description varchar(50));
```

-- The statements highlighted in yellow must be executed together

-- Declare SQL variables.

```
DECLARE @Part_Id int;  
DECLARE @Category_Id int;  
DECLARE @Desc varchar(50);
```

-- Initilize SQL variables.

```
SET @Part_Id = 0;  
SET @Category_Id = 0;
```

-- Populate the test table.

```
WHILE @Part_Id < 10  
BEGIN  
    SET @Part_Id = @Part_Id + 1;  
    WHILE @Category_Id < 3  
    BEGIN  
        SET @Category_Id = @Category_Id + 1;  
        SET @Desc = 'Part_Id is ' + cast(@Part_Id as char(1)) +  
            ' Category_Id ' + cast(@Category_Id as char(1));  
        INSERT INTO PART VALUES (@Part_Id,  
                                @Category_Id,  
                                @Desc );  
    END;  
    SET @Category_Id = 0;  
END;
```

-- Retrieve the test data.

```
SELECT * FROM PART;
```

-- Drop the test table.

```
DROP TABLE PART;
```

-- SQL View

```
USE AdventureWorks2008R2;
```

```
-- CREATE VIEW Command
```

```
-- You need to execute these statements on your own computer
```

```
CREATE VIEW vwEmployeeContactInfo
AS
SELECT e.[BusinessEntityID] as [ContactID], FirstName,
       MiddleName, LastName, JobTitle
FROM Person.Person c
INNER JOIN HumanResources.Employee e
       ON c.BusinessEntityID = e.BusinessEntityID;
```

```
-- Select from the view
```

```
SELECT *
FROM vwEmployeeContactInfo;
```

```
-- See the script that generated the view
```

```
EXEC sp_helptext vwEmployeeContactInfo;
```

```
-- Delete the view from the database
```

```
DROP VIEW vwEmployeeContactInfo;
```

```
/*  
    Create a view to include the encryption and  
    schemabinding options. Encryption protects the  
    view query definition. Schemabinding means the  
    definition of the database object(s) on which  
    the view is defined can not be changed without  
    first dropping the view.  
*/
```

```
CREATE VIEW vwEmployeeContactInfo  
WITH ENCRYPTION, SCHEMABINDING  
AS  
SELECT e.[BusinessEntityID] as [ContactID], FirstName,  
        MiddleName, LastName, JobTitle  
FROM Person.Person c  
INNER JOIN HumanResources.Employee e  
        ON c.BusinessEntityID = e.BusinessEntityID;
```

```
/*  
    Alter the view to remove schemabinding - must  
    restate everything, including changes.  
*/
```

```
ALTER VIEW vwEmployeeContactInfo  
WITH ENCRYPTION  
AS  
SELECT e.[BusinessEntityID] as [ContactID], FirstName,  
        MiddleName, LastName, JobTitle  
FROM Person.Person c  
INNER JOIN HumanResources.Employee e  
        ON c.BusinessEntityID = e.BusinessEntityID;
```


Lab 4 Questions

Part A (2 points)

Create 4 tables and the corresponding relationships to implement the ERD below in your own database.



Part B – 1 (2 points)

```
/* Write a query to retrieve the top 3 customers, based on the total purchase,
   for each year. The top 3 customers have the 3 highest total purchase amounts.
   Use TotalDue of SalesOrderHeader to calculate the total purchase.
   Also calculate the top 3 customers' total purchase amount for the year.
   Return the data in the following format.
```

```
Year    Total Sale    Top3Customers
2005    748178          29624, 29861, 29562
2006    1112218         29614, 29716, 29722
2007    1230198         29913, 29818, 29701
2008    697280          29923, 29641, 29617
*/
```

Part B – 2 (2 points)

```
/*
Using AdventureWorks2008R2, write a query to retrieve
the salespersons and their order info.
```

```
Return a salesperson's id, a salesperson's total order count,
```

the lowest total product quantity contained in an order for all orders of a salesperson, and the order values of a salesperson's bottom 3 orders. The returned data should have the format displayed below. Include only the orders which have a salesperson specified for this question.

For the lowest total product quantity contained in an order for all orders of a salesperson, an example is:

John has 3 orders.

Order #1 has a total sold quantity of 5
Order #2 has a total sold quantity of 25
Order #3 has a total sold quantity of 21

Then the lowest total product quantity contained in an order for all orders of John is 5.

The bottom 3 orders have the 3 lowest order values.
Use TotalDue in SalesOrderHeader as the order value.
If there is a tie, the tie must be retrieved.

Include only the salespersons who owned the top 3 orders for all orders which have a salesperson specified. The top 3 orders have the 3 highest numbers of total sold quantity contained in an order.

Sort the returned data by SalespersonID.

```
*/  
  
/*  
SalesPersonID  TotalOrderCount  LowestQuantity  Lowest3Values  
      XXX           XXX           XXX           XX.XX, XX.XX, XX.XX  
*/
```

Part C (2 points)

```
/* Bill of Materials - Recursive */  
/* The following code retrieves the components required for manufacturing "Mountain-500  
Black, 48" (Product 992). Modify the code to retrieve the most expensive component(s) at  
each component level. Use the list price of a component to determine the most expensive  
component for each level. Exclude the components which have a list price of 0. Sort the  
returned data by the component level. */
```

```
WITH Parts(AssemblyID, ComponentID, PerAssemblyQty, EndDate, ComponentLevel) AS  
(  
  SELECT b.ProductAssemblyID, b.ComponentID, b.PerAssemblyQty,  
         b.EndDate, 0 AS ComponentLevel  
  FROM Production.BillofMaterials AS b  
 WHERE b.ProductAssemblyID = 992 AND b.EndDate IS NULL  
 UNION ALL  
  SELECT bom.ProductAssemblyID, bom.ComponentID, bom.PerAssemblyQty,  
         bom.EndDate, ComponentLevel + 1  
  FROM Production.BillofMaterials AS bom  
 INNER JOIN Parts AS p  
  ON bom.ProductAssemblyID = p.ComponentID AND bom.EndDate IS NULL  
)  
SELECT AssemblyID, ComponentID, Name, PerAssemblyQty, ComponentLevel  
FROM Parts AS p
```

```
INNER JOIN Production.Product AS pr
ON p.ComponentID = pr.ProductID
ORDER BY ComponentLevel, AssemblyID, ComponentID;
```

Useful Links

Some great discussions about naming conventions

<http://social.msdn.microsoft.com/Forums/sqlserver/en-US/fc76df37-f0ba-4cae-81eb-d73639254821/sql-server-naming-convention?forum=databasedesign>

Create Database Using SQL Server Management Studio

http://www.youtube.com/watch?v=J59MGbQ_Shc

Create Tables Using SQL Server Management Studio

<http://technet.microsoft.com/en-us/library/ms188264.aspx>

Create Tables Using SQL Server Management Studio

<http://www.youtube.com/watch?v=8l5Hw4kQE8o>

Data Types

<http://msdn.microsoft.com/en-us/library/ms187752.aspx>

Create View

<http://technet.microsoft.com/en-us/library/ms187956.aspx>

How to Create a View

http://www.youtube.com/watch?v=MK_dWEcltWY