Lab 4

Connection values:

Server Type = Database Engine
Server Name = is-swang01.ischool.uw.edu
Authentication = SQL Server Authentication
Login = INFO6210
Password = GoHusky!

-- 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
```

-- 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

Step 1) Create a new database of the format: LASTNAME_FIRSTNAME_TEST

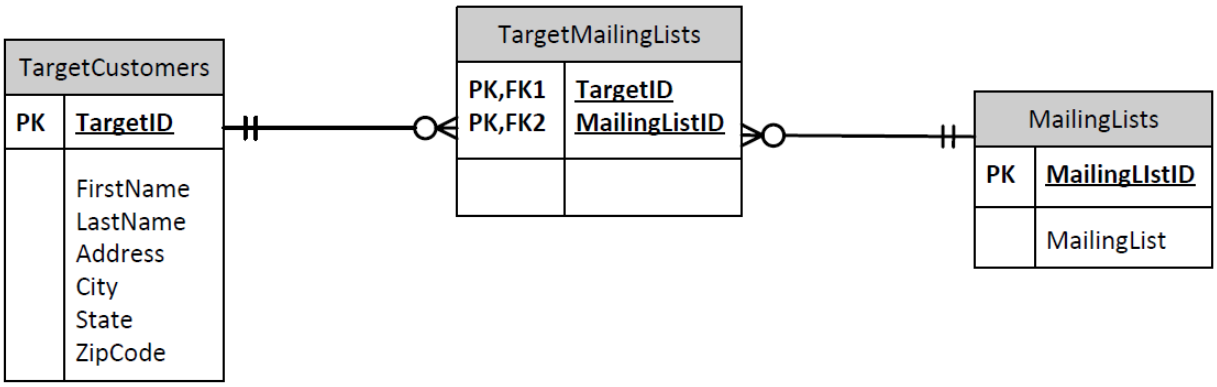
Step 2) Within this database, experiment creating, altering, and dropping tables and columns. Put in sample data and play around with other kinds of queries. Write at least 20 SQL statements for this step.

Note: 1 point for Step 2

Step 3) Eventually, create 3 tables and the corresponding relationships to implement the ERD below in the database.

Note: 2 points for Step 3

Step 4) Keep track of all the code you write and submit it to Blackboard.



Part B (1 point)

```
/* Using the content of AdventureWorks, write a query to retrieve
   all unique customers with all salespersons each customer has dealt with.
   If a customer has never worked with a salesperson, make the
   'SalesPersonID' column blank instead of displaying NULL.
   Sort the returned data by CustomerID in the descending order.
   The result should have the following format.
```

Hint: Use the SalesOrderHeadrer table.

CustomerID	SalesPersonID
30118	275, 277
30117	275, 277
30116	276
30115	289
30114	290
30113	282
30112	280, 284

*/

Part C (2 points)

```
/* Bill of Materials - Recursive */

/* The code below retrieves the components required for manufacturing
   "Mountain-500 Black, 48" (Product 992). Modify the code to calculate
   the cost if we decide to build Product 992 using components at the
   component levels 0 and 1. Use the list price of a component for
   calculating the cost. */
```

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=8I5Hw4kQE8o

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