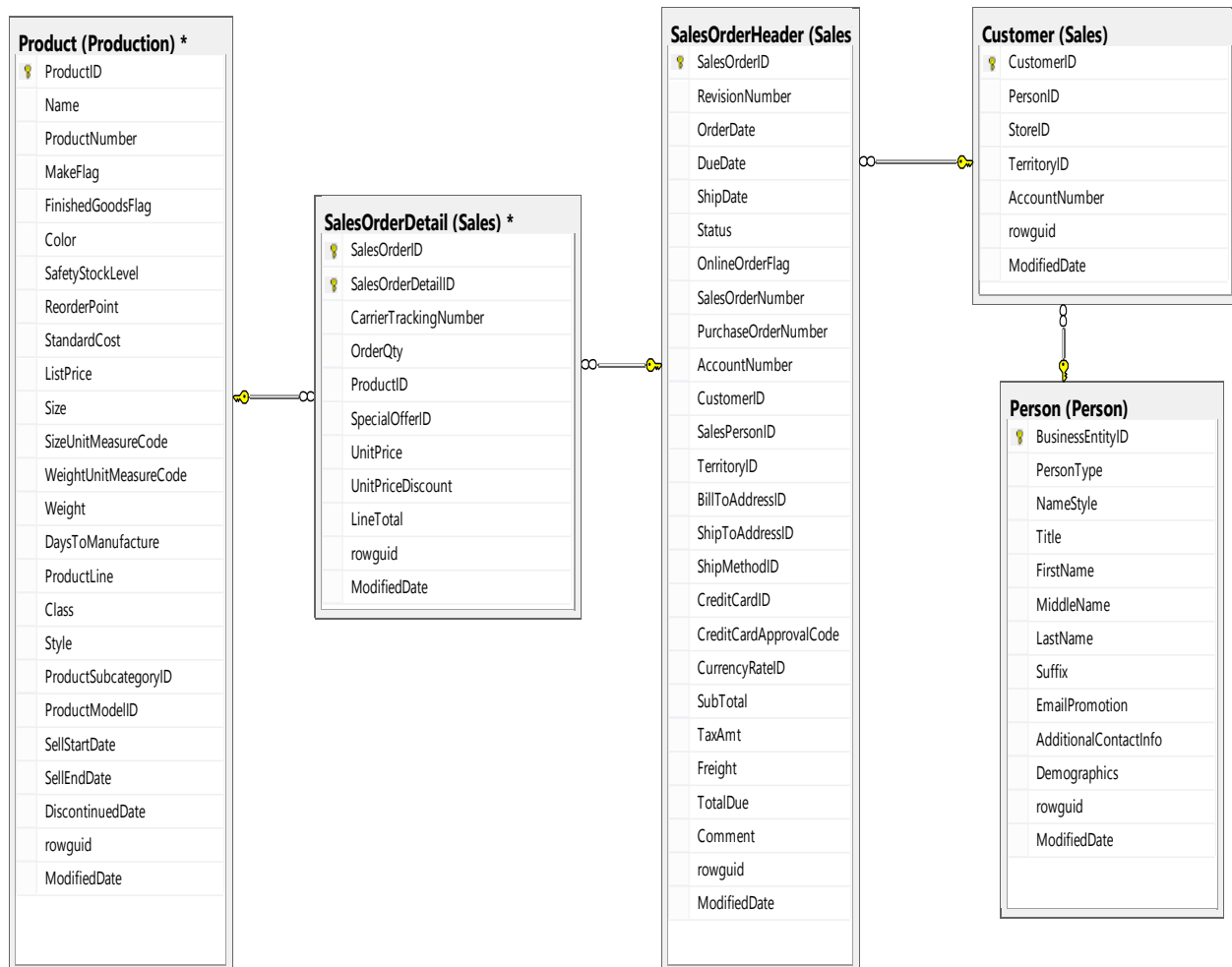


## Lab 5 Exercises

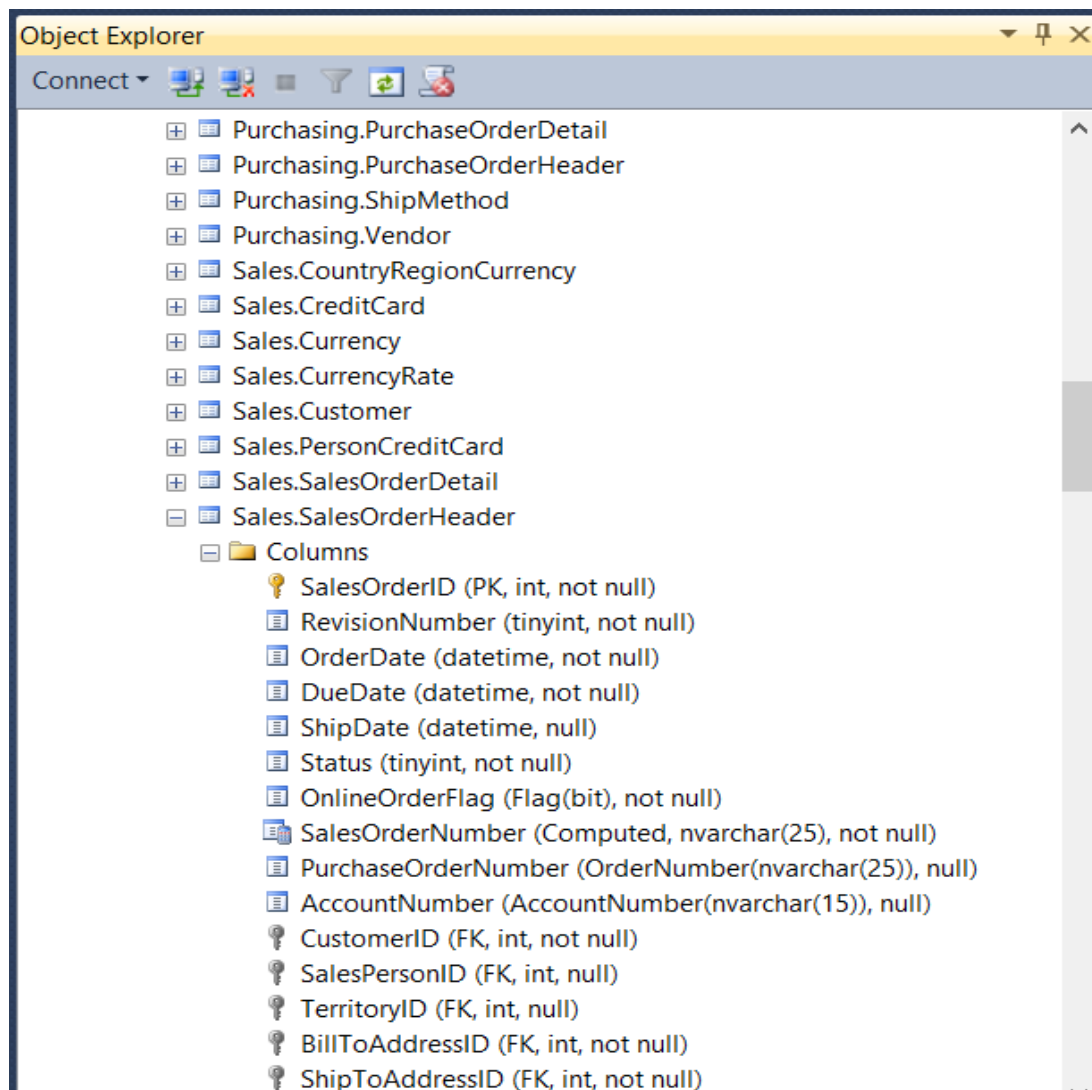
*These exercise questions are for self-practice. No submission is needed.*

**Notes:** The following partial ERD for AdventureWorks2008R2 was generated by SQL Server Management Studio. Use it to locate data when writing SQL queries.



**Notes:** If we don't have an ERD, we can also use the Object Explorer in SQL Server Management Studio to locate data by following the steps listed below.

- 1) Under Object Explorer in SQL Server Management Studio, expand Databases
- 2) Expand the database we want to work with, such as AdventureWorks2008R2
- 3) Expand Tables
- 4) Expand the table we want to work with, such as Sales.Customer
- 5) Expand Columns
- 6) Then we'll see all columns contained in a table



USE "The name of a database you have created.";

### **Exercise 1**

```
/*  
    Create a scalar function that will accept a customer ID and return  
    the customer's account number.  
*/
```

### **Exercise 2**

```
/*  
    Create a table-valued function that will accept a customer ID and  
    return all orders of the customer, including the sales order ID,  
    order date, purchase order number, and total due. Sort the results  
    first by the order date (descending), then by the sales order  
    ID (ascending).
```

Hint: We'll need to use the TOP keyword in the SELECT statement so that we can sort the results. Use an arbitrary number with the TOP keyword, but it needs to be large enough to accommodate the largest possible row set that may be returned.

```
*/
```

### Exercise 3

/\*

Use a WHILE loop to create a stored procedure that takes an int parameter and returns the result of adding all the numbers from 1 to that given number, if the number is 0 or less, return -1.

\*/

For example:

an input of 1 would return 1	(1)
an input of 2 would return 3	(1 + 2)
an input of 3 would return 6	(1 + 2 + 3)
an input of 4 would return 10	(1 + 2 + 3 + 4)
an input of 5 would return 15	(1 + 2 + 3 + 4 + 5)

### Exercise 4

/\*

Convert the following statement into a procedure that will accept a time parameter and extend the transaction by the input amount of time.

NOTE: We need to work on this exercise on our own computer using the AdventureWorks2008R2 database.

\*/

```
BEGIN TRAN;
    UPDATE [Sales].[Customer]
    SET [ModifiedDate] = getdate()
    WHERE [CustomerID] = 100;
    WAITFOR DELAY @d;
ROLLBACK TRAN;
```

### **Exercise 5**

```
/*  
Create a stored procedure containing a WHILE loop that takes  
an integer parameter and prints the consecutive integers  
from the input integer up to 10. If the number is smaller  
than 1 or greater than 10, print "Out of Range."  
*/
```

For example:

An input of 5 would return 5  
6  
7  
8  
9  
10

***Don't look at the solution until you have completed an exercise question.***

## -- Solutions

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

### -- Exercise 1 Solution

```
CREATE FUNCTION uf_GetAccountNumberForCustomer
(@CustID int)
RETURNS varchar(10)
AS
BEGIN
    DECLARE @AcctNo varchar(10);

    SELECT @AcctNo = AccountNumber
    FROM AdventureWorks2008R2.Sales.Customer
    WHERE CustomerID = @CustID;

    RETURN @AcctNo;
END
```

### -- Use the function

```
SELECT dbo.uf_GetAccountNumberForCustomer(29811);
```

### -- Do housekeeping

```
DROP FUNCTION dbo.uf_GetAccountNumberForCustomer;
```

-- Exercise 2 Solution

```
CREATE FUNCTION uf_GetAllOrdersForCustomer
(@CustID int)
RETURNS TABLE
AS
RETURN (SELECT TOP 10000
        SalesOrderID,
        OrderDate,
        PurchaseOrderNumber,
        TotalDue
        FROM AdventureWorks2008R2.Sales.SalesOrderHeader
        WHERE CustomerID = @CustID
        ORDER BY OrderDate DESC, SalesOrderID
        );
```

-- Top keyword must be used so that we can sort the results.  
-- 10000 was arbitrarily picked.

-- Use the function

```
SELECT * FROM dbo.uf_GetAllOrdersForCustomer(29811);
```

-- Do housekeeping

```
DROP FUNCTION dbo.uf_GetAllOrdersForCustomer;
```



-- Exercise 3 Solution

```
CREATE PROCEDURE usp_Calculate
    @InNumber INT,
    @OutNumber INT OUTPUT
AS
BEGIN
    IF @InNumber <= 0
        SET @OutNumber = -1;
    ELSE
        BEGIN
            DECLARE @counter INT;
            SET @counter = 0;
            SET @OutNumber = 0;
            WHILE @counter < @InNumber
                BEGIN
                    SET @counter = @counter + 1;
                    SET @OutNumber = @OutNumber + @counter;
                END
            END
        PRINT @OutNumber;
    END
GO

--Run stored procedure, retrieve output

-- Declare variables
DECLARE @MyInput INT;
DECLARE @MyOutput INT;
-- Initilize variable
SET @MyInput = -5;
-- Execute the procedure
EXEC usp_Calculate @MyInput, @MyOutput OUTPUT;
-- See result
SELECT @MyOutput;

-- Do housekeeping

DROP PROC usp_Calculate;
```

-- Exercise 4 Solution

```
/*  
    Convert the following statement into a procedure that will accept  
    a time parameter and extend the transaction by the input amount  
    of time.  
*/
```

```
CREATE PROC usp_Transaction_Delay @Delay time  
AS  
BEGIN  
    DECLARE @d datetime;  
    SET @d = @Delay;  
    BEGIN TRAN;  
        UPDATE [Sales].[Customer]  
        SET [ModifiedDate] = getdate()  
        WHERE [CustomerID] = 100;  
        WAITFOR DELAY @d;  
    ROLLBACK TRAN;  
END
```

-- Execute the procedure

```
EXEC usp_Transaction_Delay '00:1:00'
```

-- Do housekeeping

```
DROP PROC usp_Transaction_Delay;
```

-- Exercise 5 Solution

```
CREATE PROCEDURE Consecutive
    @InNumber INT
AS
BEGIN
    IF @InNumber < 1 OR @InNumber > 10
        PRINT 'Out of Range';
    ELSE
        BEGIN
            WHILE @InNumber <= 10
            BEGIN
                PRINT @InNumber;
                SET @InNumber = @InNumber + 1;
            END
        END
    END;
END;
```

-- Execute the procedure

```
EXEC dbo.Consecutive 5;
```

-- Do housekeeping

```
DROP PROC dbo.Consecutive;
```