Problem Set for Object Creation

Create and test the objects. Be sure to include drop statements. Use the AdventureWorks2014 database for these questions.

**NOTE: There is no one correct answer for these. Answers given are just suggestions.

1. Write a scalar function that returns the result of two integers multiplied together.

2. SQL Server 2016 and earlier has RTRIM and LTRIM functions which remove extra spaces from the right or left of a string respectively. Starting with SQL Server 2017, there is a new TRIM function which removes spaces from both sides. You are using SQL Server 2016, so create a scalar function called dbo.udf_Trim that has the functionality of the new TRIM function. It should be able to remove the spaces on either side of a 1000 character string.

3. Create a view for the following query:

```
SELECT Cust.StoreID, Store.Name, YEAR(OrderDate) AS OrderYear, SUM(TotalDue) AS TotalSales
FROM Sales.Store AS Store
INNER JOIN Sales.Customer AS Cust ON Store.BusinessEntityID = Cust.StoreID
INNER JOIN Sales.SalesOrderHeader AS SOH ON SOH.CustomerID = Cust.CustomerID
GROUP BY Cust.StoreID, Store.Name, YEAR(OrderDate);
```

4. Add a WHERE clause to the query from question 4 so that it is filtered by StoreID Create an Inline table-valued function using the new query. Make sure that it has the @StoreID parameter.

5. Add a WHERE clause to the query from question 4 so that it is filtered by StoreID Create an *Inline table-valued function* using the new query. Make sure that it has the @StoreID parameter.

6. Create a stored procedure using the query from question 5.

Solution

1. Write a scalar function that returns the result of two integers multiplied together.

```
DROP FUNCTION IF EXISTS dbo.udf_Multiply;
GO
CREATE FUNCTION dbo.udf_Multiply(@Val1 INT, @Val2 INT)
RETURNS INT AS
BEGIN
    RETURN @Val1 * @Val2;
END;
GO
SELECT dbo.udf_Multiply(7,12);
```

2. SQL Server 2016 and earlier has RTRIM and LTRIM functions which remove extra spaces from the right or left of a string respectively. Starting with SQL Server 2017, there is a new TRIM function which removes spaces from both sides. You are using SQL Server 2016, so create a scalar function called **dbo.udf_Trim** that has the functionality of the new TRIM function. It should be able to remove the spaces on either side of a 1000 character string.

```
DROP FUNCTION IF EXISTS dbo.udf_TRIM;

GO
CREATE FUNCTION dbo.udf_TRIM(@String NVARCHAR(1000))
RETURNS NVARCHAR(1000) AS
BEGIN
    SET @String = LTRIM(@String);
    SET @String = RTRIM(@String);

    RETURN @String;
END
GO
DECLARE @NewString NVARCHAR(20) = '    abcde    ';
SELECT @NewString, LEN(@NewString), LEN(dbo.udf_Trim(@NewString));
```

3. Function parameters can have default values. When a default is in place, providing a value for the parameter is optional, but you must specify the keyword default in place of the value. Create a scalar function that adds from 2 to 4 integers, the 3rd and 4th integers will have a default value of 0.

```
DROP FUNCTION IF EXISTS dbo.udf_Add2to4Numbers
GO
CREATE FUNCTION dbo.udf_Add2to4Numbers(@Val1 INT, @Val2 INT, @Val3 INT = 0, @Val4 INT =
0)
RETURNS INT AS
BEGIN
    RETURN @Val1 + @Val2 + @Val3 + @Val4;
END;
GO
SELECT dbo.udf_Add2to4Numbers(10,20,30,default);
SELECT dbo.udf_Add2to4Numbers(5,7, default,default);
```

4. Create a view for the following query:

```
SELECT Cust.StoreID, Store.Name, YEAR(OrderDate) AS OrderYear, SUM(TotalDue) AS TotalSales
FROM Sales.Store AS Store
INNER JOIN Sales.Customer AS Cust ON Store.BusinessEntityID = Cust.StoreID
INNER JOIN Sales.SalesOrderHeader AS SOH ON SOH.CustomerID = Cust.CustomerID
GROUP BY Cust.StoreID, Store.Name, YEAR(OrderDate);
```

```
DROP VIEW IF EXISTS dbo.vwStoreSalesByYear;
GO
CREATE VIEW dbo.vwStoreSalesByYear AS

    SELECT Cust.StoreID, Store.Name, YEAR(OrderDate) AS OrderYear,
            SUM(TotalDue) AS TotalSales
    FROM Sales.Store AS Store
    INNER JOIN Sales.Customer AS Cust ON Store.BusinessEntityID = Cust.StoreID
    INNER JOIN Sales.SalesOrderHeader AS SOH ON SOH.CustomerID = Cust.CustomerID
    GROUP BY Cust.StoreID, Store.Name, YEAR(OrderDate);
GO

SELECT * FROM dbo.vwStoreSalesByYear;
```

5. Add a WHERE clause to the query from question 4 so that it is filtered by StoreID Create an *Inline table-valued function* using the new query. Make sure that it has the @StoreID parameter.

```
DROP FUNCTION IF EXISTS dbo.udfStoreSalesByYear;
GO
CREATE FUNCTION dbo.udfStoreSalesByYear(@StoreID INT)
RETURNS TABLE
AS RETURN(

    SELECT Cust.StoreID, Store.Name, YEAR(OrderDate) AS OrderYear, SUM(TotalDue) AS
TotalSales
    FROM Sales.Store AS Store
    INNER JOIN Sales.Customer AS Cust ON Store.BusinessEntityID = Cust.StoreID
    INNER JOIN Sales.SalesOrderHeader AS SOH ON SOH.CustomerID = Cust.CustomerID
    WHERE Cust.StoreID = @StoreID
    GROUP BY Cust.StoreID, Store.Name, YEAR(OrderDate)
)
GO

SELECT * FROM dbo.udfStoreSalesByYear(292);
```

6. Create a stored procedure using the query from question 5.

```
DROP PROCEDURE IF EXISTS dbo.usp_StoreSalesByYear;
GO
CREATE PROCEDURE dbo.usp_StoreSalesByYear @StoreID INT
AS

        SELECT Cust.StoreID, Store.Name, YEAR(OrderDate) AS OrderYear, SUM(TotalDue) AS
TotalSales
        FROM Sales.Store AS Store
        INNER JOIN Sales.Customer AS Cust ON Store.BusinessEntityID = Cust.StoreID
        INNER JOIN Sales.SalesOrderHeader AS SOH ON SOH.CustomerID = Cust.CustomerID
        WHERE Cust.StoreID = @StoreID
        GROUP BY Cust.StoreID, Store.Name, YEAR(OrderDate)

GO

EXEC dbo.usp_StoreSalesByYear @StoreID = 292;
```