View and Function

.NET CORE

A View is a way to create a SQL table virtually for a specific purpose, such as to present data to a user in a way that safeguards the data from malicious intent.

HTTPS://DOCS.MICROSOFT.COM/EN-US/SQL/T-SQL/STATEMENTS/CREATE-VIEW-TRANSACT-SQL?VIEW=SQL-SERVER-VER15

SQL – Computed Columns

https://docs.microsoft.com/en-us/sql/relational-databases/tables/specify-computed-columns-in-a-table?view=sql-server-ver15

A Computed Column is a virtual column whose value is based on some computation done on other columns within the table. It is not physically stored in the table unless the column is marked ***PERSISTED***.

A computed column expression can use data from other columns to calculate a value for the column to which it belongs.

Use the keyword AS to designate a column as a Computed Column.

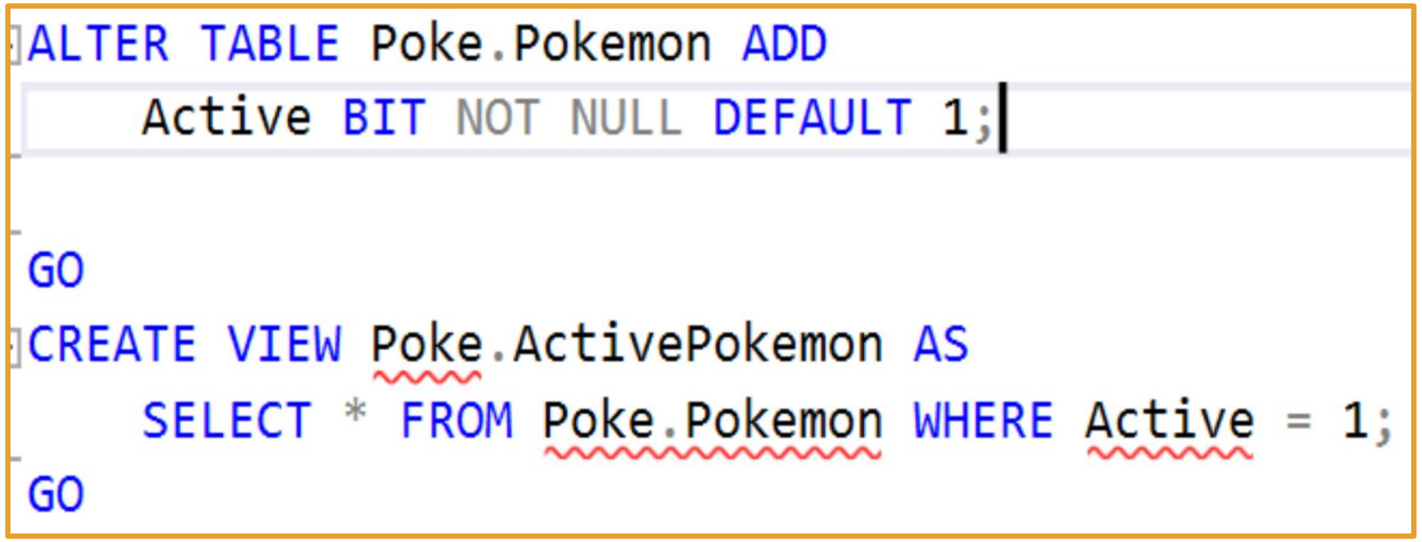
```
CREATE TABLE dbo.Products

(ProductID int IDENTITY (1,1) NOT NULL,

QtyAvailable smallint,

UnitPrice money,

InventoryValue

AS

QtyAvailable * UnitPrice);
```

SQL – Computed Tables (Views)

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql?view=sql-server-ver15

A Computed Table is a virtual table whose contents are defined by a query. A view can be used for:

- To focus, simplify, and customize the perception each user has of the database.
- As a security mechanism by allowing users to access data through the view, without granting the users permissions to directly access the underlying base tables.
- To provide a backward compatible interface to emulate a table whose schema has changed.



```
ALTER TABLE Poke.Pokemon ADD
    Active BIT NOT NULL DEFAULT 1;

GO
CREATE VIEW Poke.ActivePokemon AS
    SELECT * FROM Poke.Pokemon WHERE Active = 1;
GO
```

View – WITH SCHEMABINDING

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-view-transact-sql?view=sql-server-ver15#arguments
https://www.tutorialspoint.com/sql/sql-using-views.htm

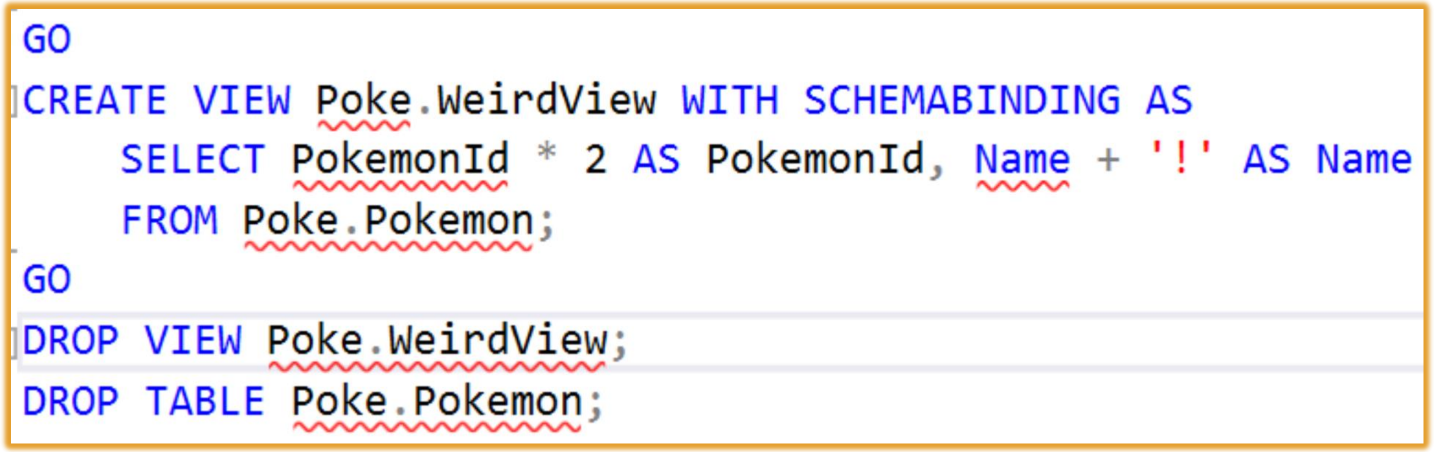
When *SCHEMABINDING* is specified:

- the base table(s) cannot be modified in a way that would affect the view definition.
- The view definition itself must first be modified or dropped to remove dependencies on the table that is to be modified.
- the *SELECT* statement must include the two-part names (schema.object) of tables, views, or user-defined functions that are referenced.
- All referenced objects must be in the same database.

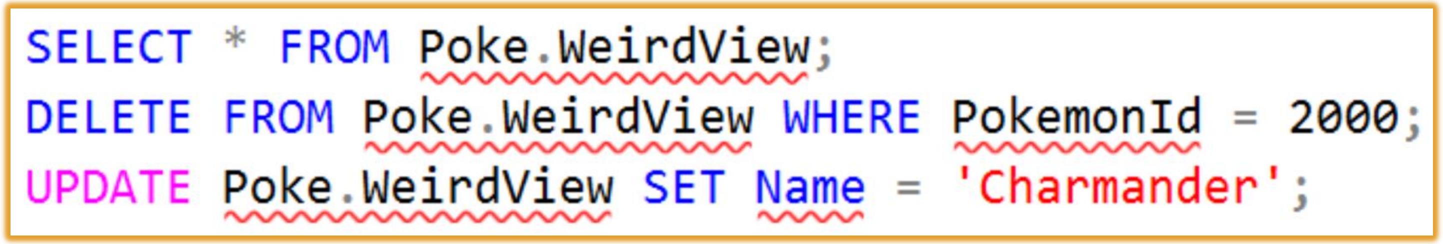
```
CREATE VIEW view_name
WITH SCHEMABINDING
AS
SELECT column1, column2...
FROM table_name
WHERE [condition];
```

View – WITH SCHEMABINDING

WITH SCHEMABINDING sets up a "hard" reference from the view to the table. The view prevents any changes to that table that would “break” the view's query



```
GO
CREATE VIEW Poke.WeirdView WITH SCHEMABINDING AS
    SELECT PokemonId * 2 AS PokemonId, Name + '!' AS Name
    FROM Poke.Pokemon;
GO
DROP VIEW Poke.WeirdView;
DROP TABLE Poke.Pokemon;
```



```
SELECT * FROM Poke.WeirdView;
DELETE FROM Poke.WeirdView WHERE PokemonId = 2000;
UPDATE Poke.WeirdView SET Name = 'Charmander';
```

SQL Scalar Functions

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-function-transact-sql?view=sql-server-ver15
https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functions/create-user-defined-functions-database-engine?view=sql-server-ver15#Scalar
https://www.sqlservertutorial.net/sql-server-user-defined-functions/sql-server-scalar-functions/

A user-defined function accepts parameters, performs an action (such as a complex calculation), and returns the result of that action as a ***scalar*** (single) value or a table.

Scalar Function - SQL Server scalar function takes one or more parameters and returns a single value.

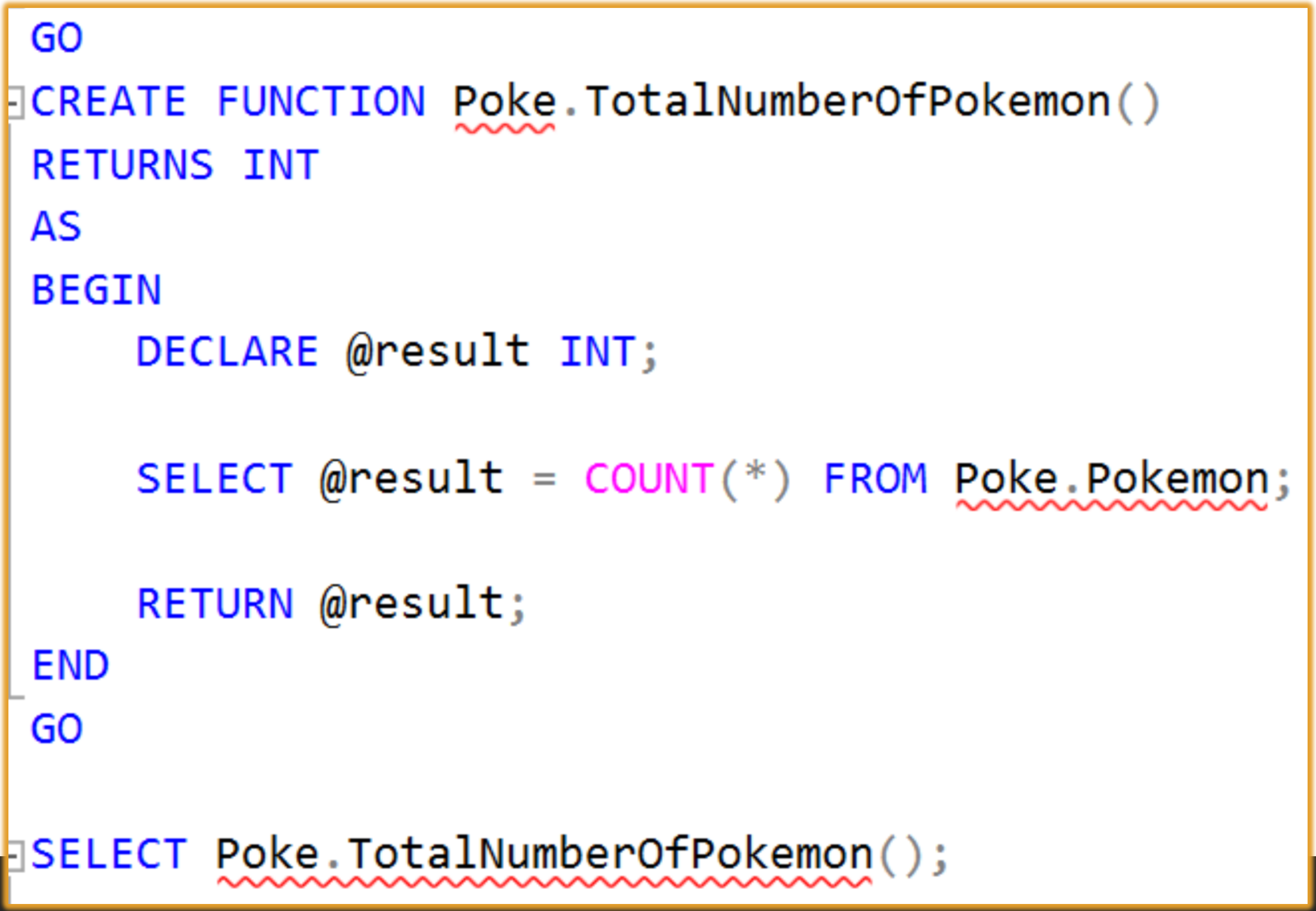
To create a Scalar Function:

1. Use the CREATE FUNCTION keywords to name the function. The schema name is optional. SQL Server may require dbo in front.
2. Specify a list of @parameters in parentheses.
3. Use the RETURNS keyword and give the data type of the return value.
4. User the AS keyword and BEGIN to start the body of the function.
5. RETURN the calculation
6. End the body of the function with END
7. To call the function, SELECT [functionName(params)] AS [name]

```
CREATE FUNCTION dbo.GetNetSale
( @quantity int,
  @unitprice dec(10,2),
  @discount dec(10,2)
)
RETURNS dec(10,2)
AS
BEGIN
  return @quantity*@unitprice*(1-@discount);
END

-- call the function
SELECT dbo.GetNetSale(10,100.00,0.1)
AS
netSale;
```

SQL – User-Defined Functions

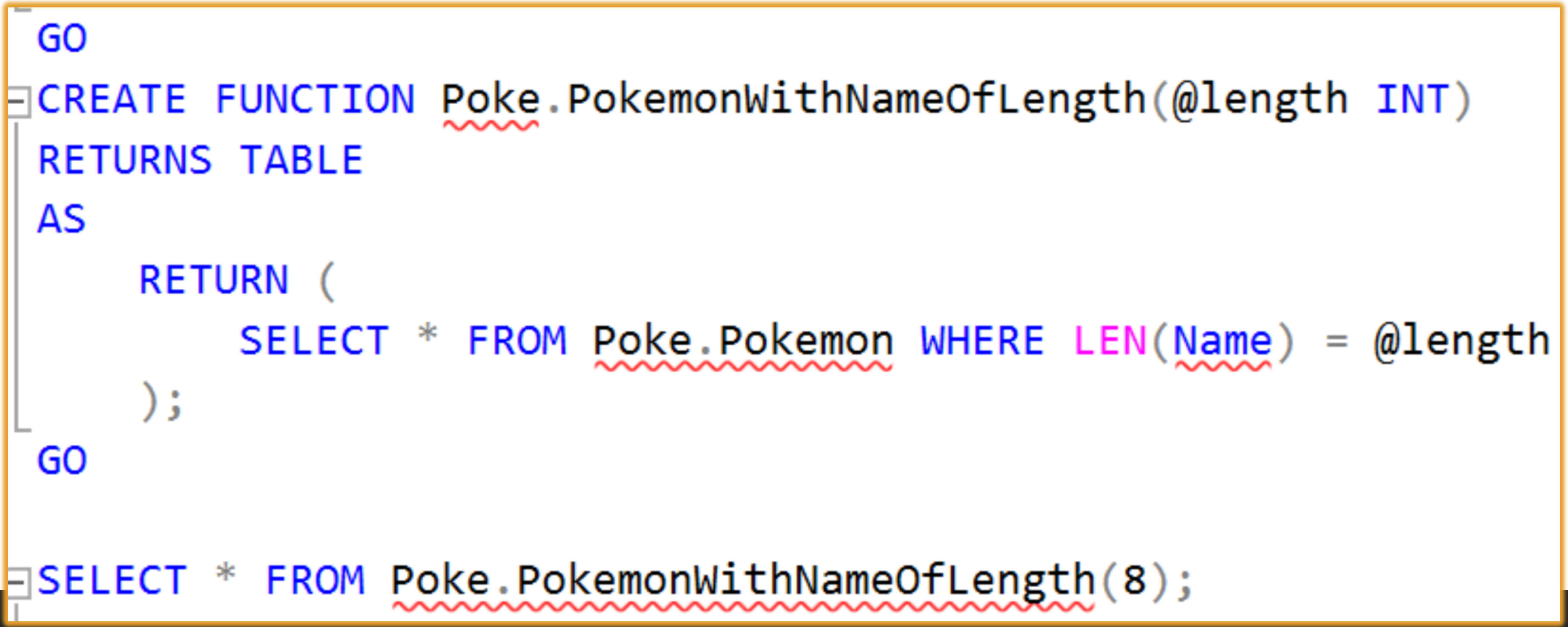
This is a Scalar Function (it returns a single value). Scalar Functions operate on a single value and then return a single value. Scalar functions can be used wherever an expression is valid.



```sql
GO
CREATE FUNCTION Poke.TotalNumberOfPokemon()
RETURNS INT
AS
BEGIN
    DECLARE @result INT;

    SELECT @result = COUNT(*) FROM Poke.Pokemon;

    RETURN @result;
END
GO

SELECT Poke.TotalNumberOfPokemon();
```

SQL – User-Defined Functions

Functions cannot make changes to the database. They have "read-only" access.



```
GO
CREATE FUNCTION Poke.PokemonWithNameOfLength(@length INT)
RETURNS TABLE
AS
    RETURN (
        SELECT * FROM Poke.Pokemon WHERE LEN(Name) = @length
    );
GO

SELECT * FROM Poke.PokemonWithNameOfLength(8);
```

Table-Valued Parameters

https://docs.microsoft.com/en-us/sql/relational-databases/tables/use-table-valued-parameters-database-engine?view=sql-server-ver15

A Table-Valued Parameter is a Function parameter that is actually a SQL table.

This example creates a ***table-valued*** parameter type, declares a variable to reference it, fills the parameter list, and then passes the values to a stored procedure in the AdventureWorks database.

```sql
/* Create a table type. */
CREATE TYPE LocationTableType
   AS TABLE
      ( LocationName VARCHAR(50)
      , CostRate INT );
GO
/* Create a procedure to receive data for the table-valued parameter. */
CREATE PROCEDURE dbo. usp_InsertProductionLocation
   @TVP LocationTableType READONLY
      AS
      SET NOCOUNT ON
      INSERT INTO AdventureWorks2012.Production.Location
         (
            Name
            , CostRate
            , Availability
            , ModifiedDate
         )
      SELECT *, 0, GETDATE()
      FROM @TVP;
GO
/* Declare a variable that references the type. */
DECLARE @LocationTVP AS LocationTableType;
/* Add data to the table variable. */
INSERT INTO @LocationTVP (LocationName, CostRate)
   SELECT Name, 0.00
   FROM AdventureWorks2012.Person.StateProvince;

/* Pass the table variable data to a stored procedure. */
EXEC usp_InsertProductionLocation @LocationTVP;
```

Function access

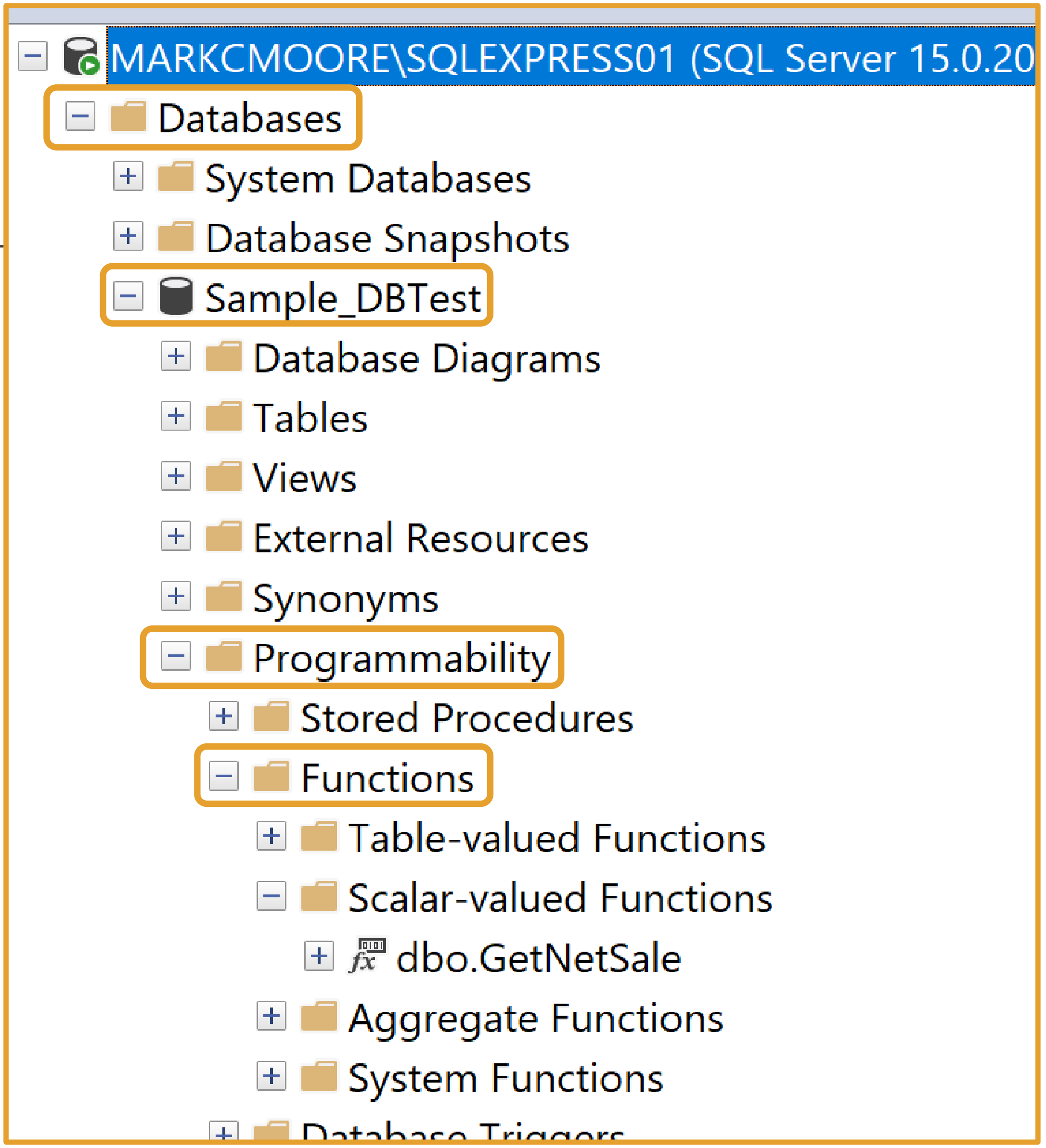
Object Explorer

>>Databases

>>[DbName]

>>Programmability

>>Functions



MARKCMOORE\SQLEXPRESS01 (SQL Server 15.0.20
- Databases
  - System Databases
  - Database Snapshots
  - Sample_DBTest
    - Database Diagrams
    - Tables
    - Views
    - External Resources
    - Synonyms
    - Programmability
      - Stored Procedures
      - Functions
        - Table-valued Functions
        - Scalar-valued Functions
          - dbo.GetNetSale
        - Aggregate Functions
        - System Functions
      - Database Triggers