(T12)自訂UserDefinedFunction  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T12)自訂UserDefinedFunction  
=======================================================================  
0. What to learn

1. CreateSampleData

2. Scalar-Valued functions ()

3. Scalar-Valued functions ()

4. InlineTableValueFunction

5. MultiStatementTableValuedFunctions

6. (Non-)Deterministic\_With(EncryptionSchemaBinding)

7. Clean up  
=======================================================================

0. What to learn

- What to learn

1.

User Defined functions has 3 types

1.1. Scalar-Valued functions ()

---------------

1.1.0.

Drop function if it exist

--IF ( EXISTS ( SELECT    \*

--              FROM      INFORMATION\_SCHEMA.ROUTINES

--              WHERE     ROUTINE\_TYPE = 'FUNCTION'

--                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

--                        AND SPECIFIC\_NAME = 'fnDurationByDate' ) )

--    BEGIN

--        DROP FUNCTION fnDurationByDate;

--    END;

--GO -- Run the previous command and begins new batch

---------------

1.1.1.

The Syntax to Create or Alter Function

--CREATE FUNCTION ScalarValuedFunctionName

----Alter FUNCTION ScalarValuedFunctionName

--(

--    @Parameter1 DataType,

--    @Parameter2 DataType,

--    ..

--    @Parametern Datatype)

--RETURNS Return\_Datatype

--AS

--BEGIN

--    Function Body

--    Return Return\_Datatype

--END

---------------

1.1.2.

2 ways to call Scalar-valued Function

-- [DatabaseName].[SchemaName].[ScalarValuedFunctionName]

or

--USE [DatabaseName]

--GO

--[SchemaName].[ScalarValuedFunctionName]

---------------

1.1.3.

Database Name --> Programmability -->

Functions  --> Scalar-valued Function

1.1.4.

May or may not have parameters.

Return a any data type single (scalar) value.

Except text, ntext, image, cursor, and timestamp.

1.1.5.

Stored Procedure can NOT be used in a

SELECT or WHERE clause,

but Scalar-Valued functions can.

----------------

1.2. Inline Table-Valued functions

1.2.1.

--CREATE FUNCTION TanleValueFunctionName

--(

--       @Param1 DataType,

--       @Param2 DataType

--       ...,

--       @ParamN DataType

--)

--RETURNS TABLE

--AS

--RETURN (

--       Select\_Statement

--)

1.2.2.

--SELECT  \*

--FROM    TanleValueFunctionName('Male');

Table-Valued functions acting as normal table.

It Can used in FROM clause, and Join Other Table.

It Can used in Functional View.

1.2.3.

Database Name --> Programmability -->

Functions  --> Table-Valued functions

----------------

1.3. Multistatement Table-Valued Functions(MSTVF)

1.3.1.

--CREATE FUNCTION fn\_MultistatementTableValuedFunctionName ( )

--RETURNS @Table TABLE

--    (

--      parameter1 dataType ,

--      parameter2 dataType ,

--      ...

--    )

--AS

--    BEGIN

--        INSERT  INTO @Table

--                SELECT  parameter1 dataType ,

--                        parameter2 dataType ,

--                        ...

--                FROM    Table

--        RETURN;

--    END;

1.3.2.

Inline Table-Valued Function can use in View or underlying table.

But MutilStatement Table-Valued function can not.

E.g.1.

--UPDATE  fn\_ILTVF\_GetGallGamers()

--SET     [Name] += 'New'

--WHERE   GamerID = 1;

ILTVF stand for inline Table-Valued Function

E.g.2.

--UPDATE  fn\_MSTVF\_GetALLGamers()

--SET     [Name] += 'New'

--WHERE   GamerID = 1;

Error Message

--Msg 270, Level 16, State 1, Line 586

--Object 'fn\_MSTVF\_GetALLGamers' cannot be modified.

2.

Deterministic VS Nondeterministic Function

Reference:

<https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functions/deterministic-and-nondeterministic-functions>

2.1. Deterministic Function

The same input always get the the same output.

E.g.

All Aggregate function are Deterministic Function.

Sum(), AVG(), Square(), Power() and Count().

----------

2.2. Nondeterministic Function

The same input but always return differenct output.

E.g.

GetDate() and CURRENT\_TIMESTAMP

Rand() function is a Non-deterministic function

But Rand(1) where seed=1 is Deterministic Function

-----------

3.

Scalar-Valued Function

3.1.

--With Encryption

After encryption, you may not read the text of Function any more.

3.2.

--With SchemaBinding

After SchemaBinding, you may NOT drop the afftected table any more.

==================================================

1. CreateSampleData

--=======================================================================

--T012\_01\_CreateSampleData

--=======================================================================

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer' ) )

    BEGIN

        TRUNCATE TABLE Gamer;

        DROP TABLE Gamer;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Team' ) )

    BEGIN

        TRUNCATE TABLE Team;

        DROP TABLE Team;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE Team

(

  TeamID INT IDENTITY(1, 1)

             PRIMARY KEY

             NOT NULL ,

  TeamName NVARCHAR(100) NULL

);

GO -- Run the previous command and begins new batch

INSERT  Team

VALUES  ( N'Team01' );

INSERT  Team

VALUES  ( N'Team02' );

INSERT  Team

VALUES  ( N'Team03' );

INSERT  Team

VALUES  ( N'Team04' );

GO -- Run the previous command and begins new batch

CREATE TABLE Gamer

(

  GamerID INT IDENTITY(1, 1)

              PRIMARY KEY

              NOT NULL ,

  [Name] NVARCHAR(100) NULL ,

  Email NVARCHAR(500) NULL ,

  TeamID INT FOREIGN KEY REFERENCES Team ( TeamID )

             NOT NULL ,

  RegisteredDateTime DATETIME NULL,

);

GO -- Run the previous command and begins new batch

INSERT  Gamer

VALUES  ( N'Name6', N'[6@6.com](mailto:6@6.com)', 1,

          CAST(N'2016-09-08T18:54:32.033' AS DATETIME) );

INSERT  Gamer

VALUES  ( N'Name7', N'[7@7.com](mailto:7@7.com)', 2,

          CAST(N'2016-01-27T21:30:28.473' AS DATETIME) );

INSERT  Gamer

VALUES  ( N'Name8', N'[8@8.com](mailto:8@8.com)', 2,

          CAST(N'2016-09-08T12:35:29.050' AS DATETIME) );

INSERT  Gamer

VALUES  ( N'Name9', N'[9@9.com](mailto:9@9.com)', 1,

          CAST(N'2016-01-27T13:19:34.267' AS DATETIME) );

INSERT  Gamer

VALUES  ( N'Name10', N'[10@10.com](mailto:10@10.com)', 3,

          CAST(N'2016-09-08T12:22:37.597' AS DATETIME) );

INSERT  Gamer

VALUES  ( N'Name11', N'[11@11.com](mailto:11@11.com)', 1,

          CAST(N'2016-01-27T12:22:37.597' AS DATETIME) );

INSERT  Gamer

VALUES  ( N'Name12', N'[12@12.com](mailto:12@12.com)', 2,

          CAST(N'2011-11-01T07:51:48.177' AS DATETIME) );

INSERT  Gamer

VALUES  ( N'Name13', N'[13@13.com](mailto:13@13.com)', 2,

          CAST(N'2012-09-03T22:01:04.580' AS DATETIME) );

INSERT  Gamer

VALUES  ( N'Name14', N'[14@14.com](mailto:14@14.com)', 2,

          CAST(N'2016-01-27T01:28:02.657' AS DATETIME) );

INSERT  Gamer

VALUES  ( N'Name15', N'[15@15.com](mailto:15@15.com)', 1,

          CAST(N'2016-09-08T00:28:44.183' AS DATETIME) );

GO -- Run the previous command and begins new batch

==================================================

2. Scalar-Valued functions ()

--================================================================

--T012\_02\_Scalar-Valued functions ()

--================================================================

--Revise

--T008\_07\_fnDurationByDate

--================================================================

--T008\_07\_01

--fnDurationByDate

/\*

/// <summary>

/// Input a date, then return the string value of duration between that date to today.

/// E.g. 33 Years 5 Months 14 Days

/// </summary>

/// <param name="Date">The input date</param>

/// <returns>The string value of duration between that date to today </returns>

\*/

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnDurationByDate' ) )

    BEGIN

        DROP FUNCTION fnDurationByDate;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fnDurationByDate ( @Date DATETIME )

RETURNS NVARCHAR(50)

AS

    BEGIN

        DECLARE @tempdate DATETIME ,

            @years INT ,

            @months INT ,

            @days INT;

        SELECT  @tempdate = @Date;

             -- Caculate Years

        SELECT  @years = DATEDIFF(YEAR, @tempdate, GETDATE())

                - CASE WHEN ( MONTH(@Date) > MONTH(GETDATE()) )

                            OR ( MONTH(@Date) = MONTH(GETDATE())

                                 AND DAY(@Date) > DAY(GETDATE())

                               ) THEN 1

                       ELSE 0

                  END;

        SELECT  @tempdate = DATEADD(YEAR, @years, @tempdate);

             -- Caculate Months

        SELECT  @months = DATEDIFF(MONTH, @tempdate, GETDATE())

                - CASE WHEN DAY(@Date) > DAY(GETDATE()) THEN 1

                       ELSE 0

                  END;

        SELECT  @tempdate = DATEADD(MONTH, @months, @tempdate);

             -- Caculate Days

        SELECT  @days = DATEDIFF(DAY, @tempdate, GETDATE());

        DECLARE @Duration NVARCHAR(50);

        SET @Duration = CAST(@years AS NVARCHAR(4)) + ' Years '

            + CAST(@months AS NVARCHAR(2)) + ' Months '

            + CAST(@days AS NVARCHAR(2)) + ' Days';

        RETURN @Duration;

    END;

GO -- Run the prvious command and begins new batch

--================================================================

--T008\_07\_02

--fnDurationByDate2

/\*

/// <summary>

/// Input a date, then return the string value of duration between that date to today.

/// E.g. 33 Years 5 Months 14 Days

/// </summary>

/// <param name="Date">The input date</param>

/// <returns>The string value of duration between that date to today </returns>

\*/

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnDurationByDate2' ) )

    BEGIN

        DROP FUNCTION fnDurationByDate2;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fnDurationByDate2 ( @Date DATETIME )

RETURNS NVARCHAR(50)

AS

    BEGIN

        DECLARE @tempdate DATETIME ,

            @years INT ,

            @months INT ,

            @days INT;

        SET @tempdate = @Date;

             -- Caculate Years

        IF ( MONTH(@Date) > MONTH(GETDATE()) )

            OR ( MONTH(@Date) = MONTH(GETDATE())

                 AND DAY(@Date) > DAY(GETDATE())

               )

            BEGIN

                SET @years = DATEDIFF(YEAR, @tempdate, GETDATE()) - 1;

            END;

        ELSE

            BEGIN

                SET @years = DATEDIFF(YEAR, @tempdate, GETDATE());

            END;

             -- Caculate Months

        SET @tempdate = DATEADD(YEAR, @years, @tempdate);

        IF DAY(@Date) > DAY(GETDATE())

            BEGIN

                SET @months = DATEDIFF(MONTH, @tempdate, GETDATE()) - 1;

            END;

        ELSE

            BEGIN

                SET @months = DATEDIFF(MONTH, @tempdate, GETDATE());

            END;

             -- Caculate Days

        SET @tempdate = DATEADD(MONTH, @months, @tempdate);

        SET @days = DATEDIFF(DAY, @tempdate, GETDATE());

        DECLARE @Duration NVARCHAR(50);

        SET @Duration = CAST(@years AS NVARCHAR(4)) + ' Years '

            + CAST(@months AS NVARCHAR(2)) + ' Months '

            + CAST(@days AS NVARCHAR(2)) + ' Days';

        RETURN @Duration;

    END;

GO -- Run the prvious command and begins new batch

PRINT dbo.fnDurationByDate('1984/11/26');

PRINT dbo.fnDurationByDate2('1984-11-26');

--32 Years 9 Months 14 Days

PRINT dbo.fnDurationByDate('1984/09/10');

PRINT dbo.fnDurationByDate2('1984-09-10');

--32 Years 11 Months 30 Days

PRINT dbo.fnDurationByDate('1984/09/09');

PRINT dbo.fnDurationByDate2('1984-09-09');

--33 Years 0 Months 0 Days

PRINT dbo.fnDurationByDate('1984/09/08');

PRINT dbo.fnDurationByDate2('1984-09-08');

--33 Years 0 Months 1 Days

/\*

2 ways to call Scalar-valued Function

-- [DatabaseName].[SchemaName].[FunctionName]

or

--USE [DatabaseName]

--GO

--[SchemaName].[FunctionName]

\*/

DECLARE @tempdate2 DATETIME;

SET @tempdate2 = CAST('1984/11/26' AS DATETIME);

PRINT @tempdate2;

--Nov 26 1984 12:00AM

SET @tempdate2 = DATEADD(YEAR, 32, @tempdate2);

PRINT @tempdate2;

--Nov 26 2016 12:00AM

SET @tempdate2 = DATEADD(MONTH, 9, @tempdate2);

PRINT @tempdate2;

--Aug 26 2017 12:00AM

SET @tempdate2 = DATEADD(DAY, 14, @tempdate2);

PRINT @tempdate2;

GO -- Run the previous command and begins new batch

--Sep  9 2017 12:00AM

/\*

I assume today is 2017/09/09  (YYYY/MM/DD)

I assume inputDate is 1984/11/26  (YYYY/MM/DD)

The difference shoud be '32 Years 9 Months 14 Days'

\*/

/\*

1.

---- Caculate Years

--IF ( MONTH(@Date) > MONTH(GETDATE()) )

--    OR ( MONTH(@Date) = MONTH(GETDATE())

--            AND DAY(@Date) > DAY(GETDATE())

--        )

--    BEGIN

--        SET @years = DATEDIFF(YEAR, @tempdate, GETDATE()) - 1;

--    END;

--ELSE

--    BEGIN

--        SET @years = DATEDIFF(YEAR, @tempdate, GETDATE());

--    END;

--SET @tempdate = DATEADD(YEAR, @years, @tempdate);

1.1.

I assume today is 2017/09/09  (YYYY/MM/DD)

I assume inputDate is 1984/11/26  (YYYY/MM/DD)

Shoud return '32 Years 9 Months 14 Days'

but 2017-1984=33, thus, It should minus 1, 33-1=32

1.2.

I assume today is 2017/09/09  (YYYY/MM/DD)

I assume inputDate is 1984/09/10  (YYYY/MM/DD)

Shoud return '32 Years 11 Months 30 Days'

but 2017-1984=33, thus, It should minus 1, 33-1=32

1.3.

I assume today is 2017/09/09  (YYYY/MM/DD)

I assume inputDate is 1984/09/09  (YYYY/MM/DD)

Shoud return '33 Years 0 Months 0 Dayss'

2017-1984=33

1.4.

I assume today is 2017/09/09  (YYYY/MM/DD)

I assume inputDate is 1984/09/08  (YYYY/MM/DD)

Should return 33 Years 0 Months 1 Days

2017-1984=33

1.5.

In Summary, when caculating the "Years"

--IF ( MONTH(@Date) > MONTH(GETDATE()) )

--    OR ( MONTH(@Date) = MONTH(GETDATE())

--            AND DAY(@Date) > DAY(GETDATE())

--        )

If the Month and Day of inputDate is later than the Month and Day of currentDate

Then the years is  DATEDIFF(YEAR, @tempdate, GETDATE()) - 1

If the Month and Day of inputDate is earlier than the Month and Day of currentDate

Then the years is  DATEDIFF(YEAR, @tempdate, GETDATE())

2.

---- Caculate Months

--SET @tempdate = DATEADD(YEAR, @years, @tempdate);

--IF DAY(@Date) > DAY(GETDATE())

--    BEGIN

--        SET @months = DATEDIFF(MONTH, @tempdate, GETDATE()) - 1;

--    END;

--ELSE

--    BEGIN

--        SET @months = DATEDIFF(MONTH, @tempdate, GETDATE());

--    END;

2.1.

--SET @tempdate = DATEADD(YEAR, @years, @tempdate);

After we get the years, then we add the years to TempDate which was originally currentDate.

Then the different between @tempdate and currentDate should be less than 1 year.

The @tempdate is less than 1 year means between 0 Months 0 days to 11 months and 30 Days.

2.2.

In Summary, when caculating the "Months"

--IF DAY(@Date) > DAY(GETDATE())

If the Day of inputDate is later than the Day of currentDate

Then the Month is  DATEDIFF(MONTH, @tempdate, GETDATE()) - 1

If the Day of inputDate is earlier than the Day of currentDate

Then the Month is  DATEDIFF(MONTH, @tempdate, GETDATE())

3.

---- Caculate Days

--SET @tempdate = DATEADD(MONTH, @months, @tempdate);

--SET @days = DATEDIFF(DAY, @tempdate, GETDATE());

3.1.

--SET @tempdate = DATEADD(YEAR, @years, @tempdate);

...

--SET @tempdate = DATEADD(MONTH, @months, @tempdate);

After we get the Months and Years, then we add the Months and Years to TempDate which was originally currentDate.

Then the different between @tempdate and currentDate should be less than the Days.

3.1.1.

--DECLARE @tempdate2 DATETIME;

--SET @tempdate2 = CAST('1984/11/26' AS DATETIME);

--PRINT @tempdate2

----Nov 26 1984 12:00AM

--SET @tempdate2 = DATEADD(YEAR, 32, @tempdate2);

--PRINT @tempdate2

----Nov 26 2016 12:00AM

--SET @tempdate2 = DATEADD(MONTH, 9, @tempdate2);

--PRINT @tempdate2

----Aug 26 2017 12:00AM

--SET @tempdate2 = DATEADD(DAY, 14, @tempdate2);

--PRINT @tempdate2

----Sep  9 2017 12:00AM

I assume today is 2017/09/09  (YYYY/MM/DD)

I assume inputDate is 1984/11/26  (YYYY/MM/DD)

Shoud return '32 Years 9 Months 14 Days'

but 2017-1984=33, thus, It should minus 1, 33-1=32

Nov to Sep is 10 months different, but, it should minus 1, 10-1=9

Then 1984/11/26 add 32 yaers and 10 Month

Thus, we add 32 years and 9 month into the inputDate.

The the difference between inputDate and CurrentDate will be less than 30 adys.

Thus,

--SET @days = DATEDIFF(DAY, @tempdate, GETDATE());

To caculate the date, we do not need the if statmet to minus 1 any more.

Go straight to get the DATEDIFF to get Days.

\*/

==================================================

3. Scalar-Valued functions ()

--================================================================

--T012\_03\_Scalar-Valued functions ()

--================================================================

--T012\_03

----fnYearDurationByDate

--================================================================

--T012\_03\_01

--fnYearDurationByDate

/\*

/// <summary>

/// Input a date, then return the int value of the years duration between that date to today.

/// E.g. 3    it means the input date is 3 years from now.

/// </summary>

/// <param name="Date">The input date</param>

/// <returns>The int value of the years duration between that date to today.</returns>

\*/

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnYearDurationByDate' ) )

    BEGIN

        DROP FUNCTION fnYearDurationByDate;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fnYearDurationByDate ( @Date DATETIME )

--ALTER FUNCTION fnYearDurationByDate ( @Date DATETIME )

RETURNS NVARCHAR(50)

AS

    BEGIN

        DECLARE @tempdate DATETIME ,

            @years INT ,

            @months INT ,

            @days INT;

        SET @tempdate = @Date;

             -- Caculate Years

        IF ( MONTH(@Date) > MONTH(GETDATE()) )

            OR ( MONTH(@Date) = MONTH(GETDATE())

                 AND DAY(@Date) > DAY(GETDATE())

               )

            BEGIN

                SET @years = DATEDIFF(YEAR, @tempdate, GETDATE()) - 1;

            END;

        ELSE

            BEGIN

                SET @years = DATEDIFF(YEAR, @tempdate, GETDATE());

            END;

        RETURN @years;

    END;

GO -- Run the prvious command and begins new batch

/\*

The Syntax to Create or Alter Function

--CREATE FUNCTION ScalarValuedFunctionName

----Alter FUNCTION ScalarValuedFunctionName

--(

--     @Parameter1 DataType,

--     @Parameter2 DataType,

--     ..

--     @Parametern Datatype)

--RETURNS Return\_Datatype

--AS

--BEGIN

--    Function Body

--    Return Return\_Datatype

--END

\*/

--================================================================

--T012\_03\_02

--Gamer

SELECT  \*

FROM    Gamer;

SELECT  p2.GamerID ,

        p2.Name ,

        p2.RegisteredDateTime ,

        dbo.fnDurationByDate(p2.RegisteredDateTime) AS DurationFromRegister ,

        dbo.fnDurationByDate2(p2.RegisteredDateTime) AS DurationFromRegister2 ,

        dbo.fnYearDurationByDate(p2.RegisteredDateTime) AS YearDurationFromRegister

FROM    Gamer p2;

--WHERE      dbo.fnYearDurationByDate(p2.RegisteredDateTime) > 2

SELECT  p2.GamerID ,

        p2.Name ,

        p2.RegisteredDateTime ,

        dbo.fnDurationByDate(p2.RegisteredDateTime) AS DurationFromRegister ,

        dbo.fnDurationByDate2(p2.RegisteredDateTime) AS DurationFromRegister2 ,

        dbo.fnYearDurationByDate(p2.RegisteredDateTime) AS YearDurationFromRegister

FROM    Gamer p2

WHERE   dbo.fnYearDurationByDate(p2.RegisteredDateTime) > 2;

GO -- Run the previous command and begins new batch

sp\_helptext fnYearDurationByDate;

GO -- Run the previous command and begins new batch

/\*

1.

Stored Procedure can NOT be used in a SELECT or WHERE clause, but Function can.

2.

sp\_helptext FunctionName will show the text of the Scalar-valuded Functions.

\*/

==================================================

4. InlineTableValueFunction

--================================================================

--T012\_04\_InlineTableValueFunction

--================================================================

/\*

--fnGamerByTeamID

You may create an Inline Table Value Function,

then join it with other table.

\*/

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnGamerByTeamID' ) )

    BEGIN

        DROP FUNCTION fnGamerByTeamID;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fnGamerByTeamID ( @TeamID NVARCHAR(10) )

RETURNS TABLE

AS

RETURN

    ( SELECT    \*

      FROM      Gamer

      WHERE     TeamID = @TeamID

    );

GO -- Run the prvious command and begins new batch

SELECT  \*

FROM    fnGamerByTeamID(1);

GO -- Run the prvious command and begins new batch

SELECT  \*

FROM    fnGamerByTeamID (1) g

        JOIN Team t ON g.TeamID = t.TeamID;

GO -- Run the prvious command and begins new batch

==================================================

5. MultiStatementTableValuedFunctions

--================================================================

--T012\_05\_MultiStatementTableValuedFunctions

--================================================================

/\*

--InLineTableValuedFunction(ILTVF)

V.S.

--MultiStatementTableValuedFunction(MSTVF)

You may use InLineTableValuedFunction(ILTVF)

to join other table.

You may also update InLineTableValuedFunction(ILTVF)

But you can not do anything with MultiStatementTableValuedFunction(MSTVF)

\*/

--================================================================

--T012\_05\_01

--InLineTableValuedFunction(ILTVF)

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fn\_ILTVF\_GetGallGamers' ) )

    BEGIN

        DROP FUNCTION fn\_ILTVF\_GetGallGamers;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fn\_ILTVF\_GetGallGamers ( )

RETURNS TABLE

AS

RETURN

    ( SELECT    g.GamerID ,

                g.Name ,

                g.Email ,

                g.TeamID ,

                g.RegisteredDateTime

      FROM      Gamer g

    );

GO -- Run the prvious command and begins new batch

--================================================================

--T012\_05\_02

--MultiStatementTableValuedFunction(MSTVF)

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fn\_MSTVF\_GetALLGamers' ) )

    BEGIN

        DROP FUNCTION fn\_MSTVF\_GetALLGamers;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fn\_MSTVF\_GetALLGamers ( )

RETURNS @Table TABLE

(

  GamerID INT ,

  [Name] NVARCHAR(100) ,

  Email NVARCHAR(500) ,

  TeamID INT ,

  RegisteredDateTime DATETIME

)

AS

    BEGIN

        INSERT  INTO @Table

                SELECT  GamerID ,

                        Name ,

                        Email ,

                        TeamID ,

                        RegisteredDateTime

                FROM    Gamer;

        RETURN;

    END;

GO -- Run the prvious command and begins new batch

--================================================================

--T012\_05\_03

--Select from fn\_ILTVF\_GetGallGamers()

--Select from fn\_MSTVF\_GetALLGamers()

--Calling the Inline Table Valued Function:

SELECT  \*

FROM    fn\_ILTVF\_GetGallGamers();

--Calling the Multi-statement Table Valued Function:

SELECT  \*

FROM    fn\_MSTVF\_GetALLGamers();

GO -- Run the prvious command and begins new batch

--================================================================

--T012\_05\_04

--Update fn\_ILTVF\_GetGallGamers()

--Update fn\_MSTVF\_GetALLGamers()

SELECT  \*

FROM    Gamer

WHERE   GamerID = 1;

UPDATE  fn\_ILTVF\_GetGallGamers()

SET     [Name] += 'New'

WHERE   GamerID = 1;

SELECT  \*

FROM    Gamer

WHERE   GamerID = 1;

UPDATE  fn\_ILTVF\_GetGallGamers()

SET     [Name] = 'Name6'

WHERE   GamerID = 1;

SELECT  \*

FROM    Gamer

WHERE   GamerID = 1;

--Error

--UPDATE  fn\_MSTVF\_GetALLGamers()

--SET     [Name] += 'New'

--WHERE   GamerID = 1;

GO -- Run the prvious command and begins new batch

/\*

1.3. Multistatement Table-Valued Functions(MSTVF)

1.3.1.

--CREATE FUNCTION fn\_MultistatementTableValuedFunctionName ( )

--RETURNS @Table TABLE

--    (

--      parameter1 dataType ,

--      parameter2 dataType ,

--      ...

--    )

--AS

--    BEGIN

--        INSERT  INTO @Table

--                SELECT  parameter1 dataType ,

--                                      parameter2 dataType ,

--                                      ...

--                FROM    Table

--        RETURN;

--    END;

1.3.2.

Inline Table-Valued Function can use in View or underlying table.

But MutilStatement Table-Valued function can not.

E.g.1.

--UPDATE  fn\_ILTVF\_GetGallGamers()

--SET     [Name] += 'New'

--WHERE   GamerID = 1;

ILTVF stand for inline Table-Valued Function

E.g.2.

--UPDATE  fn\_MSTVF\_GetALLGamers()

--SET     [Name] += 'New'

--WHERE   GamerID = 1;

Error Message

--Msg 270, Level 16, State 1, Line 586

--Object 'fn\_MSTVF\_GetALLGamers' cannot be modified.

\*/

==================================================

6. (Non-)Deterministic\_With(EncryptionSchemaBinding)

--================================================================

--T012\_06\_(Non-)Deterministic\_With(EncryptionSchemaBinding)

--================================================================

/\*

1.

Deterministic VS Nondeterministic Function

Reference:

<https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functions/deterministic-and-nondeterministic-functions>

1.1. Deterministic Function

The same input always get the the same output.

E.g.

All Aggregate function are Deterministic Function.

Sum(), AVG(), Square(), Power() and Count().

----------

1.2. Nondeterministic Function

The same input but always return differenct output.

E.g.

GetDate() and CURRENT\_TIMESTAMP

Rand() function is a Non-deterministic function

But Rand(1) where seed=1 is Deterministic Function

-----------

2.

Scalar-Valued Function

2.1.

--With Encryption

After encryption, you may not read the text of Function any more.

2.2.

--With SchemaBinding

After SchemaBinding, you may NOT drop the afftected table any more.

\*/

--================================================================

--T012\_06\_01

--Create Sample Data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer2' ) )

    BEGIN

        TRUNCATE TABLE Gamer2;

        DROP TABLE Gamer2;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE Gamer2

(

  GamerID INT IDENTITY(1, 1)

              PRIMARY KEY

              NOT NULL ,

  [Name] NVARCHAR(100) NULL ,

  Email NVARCHAR(500) NULL

);

GO -- Run the previous command and begins new batch

INSERT  Gamer2

VALUES  ( N'Name6', N'[6@6.com](mailto:6@6.com)' );

INSERT  Gamer2

VALUES  ( N'Name7', N'[7@7.com](mailto:7@7.com)' );

INSERT  Gamer2

VALUES  ( N'Name8', N'[8@8.com](mailto:8@8.com)' );

INSERT  Gamer2

VALUES  ( N'Name9', N'[9@9.com](mailto:9@9.com)' );

GO -- Run the previous command and begins new batch

--================================================================

--T012\_06\_02

--Scalar-Valued Function

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fn\_GetGamer2ById' ) )

    BEGIN

        DROP FUNCTION fn\_GetGamer2ById;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fn\_GetGamer2ById ( @Id int )

RETURNS nvarchar(50)

AS

    BEGIN

        RETURN (

                    SELECT [Name]

                    FROM Gamer2

                    WHERE GamerID = @Id

             );

    END;

GO -- Run the prvious command and begins new batch

sp\_helptext fn\_GetGamer2ById;

GO -- Run the prvious command and begins new batch

--================================================================

--T012\_06\_03

--Scalar-Valued Function WITH Encryption

ALTER FUNCTION fn\_GetGamer2ById ( @Id int )

RETURNS nvarchar(50)

--The Change here

    WITH Encryption

AS

    BEGIN

        RETURN (

                    SELECT [Name]

                    FROM Gamer2

                    WHERE GamerID = @Id

             );

    END;

GO -- Run the prvious command and begins new batch

sp\_helptext fn\_GetGamerById;

--Error

GO -- Run the prvious command and begins new batch

/\*

After you Encrypt, then you can not modify or view the function any more.

\*/

--================================================================

--T012\_06\_04

--Scalar-Valued Function With SchemaBinding

--Drop the Table if it exist.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer2' ) )

    BEGIN

        TRUNCATE TABLE Gamer2;

        DROP TABLE Gamer2;

    END;

GO -- Run the previous command and begins new batch

--Drop the function if it exist.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fn\_GetGamer2ById' ) )

    BEGIN

        DROP FUNCTION fn\_GetGamer2ById;

    END;

GO -- Run the previous command and begins new batch

/\*

1.

The normal Scalar-Valued Function can NOT

prevent you to Drop the affected Table.

--sp\_depends databaseObjectName

Normally use sp\_depends to find out the dependency before Drop Table.

2.

Thus, we need With SchemaBinding

\*/

--================================================================

--T012\_06\_05

--Create Sample Data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer2' ) )

    BEGIN

        TRUNCATE TABLE Gamer2;

        DROP TABLE Gamer2;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE Gamer2

(

  GamerID INT IDENTITY(1, 1)

              PRIMARY KEY

              NOT NULL ,

  [Name] NVARCHAR(100) NULL ,

  Email NVARCHAR(500) NULL

);

GO -- Run the previous command and begins new batch

INSERT  Gamer2

VALUES  ( N'Name6', N'[6@6.com](mailto:6@6.com)' );

INSERT  Gamer2

VALUES  ( N'Name7', N'[7@7.com](mailto:7@7.com)' );

INSERT  Gamer2

VALUES  ( N'Name8', N'[8@8.com](mailto:8@8.com)' );

INSERT  Gamer2

VALUES  ( N'Name9', N'[9@9.com](mailto:9@9.com)' );

GO -- Run the previous command and begins new batch

--================================================================

--T012\_06\_06

--Scalar-Valued Function With SchemaBinding

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fn\_GetGamer2ById' ) )

    BEGIN

        DROP FUNCTION fn\_GetGamer2ById;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fn\_GetGamer2ById ( @Id int )

RETURNS nvarchar(50)

--The Change Here

    WITH SchemaBinding

AS

    BEGIN

        RETURN (

                    SELECT [Name]

                    --FROM Gamer2   --Error, WITH SchemaBinding need 2 parts table name, [SchemaName].[TableName]

                    FROM dbo.Gamer2

                    WHERE GamerID = @Id

             );

    END;

GO -- Run the prvious command and begins new batch

sp\_helptext fn\_GetGamer2ById;

GO -- Run the prvious command and begins new batch

--================================================================

--T012\_06\_07

--Drop Scalar-Valued Function With SchemaBinding

DROP TABLE dbo.Gamer2;

GO -- Run the prvious command and begins new batch

/\*

Error Message

--Cannot DROP TABLE 'dbo.Gamer2'

--because it is being referenced by object 'fn\_GetGamer2ById'.

fn\_GetGamer2ById is a WITH SchemaBinding Scalar-Valued Function.

And the dbo.Gamer2 Table is the affected tabled.

Thus,

--DROP TABLE dbo.Gamer;

is not allowed.

\*/

--================================================================

--T012\_06\_08

DROP FUNCTION fn\_GetGamer2ById;

DROP TABLE dbo.Gamer2;

GO -- Run the prvious command and begins new batch

/\*

Once you drop the fn\_GetGamer2ById  Scalar-Valued Function.

Then you may drop the affected table,  dbo.Gamer2.

\*/

==================================================

7. Clean up

--=======================================================================

--T012\_07\_Clean up

--=======================================================================

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer' ) )

    BEGIN

        TRUNCATE TABLE Gamer;

        DROP TABLE Gamer;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Team' ) )

    BEGIN

        TRUNCATE TABLE Team;

        DROP TABLE Team;

    END;

GO -- Run the previous command and begins new batch

-------------------------------------------------------------------------

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnDurationByDate' ) )

    BEGIN

        DROP FUNCTION fnDurationByDate;

    END;

GO -- Run the previous command and begins new batch

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnDurationByDate2' ) )

    BEGIN

        DROP FUNCTION fnDurationByDate2;

    END;

GO -- Run the previous command and begins new batch

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnYearDurationByDate' ) )

    BEGIN

        DROP FUNCTION fnYearDurationByDate;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fn\_ILTVF\_GetGallGamers' ) )

    BEGIN

        DROP FUNCTION fn\_ILTVF\_GetGallGamers;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fn\_MSTVF\_GetALLGamers' ) )

    BEGIN

        DROP FUNCTION fn\_MSTVF\_GetALLGamers;

    END;

GO -- Run the previous command and begins new batch

-------------------------------------------------------------------------

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fn\_GetGamer2ById' ) )

    BEGIN

        DROP FUNCTION fn\_GetGamer2ById;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Gamer2' ) )

    BEGIN

        TRUNCATE TABLE Gamer2;

        DROP TABLE Gamer2;

    END;

GO -- Run the previous command and begins new batch