User Defined function

1. Non-Parameterize function
2. Parametrized function
3. Non-Parameterize function

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Create \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CREATE FUNCTION dbo.GetCurrentDateTime()

RETURNS DATETIME

AS

BEGIN

RETURN GETDATE();

END;

--- select ---

select dbo.GetCurrentDateTime();

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Alter \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

alter FUNCTION dbo.GetCurrentDateTime()

RETURNS DATETIME

AS

BEGIN

-- initialize variable--

declare @date datetime;

-- declare variable--

set @date = Convert(varchar(25),GETDATE(),100);

-- assighn to return --

RETURN @date;

END;

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Custome Date \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

alter function dbo.GetCurrentDateTime()

returns nvarchar(45) --- jo return kra ga us ke type

as

begin

return Format(getDate() , 'yyyy-MMMM-dddd hh:mm:ss tt');

end

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ drop \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

drop function dbo.GetCurrentDateTime

1. Parametrize function

CREATE FUNCTION dbo.AddNumbers ( @Num1 INT, @Num2 INT )

RETURNS INT

AS

BEGIN

RETURN @Num1 + @Num2;

END;

-- select ---

select dbo.AddNumbers(25,25);

-- drop ---

Drop function dbo.AddNumbers();

**Example :** Calculate Age

CREATE FUNCTION dbo.AgeInYearsAndMonths(@date\_of\_birth DATE)

RETURNS VARCHAR(50)

AS

BEGIN

DECLARE @age\_in\_years INT, @age\_in\_months INT, @result VARCHAR(50)

-- Calculate age in years and months

SELECT @age\_in\_years = DATEDIFF(YEAR, @date\_of\_birth, GETDATE()) -

CASE

WHEN MONTH(@date\_of\_birth) > MONTH(GETDATE())

OR (MONTH(@date\_of\_birth) = MONTH(GETDATE())

AND DAY(@date\_of\_birth) > DAY(GETDATE()))

THEN 1

ELSE 0

END,

@age\_in\_months = (YEAR(GETDATE()) - YEAR(@date\_of\_birth)) \* 12 + MONTH(GETDATE()) - MONTH(@date\_of\_birth) -

CASE WHEN DAY(@date\_of\_birth) > DAY(GETDATE()) THEN 1

ELSE 0 END

-- Format the result string

SELECT @result = CONCAT(@age\_in\_years, ' years, ', @age\_in\_months, ' months')

-- Return the result

RETURN @result

END

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Execute Function \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

select dbo.AgeInYearsAndMonths('1996-10-28')

**Types of Functions**

1. Scaler function

* *Return Single values in the form of integer, string, date ……*
* *Table is Not Nassery.*

***Example 1:*** *take 1st,last name give full name*

CREATE FUNCTION dbo.GetFullName (@firstName NVARCHAR(50), @lastName NVARCHAR(50))

RETURNS NVARCHAR(101)

BEGIN

DECLARE @fullName NVARCHAR(101)

SELECT @fullName = @firstName + ' ' + @lastName

RETURN @fullName

END

select dbo.GetfullName('Muhammad','Boota') as fullName;

***Example 2:*** *Extract FULLName to 1st , last name*

-- create function to extract first name from full name

CREATE FUNCTION dbo.GetFirstName(@fullName VARCHAR(50))

RETURNS VARCHAR(50)

AS

BEGIN

DECLARE @firstName VARCHAR(50)

SET @firstName = SUBSTRING(@fullName, 1, CHARINDEX(' ', @fullName) - 1)

RETURN @firstName

END;

-- create function to extract last name from full name

CREATE FUNCTION dbo.GetLastName(@fullName VARCHAR(50))

RETURNS VARCHAR(50)

AS

BEGIN

DECLARE @lastName VARCHAR(50)

SET @lastName = SUBSTRING(@fullName, CHARINDEX(' ', @fullName) + 1, LEN(@fullName) - CHARINDEX(' ', @fullName))

RETURN @lastName

END;

--- display ---

declare @fullName varchar(25) = 'Muhammad Boota';

SELECT dbo.GetFirstName(@fullName) AS firstName, dbo.GetLastName(@fullName) AS lastName

***Select all Scalar function***

SELECT name

FROM sys.objects

WHERE type\_desc = 'SQL\_SCALAR\_FUNCTION'

AND schema\_id = SCHEMA\_ID('dbo');

1. Table-Valued function

* *Return table values in the form of*

*Table……*

* *Need Created table for this Type.*
* *Ex: fun that take 1st date, 2nd date and fatch the data from table*

***Example 1:***

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

CompanyName NVARCHAR(50),

ContactName NVARCHAR(50),

ContactTitle NVARCHAR(50),

Address NVARCHAR(50),

City NVARCHAR(50),

Region NVARCHAR(50),

PostalCode NVARCHAR(10),

Country CHAR(3)

);

INSERT INTO Customers (CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country)

VALUES

(1, 'Alfreds Futterkiste', 'Maria Anders', 'Sales Representative', 'Obere Str. 57', 'Berlin', NULL, '12209', 'Pak'),

(2, 'Ana Trujillo Emparedados y helados', 'Ana Trujillo', 'Owner', 'Avda. de la Constitución 2222', 'México D.F.', NULL, '05021', 'Pak'),

(3, 'Antonio Moreno Taquería', 'Antonio Moreno', 'Owner', 'Mataderos 2312', 'México D.F.', NULL, '05023', 'Ind'),

(4, 'Around the Horn', 'Thomas Hardy', 'Sales Representative', '120 Hanover Sq.', 'London', NULL, 'WA1 1DP', 'Ind'),

(5, 'Berglunds snabbköp', 'Christina Berglund', 'Order Administrator', 'Berguvsvägen 8', 'Luleå', NULL, 'S-958 22', 'Ban');

select \* from Customers

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Select Customer by Country \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CREATE FUNCTION dbo.GetCustomersByCountry (@countryCode CHAR(3))

RETURNS TABLE

AS

RETURN

SELECT CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country

FROM Customers

WHERE Country = @countryCode;

--\_\_\_\_ Execute function \_\_\_

SELECT \*

FROM dbo.GetCustomersByCountry('Pak');

***Example 2:***

CREATE FUNCTION dbo.ExtractNames (@fullName VARCHAR(50))

RETURNS TABLE

AS

RETURN (

SELECT

SUBSTRING(@fullName, 1, CHARINDEX(' ', @fullName) - 1) AS FirstName,

SUBSTRING(@fullName, CHARINDEX(' ', @fullName) + 1, LEN(@fullName) - CHARINDEX(' ', @fullName)) AS LastName

);

SELECT FirstName, LastName

FROM dbo.ExtractNames('Muhammad Boota');

***Example 3:***

CREATE TABLE orderdetail (

id INT PRIMARY KEY,

customerId INT,

totalprice INT,

quantity INT,

region VARCHAR(50),

city VARCHAR(50),

datetime DATETIME

);

INSERT INTO orderdetail (id, customerId, totalprice, quantity, region, city, datetime)

VALUES (1, 1, 100, 2, 'North', 'New York', '2022-01-01 10:00:00'),

(2, 2, 50, 1, 'South', 'Atlanta', '2022-01-02 12:30:00'),

(3, 3, 200, 4, 'West', 'Los Angeles', '2022-01-03 15:45:00');

INSERT INTO orderdetail (id, customerId, totalprice, quantity, region, city, datetime)

VALUES (4, 1, 75, 3, 'North', 'Boston', '2022-01-04 08:15:00'),

(5, 2, 150, 2, 'South', 'Miami', '2022-01-05 11:30:00'),

(6, 3, 50, 1, 'West', 'San Francisco', '2022-01-06 14:45:00'),

(7, 4, 100, 4, 'East', 'Washington DC', '2022-01-07 17:00:00'),

(8, 5, 25, 1, 'East', 'New York', '2022-01-08 09:15:00');

create function dbo.getCustomerOrderDetail(@custId int)

returns table

as

return (

select

id,

customerId,

totalprice,

quantity,

totalprice\*quantity as AmountofTotalQuantity,

region ,

avg(totalprice\*quantity) over() AvergAmount,

sum(totalprice\*quantity) over() totalAmount

from orderdetail

where customerId = @custId

)

select id,customerId,quantity,totalprice,AmountofTotalQuantity,region,AvergAmount,totalAmount from dbo.getCustomerOrderDetail(1)

1. Aggregated function

* user-defined aggregate functions are used with the GROUP BY clause and can return a single value for each group in a result set.
* Performing complex calculations that are not possible with the built-in aggregate functions.
* This Required .Net Application .dll haahahaha

Syntax:

CREATE AGGREGATE MyAggregateFunction(@inputParameterDataType)

RETURNS @outputTableVariable TABLE (

-- Define columns for output table variable

)

BEGIN

-- Define calculation logic for the aggregate function

END

Example 1,2,3:

CREATE AGGREGATE dbo.Median (@inputValue FLOAT)

RETURNS FLOAT

EXTERNAL NAME MyAssembly.[UserDefinedAggregateFunctions].Mode

CREATE AGGREGATE dbo.Concatenate (@inputValue NVARCHAR(MAX))

RETURNS NVARCHAR(MAX)

EXTERNAL NAME MyAssembly.[UserDefinedAggregateFunctions].Concatenate

This function takes a single input p

CREATE AGGREGATE dbo.Mode (@inputValue INT)

RETURNS INT

EXTERNAL NAME MyAssembly.[UserDefinedAggregateFunctions].Mode