Name: Sayali Bhosale
Date: Nov 27, 2024
Course: Foundations Of Databases & SQL Programming

Assignment 07- Functions

Introduction:

This module introduces SQL functions, essential tools for performing specific operations and calculations in your queries. You will learn about scalar functions that return a single value, and table-valued functions that return a set of rows. The module covers built-in functions for string manipulation, date formatting, mathematical calculations, and how to create user-defined functions for customized operations. Mastering these functions will help you simplify complex queries and improve code reusability in your database.

Learning:

Explain when you would use a SQL UDF.

A **User-Defined Function (UDF)** in SQL is used when you want to create reusable logic to simplify your queries or perform specific operations that aren't directly available through built-in SQL functions. Here's when you would use an SQL UDF:

1. **Reusability**: When you need to perform the same calculation or logic in multiple queries, a UDF lets you define it once and reuse it, saving time and effort.
2. **Custom Logic**: When built-in SQL functions don't meet your requirements, you can create a UDF to handle custom calculations, transformations, or operations specific to your database.
3. **Data Validation**: Use a UDF to implement validation rules (e.g., checking if a value is within a range) that can be consistently applied across queries.
4. **Simplify Complex Queries**: If a query requires repeated or complex expressions, a UDF can encapsulate that logic, making the main query easier to read and maintain.
5. **Return Scalar or Tabular Data**: Scalar UDFs return a single value (e.g., a discounted price). Table-valued UDFs return a table, allowing you to integrate the result into other queries like a virtual table.

Explain the differences between Scalar, Inline, and Multi-Statement Functions.

In SQL, functions are powerful tools that help us reuse logic and streamline queries. There are three main types of functions: Scalar Functions, Inline Table-Valued Functions (ITVF), and Multi-Statement Table-Valued Functions (MSTVF).

Scalar Functions

A scalar function returns a single value, like a number or string, and is useful for calculations or transformations. For example, it can calculate a product's discounted price by taking the product price and discount rate as inputs. However, scalar functions are limited because they can only handle one value at a time.

Inline Table-Valued Functions (ITVF)

An Inline Table-Valued Function (ITVF) returns an entire table from a single query, allowing for dynamic filtering and manipulation of rows. The function **'fProductInventoriesWithPreviousMonthCountsWithKPIs'** is an example of an ITVF. It takes a KPI value as a parameter (1, 0, or -1) and returns a table with product names, inventory dates, and counts filtered by that KPI.

```
CREATE FUNCTION fProductInventoriesWithPreviousMonthCountsWithKPIs(@KPIValue INT)
RETURNS TABLE
AS
RETURN SELECT
    ProductName,
    InventoryDate,
    InventoryCount,
    PreviousMonthCount,
    CountVsPreviousCountKPI
FROM vProductInventoriesWithPreviousMonthCountsWithKPIs
WHERE CountVsPreviousCountKPI = @KPIValue;
```

This function is highly efficient because it uses a single query to fetch the data dynamically.

Multi-Statement Table-Valued Functions (MSTVF)

A Multi-Statement Table-Valued Function (MSTVF) allows for multiple steps or intermediate calculations before returning a table. For instance, it can be used to calculate inventory levels below the reorder point and group data by product. Unlike Inline Table-Valued Functions (ITVF), MSTVFs use table variables to store intermediate results, making them versatile but slightly slower. Here's an example:

```
CREATE FUNCTION GetProductsBelowReorderLevel()
RETURNS @Result TABLE
  (ProductName NVARCHAR(100),
   ReorderLevel INT,
   InventoryCount INT)
AS
BEGIN
   INSERT INTO @Result
   SELECT
      P.ProductName,
      I.ReorderLevel,
      SUM(I.Count) AS InventoryCount
   FROM vProducts AS P
   JOIN vInventories AS I ON P.ProductID = I.ProductID
   WHERE I.Count < I.ReorderLevel
   GROUP BY P.ProductName, I.ReorderLevel;
   RETURN;
END;
```

Key Differences

- **Scalar Functions** return a single value and are simple to use for transformations or calculations.
- **Inline Table-Valued Functions** return a table with one query and are highly efficient for filtering and joining operations.
- **Multi-Statement Table-Valued Functions** allow complex logic and intermediate steps but are slower due to additional processing.

Each function type has a specific purpose, and its application depends on the task's requirements. By using functions effectively, we can simplify SQL queries, making them reusable and maintainable.

Summary:

SQL functions— Scalar, Inline Table-Valued (ITVF), and Multi-Statement Table-Valued Functions (MSTVF)—are vital for simplifying and reusing logic in database operations.

Scalar functions return single values for calculations, while Inline Table-Valued Functions offer efficient query results for filtering and joining data. Multi-Statement Table-Valued Functions manage complex operations with intermediate steps.

By understanding their differences and appropriate use cases, we can optimize query performance and address various database challenges effectively.