SQL Functions in SQL Server

SQL Server functions are powerful tools that allow you to perform various operations on data. There are several types of SQL functions, each serving different purposes. Here's a detailed overview of SQL functions in SQL Server:

Types of SQL Functions

1. Scalar Functions

Scalar functions operate on a single value and return a single value. Examples include mathematical functions, string functions, and date functions.

Examples:

- Mathematical Functions: `ABS()`, `CEILING()`, `FLOOR()`, `POWER()`, `ROUND()`

```sql
SELECT ABS(-5); -- Returns 5
SELECT ROUND(123.456, 2); -- Returns 123.46
```

- String Functions: `LEN()`, `LOWER()`, `UPPER()`, `SUBSTRING()`, `REPLACE()`

```sql
SELECT LEN('Hello'); -- Returns 5
SELECT SUBSTRING('Hello World', 1, 5); -- Returns 'Hello'
```

- Date Functions: `GETDATE()`, `DATEADD()`, `DATEDIFF()`, `FORMAT()`

```sql
SELECT GETDATE(); -- Returns the current date and time
SELECT DATEADD(day, 5, GETDATE()); -- Returns the date 5 days from now
```

2. Aggregate Functions

Aggregate functions perform a calculation on a set of values and return a single value. They are often used with `GROUP BY` clauses.

Examples:

- Common Aggregate Functions: `AVG()`, `COUNT()`, `SUM()`, `MIN()`, `MAX()`

```sql
SELECT AVG(Salary) FROM Employees; -- Returns the average salary
SELECT COUNT(*) FROM Orders; -- Returns the number of orders
```

3. Table-Valued Functions (TVFs)

TVFs return a table. They can be invoked in the `FROM` clause of a query. There are two types: inline TVFs and multi-statement TVFs.

Examples:

- Inline TVFs: Defined with a single `RETURN` statement.

```sql
CREATE FUNCTION GetEmployeesByDepartment(@DeptID INT)
RETURNS TABLE
AS
RETURN (
  SELECT EmployeeID, EmployeeName
  FROM Employees
  WHERE DepartmentID = @DeptID
);

SELECT * FROM GetEmployeesByDepartment(1);
```

- Multi-Statement TVFs: Defined with a `BEGIN...END` block and can contain multiple statements.

```sql
CREATE FUNCTION GetRecentOrders(@Days INT)
RETURNS @RecentOrders TABLE (OrderID INT, OrderDate DATETIME)
AS
BEGIN
  INSERT INTO @RecentOrders
  SELECT OrderID, OrderDate
  FROM Orders
  WHERE OrderDate > DATEADD(day, -@Days, GETDATE());

  RETURN;
END;

SELECT * FROM GetRecentOrders(30);
```

4. System Functions

System functions provide information about the SQL Server instance and its configuration. Examples include metadata functions and system information functions.

Examples:

- Metadata Functions: `OBJECT_NAME()`, `COL_LENGTH()`, `DB_NAME()`

```sql
SELECT OBJECT_NAME(OBJECT_ID('Employees')); -- Returns 'Employees'
SELECT COL_LENGTH('Employees', 'EmployeeName'); -- Returns the length of the 'EmployeeName' column
```

- System Information Functions: `@@VERSION`, `@@TRANCOUNT`

```sql
SELECT @@VERSION; -- Returns the SQL Server version
SELECT @@TRANCOUNT; -- Returns the number of active transactions
```

5. User-Defined Functions (UDFs)

UDFs are custom functions created by users to encapsulate reusable logic. They can be scalar or table-valued.

Examples:

- Scalar UDF: Returns a single value.
```sql
CREATE FUNCTION CalculateDiscount(@Price DECIMAL(10, 2), @DiscountRate
DECIMAL(5, 2))
RETURNS DECIMAL(10, 2)
AS
BEGIN
  RETURN @Price * (1 - @DiscountRate);
END;

SELECT dbo.CalculateDiscount(100, 0.1); -- Returns 90
```
- Table-Valued UDF: Returns a table.
```sql
CREATE FUNCTION GetOrdersByCustomer(@CustomerID INT)
RETURNS TABLE
AS
RETURN (
  SELECT OrderID, OrderDate
  FROM Orders
  WHERE CustomerID = @CustomerID
);

SELECT * FROM dbo.GetOrdersByCustomer(1);
```

Using Functions in SQL Server

Functions can be used in various parts of a SQL query, including the `SELECT` clause, `WHERE` clause, `FROM` clause (for table-valued functions), and `ORDER BY` clause. They provide a powerful way to encapsulate logic, perform calculations, and manipulate data.

Example Query Using Different Types of Functions

```sql
SELECT
```

```
  e.EmployeeID,
  e.EmployeeName,
  dbo.CalculateDiscount(e.Salary, 0.1) AS DiscountedSalary, -- Scalar UDF
  COUNT(o.OrderID) AS OrderCount -- Aggregate Function
FROM
  Employees e
LEFT JOIN
  Orders o ON e.EmployeeID = o.EmployeeID
WHERE
  e.DepartmentID = 1 -- Scalar Function used in WHERE clause
GROUP BY
  e.EmployeeID, e.EmployeeName
ORDER BY
  dbo.CalculateDiscount(e.Salary, 0.1) DESC; -- Scalar UDF used in ORDER BY clause
```