Chapter-3 DQL - SQL Functions with MS SQL Server

SQL functions are essential tools that help manipulate and transform data within a query. MS SQL Server provides a wide range of built-in SQL functions that can be used in Data Query Language (DQL) operations, specifically within the `SELECT` statement, to calculate values, modify data formats, and perform complex queries more efficiently. These functions can be categorized into several types, including aggregate functions, scalar functions, and window functions.

---

1. Aggregate Functions

Aggregate functions perform calculations on multiple rows of data to return a single result. They are typically used with the `GROUP BY` clause to group records and calculate summaries.

- **COUNT()**: Returns the number of rows in a group.

```
SELECT Department, COUNT(*) AS EmployeeCount
FROM Employees
GROUP BY Department;
```

- **SUM()**: Calculates the total sum of a numeric column.

```
SELECT Department, SUM(Salary) AS TotalSalary
FROM Employees
GROUP BY Department;
```

- **AVG()**: Returns the average value of a numeric column.

```
SELECT Department, AVG(Salary) AS AverageSalary
FROM Employees
GROUP BY Department;
```

- **MAX()**: Returns the maximum value in a column.

```
SELECT Department, MAX(Salary) AS HighestSalary
FROM Employees
GROUP BY Department;
```

- **MIN()**: Returns the minimum value in a column.

```
SELECT Department, MIN(Salary) AS LowestSalary
FROM Employees
GROUP BY Department;
```

- **GROUP_CONCAT()** (Not available directly in MS SQL Server, but can be simulated using `STRING_AGG()` in newer versions): Concatenates the values of a column in a group.

```
SELECT Department, STRING_AGG(EmployeeName, ', ') AS Employees
FROM Employees
GROUP BY Department;
```

---

2. Scalar Functions

Scalar functions perform operations on a single value and return a single value. They can be used in the `SELECT` statement or anywhere expressions are allowed.

- **LEN()**: Returns the length of a string (number of characters).

```
SELECT EmployeeName, LEN(EmployeeName) AS NameLength
FROM Employees;
```

- **UPPER()**: Converts a string to uppercase.

```
SELECT EmployeeName, UPPER(EmployeeName) AS UppercaseName
FROM Employees;
```

- **LOWER()**: Converts a string to lowercase.

```
SELECT EmployeeName, LOWER(EmployeeName) AS LowercaseName
FROM Employees;
```

- **CONCAT()**: Concatenates two or more strings into one string.

```
SELECT CONCAT(FirstName, ' ', LastName) AS FullName
FROM Employees;
```

- **TRIM()**: Removes leading and trailing spaces from a string.

```
SELECT TRIM(EmployeeName) AS TrimmedName
FROM Employees;
```

- **SUBSTRING()**: Extracts a substring from a string.

```
SELECT SUBSTRING(EmployeeName, 1, 5) AS NameSubstring
FROM Employees;
```

- **ROUND()**: Rounds a numeric value to a specified number of decimal places.

```
SELECT ROUND(Salary, 2) AS RoundedSalary
FROM Employees;
```

- **GETDATE()**: Returns the current system date and time.

```
SELECT GETDATE() AS CurrentDateTime;
```

- **DATEADD()**: Adds a specified time interval to a date.

```
SELECT DATEADD(MONTH, 1, GETDATE()) AS NextMonth
FROM Employees;
```

- **DATEDIFF()**: Returns the difference between two dates in terms of a specified date part (such as days, months, years).

```
SELECT DATEDIFF(DAY, HireDate, GETDATE()) AS DaysSinceHired
FROM Employees;
```

---

3. String Functions

These functions allow you to manipulate string data.

- **CHARINDEX()**: Returns the position of the first occurrence of a substring within a string.

```
SELECT CHARINDEX('a', 'Database') AS Position;
```

- **REPLACE()**: Replaces all occurrences of a specified substring within a string with a new substring.

```
SELECT REPLACE(EmployeeName, 'John', 'Jon') AS UpdatedName
FROM Employees;
```

- **LEFT()**: Extracts a specified number of characters from the left side of a string.

```
SELECT LEFT(EmployeeName, 3) AS LeftCharacters
FROM Employees;
```

- **RIGHT()**: Extracts a specified number of characters from the right side of a string.

```
SELECT RIGHT(EmployeeName, 3) AS RightCharacters
FROM Employees;
```

4. Mathematical Functions

Mathematical functions help perform calculations on numeric values.

- **ABS()**: Returns the absolute value of a number.

```
SELECT ABS(Salary) AS AbsoluteSalary
FROM Employees;
```

- **CEILING()**: Returns the smallest integer greater than or equal to the given number.

```
SELECT CEILING(Salary) AS RoundedUpSalary
FROM Employees;
```

- **FLOOR()**: Returns the largest integer less than or equal to the given number.

```
SELECT FLOOR(Salary) AS RoundedDownSalary
FROM Employees;
```

- **POWER()**: Returns the result of a number raised to a specified power.

```
SELECT POWER(2, 3) AS Result;
```

- **SQRT()**: Returns the square root of a number.

```
SELECT SQRT(Salary) AS SalarySquareRoot
FROM Employees;
```

---

5. Conversion Functions

Conversion functions allow for changing data types.

- **CAST()**: Converts an expression from one data type to another.

```
SELECT CAST(Salary AS VARCHAR(50)) AS SalaryAsString
FROM Employees;
```

- **CONVERT()**: Similar to `CAST()`, but with more options for formatting date and time types.

```
SELECT CONVERT(VARCHAR(10), HireDate, 101) AS HireDateFormatted
FROM Employees;
```

---

6. Window Functions

Window functions perform calculations across a set of table rows related to the current row. They are often used to calculate running totals, ranks, and other cumulative metrics.

- **ROW_NUMBER()**: Assigns a unique number to each row, starting from 1.

```
SELECT EmployeeName, Salary,
    ROW_NUMBER() OVER (ORDER BY Salary DESC) AS RowNum
FROM Employees;
```

- **RANK()**: Similar to ROW_NUMBER(), but it assigns the same rank to rows with equal values, leaving gaps in the rank.

```
SELECT EmployeeName, Salary,
    RANK() OVER (ORDER BY Salary DESC) AS Rank
FROM Employees;
```

- **DENSE_RANK()**: Similar to RANK(), but it does not leave gaps in the ranking.

```
SELECT EmployeeName, Salary,
    DENSE_RANK() OVER (ORDER BY Salary DESC) AS DenseRank
FROM Employees;
```

- **NTILE()**: Divides the result set into a specified number of buckets (tiles) and assigns a bucket number to each row.

```
SELECT EmployeeName, Salary,
    NTILE(4) OVER (ORDER BY Salary DESC) AS SalaryQuartile
FROM Employees;
```