Bella Amarbat
Nov 26, 2025
IT FDN 130 A
Assignment 07

Using SQL Functions for Data Analysis and Reporting

Intro of SQL Functions

SQL includes many built-in functions that help users analyze, format, and transform data. These functions make it possible to summarize information, create reports, enforce business rules, and support more advanced logic in the database. This paper talks about several categories of SQL functions: aggregate, string, date/time, conditional, window, and user-defined functions (UDF).

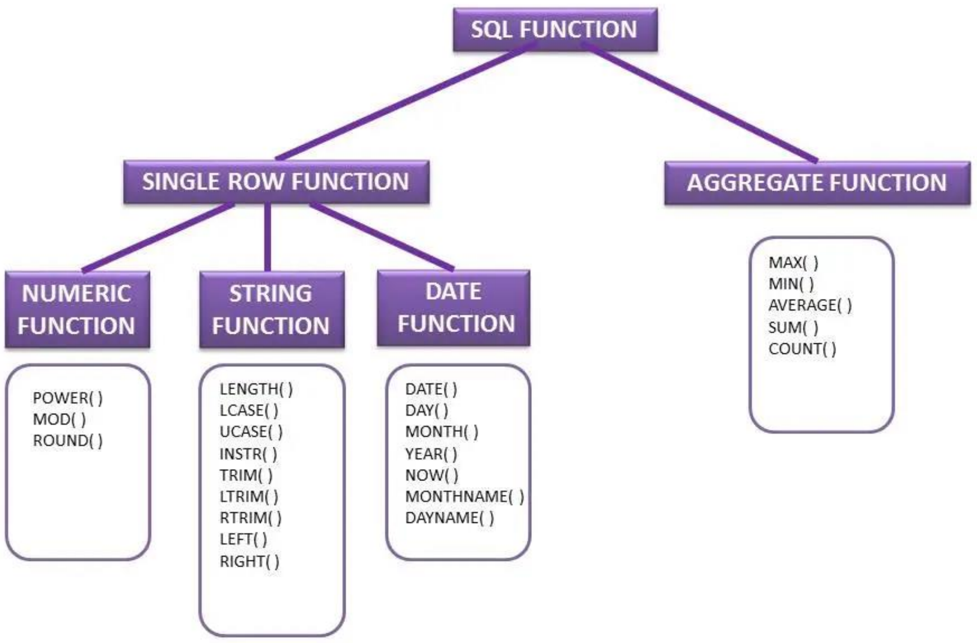
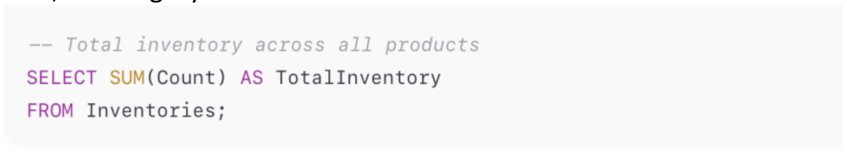


Figure 1. SQL functions type

Aggregate and Grouping Functions

Aggregate functions such as MAX, MIN, SUM, AVG, and COUNT allow SQL to summarize groups of rows. They are commonly combined with GROUP BY to calculate subtotals and with HAVING to filter groups based on aggregate results. These tools are essential for reporting tasks like finding monthly totals, average performance, or category summaries.



```
-- Total inventory across all products
SELECT SUM(Count) AS TotalInventory
FROM Inventories;
```

Figure 2. Example of aggregate function - SUM

String, Date/Time, and Conditional Functions

SQL also provides functions that help clean and format data:

- **String functions** (LEFT, RIGHT, CONCAT, REPLACE) are useful for preparing labels and standardizing text.
- **Date/time functions** (DATEADD, DATEDIFF, YEAR, MONTH) support time-based calculations and reporting.
- **Conditional functions** (CASE, IIF) allow queries to apply business logic and produce different results based on rules.

```sql
SELECT DATEDIFF(DAY, '2025-01-01', '2025-01-31');   -- 30 days
```

Figure 3. Example of Date function that shows difference between 2 dates

Advanced Reporting Functions

More complex analysis is enabled by window functions and ranking functions, which calculate running totals, percentages, or rank rows within a partition, without losing detail. These functions are commonly used in dashboards, KPIs, and analytical queries.

User- Defined Functions (UDF)

SQL Server allows developers to create custom functions to standardize logic and calculations. UDFs are especially helpful for enforcing business rules, performing repeated calculations, or returning reusable datasets. They support cleaner code and maintain consistency across the database.

When to use UDF functions

UDF is used when a database needs reusable logic that consistently returns a value or a table. UDFs are helpful for repeated calculations, formatting tasks, or business rules that appear in multiple queries. UDFs are also valuable in reporting because they simplify formulas and centralize logic into one location. Additionally, scalar UDFs can be used in CHECK constraints, allowing more advanced validation rules. In all cases, UDFs support cleaner, more maintainable SQL code. For example, the below query can be used for month-end reporting, closing schedule, and variance analysis by fiscal periods:

```
CREATE FUNCTION dbo.GetFiscalPeriod(@Date DATE)
RETURNS VARCHAR(7)
AS
BEGIN
    RETURN (
        SELECT FiscalYear + '-' + RIGHT('0' + CAST(FiscalPeriod AS VARCHAR(2)), 2)
        FROM FiscalCalendar
        WHERE CalendarDate = @Date
    );
END
```

Figure 4. Example of UDF function

Differences Between Scalar, Inline, and Multi-Statement Functions

These functions differ mainly in what they return and how SQL Server processes them. Please refer to the comparison table below.

Feature	Scalar Function	Inline Table-Valued Function (iTVF)	Multi-Statement Table-Valued Function (mTVF)
Return Type	Single value (number, string, date, etc.)	Table	Table
Structure	Can include multiple statements, but returns one value	Single SELECT statement	Multiple statements, including variables and logic
Purpose / Use Case	Calculations, formatting, conditional logic	Return filtered or computed datasets, parameterized views	Complex operations requiring multiple steps before returning a table
Parameters	Can accept one or more	Can accept one or more	Can accept one or more
Performance	May be slower on large datasets (row-by-row execution)	High performance, SQL Server can optimize efficiently	Slower than iTVF due to table variable handling and lack of statistics
Example	Compute tax, concatenate first and last name	Return all orders over $100, or transformed columns	Build temporary results, complex conditional datasets

Summary

SQL's wide variety of built-in and user-defined functions enables users to analyze data, produce summaries, cleanse values, and create reusable calculations. Aggregate, string, date/time, conditional, ranking, and window functions provide the foundation for complex reporting, while User-Defined Functions offer a structured way to centralize repeated business logic. Understanding when to use a UDF, and how Scalar, Inline, and Multi-Statement functions differ helps ensure that SQL code remains efficient, readable, and maintainable.