7. Expressions, Aliases, and Computed Columns (MS SQL)

This chapter is your tour through the everyday arithmetic of T-SQL — expressions, naming things sensibly with aliases, and the delightful trick of computed columns. These tools shape how your data behaves without being noisy about it.

Expressions — little formulas hiding in plain sight

An expression is any snippet of code that produces a value. SQL expressions can involve arithmetic, string manipulation, date arithmetic, logical comparisons, and even nested subqueries.

Simple arithmetic

```
SELECT 10 + 5 AS SumResult,
       10 * 5 AS ProductResult;
```

String concatenation (SQL Server uses `+`)

```
SELECT FirstName + ' ' + LastName AS FullName
FROM dbo.Customers;
```

Be conscious of `NULL` here — `'abc' + NULL` becomes `NULL`. You often want `ISNULL(LastName, '')` to protect the string.

Date expressions

```
SELECT OrderDate,
       DATEADD(day, 7, OrderDate) AS OneWeekLater,
       DATEDIFF(day, OrderDate, GETDATE()) AS DaysSinceOrder
FROM dbo.Orders;
```

These functions behave deterministically, which helps the optimizer.

Aliases — giving names to ideas so they behave

Aliases label expressions or tables so you don't drown in verbose code.

Column alias

```sql
SELECT TotalAmount * 1.18 AS TotalWithGST
FROM dbo.Sales;
```

Column aliases can also be referenced in `ORDER BY`, because that clause executes after `SELECT` logically.

Table alias

```sql
SELECT c.CustomerID, o.OrderID
FROM dbo.Customers AS c
JOIN dbo.Orders AS o ON o.CustomerID = c.CustomerID;
```

Short table aliases reduce eye strain and help avoid naming collisions.

Computed columns — logic living inside the table

A computed column is a virtual column defined by an expression. SQL Server calculates it on the fly unless it's marked as `PERSISTED`, which means the result is stored physically.

Virtual computed column

```sql
ALTER TABLE dbo.Sales
ADD Total AS (Quantity * UnitPrice);
```

This column behaves like part of the table but isn't stored on disk. SQL Server recalculates it each time it's referenced.

Persisted computed column

```sql
ALTER TABLE dbo.Sales
ADD TotalPersisted AS (Quantity * UnitPrice) PERSISTED;
```

Persisted columns are physically stored and can be indexed. They behave like any other column during query compilation, which can make filtered searches faster.

When to use computed columns

- When the value can always be derived from existing fields.
- When you want to avoid storing redundant data.
- When indexing a calculation makes the difference between a scan and a seek.

Example — index on a computed column

```sql
ALTER TABLE dbo.Payments
ADD PaymentYear AS (YEAR(PaymentDate)) PERSISTED;

CREATE INDEX IX_Payments_PaymentYear
  ON dbo.Payments (PaymentYear);
```

Now filters like `WHERE PaymentYear = 2024` become SARGable.

Small traps to keep an eye on

`CONVERT` and `CAST` can change precision in ways that surprise you.

```sql
SELECT CAST(10/4 AS DECIMAL(10,2));  -- integer division: result 2.00
SELECT 10.0/4;                       -- decimal division: result 2.5
```

SQL defaults to integer math when both operands are integers.

String concatenation across mixed data types will force implicit conversions. Be explicit when types matter.

```sql
SELECT 'Amount: ' + CONVERT(varchar(20), Amount)
FROM dbo.Invoices;
```

Expressions, aliases, and computed columns together give you a small but powerful vocabulary for shaping data behaviour. They are the quiet infrastructure of T-SQL queries, invisible until you misuse them. The next step is the world of filtering and SARGability, where your predicates decide whether indexes cheer or cry.