03

Using Built-in Functions & Expressions

Use built-in functions to cleaning data

Data Types

Group	Data type	Description
String	char(n)	Fixed width character string
	varchar(n)	Variable width character string
	nvarchar(n)	Variable width Unicode string
	text	Variable width character string
	image	Variable width binary string
	binary	Fixed width binary string
Number	smallint	Allows whole numbers between -215 and 215
	int	Allows whole numbers between -231 and 231
	bigint	Return records that do not satisfy any of the conditions in WHERE clause.
	decimal	Fixed precision and scale numbers. Allows numbers from -1038 +1 to 1038 -1.
	float	Floating precision number data from -1.79E + 308 to 1.79E + 308.
Datetime	datetime	From 1/1/1753 to 31/12/9999 with an accuracy of 3.33 milliseconds
	date	Store a date only. From 1/1/0001 to 31/12/9999

Expressions Using Operators

In this section, you will learn how to use operators to concatenate strings and perform mathematical calculations in T-SQL queries.

1. Concatenating Strings

```
--1
SELECT 'ab' + 'c' ;
--2
SELECT CustomerID, FirstName + ' ' + LastName AS full_name
FROM SalesLT.Customer;
--3
SELECT CustomerID, FirstName + ', ' + MiddleName AS full_name
FROM SalesLT.Customer;
```

Deal With NULL Values

NULL represents a missing or unknown value.

- The result of any expression containing a NULL value is NULL

  10 + NULL = NULL

  'String ' + NULL = NULL

- Use **IS NULL** or **IS NOT NULL** to compare with Null value

Handling NULL via these functions

- **ISNULL** (column/variable, value): Returns value if the column or variable is NULL
- **NULLIF** (column/variable, value): Returns NULL if the column or variable is value
- **COALESCE** (column/variable1, column/variable2, ...): Returns the value of the first non-NULL column or variable in the list

Built-In Functions

Function Category	Description
Scalar	Operate on a single row, return a single value
Logical	Compare multiple values to determine a single output
Ranking	Operate on a partition (set) of rows
Aggregate	Take one or more input values, return a single summarizing value

Scalar Functions

Categories

- **Conversion**
- Date and Time
- Mathematical
- String

CAST: convert an expression of one data type to another.

```
-- CAST( expression AS data_type [length] )
SELECT CAST( '2022' as INT )
```

CONVERT: converts a value (of any type) to another

```
-- CONVERT ( data_type[length], expression)
SELECT CONVERT( INT, '2022')
SELECT CONVERT( VARCHAR, GETDATE() ,21)
```

STR: return a number as a string

```
SELECT STR ( 2022 )
```

Scalar Functions

Categories

- Conversion
- **Date and Time**
- Mathematical
- String

DAY: returns the day of the month for a given date

```
-- DAY( date ) --
SELECT DAY( '2022/02/23' ) AS get_day
```

MONTH: returns the month part for a given date (a number from 1 to 12)

```
-- MONTH( date ) --
SELECT MONTH( '2022/02/23' ) AS get_day
```

YEAR: returns the year part for a given date

```
-- YEAR( date) --
SELECT YEAR( '2022/02/23' ) AS get_day
```

DATEPART: returns a specified part of a date (as integer)

```
-- DATEPART(interval, date) --
SELECT DATEPART(year, '2022/02/23') AS date_part_int
```

Scalar Functions

Categories

- Conversion
- **Date and Time**
- Mathematical
- String

CURRENT_TIMESTAMP() returns the current date and time

```
SELECT CURRENT_TIMESTAMP
```

DATEADD() add a number of time units to a date

```
-- DATEADD (interval, number, date)
SELECT DATEADD (year, 1, '2022-02-23') as date_add
```

DATEDIFF() allows you to find the difference between two dates

```
-- DATEDIFF (interval, start_date, end_date)
SELECT DATEDIFF (year, '2021-02-23', '2022-02-23') as diff
```

FORMAT() simplify the conversion of date/time values as string values

```
-- FORMAT(value, format [, culture ] )
```

Scalar Functions

Categories

- Conversion
- Date and Time
- **Mathematical**
- String

You can use several operators to perform simple mathematical operations on numeric values

- ```
  SELECT
      1 + 1 AS addition
      , 10.0 / 3 AS division
      , 10 / 3 AS [Integer Division]
      , 10 % 3 AS modulo;
  ```
- ```
  SELECT
      OrderQty * UnitPrice * (1.0 - UnitPriceDiscount) AS Calculated,
      LineTotal
  FROM SalesLT.SalesOrderDetail;
  ```

Scalar Functions

Categories

- Conversion
- Date and Time
- **Mathematical**
- String

ABS returns the absolute value of the number—the distance between the number and zero

```
SELECT ABS(2) AS "2", ABS(-2) AS "-2"
```

POWER returns the power of one number to another number

```
SELECT POWER(10,1) AS "Ten to the First",
POWER(10,2) AS "Ten to the Second",
POWER(10,3) AS "Ten to the Third";
```

SQUARE returns the square of a number, or the number multiplied to itself

SQRT returns the opposite, the square root of a number

```
SELECT SQUARE(10) AS "Square of 10" ,
SQRT(10) AS "Square Root of 10" ,
SQRT(SQUARE(10)) AS "The Square Root of the Square of 10" ;
```

Scalar Functions

Categories

- Conversion
- Date and Time
- Mathematical
- **String**

RTRIM and LTRIM remove spaces from the right side or left side

```
SELECT  '*' + RTRIM (Col1) + '*' AS 'RTRIM'
        , '*' + LTRIM (Col1) + '*' AS 'LTRIM'
```

LEFT and RIGHT return a specified number of characters on the left or right side of a string

```
LEFT (<string>, <number of characters)
RIGHT (<string>, <number of characters)
```

LEN return the number of characters in a string

```
LEN (<string>)
```

CHARINDEX() find the numeric starting position of a search string inside another string

```
CHARINDEX (<search string>,<target string>[,<start location>])
SELECT  LastName, CHARINDEX ('e', LastName) AS find_e
        , CHARINDEX ('be', LastName) AS find_be
```

Scalar Functions

Categories

- Conversion
- Date and Time
- Mathematical
- **String**

SUBSTRING return a portion of a string starting at a given position and for a specified number of characters

```
-- SUBSTRING (<string>, <start location>, <length>)
SELECT LastName, SUBSTRING ( LastName, 1, 4 ) AS first_4
```

REVERSE returns a string in reverse order.

```
SELECT REVERSE ( '!dlroW, olleH' )
```

REPLACE substitute one string value inside another string value

```
REPLACE (<string value>, <string to replace>, <replacement>)
```

Logical Functions and Expressions

The CASE Expression evaluate a list of expressions and return the first one that evaluates to true

```
CASE
      WHEN <test expression1>  THEN <value1>
      WHEN <test expression2> THEN <value2>
      ELSE  <value3>
      END
```

IIF return a result based on whether a Boolean expression is true or false

```
-- IIF ( boolean_expression, true_value, false_value )
SELECT IIF ( 50 > 20, 'TRUE', 'FALSE' ) AS RESULT;
```

Time for practicing

Exercise 5: Retrieve shipping status

You have been asked to create a query that returns a list of sales order IDs and order dates with a column named **ShippingStatus** that contains the text **Shipped** for orders with a known ship date, and **Awaiting Shipment** for orders with no ship date.