

# Shaping Data in a Query

---



**Tamara Pattinson**

Consultant | Software Engineer | Instructor

@PattinsonTamara [www.tamarapattinson.com](http://www.tamarapattinson.com)



# Shaping Data in a Query

## Overview



- **SELECT Query Syntax**
- **Concatenation**
- **Cast and Convert**
  - Try\_Cast( ) and Try\_Convert( )
- **Formatting**
  - Strings
  - Characters
  - Introduction to User Defined Functions
- **Dates and Numeric Values**
  - Calculating and formatting
  - Dates
  - Numeric values
  - CASE Expressions
- **NULL Values**
- **Data integrity and reporting accuracy!**



# Syntax

**The rules that state how, and in which order, words and symbols are used in a computer language**



`SELECT` expression (WHAT?)

`FROM` tables (WHERE?)

`GROUP BY` fields (SUM, AVG, MIN, MAX) (**aggregation**)

`HAVING` or `WHERE` or `MATCH` conditions (FILTERED?) (**optional**)

`ORDER BY` Ascending or Descending (ORGANIZED HOW?) (**optional**)

## Syntax of the `SELECT` statement

The “grammar” of writing the select query

“Query” is the question

“Result set” is the answer

# Summary



- SELECT \* FROM \* WHERE \* ORDER BY
- **Syntax**
  - Query = question
  - Result set = answer
- **Formatting**
- **Database Namespaces**



# Concatenation

**The action of linking things together in a series**



```
/* the 3 T-SQL concatenation methods */
```

```
--string concatenation
```

```
SELECT
```

◀ **String** concatenation (prior to SQL Server 2012)

```
    FirstName + ' ' + LastName
```

```
--CONCAT method
```

```
SELECT
```

◀ **CONCAT**

```
    CONCAT(FirstName, ' ', LastName)
```

```
--CONCAT With Separator
```

```
SELECT
```

◀ **CONCAT\_WS** (concat with separator)

```
    CONCAT_WS(' ', FirstName, LastName)
```

# Summary



- **Concatenation means linking**
- **Three methods available**
  - + (string concatenation prior to SQL Server 2012)
  - CONCAT()
  - CONCAT\_WS()
- **Alias names**





# Casting and Converting Data Types

---



--CAST( ) and TRY\_CAST( ) syntax

CAST(datatype AS data\_type[(length)])

TRY\_CAST(datatype AS data\_type[(length)])

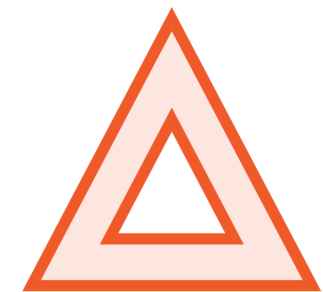
--CONVERT( ) and TRY\_CONVERT( ) syntax

CONVERT(data\_type[length], expression[, style])

TRY\_CONVERT(data\_type[length], expression[, style])

◀ **CAST ( ) and TRY\_CAST( ) syntax**

◀ **CONVERT( ) and TRY\_CONVERT( )**

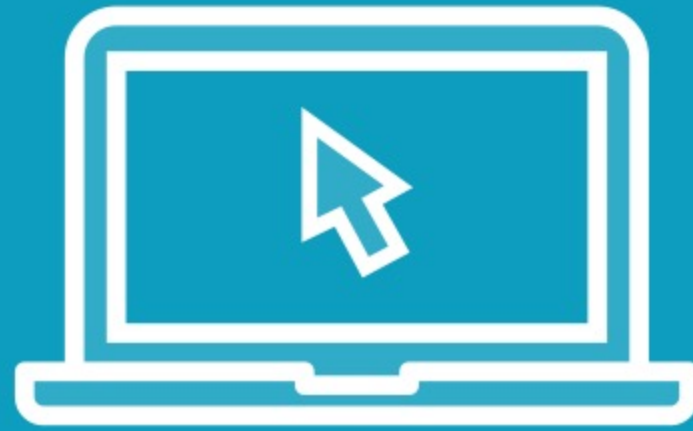


**LIMITED** error handling

Why `CAST()` and `CONVERT()`?



# Demo



- Differences between CAST ( ) and CONVERT( )
- CAST( ) & CONVERT( ) Syntax
- **Alter data types**
  - Which data types to use
  - Alter and format with CAST( )
  - Alter and format with CONVERT( )
- **Error handling**
  - TRY\_CAST( )
  - TRY\_CONVERT( )
- **Joining Tables**



# Differences between `CAST()` and `CONVERT()`

## `CAST()`

**ANSI Standard**

**Supported by all RDMS**

## `CONVERT()`

**Supported by:**

Microsoft SQL Server

MySQL

Oracle

**NOT supported by:**

PostgreSQL

SQLite

**Uses and optional parameter for styling**



# Cast and Convert

## Summary



### Similarities and Differences

- **When to use** `CAST()`
- **When to use** `CONVERT()`

### Error Handling

- `TRY_CAST()`
- `TRY_CONVERT()`

**Conversion rules & explicitly not allowed**

**More on formatting coming soon!**



# Demo



- **Format with proper casing LastName, FirstName**
  - TRIM
  - SUBSTRING
  - LEN
  - UPPER and LOWER
- **Apply concatenation**
- **Apply alias names**



## Step 1:

Remove trailing and leading spaces using **TRIM**





## Step 2:

Isolate the characters for upper and lower casing using  
**SUBSTRING**



Step 3:

Apply UPPER and LOWER



## Step 4:

Apply concatenation to return Lastname, Firstname using

**CONCAT**



# String:

A group of letters, numbers, or symbols

# Character:

A single letter, number, or symbol



```
SUBSTRING ( expression, start, length )
```

```
/*
```

```
expression can be character, binary, text, ntext, or image
```

```
start: 1 is the first character
```

```
length: positive integer specifies how many characters of the expression will be  
        returned.
```

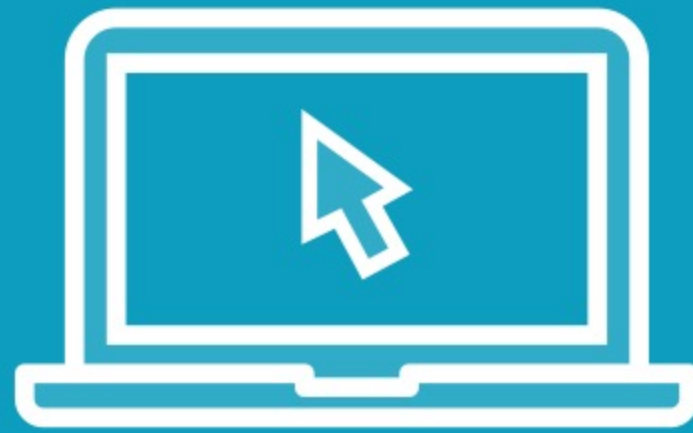
```
*/
```

## SUBSTRING

Returns part of a character, binary, text, or image expression in SQL Server

Return type is either **varchar**, **nvarchar**, or **varbinary** depending on the expression type

Demo



## Simplifying with User Defined Functions

### String manipulation on a character level

- TRIM
- REPLACE
- CHARINDEX
- REGEX
- PATINDEX

### PRINT command

### Introduction to WHILE & IF/ELSE



```
// TRIM methods
```

```
TRIM(expression) '  hello  ' = 'hello'
```

◀ Removes leading and trailing spaces

```
LTRIM(expression) '  hello  ' = 'hello '
```

◀ Removes leading spaces

```
RTRIM(expression) '  hello  ' = ' hello'
```

◀ Removes trailing spaces

```
TRIM(characters FROM expression)  
'*_ hello' = 'hello'
```

◀ Optional TRIM Function

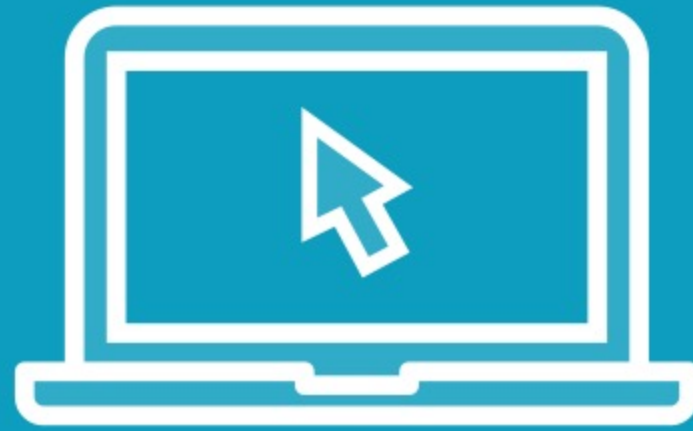
# Formatting Dates and Numbers

---





# Demo



- **Formatting and calculating dates**
  - DATEPART()
  - DATENAME()
  - Concatenate with dates
  - Cases for CAST() and CONVERT()
- **Use case for manipulating and calculating data from date fields**
- **Data integrity study**



## Formatting

```
SELECT DATENAME(month, '06/25/2025') -- Returns June
```

```
SELECT DATEPART(month, '06/25/2025') -- Returns 5
```

```
SELECT CONVERT(varchar(50), GETDATE(), 101) -- Returns mm/dd/yyyy
```

```
SELECT CAST(GETDATE() as varchar(11)) -- Returns mon d yyyy
```

## Calculating

```
SELECT DATEADD(d, 30, '06/25/2025') -- Returns 3/3/2025
```

```
SELECT DATEDIFF(d, '01/01/2025', '02/01/2025') -- Returns 31
```

# Formatting and Calculating Dates

May have to use with **Error handling** with `TRY_CAST()` and `TRY_CONVERT()`

Formatting for reports

Calculating for analysis

```
CASE WHEN when_expression THEN result_expression  
  
ELSE else_result_expression  
  
END
```

### Example

```
CASE  
  
    WHEN 'y' THEN 'yes'  
  
    WHEN 'n' THEN 'no'  
  
    ELSE false  
  
END
```

## Simple CASE Expression

Evaluates a list of conditions on an equality check only

Nested evaluation for 10 levels

Can be used in any statement or clause that allows a valid expression.

SELECT, UPDATE, DELETE, SET, WHERE, ORDER BY, and HAVING

CASE

WHEN price = 0 THEN 'not for sale'

WHEN price < 50 THEN 'bargain price'

WHEN price >= 51 THEN 'list price'

ELSE 'price unknown'

END

## Searched CASE Expression

Evaluates a list of conditions on a comparison check

Nested evaluation for 10 levels

Can be used in any statement or clause that allows a valid expression.

SELECT, UPDATE, DELETE, SET, WHERE, ORDER BY, and HAVING

# Shaping Data in a Query

## Summary



- **SELECT Query Syntax**
- **Concatenation**
- **Cast and Convert**
  - Try\_Cast( ) and Try\_Convert( )
- **Formatting**
  - Strings
  - Characters
  - Introduction to User Defined Functions
- **Dates and Numeric Values**
  - Calculating and formatting
  - Dates
  - Numeric values
  - CASE Expressions
- **NULL Values**
- **Data integrity and reporting accuracy!**



# Exercise Files

M3 – Calling a User Defined Function  
M3 – Casting and Converting Data Types 2  
M3 – Casting and Converting Data Types  
M3 – Casting and Converting in T-SQL  
M3 – Character Functions  
M3 – Concatenation sample  
M3 – Datepart  
M3 – Formatting Character Functions  
M3 – Formatting Strings – PatIndex  
M3 – Formatting Strings – Adding Concatenation  
M3 – Formatting Strings – Characters 2  
M3 – Formatting Strings – Characters

M3 – Formatting Strings – REPLACE  
M3 – Formatting Strings – Simplifying with Temps and UDFs  
M3 – Formatting Strings – SUBSTRINGS  
M3 – Formatting Strings – TRIM  
M3 – Formatting Strings – UPPER and LOWER  
M3 – Formatting Strings 2  
M3 – Formatting Strings  
M3 – REGEX  
M3 – User Defined Functions OrderAmountPurchased  
M3 – User Defined Functions Proper  
M3 – User Defined Functions ProperWithCharacters  
M3 – Working with Numbers



Up Next:

Querying Data from Multiple Sources using  
Joins

---

