

Built-in Functions

Functions and Wildcards in SQL Server



SoftUni Team
Technical Trainers



SoftUni

Software University

<https://softuni.bg>

Table of Contents

1. Function Overview
2. String Functions
3. Math Functions
4. Date Functions
5. Other Useful Functions
6. Wildcards



sli.do

#csharp-db



Functions in SQL Server

Overview

SQL Functions

- **Aggregate functions**

- Perform a calculation on a set of values and return a single value
- Examples: **AVG, COUNT, MIN, MAX, SUM**

- **Analytic functions**

- Compute an aggregate value based on a group of rows
- Unlike aggregate functions, analytic functions can return multiple rows for each group

```
PERCENTILE_CONT(0.5) WITHIN GROUP (ORDER BY Salary DESC)  
OVER (PARTITION BY DepartmentId) AS MedianCont
```



SQL Functions

- **Ranking functions**

- Return a ranking value for each row in a partition
- **RANK, ROW_NUMBER, DENSE_RANK, NTILE (OVER)**

- **Rowset functions**

- Return an object that can be used like table references in a statement
- **OPENDATASOURCE, OPENJSON, OPENXML, OPENROWSET**

- **Scalar functions**

- Operate on a single value and then return a single value
- Scalar functions can be used wherever an expression is valid





String Functions

- **Concatenation** – combines strings

```
SELECT FirstName + ' ' + LastName  
       AS [Full Name]  
FROM Employee
```

```
SELECT CONCAT(FirstName, ' ', LastName)  
       AS [Full Name]  
FROM Employee
```

- **CONCAT** replaces **NULL** values with **empty string**
- **CONCAT_WS** combines strings with separator

String Functions (2)

- **SUBSTRING** – extracts a part of a string

```
SUBSTRING(String, StartIndex, Length)
```

```
SUBSTRING('SoftUni', 5, 3)
```



Uni

- Example: get short **summary** of an article

```
SELECT ArticleId, Author, Content,  
       SUBSTRING(Content, 1, 200) + '...' AS Summary  
FROM Articles
```

- **REPLACE** – replaces a specific string with another

```
REPLACE(String, Pattern, Replacement)
```

```
REPLACE('SoftUni', 'Soft', 'Hard')
```

HardUni

- Example: **cancel** the word blood from album names

```
SELECT REPLACE(Title, 'blood', '*****')  
  AS Title  
FROM Album
```

- **LTRIM** & **RTRIM** – remove spaces from either side of string

```
LTRIM(String)
```

```
RTRIM(String)
```

- **LEN** – counts the number of characters

```
LEN(String)
```

- **DATALength** – gets the number of used bytes

```
DATALength(String)
```

String Functions (5)

- **LEFT & RIGHT** – get characters from the beginning or the end of a string

```
LEFT(String, Count)
```

```
RIGHT(String, Count)
```

- Example: name **shortened** (first 3 letters)

```
SELECT Id, Start,  
       LEFT(Name, 3) AS Shortened  
FROM Games
```

String Functions (6)

- **LOWER & UPPER** – change letter casing

```
LOWER(String)
```

```
UPPER(String)
```

- **REVERSE** – reverses order of all characters in a string

```
REVERSE(String)
```

- **REPLICATE** – repeats a string

```
REPLICATE(String, Count)
```

- **FORMAT** – format a value with a valid .NET format string

```
FORMAT(SomeDate, 'yyyy-MMMM-dd', 'bg-BG')
```

Problem: Obfuscate CC Numbers

- The database contains credit card details for customers
- Provide a summary without revealing the serial numbers

ID	FirstName	LastName	PaymentNumber
1	Guy	Gilbert	5645322227179083
2	Kevin	Brown	4417937746396076
...



ID	FirstName	LastName	PaymentNumber
1	Guy	Gilbert	564532*****
2	Kevin	Brown	441793*****
...

Solution : Obfuscate CC Numbers

- Reveal the first 6 digits and obfuscate the rest

```
SELECT CustomerID,  
       FirstName,  
       LastName,  
       LEFT(PaymentNumber, 6) + '*****'  
FROM Customers
```

- Bonus – create a View for the use of clients

```
CREATE VIEW v_PublicPaymentInfo AS  
...
```

- **CHARINDEX** – locates a specific pattern (substring) in a string

Optional, begins at 1

CHARINDEX(Pattern, String, [StartIndex])

- **STUFF** – inserts a substring at a specific position

STUFF(String, StartIndex, Length, Substring)

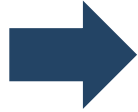
Number of chars
to delete



Math Functions

Arithmetic, PI, ABS, ROUND, Etc.

- | Id | A | H |
|----|-----|----|
| 1 | 2 | 4 |
| 2 | 1 | 18 |
| 3 | 4.5 | 3 |
| 4 | 8 | 12 |
| 5 | 3 | 5 |



Id	Area
1	4
2	9
3	6.75
4	48
5	7.5

Math Functions (2)

- **PI** – gets the value of Pi as a float (15 –digit precision)

```
SELECT PI() --3.14159265358979
```

- **ABS** – absolute value

```
ABS(Value)
```

- **SQRT** – square root (the result will be float)

```
SQRT(Value)
```

- **SQUARE** – raise to power of two

```
SQUARE(Value)
```

Example: Line Length

- Find the length of a line by given coordinates of the end points

Id	X1	Y1	X2	Y2
1	0	0	10	0
2	0	0	5	3
4	-1	5	8	-3
5	18	23	8882	134



Id	Length
1	10
2	5.8309518948453
4	12.0415945787923
5	8864.69497501183

```
SELECT Id,  
       SQRT(SQUARE(X1-X2) + SQUARE(Y1-Y2))  
       AS Length  
FROM Lines
```

- **POWER** – raises value to the desired exponent

```
POWER(Value, Exponent)
```

- **ROUND** – obtains the desired precision
 - Negative precision rounds characters before the decimal point

```
ROUND(Value, Precision)
```

- **FLOOR & CEILING** – return the nearest integer

```
FLOOR(Value)
```

```
CEILING(Value)
```

Problem: Pallets

- Calculate the required number of pallets to ship each item
 - **BoxCapacity** specifies how many items can fit in one box
 - **PalletCapacity** specifies how many boxes can fit in a pallet

Id	Name	Quantity	BoxCapacity	PalletCapacity
1	Perlenbacher 500ml	108	6	18
2	Perlenbacher 500ml	10	6	18
3	Chocolate Chips	350	24	3
4	Oil Pump	100	1	12
5	OLED TV 50-Inch	13	1	5
6	Penny	1	2239488	1



Number of pallets
1
1
5
9
3
1

- Since we can't use half a box or half a pallet, we need to round up to the nearest integer value

```
SELECT
    CEILING(
        CEILING(
            CAST(Quantity AS float) /
            BoxCapacity) / PalletCapacity)
    AS [Number of pallets]
FROM Products
```

- **SIGN** – returns 1, -1 or 0, depending on the value of the sign

SIGN(Value)

- **RAND** – gets a random float value in the range [0, 1]
 - If Seed is not specified, it will be assigned randomly

RAND()

RAND(Seed)



Date Functions

GETDATE, DATEDIFF, DATEPART, Etc.

- **DATEPART** – extract a segment from a date as an integer
 - Part can be any part and format of date or time

DATEPART(Part, Date)

year, yyyy, yy

month, mm, m

day, dd, d

YEAR(Date)

MONTH(Date)

DAY(Date)

- For a full list, take a look at the [official documentation](#)

Problem: Quarterly Report

- Prepare sales data for aggregation by displaying yearly quarter, month, year and day of sale

InvoiceId	InvoiceDate	Total
1	2023-01-01	1.98
2	2023-01-02	3.96
3	2023-01-03	5.94
4	2023-01-06	8.91



InvoiceId	Total	Quarter	Month	Year	Day
1	1.98	1	1	2023	1
2	3.96	1	1	2023	2
3	5.94	1	1	2023	3
4	8.91	1	1	2023	6

Solution: Quarterly Report

- Use **DATEPART** to get the relevant parts of the date

```
SELECT InvoiceId, Total,  
       DATEPART(QUARTER, InvoiceDate) AS Quarter,  
       DATEPART(MONTH, InvoiceDate) AS Month,  
       DATEPART(YEAR, InvoiceDate) AS Year,  
       DATEPART(DAY, InvoiceDate) AS Day  
FROM Invoice
```

- This statement might be useful as a View

Date Functions (2)

- **DATEDIFF** – finds the difference between two dates
 - **Part** can be **any part** and **format** of date or time

```
DATEDIFF(Part, FirstDate, SecondDate)
```

- Example: Show employee experience

```
SELECT ID, FirstName, LastName,  
       DATEDIFF(YEAR, HireDate, '2017/01/25')  
       AS [Years In Service]  
FROM Employees
```

Date Functions (3)

- **DATENAME** – gets a string representation of a date's part

```
DATENAME(Part, Date)
```

```
SELECT DATENAME(weekday, '2017/01/27')
```

- **DATEADD** – performs date arithmetic

- **Part** can be **any part** and **format** of date or time

```
DATEADD(Part, Number, Date)
```

- **GETDATE** – obtains the current date and time

```
SELECT GETDATE()
```

- **EOMONTH** – returns the last day of the month



Other Functions

CAST, CONVERT, OFFSET, FETCH

- **CAST & CONVERT** – conversion between data types

```
CAST(Data AS NewType)
```

```
CONVERT(NewType, Data)
```

- **ISNULL** – swaps **NULL** values with a specified **default value**

```
ISNULL(Data, DefaultValue)
```

- Example: Display "Not Finished" for projects with no **EndDate**

```
SELECT ProjectID, Name,  
       ISNULL(CAST(EndDate AS varchar), 'Not Finished')  
FROM Projects
```


- **COALESCE** – evaluates the arguments in order and returns the current value of the first expression that initially does not evaluate to **NULL**

```
SELECT COALESCE(NULL, NULL, 'third_value',  
  'fourth_value');  
  
// third_value
```

- **OFFSET & FETCH** – get only specific rows from the result set
 - Used in combination with **ORDER BY** for pagination

```
SELECT ID, FirstName, LastName  
FROM Employees  
ORDER BY ID  
OFFSET 10 ROWS  
FETCH NEXT 5 ROWS ONLY
```

Rows to skip

Rows to include

- **ROW_NUMBER** – always generate unique values without any gaps, even if there are ties
- **RANK** – can have gaps in its sequence and when values are the same, they get the same rank
- **DENSE_RANK** – returns the same rank for ties, but it doesn't have any gaps in the sequence
- **NTILE** – Distributes the rows in an ordered partition into a specified number of groups



Wildcards

Selecting Results by Partial Match

- **Wildcards** are used with **WHERE** for partial filtration
- Similar to **Regular Expressions**, but **less capable**
- Example: Find all employees who's first name **starts with "Ro"**

```
SELECT ID, FirstName, LastName  
FROM Employees  
WHERE FirstName LIKE 'Ro%'
```

Wildcard symbol

- Supported characters include:

```
%      -- any string, including zero-length  
_      -- any single character  
[...]  -- any character within range  
[^...] -- any character not in the range
```

- ESCAPE** – specify a prefix to treat special characters as normal

```
SELECT ID, Name  
FROM Tracks  
WHERE Name LIKE '%max!%' ESCAPE '!'
```

- Various **built-in functions**
- String functions - **CONCAT**, **LEFT/RIGHT**, **REPLACE**, etc.
- Math functions - **PI**, **ABS**, **POWER**, **ROUND**, etc.
- Date functions - **DATEPART**, **DATEDIFF**, **GETDATE**, etc.
- Using **Wildcards**, we can obtain results by partial string matches



SoftUni Diamond Partners

SCHWARZ



Coca-Cola HBC
Bulgaria



Postbank

Решения за твоето утре



POKERSTARS



CAREERS



AMBITIONED

DXC
TECHNOLOGY



SOFTWARE
GROUP

Bosch.IO

INDEAVR
Serving the high achievers

DRAFT
KINGS

PHAR
VISION



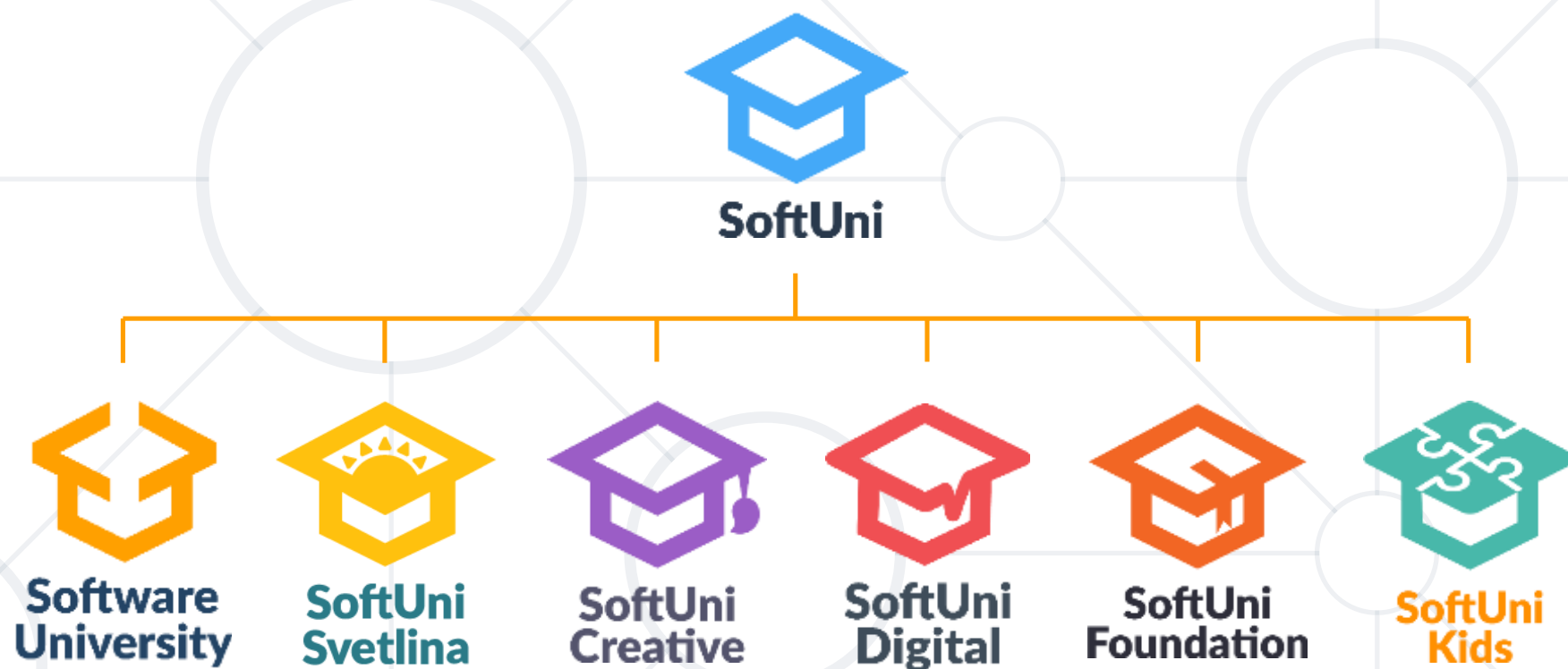
SmartIT

createX

SUPER
HOSTING
.BG



Questions?



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity

- Software University Forums

- forum.softuni.bg



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

