# Database Management
# Scalar Functions

By: Meyer Tanuan (2022), Rick Kozak (2019), Glenn Paulley (2015), John Mckay (2011)

## Introduction to Scalar Functions

- SQL functions can be used in the column list and in any search condition in a statement, including the WHERE and HAVING clauses, a join condition, even an ORDER BY clause
- Functions can accept zero, one, or more arguments
- Functions can return zero or one value
- Many functions such as GETDATE() and UPPER() are “built in” to SQL Server
- It is possible for a developer to write their own SQL function – termed a user-defined function

## Scalar function examples

```
SELECT 2 + 2;

SELECT GETDATE( );

SELECT SQRT( 2 );

SELECT SOUNDEX( 'Roselius');
```

CONESTOGA
Connect Life and Learning

## Scalar functions in the SELECT clause

```
SELECT invoiceNumber, [date],
DATEPART(MM, [date]) AS [month]
FROM dbo.Audit;
```

```
SELECT DISTINCT
SUBSTRING(mainPhone, 1, 3) AS prefix
FROM Person;
```

## Scalar functions in the WHERE clause

```
SELECT invoiceNumber, [date]
FROM dbo.Audit
WHERE DATEPART(MM, [date]) = 8;


SELECT lastName, firstName
FROM Person
WHERE UPPER(SUBSTRING(lastName,1, 2)) = 'MC';
```

CONESTOGA
Connect Life and Learning

## String Functions: SUBSTRING()

**SUBSTRING**(*string*, *start*, *count*):

```
SELECT SUBSTRING(city, 1, 3)
FROM Person;
```

*Note: In SQL, strings are indexed starting with 1 (not 0)*

## String Functions: LEFT(), RIGHT()

**LEFT**(*string*, *count*):

```
SELECT LEFT(city, 3)
FROM Person;
```

**RIGHT**(*string*, *count*):

- Does the same thing, but starting from the right side (end) of the string

## String Functions: CHARINDEX ()

**CHARINDEX**(*searchFor*, *searchIn*):

```
SELECT *
FROM Person
WHERE CHARINDEX('Mc', lastName) > 0;
```

*Why is the filter condition “> 0”?*

## String Concatenation

**String concatenation** is done with “+” in SQL Server

```
SELECT lastName + '(' + firstName + ')' AS
completeName
FROM Person;
```

## Other String Functions

Other common string functions

- LEN(string) – get the length of a string
- LTRIM(string) – trim a string of leading whitespace characters
- RTRIM(string) – trim a string of trailing whitespace characters
- LOWER(string) – convert a string to lower case
- UPPER(string) – convert a string to upper case

Note: in SQL Server (by default) all string comparisons are case insensitive

## Formatting Money Amounts

To display a money amount using Canadian / U.S. conventions:

- **CAST** the value to the **MONEY** data type
- Then use the **CONVERT** function to convert the MONEY type to CHAR(n) or VARCHAR(n) with style 1

```
'$' + CONVERT(CHAR(12), CAST(amount AS MONEY),
1)
```

*Note: CHAR(n) will make it column aligned on the decimal point*

## Math Functions

| | | |
|---|---|---|
| ABS | DEGREES | RAND |
| ACOS | EXP | ROUND |
| ASIN | FLOOR | SIGN |
| ATAN | LOG | SIN |
| ATN2 | LOG10 | SQUARE |
| CEILING | PI | SQRT |
| COS | POWER | TAN |
| COT | RADIANS | |

CONESTOGA
Connect Life and Learning

## Math Functions: ROUND()

This query demonstrates rounding and truncating numeric columns:

```
SELECT id, item, amountPerSemester,
   ROUND(amountPerSemester,0) AS rounded,
   ROUND(amountPerSemester,0,1) AS truncated
FROM IncidentalFee
ORDER BY amountPerSemester;
```

*Note: The third parameter of the ROUND function toggles between rounding (missing or 0) and truncating (1)*

## Random Numbers with RAND()

**RAND**() returns a pseudo-random, double-precision floating point value in range [0,1):

```
SELECT RAND() AS 'Random Number';
SELECT RAND() AS 'Random Number';
```

You can scale the value returned by **RAND**()

- To obtain a random number in the range 0-n, multiply by n+1
- To obtain a random number in the range 0-999,999 multiply by 1000000
- A random number in this range will be up to 6 digits long

*Note: You can scale the value returned by RAND() and pad it on the right with zeroes to create a fixed-size result*

CONESTOGA
Connect Life and Learning

## Date Types

- DATE
- TIME
- DATETIME
- SMALLDATETIME
- DATETIME2
- DATETIMEOFFSET

## Date Arithmetic (add)

- A number of days can be added to a DATETIME column with the “+” operator

```
SELECT dueDate, dueDate + 7 'one week later'
FROM dbo.Invoice
ORDER BY dueDate;
```

CONESTOGA
Connect Life and Learning

## Date Arithmetic (subtract)

- A number of days can be subtracted from a DATETIME column with the “-” operator

```
SELECT dueDate, dueDate + 7 'one week earlier'
FROM dbo.Invoice
ORDER BY dueDate;
```

CONESTOGA
Connect Life and Learning

## Date / Time Functions

- DATEADD()
- DATEDIFF()
- DATENAME()
- DATEPART()
- DAY()
- GETDATE()
- GETUTCDATE()
- MONTH()
- YEAR()

CONESTOGA
Connect Life and Learning

## DAY(), MONTH(), YEAR()

- **GETDATE**( ) returns the current date
  - Returns the date as a DATETIME type with up to 1/300 second accuracy

- There are specific functions to extract day, month and year from a date

```
SELECT DAY( expression )
SELECT MONTH( expression )
SELECT YEAR( expression )
```

## Date part

- Use **DATEPART**( ) to extract “part” of a date:

  SELECT **DATEPART**( **dp**, column )

- Where date part (**dp**) is:
  - A datepart code
    - Year, Quarter, Month, DayofYear, Day, Week, Weekday, Hour, Minute, Second, Millisecond
  - A datepart abbreviation
    - yy, yyyy, qq, q, mm, m, dy, y, dd, d, wk, ww, dw, hh, mi, n, ss, s, ms

## DATEPART(): Date and Time

- Returns the same results as DAY(), MONTH() and YEAR():
  - SELECT DATEPART(DD, column)
  - SELECT DATEPART(MM, column)
  - SELECT DATEPART(YYYY, column)

- Returns the time (hours – 24 hour clock, minutes, seconds, milliseconds):

```
SELECT DATEPART(HH, column)
SELECT DATEPART(MI, column)
SELECT DATEPART(SS, column)
SELECT DATEPART(MS, column)
```

CONESTOGA
Connect Life and Learning

## More DATEPART()

Returns the financial quarter (1-4):

```
SELECT DATEPART(Q, column)
```

Returns the day of the year:

```
SELECT DATEPART(DY, column)
```

Returns the weekday (Sunday = 1):

```
SELECT DATEPART(DW, column)
```

Note: The date part parameter is ***not*** a string literal enclosed in quotation marks

## DATEADD() and DATEDIFF()

**DATEADD** – Adds a number of units to a given date:

```
SELECT DATEADD( YEAR, 2, GETDATE())
```

**DATEDIFF** – Determines the difference in day/time units between two DATETIME types:

```
SELECT DATEDIFF( YEAR,
  DATEADD(YEAR, 2, GETDATE()) , GETDATE())
```

## More DATEADD()

You can also use **DATEADD( )** to do date arithmetic:

```
   SELECT DATEADD(dp, number, column)
```

where:

- dp is a datepart code (e.g. dd)
- number is the number to add (e.g. 1)
  - To subtract (go backwards in time), use a negative number (e.g. -1)

## DATEADD() Example

```
SELECT [date],
  DATEADD(day, 1, [date]) AS 'tomorrow',
  DATEADD(ww, 1, [date]) AS 'next week',
  DATEADD(mm, 1, [date]) AS 'next month',
  DATEADD(yy, -1, [date]) AS 'last year',
  DATEADD(yy, 1, [date]) AS 'next year'
FROM StudentOffence;
```

## More DATEDIFF()

Use DATEDIFF( ) to calculate the number of datepart units between two dates:

```
DATEDIFF(dp, column1, column2)
```

where

- dp is a datepart code
- column2 is subtracted from column1

Note: The value returned is an integer, not a date.

CONESTOGA
Connect Life and Learning

## DATEDIFF() Example

```
SELECT
 transactionDate, dueDate,
 DATEDIFF(DAY, transactionDate, dueDate)
     AS 'Days to Pay'
FROM Invoice;
```

CONESTOGA
Connect Life and Learning

## GETDATE() returns “now”

**GETDATE**() returns “now” as a DATETIME type.

- Because of this, the result of an expression using date arithmetic can be misleading or confusing because of implicit type conversions:

```
SELECT GETDATE() - '2000-01-01'
  AS 'Days since the millenium';
```

Returns a weird result:

```
1922-09-03 14:40:58.800
```

Because SQL Server interprets this as the number of days since January 1, 1900

## CASTing Dates

If we cast the result to an INTEGER:

```
SELECT
 CAST(
  GETDATE()-'2000-01-01' AS INTEGER)
  AS 'Days since the millenium';
```

Returns the correct result:

```
8281
```

## DATE Formatting: CONVERT()

Use the **CONVERT**( ) function to format the date differently, such as:

```
2022.09.02
```

Example:

```
SELECT CONVERT(
   CHAR(10), GETDATE(), 102);
```

- CHAR(10) defines a 10 character string, big enough for YYYY.MM.DD
- 102 is a style code. For a list of style codes see the Microsoft SQL Server 2017 documentation:

  https://msdn.microsoft.com/en-us/library/ms187928.aspx

CONESTOGA
Connect Life and Learning

## DATE Formatting: FORMAT()

Use the **FORMAT**( ) function to format the date differently, such as:

```
September 02, 2022
```

Example:

```
SELECT FORMAT( GETDATE(), 'MMMM dd, yyyy')
  AS FormattedDate;
```

For a list of all formatting keywords see the Microsoft SQL Server documentation:

https://docs.microsoft.com/en-us/sql/t-sql/functions/format-transact-sql?view=sql-server-2017

Experiment with other style codes to see what the output looks like.

CONESTOGA
Connect Life and Learning