



# Introduction to Concatenation and Scalar Functions

- Concatenate Columns
- Scalar Function Types
- String Functions
- Date and Time Functions
- Math Functions
- System Functions



# Column Concatenation

- It is possible to stitch together two or more columns or expressions to create a new derived column. This is known as concatenation
- There are two ways to concatenate expressions
  - Use the plus sign (+)
  - Use the CONCAT() function
- The new column you create will not have a name. Use the “AS” keyword to assign a name



# Concatenate with + Sign

- The + sign can be used to concatenate strings together
- All values must be a string otherwise an error is generated (The + sign will attempt to add together number and date datatypes)
- If a single value in the expression is Null then the entire expression will evaluate to Null

```
SELECT  
    FirstName + ' ' + LastName AS FullName  
    ,FirstName + ' ' + LastName + NULL AS NullTrumps  
    ,1 + 2 AS Addition  
    , '1'+'2' AS Concatenation  
FROM Employee
```

	FullName	NullTrumps	Addition	Concatenation
1	Andrew Adams	NULL	3	12
2	Nancy Edwards	NULL	3	12
3	Jane Peacock	NULL	3	12
4	Margaret Park	NULL	3	12
5	Steve Johnson	NULL	3	12
6	Michael Mitchell	NULL	3	12
7	Robert King	NULL	3	12
8	Laura Callahan	NULL	3	12



# Scalar Functions

- A function is a SQL statement that accepts input parameters, performs actions, and returns a result
- A scalar function is a type of function that operates on zero or more values in a row and returns a single value
- The general syntax of a function consists of the function name followed by parenthesis
- Zero or more parameters can be entered between the parenthesis. Each function has its own parameter requirements
- Examples: `GETDATE(); LEFT(FirstName,5); LEN(AlbumTitle); YEAR('6/15/2015')`



# Scalar String Functions

- Concat()
- Left()
- Right()
- SubString()
- Len()
- Ltrim()
- Rtrim()
- PatIndex()
- CharIndex()
- Replace()
- Reverse()
- Upper()
- Lower()

# Concatenate with CONCAT() Function

- This function takes multiple string values separated by commas and merges them together on a single line
- Numbers and Dates will be implicitly converted into string format
  - (+ sign will generate an error if you concatenate a string and non-string)
- Null values are converted to an empty string which prevents a single Null value from nullifying the entire expression

```
SELECT
    Company
    ,Country
    ,CONCAT(Country,', ',Company) AS ConcatExample
    ,Country + ', ' + Company AS PlusSignExample
FROM Customer
```

	Company	Country	ConcatExample	PlusSignExample
1	Embraer - E...	Brazil	Brazil, Embraer - Em...	Brazil, Embraer - Em...
2	NULL	Germany	Germany,	NULL
3	NULL	Canada	Canada,	NULL
4	NULL	Norway	Norway,	NULL
5	JetBrains s.r.o.	Czech Republic	Czech Republic, Jet...	Czech Republic, Jet...
6	NULL	Czech Republic	Czech Republic,	NULL
7	NULL	Austria	Austria,	NULL
8	NULL	Belgium	Belgium,	NULL
9	NULL	Denmark	Denmark,	NULL
10	Woodstock ...	Brazil	Brazil, Woodstock D...	Brazil, Woodstock Di...
11	Banco do Br...	Brazil	Brazil, Banco do Bra...	Brazil, Banco do Bra...
12	Riotur	Brazil	Brazil, Riotur	Brazil, Riotur
13	NULL	Brazil	Brazil,	NULL
14	Telus	Canada	Canada, Telus	Canada, Telus

# Left() and Right()

- Functions return a set number of characters starting from the Left or Right of an expression
- Takes 2 parameters
  - Expression
  - Length

```
SELECT  
    Name,  
    LEFT(Name,5) LeftSide  
    ,RIGHT(Name,5) RightSide  
FROM Artist
```

Results			
Messages			
	Name	Left Side	Right Side
1	AC/DC	AC/DC	AC/DC
2	Accept	Accep	ccept
3	Aerosmith	Aeros	smith
4	Alanis Morissette	Alani	sette
5	Alice In Chains	Alice	hains
6	Antônio Carlos Jobim	Antôn	Jobim
7	Apocalyptica	Apoca	ptica
8	Audioslave	Audio	slave
9	BackBeat	BackB	kBeat
10	Billy Cobham	Billy	obham
11	Black Label Society	Black	ciety
12	Black Sabbath	Black	bbath
13	Body Count	Body	Count

# SubString()

- Substring can be used to pull data of a specific starting point and length
- Takes 3 parameters
  - Expression
  - Starting Point
  - Length

```
SELECT
    ArtistId
    ,Name
    ,SUBSTRING(Name,1,5) Beginning
    ,SUBSTRING(Name,7,2) Middle
    ,SUBSTRING(Name,10,6) Ending
FROM Artist
```

Results		Messages			
	ArtistId	Name	Beginning	Middle	Ending
1	1	AC/DC	AC/DC		
2	2	Accept	Accep		
3	3	Aerosmith	Aeros	it	
4	4	Alanis Morissette	Alani	M	risset
5	5	Alice In Chains	Alice	In	Chains
6	6	Antônio Carlos Jobim	Antôn	o	arlos



# Len()

- The Len function returns the number of characters present in a string
- Takes a single expression parameter
- Spaces are counted as a character

```
SELECT  
    Name  
    , Len(Name) LengthOfName  
FROM Artist
```

Results			Messages		
	Name	LengthOfName			
1	AC/DC	5			
2	Accept	6			
3	Aerosmith	9			
4	Alanis Morissette	17			
5	Alice In Chains	15			
6	Antônio Carlos Jobim	20			
7	Apocalyptica	12			
8	Audioslave	10			
9	BackBeat	8			
10	Billy Cobham	12			
11	Black Label Society	19			
12	Black Sabbath	13			

# Ltrim() and Rtrim()

- LTrim and RTrim are functions used to trim leading or trailing spaces off of a character expression
- Char and Nchar datatypes will pad their values that don't fill the column length with trailing spaces

```
--I created a table variable to demonstrate the RTRIM function
--Focus on the SELECT statement for now
DECLARE @Person TABLE(FirstName char(15), LastName char(20))
INSERT INTO @Person
VALUES('Dave', 'Smith'),('Hilary', 'Thompson'),('George', 'Humphries')

SELECT
    FirstName
    ,LastName
    ,FirstName + LastName AS FullName
    ,RTRIM(FirstName) + ' ' + LastName AS TrimmedName
FROM @Person
```

	FirstName	LastName	FullName		TrimmedName
1	Dave	Smith	Dave	Smith	Dave Smith
2	Hilary	Thompson	Hilary	Thompson	Hilary Thompson
3	George	Humphries	George	Humphries	George Humphries

# Upper() and Lower()

- These functions take an expression and convert it to upper or lower case respectively

```
SELECT  
    Name  
    , LOWER(Name) AS UpperCase  
    , UPPER(Name) AS LowerCase  
FROM Artist
```

Results			
Messages			
	Name	UpperCase	LowerCase
1	AC/DC	ac/dc	AC/DC
2	Accept	accept	ACCEPT
3	Aerosmith	aerosmith	AEROSMITH
4	Alanis Morissette	alanis morissette	ALANIS MORISSETTE
5	Alice In Chains	alice in chains	ALICE IN CHAINS
6	Antônio Carlos Jobim	antônio carlos jobim	ANTÔNIO CARLOS JOBIM
7	Apocalyptica	apocalyptica	APOCALYPTICA
8	Audioslave	audioslave	AUDIOSLAVE
9	BackBeat	backbeat	BACKBEAT
10	Billy Cobham	billy cobham	BILLY COBHAM
11	Black Label Society	black label society	BLACK LABEL SOCIETY
12	Black Sabbath	black sabbath	BLACK SABBATH
13	Body Count	body count	BODY COUNT
14	Bruce Dickinson	bruce dickinson	BRUCE DICKINSON

# Reverse()

- This function takes an expression and reverses the order of the data in the field

```
SELECT  
    Name  
    , REVERSE(Name)  
FROM Artist
```

Results Messages		
	Name	(No column name)
1	AC/DC	CD/CA
2	Accept	tpeccA
3	Aerosmith	htimsoreA
4	Alanis Morissette	ettessiroM sinalA
5	Alice In Chains	sniahC nI ecilA
6	Antônio Carlos Jobim	miboJ solraC oinôtnA
7	Apocalyptica	acitpylacopA
8	Audioslave	evalsoiduA
9	BackBeat	taeBkcaB
10	Billy Cobham	mahboC ylliB
11	Black Label Society	yteicoS lebaL kcalB
12	Black Sabbath	htabbaS kcalB
13	Body Count	tnuoC ydoB
14	Bruce Dickinson	nosnikciD ecurB

# Replace()

- This function searches an expression for one character string and replaces it with another
- Takes 3 parameters
  - Expression
  - Search String
  - Replacement String

```
SELECT TOP 12
    Name
    , REPLACE(Name, 'in', 'under') Example1
    , REPLACE(Name, 'A', 'X') Example2
    , REPLACE(Name, 'black', 'Green') Example3
FROM Artist
```

	Name	Example1	Example2	Example3
1	AC/DC	AC/DC	XC/DC	AC/DC
2	Accept	Accept	Xcept	Accept
3	Aerosmith	Aerosmith	Xerosmith	Aerosmith
4	Alanis Morissette	Alanis Morissette	XIXnis Morissette	Alanis Morissette
5	Alice In Chains	Alice under Chaunders	Xlice In ChXins	Alice In Chains
6	Antônio Carlos Jobim	Antônio Carlos Jobim	Xntônio CXrlos Jobim	Antônio Carlos Jobim
7	Apocalyptica	Apocalyptica	XpocXlypticX	Apocalyptica
8	Audioslave	Audioslave	XudioslXve	Audioslave
9	BackBeat	BackBeat	BXckBeXt	BackBeat
10	Billy Cobham	Billy Cobham	Billy CobhXm	Billy Cobham
11	Black Label Society	Black Label Society	BIXck LXbel Society	Green Label Society
12	Black Sabbath	Black Sabbath	BIXck SXbbXth	Green Sabbath

# PatIndex()

- This function searches for a character pattern and returns the starting position of the first character if found
- Takes 2 parameters
  - Character pattern
  - Expression
- PatIndex takes wildcards
  - % zero or more of any type
  - \_ one of any type
  - [ ] one in the set or range
  - [^ ] one not in the set or range
- If the string is not found then a zero is returned

```
SELECT
    Name
    ,PATINDEX('%in%',Name) [In Search]
    ,PATINDEX('%A[bip]%',Name) [A+(B or I or P)]
    ,PATINDEX('B_ack%',Name) [B+(any)+ack]
FROM Artist
```

	Name	In Search	A+(B or I or P)	B+(any)+ack
1	AC/DC	0	0	0
2	Accept	0	0	0
3	Aerosmith	0	0	0
4	Alanis Morissette	0	0	0
5	Alice In Chains	7	12	0
6	Antônio Carlos Jobim	0	0	0
7	Apocalyptica	0	1	0
8	Audioslave	0	0	0
9	BackBeat	0	0	0
10	Billy Cobham	0	0	0
11	Black Label Society	0	8	1
12	Black Sabbath	0	8	1
13	Body Count	0	0	0
14	Bruce Dickinson	11	0	0

# CharIndex()

- Similar to Patindex except doesn't take wildcards
- However does have optional 3<sup>rd</sup> parameter for starting position
- Takes 3 parameters
  - Character pattern
  - Expression
  - Starting position (optional)
- If the string is not found then a zero is returned
- Useful for finding later instances of a character pattern in an expression

```
SELECT
    Name
    ,CHARINDEX('A',Name) [A Search]
    ,CHARINDEX('A',Name,CHARINDEX('A',Name)+1) [2nd A Search]
    ,CHARINDEX('in',Name) [In Seach]
    ,CHARINDEX('in',Name,8) [In+8 Search]
FROM Artist
```

	Name	A Search	2nd A Search	In Seach	In+8 Search
1	AC/DC	1	0	0	0
2	Accept	1	0	0	0
3	Aerosmith	1	0	0	0
4	Alanis Morissette	1	3	0	0
5	Alice In Chains	1	12	7	13
6	Antônio Carlos Jobim	1	10	0	0
7	Apocalyptica	1	5	0	0
8	Audioslave	1	8	0	0
9	BackBeat	2	7	0	0
10	Billy Cobham	11	0	0	0
11	Black Label Society	3	8	0	0
12	Black Sabbath	3	8	0	0
13	Body Count	0	0	0	0
14	Bruce Dickinson	0	0	11	11



# Scalar Date Functions

- GetDate()
- Current\_TimeStamp
- GetUTCDate()
- Day()
- Month()
- Year()
- DateName()
- DatePart()
- DateFromParts()
- DateTimeFromParts()
- DateDiff()
- DateAdd()
- IsDate()
- EOMonth()



# GetDate(), Current\_TimeStamp and GetUTCDate()

- All 3 functions return the current datetime
- There are not parameters for any of these functions
- GetDate and Current\_TimeStamp are identical. They return the current local datetime
- GetUTCDate returns the datetime without a timezone offset.

```
SELECT  
    GetDate() AS [GetDate]  
    ,Current_TimeStamp AS [TimeStamp]  
    ,GETUTCDATE() AS [GetUTCDate]
```

Results			
Messages			
	GetDate	TimeStamp	GetUTCDate
1	2015-09-01 20:16:45.830	2015-09-01 20:16:45.830	2015-09-02 03:16:45.840

# Day(), Month() and Year()

- Returns the specified date part as an integer
- Takes a date as a parameter

```
SELECT
    InvoiceId
    , InvoiceDate
    , DAY(InvoiceDate) AS InvoiceDay
    , MONTH(InvoiceDate) AS InvoiceMonth
    , YEAR(InvoiceDate) AS InvoiceYear
FROM Invoice
```

	InvoiceId	InvoiceDate	InvoiceDay	InvoiceMonth	InvoiceYear
1	1	2009-01-01 00:00:00.000	1	1	2009
2	2	2009-01-02 00:00:00.000	2	1	2009
3	3	2009-01-03 00:00:00.000	3	1	2009
4	4	2009-01-06 00:00:00.000	6	1	2009
5	5	2009-01-11 00:00:00.000	11	1	2009
6	6	2009-01-19 00:00:00.000	19	1	2009
7	7	2009-02-01 00:00:00.000	1	2	2009
8	8	2009-02-01 00:00:00.000	1	2	2009
9	9	2009-02-02 00:00:00.000	2	2	2009
10	10	2009-02-03 00:00:00.000	3	2	2009
11	11	2009-02-06 00:00:00.000	6	2	2009
12	12	2009-02-11 00:00:00.000	11	2	2009
13	13	2009-02-19 00:00:00.000	19	2	2009
14	14	2009-03-04 00:00:00.000	4	3	2009

# DateName()

- DateName accepts 2 parameters
  - datepart
  - date
- All output is returned as a \*string\*
- A full list of available dateparts can be found on the DateName webpage of the MSDN Library

```
SELECT
    EmployeeId
    , HireDate
    , DATENAME(DAYOFYEAR, HireDate) AS [DayOfYear]
    , DATENAME(QUARTER, HireDate) AS [Quarter]
    , DATENAME(WEEK, HireDate) AS [Week]
    , DATENAME(WEEKDAY, HireDate) AS [Weekday]
    , DATENAME(MONTH, HireDate) AS [Month]
FROM Employee
ORDER BY HireDate
```

	EmployeeId	HireDate	DayOfYear	Quarter	Week	Weekday	Month
1	3	2002-04-01 00:00:00.000	91	2	14	Monday	April
2	2	2002-05-01 00:00:00.000	121	2	18	Wednesday	May
3	1	2002-08-14 00:00:00.000	226	3	33	Wednesday	August
4	4	2003-05-03 00:00:00.000	123	2	18	Saturday	May
5	5	2003-10-17 00:00:00.000	290	4	42	Friday	October
6	6	2003-10-17 00:00:00.000	290	4	42	Friday	October
7	7	2004-01-02 00:00:00.000	2	1	1	Friday	January
8	8	2004-03-04 00:00:00.000	64	1	10	Thursday	March

# DatePart()

- DatePart accepts 2 parameters
  - datepart
  - date
- All output is returned as an \*integer\*
- A full list of available dateparts can be found on the DatePart webpage of the MSDN Library

```
SELECT
    EmployeeId
  , HireDate
  , DATEPART(DAYOFYEAR, HireDate) AS [DayOfYear]
  , DATEPART(QUARTER, HireDate) AS [Quarter]
  , DATEPART(WEEK, HireDate) AS [Week]
  , DATEPART(WEEKDAY, HireDate) AS [Weekday]
  , DATEPART(MONTH, HireDate) AS [Month]
FROM Employee
ORDER BY HireDate
```

	EmployeeId	HireDate	DayOfYear	Quarter	Week	Weekday	Month
1	3	2002-04-01 00:00:00.000	91	2	14	2	4
2	2	2002-05-01 00:00:00.000	121	2	18	4	5
3	1	2002-08-14 00:00:00.000	226	3	33	4	8
4	4	2003-05-03 00:00:00.000	123	2	18	7	5
5	5	2003-10-17 00:00:00.000	290	4	42	6	10
6	6	2003-10-17 00:00:00.000	290	4	42	6	10
7	7	2004-01-02 00:00:00.000	2	1	1	6	1
8	8	2004-03-04 00:00:00.000	64	1	10	5	3

# DateFromParts() and DateTimeFromParts()

- Both functions take integer input and output a date or datetime datatype

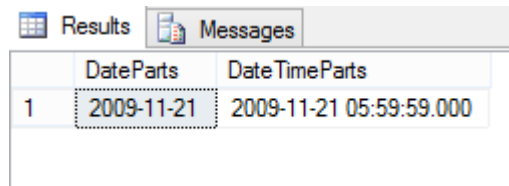
```
SELECT  
    DATEFROMPARTS(2009,11,21) AS DateParts  
    ,DATETIMEFROMPARTS(2009,11,21,5,59,59,0) AS DateTimeParts
```

- DateFromParts takes 3 parameters

- Year
- Month
- Day

- DateTimeFromParts takes 7 parameters

- Year
- Month
- Day
- Hour
- Minutes
- Seconds
- Milliseconds



The screenshot shows a SQL Server query results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active, displaying a table with two columns: 'DateParts' and 'DateTimeParts'. The first row of data shows the date '2009-11-21' under 'DateParts' and the datetime '2009-11-21 05:59:59.000' under 'DateTimeParts'.

	DateParts	DateTimeParts
1	2009-11-21	2009-11-21 05:59:59.000

# DateDiff()

- Returns the difference between two dates
- Takes 3 parameters
  - DatePart
  - StartDate
  - EndDate
- DatePart is used to define the unit of measurement

```
SELECT
    FirstName
    , LastName
    , BirthDate
    , HireDate
    , DATEDIFF(YEAR, BirthDate, HireDate) AgeWhenHired
    , DATEDIFF(DAY, HireDate, '1/1/2005') DaysEmployed
FROM Employee
```

	FirstName	LastName	BirthDate	HireDate	AgeWhenHired	DaysEmployed
1	Andrew	Adams	1962-02-18 00:00:00.000	2002-08-14 00:00:00.000	40	871
2	Nancy	Edwards	1958-12-08 00:00:00.000	2002-05-01 00:00:00.000	44	976
3	Jane	Peacock	1973-08-29 00:00:00.000	2002-04-01 00:00:00.000	29	1006
4	Margaret	Park	1947-09-19 00:00:00.000	2003-05-03 00:00:00.000	56	609
5	Steve	Johnson	1965-03-03 00:00:00.000	2003-10-17 00:00:00.000	38	442
6	Michael	Mitchell	1973-07-01 00:00:00.000	2003-10-17 00:00:00.000	30	442
7	Robert	King	1970-05-29 00:00:00.000	2004-01-02 00:00:00.000	34	365
8	Laura	Callahan	1968-01-09 00:00:00.000	2004-03-04 00:00:00.000	36	303

# DateAdd()

- Adds an integer based on a unit of measurement to a date
- Takes 3 parameters
  - DatePart
  - Number
  - Date
- DatePart is used to define the unit of measurement
- If the number is negative then the function will subtract the amount from the date

```
SELECT
    InvoiceId
    ,InvoiceDate
    ,DATEADD(YEAR,5,InvoiceDate) AS AddYear
    ,DATEADD(MONTH,5,InvoiceDate) AS AddMonth
    ,DATEADD(DAY,5,InvoiceDate) AS AddDay
    ,DATEADD(WEEK,5,InvoiceDate) AS AddWeek
FROM Invoice
```

Results		Messages				
	InvoiceId	InvoiceDate	AddYear	AddMonth	AddDay	AddWeek
1	1	2009-01-01 00:00:00.000	2014-01-01 00:00:00.000	2009-06-01 00:00:00.000	2009-01-06 00:00:00.000	2009-02-05 00:00:00.000
2	2	2009-01-02 00:00:00.000	2014-01-02 00:00:00.000	2009-06-02 00:00:00.000	2009-01-07 00:00:00.000	2009-02-06 00:00:00.000
3	3	2009-01-03 00:00:00.000	2014-01-03 00:00:00.000	2009-06-03 00:00:00.000	2009-01-08 00:00:00.000	2009-02-07 00:00:00.000
4	4	2009-01-06 00:00:00.000	2014-01-06 00:00:00.000	2009-06-06 00:00:00.000	2009-01-11 00:00:00.000	2009-02-10 00:00:00.000
5	5	2009-01-11 00:00:00.000	2014-01-11 00:00:00.000	2009-06-11 00:00:00.000	2009-01-16 00:00:00.000	2009-02-15 00:00:00.000
6	6	2009-01-19 00:00:00.000	2014-01-19 00:00:00.000	2009-06-19 00:00:00.000	2009-01-24 00:00:00.000	2009-02-23 00:00:00.000
7	7	2009-02-01 00:00:00.000	2014-02-01 00:00:00.000	2009-07-01 00:00:00.000	2009-02-06 00:00:00.000	2009-03-08 00:00:00.000
8	8	2009-02-01 00:00:00.000	2014-02-01 00:00:00.000	2009-07-01 00:00:00.000	2009-02-06 00:00:00.000	2009-03-08 00:00:00.000
9	9	2009-02-02 00:00:00.000	2014-02-02 00:00:00.000	2009-07-02 00:00:00.000	2009-02-07 00:00:00.000	2009-03-09 00:00:00.000
10	10	2009-02-03 00:00:00.000	2014-02-03 00:00:00.000	2009-07-03 00:00:00.000	2009-02-08 00:00:00.000	2009-03-10 00:00:00.000
11	11	2009-02-06 00:00:00.000	2014-02-06 00:00:00.000	2009-07-06 00:00:00.000	2009-02-11 00:00:00.000	2009-03-13 00:00:00.000
12	12	2009-02-11 00:00:00.000	2014-02-11 00:00:00.000	2009-07-11 00:00:00.000	2009-02-16 00:00:00.000	2009-03-18 00:00:00.000
13	13	2009-02-19 00:00:00.000	2014-02-19 00:00:00.000	2009-07-19 00:00:00.000	2009-02-24 00:00:00.000	2009-03-26 00:00:00.000
14	14	2009-03-04 00:00:00.000	2014-03-04 00:00:00.000	2009-08-04 00:00:00.000	2009-03-09 00:00:00.000	2009-04-08 00:00:00.000
15	15	2009-03-04 00:00:00.000	2014-03-04 00:00:00.000	2009-08-04 00:00:00.000	2009-03-09 00:00:00.000	2009-04-08 00:00:00.000
16	16	2009-03-05 00:00:00.000	2014-03-05 00:00:00.000	2009-08-05 00:00:00.000	2009-03-10 00:00:00.000	2009-04-09 00:00:00.000
17	17	2009-03-06 00:00:00.000	2014-03-06 00:00:00.000	2009-08-06 00:00:00.000	2009-03-11 00:00:00.000	2009-04-10 00:00:00.000



# IsDate()

- Checks whether the input is a valid date or not
- Takes a single parameter
- Returns 1 if valid
- Returns 0 if not valid

```
SELECT
    LastName
    ,BirthDate
    ,HireDate
    ,ISDATE(LastName) NotValidDate
    ,ISDATE(BirthDate) ValidDate1
    ,ISDATE(HireDate) ValidDate2
FROM Employee
```

Results		Messages				
	LastName	BirthDate	HireDate	NotValidDate	ValidDate1	ValidDate2
1	Adams	1962-02-18 00:00:00.000	2002-08-14 00:00:00.000	0	1	1
2	Edwards	1958-12-08 00:00:00.000	2002-05-01 00:00:00.000	0	1	1
3	Peacock	1973-08-29 00:00:00.000	2002-04-01 00:00:00.000	0	1	1
4	Park	1947-09-19 00:00:00.000	2003-05-03 00:00:00.000	0	1	1
5	Johnson	1965-03-03 00:00:00.000	2003-10-17 00:00:00.000	0	1	1
6	Mitchell	1973-07-01 00:00:00.000	2003-10-17 00:00:00.000	0	1	1
7	King	1970-05-29 00:00:00.000	2004-01-02 00:00:00.000	0	1	1
8	Callahan	1968-01-09 00:00:00.000	2004-03-04 00:00:00.000	0	1	1



# EOMonth()

- Returns the last day of the month for a given date
- Takes 2 parameters
  - Start Date
  - Month(s) to add (optional)
- The optional parameter takes an integer

```
SELECT  
    LastName  
    , HireDate  
    , EOMONTH(HireDate) AS HireDateEOM  
    , EOMONTH(HireDate, 6) AS SixMonthReview  
FROM Employee
```

	LastName	HireDate	HireDateEOM	SixMonthReview
1	Adams	2002-08-14 00:00:00.000	2002-08-31	2003-02-28
2	Edwards	2002-05-01 00:00:00.000	2002-05-31	2002-11-30
3	Peacock	2002-04-01 00:00:00.000	2002-04-30	2002-10-31
4	Park	2003-05-03 00:00:00.000	2003-05-31	2003-11-30
5	Johnson	2003-10-17 00:00:00.000	2003-10-31	2004-04-30
6	Mitchell	2003-10-17 00:00:00.000	2003-10-31	2004-04-30
7	King	2004-01-02 00:00:00.000	2004-01-31	2004-07-31
8	Callahan	2004-03-04 00:00:00.000	2004-03-31	2004-09-30



# Scalar Math, System and Logical Functions

- Math
  - SQRT()
  - Square()
  - Power()
- System
  - IsNull()
  - IsNumeric()
- Logical
  - IIF()
  - Cast()
  - Convert()



# Math Functions

## Sqrt(), Square() and Power()

- Mathematical functions take numeric input apply the operation consistent with their name
- Sqrt() and Square() take a single parameter
- Power() takes a number parameter plus a second as the power to use

```
SELECT  
    InvoiceId  
    ,SQRT(InvoiceId) AS IdSquareRoot  
    ,SQUARE(InvoiceId) AS IdSquare  
    ,POWER(InvoiceID,3) AS IdPower3  
FROM Invoice  
ORDER BY InvoiceId
```

Results		Messages		
	InvoiceId	IdSquareRoot	IdSquare	IdPower3
1	1	1	1	1
2	2	1.4142135623731	4	8
3	3	1.73205080756888	9	27
4	4	2	16	64
5	5	2.23606797749979	25	125
6	6	2.44948974278318	36	216
7	7	2.64575131106459	49	343
8	8	2.82842712474619	64	512
9	9	3	81	729
10	10	3.16227766016838	100	1000
11	11	3.3166247903554	121	1331
12	12	3.46410161513775	144	1728
13	13	3.60555127546399	169	2197
14	14	3.74165738677394	196	2744
15	15	3.87298334620742	225	3375
16	16	4	256	4096

# System Function IsNull()

- The function checks for null values in an expression and replaces nulls with a replacement expression
- Takes two parameters
  - Expression
  - Replacement Expression

```
SELECT  
    Company  
    , ISNULL(Company, 'N/A') AS CompanyNoNulls  
FROM Customer
```

Results		Messages
	Company	CompanyNoNulls
1	Embraer - Empresa Brasileira de Aeronáutica S.A.	Embraer - Empresa Brasileira de Aeronáutica S.A.
2	NULL	N/A
3	NULL	N/A
4	NULL	N/A
5	JetBrains s.r.o.	JetBrains s.r.o.
6	NULL	N/A
7	NULL	N/A
8	NULL	N/A
9	NULL	N/A
10	Woodstock Discos	Woodstock Discos
11	Banco do Brasil S.A.	Banco do Brasil S.A.
12	Riotur	Riotur
13	NULL	N/A
14	Telus	Telus
15	Rogers Canada	Rogers Canada
16	Google Inc.	Google Inc.

# IsNumeric()

- The function is used to determine if a string is a valid number or not.
- Takes a single parameter
- If the expression is numeric then output is 1 otherwise 0

```
SELECT  
    PostalCode  
    , ISNUMERIC(PostalCode) NumericValue  
FROM Customer
```

	PostalCode	NumericValue
1	12227-000	0
2	70174	1
3	H2G 1A7	0
4	0171	1
5	14700	1
6	14300	1
7	1010	1
8	1000	1
9	1720	1
10	01007-010	0
11	01310-200	0
12	20040-020	0
13	71020-677	0
14	T6G 2C7	0

# Logical Function Immediate IF IIF()

- Function is a simple IF THEN ELSE statement
- Takes 3 parameters
  - Condition
  - If true
  - If false

```
SELECT
    Fax
    ,Email
    ,IIF(Fax IS NULL, Email, Fax)
FROM Customer
```

Results		Messages	
	Fax	Email	(No column name)
1	+55 (12) 3923-5566	luisg@embraer.com.br	+55 (12) 3923-5566
2	NULL	leonekohler@surfeu.de	leonekohler@surfeu.de
3	NULL	ftremblay@gmail.com	ftremblay@gmail.com
4	NULL	bjom.hansen@yahoo.no	bjom.hansen@yahoo.no
5	+420 2 4172 5555	frantisekw@jetbrains.com	+420 2 4172 5555
6	NULL	hholy@gmail.com	hholy@gmail.com
7	NULL	astrid.gruber@apple.at	astrid.gruber@apple.at
8	NULL	daan_peeters@apple.be	daan_peeters@apple.be
9	NULL	kara.nielsen@jubii.dk	kara.nielsen@jubii.dk
10	+55 (11) 3033-4564	eduardo@woodstock.com.br	+55 (11) 3033-4564
11	+55 (11) 3055-8131	alero@uol.com.br	+55 (11) 3055-8131
12	+55 (21) 2271-7070	roberto.almeida@riotur.gov.br	+55 (21) 2271-7070
13	+55 (61) 3363-7855	femadaramos4@uol.com.br	+55 (61) 3363-7855

# Cast()

- Function is used to convert an expression from one datatype to another
- Takes 2 parameters but the parameters are separated by the AS keyword instead of a comma
- Parameters
  - Expression
  - Datatype to convert to

```
SELECT
    BillingPostalCode
    ,Total
    ,BillingPostalCode + ' | ' + Total
FROM Invoice
```

Results Messages

Msg 8114, Level 16, State 5, Line 1  
Error converting data type nvarchar to numeric.

```
SELECT
    BillingPostalCode
    ,Total
    ,BillingPostalCode + ' | ' + CAST(Total AS varchar)
FROM Invoice
```

	BillingPostalCode	Total	(No column name)
1	70174	1.98	70174   1.98
2	0171	3.96	0171   3.96
3	1000	5.94	1000   5.94
4	T6G 2C7	8.91	T6G 2C7   8.91
5	2113	13.86	2113   13.86

# Convert()

- Function is used to convert an expression from one datatype to another
- Takes 3 parameters
  - Datatype
  - Expression
  - Style
- The third parameter is optional. It can be used to display data in a certain format

```
SELECT
    InvoiceDate
    , CONVERT(varchar, InvoiceDate) ConvertDefault
    , CONVERT(varchar, InvoiceDate, 1) WithoutCentury
    , CONVERT(varchar, InvoiceDate, 101) WithCentury
FROM Invoice
```

	InvoiceDate	ConvertDefault	WithoutCentury	WithCentury
1	2009-01-01 00:00:00.000	Jan 1 2009 12:00AM	01/01/09	01/01/2009
2	2009-01-02 00:00:00.000	Jan 2 2009 12:00AM	01/02/09	01/02/2009
3	2009-01-03 00:00:00.000	Jan 3 2009 12:00AM	01/03/09	01/03/2009
4	2009-01-06 00:00:00.000	Jan 6 2009 12:00AM	01/06/09	01/06/2009
5	2009-01-11 00:00:00.000	Jan 11 2009 12:00AM	01/11/09	01/11/2009
6	2009-01-19 00:00:00.000	Jan 19 2009 12:00AM	01/19/09	01/19/2009
7	2009-02-01 00:00:00.000	Feb 1 2009 12:00AM	02/01/09	02/01/2009
8	2009-02-01 00:00:00.000	Feb 1 2009 12:00AM	02/01/09	02/01/2009
9	2009-02-02 00:00:00.000	Feb 2 2009 12:00AM	02/02/09	02/02/2009
10	2009-02-03 00:00:00.000	Feb 3 2009 12:00AM	02/03/09	02/03/2009
11	2009-02-06 00:00:00.000	Feb 6 2009 12:00AM	02/06/09	02/06/2009
12	2009-02-11 00:00:00.000	Feb 11 2009 12:00AM	02/11/09	02/11/2009
13	2009-02-19 00:00:00.000	Feb 19 2009 12:00AM	02/19/09	02/19/2009



# Multiple Functions

- You can merge multiple functions together in one query to create useful reports

```
SELECT
    CONCAT(FirstName, ' ', LastName) FullName
    , CONVERT(varchar, BirthDate, 101) BirthDate
    , CONVERT(varchar, HireDate, 101) HireDate
    , RIGHT(Phone, 8) PartialPhone
    , DATEDIFF(DAY, HireDate, GETDATE())/365 AS YearsOfService --GetDate = 9/7/2015 in this example
    , 65 - DATEDIFF(DAY, BirthDate, GETDATE())/365 YearsUntil65
    , IIF(DATEDIFF(DAY, HireDate, GETDATE())/365 > 12, 'Vested', 'Not Vested') RetirementStatus
FROM Employee
```

	FullName	BirthDate	HireDate	PartialPhone	YearsOfService	YearsUntil65	RetirementStatus
1	Andrew Adams	02/18/1962	08/14/2002	428-9482	13	12	Vested
2	Nancy Edwards	12/08/1958	05/01/2002	262-3443	13	9	Vested
3	Jane Peacock	08/29/1973	04/01/2002	262-3443	13	23	Vested
4	Margaret Park	09/19/1947	05/03/2003	263-4423	12	-3	Not Vested
5	Steve Johnson	03/03/1965	10/17/2003	836-9987	11	15	Not Vested
6	Michael Mitchell	07/01/1973	10/17/2003	246-9887	11	23	Not Vested
7	Robert King	05/29/1970	01/02/2004	456-9986	11	20	Not Vested
8	Laura Callahan	01/09/1968	03/04/2004	467-3351	11	18	Not Vested



# Summary

- Concatenate Columns
- Function Types
- String Functions
- Date and Time Functions
- Math Functions
- System Functions