

Aggregate Functions

Aggregate functions help define how we want values to be treated when we use a GROUP BY in our query.

Function	Description
SUM	Returns the sum of all the values, or only the DISTINCT values, in the expression. SUM works with numeric columns only. Null values are ignored.
AVG	Returns the average of the values in a group. It ignores null values.
COUNT	Returns the number of items found in a group.
MAX	Returns the maximum value in a group.
MIN	Returns the minimum value in a group

Numeric Data Types

Data Type	Description	Storage
bit	An integer data type that can take a value of 1, 0, or NULL	1 byte
tinyint	whole numbers from 0 to 255	1 byte
smallint	whole numbers between -32,768 and 32,767	2 bytes
int	whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
bigint	whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
Decimal(p,s)	<p>P = The maximum number of decimal digits to be stored. This number includes both the left and the right sides of the decimal point.</p> <p>S = The number of decimal digits that are stored to the right of the decimal point.</p> <p>Ex. decimal (4,2), 2 digits before the decimal point, two digits after. Allows up to +/- 10³⁸.</p>	5-17 bytes
Numeric(p,s)	Same as above	5-17 bytes
Smallmoney	Accurate to a ten-thousandth of the monetary units, from - 214,748.3648 to 214,748.3647	4 bytes
money	Accurate to a ten-thousandth of the monetary units, up to +/- 922,337,203,685,477.5808	8 bytes
Float(n)	<p>n is the number of bits that are used to store the mantissa of the float number and, therefore, dictates the precision and storage size. If n is specified, it must be a value between 1 and 53. The default value of n is 53.</p> <p>Takes values from -1.79E + 308 to 1.79E + 308.</p>	4-8 bytes
real	Floating precision number data from -3.40E + 38 to 3.40E + 38	4 bytes

Numeric Functions

Function	Description	Statement	Result
ROUND	Rounds a number to a specified number of decimal places	SELECT ROUND(2.36, 1)	2.4
ABS	Returns the absolute value of a number	SELECT ABS (-353)	353
CEILING	Returns the smallest integer value that is >= a number	SELECT CEILING (5.75)	6
FLOOR	Returns the largest integer value that is <= to a number	SELECT FLOOR (5.75)	5

Date and Time Data Types

Data Type	Format	Storage	Date range
DATETIME*	'YYYYMMDD hh:mm:ss.nnn' 2020-01-01 11:45:32.547	8 bytes	January 1 st ,1753 though December 31 st ,9999
SMALLDATETIME*	'YYYYMMDD hh:mm' 2020-01-01 11:45	4 bytes	January 1 st ,1900 through June 6 th ,2079
DATE	'YYYY-MM-DD' 2020-01-01	3 bytes	January 1 st ,0001 through Dec 31,9999
TIME	'hh:mm:ss:nnnnnnnn' 11:45:42.4356456	3-5 bytes	Stores times only to an accuracy of 100 nanoseconds
DATETIME2	'YYYYMMDD hh:mm:ss:nnnnnnnn' 2020-01-01 11:45:42.4356456	6 to 8 bytes	January 1 st ,0001 through Dec 31,9999
DATETIMEOFFSET	'YYYYMMDD hh:mm:ss:nnnnnnnn' +/- hh:mm' 2020-01-01 11:45:42.4356456 + 08:00	8 to 10 bytes	January 1 st ,0001 through Dec 31,9999

*These data types are legacy types, meaning they are still supported, but are not as up to date or accurate.

Datepart Values

Parameter	Description
year, yyyy, yy	Returns the year
quarter, qq, q	Returns the quarter
month, mm, m	Returns the year
day, dd, d	Returns the day of the month
dayofyear, dy, y	Returns the day of the year
week, ww, wk	Returns the week
weekday, dw, w	Returns the weekday
hour, hh	Returns the hour

Date and Time Functions

Function	Description	Statement	Result
GETDATE()	Returns the current date and time	SELECT GETDATE()	2020-12-03 04:16:09.247
DATENAME (<i>datepart, date</i>)	Returns a character string representing a specified datepart of a specified date.	SELECT DATENAME(month,'20200101')	January
DATEPART (<i>datepart, date</i>)	Returns an integer representing the specified datepart of the specified date.	SELECT DATEPART(year,'20200101')	2020
MONTH (<i>date</i>)	Returns an integer representing the month part of a specified date.	SELECT MONTH('20200301')	3
YEAR (<i>date</i>)	Returns an integer representing the year part of a specified date.	SELECT YEAR('20190301')	2019
DATEDIFF (<i>datepart, startdate, enddate</i>)	Returns the number of days or time datepart boundaries, crossed between two specified dates.	SELECT DATEDIFF(day,'20201201','2020 1230')	29
DATEADD (<i>datepart, number, date</i>)	Returns a new datetime value by adding an interval to the specified <i>datepart</i> of the specified <i>date</i> .	SELECT DATEADD(day,29,'20201201')	2020-12-30

String or Text Data Types

Text, **String** and **Char** are all used interchangeably when referring to text.

Data Type	Description	Max Size (characters)	Storage (bytes)
Char(n)	REGULAR: Fixed length	8000	1x Defined length
varchar(n)	REGULAR: Variable length	8000	Number of characters + 2
varchar(max)	REGULAR: Variable length	Very Large	Number of characters + 2
text	REGULAR: Variable length	Very Large	Number of characters + 4
nchar(n)	UNICODE: Fixed length, multiple languages	4000	2x Defined length
nvarchar(n)	UNICODE: Variable length, multiple languages	4000	2x number of characters + 2
nvarchar(max)	UNICODE: Variable length, multiple languages	Very Large	2x number of characters + 2
ntext	UNICODE: Variable length, multiple languages	Very Large	2x Defined length
binary(n)	BINARY: Fixed length	8000	Defined length
varbinary(n)	BINARY: Variable length	8000	Defined length + 2
varbinary(max)	BINARY: Variable length	Very Large	Defined length + 2







String Functions

Function	Description	Statement	Result
CONCAT	returns a string resulting from the concatenation, or joining, of two or more string values	SELECT CONCAT('Short, 'sell')	Shortsell
LEFT	Returns the left n characters of a string.	SELECT LEFT('Amazon', 3)	Ama
RIGHT	Returns the right characters of a string.	SELECT RIGHT('Amazon', 3)	zon
REPLACE	Replaces all occurrences of a specified string value with another string	SELECT REPLACE('Buy Stock', 'Buy', 'Sell')	Sell Stock
UPPER	Changes the format to UPPERCASE	SELECT UPPER('futures')	FUTURES
LOWER	Changes the format to LOWERCASE	SELECT LOWER('FUTURES')	futures
LEN	Returns the string length, excluding trailing spaces.	SELECT LEN ('stock ')	5

Comparison Operators

Comparison operators are usually used to determine if a certain condition is **TRUE OR FALSE**.

However, comparisons that involve a NULL return **UNKNOWN**.

	 Equal To	 Greater than	 Less than	 Not equal to	 Greater or equal to	 Less or equal to
Syntax	=	>	<	!= OR <>	>=	<=
Example Code	Date = '2020-08-01'	SUM(SaleAmount) > 25000	SUM(TaxAmt) < 9,000	[ProductName] <> 'Mountain-500'	SUM(SaleAmount) >= 3000	SUM(TaxAmt) <= 2000

Logical Operators

Logical operators allow us to test multiple conditions at once, or to reverse a condition.

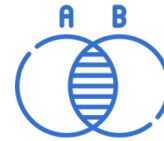


OR

TRUE if either Boolean expression is TRUE

Example

Class <> 'H' OR Class IS NULL



AND

TRUE if both Boolean expressions are TRUE

Size = 'XL' AND Color = 'Black'



NOT

Used to reverse a condition

NOT(Color = 'Black')

Advanced Logical Operators - Simplifying Code



IN

TRUE if the value is equal to any item in a list

Example

Color IN ('Red', 'Blue', 'White')

**Same
as...**

Color = 'Red' OR Color = 'Blue'
OR Color = 'White'



BETWEEN

TRUE if the value is within a range (inclusive).

**SalesAmount BETWEEN 500
AND 1000**

SalesAmount >= 500 AND
SalesAmount <= 1000

Advanced Logical Operators - Partial Matches



LIKE

TRUE if the value matches a specified pattern

Example

**EnglishProductName
LIKE '%BRAKE%'**

WILDCARDS

% represents **any number of characters**
_ represents **one character**

Precedence Among Operators

Earlier, we mentioned that the AND operator is executed before the OR operator.

Here is a **full list of precedence** for operators in SQL.

1. () (Parentheses)
2. * (Multiplication), / (Division), % (Modulo)
3. + (Positive), - (Negative), + (Addition), + (Concatenation), - (Subtraction)
4. =, >, <, >=, <=, <>, !=, !>, !< (Comparison operators)
5. NOT
6. AND
7. BETWEEN, IN, LIKE, OR
8. = (Assignment)