bitfarm® ARCHIV

Dokumentenmanagement

Syntax Zusatzfeld „Grid“ Expressions

Enterprise Version 3.6.2

Inhalt

Konstanten

Konstante	Beschreibung	Beispiel	Erläuterung Beispiel
String-Konstanten	Schreiben Sie String-Konstanten in Apostrophe. Wenn ein String ein Apostroph enthält, verdoppeln Sie das Apostroph.	`[Country] == 'France' or [Name] == 'O''Neil'`	Vergleich eines Textfeldes "Country" mit dem Wert "France". Oder Vergleich des Textfeldes "Name" mit "O'Neil".
Datums- und Zeitkonstanten	Schreiben Sie Datums- und Zeitkonstanten in '#'.	`[OrderDate] >= #2018-03-22 13:18:51.94944#`	Überprüfung, ob ein Bestelldatum ("OrderDate") nach dem 22. März 2018 liegt.
True	Steht für den booleschen Wert "Wahr".	`[InStock] == True`	Prüfung, ob ein Artikel auf Lager ist ("InStock").
False	Steht für den booleschen Wert "Falsch".	`[InStock] == False`	Prüfung, ob ein Artikel *nicht* auf Lager ist.

Konstante	Beschreibung	Beispiel	Erläuterung Beispiel
Enumeration	Geben Sie einen Aufzählungswert mit seinem zugrunde liegenden Integer-Wert an. Sie können einen Aufzählungswert nicht mit seinem qualifizierten Namen angeben.	`[Status] == 1`	Überprüfung, ob ein Statusfeld den Wert 1 hat (z.B. 1 für "Aktiv").
Guid	Schreiben Sie eine Guid-Konstante in geschweifte Klammern. Verwenden Sie Guid-Konstanten in einer relationalen Operation nur mit Gleichheits- oder Ungleichheitsoperatoren.	`[OrderID] == {513724e5-17b7-4ec6-abc4-0eae12c72c1f}`	Vergleich einer eindeutigen Bestellnummer ("OrderID") mit einer bestimmten GUID.
Numerisch	Geben Sie verschiedene numerische Konstantentypen in String-Form mit Suffixen an: Int32 (int) - `1`, Int16 (short) - `1s`, Byte (byte) - `1b`, Double (double) - `1.0`, Single (float) - `1.0f`, Decimal (decimal) - `1.0m`	`[Price] == 25.0m`	Vergleich eines Preisfeldes ("Price") mit dem Dezimalwert 25.0.

Konstante	Beschreibung	Beispiel	Erläuterung Beispiel
?	Steht für einen Null-Verweis, der sich auf kein Objekt bezieht. Es wird empfohlen, stattdessen den unären Operator `IsNull` (z. B. `"[Region] is null"`) oder die logische Funktion `IsNull` (z. B. `"IsNull([Region])"`) zu verwenden.	`[Region] != ?`	Prüfung, ob ein Feld "Region" *nicht* null ist (veraltet, `IsNull` ist besser).

Operatoren

Operator	Beschreibung	Beispiel	Erläuterung Beispiel
+	Addiert den Wert eines numerischen Ausdrucks zu einem anderen oder verknüpft zwei Strings.	`[UnitPrice] + 4 or [FirstName] + ' ' + [LastName]`	Berechnung eines Gesamtpreises durch Addition von 4 zum Einzelpreis. Oder: Kombination von Vor- und Nachname zu einem vollständigen Namen.
-	Findet die Differenz zwischen zwei Zahlen.	`[Price1] - [Price2]`	Berechnung der Preisdifferenz zwischen zwei Produkten.
*	Multipliziert den Wert zweier Ausdrücke.	`[Quantity] * [UnitPrice] * (1 - [BonusAmount])`	Berechnung des Gesamtpreises unter Berücksichtigung von Menge, Einzelpreis und einem prozentualen Rabatt.
/	Dividiert den ersten Operanden durch den zweiten.	`[Quantity] / 2`	Berechnung der Hälfte einer Menge.

Operator	Beschreibung	Beispiel	Erläuterung Beispiel
%	Gibt den Rest (Modulus) zurück, der durch Dividieren eines numerischen Ausdrucks durch einen anderen entsteht.	`[Quantity] % 3`	Ermittlung, ob eine Menge durch 3 teilbar ist (Rest = 0) oder nicht.
|	Führt ein bitweises inklusives ODER für zwei numerische Ausdrücke aus.	`[Flag1]`	Wird selten in Dashboard-Ausdrücken verwendet, eher in systemnaher Programmierung.
&	Der bitweise UND-Operator.	`[Flag] & 10`	Wird selten in Dashboard-Ausdrücken verwendet, eher in systemnaher Programmierung.
^	Führt ein bitweises exklusives ODER für zwei numerische Ausdrücke aus.	`[Flag1] ^ [Flag2]`	Wird selten in Dashboard-Ausdrücken verwendet, eher in systemnaher Programmierung.
==	Gibt "wahr" zurück, wenn beide Operanden denselben Wert haben.	`[Quantity] == 10`	Prüfung, ob eine Menge gleich 10 ist.

Operator	Beschreibung	Beispiel	Erläuterung Beispiel
=	Gibt "wahr" zurück, wenn beide Operanden denselben Wert haben. (Identisch zu ==)	`[Quantity] = 10`	Prüfung, ob eine Menge gleich 10 ist.
!=	Gibt "wahr" zurück, wenn die Operanden nicht denselben Wert haben.	`[Country] != 'France'`	Prüfung, ob ein Land *nicht* Frankreich ist.
<	Kleiner-als-Operator.	`[UnitPrice] < 20`	Prüfung, ob ein Einzelpreis unter 20 liegt.
<=	Kleiner-gleich-Operator.	`[UnitPrice] <= 20`	Prüfung, ob ein Einzelpreis kleiner oder gleich 20 ist.
>=	Größer-gleich-Operator.	`[UnitPrice] >= 30`	Prüfung, ob ein Einzelpreis größer oder gleich 30 ist.
>	Größer-als-Operator.	`[UnitPrice] > 30`	Prüfung, ob ein Einzelpreis über 30 liegt.

Operator	Beschreibung	Beispiel	Erläuterung Beispiel
In („,)	Prüft, ob eine Eigenschaft in einer Liste von Werten enthalten ist.	`[Country] In ('USA', 'UK', 'Italy')`	Prüfung, ob ein Land in einer Liste von Ländern (USA, UK, Italien) enthalten ist.
Between	Gibt einen zu testenden Bereich an.	`[Quantity] Between (10, 20)`	Prüfung, ob eine Menge zwischen 10 und 20 (einschließlich) liegt.
And	Führt eine logische Konjunktion für zwei boolesche Ausdrücke aus.	`[InStock] And ([Extended-Price] > 100)`	Prüfung, ob ein Artikel *sowohl* auf Lager ist *als auch* der Gesamtpreis über 100 liegt.
&&	Führt eine logische Konjunktion aus. (Identisch zu And)	`[InStock] && ([Extended-Price] > 100)`	Prüfung, ob ein Artikel auf Lager ist und der Gesamtpreis über 100 liegt.
Or	Führt eine logische Disjunktion für zwei boolesche Ausdrücke aus.	`[Country] == 'USA' Or [Country] == 'UK'`	Prüfung, ob ein Land entweder USA *oder* UK ist.
|	Führt eine logische Disjunktion aus. (Identisch zu Or)	`[Country] == 'USA' | [Country] == 'UK'`	Prüfung, ob ein Land entweder USA oder UK ist.

Operator	Beschreibung	Beispiel	Erläuterung Beispiel
~	Führt eine bitweise Negation aus.	`~[Roles] = 251`	Wird selten in Dashboard-Ausdrücken verwendet, eher in systemnaher Programmierung.
Not	Führt eine logische Negation für einen booleschen Ausdruck aus.	`Not [InStock]`	Prüft, ob ein Artikel *nicht* auf Lager ist (kehrt den Wert von [InStock] um).
!	Führt eine logische Negation aus. (Identisch zu Not)	`![InStock]`	Prüft, ob ein Artikel *nicht* auf Lager ist.
+	Gibt den Wert eines numerischen Ausdrucks zurück (unärer Operator).	`+[Value] = 10`	Kann verwendet werden, um die Positivität einer Zahl zu betonen, hat aber meist keine praktische Auswirkung.
-	Gibt den negativen Wert eines numerischen Ausdrucks zurück (unärer Operator).	`-[Value] = 20`	Macht aus einer positiven Zahl eine negative Zahl.
Is Null	Gibt "wahr" zurück, wenn ein Ausdruck ein Null-Verweis ist.	`[Region] is null`	Prüft, ob ein Feld "Region" keinen Wert enthält.

Funktionen

Erweiterte Funktionen - Intermediate Aggregation Level

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`aggr(SummaryExpression, Dimensions)`	Verwendet die Detailebene, die durch einen vordefinierten Satz von Dimensionen und einer angegebenen Zusammenfassungsfunktion spezifiziert wird, um zugrunde liegende Daten zu aggregieren.	`aggr(Sum([Sales]),[Category],[Product])`	Berechnet die Summe der Umsätze ("Sales") für jede Kombination aus Kategorie ("Category") und Produkt ("Product").
`w(WindowExpression, partitionBy-Funktion, orderBy-Funktion)`	Berechnet aggregierte Werte mit der angegebenen Fensterfunktion für das Fenster, das durch die angegebene Partitionierung und Sortierung definiert ist.	`w(RankDense(Sum([ProductSales]),'desc'), partitionBy([Category-Name]),orderBy())`	Berechnet einen dichten Rang der Produktverkäufe ("ProductSales") innerhalb jeder Kategorie ("CategoryName"), absteigend sortiert.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`partitionBy(column1, column2, ...)`	Gibt die Spalten an, nach denen die Datenzeilen partitioniert werden. Die Fensterfunktion wird auf jede Partition separat angewendet. Die `partitionBy` -Funktion kann nur als Argument der `w` -Funktion verwendet werden.	`partitionBy([Product])`	Teilt die Daten nach Produkten auf, sodass Fensterfunktionen (z.B. Rang) innerhalb jedes Produkts separat berechnet werden.
`orderBy(column1, column2,...)`	Bestimmt die logische Reihenfolge, in der die Berechnung der Fensterfunktion in den Zeilen des Fensters durchgeführt wird. Die `orderBy` Funktion kann nur als Argument der `w` -Funktion verwendet werden.	`orderBy(GetYear([Date]),desc(Sum([Sales])))`	Innerhalb eines Fensters (z.B. definiert durch `partitionBy`) werden die Daten zuerst nach dem Jahr des Datums und dann absteigend nach der Summe der Umsätze sortiert.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`asc(column)`	Legt fest, dass die Werte in der angegebenen Spalte in aufsteigender Reihenfolge sortiert sind. Dies ist die Standardsortierreihenfolge. Die `asc` Funktion kann nur als Argument der `w` -Funktion verwendet werden.	`asc(Sum([Sales]))`	Innerhalb eines Fensters werden die Daten nach der Summe der Umsätze in aufsteigender Reihenfolge sortiert.
`desc(column)`	Legt fest, dass die Werte in der angegebenen Spalte in absteigender Reihenfolge sortiert sind. Die `desc` Funktion kann nur als Argument der `w` -Funktion verwendet werden.	`desc(Sum([Sales]))`	Innerhalb eines Fensters werden die Daten nach der Summe der Umsätze in *absteigender* Reihenfolge sortiert.

Aggregatfunktionen

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Avg(Value)`	Gibt den Durchschnitt aller Werte im Ausdruck zurück.	`Avg([Profit])`	Berechnung des durchschnittlichen Gewinns ("Profit") über alle Datensätze.
`Count()`	Gibt die Anzahl der Werte zurück.	`Count()`	Zählt alle Datensätze (Zeilen) in der Datenquelle.
`CountNotNull(Value)`	Gibt die Anzahl der Nicht-Null-Objekte in einer Sammlung zurück.	`CountNotNull([Orders])`	Zählt alle Datensätze, in denen das Feld "Orders" einen Wert hat (nicht NULL ist).
`CountDistinct(Value)`	Gibt die Anzahl der eindeutigen Werte zurück.	`CountDistinct([Orders])`	Zählt die Anzahl der *verschiedenen* Bestellungen (entfernt Duplikate).
`Max(Value)`	Gibt den Maximalwert über alle Datensätze zurück.	`Max([Profit])`	Findet den höchsten Gewinnwert in allen Datensätzen.
`Min(Value)`	Gibt den Minimalwert über alle Datensätze zurück.	`Min([Profit])`	Findet den niedrigsten Gewinnwert in allen Datensätzen.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Mode(Value)`	Gibt den Modus der Werte zurück.	`Mode([Profit])`	Findet den am häufigsten vorkommenden Gewinnwert.
`Median(Value)`	Gibt den Median der Werte zurück.	`Median([Profit])`	Findet den mittleren Gewinnwert (50% der Werte sind größer, 50% sind kleiner).
`Sum(Value)`	Gibt die Summe aller Werte zurück.	`Sum([Profit])`	Berechnet die Summe aller Gewinne.
`Var(Value)`	Gibt eine Schätzung der Varianz einer Population zurück, wobei die Stichprobe eine Teilmenge der Gesamtpopulation ist.	`Var([Orders])`	Berechnet die Varianz der Bestellmengen (wie stark die Bestellmengen streuen). Da es eine Schätzung ist, wird angenommen, dass die Daten eine Stichprobe sind.
`Varp(Value)`	Gibt die Varianz einer Population zurück, wobei die Population die gesamten zu summierenden Daten sind.	`Varp([Orders])`	Berechnet die Varianz der Bestellmengen. Im Gegensatz zu `Var` wird angenommen, dass die Daten die *gesamte* Population darstellen.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`StdDev(Value)`	Gibt eine Schätzung der Standardabweichung einer Population zurück, wobei die Stichprobe eine Teilmenge der Gesamtpopulation ist.	`StdDev([Orders])`	Berechnet die Standardabweichung der Bestellmengen (ein Maß für die Streuung um den Mittelwert). Wie bei `Var` wird eine Stichprobe angenommen.
`StdDevp(Value)`	Gibt die Standardabweichung einer Population zurück, wobei die Population die gesamten zu summierenden Daten sind.	`StdDevp([Orders])`	Berechnet die Standardabweichung der Bestellmengen, wobei angenommen wird, dass die Daten die gesamte Population darstellen.

Fensterfunktionen

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Last()`	Gibt die Anzahl der Zeilen von der aktuellen Zeile bis zur letzten Zeile im Fenster zurück.	`Last()`	Innerhalb eines Fensters (z.B. alle Bestellungen eines Monats) gibt `Last()` die Anzahl der verbleibenden Bestellungen bis zum Monatsende zurück.
`First()`	Gibt die Anzahl der Zeilen von der aktuellen Zeile bis zur ersten Zeile im Fenster zurück.	`First()`	Innerhalb eines Fensters gibt `First()` die Anzahl der Zeilen von der aktuellen Zeile bis zum Anfang des Fensters zurück (beginnend bei 0).
`Index()`	Gibt den Index der aktuellen Zeile im Fenster zurück.	`Index()`	Gibt die Position der aktuellen Zeile innerhalb des Fensters zurück (beginnend bei 0).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Size()`	Gibt die Anzahl der Zeilen im Fenster zurück.	`Size()`	Gibt die Gesamtanzahl der Zeilen innerhalb des aktuellen Fensters zurück (z.B. die Anzahl aller Bestellungen in einem Monat).
`Lookup(SummaryExpression, Position)`	Gibt den Wert des Ausdrucks in einer Zielposition zurück, die als relativer Offset von der aktuellen Position angegeben ist.	`Lookup(Sum([Sales]), 3)`	Ruft die Summe der Umsätze ("Sales") von der Zeile ab, die 3 Positionen *nach* der aktuellen Zeile liegt (innerhalb des Fensters).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`RankCompetition(SummaryExpression, ['asc' | 'desc'])`	Gibt den Standard-Wettbewerbsrang für die aktuelle Zeile im Fenster zurück.	`RankCompetition(Sum([Sales]), 'desc')`	Weist jeder Zeile (z. B. jedem Verkäufer) einen Rang basierend auf der Summe der Umsätze zu. Bei gleichen Umsätzen erhalten Verkäufer den *gleichen* Rang (z.B. 1, 2, 2, 4...). 'desc' bedeutet absteigende Sortierung (höchster Umsatz = Rang 1).
`RankDense(SummaryExpression, ['asc' | 'desc'])`	Gibt den dichten Rang für die aktuelle Zeile im Fenster zurück.	`RankDense(Sum([Sales]), 'desc')`	Ähnlich wie `RankCompetition`, aber bei gleichen Werten werden *keine* Ränge übersprungen (z.B. 1, 2, 2, 3...).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`RankUnique(SummaryExpression, ['asc' | 'desc'])`	Gibt den eindeutigen Rang für die aktuelle Zeile im Fenster zurück.	`RankUnique(Sum([Sales]), 'desc')`	Jeder Zeile wird ein eindeutiger Rang zugewiesen, selbst wenn die Werte gleich sind. Die Reihenfolge bei gleichen Werten ist nicht definiert.
`RankModified(SummaryExpression, ['asc' | 'desc'])`	Gibt den modifizierten Wettbewerbsrang für die aktuelle Zeile im Fenster zurück.	`RankModified(Sum([Sales]), 'desc')`	Eine Variante von `RankCompetition`, bei der der Rang bei gleichen Werten dem *niedrigsten* gemeinsamen Rang entspricht (z.B. 1, 3, 3, 4...).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`RankPercentile(SummaryExpression, ['asc' | 'desc'])`	Gibt den Perzentilrang für die aktuelle Zeile im Fenster zurück.	`RankPercentile(Sum([Sales]), 'desc')`	Gibt an, in welchem Perzentil sich die aktuelle Zeile befindet (z.B. 0.9 bedeutet, dass 90% der Werte kleiner oder gleich dem aktuellen Wert sind).
`RunningAvg(SummaryExpression)`	Gibt den gleitenden Durchschnitt des angegebenen Ausdrucks von der ersten Zeile im Fenster bis zur aktuellen Zeile zurück.	`RunningAvg(Sum([Sales]))`	Berechnet den laufenden Durchschnitt der Umsätze. Für jede Zeile wird der Durchschnitt der Umsätze von der ersten Zeile bis zur *aktuellen* Zeile berechnet.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`RunningCount(SummaryExpression)`	Gibt die laufende Anzahl des angegebenen Ausdrucks von der ersten Zeile im Fenster bis zur aktuellen Zeile zurück.	`RunningCount(Sum([Sales]))`	Berechnet die laufende Anzahl der Umsätze. Für jede Zeile wird gezählt, wie viele Umsatzwerte bis zu dieser Zeile aufgetreten sind.
`RunningMax(SummaryExpression)`	Gibt das laufende Maximum des angegebenen Ausdrucks von der ersten Zeile im Fenster bis zur aktuellen Zeile zurück.	`RunningMax(Sum([Sales]))`	Berechnet das laufende Maximum der Umsätze. Für jede Zeile wird der höchste Umsatzwert von der ersten Zeile bis zur aktuellen Zeile ermittelt.
`RunningMin(SummaryExpression)`	Gibt das laufende Minimum des angegebenen Ausdrucks von der ersten Zeile im Fenster bis zur aktuellen Zeile zurück.	`RunningMin(Sum([Sales]))`	Berechnet das laufende Minimum der Umsätze von der ersten Zeile bis zur aktuellen Zeile.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`RunningSum(SummaryExpression)`	Gibt die laufende Summe des angegebenen Ausdrucks von der ersten Zeile im Fenster bis zur aktuellen Zeile zurück.	`RunningSum(Sum([Sales]))`	Berechnet die laufende Summe der Umsätze von der ersten Zeile bis zur aktuellen Zeile.
`WindowAvg(SummaryExpression, StartOffset, EndOffset)`	Gibt den Durchschnitt des Ausdrucks innerhalb des Fensters zurück, das mithilfe von Offsets von der aktuellen Zeile definiert wird.	`WindowAvg(Sum([Sales]), First(), Last())`	Berechnet den Durchschnitt der Umsätze über das *gesamte* Fenster (von `First()` bis `Last()`). Im Gegensatz zu `RunningAvg` ist das Ergebnis für alle Zeilen im Fenster gleich.
`WindowCountDistinct(SummaryExpression, StartOffset, EndOffset)`	Gibt die eindeutige Anzahl des Ausdrucks innerhalb des Fensters zurück.	`WindowCountDistinct(Sum([Sales]),First(), Last())`	Zählt die Anzahl der *unterschiedlichen* Umsatzwerte innerhalb des gesamten Fensters.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`WindowMax(SummaryExpression, StartOffset, EndOffset)`	Gibt das Maximum des Ausdrucks innerhalb des Fensters zurück.	`WindowMax(Sum([Sales]), First(), Last())`	Findet den höchsten Umsatzwert innerhalb des gesamten Fensters.
`WindowMin(SummaryExpression, StartOffset, EndOffset)`	Gibt das Minimum des Ausdrucks innerhalb des Fensters zurück.	`WindowMin(Sum([Sales]), First(), Last())`	Findet den niedrigsten Umsatzwert innerhalb des gesamten Fensters.
`WindowMode(SummaryExpression, StartOffset, EndOffset)`	Gibt den Modus des Ausdrucks innerhalb des Fensters zurück.	`WindowMode(Sum([Sales]), First(), Last())`	Findet den am häufigsten vorkommenden Umsatzwert innerhalb des gesamten Fensters.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`WindowMedian(SummaryExpression, StartOffset, EndOffset)`	Gibt den Median des Ausdrucks innerhalb des Fensters zurück.	`WindowMedian(Sum([Sales]),First(), Last())`	Findet den mittleren Umsatzwert (Median) innerhalb des gesamten Fensters.
`WindowSum(SummaryExpression, StartOffset, EndOffset)`	Gibt die Summe des Ausdrucks innerhalb des Fensters zurück.	`WindowSum(Sum([Sales]),First()+2, Last())`	Berechnet die Summe der Umsätze innerhalb des definierten Fensters (hier von `First()+2` bis `Last()`).
`WindowVar(SummaryExpression, StartOffset, EndOffset)`	Gibt die Varianz des Ausdrucks innerhalb des Fensters zurück.	`WindowVar(Sum([Sales]), First(), Last())`	Berechnet die Varianz der Umsätze (Streuung) innerhalb des gesamten Fensters.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`WindowVarp(SummaryExpression, StartOffset, EndOffset)`	Gibt die voreingenommene Varianz des Ausdrucks innerhalb des Fensters zurück.	`WindowVarp(Sum([Sales]), First(), Last())`	Berechnet die *bevölkerungsbezogene* Varianz der Umsätze innerhalb des gesamten Fensters (unterscheidet sich leicht von `WindowVar`).
`WindowStdDev(SummaryExpression, StartOffset, EndOffset)`	Gibt die Standardabweichung der Stichprobe des Ausdrucks innerhalb des Fensters zurück.	`WindowStdDev(Sum([Sales]),First(), Last())`	Berechnet die Standardabweichung der Umsätze (Streuung um den Mittelwert) innerhalb des gesamten Fensters, basierend auf einer Stichprobe.
`WindowStdDevp(SummaryExpression, StartOffset, EndOffset)`	Gibt die voreingenommene Standardabweichung des Ausdrucks innerhalb des Fensters zurück.	`WindowStdDevp(Sum([Sales]),First(), Last())`	Berechnet die *bevölkerungsbezogene* Standardabweichung der Umsätze innerhalb des gesamten Fensters.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Total(Summa-ryExpression)`	Gibt die Gesamtsumme basierend auf den Werten aus der zugrunde liegenden Datenquelle für den angegebenen Ausdruck in einem Berechnungsfenster zurück.	`To-tal(Sum([Sa-les]))`	Berechnet die Gesamtsumme der Umsätze *über alle Daten*, unabhängig von Fenstern oder Partitionen.

WICHTIG
Beachten Sie, dass Fensterfunktionen nicht innerhalb von Aggregationen verwendet werden können.

Datums- und Zeitfunktionen

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`AddDays(DateTime, DaysCount)`	Gibt einen Datum-Uhrzeit-Wert zurück, der der angegebenen Anzahl von Tagen ab dem angegebenen DateTime-Wert entspricht.	`AddDays([OrderDate], 30)`	Berechnet das Datum, das 30 Tage nach dem Bestelldatum liegt (z.B. für ein Zahlungsziel).
`AddHours(DateTime, HoursCount)`	Gibt einen Datum-Uhrzeit-Wert zurück, der der angegebenen Anzahl von Stunden ab dem angegebenen DateTime-Wert entspricht.	`AddHours([StartTime], 2)`	Berechnet die Uhrzeit, die 2 Stunden nach einer Startzeit liegt.
`AddMilliSeconds(DateTime, MilliSecondsCount)`	Gibt einen Datum-Uhrzeit-Wert zurück, der der angegebenen Anzahl von Millisekunden ab dem angegebenen DateTime-Wert entspricht.	`AddMilliSeconds([StartTime], 5000))`	Berechnet die Zeit, die 5000 Millisekunden (5 Sekunden) nach einer Startzeit liegt. Wird selten direkt in Dashboards verwendet.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`AddMinutes(DateTime, MinutesCount)`	Gibt einen Datum-Uhrzeit-Wert zurück, der der angegebenen Anzahl von Minuten ab dem angegebenen DateTime-Wert entspricht.	`AddMinutes([StartTime], 30)`	Berechnet die Uhrzeit, die 30 Minuten nach einer Startzeit liegt (z.B. für eine Meeting-Dauer).
`AddMonths(DateTime, MonthsCount)`	Gibt einen Datum-Uhrzeit-Wert zurück, der der angegebenen Anzahl von Monaten ab dem angegebenen DateTime-Wert entspricht.	`AddMonths([OrderDate], 1)`	Berechnet das Datum, das einen Monat nach dem Bestelldatum liegt (z.B. für eine monatliche Rechnung).
`AddSeconds(DateTime, SecondsCount)`	Gibt einen Datums-/Uhrzeitwert zurück, der der angegebenen Anzahl von Sekunden ab dem angegebenen DateTime-Wert entspricht.	`AddSeconds([StartTime], 60)`	Berechnet die Zeit, die 60 Sekunden (1 Minute) nach einer Startzeit liegt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`AddTicks(DateTime, TicksCount)`	Gibt einen Datums-/Uhrzeitwert zurück, der der angegebenen Anzahl von Ticks ab dem angegebenen DateTime-Wert entspricht.	`AddTicks([StartTime], 5000)`	Wird selten direkt in Dashboards verwendet, da Ticks sehr feine Zeiteinheiten sind (1 Tick = 100 Nanosekunden).
`AddTimeSpan(DateTime, TimeSpan)`	Gibt einen Datums-/Uhrzeitwert zurück, der vom angegebenen DateTime-Wert für die angegebene TimeSpan-Zeitspanne gilt.	`AddTimeSpan([StartTime], [Duration])`	Addiert eine Zeitspanne (gespeichert in der Variable `[Duration]`) zu einer Startzeit. `[Duration]` muss ein TimeSpan-Objekt sein.
`AddYears(DateTime, YearsCount)`	Gibt einen Datums-/Uhrzeitwert zurück, der der angegebenen Anzahl von Jahren ab dem angegebenen DateTime-Wert entspricht.	`AddYears([EndDate], -1)`	Berechnet das Datum, das ein Jahr *vor* einem Enddatum liegt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`DateDiffDay(startDate, endDate)`	Gibt die Anzahl der Tagesgrenzen zwischen zwei nicht auf Null setzbaren Daten zurück.	`DateDiffDay([StartTime], Now())`	Berechnet die Anzahl der *vollen Tage* zwischen einer Startzeit und dem aktuellen Datum/Uhrzeit (`Now()`).
`DateDiffHour(startDate, endDate)`	Gibt die Anzahl der Stundengrenzen zwischen zwei Daten zurück, die nicht auf Null gesetzt werden können.	`DateDiffHour([StartTime], Now())`	Berechnet die Anzahl der *vollen Stunden* zwischen einer Startzeit und jetzt.
`DateDiffMiliSecond(startDate, endDate)`	Gibt die Anzahl der Millisekundengrenzen zwischen zwei nicht auf Null setzbaren Daten zurück.	`DateDiffMiliSecond([StartTime], Now())`	Berechnet die Anzahl der Millisekunden zwischen zwei Zeitpunkten. Selten direkt in Dashboards verwendet.
`DateDiffMinute(startDate, endDate)`	Gibt die Anzahl der Minutengrenzen zwischen zwei Daten zurück, die nicht auf Null gesetzt werden können.	`DateDiffMinute([StartTime], Now())`	Berechnet die Anzahl der *vollen Minuten* zwischen einer Startzeit und jetzt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`DateDiffMonth(startDate, endDate)`	Gibt die Anzahl der Monatsgrenzen zwischen zwei Daten zurück, die nicht auf Null gesetzt werden können.	`DateDiffMonth([StartTime], Now())`	Berechnet die Anzahl der *vollen Monate* zwischen einer Startzeit und jetzt.
`DateDiffSecond(startDate, endDate)`	Gibt die Anzahl der Sekundengrenzen zwischen zwei Daten zurück, die nicht auf Null gesetzt werden können.	`DateDiffSecond([StartTime], Now())`	Berechnet die Anzahl der *vollen Sekunden* zwischen einer Startzeit und jetzt.
`DateDiffTick(startDate, endDate)`	Gibt die Anzahl der Tickgrenzen zwischen zwei Daten zurück, die nicht auf Null gesetzt werden können.	`DateDiffTick([StartTime], Now())`	Berechnet die Anzahl der Ticks (100-Nanosekunden-Intervalle) zwischen zwei Zeitpunkten. Selten direkt in Dashboards verwendet.
`DateDifYear(startDate, endDate)`	Gibt die Anzahl der Jahresgrenzen zwischen zwei Daten zurück, die nicht auf Null gesetzt werden können.	`DateDifYear([StartTime], Now())`	Berechnet die Anzahl der *vollen Jahre* zwischen einer Startzeit und jetzt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`GetDate(DateTime)`	Extrahiert ein Datum aus dem definierten DateTime.	`GetDate([OrderDateTime])`	Extrahiert den Datumsteil (Jahr, Monat, Tag) aus einem DateTime-Wert, der Datum und Uhrzeit enthält.
`GetDateHour(DateTime)`	Extrahiert den Datumsteil mit dem Stundenwert aus dem definierten DateTime.	`GetDateHour([OrderDate])`	Extrahiert Jahr, Monat, Tag und Stunde aus einem DateTime-Wert. Die Minuten und Sekunden werden auf 0 gesetzt.
`GetDateHourMinute(DateTime)`	Extrahiert den Datumsteil mit den Stunden- und Minutenwerten aus dem definierten DateTime.	`GetDateHourMinute([OrderDate])`	Extrahiert Jahr, Monat, Tag, Stunde und Minute aus einem DateTime-Wert. Die Sekunden werden auf 0 gesetzt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`GetDateHourMinuteSecond(DateTime)`	Extrahiert den Datumsteil mit den Stunden-, Minuten- und Sekundenwerten aus dem definierten DateTime.	`GetDateHourMinuteSecond([OrderDate])`	Extrahiert Jahr, Monat, Tag, Stunde, Minute *und* Sekunde aus einem DateTime-Wert.
`GetDateMonthYear(DateTime)`	Extrahiert das Datum mit dem Monat und Jahr aus dem definierten DateTime.	`GetDateMonthYear([OrderDate])`	Extrahiert Jahr und Monat aus einem DateTime-Wert. Der Tag wird auf 1 und die Uhrzeit auf 00:00:00 gesetzt.
`GetDateQuarterYear(DateTime)`	Extrahiert das Datum mit dem Quartal und Jahr aus dem definierten DateTime.	`GetDateQuarterYear([OrderDate])`	Extrahiert das Jahr und das Quartal (1, 2, 3 oder 4) aus einem DateTime-Wert.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`GetDateWeekYear()`	Gibt das Datum des ersten Tages der Woche für einen angegebenen DateTime-Wert zurück (verwendet Kultureinstellungen).	`GetDateHourMinuteSecond([OrderDate])`	Ermittelt den ersten Tag der Woche (abhängig von den regionalen Einstellungen, oft Montag oder Sonntag), in der ein bestimmtes Datum liegt.
`GetDay(DateTime)`	Extrahiert einen Tag aus dem definierten DateTime.	`GetDay([OrderDate])`	Extrahiert den Tag des Monats (1-31) aus einem DateTime-Wert.
`GetDayOfWeek(DateTime)`	Extrahiert einen Wochentag aus dem definierten DateTime.	`GetDayOfWeek([OrderDate])`	Extrahiert den Wochentag (z.B. als Zahl, wobei die Zuordnung von Zahlen zu Wochentagen von den regionalen Einstellungen abhängt).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`GetDayOfYear(DateTime)`	Extrahiert einen Tag des Jahres aus dem definierten DateTime.	`GetDayOfYear([OrderDate])`	Extrahiert den Tag des Jahres (1-366) aus einem DateTime-Wert.
`GetHour(DateTime)`	Extrahiert eine Stunde aus dem definierten DateTime.	`GetHour([StartTime])`	Extrahiert die Stunde (0-23) aus einem DateTime-Wert.
`GetMilliSecond(DateTime)`	Extrahiert Millisekunden aus dem definierten DateTime.	`GetMilliSecond([StartTime])`	Extrahiert die Millisekunden (0-999) aus einem DateTime-Wert.
`GetMinute(DateTime)`	Extrahiert Minuten aus dem definierten DateTime.	`GetMinute([StartTime])`	Extrahiert die Minute (0-59) aus einem DateTime-Wert.
`GetMonth(DateTime)`	Extrahiert einen Monat aus dem definierten DateTime.	`GetMonth([StartTime])`	Extrahiert den Monat (1-12) aus einem DateTime-Wert.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`GetSecond(DateTime)`	Extrahiert Sekunden aus dem definierten DateTime.	`GetSecond([StartTime])`	Extrahiert die Sekunde (0-59) aus einem DateTime-Wert.
`GetTimeOfDay(DateTime)`	Extrahiert die Tageszeit aus dem definierten DateTime in Ticks.	`GetTimeOfDay([StartTime])`	Gibt die Anzahl der Ticks seit Mitternacht (00:00:00) zurück.
`GetWeekOfMonth(DateTime)`	Extrahiert die Woche des Monats aus dem definierten DateTime.	`GetWeekOfMonth([OrderDate])`	Ermittelt, die wievielte Woche des Monats ein bestimmtes Datum ist (die genaue Definition, was eine "Woche des Monats" ist, kann variieren).
`GetWeekOfYear(DateTime)`	Extrahiert die Woche des Jahres aus dem definierten DateTime.	`GetWeekOfYear([OrderDate])`	Ermittelt die Kalenderwoche (ISO-Woche) eines Datums.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`GetYear(DateTime)`	Extrahiert ein Jahr aus dem definierten DateTime.	`GetYear([StartTime])`	Extrahiert das Jahr aus einem DateTime-Wert.
`IsApril(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den April fällt.	`IsApril([OrderDate])`	Prüft, ob ein Datum im April liegt.
`IsAugust(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den August fällt.	`IsAugust([OrderDate])`	Prüft, ob ein Datum im August liegt.
`IsDecember(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den Dezember fällt.	`IsDecember([OrderDate])`	Prüft, ob ein Datum im Dezember liegt.
`IsFebruary(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den Februar fällt.	`IsFebruary([OrderDate])`	Prüft, ob ein Datum im Februar liegt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`IsJanuary(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den Januar fällt.	`IsJanuary([OrderDate])`	Prüft, ob ein Datum im Januar liegt.
`IsJuly(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den Juli fällt.	`IsJuly([OrderDate])`	Prüft, ob ein Datum im Juli liegt.
`IsJune(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den Juni fällt.	`IsJune([OrderDate])`	Prüft, ob ein Datum im Juni liegt.
`IsLastMonth(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den vorherigen Monat fällt.	`IsLastMonth([OrderDate])`	Prüft, ob ein Datum im Vormonat liegt.
`IsLastYear(DateTime)`	Gibt True zurück, wenn das angegebene Datum in das vorherige Jahr fällt.	`IsLastYear([OrderDate])`	Prüft, ob ein Datum im Vorjahr liegt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`IsMarch(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den März fällt.	`IsMarch([OrderDate])`	Prüft, ob ein Datum im März liegt.
`IsMay(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den Mai fällt.	`IsMay([OrderDate])`	Prüft, ob ein Datum im Mai liegt.
`IsNextMonth(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den nächsten Monat fällt.	`IsNextMonth([OrderDate])`	Prüft, ob ein Datum im nächsten Monat liegt.
`IsNextYear(DateTime)`	Gibt True zurück, wenn das angegebene Datum in das nächste Jahr fällt.	`IsNextYear([OrderDate])`	Prüft, ob ein Datum im nächsten Jahr liegt.
`IsNovember(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den November fällt.	`IsNovember([OrderDate])`	Prüft, ob ein Datum im November liegt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`IsOctober(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den Oktober fällt.	`IsOctober([OrderDate])`	Prüft, ob ein Datum im Oktober liegt.
`IsSameDay(DateTime)`	Gibt True zurück, wenn die angegebenen Datum/Zeit-Werte auf denselben Tag fallen.	`IsSameDay([OrderDate])`	Prüft, ob zwei DateTime-Werte am gleichen Tag (unabhängig von der Uhrzeit) liegen.
`IsSeptember(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den September fällt.	`IsSeptember([OrderDate])`	Prüft, ob ein Datum im September liegt.
`IsThisMonth(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den aktuellen Monat fällt.	`IsThisMonth([OrderDate])`	Prüft, ob ein Datum im aktuellen Monat liegt.
`IsThisWeek(DateTime)`	Gibt True zurück, wenn das angegebene Datum in die aktuelle Woche fällt.	`IsThisWeek([OrderDate])`	Prüft, ob ein Datum in der aktuellen Woche liegt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`IsYearToDate(DateTime)`	Gibt True zurück, wenn das angegebene Datum in den Zeitraum vom Jahresbeginn bis zum heutigen Tag fällt. Dieser Zeitraum beginnt am ersten Tag des laufenden Jahres und geht bis zum aktuellen Datum (einschließlich des aktuellen Datums).	`IsYearToDate([OrderDate])`	Prüft, ob ein Datum im laufenden Jahr liegt (vom 1. Januar bis zum *heutigen* Tag).
`IsThisYear(DateTime)`	Gibt True zurück, wenn das angegebene Datum in das aktuelle Jahr fällt.	`IsThisYear([OrderDate])`	Prüft, ob ein Datum im aktuellen Jahr liegt.
`LocalDateTime-DayAfterTomorrow()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem Tag nach Übermorgen entspricht.	`AddDays(LocalDateTime-DayAfterTomorrow(),5)`	Gibt das Datum von Übermorgen zurück. Im Beispiel wird dann noch 5 Tage addiert.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`LocalDateTimeLastMonth()`	Gibt den DateTime-Wert zurück, der dem ersten Tag des vorherigen Monats entspricht.	`AddMonths(LocalDateTimeLastMonth(),)`	Gibt das Datum des ersten Tages des letzten Monats zurück.
`LocalDateTimeLastWeek()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem ersten Tag der vorherigen Woche entspricht.	`AddDays(LocalDateTimeLastWeek(), 5)`	Gibt das Datum des ersten Tages der letzten Woche zurück. Im Beispiel werden 5 Tage addiert.
`LocalDateTimeLastYear()`	Gibt den DateTime-Wert zurück, der dem ersten Tag des vorherigen Jahres entspricht.	`AddYears(LocalDateTimeLastYear(), 5)`	Gibt das Datum des ersten Tages des letzten Jahres zurück. Im Beispiel: plus 5 Jahre
`LocalDateTimeNextMonth()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem ersten Tag des nächsten Monats entspricht.	`AddMonths(LocalDateTimeNextMonth(),)`	Gibt das Datum des ersten Tages des nächsten Monats zurück.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`LocalDateTimeNextWeek()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem ersten Tag der folgenden Woche entspricht.	`AddDays(LocalDateTimeNextWeek(), 5)`	Gibt das Datum des ersten Tages der nächsten Woche zurück, plus 5 Tage im Beispiel.
`LocalDateTimeNextYear()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem ersten Tag des folgenden Jahres entspricht.	`AddYears(LocalDateTimeNextYear(), 5)`	Gibt das Datum des ersten Tages des nächsten Jahres zurück, plus 5 Jahre.
`LocalDateTimeNow()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem aktuellen Zeitpunkt entspricht.	`AddDays(LocalDateTimeNow(), 5)`	Gibt das aktuelle Datum und die aktuelle Uhrzeit zurück (zum Zeitpunkt der Ausführung). Im Beispiel werden 5 Tage addiert.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`LocalDateTimeThisMonth()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem ersten Tag des aktuellen Monats entspricht.	`AddMonths(LocalDateTimeThisMonth(),)`	Gibt das Datum des ersten Tages des *aktuellen* Monats zurück.
`LocalDateTimeThisWeek()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem ersten Tag der aktuellen Woche entspricht.	`AddDays(LocalDateTimeThisWeek(), 5)`	Gibt das Datum des ersten Tages der *aktuellen* Woche zurück. Plus 5 Tage.
`LocalDateTimeThisYear()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem ersten Tag des aktuellen Jahres entspricht.	`AddDays(LocalDateTimeThisYear(), 5)`	Gibt das Datum des ersten Tages des *aktuellen* Jahres zurück, plus 5 Jahre.
`LocalDateTimeToday()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem heutigen Tag entspricht.	`AddDays(LocalDateTimeToday(), 5)`	Gibt das *heutige* Datum zurück (die Uhrzeit ist auf 00:00:00 gesetzt). Plus 5 Tage.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`LocalDateTimeTomorrow()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem morgigen Tag entspricht.	`AddDays(LocalDateTimeTomorrow(), 5)`	Gibt das Datum von *morgen* zurück (Uhrzeit 00:00:00). Plus 5 Tage.
`LocalDateTimeTwoMonthsAway()`	Gibt den DateTime-Wert zurück, der dem ersten Tag des übernächsten Monats entspricht.	`AddMonths(LocalDateTimeTwoMonthsAw(),5)`	Gibt das Datum des ersten Tages in zwei Monaten zurück. Plus 5 Monate.
`LocalDateTimeTwoWeeksAway()`	Gibt den DateTime-Wert zurück, der dem ersten Tag der übernächsten Woche entspricht.	`AddDays(LocalDateTimeTwoWeeksAway(),)`	Gibt das Datum des ersten Tages in zwei Wochen zurück.
`LocalDateTimeTwoYearsAway()`	Gibt den DateTime-Wert zurück, der dem ersten Tag des übernächsten Jahres entspricht.	`AddYears(LocalDateTimeTwoYearsAway(),)`	Gibt das Datum des ersten Tages in zwei Jahren zurück.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`LocalDateTimeYearBeforeToday()`	Gibt den DateTime-Wert zurück, der dem Tag vor einem Jahr entspricht.	`AddYears(LocalDateTimeYearBeforeToday(),5)`	Gibt das Datum von vor einem Jahr zurück. Im Beispiel plus 5 Jahre.
`LocalDateTimeYesterday()`	Gibt einen Datum-Uhrzeit-Wert zurück, der dem gestrigen Tag entspricht.	`AddDays(LocalDateTimeYesterday(), 5)`	Gibt das Datum von *gestern* zurück (Uhrzeit 00:00:00). Plus 5 Tage.
`MakeDateTime(Year, Month, Day)`	Gibt einen Datumswert zurück, der aus dem angegebenen Jahr, Monat und Tag konstruiert wurde.	`MakeDateTime(2018, 5, 5)`	Erstellt ein Datumsobjekt für den 5. Mai 2018.
`MakeDateTime(2018, 5, 5, 20)`	Erstellt ein Datumsobjekt für den 5. Mai 2018, 20 Uhr.		

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`MakeDateTime(Year, Month, Day, Hour, Minute)`	Gibt einen Datumswert zurück, der aus dem angegebenen Jahr, Monat, Tag, der angegebenen Stunde und Minute konstruiert wurde.	`MakeDateTime(2018, 5, 5, 20, 18)`	Erstellt ein Datumsobjekt für den 5. Mai 2018, 20:18 Uhr.
`MakeDateTime(Year, Month, Day, Hour, Minute, Second)`	Gibt einen Datumswert zurück, der aus dem angegebenen Jahr, Monat, Tag, der angegebenen Stunde, Minute und Sekunde konstruiert wurde.	`MakeDateTime(2018, 5, 5, 20, 18, 30)`	Erstellt ein Datumsobjekt für den 5. Mai 2018, 20:18:30 Uhr.
`Now()`	Gibt das aktuelle Systemdatum und die aktuelle Uhrzeit zurück.	`AddDays(Now(), 5)`	Gibt das aktuelle Datum und die aktuelle Uhrzeit zurück (zum Zeitpunkt der Ausführung). Im Beispiel werden 5 Tage hinzugefügt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`ToDateTime(Value)`	Konvertiert einen Wert in einen DateTime-Wert.	`ToDateTime([Orders])`	Versucht, den Wert in der Spalte "[Orders]" in ein DateTime-Objekt zu konvertieren. Der Erfolg hängt vom Datentyp von "[Orders]" ab.
`Today()`	Gibt das aktuelle Datum zurück. Unabhängig von der tatsächlichen Uhrzeit gibt diese Funktion Mitternacht des aktuellen Datums zurück.	`AddMonths(Today(), 1)`	Gibt das heutige Datum zurück (Uhrzeit 00:00:00). Im Beispiel wird ein Monat hinzugefügt.
`UtcNow()`	Gibt das aktuelle Systemdatum und die aktuelle Uhrzeit zurück, ausgedrückt als koordinierte Weltzeit (UTC).	`AddDays(UtcNow(), 7)`	Gibt das aktuelle Datum und die aktuelle Uhrzeit in UTC zurück. Im Beispiel werden 7 Tage hinzugefügt.

Logische Funktionen

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Iif(Expression1, True_Value1, ..., ExpressionN, True_ValueN, False_Value)`	Gibt einen von mehreren angegebenen Werten zurück, abhängig von den Werten logischer Ausdrücke. Die Funktion kann 2N+1 Argumente annehmen (N - die Anzahl der angegebenen logischen Ausdrücke): • Jedes ungerade Argument gibt einen logischen Ausdruck an. • Jedes gerade Argument gibt den Wert an, der zurückgegeben wird, wenn der	`Iif(Name = 'Bob', 1, 0)` `Iif(Name = 'Bob', 1, Name = 'Dan', 2, Name = 'Sam', 3, 0)`	Einfache Wenn-Dann-Sonst-Bedingung: Wenn Name = 'Bob', dann 1, sonst 0. Komplexeres Beispiel: Wenn Name = 'Bob', dann 1; wenn Name = 'Dan', dann 2; wenn Name = 'Sam', dann 3; sonst 0.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
	vorherige Ausdruck als wahr ausgewertet wird. • Das letzte Argument gibt den Wert an, der zurückgegeben wird, wenn die zuvor ausgewerteten logischen Ausdrücke falsch ergeben haben.		
`IsNull(Value)`	Gibt True zurück, wenn der angegebene Wert NULL ist.	`IsNull([OrderDate])`	Prüft, ob ein Bestelldatum ("OrderDate") fehlt (NULL ist).
`IsNull(Value1, Value2)`	Gibt Value1 zurück, wenn es nicht auf NULL gesetzt ist; andernfalls wird Value2 zurückgegeben.	`IsNull([ShipDate], [RequiredDate])`	Wenn das Versanddatum ("ShipDate") fehlt, wird stattdessen das

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
			Anforderungsdatum ("RequiredDate") verwendet.
`IsNullOrEmpty(String)`	Gibt True zurück, wenn das angegebene String-Objekt NULL oder ein leerer String ist; andernfalls wird False zurückgegeben.	`IsNullOrEmpty([ProductName])`	Prüft, ob ein Produktname ("ProductName") fehlt (NULL oder leerer String).
`ToBoolean(Value)`	Konvertiert einen Wert in einen äquivalenten booleschen Wert.	`ToBoolean([Value])`	Versucht, den Wert in der Spalte "[Value]" in einen booleschen Wert (wahr/falsch) zu konvertieren. Der Erfolg hängt vom ursprünglichen Datentyp ab.

Mathematische Funktionen

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Abs(Value)`	Gibt den absoluten, positiven Wert des gegebenen numerischen Ausdrucks zurück.	`Abs(1 - [Discount])`	Berechnet den absoluten Wert der Differenz zwischen 1 und einem Rabatt ("Discount"). Nützlich, um sicherzustellen, dass ein Wert immer positiv ist.
`Acos(Value)`	Gibt den Arkuskosinus einer Zahl zurück (den Winkel im Bogenmaß, dessen Kosinus der gegebene float-Ausdruck ist).	`Acos([Value])`	Wird seltener in typischen Geschäftsanwendungen/Dashboards verwendet. Kann für trigonometrische Berechnungen nützlich sein.
`Asin(Value)`	Gibt den Arkussinus einer Zahl zurück (den Winkel im Bogenmaß, dessen Sinus der gegebene float-Ausdruck ist).	`Asin([Value])`	Wird seltener in typischen Geschäftsanwendungen verwendet.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Atn(Value)`	Gibt den Arkustangens einer Zahl zurück (den Winkel im Bogenmaß, dessen Tangens der gegebene float-Ausdruck ist).	`Atn([Value])`	Wird seltener in typischen Geschäftsanwendungen verwendet.
`Atn2(Value1, Value2)`	Gibt den Winkel zurück, dessen Tangens der Quotient zweier angegebener Zahlen im Bogenmaß ist.	`Atn2([Value1], [Value2])`	Wird seltener in typischen Geschäftsanwendungen verwendet. Kann nützlich sein, um Winkel in einem kartesischen Koordinatensystem zu berechnen.
`BigMul(Value1, Value2)`	Gibt einen Int64 zurück, der das vollständige Produkt zweier angegebener 32-Bit-Zahlen enthält.	`BigMul([Amount], [Quantity])`	Berechnet das Produkt aus Menge und Anzahl, auch wenn das Ergebnis sehr groß ist (größer als ein normaler Integer).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Ceiling(Value)`	Gibt die kleinste ganze Zahl zurück, die größer oder gleich dem numerischen Ausdruck ist.	`Ceiling([Value])`	Rundet eine Zahl immer auf die nächste ganze Zahl *auf* (z.B. Ceiling(3.2) = 4).
`Cos(Value)`	Gibt den Kosinus des Winkels im Bogenmaß zurück.	`Cos([Value])`	Wird seltener in typischen Geschäftsanwendungen verwendet.
`Cosh(Value)`	Gibt den hyperbolischen Kosinus des Winkels im Bogenmaß zurück.	`Cosh([Value])`	Wird seltener in typischen Geschäftsanwendungen verwendet.
`Exp(Value)`	Gibt den Exponentialwert des float-Ausdrucks zurück.	`Exp([Value])`	Berechnet e (Eulersche Zahl) hoch dem angegebenen Wert. Kann für exponentielles Wachstum/Zerfall verwendet werden.
`Floor(Value)`	Gibt die größte ganze Zahl zurück, die kleiner oder gleich dem numerischen Ausdruck ist.	`Floor([Value])`	Rundet eine Zahl immer auf die nächste ganze Zahl *ab* (z.B. Floor(3.8) = 3).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Log(Value)`	Gibt den natürlichen Logarithmus einer angegebenen Zahl zurück.	`Log([Value])`	Berechnet den natürlichen Logarithmus (Basis e).
`Log(Value, Base)`	Gibt den Logarithmus einer angegebenen Zahl in einer angegebenen Basis zurück.	`Log([Value], 2)`	Berechnet den Logarithmus zur Basis 2 (z.B. Log([Value], 2) von 8 ist 3).
`Log10(Value)`	Gibt den Logarithmus einer angegebenen Zahl zur Basis 10 zurück.	`Log10([Value])`	Berechnet den Logarithmus zur Basis 10.
`Max(Value1, Value2)`	Gibt den Maximalwert aus den angegebenen Werten zurück.	`Max([Value1], [Value2])`	Vergleicht zwei Werte und gibt den größeren zurück.
`Min(Value1, Value2)`	Gibt den Minimalwert aus den angegebenen Werten zurück.	`Min([Value1], [Value2])`	Vergleicht zwei Werte und gibt den kleineren zurück.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Power(Value, Power)`	Gibt eine angegebene Zahl potenziert mit einer angegebenen Potenz zurück.	`Power([Value], 3)`	Berechnet [Value] hoch 3 (z.B. Power(2, 3) = 8).
`Rnd()`	Gibt eine Zufallszahl zurück, die kleiner als 1, aber größer oder gleich Null ist.	`Rnd()*100`	Erzeugt eine Zufallszahl zwischen 0 (einschließlich) und 1 (ausschließlich). Im Beispiel wird sie mit 100 multipliziert, um eine Zahl zwischen 0 und 100 zu erhalten.
`Round(Value)`	Rundet den gegebenen Wert auf die nächste ganze Zahl.	`Round([Value])`	Rundet eine Zahl kaufmännisch (z.B. Round(3.5) = 4, Round(3.4) = 3).
`Round(Value, Precision)`	Rundet den gegebenen Wert auf die nächste ganze Zahl oder auf eine angegebene Anzahl von Dezimalstellen.	`Round([Value], 2)`	Rundet eine Zahl auf 2 Dezimalstellen (z.B. Round(3.14159, 2) = 3.14).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Sign(Value)`	Gibt das positive (+1), Null (0) oder negative (-1) Vorzeichen des gegebenen Ausdrucks zurück.	`Sign([Value])`	Gibt -1 zurück, wenn [Value] negativ ist, 0 wenn [Value] 0 ist, und +1 wenn [Value] positiv ist.
`Sin(Value)`	Gibt den Sinus des im Bogenmaß definierten Winkels zurück.	`Sin([Value])`	Wird seltener in typischen Geschäftsanwendungen verwendet.
`Sinh(Value)`	Gibt den hyperbolischen Sinus des im Bogenmaß definierten Winkels zurück.	`Sinh([Value])`	Wird seltener in typischen Geschäftsanwendungen verwendet.
`Sqr(Value)`	Gibt die Quadratwurzel einer gegebenen Zahl zurück.	`Sqr([Value])`	Berechnet die Quadratwurzel einer Zahl.
`Tan(Value)`	Gibt den Tangens des im Bogenmaß definierten Winkels zurück.	`Tan([Value])`	Wird seltener in typischen Geschäftsanwendungen verwendet.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Tanh(Value)`	Gibt den hyperbolischen Tangens des im Bogenmaß definierten Winkels zurück.	`Tanh([Value])`	Wird seltener in typischen Geschäftsanwendungen verwendet.
`ToDecimal(Value)`	Konvertiert einen Wert in eine äquivalente Dezimalzahl.	`ToDecimal([Value])`	Konvertiert einen Wert (z.B. einen String oder einen Integer) in eine Dezimalzahl.
`ToDouble(Value)`	Konvertiert einen Wert in eine äquivalente 64-Bit-Gleitkommazahl mit doppelter Genauigkeit.	`ToDouble([Value])`	Konvertiert einen Wert in eine Double-Genauigkeit-Gleitkommazahl.
`ToFloat(Value)`	Konvertiert einen Wert in eine äquivalente 32-Bit-Gleitkommazahl mit einfacher Genauigkeit.	`ToFloat([Value])`	Konvertiert einen Wert in eine Single-Genauigkeit-Gleitkommazahl.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`ToInt(Value)`	Konvertiert einen Wert in eine äquivalente 32-Bit-Ganzzahl mit Vorzeichen.	`ToInt([Value])`	Konvertiert einen Wert in eine 32-Bit-Ganzzahl. Nachkommastellen werden abgeschnitten.
`ToLong(Value)`	Konvertiert einen Wert in eine äquivalente 64-Bit-Ganzzahl mit Vorzeichen.	`ToLong([Value])`	Konvertiert einen Wert in eine 64-Bit-Ganzzahl (Long). Nachkommastellen werden abgeschnitten.

String Funktionen

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Ascii(String)`	Gibt den ASCII-Codewert des Zeichens ganz links in einem Zeichenausdruck zurück.	`Ascii('a')`	Gibt den ASCII-Code für 'a' zurück (97). Wird seltener direkt in Dashboards verwendet.
`Char(Number)`	Konvertiert einen integer ASCII-Code in ein Zeichen.	`Char(65) + Char(51)`	Erzeugt den String "A3" (ASCII 65 = A, ASCII 51 = 3).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`CharIndex(String1, String2)`	Gibt die Startposition von String1 innerhalb von String2 zurück, beginnend von der Zeichenposition Null bis zum Ende eines Strings.	`CharIndex('e', 'devexpress')`	Sucht die Position des ersten 'e' in 'devexpress' (Ergebnis: 1, da die Zählung bei 0 beginnt).
`CharIndex(String1, String2, StartLocation)`	Gibt die Startposition von String1 innerhalb von String2 zurück, beginnend von der StartLocation-Zeichenposition bis zum Ende eines Strings.	`CharIndex('e', 'devexpress', 2)`	Sucht die Position des ersten 'e' in 'devexpress', *beginnend bei Position 2* (Ergebnis: 6).
`Concat(String1, ..., StringN)`	Gibt einen String-Wert zurück, der die Verkettung des aktuellen Strings mit allen zusätzlichen Strings enthält.	`Concat('A', ')', [ProductName])`	Verbindet den Buchstaben "A", eine Klammer ")" und den Wert des Feldes "ProductName" zu einem neuen String.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`EndsWith(String1, SubString1)`	Gibt True zurück, wenn das Ende von String1 mit SubString1 übereinstimmt; andernfalls wird False zurückgegeben.	`EndsWith([Description], 'The end.')`	Prüft, ob eine Beschreibung ("Description") mit dem Text "The end." endet.
`Insert(String1, StartPosition, String2)`	Fügt String2 in String1 an der durch StartPosition angegebenen Position ein.	`Insert([Name], 0, 'ABC-')`	Fügt "ABC-" am Anfang des Wertes im Feld "Name" ein.
`Len(Value)`	Gibt eine ganze Zahl zurück, die entweder die Anzahl der Zeichen in einem String oder die nominale Anzahl der Bytes enthält, die zum Speichern einer Variablen benötigt werden.	`Len([Description])`	Gibt die Länge (Anzahl der Zeichen) einer Beschreibung ("Description") zurück.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Lower(String)`	Gibt den String in Kleinbuchstaben zurück.	`Lower([ProductName])`	Wandelt einen Produktnamen ("ProductName") in Kleinbuchstaben um.
`PadLeft(String, Length)`	Richtet die Zeichen des definierten Strings links aus und füllt die linke Seite mit Leerzeichen bis zu einer angegebenen Gesamtlänge auf.	`PadLeft([Name], 30)`	Füllt einen Namen ("Name") links mit Leerzeichen auf, bis er 30 Zeichen lang ist.
`PadLeft(String, Length, Char)`	Richtet die Zeichen des definierten Strings links aus und füllt die linke Seite mit dem angegebenen Zeichen Char bis zu einer angegebenen Gesamtlänge auf.	`PadLeft([Name], 30, '<')`	Füllt einen Namen links mit dem Zeichen '<' auf, bis er 30 Zeichen lang ist.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`PadRight(String, Length)`	Richtet die Zeichen des definierten Strings rechts aus und füllt die linke Seite mit Leerzeichen bis zu einer angegebenen Gesamtlänge auf.	`PadRight([Name], 30)`	Füllt einen Namen *rechts* mit Leerzeichen auf, bis er 30 Zeichen lang ist.
`PadRight(String, Length, Char)`	Richtet die Zeichen des definierten Strings rechts aus und füllt die linke Seite mit dem angegebenen Zeichen Char bis zu einer angegebenen Gesamtlänge auf.	`PadRight([Name], 30, '>')`	Füllt einen Namen *rechts* mit dem Zeichen '>' auf, bis er 30 Zeichen lang ist.
`Remove(String, StartPosition)`	Löscht alle Zeichen aus dieser Instanz, beginnend an einer angegebenen Position.	`Remove([Name], 3)`	Entfernt alle Zeichen aus einem Namen ("Name"), beginnend bei Position 3 (das vierte Zeichen).

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Remove(String, StartPosition, Length)`	Löscht eine angegebene Anzahl von Zeichen aus dieser Instanz, beginnend an einer angegebenen Position.	`Remove([Name], 0, 3)`	Entfernt die ersten 3 Zeichen aus einem Namen.
`Replace(String, SubString2, String3)`	Gibt eine Kopie von String1 zurück, in der SubString2 durch String3 ersetzt wurde.	`Replace([Name], 'The ', '')`	Ersetzt in einem Namen ("Name") den Teilstring "The " durch einen leeren String (entfernt also "The ").
`Reverse(String)`	Kehrt die Reihenfolge der Elemente innerhalb eines Strings um.	`Reverse([Name])`	Kehrt die Buchstabenreihenfolge in einem Namen um (z.B. "Anna" wird zu "annA").
`StartsWith(String1, SubString1)`	Gibt True zurück, wenn der Anfang von String1 mit SubString1 übereinstimmt; andernfalls False.	`StartsWith([Title], 'The best')`	Prüft, ob ein Titel ("Title") mit "The best" beginnt.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Substring(String, StartPosition, Length)`	Ruft einen Teilstring von String ab. Der Teilstring beginnt bei StartPosition und hat eine angegebene Länge.	`Substring([Description], 2, 3)`	Extrahiert einen Teilstring aus einer Beschreibung ("Description"): 3 Zeichen, beginnend bei Position 2 (dem dritten Zeichen).
`Substring(String, StartPosition)`	Ruft einen Teilstring von String ab. Der Teilstring beginnt bei StartPosition.	`Substring([Description], 2)`	Extrahiert einen Teilstring aus einer Beschreibung, beginnend bei Position 2 und bis zum Ende der Beschreibung.
`ToStr(Value)`	Gibt eine String-Repräsentation eines Objekts zurück.	`ToStr([ID])`	Konvertiert eine ID (die möglicherweise eine Zahl ist) in einen String.
`Trim(String)`	Entfernt alle führenden und nachfolgenden Leerzeichen aus dem String.	`Trim([ProductName])`	Entfernt Leerzeichen am Anfang und Ende eines Produktnamens.

Funktion	Beschreibung	Beispiel	Erläuterung Beispiel
`Upper(String)`	Gibt den String in Großbuchstaben zurück.	`Upper([Product-Name])`	Wandelt einen Produktnamen in Großbuchstaben um.

Operatorrangfolge

Wenn ein Ausdruck mehrere Operatoren enthält, steuert ihre Rangfolge die Reihenfolge, in der Ausdruckselemente ausgewertet werden.

- Literale Werte
- Parameter
- Bezeichner
- OR (links-assoziativ)
- AND (links-assoziativ)
- ==, !=
- , <=, >=
- -, + (links-assoziativ)
- *, /, % (links-assoziativ)
- NOT
- unäres -
- In
- Iif
- Trim(), Len(), Substring(), IsNull()
- '[]' (für Mengenbeschränkung)
- '()'

Die Standardrangfolge kann durch Gruppieren von Elementen mit Klammern geändert werden. Zum Beispiel werden die Operatoren in den folgenden beiden Codebeispielen in der Standardreihenfolge ausgeführt. Im zweiten Codebeispiel wird die Additionsoperation zuerst ausgeführt, da die zugehörigen Elemente gruppiert sind und die Multiplikationsoperation zuletzt ausgeführt wird.

```
Amount == 2 + 48 * 2
```

```
Amount == (2 + 48) * 2
```

Groß- und Kleinschreibung

Operatoren unterscheiden nicht zwischen Groß- und Kleinschreibung. Obwohl die Groß- und Kleinschreibung von Feldwerten von der Datenquelle abhängt.

HINWEIS
Eine Datenquelle beeinflusst das Verhalten bestimmter Operatoren. Beispielsweise ist der SQL Server Express 2005 standardmäßig so konfiguriert, dass die Groß-/Kleinschreibung nicht beachtet wird. In diesem Fall ergibt der folgende Ausdruck immer true:

```
Lower(Name) == Upper(Name)
```

Schlüsselwörter escapen

Sie können einen schlüsselwortähnlichen Feldnamen mit einem Escapezeichen (@-Zeichen) markieren. Im Ausdruck unten interpretiert die Methode `CriteriaOperator.Parse` @Or als das Feld mit dem Namen "Or", nicht den logischen Operator OR.

```
@Or = 'value'
```

Escape-Zeichen

Verwenden Sie einen Backslash (\) als Escapezeichen für Zeichen in Ausdrücken. Beispiele:

- \[
- \\
- \'

Hinweis: Da es sich bei den verwendeten Komponenten um einen Drittanbieter handelt, werden die hier beschriebenen Funktionen „as is“ aufgelistet. Weitergehende Garantien können nicht gegeben werden.

Kontaktinformationen:

bitfarm Informationssysteme GmbH

Spandauer Str. 18

57072 Siegen

info@bitfarm-archiv.de