Aggregate Function Reference

All standard aggregate functions are listed on this page.

An aggregate function call contains an expression in curly braces ("{}"), which is calculated for the item and all sub-items (or, in some cases, for other subset of related items in the structure), and then the resulting values are aggregated according to the meaning of the aggregate function.

An aggregate function may have modifiers, which are all listed here.

Aggregation Functions

SUM

Sum calculates numerical total for the values calculated for the item and/or its sub-items.

Summary	X	SUM{X}
T1	3	6
T1.1	2	2
T1.2		1
T1.2.1	1	1

Note that when the value of the expression under aggregation is not numeric (cannot be converted to number), it is ignored.

If a certain issue (or another kind of item) is included multiple times in the sub-tree, the sum will include the value for that issue only *once*. This behavior can be overridden by #all modifier.

Accepts modifiers: #all, #children, #leaves, #strict.

COUNT

Count calculates count of defined (or truthy if #truthy modifier is specified) values for the item and/or its sub-items.

Summary	X	COUNT{X}
T1	3	3
T1.1	2	1
T1.2		1
T1.2.1	1	1

If a certain issue (or another kind of item) is included multiple times in the sub-tree, it will be counted only *once*. This behavior can be overridden by #all modifier.

Accepts modifiers: #all, #children, #leaves, #strict, #truthy.

AVG

Avg calculates an average of defined values for the item and/or its sub-items. The result for avg is generally the same as sum/count. Returns nothing in case there are no defined values for {x}.

Summary	X	AVG{X}
T1	3	2
T1.1	2	2
T1.2		1
T1.2.1	1	1

If a certain issue (or another kind of item) is included multiple times in the sub-tree, the average value will include the value for that issue only *once*. This behavior can be overridden by #all modifier.

Accepts modifiers: #all, #children, #leaves, #strict.

MAX

Max calculates maximum of defined values for the item and/or its sub-items. Numeric, date, duration and text fields can be compared. Text fields are compared lexicographically.

Summary	X	MAX{X}
T1	3	3
T1.1	2	2
T1.2		1
T1.2.1	1	1

Accepts modifiers: #children, #leaves, #strict.

MIN

Min calculates minimum of defined values for the item and/or its sub-items. Numeric, date, duration and text fields can be compared. Text fields are compared lexicographically.

Summary	X	MIN{X}
T1	3	1
T1.1	2	2
T1.2		1
T1.2.1	1	1

Accepts modifiers: #children, #leaves, #strict.

JOIN

Join calculates concatenation of strings. If current row has children and #subtree modifier is set, join appends values for children wrapping them into characters (braces by default). Wrapping characters can be set by #beforeChildren and #afterChildren (see example for #subtree to see how it works). By default it joins all parent string values from root to self value.

Summary	X	JOIN{X}
T1	3	3
T1.1	2	3, 2
T1.2		3, ?
T1.2.1	1	3, ?, 1

Accepts modifiers: #ancestors, #subtree, #children, #leaves, #strict, #reverse, #separator, #beforeChildren, #afterChildren, #fromDepth, #toDepth.

PARENT

Parent extracts value from the parent row or from ancestor row by specified depth.

Summary	X	PARENT{X}
T1	3	
T1.1	2	3
T1.2		3
T1.2.1	1	

Accepts modifier: #depth.

Aggregation Modifiers

#all

When this modifier is accessible aggregation function applies to distinct items by default.

This modifier turns off distinct.

Summary	X	SUM{X}	SUM#all{X}	COUNT{X}	COUNT#all{X}
T1		2	6	1	3
T1.1	2	2	2	1	1
T1.1	2	2	2	1	1
T1.1	2	2	2	1	1

```
SUM#all{X}
COUNT#all{X}
```

Can be used with: sum, count, avg.

#truthy

Only count row if subexpression produces truthy value.

Summary	X	COUNT#truthy{X}
T1	0	2
T1.1	2	1
T1.2		1
T1.2.1	1	1

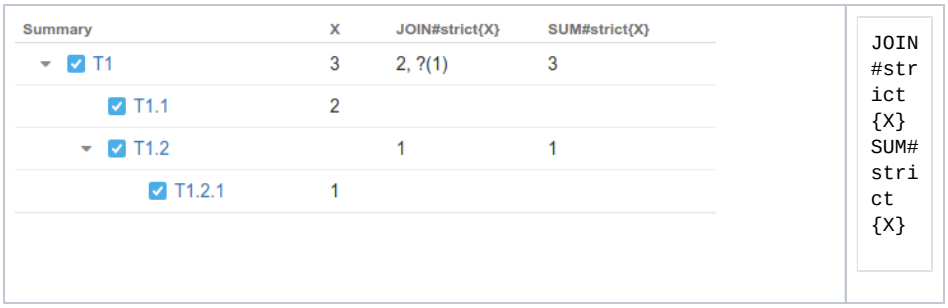
```
COUNT#truthy{X}
```

Can be used with: count.

#strict

Do not process current row item as part of aggregation.

Cannot be used together with #children (it implies the same effect), #ancestors (use depth modifiers for that), #leaves (together they're useless).



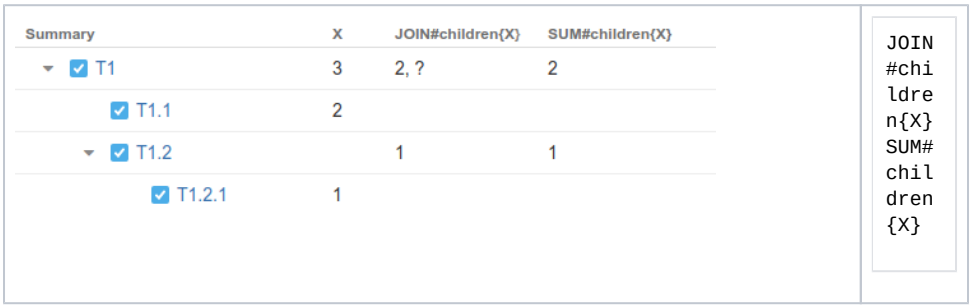
Summary	X	JOIN#strict{X}	SUM#strict{X}
T1	3	2, ?(1)	3
T1.1	2		
T1.2		1	1
T1.2.1	1		

```
JOIN
#str
ict
{X}
SUM#
stri
ct
{X}
```

Can be used with: sum, count, avg, join, min, max.

#children

Only process direct children of current row.



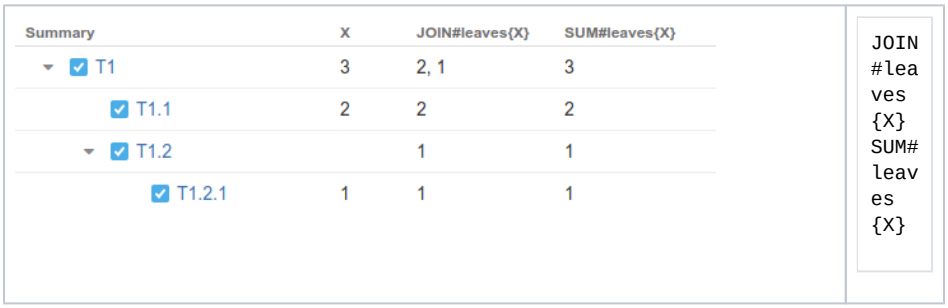
Summary	X	JOIN#children{X}	SUM#children{X}
T1	3	2, ?	2
T1.1	2		
T1.2		1	1
T1.2.1	1		

```
JOIN
#chi
ldre
n{X}
SUM#
chil
dren
{X}
```

Can be used with: sum, count, avg, join, min, max.

#leaves

Only process leaves of subtree of current row.



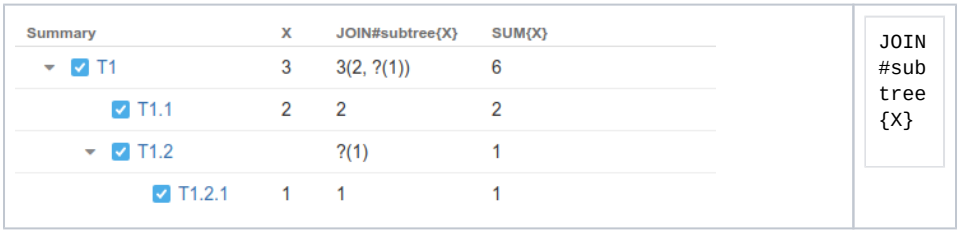
Summary	X	JOIN#leaves{X}	SUM#leaves{X}
T1	3	2, 1	3
T1.1	2	2	2
T1.2		1	1
T1.2.1	1	1	1

```
JOIN
#lea
ves
{X}
SUM#
leav
es
{X}
```

Can be used with: sum, count, avg, join, min, max.

#subtree

Process whole subtree of current row. This is default behavior for sum, count, avg, min, max.



Summary	X	JOIN#subtree{X}	SUM{X}
T1	3	3(2, ?(1))	6
T1.1	2	2	2
T1.2		?(1)	1
T1.2.1	1	1	1

```
JOIN
#sub
tree
{X}
```

Can be used with: join.

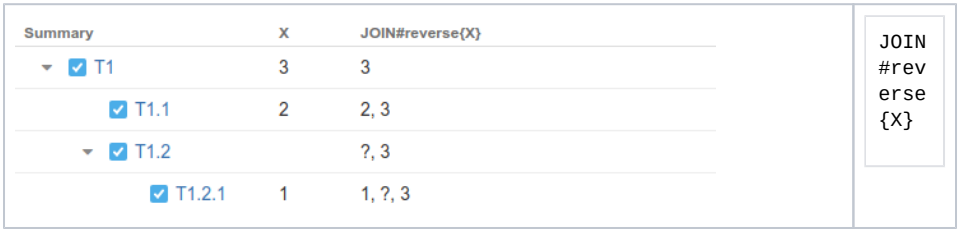
#ancestors

Only process ancestors of current row. This is default behavior for join, parent.

Can be used with: join.

#reverse

Reverses the order of row processing.



Summary	X	JOIN#reverse{X}
T1	3	3
T1.1	2	2, 3
T1.2		?, 3
T1.2.1	1	1, ?, 3

```
JOIN
#rev
erse
{X}
```

Can be used with: join.

#separator

Defines separator for string joining. This modifier has string parameter, default is ", ".

Summary	X	JOIN#separator="->"{X}
T1	3	3
T1.1	2	3->2
T1.2		3->?
T1.2.1	1	3->?->1

```
JOIN#se
parator
="->"
{X}
```

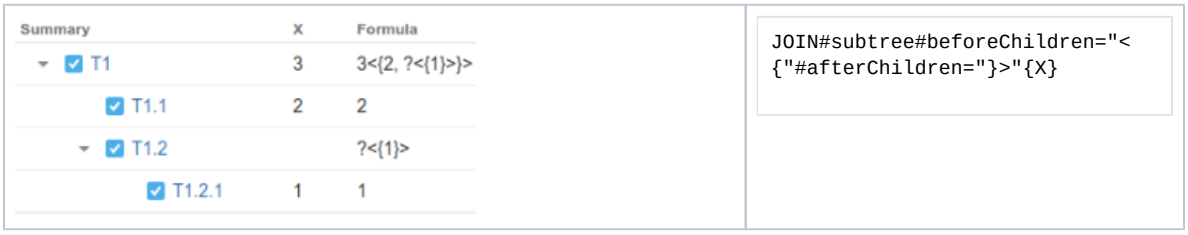
Can be used with: join.

#beforeChildren

See #afterChildren.

#afterChildren

Defines exit separator between children and parent rows. This modifier has string parameter, default is "(" for #beforeChildren and ")" for #afterChildren.



Summary	X	Formula
T1	3	3<{2, ?<{1}>}>
T1.1	2	2
T1.2		?<{1}>
T1.2.1	1	1

```
JOIN#subtree#beforeChildren="<
{"#afterChildren="}>"{X}
```

Can be used with: join.

#fromDepth

Specifies position of the row that would be the first in sequence of rows aggregate function takes as an input.

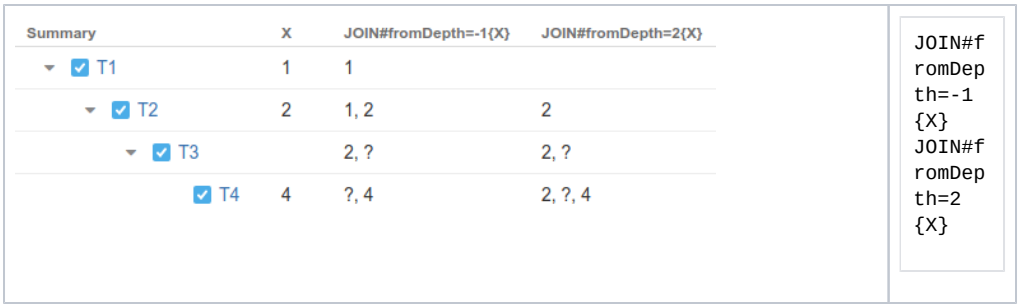
Position is specified by integer parameter denoted as n below.

Positive values mean absolute depth of row in the structure, e.g. n=1 means root.

Negative values mean depth relative to current row, e.g. n=-1 is direct parent.

Default is 1. n shouldn't be 0.

This modifier doesn't work with any tree types except #ancestors.



Summary	X	JOIN#fromDepth=-1{X}	JOIN#fromDepth=2{X}
T1	1	1	
T2	2	1, 2	2
T3		2, ?	2, ?
T4	4	?, 4	2, ?, 4

```
JOIN#fromDepth=-1{X}
JOIN#fromDepth=2{X}
```

Can be used with: join.

#toDepth

Specifies position of the row that would be the last in sequence of rows aggregate function takes as an input.

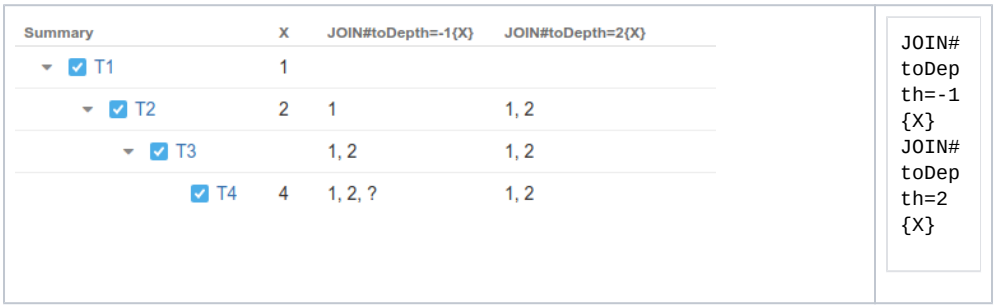
Position is specified by integer parameter denoted as n below.

Positive values mean absolute depth of row in the structure, e.g. n=1 means root.

Negative values mean depth relative to current row, e.g. n=-1 is direct parent.

Default is 0. 0 means current row.

This modifier doesn't work with any tree types except #ancestors.



Summary	X	JOIN#toDepth=-1{X}	JOIN#toDepth=2{X}
T1	1		
T2	2	1	1, 2
T3		1, 2	1, 2
T4	4	1, 2, ?	1, 2

```
JOIN#toDepth=-1{X}
JOIN#toDepth=2{X}
```

Can be used with: join.

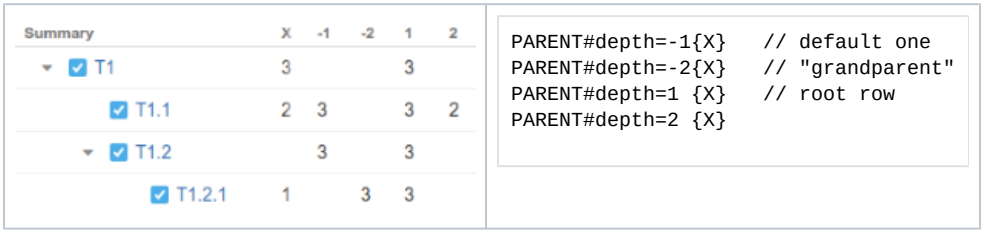
#depth

Denotes the parent that possesses value. This is specified via integer parameter denoted as n below.

Positive values mean absolute depth of row in the structure, e.g. n=1 means root.

Negative values mean depth relative to current row, e.g. n=-1 is direct parent.

Default is -1. n shouldn't be 0.



Summary	X	-1	-2	1	2
T1	3			3	
T1.1	2	3		3	2
T1.2		3		3	
T1.2.1	1		3	3	

```
PARENT#depth=-1{X}   // default one
PARENT#depth=-2{X}   // "grandparent"
PARENT#depth=1 {X}   // root row
PARENT#depth=2 {X}
```

Can be used with: parent.