Aggregate Function Reference

All standard aggregate functions and available modifiers 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 another subset of related items in the structure), and then the resulting values are aggregated according to the meaning of the aggregate function.

- **Aggregation Functions**
  - AVG
  - COUNT
  - JOIN
  - MAX
  - MIN
  - PARENT
  - SUM

- **Aggregation Modifiers**
  - #afterChildren
  - #all
  - #ancestors
  - #baseLevel
  - #beforeChildren
  - #children
  - #depth
  - #distinct
  - #fromDepth
  - #fromLevel
  - #leaves
  - #level
  - #levels
  - #reverse
  - #separator
  - #strict
  - #subtree
  - #toDepth
  - #toLevel
  - #truthy

In Structure versions prior to 6.0, it was not possible to include both upward-looking and downward-looking aggregate functions within the same formula. When using one of the two upward-looking aggregate functions, PARENT and JOIN (when used with an upward-looking modifier), you could not include any of the other aggregate functions listed above.

Starting with Structure 6, this limitation is removed.

**Aggregation Functions**

**AVG**

Avg calculates an average of the defined values for the item and/or its sub-items. The result for avg is generally the same as sum/count. It returns nothing if there are no defined values for \( x \).

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 using the #all modifier.

Accepts modifiers: #all, #children, #leaves, #strict, #preceding, #levels (together with #preceding), #baseLevel (together with #preceding).

**COUNT**

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

Join concatenates (joins) strings from the item and its parents (or other items, if modifiers are used).

- By default, it joins all parent string values from root to the self value.
- If the current row has children and the #subtree modifier is set, join appends the 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).

MAX

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

MIN

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

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

Other variations of SUM allow different types of aggregation:

- SUM{} aggregates values from all descendants
- SUM#children{} aggregates values from direct children only
- SUM#leaves{} aggregates values from leaves
- SUM#preceding{} aggregates values from the preceding rows

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

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

Aggregation Modifiers

#afterChildren

Defines the exit separator between children and parent rows. This modifier has a string parameter. The default exit separator is:

- "(" - for #beforeChildren
- ")" - for #afterChildren

Can be used with: join.

#all

Tells the aggregate function to include duplicate items. By defaults, aggregate functions ignore duplicate items.
Can be used with: sum, count, avg.

#ancestors
Only process ancestors of the current row. This is the default behavior for join and parent.
Can be used with: join.

#baseLevel
Can be used with SUM with the #preceding modifier to specify at which level the accumulation will be reset and start over from zero. This allows for accruing independent cumulative values in different sub-trees.

- By default, baseLevel is 0, which means that the accumulation will never start over and will cover the whole structure.
- If baseLevel is not zero, then the accumulation will be restarted once it reaches a row at the base level or higher in the hierarchy. For example, if you have epics at the top level, and stories underneath them, SUM#preceding#baseLevel=1{story_points} will accumulate the Story Points within the scope of each epic independently.

Can be used with: avg, count, sum (together with #preceding).

#beforeChildren
See #afterChildren.

#children
Only process direct children of the current row.

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

#depth
Same as #level modifier.

#distinct
Makes join only concatenate distinct values. A duplicate value won't be added more than once if this modifier is on.
Modifiers #beforeChildren and #afterChildren don't work when this option is on.
Can be used with: `join`.

**#fromDepth**

Same as `#fromLevel`.

**#fromLevel**

Specifies the position of the first row the aggregate function should take as input for a sequence.

Position is specified by an integer parameter denoted as n below:

- Positive values mean the absolute depth of the row in the structure, e.g. n=1 means root.
- Negative values mean the depth relative to current row, e.g. n=-1 is the current item's direct parent.
- Default is 1.
- Zero means the "super-root" row, which is a fictional parent of all the top rows. It can be used to get the value of another aggregate function applied to the whole structure. For example, `JOIN#fromLevel=0{MIN{due_date}}` will provide a sequence of the earliest due dates, starting from the earliest throughout the whole structure, then the earliest throughout the root tree this item is in, and so on.

This modifier does not work with any tree types except `ancestors`. 
Can be used with: `join`.

#leaves

Only process leaves (items without children) in the subtree of the current row.

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

#level

When used with `PARENT`, specifies the position of the parent that possesses value.

Position is specified by an integer parameter denoted as `n` below:

- Positive values mean the absolute depth of the row in the structure: `n=1` means root element, `n=2` means an element at the 2nd level, etc.
- Negative values mean the depth relative to the current row: `n=-1` is the current item's direct parent.
- Default is `-1`.
- Zero means the "super-root" row, which is a fictional parent of all the top rows. It can be used to get the value of another aggregate function applied to the whole structure. For example, `PARENT#level=0{SUM{story_points}}` means total story points for the whole structure (including subtrees for all roots).

When used with `SUM` with the `#preceding` modifier, this specifies the level at which the values should be aggregated.

Can be used with: `parent`, `sum` (together with `#preceding`), `count` (together with `#preceding`), `avg` (together with `#preceding`).

#levels

Can be used with `SUM` with the `#preceding` modifier to specify at which levels the accrual of the values happen.

- It can be a single numeric value, for example: `SUM#preceding#levels=1{story_points}` will accumulate Story Points from top to bottom at level 1.
- It can be a list of numbers, in which case the list must be wrapped in quotes: `SUM#preceding#levels="2,4"{time_spent}` will accumulate Time Spent on levels 2 and 4.
- By default, all levels are counted.

Note that if you use the `#baseLevel` modifier, only values at levels that are deeper than the base level will be counted.

You can also use `#level` instead of `#levels`. 
Can be used with: `avg`, `count`, `sum` (together with #preceding).

### #reverse

Reverses the order of row processing.

<table>
<thead>
<tr>
<th>Summary</th>
<th>X</th>
<th>JOIN#reverse(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>T1.1</td>
<td>2</td>
<td>2, 3</td>
</tr>
<tr>
<td>T1.2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>T1.2.1</td>
<td>1</td>
<td>1, 3, 3</td>
</tr>
</tbody>
</table>

Can be used with: `join`.

### #separator

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

<table>
<thead>
<tr>
<th>Summary</th>
<th>X</th>
<th>JOIN#separator=&quot;-&gt;&quot;(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>3</td>
<td>3-&gt;2</td>
</tr>
<tr>
<td>T1.1</td>
<td>2</td>
<td>3-&gt;2</td>
</tr>
<tr>
<td>T1.2</td>
<td>3</td>
<td>3-&gt;?</td>
</tr>
<tr>
<td>T1.2.1</td>
<td>1</td>
<td>3-&gt;?-&gt;1</td>
</tr>
</tbody>
</table>

Can be used with: `join`.

### #strict

Do not process the current row item as part of the aggregation.

Cannot be used together with `children`, `ancestors` or `leaves`, since these already exclude the current row.

<table>
<thead>
<tr>
<th>Summary</th>
<th>X</th>
<th>JOIN#strict(X)</th>
<th>SUM#strict(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>3</td>
<td>2, ?(1)</td>
<td>3</td>
</tr>
<tr>
<td>T1.1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>T1.2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T1.2.1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

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

### #subtree

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

<table>
<thead>
<tr>
<th>Summary</th>
<th>X</th>
<th>JOIN#subtree(X)</th>
<th>SUM(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>3</td>
<td>3(2, ?(1))</td>
<td>6</td>
</tr>
<tr>
<td>T1.1</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>T1.2</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>T1.2.1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Can be used with: `join`.
#toDepth

Same as #toLevel.

#toLevel

Specifies the position of the last row the aggregate function should take as input for a sequence.

Position is specified by an integer parameter denoted as n below:

- Positive values mean the absolute depth of row in the structure, e.g. n=1 means root.
- Negative values mean the depth relative to current row, e.g. n=-1 is the current item's direct parent.
- 0 means current row.
- Default is 0.

This modifier does not work with any tree types except ancestors.

![Diagram of #toLevel](image)

Can be used with: join.

#truthy

Only count row if the subexpression produces a truthy value.

![Diagram of #truthy](image)

Can be used with: count.