Non-Additive Measures

Non-additive measures are those that cannot be summed across any dimensional groupings using basic addition, since this would usually produce an inaccurate result. The most common example of a non-additive measure is a distinct count of an attribute value.

The AtScale engine does not create aggregates for non-additive measures. You can, however, manually create a user-defined aggregate for a non-additive measure if necessary.

The aggregate calculations listed below produce non-additive measures in AtScale.

DISTINCT COUNT (Exact)

For example, suppose you wanted to know how many distinct customers purchased a particular product. If a single customer buys a product more than once, an additive count will count the customer twice and give incorrect results. For example, there was a total of 6 sales for the product, but those sales came from only 5 distinct customers. In order to get the correct results, you need to look at the individual customers who bought the product, and take out the duplicates.

The AtScale engine does not create aggregates for this type of measure because the engine would have to compute and store every exact combination of dimension members to ensure accuracy, which is not scalable. For example, to count the number of customer purchases by gender (which has 2 distinct values - Male, Female) and by state (which has potentially 50 distinct values - AZ, CA, TX, and so on), the intersection of those two dimensions alone would require 100 pre-computed values for each non-additive measure.

Percentile

Before trying to create a percentile measure:

1. Make sure you are using a data warehouse that supports Percentile Measures. This feature is supported on the Data Warehouses with **NTILE** checked on the Data Warehouse Support Matrix.
2. Ensure that the **Percentile Measures** feature is enabled for your AtScale organization. For details on how to enable it, see Enable or Disable Features.

With Percentile measure, analysts using client BI tools can query data sources to show data values at percentile ranks, to insulate their analyses from outliers, and to find the percentile rank for specific records.

You can create percentile measures on numeric columns that use the float or integer data type. These measures can estimate the median values, the values at quartile ranks, the values at decile ranks, or the values at custom ranks. AtScale estimates the values through an algorithm that requires memory and CPU resources in your data warehouse.

The higher the quality of the estimate you want AtScale to calculate, the more memory is required. You select a level of quality (Low, Medium, High, or Custom) when you create a percentile measure; for details, see Add Additive or Non-Additive Measures. Consider the following when deciding which level of quality to select:

- Dimensionality of the queries that will use your percentile measures

  Queries that reference a large number of dimensions require more memory on the nodes of your data warehouse than do queries that reference a smaller number of dimensions. Of course, "large" and "small" are relative to your system resources and the number of dimensions in your data models.

  If the bulk of the queries that will use the measure will reference a large number of dimensions, select either Low or Medium. The Low setting requires less memory in your data warehouse than the High setting (with the requirement for Medium being somewhere in between High and Low), but at the cost of less accurate estimates of percentile values.

  If the bulk of the queries that will use the measure will reference a smaller number of dimensions, select either Medium or High. The High setting gives more accurate percentile estimates (with the accuracy at the Medium level somewhere in between High and Low), but at the cost of more memory in your data warehouse.

- Amount of memory on each node of your data warehouse

  The task of percentile estimation during query processing is divided across your nodes. High-quality estimates require more memory, whereas as low-quality estimates require relatively less memory. Data warehouses that have a smaller number of nodes must have relatively more memory per node than data warehouses that have a larger number of nodes; the reason is that, if you have fewer nodes, more processing must be done on each node than is the case when you have more nodes.

- Using Custom quality

  The level of quality can also be set using an integer number between 1 and 50,000. For the three predefined levels Low, Medium, and High it is 50, 200 and 1000, respectively. For large data sets (billions of rows), using the predefined levels might occasionally lead to incorrect results. In such situation, choosing the Custom level and setting a higher quality value (for example, 15000) can help improving the results.

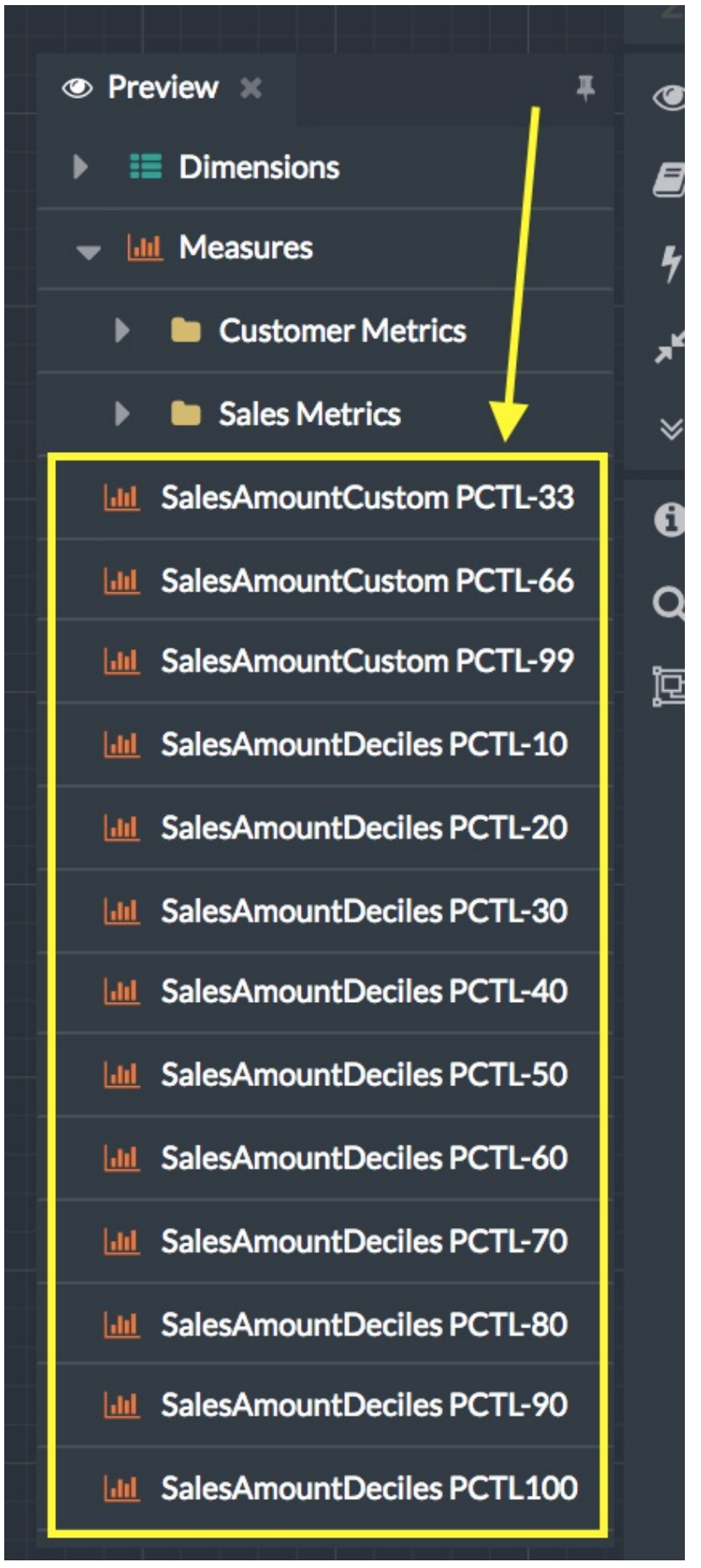
Ultimately, you will need to run tests that are based on the criteria listed above to decide which quality setting to use.

After you create a percentile measure, it appears in the Measures pane of the Cube Designer canvas with your other measures.

Figure 1. How percentile measures appear in the Measures pane, the first measure having custom percetile ranks and the second ranked by decile

To see how a percentile measure will appear in a BI client, open the Preview pane.

Figure 2. How these two percentile measures appear in the Preview pane



Percentile Example

Suppose that an aggregate exists that has these two characteristics:

- It contains one or more percentile measures.
- It contains or joins to the lowest level of a hierarchy in a dimension.

Next, suppose that a query is issued for a percentile ranking based on a level higher in the hierarchy of the dimension.

If `AGGREGATES.CREATE.BUILDFROMEXISTING`, an advanced setting for the AtScale engine, is set to **True** (its default value), then the engine can satisfy the query by creating a smaller aggregate that is based on the current aggregate, rather than by scanning the fact table. The engine can do this whether the first aggregate is system-defined or user-

defined.

For example, suppose that the engine created an aggregate in response to queries for the median net sales price for an item in each state. If a query subsequently asks for the median net sales per country, the engine can create a new aggregate by rolling up the data in the existing aggregate. The alternative to creating this second aggregate would be for the engine to scan the fact table.