Non-Additive Metrics

Non-additive metrics 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 metric is a distinct count of an attribute value.

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

The aggregate calculations listed below produce non-additive metrics 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 metric 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 metric.

Percentile

Before trying to create a percentile metric, make sure you are using a data warehouse that supports Percentile Metrics. This feature is supported on the Data Warehouses with **NTILE** checked on the Data Warehouse Support Matrix.

With Percentile metrics, 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 metrics on numeric columns that use the float or integer data type. These metrics can estimate values at specific quantiles: median, quartile, or decile. AtScale estimates the values through an algorithm that requires memory and CPU resources in your data warehouse.

Note: You can only set the quantile of the metric by editing the file's underlying SML. For more information, see Metrics.

The higher the quality of the estimate you want AtScale to calculate, the more memory is required. Whn you create a percentile metric, you can set a level of quality: 1 - 50,000. Quality values can be broken down as follows:

- 50: Low quality
- 200: Medium qualtiy
- 1000: High quality

Note: You can only set the quality of the metric by editing the file's underlying SML. For more information, see Metrics.

Consider the following when deciding which level of quality to set:

- Dimensionality of the queries that will use your percentile metrics

  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 metric will reference a large number of dimensions, use either a low (50) or medium (200) quality estimate. Lower quality estimates require less memory in your data warehouse than high quality ones (1000), but at the cost of less accurate estimates of percentile values.

  If the bulk of the queries that will use the metric will reference a smaller number of dimensions, use either a medium or high quality estimate. A high estimate provides more accurate percentile estimates (with the accuracy of a medium value 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.

- Higher qualities

  As previously mentioned, the level of quality can be set using an integer number between 1 - 50,000. For large data sets (billions of rows), setting a higher quality value (for example, 15000) can help improve the results.

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

More information on creating percentile metrics, see Add Additive or Non-Additive Metrics and Metrics.

Percentile Example

Suppose you have an aggregate with these characteristics:

- It contains one or more percentile metrics.
- 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 the `AGGREGATES.CREATE.BUILDFROMEXISTING` engine setting is set to **True** (its default value), then the engine can satisfy the query by creating a smaller aggregate 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.