Flexible Aggregation Functions

Alchemer Dashboard is currently waitlist only. Visit this page **to learn more about Dashboard or join the waitlist!**

How Aggregation Formulas Work

Typically, the groupings and filters used in a formula use the same fields as columns returned in the search results. The concept of a grouping equates to an attribute column.

For example, in the search `revenue ship mode`, `revenue` is the measure, and `ship mode` is the attribute, or grouping. The result of this search shows total revenue for each ship mode:

revenue	ship mode
$ *a*	air
$ *r*	rail
$ *t*	truck
$ *s*	sea transport

The aggregation formulas are described in Overview of aggregate formulas.

About Flexible Aggregation

Alchemer Dashboard provides flexible aggregation with the `group_*` functions. You can use `group_*` formulas when you want to specify columns and filters to include or ignore in your query.

The `group_*` formulas use a sub-query to perform these custom aggregations. If the sub-query is at a different level of detail than the original query, Alchemer Dashboard adds the result column to the result of the original query.

To use the groups and filters, specify them using the `query_groups` and `query_filters` keywords, respectively. You can also add or exclude groups or filters.

Best Practices for Flexible Aggregations

The `group_aggregate` function enables you to calculate a result at a specific aggregation level, and then returns it at a different aggregation level. For this reaggregation result to return correctly, follow these syntax guidelines:

- Wrap `group_aggregate` in an aggregate function, such as `sum` or `average`
- The wrapping function must be the immediate preceding function, such as `sum(group_aggregate(...))`

Examples

For a search on `revenue monthly ship mode`, you can add a formula to calculate yearly revenue by ship mode:

```
group_aggregate(sum(revenue), {ship mode, year(commit date)}, {})
```

The same formula can also be written using `query_groups()` and `query_filters()` as following:

```
group_aggregate(sum(revenue), query_groups() - {commit date} + {year(commit date)}, {})
```

This is helpful to include the main query groups that are not known at formula creation time. You can use `+/-` to modify the set of groups included from the query.

Groups and filters

Flexible group aggregate formulas allow for flexibility in both groupings and filters. The formulas give you the ability to specify only groupings or only filters.

Query groups

With `query_groups()+ {attribute_column}` or `query_groups()-{attribute_column}`, you can aggregate results while including/excluding a column from the original search.

The `query_groups()` function returns all attribute columns defined in the base search, when the table view is displayed.

> Columns are not included in the `query_groups()` definition when a chart is displayed NON-VISUALIZED.

If, for example, you use the condition `query_groups() - {region, sku, name}`, this changes the level of detail for the group aggregate formula. In this scenario, region, sku, and name have been removed. If these columns are not included in the base search, then this definition is ignored.

Under the new logic, the `{region, sku, name}` condition fixes the level of detail for the `group by` columns to the columns defined.

You can combine options to both add and remove columns like in the following example:

`query_groups()+ {region} - {sku, name}`.

Regarding dates, when query_groups() is defined, the date period defined in the search will be passed into the group function. Assuming the search is **monthly**, then the group function will also be at the monthly grain. You can use date functions to change the grain, such as `{start_of_year(transaction date)}`.

Query Filters

With `query_filters()+ {filter_condition}` or `query_filters()-{filter_condition}`, you can aggregate the results while including/excluding a filter condition.

Filter condition: `Ship Mode='car'`

For a search on `Category Customer ID sales by customer id and category Ship Mode='car'`, you can add a formula to calculate sales by category for each customer as:

```
sales by Customer ID and Category= group_aggregate (sum(Sales), {Category, Customer ID }, query_filters()+{Ship Mode='a
ir'})
```

In this case, the results will be aggregated based on the dimensions: `'Category'` and `'Customer ID'` and filters: `'air'` and `'car'`.

With `query_filters()-{column}`, users will be able to aggregate the results while removing any expression related to a column.

Filter condition: `Ship Mode='car'`

For a search on `Customer ID sales by customer id and category Ship Mode='car'`, you can add a formula to calculate sales for each customer while ignoring the filter on a column as:

```
sales by Customer ID and Category= group_aggregate (sum(Sales), {Customer ID, Category }, query_filters()-{Ship Mode})
```

In this case, the results will be aggregated based on the dimensions in the search; Customer ID and any filter related to Ship Mode will not be considered while aggregating the results.

Group aggregation filters enhancement

You can specify a column in the third argument of a group_aggregate function in order to include all filters defined in the search that pertain to that column, and ignore unrelated filters. This enhancement allows the analyst to specify which fields to accept filters from, improving readability of formulas, and reducing operational change and related errors.

For example, say that you would like to determine how many stores a certain product was sold in during a given time period, expressed as a ratio of total available stores. In order to calculate the total available stores you require a group_aggregate function that will accept the filters associated with transaction date (for example, this year, last six months, august), and ignore filters from other fields.

Simply specify the formula as `group_aggregate(count(storeid),query_groups, {date})`, and the formula

will automatically ignore all unrelated filters in the search bar.

Related Articles