Custom Expression Examples

Apply Functions to Simple Metrics

Example: Total number of teachers and students.

Add the count of teachers to the count of students.

```
[Distinct Count: Teacher ID] + [Distinct Count: Student ID]
```

Example: Rounding the average test score to two decimal places.

Take the average test score and apply the `round(x, p)` function, setting the precision p equal to 2.

```
round([Average: Test Score], 2)
```

Example: What is the average of all dimensional scores?

Use variables to hold the total of each dimensional score. Then add all of those scores to get the overall total. Then divide the overall total by the number of test sessions in order to calculate the average. Finally, apply the `round(x, p)` function to round to two decimal places.

You must use the semi-colon (;) character to separate each line in the expression.

When an expression contains multiple lines, the expression that appears on the last line (in this example, the line `round(overallAverage, 2)`) will be the final result returned.

```
dim1Total = [Total: Dimension 1 Score];
dim2Total = [Total: Dimension 2 Score];
dim3Total = [Total: Dimension 3 Score];
dim4Total = [Total: Dimension 4 Score];
dim5Total = [Total: Dimension 5 Score];

overallTotal = sum(dim1Total, dim2Total, dim3Total, dim4Total, dim5Total);
overallCount = [Count: Session ID];

overallAverage = overallTotal / overallCount;

round(overallAverage, 2)
```

Compare two or more Metrics

Example: Student-Teacher ratio.

The number of students divided by the number of teachers. Distinct counts must be used here if the data contains more than one row per student or teacher.

```
[Distinct Count: Student ID] / [Distinct Count: Teacher ID]
```

Use a Comparison to Create a Textual Result

Example: Which scoring dimension had the lowest average score?

Use variables to hold the average of each of the dimensional scores. Use `min(...)` function to find the lowest dimensional score. Use nested `if (cond, trueValue, falseValue)` blocks to return which dimension matches the lowest score. Note: If more than one dimension shares the same lowest score, this only returns the first dimension found.

```
dim1Avg = [Average: Dimension 1 Score];
dim2Avg = [Average: Dimension 2 Score];
dim3Avg = [Average: Dimension 3 Score];
dim4Avg = [Average: Dimension 4 Score];
dim5Avg = [Average: Dimension 5 Score];

lowestScore = min(dim1Avg, dim2Avg, dim3Avg, dim4Avg, dim5Avg);

result = if (dim5Avg == lowestScore, "Dimension 5", "Unknown");
result = if (dim4Avg == lowestScore, "Dimension 4", result);
result = if (dim3Avg == lowestScore, "Dimension 3", result);
result = if (dim2Avg == lowestScore, "Dimension 2", result);
result = if (dim1Avg == lowestScore, "Dimension 1", result);
```

Example: Indicate "Low", "Medium", and "High" Student-Teacher ratios.

Use a variable to record the student-teacher ratio. Then set variables to mark the points at which the ratio is considered medium or high. Finally, use two `if (cond, trueValue, falseValue)` blocks to return the "Low", "Medium" or "High" text based on the ratio value.

```
stRatio = [Distinct Count: Student ID] / [Distinct Count: Teacher ID];
mediumThreshold = 20;
highThreshold = 40;

if (stRatio < mediumThreshold,
    "Low",
    if (stRatio >= highThreshold,
        "High",
        "Medium"
    )
)
```

Example: Add styling when indicating the "Low", "Medium", and "High" Student-Teacher ratios.

This example is the same as above, but applies green, red, and yellow colors to the text. It also makes the word "High" appear in bold formatting.

```
stRatio = [Distinct Count: Student ID] / [Distinct Count: Teacher ID];
mediumThreshold = 20;
highThreshold = 40;

if (stRatio < mediumThreshold,
    "<span style=\"color:green\">Low</span>",
    if (stRatio >= highThreshold,
        "<span style=\"color:red;font-weight:bold\">High</span>",
        "<span style=\"color:yellow\">Medium</span>"
    )
)
```

Example: Determine the proficiency level of the average score.

Compare the average score of the 1st essay submission to what the desired proficiency level (e.g. 4). Then use an `if (cond, trueValue, falseValue)` block to return the text "Below" if the average score is less than 4, or to return "Above" if the score is equal to or greater than 4.

```
profLevel = 4;

if ([Average: Essay 1st Submission Score] < 4,
    "Below",
    "Above"
);
```

Example: Indicate Benchmark Levels Using Percent Score

Use a variable to record the percent score. Then set variables to mark the points at which the benchmark level is considered a 1 or higher. Finally, use an `if (cond, trueValue, falseValue)` blocks to return the 1, 2, 3, 4 values text based on the benchmark level value.

```
score= [Average: Objective Percent Score];
b1=25;
b2=50;
b3=75;
b4=100;
if(score<=b1,"1",if(score<=b2,"2",if(score<=b3,"3",if(score<=b4,"4","N/A"))))
;
```

Example: Indicate whether the proficiency level has "Improved", remained "Unchanged", or "Got Worse" from the first essay submission compared to the most recent.

Use a variable to set the desired proficiency level (e.g. 4). Then create two variables, `before` and `after` which indicate the proficiency level of the 1st essay submission and most recent submission, respectively. Finally, use nested conditional blocks to return "Improved" if the proficiency went from "Below" to "Above", "Unchanged" if it remained the same, or "Got Worse" if it went from "Above" to "Below".

```
profLevel = 4;

before = if ([Average: Essay 1st Submission Score] < profLevel,
    "Below",
    "Above"
);

after = if ([Average: Essay Most Recent Score] < profLevel,
    "Below",
    "Above"
);

if (before  == "Below" && after == "Above",
    "Improved",
    if (before == "Above" && after == "Below",
        "Got Worse",
        "Unchanged"
    )
)
```

Using Metrics from Different Reports

Referring to Other Reports

By default, metrics used within a custom expression always use the current report as a filter of the data. However, by qualifying the metric with the name of a report, you can force the filters of that report to be applied when calculating the metric.

For example, this represents the average score within the current report:

```
[Average: Score]
```

This represents the average score in the state of Utah, regardless of what the current report is: (assuming there is a report named "Utah Students")

```
[Average: Utah Students: Score]
```

Example: Percentage of students who are proficient.

This expression requires that the **Proficient Students** report first be created. This report must contain what you would consider to be the proficient students, such as those with an "Average Most Recent Submission Score" between 4 and 6.

Caution on filtering by aggregates.

You can not yet filter by an aggregate that is calculated on the fly. So if you have a report with a **Score** field, you can not create a filter that says "students who had an **Average Score** >= 4". You will need to have the **Average Score** as a pre-computed field in the data set.

Be careful of what filters are being compared.

In this example, `countOfMyStudents` refers to all students within the current report. Keep in mind that the **Proficient Students** report may only be applying a filter against the **Score** field. Therefore, if you apply this expression to a report named **Utah Students**, but the **Proficient Students** report does not also filter by Utah, then you will be comparing the number of Utah students to all Proficient students. Currently, the solution is to create a **Utah Proficient Students** report.

```
countOfProficientStudents = [Distinct Count: Proficient Students: Student ID];
countOfMyStudents = [Distinct Count: Student ID];
percentageProficient = round(countOfProficientStudents / countOfMyStudents * 100, 2)
```

Example: Measuring distance from a desired goal, expressed as a ratio.

Here we are taking the ratio of the current percentage of proficiency compared to the desired percentage.

For example, if currently 20% of students are proficient, and our goal is 80%, we could say we are 25% of the way to our goal (that is, 20 / 80).

```
countOfProficientStudents = [Distinct Count: Proficient Students: Student ID];
countOfMyStudents = [Distinct Count: Student ID];
percentageProficient = round(countOfProficientStudents / countOfMyStudents * 100, 2);
percentageGoal = 80;
round(percentageProficient / percentageGoal * 100, 2)
```

Example: Measuring distance from a desired goal, expressed as a difference.

Here we are taking the difference between our current percentage of proficiency and the desired percentage.

If currently 20% of students are proficient, and our goal is 80%, we could also say that we need 60% more students to become proficient to meet our goal (80 - 20).

```
countOfProficientStudents = [Distinct Count: Proficient Students: Student ID];
countOfMyStudents = [Distinct Count: Student ID];
percentageProficient = round(countOfProficientStudents / countOfMyStudents * 100, 2);
percentageGoal = 80;
round(percentageGoal - percentageProficient, 2)
```

Using Overall Metrics from Different Reports

Referring to Overall Calculations

By default, when a custom expression is applied to a pivot table, the calculation will be aggregated against each unique set of values that are being grouped on. However, in some cases you may want to compare these aggregates with an overall aggregate. You can refer to the overall aggregate of any metric by including the word "Overall" at the beginning of the name.

In this example, this calculation will be aggregated according to how the expression is applied to a pivot table:

```
[Average: Score]
```

Now in this example, we explicitly desire to use the overall average score, regardless of how this expression is being grouped within a pivot table:

```
[Overall Average: Score]
```

The power of using the overall aggregate becomes clear in the example below.

Example: Benchmarking my schools' performance against state-wide or application-wide performance.

In these examples, **All Sessions** is a report that applies no filters so it includes all scores within the application. **Utah Students** is a report that filters by the state of Utah, so only students from Utah are included.

Application-wide comparison:

```
appwideAverage = [Overall Average: All Sessions: Essay Score];
reportAverage = [Average: Essay Score];
round(reportAverage - appwideAverage, 2)
```

State-wide comparison:

```
stateAverage = [Overall Average: Utah Students: Essay Score];
reportAverage = [Average: Essay Score];
round(reportAverage - appwideAverage, 2)
```

Handling Missing Input Values

Some of your data may contain missing values. If so, there is an isNull() function that can be used to change the outcome of the expression based on the presence of a missing value. The following example generates a text results based on the Age field but does not handle missing values:

```
if (
    [Age] >= 50,
    "Over the Hill",
    "Under the Hill"
)
```

But if any records were missing a value for Age, then the expression would throw an error. The following improvement handles missing values:

```
if (
    isNull([Age]),
    "Don't Know",
    if (
        [Age] >= 50,
        "Over the Hill",
        "Under the Hill"
    )
)
```