Troubleshoot Invalid Calculated Columns

When you add a calculated column to a dataset, you use the **Create a Calculated Column** dialog. Before you can close this dialog and save a calculated column, AtScale tests the validity of the formula. If the formula is invalid, you must fix it and retest it before you can save the column. This section describes the most common causes of invalid formulas.

The Formula Does Not Use Syntax That Is Valid For The SELECT List Of A Query

Any formula you use must be valid in the column `SELECT` list of a query, such as this:

```
SELECT your_formula_here FROM current_dataset LIMIT 1;
```

Try executing this query directly through your data warehouse engine to see if it succeeds.

The Arguments Of The SQL Function Don't Match The Expected Data Types

SQL functions expect input arguments to be of certain data types. For example, the `CONCAT` function expects inputs to be a `STRING`.

Look at the function's arguments and verify that they use the proper data types. If the argument is a column, you might need to cast it to the correct data type within the formula itself.

The Calculated Column Uses Other Calculated Columns

Formulas can operate only on the original dataset columns, not on other calculated columns.

The Calculated Column Uses Columns Of Other Datasets

Formulas can refer only to columns in the current dataset, not in other datasets.

Column Names Are Not Escaped

Column names used in a formula must be enclosed in quotes if they contain spaces, special characters, reserved keywords, or if they start with numeric characters. For example, when using a `date` column, instead of `cast(date as varchar(20))` you should use `cast("date" as varchar(20))`.

Consider that database engines have different ways of quoting. Make sure you use the quotation rules for your database.

It is recommended to avoid naming columns after protected keywords.

Literal Strings Are Not Enclosed In Double Quotation Marks

You can include a literal string value as a function argument, but it must be enclosed in double quotation marks (`"`). For example, the literal comma in this example is enclosed in double quotation marks:

```
CONCAT("lastname",",","firstname")
```

Special Characters Are Not Escaped

Input values can contain special characters that need to be escaped. For example, literal string values that contain a double quotation mark (`"`), must be escaped by a pair of double quotation marks (`""`).

For example, suppose you want to concatenate the strings `"Hello` and `world."` to make the string `"Hello world."`. The double quotation marks in each string are special characters and must be escaped in the formula:

```
CONCAT("""Hello", " world.""")
```

The Syntax Of The Function Is Invalid

SQL functions have specific requirements, including required arguments and keywords. For example, the `CASE` function requires the `END` keyword to complete the formula.

Make sure you are using the syntax expected by the data warehouse engine that you are using.