

Customize Your Data Analysis: A Deep Dive into Calculated Fields

Smart ways to Customize your data with Calculated Fields.





Table of Contents

1. Introduction	02
2. Categories in Calculated Fields	02
○ 2.1 Dates	02
○ 2.2 Dimensions	03
○ 2.3 Metrics	03
3. Creating a Calculated Field: A Step-by-Step Guide	04
4. Exploring Functions and Syntax	09
○ 4.1 Arithmetic Operators	09
○ 4.2 Comparison Operators	10
○ 4.3 Date and Time Functions	11
○ 4.4 String Functions	12
○ 4.5 Pattern Matching Functions	16
5. Conclusion	16



I. Introduction

In the ever-evolving landscape of business analytics, the ability to extract meaningful insights from vast amounts of data is paramount. Calculated Fields in ConverSight platform emerge as a fundamental aspect of analytical process, offering users the power to tailor their data analysis to specific business requirements. Calculated Field is a custom field created by applying filters or some operations on existing fields to gain unparalleled flexibility in examining and interpreting the data, enabling users to make informed decisions and driving business growth.


This datasheet explores the essential role of Calculated Fields in the analysis and visualization of business data in the ConverSight platform. It highlights how Calculated Fields empower users to construct personalized metrics, formulas and KPIs (Key Performance Indicators) that align with their unique analytical needs. By harnessing the capabilities of Calculated Fields, businesses can unlock a new level of data exploration and gain insights that transcend the limitations of pre-defined fields. Whether it's evaluating performance, tracking trends, or uncovering correlations, the creation of custom fields through Calculated Fields empowers users to unlock the full potential of their data.

2. Categories in Calculated Fields

Calculated Fields offer users a powerful toolset for refining their search fields and organizing data in a structured and meaningful way. When utilizing Calculated Fields, users can categorize Custom field into three key aspects: Metrics, Dimensions, and Dates. These categorization options provide enhanced flexibility and precision when analyzing and visualizing data within the ConverSight platform. Calculated Fields are custom columns the user can create from existing columns in the dataset.

2.1 Dates

The Date option within Calculated Fields equips users with powerful capabilities to perform date-related operations on their Fields. This functionality proves particularly valuable when working with time-series data, enabling users to derive valuable insights and calculate key metrics, such as year-over-year growth or month-to-date sales. By leveraging the Date option, users can unlock a range of sophisticated calculations and comparisons based on temporal information.

-  **Time-Series Analysis:** Time-series data, characterized by its chronological order, is commonly encountered in business analytics. The Date option enables users to analyze and explore data by manipulating dates, extracting relevant components (e.g., year, month, day); and aggregating information across specific time periods. This allows for the identification of trends, patterns and seasonality, supporting informed decision-making and predictive modeling.



2.2 Dimensions

The Dimension option within Calculated Fields provides users with the capability to perform conditional operations on columns, allowing for the slicing and dicing of data to examine trends, patterns, and relationships. This powerful functionality enables users to gain deeper insights by grouping data based on specific categories such as region, product, or customer. This facilitates granular analysis, providing users with a comprehensive understanding of their data and enabling targeted strategies and decision-making.

- 🌱 **Conditional Operations:** Calculated Fields with the Dimension option provide users with the ability to perform conditional operations on columns. This allows for dynamic data manipulation and the creation of custom segments or subsets based on specific criteria. Through these conditional operations, users can analyze subsets of data and uncover patterns or behaviors that significantly impact the business's success or pose challenges.
- 🌱 **Advanced Analysis and Comparison:** The Dimension option facilitates advanced analysis by allowing users to compare data across different dimensions or categories. This capability enables users to evaluate performance and uncover relationships or correlations between different dimensions. For example, users can compare sales figures across different regions or compare customer satisfaction scores across various product categories. This comparative analysis empowers users to identify growth opportunities.



2.3 Metrics

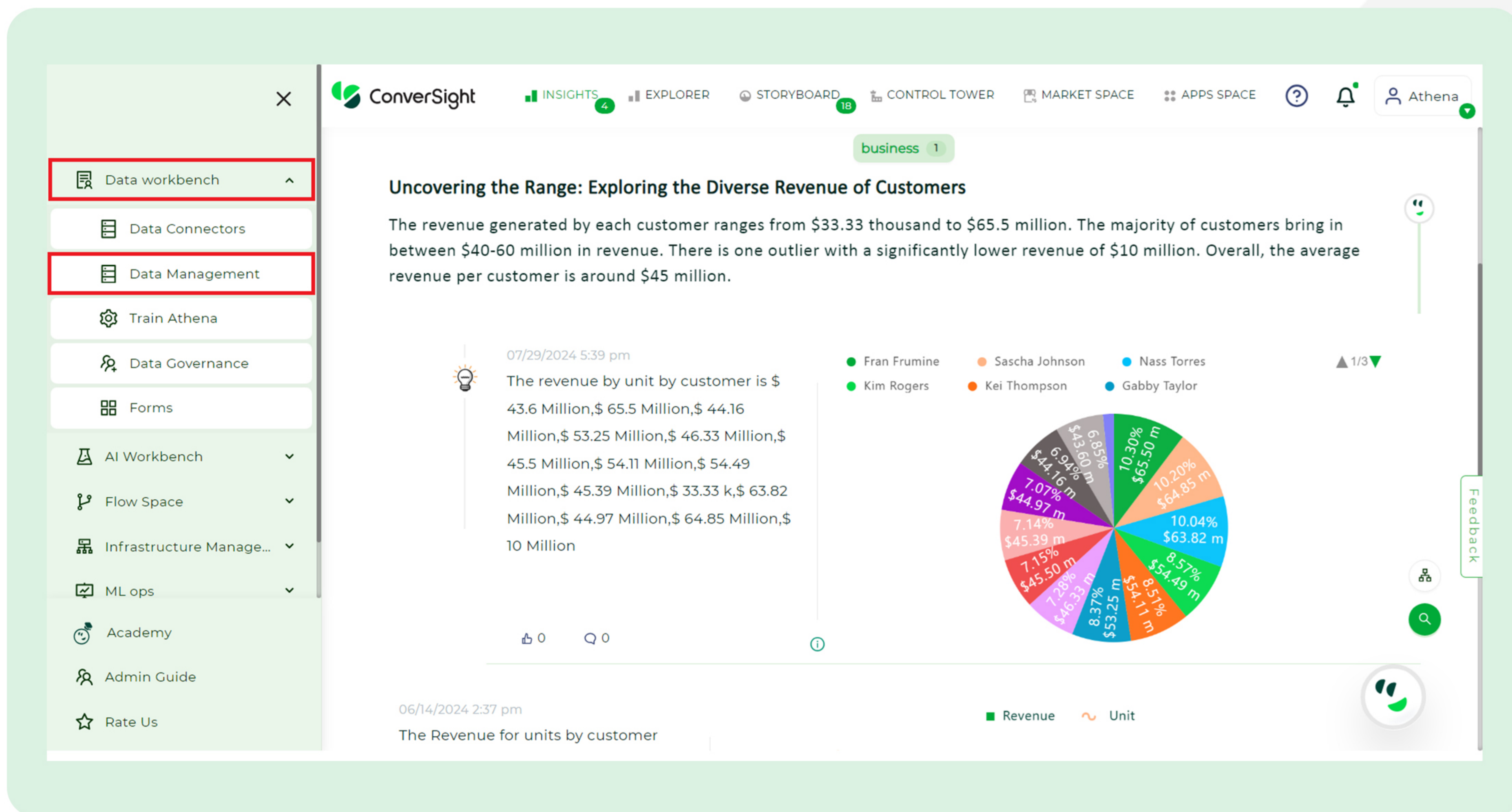
The Metric option within Calculated Fields empowers users to perform arithmetic operations in their fields, particularly when working with numerical data. This option proves invaluable for calculating aggregates such as sum, average and maximum, enabling users to gain insights and characteristics of their data.

- 🌱 **Arithmetic Operations:** The Metric option allows users to apply a wide range of arithmetic operations to their numerical data and combine multiple fields to create new calculated metrics. This flexibility supports complex calculations, enabling users to derive valuable insights from their data.
- 🌱 **Aggregates and Summary Statistics:** Through the Metric option in Calculated Fields, users can aggregate and summarize statistics, offering a comprehensive overview of their numerical data. By computing statistical indicators, users gain insights into the distribution and behavior of their data. These consolidated values provide a clear view, allowing users to identify anomalies that have an impact on business performance.
- 🌱 **Data Transformation:** The Metric option also facilitates data transformation by allowing users to convert and format numerical values. Users can apply rounding, formatting or scaling operations to make the data more presentable and understandable. This transformation enhances the clarity and visual appeal of numerical insights, enabling effective communication of findings and facilitating decision-making.

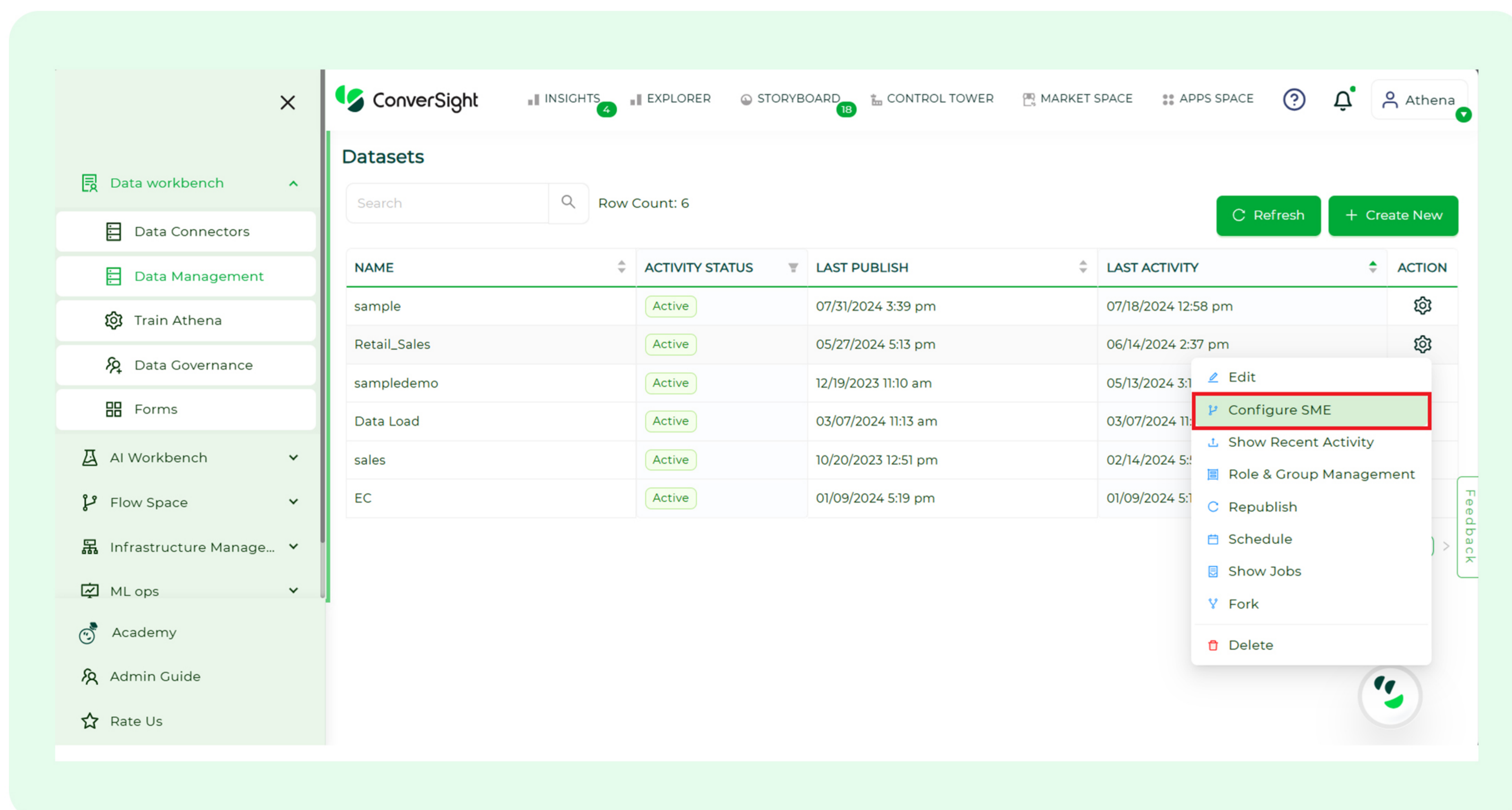
3. Creating calculated Fields: A Step-by-Step Guide

To create a Calculated Field within the platform, follow these simple steps after logging in:

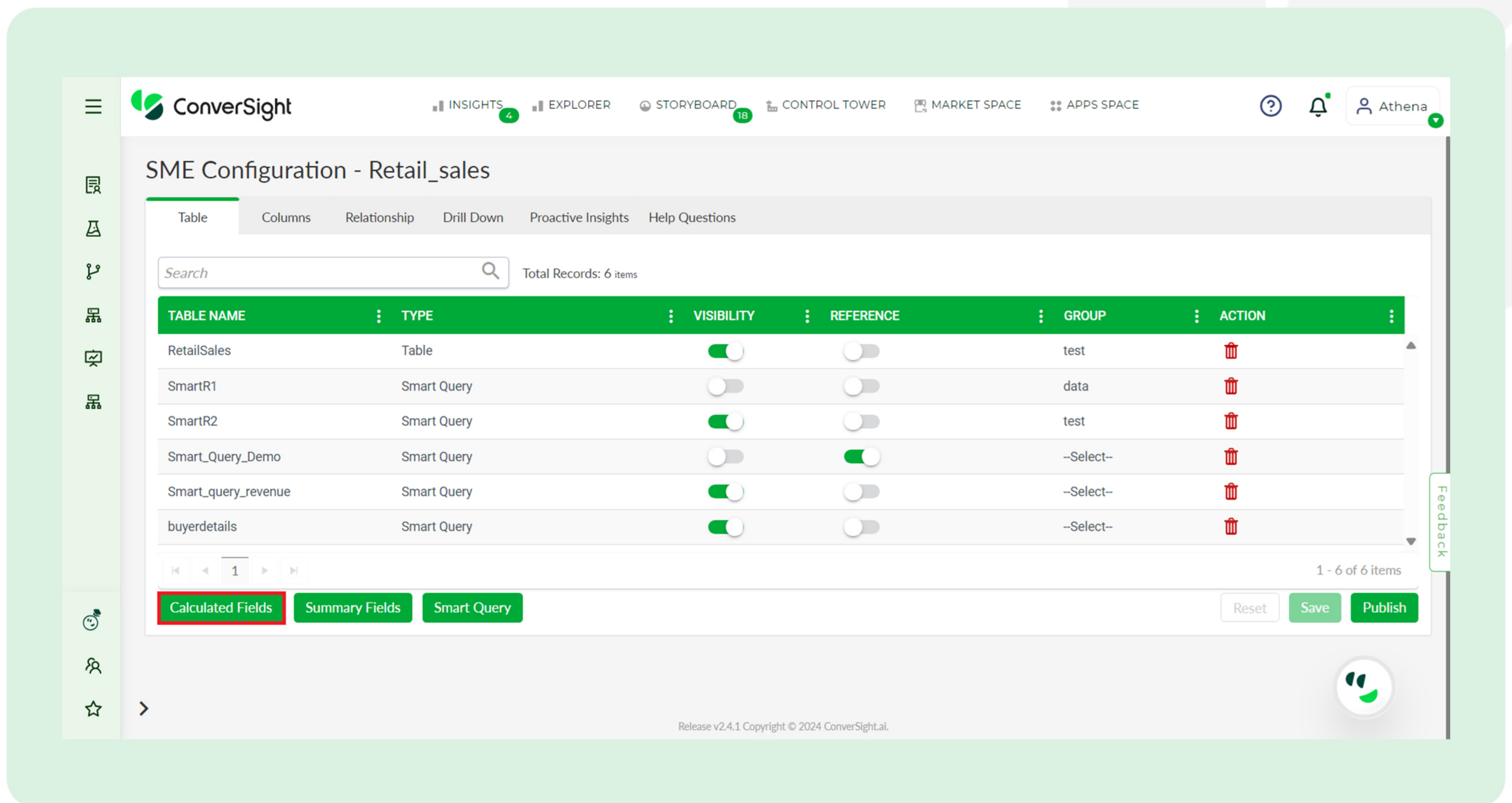
Step 1: Select the 'Data Management' option from the 'Data Workbench' menu.



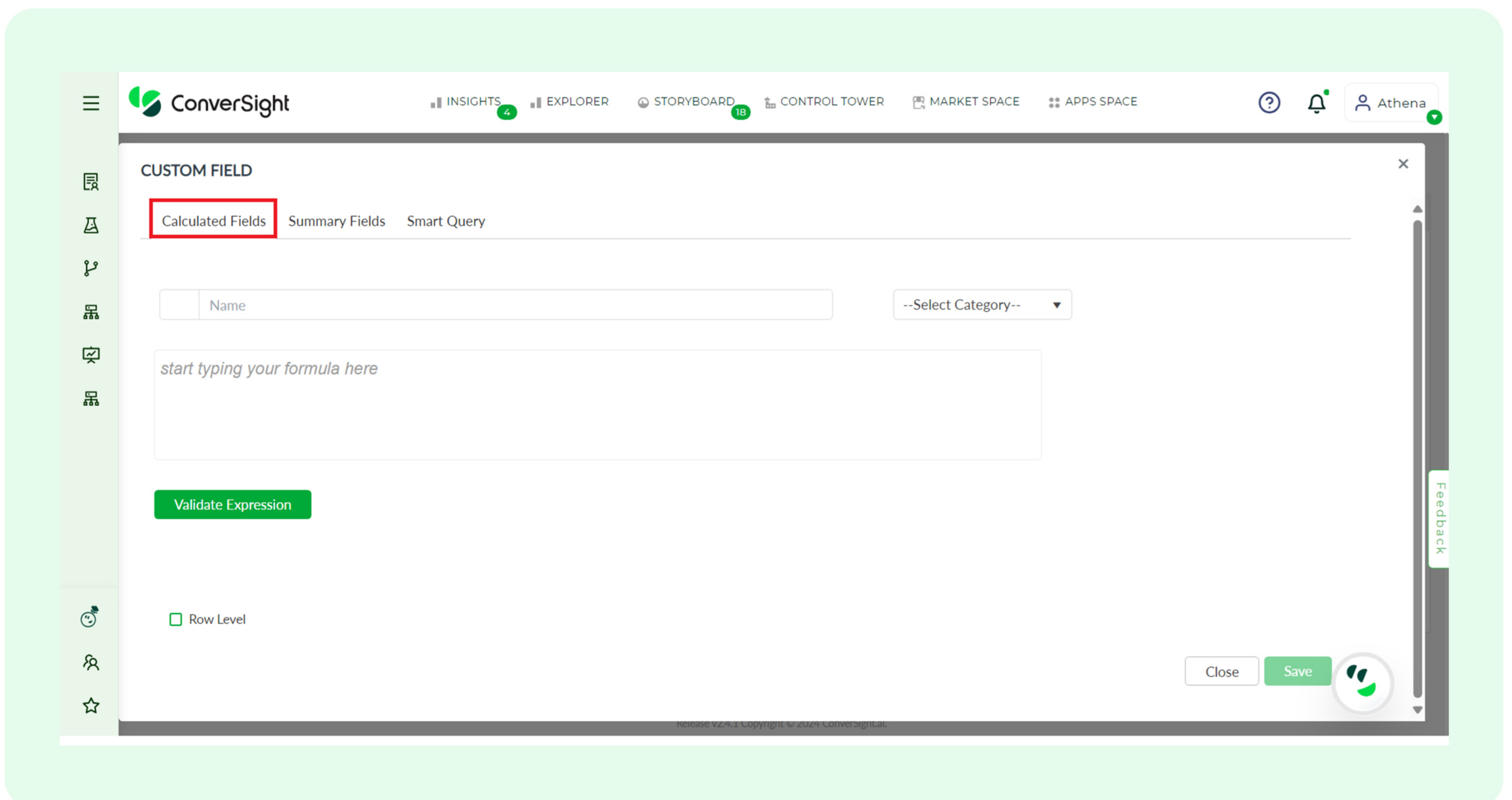
Step 2: Select the dataset for which you want to create Calculated Fields. Go to the 'Settings' icon in the Action column of the dataset and select 'Configure SME'.

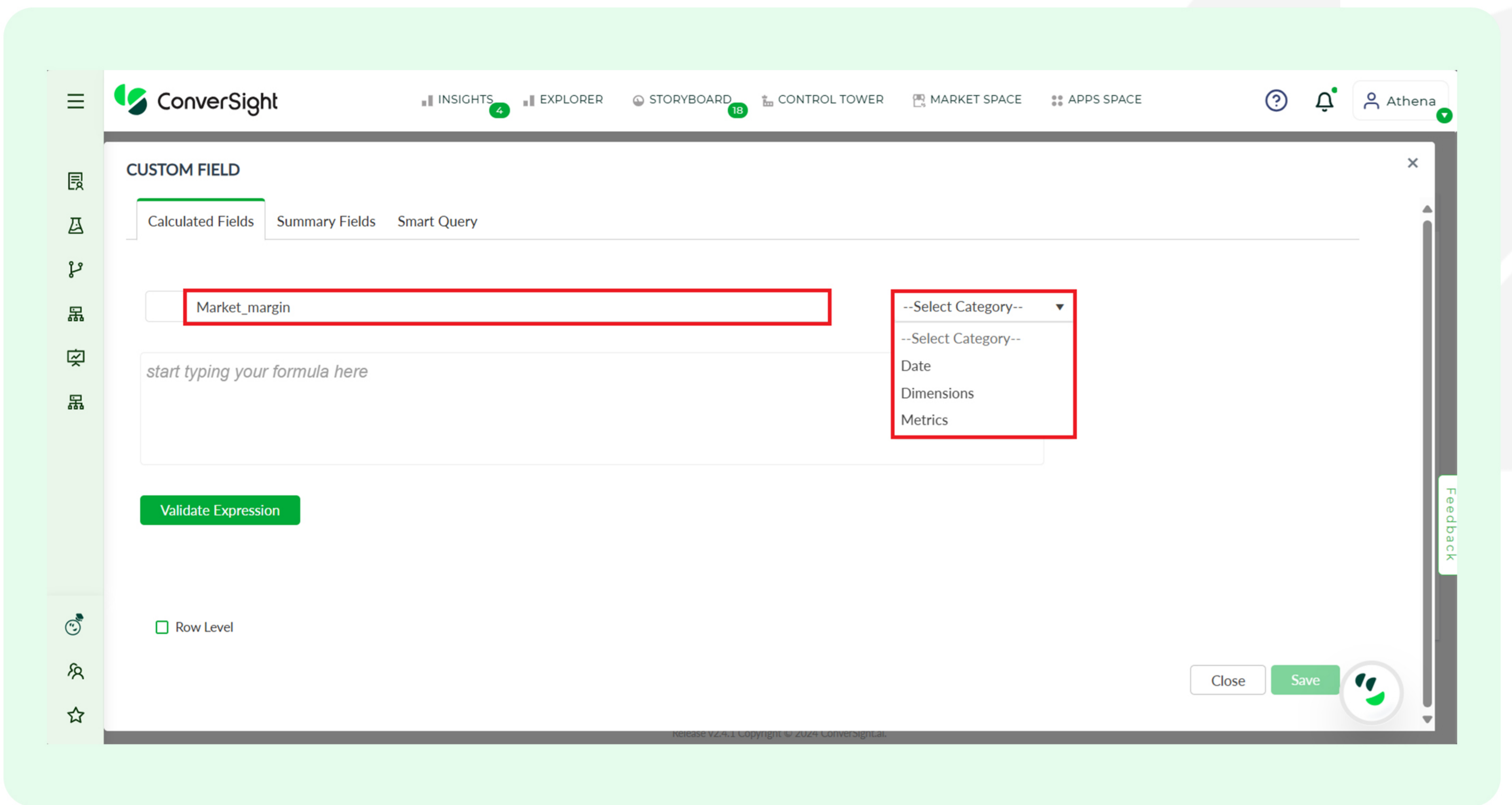


Step 3: Select the 'Calculated Fields' button in the SME Coaching page

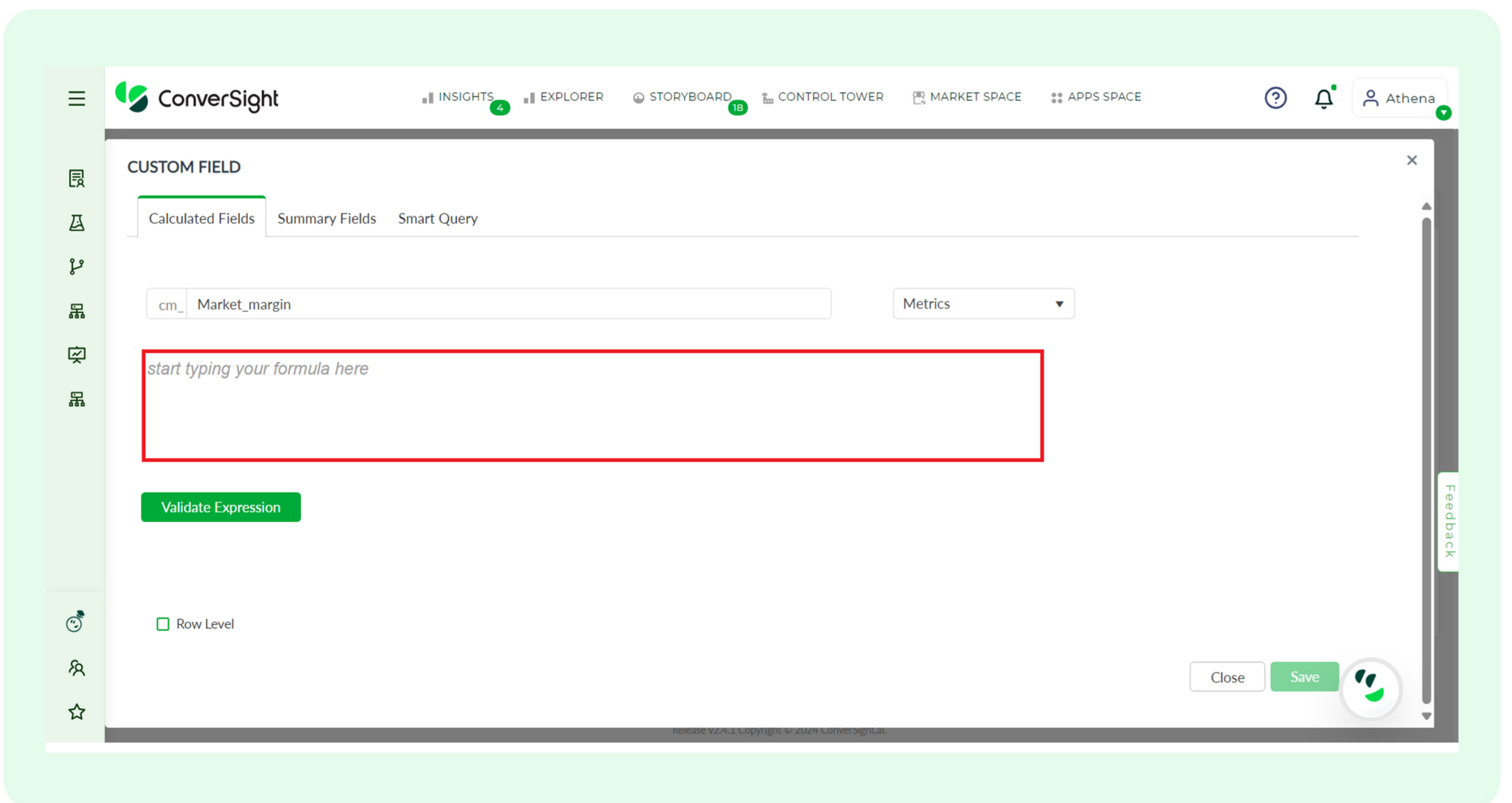


Step 4: On the Custom Field page, under the Calculated Fields tab. Provide a name for your Calculated Fields, then select the category from the 'Select Category' dropdown.

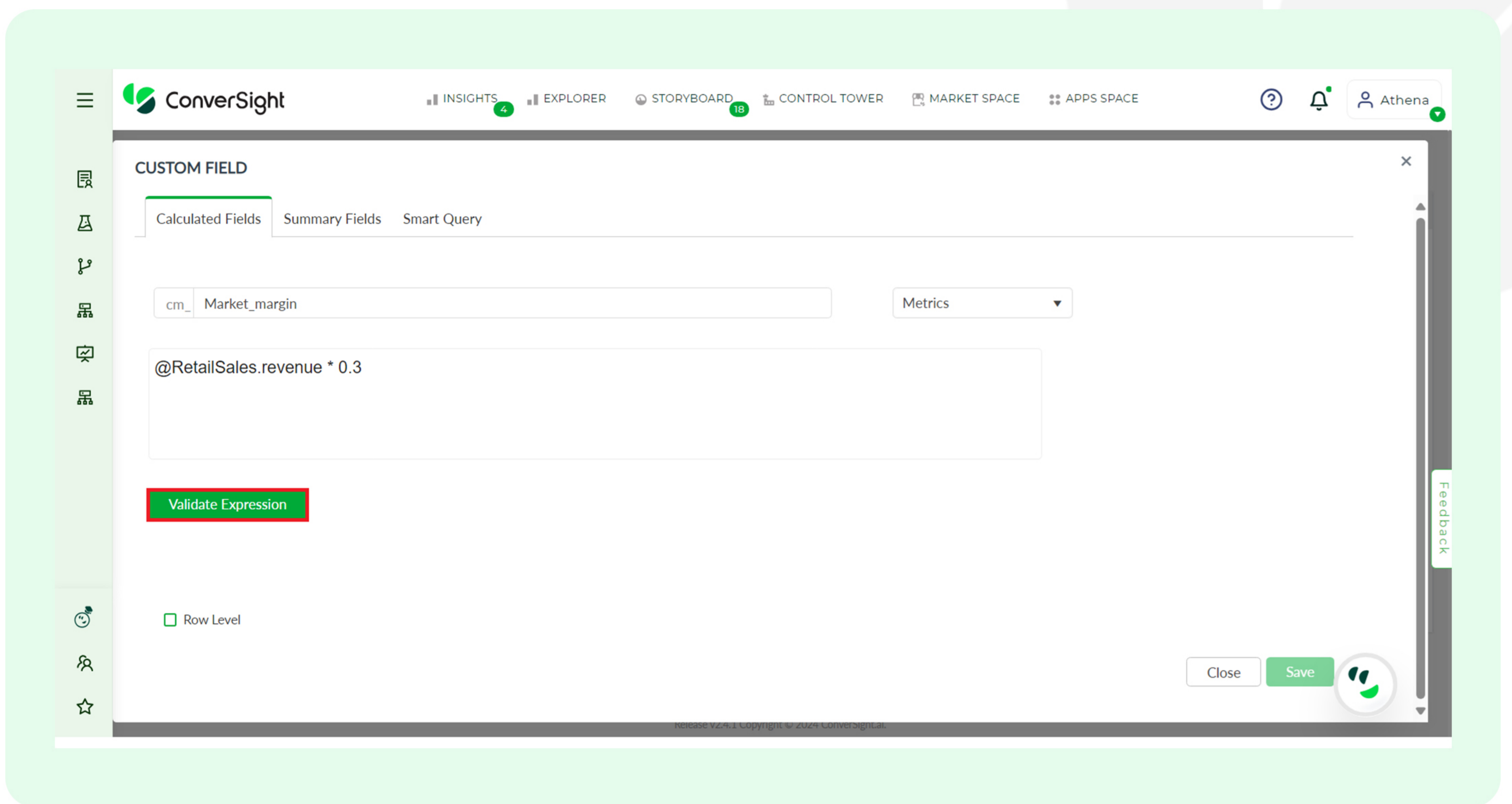




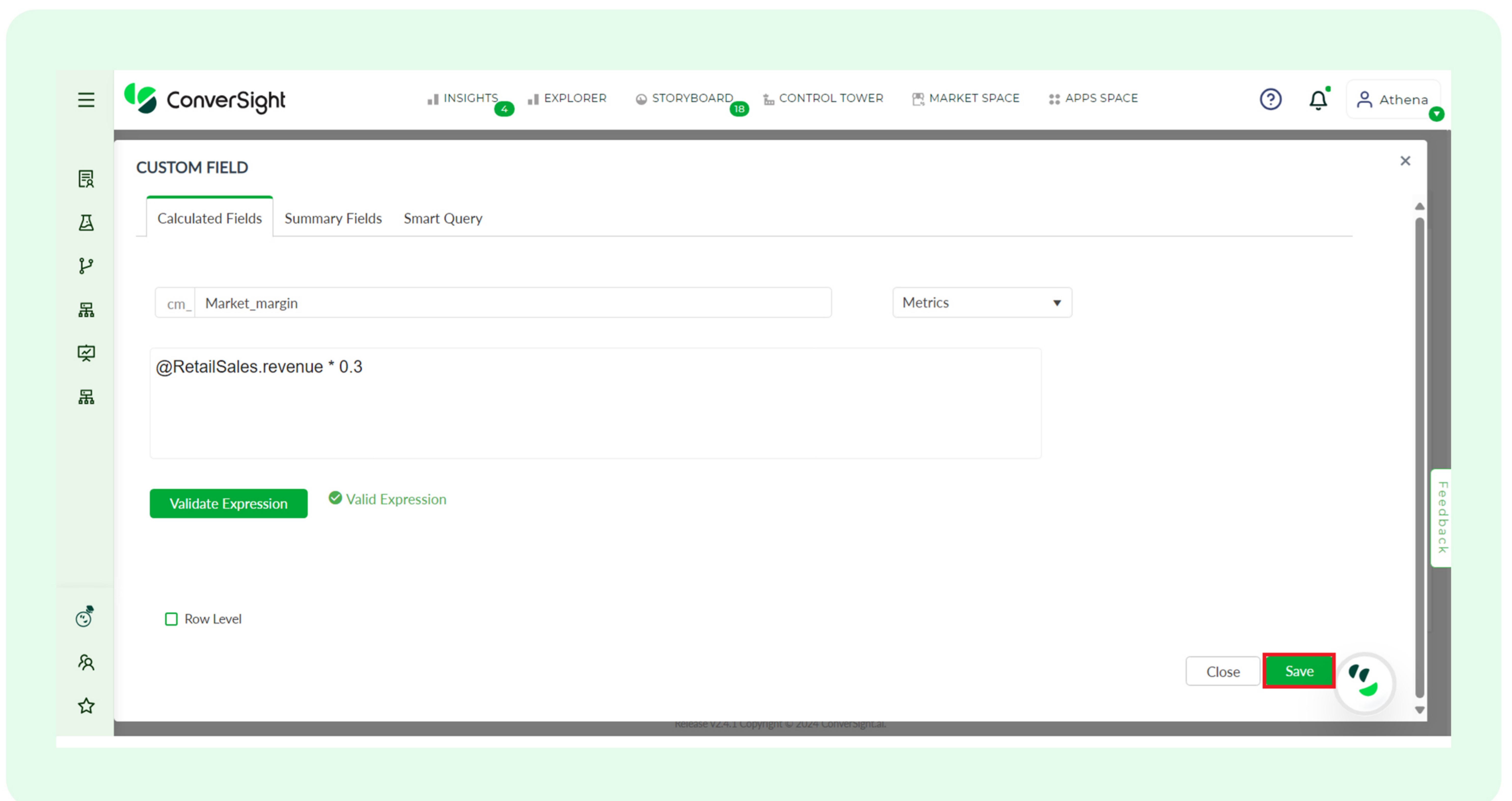
Step 5: Once you have named and categorized your Calculated Fields, you can enter your operations in the text box provided.



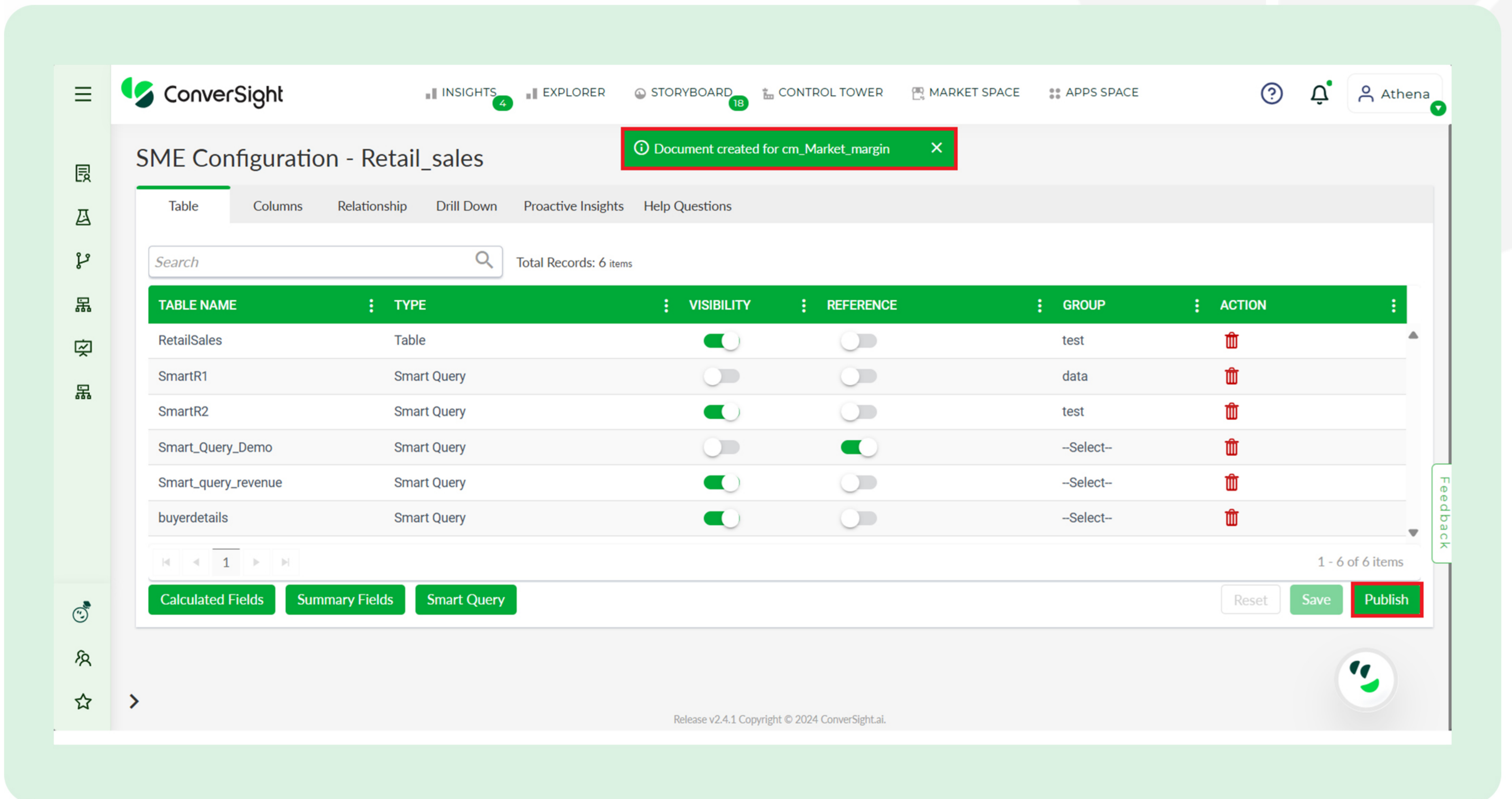
Step 6: You can validate your expression by clicking on the **'Validate Expression'** button.



Step 7: After validating your expression, you can save your Calculated Fields by clicking on the **'Save'** button.

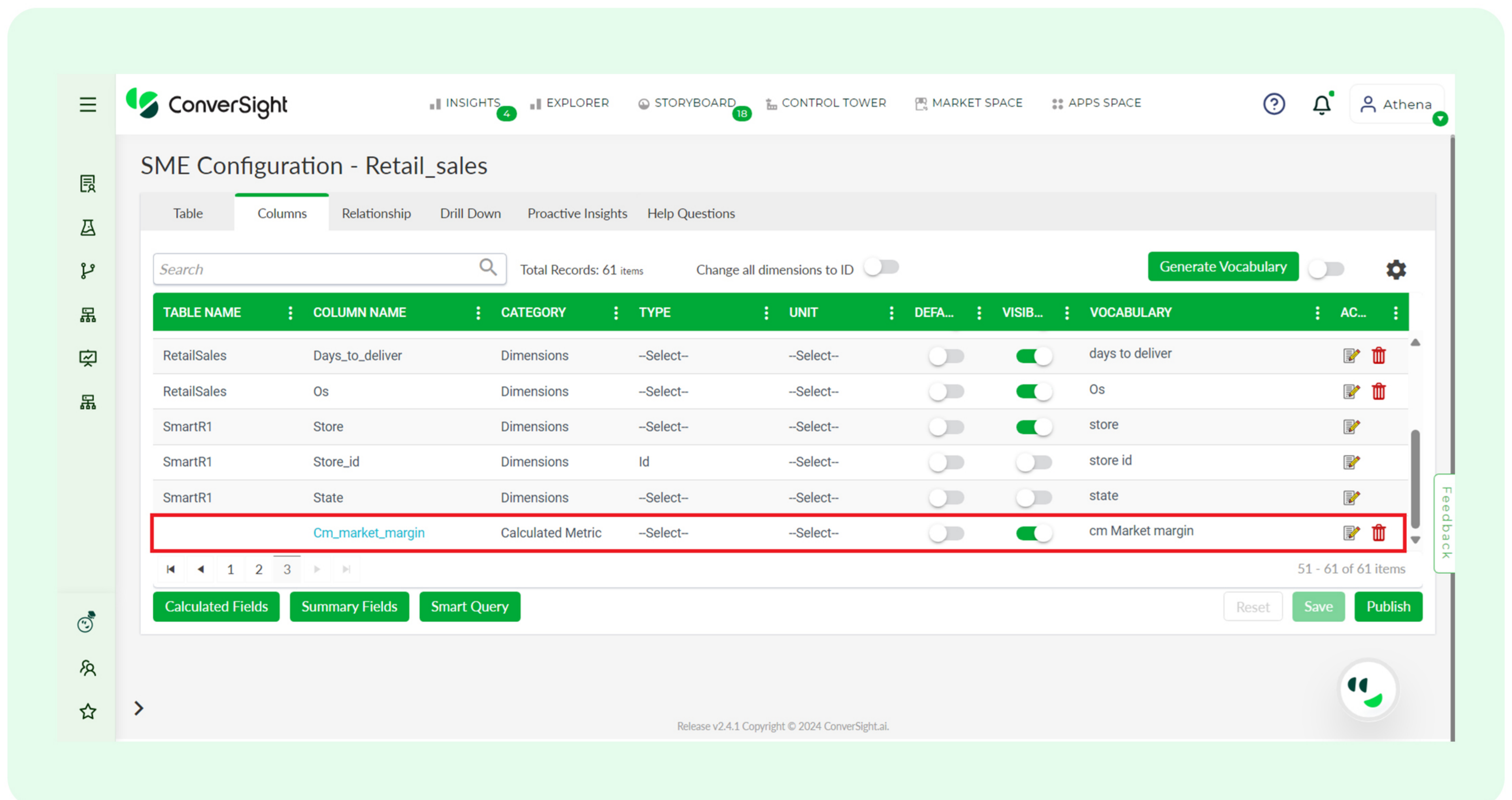


Step 8: The page will then redirect to the SME Coaching page, where you need to publish your Calculated Fields by clicking the **'Publish'** button.



Step 9: After publishing your Calculated Fields, you can access it in the data table. You will be able to perform analyses and gain insights based on the Calculated Fields you have created.

Step 10: The created Calculated Field will be displayed in the Columns tab of the SME Coaching page.



Step 11: You can access your calculated field by querying the created custom field name in Athena.

The screenshot displays the ConverSight interface. On the left, the 'SME Configuration - Retail_sales' window shows a table configuration with columns: 'Days_to_deliver', 'Os', 'Store', 'Store_id', 'State', and 'Cm_market_margin'. The 'Cm_market_margin' column is highlighted as a 'Calculated Metric'. On the right, the 'Athena Conversation - Retail_Sales' window shows a query result for 'Market margin' with a value of '\$635.99 m'.

4. Exploring Functions and Syntax

Within Calculated Fields, users can harness Arithmetic, comparison and a plethora of other functions for sophisticated analytical operations.

4.1 Arithmetic Operators

In Calculated Fields, users can leverage arithmetic operators to execute mathematical computations. The general syntax to utilize the operators is:

@TableName.Column1 operator @TableName.Column2

FUNCTION	DESCRIPTION
Addition (+)	Sum of two columns
Subtraction(-)	Difference of the columns
Multiplication(*)	Product of two columns
Division(/)	Returns the Quotient
Modulo(%)	Returns the Remainder



4.2 Comparison Operators

Comparison operators facilitate the comparison of two columns or a column with a specific value within a dataset. These operators assess the relationship between the elements being compared and typically yield a Boolean outcome, indicating whether the comparison is true or false. Through these operations, datasets can be filtered, segmented, and analyzed based on predetermined conditions, aiding in various analytical tasks such as data filtering, conditional aggregation, and trend analysis. The general syntax for using comparison operator in Calculated Field is:

```
CASE  
WHEN @TableName.Column1 Comparison Operator@TableName.Column2  
THEN result1  
ELSE result2  
END
```

FUNCTIONS

DESCRIPTION

Equal To (=)

Evaluates the equality of values within a designated column.

Not Equal To (!=) or (<>)

Evaluates the non-equality of values within a designated column.

Greater than (>)

Evaluates if the value on the left column is greater than the right column.

Greater than or equal to (>=)

Determines if the value on the left column is equal to or greater than the right column.

Less than (<)

Assess whether the value on the left column is less than the right column.

Less than or equal to (<=)

Assess whether the value on the left column is less than or equal to the right column.

Is null

Checks if the column has null values.

Is not null

Verifies if a value in the specified column is not null

Between

Checks if a value is within a given range, including both lower and upper bounds, in the designated column.

Not between

Checks if a value is outside a given range, excluding both lower and upper bounds, in the specified column.



4.3 Date and Time Functions

Date and time functions streamline the handling of temporal data by facilitating various actions such as extracting specific components, calculating durations, and adjusting timestamps. These functions simplify tasks associated with managing date and time information, enhancing efficiency in working with time-based data. The general syntax for using the date and time functions in the Calculated Fields:

Case

When (@TableName.DateColumnName condition 1 CURRENT_DATE())

Then 'result 1'

Else 'result 2'

End

Functions and Descriptions:

- **CURRENT_DATE()**
Returns the current date in the GMT time zone.
- **CURRENT_TIME()**
Provides the current time of day in the GMT time zone.
- **CURRENT_TIMESTAMP()**
Supplies the current time of day within the GMT time zone.
- **DATEADD('date_part', interval, date| timestamp)**
Provides the current timestamp in the GMT time zone.
- **DATEDIFF('date_part', date, date)**
Returns a date following the addition of a specified time or date interval.
- **DATEPART('date_part', date)**
The Datepart function is utilized to extract a particular part or component from a date or timestamp.
- **DATE_TRUNC(date_part, timestamp)**
Rounds a date or time down to a specific unit such as month, day or hour.
- **INTERVAL 'count' date_part**
Adds or subtracts a specified number of date_part units from a timestamp. Please note that '**count**' is enclosed in single quotes and can refer to either a timestamp or date column.
- **NOW()**
Provides the current timestamp in the GMT time zone, equivalent to using CURRENT_TIMESTAMP().

- **TIMESTAMPADD(date_part, count, timestamp | date)**

Increments a timestamp or date by a specified count of date_part units.

- **TIMESTAMPDIFF(date_part, timestamp1, timestamp2)**

Timestamp diff function computes the variance between two timestamps, considering a designated date part.

- **EXTRACT(date_part FROM timestamp)**

Extract function retrieves components (such as year, month, or day) from a complete date or timestamp.



4.4 String Functions

String functions facilitate effective manipulation of textual data, encompassing operations such as concatenation, searching and formatting. These functions streamline various tasks related to text processing, enabling users to concatenate strings, search for specific substrings within larger texts, and format text according to predefined patterns or requirements. Overall, string functions play a crucial role in data manipulation and analysis, particularly when dealing with textual information. The general syntax for using string function is :

FunctionName (@TableName.ColumnName)

Functions and Descriptions

- **BASE64_ENCODE (str)**

Encodes a string to a BASE64-encoded string.

- **BASE64_DECODE (str)**

Decodes a BASE64-encoded string.

- **CHAR_LENGTH (str)**

Char_length function returns the number of characters in a string.

- **str1||str2**

Combines specified strings, automatically converting numeric, date, timestamp, and time types to strings as required. Thus, explicit casting of non-string types is unnecessary when using the concatenation operator with these inputs.

INITCAP (str)

Creates a string where the first letter, along with any following letters after specified delimiter characters, are capitalized, while the rest are lowercase. Accepted delimiters include various symbols and punctuation marks.

KEY_FOR_STRING (str)

Key_for_string function returns the dictionary key of a dictionary-encoded string column.

LCASE (str)

Returns the string in lowercase, with current support limited to the ASCII character set. Equivalent to the LOWER function.

LENGTH (str)

Length function returns the length of a string in bytes.

LOWER (str)

Returns the string in lowercase, with current support limited to the ASCII character set. Equivalent to the LCASE function.

LEFT (str, num)

Provides the leftmost specified number (num) of characters from the string (str).

LPAD (str, len, lpad_str)

Left pads the string (str) to achieve a total length of len, using the specified padding string (lpad_str) or the space character if not provided. If the length of str exceeds len, characters from the end are truncated to match the length of len. If necessary, lpad_str is repeated to meet the target length.

LTRIM (str, chars)

Removes leading characters defined in chars from the string. Functions identically to the TRIM operation.

OVERLAY (str PLACING replacement_str FROM start FOR len)

This operation substitutes a specified number of characters (len) in the given string (str) starting from the specified position (start) with the characters defined in replacement_str. If start is negative, it indicates the number of characters counted from the end of str. If the combined length of start and replacement_str exceeds the length of str, all characters from the start position to the end of str are replaced.

**POSITION(search_str
IN str FROM
start_position)**

Provides the position of the initial character in the search_str within the str, with the option to commence the search from start_position. Returns 0 if search_str is not located. Returns null if either search_str or str is null.

REPEAT (str, num)

Duplicates the string based on the specified repetition count(num).

**REPLACE (str,
existing_substr,
new_str)**

Updates the string by replacing every instance of the substring from_str with the new substring new_str.

REVERSE (str)

Reverses the given string.

**RPAD (str, len,
rpad_str)**

Right-pads the string (str) to achieve a total length of len using the specified padding string (rpad_str) or space characters if not provided. If the length of str exceeds len, characters from the start are truncated to match the length of len. The padding process involves adding characters from rpad_str sequentially until the desired length len is reached. If necessary, rpad_str is repeated to meet the target length.

RTRIM (str)

Removes any trailing spaces from the string.

**SPLIT_PART (str,
'delim', field_num)**

Divide the string using a specified delimiter (delim) and retrieve the field identified by the field number (field_num). Fields are numbered sequentially from left to right.

**STRTOK_TO_ARRAY
(str, 'delim')**

Splits the string, str, into tokens using optional delimiter(s), delim and provides an array of these tokens. If no tokens are generated during the process, an empty array is returned. If either parameter is NULL, the result is also NULL.

**SUBSTR (str, start, len)
or SUBSTRING (str
FROM start FOR len)**

Returns a substring of the string (str) starting at the specified index (start) and extending for a specified number of characters (len). The indexing starts from 1, where the first character of str is at index 1. However, start 0 is equivalent to start 1. If start is negative, it indicates the number of characters counted from the end of the string. If len is not provided, the substring from start to the end of str is returned. If start + len exceeds the length of str, the substring from start to the end of str is returned.

**TRIM(BOTH | LEADING
| TRAILING trim_str
FROM str)**

Removes characters defined in trim_str from the beginning, end, or both of str. If trim_str is not specified, the default is the space character. If the trim location is not specified, characters defined in trim_str are trimmed from both the beginning and end of str.

TRY_CAST(str AS type)

TRY_CAST attempts to convert a string into a numeric, timestamp, date, or time format. If successful, it returns the converted value; otherwise, if the string isn't in a valid format for the chosen type, it returns null. Note that TRY_CAST is exclusively for string conversions.

UCASE(str)

Provides the string in uppercase format, with current support limited to the ASCII character set. Equivalent to the UPPER function.

UPPER (str)

Returns the string in uppercase, with current support limited to the ASCII character set. Equivalent to the UCASE function.



4.5 Pattern Matching Functions

Pattern matching functions in SQL enable the search for specific patterns or substrings within textual or character data. They are widely used for querying, filtering, or manipulating data based on pattern matches in strings. The general syntax for Pattern Matching function is :

```
Case
When @TableName.ColumnName FunctionName '%Pattern%'
Then result1
Else result2
End
```

FUNCTIONS	DESCRIPTION
like	Returns true if the string matches the pattern.
not like	Returns true if the string does not match the pattern.
ilike	Used for case-insensitive pattern matching, ignoring letter case.
regexp	Used for complex text pattern matching with POSIX regular expressions.
regexp_like	Checks if a string matches a specified POSIX regular expression pattern.

5. Conclusion

In conclusion, Calculated Fields in ConverSight platform enable users to tailor their data manipulation, calculations and insights to their unique business needs by providing the ability to create custom fields based on specific logic. Through the flexibility offered by Calculated Fields, users can perform a wide range of operations, including mathematical calculations, string manipulations and date-based analyses. This enables the creation of customized metrics, dimensions, and date-related calculations that provide deeper insights and a granular understanding of their data. The ability to categorize fields using Metrics, Dimensions and Dates allows users to refine their searches, examine trends and identify patterns within their data.

Join our customers who have accelerated growth with ConverSight



About ConverSight

ConverSight's Adaptive Analytics platform uses conversational AI, Natural Language Processing and machine learning to converge the distance between humans and data through data stories, presenting the meaning of data in the most effective, personalized and efficient form possible. ConverSight's patented AI business assistant, Athena, connects distributed databases to answer questions and Augment the consumers through 4 key functions: Information on demand, Automated Story Telling, Proactive Insights, and Recommended Actions.

For more information, visit www.conversight.ai

