

MODE Function

Computes the mode (most frequent value) from all row values in a column, according to their grouping. Input column can be of Integer, Decimal, or Datetime type.

- If a row contains a missing or null value, it is not factored into the calculation. If the entire column contains no values, the function returns a null value.
- If there is a tie in which the most occurrences of a value is shared between values, then the lowest value of the evaluated set is returned.
- When used in a `pivot` transform, the function is computed for each instance of the value specified in the `group` parameter. See *Pivot Transform*.

For a version of this function computed over a rolling window of rows, see *ROLLINGMODE Function*.

Datetime inputs to this function return Unixtime values.

- These values can be wrapped in a `DATEFORMAT` function. See *DATEFORMAT Function*.
- For a date-native version of this function, see *MODEDATE Function*.

Wrangle vs. SQL: This function is part of Wrangle, a proprietary data transformation language. Wrangle is not SQL. For more information, see *Wrangle Language*.

Basic Usage

```
mode(count_visits)
```

Output: Returns the mode of the values in the `count_visits` column.

Syntax and Arguments

```
mode(function_col_ref) [group:group_col_ref] [limit:limit_count]
```

Argument	Required?	Data Type	Description
<code>function_col_ref</code>	Y	string	Name of column to which to apply the function

For more information on the `group` and `limit` parameters, see *Pivot Transform*.

For more information on syntax standards, see *Language Documentation Syntax Notes*.

`function_col_ref`

Name of the column the values of which you want to calculate the function. Column must contain Integer, Decimal, or Datetime values.

NOTE: If the input is in Datetime type, the output is in unixtime format. You can wrap these outputs in the `DATEFORMAT` function to generate the results in the appropriate Datetime format. See *DATEFORMAT Function*.

- Literal values are not supported as inputs.
- Multiple columns and wildcards are not supported.

Usage Notes:

Required?	Data Type	Example Value
Yes	String (column reference)	myValues

Examples

Tip: For additional examples, see *Common Tasks*.

Example - Statistics on Test Scores

This example illustrates how you can apply statistical functions to your dataset. Calculations include average (mean), max, min, standard deviation, and variance.

Source:

Students took a test and recorded the following scores. You want to perform some statistical analysis on them:

Student	Score
Anna	84
Ben	71
Caleb	76
Danielle	87
Evan	85
Faith	92
Gabe	85
Hannah	99
Ian	73
Jane	68

Transformation:

You can use the following transformations to calculate the average (mean), minimum, and maximum scores:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	AVERAGE(Score)
Parameter: New column name	'avgScore'

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	MIN(Score)

Parameter: New column name	'minScore'
Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	MAX(Score)
Parameter: New column name	'maxScore'

To apply statistical functions to your data, you can use the VAR and STDEV functions, which can be used as the basis for other statistical calculations.

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	VAR(Score)
Parameter: New column name	var_Score

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	STDEV(Score)
Parameter: New column name	stdev_Score

For each score, you can now calculate the variation of each one from the average, using the following:

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	$((\text{Score} - \text{avg_Score}) / \text{stdev_Score})$
Parameter: New column name	'stDevs'

Now, you want to apply grades based on a formula:

Grade	standard deviations from avg (stDevs)
A	stDevs > 1
B	stDevs > 0.5
C	-1 <= stDevs <= 0.5
D	stDevs < -1
F	stDevs < -2

You can build the following transformation using the IF function to calculate grades.

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	IF((stDevs > 1), 'A', IF((stDevs < -2), 'F', IF((stDevs < -1), 'D', IF((stDevs > 0.5), 'B', 'C'))))

For more information, see *IF Function*.

To clean up the content, you might want to apply some formatting to the score columns. The following reformats the `stdev_Score` and `stDevs` columns to display two decimal places:

Transformation Name	Edit column with formula
Parameter: Columns	stdev_Score
Parameter: Formula	NUMFORMAT(stdev_Score, '##.00')

Transformation Name	Edit column with formula
Parameter: Columns	stDevs
Parameter: Formula	NUMFORMAT(stDevs, '##.00')

Transformation Name	New formula
Parameter: Formula type	Single row formula
Parameter: Formula	MODE(Score)
Parameter: New column name	'modeScore'

Results:

Student	Score	modeScore	avgScore	minScore	maxScore	var_Score	stdev_Score	stDevs	Grade
Anna	84	85	82	68	99	87.000000000000001	9.33	0.21	C
Ben	71	85	82	68	99	87.000000000000001	9.33	-1.18	D
Caleb	76	85	82	68	99	87.000000000000001	9.33	-0.64	C
Danielle	87	85	82	68	99	87.000000000000001	9.33	0.54	B
Evan	85	85	82	68	99	87.000000000000001	9.33	0.32	C
Faith	92	85	82	68	99	87.000000000000001	9.33	1.07	A
Gabe	85	85	82	68	99	87.000000000000001	9.33	0.32	C
Hannah	99	85	82	68	99	87.000000000000001	9.33	1.82	A
Ian	73	85	82	68	99	87.000000000000001	9.33	-0.96	C

Jane	68	85	82	68	99	87.000000000 00001	9.33	-1.50	D
------	----	----	----	----	----	-----------------------	------	-------	---