has operator

Article • 02/02/2023

Filters a record set for data with a case-insensitive string. `has` searches for indexed terms, where an indexed term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

The following table compares the `has` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
has	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
!has	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
has_cs	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
!has_cs	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use the case-sensitive has_cs.

Syntax

T `|` `where` *Column* `has` `(`*Expression*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
Column	string	✓	The column used to filter the records.
Expression	scalar or tabular	✓	An expression for which to search. If the value is a tabular expression and has multiple columns, the first column is used.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State has "New"
| where event_count > 10
| project State, event_count
```

Output

State	event_count
NEW YORK	1,750
NEW JERSEY	1,044
NEW MEXICO	527
NEW HAMPSHIRE	394

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

has_cs operator

Article • 02/02/2023

Filters a record set for data with a case-sensitive search string. `has_cs` searches for indexed terms, where an indexed term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

The following table compares the `has` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
has	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
!has	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
has_cs	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
!has_cs	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

Syntax

T `|` `where` *Column* `has_cs` `(` *Expression* `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
Column	string	✓	The column used to filter the records.
Expression	scalar or tabular	✓	An expression for which to search. If the value is a tabular expression and has multiple columns, the first column is used.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State has_cs "FLORIDA"
```

Output

State	event_count
FLORIDA	1042

Since all `State` values are capitalized, searching for a lowercase string with the same value, such as "florida", won't yield any results.

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State has_cs "florida"
```

Output

State	event_count

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

!has operator

Article • 03/12/2023

Filters a record set for data that doesn't have a matching case-insensitive string. `!has` searches for indexed terms, where an indexed term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

The following table compares the `has` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
has	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
!has	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
has_cs	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
!has_cs	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use the case-sensitive !has_cs.

Syntax

T | `where` *column* `!has` `(` *expression* `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
column	string	✓	The column by which to filter.
expression	scalar	✓	The scalar or literal expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State !has "NEW"
| where event_count > 3000
| project State, event_count
```

Output

State	event_count
TEXAS	4,701
KANSAS	3,166

Feedback

!has_cs operator

Article • 03/12/2023

Filters a record set for data that doesn't have a matching case-sensitive string. `!has_cs` searches for indexed terms, where an indexed term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

The following table compares the `has` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
has	Right-hand-side (RHS) is a whole term in left-hand-side (LHS)	No	`"North America" has "america"`
!has	RHS isn't a full term in LHS	No	`"North America" !has "amer"`
has_cs	RHS is a whole term in LHS	Yes	`"North America" has_cs "America"`
!has_cs	RHS isn't a full term in LHS	Yes	`"North America" !has_cs "amer"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

Syntax

T `|` `where` *column* `!has_cs` `(`*expression*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
column	string	✓	The column by which to filter.
expression	scalar	✓	The scalar or literal expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State !has_cs "new"
| count
```

Output

Count
67

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

hasprefix operator

Article • 01/30/2023

Filters a record set for data with a case-insensitive starting string.

For best performance, use strings of three characters or more. `hasprefix` searches for indexed terms, where a term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

The following table compares the `hasprefix` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
hasprefix	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
!hasprefix	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`
hasprefix_cs	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
!hasprefix_cs	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use the case-sensitive hasprefix_cs.

Syntax

T `|` `where` *Column* `hasprefix` `(` *Expression* `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
Column	string	✓	The column used to filter.
Expression	string	✓	The expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State hasprefix "la"
| project State, event_count
```

State	event_count
LAKE MICHIGAN	182
LAKE HURON	63
LAKE SUPERIOR	34
LAKE ST CLAIR	32
LAKE ERIE	27
LAKE ONTARIO	8

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

hasprefix_cs operator

Article • 01/30/2023

Filters a record set for data with a case-sensitive starting string.

For best performance, use strings of three characters or more. `hasprefix_cs` searches for indexed terms, where a term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

The following table compares the `hasprefix` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
hasprefix	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
!hasprefix	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`
hasprefix_cs	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
!hasprefix_cs	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

Syntax

T `|` `where` *Column* `hasprefix_cs` `(`*Expression*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
Column	string	✓	The column used to filter.
Expression	string	✓	The expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Examples

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State hasprefix_cs "P"
| count
```

Count
3

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State hasprefix_cs "P"
| project State, event_count
```

State	event_count

State	event_count
PENNSYLVANIA	1687
PUERTO RICO	192
E PACIFIC	10

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

!hasprefix operators

Article • 01/30/2023

Filters a record set for data that doesn't include a case-insensitive starting string.

For best performance, use strings of three characters or more. `!hasprefix` searches for indexed terms, where an indexed term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

The following table compares the `hasprefix` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
hasprefix	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
!hasprefix	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`
hasprefix_cs	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
!hasprefix_cs	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use the case-sensitive !hasprefix_cs.

Syntax

T `|` `where` *Column* `!hasprefix` `(` *Expression* `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
Column	string	✓	The column used to filter.
Expression	string	✓	The expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State !hasprefix "N"
| where event_count > 2000
| project State, event_count
```

State	event_count
TEXAS	4701
KANSAS	3166
IOWA	2337
ILLINOIS	2022
MISSOURI	2016

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

!hasprefix_cs operator

Article • 03/12/2023

Filters a record set for data that doesn't have a case-sensitive starting string. `!hasprefix_cs` searches for indexed terms, where an indexed term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

Operator	Description	Case-Sensitive	Example (yields `true`)
hasprefix	RHS is a term prefix in LHS	No	`"North America" hasprefix "ame"`
!hasprefix	RHS isn't a term prefix in LHS	No	`"North America" !hasprefix "mer"`
hasprefix_cs	RHS is a term prefix in LHS	Yes	`"North America" hasprefix_cs "Ame"`
!hasprefix_cs	RHS isn't a term prefix in LHS	Yes	`"North America" !hasprefix_cs "CA"`

> ⓘ Note
>
> The following abbreviations are used in the above table:
>
> - RHS = right hand side of the expression
> - LHS = left hand side of the expression

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

Syntax

T `|` `where` *column* `!hasprefix_cs` `(`*expression*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
column	string	✓	The column by which to filter.
expression	scalar	✓	The scalar or literal expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State !hasprefix_cs "P"
| count
```

Output

Count
64

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

hassuffix operator

Article • 03/12/2023

Filters a record set for data with a case-insensitive ending string. `hassuffix` returns `true` if there is a term inside the filtered string column ending with the specified string expression.

The following table compares the `hassuffix` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
hassuffix	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
!hassuffix	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
hassuffix_cs	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
!hassuffix_cs	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use the case-sensitive hassuffix_cs.

> ⓘ Note

Text index cannot be fully utilized for this function, therefore the performance of this function is comparable to **endswith** function, though the semantics is different.

Syntax

T `|` `where` *Column* `hassuffix` `(` *Expression* `)`

Parameters

Name	Type	Required	Description
T	string	The tabular input whose records are to be filtered.	
Column	string	The column by which to filter.	
Expression	scalar	The scalar or literal expression for which to search.	

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State hassuffix "o"
| project State, event_count
```

Output

State	event_count
COLORADO	1654
OHIO	1233
GULF OF MEXICO	577

State	event_count
NEW MEXICO	527
IDAHO	247
PUERTO RICO	192
LAKE ONTARIO	8

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

hassuffix_cs operator

Article • 03/12/2023

Filters a record set for data with a case-insensitive ending string. `hassuffix_cs` returns `true` if there is a term inside the filtered string column ending with the specified string expression.

The following table compares the `hassuffix` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
hassuffix	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
!hassuffix	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
hassuffix_cs	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
!hassuffix_cs	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

> ⓘ Note
>
> Text index cannot be fully utilized for this function, therefore the performance of this function is comparable to **endswith_cs** function, though the semantics is different.

Syntax

T `|` `where` *column* `hassuffix_cs` `(` *expression* `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
column	string	✓	The column by which to filter.
expression	scalar	✓	The scalar or literal expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Examples

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State hassuffix_cs "AS"
| where event_count > 2000
| project State, event_count
```

Output

State	event_count
TEXAS	4701
KANSAS	3166

Feedback

!hassuffix operator

Article • 03/12/2023

Filters a record set for data that doesn't have a case-insensitive ending string. `!hassuffix` returns `true` if there's no term inside string column ending with the specified string expression.

The following table compares the `hassuffix` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
hassuffix	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
!hassuffix	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
hassuffix_cs	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
!hassuffix_cs	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use !hassuffix_cs - a case-sensitive version of the operator.

> ⓘ Note

Text index cannot be fully utilized for this function, therefore the performance of this function is comparable to **!endswith** function, though the semantics is different.

Syntax

T `|` `where` *column* `!hassuffix` `(`*expression*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
column	string	✓	The column by which to filter.
expression	scalar	✓	The scalar or literal expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State !hassuffix "A"
| where event_count > 2000
| project State, event_count
```

Output

State	event_count
TEXAS	4701
KANSAS	3166

State	event_count
ILLINOIS	2022
MISSOURI	2016

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

!hassuffix_cs operator

Article • 03/12/2023

Filters a record set for data that doesn't have a case-sensitive ending string. `!hassuffix_cs` returns `true` if there is no term inside string column ending with the specified string expression.

The following table compares the `hassuffix` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
hassuffix	RHS is a term suffix in LHS	No	`"North America" hassuffix "ica"`
!hassuffix	RHS isn't a term suffix in LHS	No	`"North America" !hassuffix "americ"`
hassuffix_cs	RHS is a term suffix in LHS	Yes	`"North America" hassuffix_cs "ica"`
!hassuffix_cs	RHS isn't a term suffix in LHS	Yes	`"North America" !hassuffix_cs "icA"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

> ⓘ Note
>
> Text index cannot be fully utilized for this function, therefore the performance of this function is comparable to **!endswith_cs** function, though the semantics is different.

Syntax

T `|` `where` *column* `!hassuffix_cs` `(`*expression*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
column	string	✓	The column by which to filter.
expression	scalar	✓	The scalar or literal expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State !hassuffix_cs "AS"
| where event_count > 2000
| project State, event_count
```

Output

State	event_count
IOWA	2337
ILLINOIS	2022
MISSOURI	2016

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

has_all operator

Article • 04/03/2023

Filters a record set for data with one or more case-insensitive search strings. `has_all` searches for indexed terms, where an indexed term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Syntax

T `|` `where` *col* `has_all` `(`*expression*`,` `...` `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input to filter.
col	string	✓	The column by which to filter.
expression	scalar or tabular	✓	An expression that specifies the values for which to search. Each expression can be a scalar value or a tabular expression that produces a set of values. If a tabular expression has multiple columns, the first column is used. The search will consider up to 256 distinct values.

Returns

Rows in *T* for which the predicate is `true`.

Examples

Set of scalars

The following query shows how to use `has_all` with a comma-separated set of scalar values.

Run the query

```kusto
StormEvents
| where EpisodeNarrative has_all ("cold", "strong", "afternoon", "hail")
| summarize Count=count() by EventType
| top 3 by Count
```

Output

EventType	Count
Thunderstorm Wind	517
Hail	392
Flash Flood	24

Dynamic array

The same result can be achieved using a dynamic array notation.

Run the query

```kusto
StormEvents
| where EpisodeNarrative has_all (dynamic(["cold", "strong", "afternoon",
"hail"]))
| summarize Count=count() by EventType
| top 3 by Count
```

Output

EventType	Count
Thunderstorm Wind	517
Hail	392
Flash Flood	24

The same query can also be written with a let statement.

Run the query

Kusto

```
let criteria = dynamic(["cold", "strong", "afternoon", "hail"]);
StormEvents
| where EpisodeNarrative has_all (criteria)
| summarize Count=count() by EventType
| top 3 by Count
```

EventType	Count
Thunderstorm Wind	517
Hail	392
Flash Flood	24

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

has_any operator

Article • 04/03/2023

Filters a record set for data with any set of case-insensitive strings. `has_any` searches for indexed terms, where an indexed term is three or more characters. If your term is fewer than three characters, the query scans the values in the column, which is slower than looking up the term in the term index.

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Syntax

T `|` `where` *col* `has_any` `(`*expression*`,` `...` `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input to filter.
col	string	✓	The column by which to filter.
expression	scalar or tabular	✓	An expression that specifies the values for which to search. Each expression can be a scalar value or a tabular expression that produces a set of values. If a tabular expression has multiple columns, the first column is used. The search will consider up to 10,000 distinct values.

> ⓘ Note
>
> An inline tabular expression must be enclosed with double parentheses. See example.

Returns

Rows in *T* for which the predicate is `true`.

Examples

List of scalars

The following query shows how to use `has_any` with a comma-separated list of scalar values.

Run the query

Kusto

```
StormEvents
| where State has_any ("CAROLINA", "DAKOTA", "NEW")
| summarize count() by State
```

Output

State	count_
NEW YORK	1750
NORTH CAROLINA	1721
SOUTH DAKOTA	1567
NEW JERSEY	1044
SOUTH CAROLINA	915
NORTH DAKOTA	905
NEW MEXICO	527
NEW HAMPSHIRE	394

Dynamic array

The following query shows how to use `has_any` with a dynamic array.

Run the query

Kusto

```
StormEvents
| where State has_any (dynamic(['south', 'north']))
| summarize count() by State
```

Output

State	count_
NORTH CAROLINA	1721
SOUTH DAKOTA	1567
SOUTH CAROLINA	915
NORTH DAKOTA	905
ATLANTIC SOUTH	193
ATLANTIC NORTH	188

The same query can also be written with a let statement.

Run the query

Kusto

```
let areas = dynamic(['south', 'north']);
StormEvents
| where State has_any (areas)
| summarize count() by State
```

Output

State	count_
NORTH CAROLINA	1721
SOUTH DAKOTA	1567
SOUTH CAROLINA	915
NORTH DAKOTA	905
ATLANTIC SOUTH	193
ATLANTIC NORTH	188

Tabular expression

The following query shows how to use `has_any` with an inline tabular expression. Notice that an inline tabular expression must be enclosed with double parentheses.

Run the query

Kusto

```
StormEvents
| where State has_any ((PopulationData | where Population > 5000000 |
project State))
| summarize count() by State
```

Output

State	count_
TEXAS	4701
ILLINOIS	2022
MISSOURI	2016
GEORGIA	1983
MINNESOTA	1881
...	...

The same query can also be written with a let statement. Notice that the double parentheses as provided in the last example aren't necessary in this case.

Run the query

Kusto

```
let large_states = PopulationData | where Population > 5000000 | project
State;
StormEvents
| where State has_any (large_states)
| summarize count() by State
```

Output

State	count_
TEXAS	4701
ILLINOIS	2022
MISSOURI	2016
GEORGIA	1983
MINNESOTA	1881

State	count_
...	...

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

in operator

Article • 03/29/2023

Filters a record set for data with a case-sensitive string.

The following table provides a comparison of the `in` operators:

Operator	Description	Case-Sensitive	Example (yields `true`)
in	Equals to one of the elements	Yes	`"abc" in ("123", "345", "abc")`
!in	Not equals to any of the elements	Yes	`"bca" !in ("123", "345", "abc")`
in~	Equals to any of the elements	No	`"Abc" in~ ("123", "345", "abc")`
!in~	Not equals to any of the elements	No	`"bCa" !in~ ("123", "345", "ABC")`

> ⓘ Note
>
> Nested arrays are flattened into a single list of values. For example, `x in (dynamic([1,[2,3]]))` becomes `x in (1,2,3)`.

For further information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Case-insensitive operators are currently supported only for ASCII-text. For non-ASCII comparison, use the tolower() function.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

Syntax

T `|` `where` *col* `in` `(`*expression*`,` ... `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input to filter.
col	string	✓	The column by which to filter.
expression	scalar or tabular	✓	An expression that specifies the values for which to search. the values for which to search. Each expression can be a scalar value or a tabular expression that produces a set of values. If a tabular expression has multiple columns, the first column is used. The search will consider up to 1,000,000 distinct values.

> ⓘ Note
>
> An inline tabular expression must be enclosed with double parentheses. See example.

Returns

Rows in *T* for which the predicate is `true`.

Examples

List of scalars

The following query shows how to use `in` with a list of scalar values.

Run the query

Kusto

```
StormEvents
| where State in ("FLORIDA", "GEORGIA", "NEW YORK")
| count
```

Output

Count
4775

Dynamic array

The following query shows how to use `in` with a dynamic array.

Run the query

Kusto

```
let states = dynamic(['FLORIDA', 'ATLANTIC SOUTH', 'GEORGIA']);
StormEvents
| where State in (states)
| count
```

Output

Count
3218

Tabular expression

The following query shows how to use `in` with a tabular expression.

Run the query

Kusto

```
let Top_5_States =
    StormEvents
    | summarize count() by State
    | top 5 by count_;
StormEvents
| where State in (Top_5_States)
| count
```

The same query can be written with an inline tabular expression statement. Notice that an inline tabular expression must be enclosed with double parentheses.

Run the query

Kusto

```
StormEvents
| where State in ((
    StormEvents
    | summarize count() by State
    | top 5 by count_
    ))
| count
```

Output

Count
14242

Top with other example

Run the query

Kusto

```
let Lightning_By_State = materialize(StormEvents
    | summarize lightning_events = countif(EventType == 'Lightning') by
State);
let Top_5_States = Lightning_By_State | top 5 by lightning_events | project
State;
Lightning_By_State
| extend State = iff(State in (Top_5_States), State, "Other")
| summarize sum(lightning_events) by State
```

Output

State	sum_lightning_events
ALABAMA	29
WISCONSIN	31
TEXAS	55
FLORIDA	85
GEORGIA	106
Other	415

Use a static list returned by a function

Run the query

```
Kusto

StormEvents
| where State in (InterestingStates())
| count
```

Output

Count
4775

The function definition.

Run the query

```
Kusto

.show function InterestingStates
```

Output

Name	Parameters	Body	Folder	DocString
InterestingStates	()	{ dynamic(["WASHINGTON", "FLORIDA", "GEORGIA", "NEW YORK"]) }		

Feedback

in~ operator

Article • 04/03/2023

Filters a record set for data with a case-insensitive string.

The following table provides a comparison of the `in` operators:

Operator	Description	Case-Sensitive	Example (yields `true`)
in	Equals to one of the elements	Yes	`"abc" in ("123", "345", "abc")`
!in	Not equals to any of the elements	Yes	`"bca" !in ("123", "345", "abc")`
in~	Equals to any of the elements	No	`"Abc" in~ ("123", "345", "abc")`
!in~	Not equals to any of the elements	No	`"bCa" !in~ ("123", "345", "ABC")`

> ⓘ Note
>
> Nested arrays are flattened into a single list of values. For example, `x in (dynamic([1,[2,3]]))` becomes `x in (1,2,3)`.

For further information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Case-insensitive operators are currently supported only for ASCII-text. For non-ASCII comparison, use the tolower() function.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use the case-sensitive in.

Syntax

T `|` `where` *col* `in~` `(` *expression* `,` `...` `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input to filter.
col	string	✓	The column by which to filter.
expression	scalar or tabular	✓	An expression that specifies the values for which to search. Each expression can be a scalar value or a tabular expression that produces a set of values. If a tabular expression has multiple columns, the first column is used. The search will consider up to 1,000,000 distinct values.

> ⓘ **Note**
>
> An inline tabular expression must be enclosed with double parentheses. See **example**.

Returns

Rows in *T* for which the predicate is `true`.

Examples

List of scalars

The following query shows how to use `in~` with a comma-separated list of scalar values.

Run the query

Kusto

```
StormEvents
| where State in~ ("FLORIDA", "georgia", "NEW YORK")
| count
```

Output

Count
4775

Dynamic array

The following query shows how to use `in~` with a dynamic array.

Run the query

Kusto

```
StormEvents
| where State in~ (dynamic(["FLORIDA", "georgia", "NEW YORK"]))
| count
```

Output

Count
4775

The same query can also be written with a let statement.

Run the query

Kusto

```
let states = dynamic(["FLORIDA", "georgia", "NEW YORK"]);
StormEvents
| where State has_any (states)
| summarize count() by State
```

Output

Count
4775

Tabular expression

The following query shows how to use `in~` with an inline tabular expression. Notice that an inline tabular expression must be enclosed with double parentheses.

Run the query

Kusto

```
StormEvents
| where State in~ ((PopulationData | where Population > 5000000 | project
State))
| summarize count() by State
```

Output

State	count_
TEXAS	4701
ILLINOIS	2022
MISSOURI	2016
GEORGIA	1983
MINNESOTA	1881
...	...

The same query can also be written with a let statement. Notice that the double parentheses as provided in the last example aren't necessary in this case.

Run the query

Kusto

```
let large_states = PopulationData | where Population > 5000000 | project
State;
StormEvents
| where State in~ (large_states)
| summarize count() by State
```

Output

State	count_
TEXAS	4701

State	count_
ILLINOIS	2022
MISSOURI	2016
GEORGIA	1983
MINNESOTA	1881
...	...

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

!in operator

Article • 04/03/2023

Filters a record set for data without a case-sensitive string.

The following table provides a comparison of the `in` operators:

Operator	Description	Case-Sensitive	Example (yields `true`)
in	Equals to one of the elements	Yes	`"abc" in ("123", "345", "abc")`
!in	Not equals to any of the elements	Yes	`"bca" !in ("123", "345", "abc")`
in~	Equals to any of the elements	No	`"Abc" in~ ("123", "345", "abc")`
!in~	Not equals to any of the elements	No	`"bCa" !in~ ("123", "345", "ABC")`

> ⓘ Note
>
> Nested arrays are flattened into a single list of values. For example, `x in (dynamic([1,[2,3]]))` becomes `x in (1,2,3)`.

For further information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Case-insensitive operators are currently supported only for ASCII-text. For non-ASCII comparison, use the tolower() function.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

Syntax

T `|` `where` *col* `!in` `(` *expression* `,` `...` `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input to filter.
col	string	✓	The column by which to filter.
expression	scalar or tabular	✓	An expression that specifies the values for which to search. Each expression can be a scalar value or a tabular expression that produces a set of values. If a tabular expression has multiple columns, the first column is used. The search will consider up to 1,000,000 distinct values.

> ⓘ Note
>
> An inline tabular expression must be enclosed with double parentheses. See example.

Returns

Rows in *T* for which the predicate is `true`.

Example

List of scalars

The following query shows how to use `!in` with a comma-separated list of scalar values.

Run the query

Kusto

```
StormEvents
| where State !in ("FLORIDA", "GEORGIA", "NEW YORK")
| count
```

Output

Count
54291

Dynamic array

The following query shows how to use `!in` with a dynamic array.

Run the query

Kusto

```
StormEvents
| where State !in (dynamic(["FLORIDA", "GEORGIA", "NEW YORK"]))
| count
```

Output

Count
54291

The same query can also be written with a let statement.

Run the query

Kusto

```
let states = dynamic(["FLORIDA", "GEORGIA", "NEW YORK"]);
StormEvents
| where State !in (states)
| summarize count() by State
```

Output

Count
54291

Tabular expression

The following query shows how to use `!in` with an inline tabular expression. Notice that an inline tabular expression must be enclosed with double parentheses.

Run the query

```
Kusto
```

```
StormEvents
| where State !in ((PopulationData | where Population > 5000000 | project
State))
| summarize count() by State
```

Output

State	Count
KANSAS	3166
IOWA	2337
NEBRASKA	1766
OKLAHOMA	1716
SOUTH DAKOTA	1567
...	...

The same query can also be written with a let statement. Notice that the double parentheses as provided in the last example aren't necessary in this case.

Run the query

```
Kusto
```

```
let large_states = PopulationData | where Population > 5000000 | project
State;
StormEvents
| where State !in (large_states)
| summarize count() by State
```

Output

State	Count
KANSAS	3166
IOWA	2337
NEBRASKA	1766

State	Count
OKLAHOMA	1716
SOUTH DAKOTA	1567
...	...

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

!in~ operator

Article • 03/29/2023

Filters a record set for data without a case-insensitive string.

The following table provides a comparison of the `in` operators:

Operator	Description	Case-Sensitive	Example (yields `true`)
in	Equals to one of the elements	Yes	`"abc" in ("123", "345", "abc")`
!in	Not equals to any of the elements	Yes	`"bca" !in ("123", "345", "abc")`
in~	Equals to any of the elements	No	`"Abc" in~ ("123", "345", "abc")`
!in~	Not equals to any of the elements	No	`"bCa" !in~ ("123", "345", "ABC")`

> ⓘ Note
>
> Nested arrays are flattened into a single list of values. For example, `x in (dynamic([1,[2,3]]))` becomes `x in (1,2,3)`.

For further information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Case-insensitive operators are currently supported only for ASCII-text. For non-ASCII comparison, use the tolower() function.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use the case-sensitive !in~.

Syntax

T `|` `where` *col* `!in~` `(` *expression* `,` `...` `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input to filter.
col	string	✓	The column by which to filter.
expression	scalar or tabular	✓	An expression that specifies the values for which to search. Each expression can be a scalar value or a tabular expression that produces a set of values. If a tabular expression has multiple columns, the first column is used. The search will consider up to 1,000,000 distinct values.

> ⓘ Note
>
> An inline tabular expression must be enclosed with double parentheses. See example.

Returns

Rows in *T* for which the predicate is `true`.

Example

List of scalars

The following query shows how to use `!in~` with a comma-separated list of scalar values.

Run the query

Kusto

```
StormEvents
| where State !in~ ("Florida", "Georgia", "New York")
| count
```

Output

Count
54,291

Dynamic array

The following query shows how to use `!in~` with a dynamic array.

Run the query

Kusto

```
StormEvents
| where State !in~ (dynamic(["Florida", "Georgia", "New York"]))
| count
```

Output

Count
54291

The same query can also be written with a let statement.

Run the query

Kusto

```
let states = dynamic(["Florida", "Georgia", "New York"]);
StormEvents
| where State !in~ (states)
| summarize count() by State
```

Output

Count
54291

Tabular expression

The following query shows how to use `!in~` with an inline tabular expression. Notice that an inline tabular expression must be enclosed with double parentheses.

Run the query

Kusto

```
StormEvents
| where State !in~ ((PopulationData | where Population > 5000000 | project
State))
| summarize count() by State
```

Output

State	count_
KANSAS	3166
IOWA	2337
NEBRASKA	1766
OKLAHOMA	1716
SOUTH DAKOTA	1567
...	...

The same query can also be written with a let statement. Notice that the double parentheses as provided in the last example aren't necessary in this case.

Run the query

Kusto

```
let large_states = PopulationData | where Population > 5000000 | project
State;
StormEvents
| where State !in~ (large_states)
| summarize count() by State
```

Output

State	count_
KANSAS	3166

State	count_
IOWA	2337
NEBRASKA	1766
OKLAHOMA	1716
SOUTH DAKOTA	1567
...	...

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

matches regex operator

Article • 03/10/2023

Filters a record set based on a case-sensitive regex value.

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

Syntax

T `|` `where` *col* `matches` `regex` `(`*expression*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
col	string	✓	The column by which to filter.
expression	scalar	✓	The expression used to filter.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State matches regex "K.*S"
```

```
| where event_count > 10
| project State, event_count
```

Output

State	event_count
KANSAS	3166
ARKANSAS	1028
LAKE SUPERIOR	34
LAKE ST CLAIR	32

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

startswith operator

Article • 02/06/2023

Filters a record set for data with a case-insensitive string starting sequence.

The following table compares the `startswith` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
startswith	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
!startswith	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
startswith_cs	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
!startswith_cs	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use the case-sensitive startswith_cs.

Syntax

T `|` `where` *col* `startswith` `(`*expression*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input to filter.
col	string	✓	The column used to filter.
expression	string	✓	The expression by which to filter.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State startswith "Lo"
| where event_count > 10
| project State, event_count
```

Output

State	event_count
LOUISIANA	463

Feedback

startswith_cs operator

Article • 02/06/2023

Filters a record set for data with a case-sensitive string starting sequence.

The following table compares the `startswith` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
startswith	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
!startswith	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
startswith_cs	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
!startswith_cs	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

Syntax

T `|` `where` *col* `startswith_cs` `(`*expression*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input to filter.
col	string	✓	The column used to filter.
expression	string	✓	The expression by which to filter.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State startswith_cs "I"
| where event_count > 2000
| project State, event_count
```

Output

State	event_count
IOWA	2337
ILLINOIS	2022

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

!startswith operator

Article • 03/12/2023

Filters a record set for data that doesn't start with a case-insensitive search string.

The following table compares the `startswith` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
startswith	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
!startswith	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
startswith_cs	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
!startswith_cs	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

When possible, use the case-sensitive !startswith_cs.

Syntax

T `|` `where` *column* `!startswith` `(` *expression* `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
column	string	✓	The column by which to filter.
expression	scalar	✓	The scalar or literal expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State !startswith "i"
| where event_count > 2000
| project State, event_count
```

Output

State	event_count
TEXAS	4701
KANSAS	3166
MISSOURI	2016

Feedback

!startswith_cs operators

Article • 03/12/2023

Filters a record set for data that doesn't start with a case-sensitive search string.

The following table compares the `startswith` operators using the abbreviations provided:

- RHS = right-hand side of the expression
- LHS = left-hand side of the expression

Operator	Description	Case-Sensitive	Example (yields `true`)
startswith	RHS is an initial subsequence of LHS	No	`"Fabrikam" startswith "fab"`
!startswith	RHS isn't an initial subsequence of LHS	No	`"Fabrikam" !startswith "kam"`
startswith_cs	RHS is an initial subsequence of LHS	Yes	`"Fabrikam" startswith_cs "Fab"`
!startswith_cs	RHS isn't an initial subsequence of LHS	Yes	`"Fabrikam" !startswith_cs "fab"`

For more information about other operators and to determine which operator is most appropriate for your query, see datatype string operators.

Performance tips

> ⓘ Note
>
> Performance depends on the type of search and the structure of the data. For best practices, see **Query best practices**.

Syntax

T `|` `where` *column* `!startswith_cs` `(` *expression* `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose records are to be filtered.
column	string	✓	The column by which to filter.
expression	scalar	✓	The scalar or literal expression for which to search.

Returns

Rows in *T* for which the predicate is `true`.

Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State !startswith_cs "I"
| where event_count > 2000
| project State, event_count
```

Output

State	event_count
TEXAS	4701
KANSAS	3166
MISSOURI	2016

Feedback