# String operators

Article • 04/16/2023

Kusto offers various query operators for searching string data types. The following article describes how string terms are indexed, lists the string query operators, and gives tips for optimizing performance.

## Understanding string terms

Kusto indexes all columns, including columns of type `string`. Multiple indexes are built for such columns, depending on the actual data. These indexes aren't directly exposed, but are used in queries with the `string` operators that have `has` as part of their name, such as `has`, `!has`, `hasprefix`, `!hasprefix`. The semantics of these operators are dictated by the way the column is encoded. Instead of doing a "plain" substring match, these operators match *terms*.

### What is a term?

By default, each `string` value is broken into maximal sequences of alphanumeric characters, and each of those sequences is made into a term.

For example, in the following `string`, the terms are `Kusto`, `KustoExplorerQueryRun`, and the following substrings: `ad67d136`, `c1db`, `4f9f`, `88ef`, `d94f3b6b0b5a`.

Kusto

```
Kusto: ad67d136-c1db-4f9f-88ef-d94f3b6b0b5a;KustoExplorerQueryRun
```

Kusto builds a term index consisting of all terms that are *three characters or more*, and this index is used by string operators such as `has`, `!has`, and so on. If the query looks for a term that is smaller than three characters, or uses a `contains` operator, then the query will revert to scanning the values in the column. Scanning is much slower than looking up the term in the term index.

> ⓘ **Note**
>
> In EngineV2, a term consists of four or more characters.

# Operators on strings

The following abbreviations are used in this article:

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

Operators with an `_cs` suffix are case sensitive.

| Operator | Description | Case-Sensitive | Example (yields `true`) |
|---|---|---|---|
| == | Equals | Yes | `"aBc" == "aBc"` |
| != | Not equals | Yes | `"abc" != "ABC"` |
| =~ | Equals | No | `"abc" =~ "ABC"` |
| !~ | Not equals | No | `"aBc" !~ "xyz"` |
| contains | RHS occurs as a subsequence of LHS | No | `"FabriKam" contains "BRik"` |
| !contains | RHS doesn't occur in LHS | No | `"Fabrikam" !contains "xyz"` |
| contains_cs | RHS occurs as a subsequence of LHS | Yes | `"FabriKam" contains_cs "Kam"` |
| !contains_cs | RHS doesn't occur in LHS | Yes | `"Fabrikam" !contains_cs "Kam"` |
| endswith | RHS is a closing subsequence of LHS | No | `"Fabrikam" endswith "Kam"` |
| !endswith | RHS isn't a closing subsequence of LHS | No | `"Fabrikam" !endswith "brik"` |
| endswith_cs | RHS is a closing subsequence of LHS | Yes | `"Fabrikam" endswith_cs "kam"` |
| !endswith_cs | RHS isn't a closing subsequence of LHS | Yes | `"Fabrikam" !endswith_cs "brik"` |
| 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_all | Same as `has` but works on all of the elements | No | `"North and South America" has_all("south", "north")` |

| Operator | Description | Case-Sensitive | Example (yields `true`) |
|---|---|---|---|
| has_any | Same as `has` but works on any of the elements | No | `"North America" has_any("south", "north")` |
| 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"` |
| 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"` |
| 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"` |
| in | Equals to any 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")` |
| matches regex | LHS contains a match for RHS | Yes | `"Fabrikam" matches regex "b.*k"` |

| 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"` |

# Performance tips

For better performance, when there are two operators that do the same task, use the case-sensitive one. For example:

- Use `==`, not `=~`
- Use `in`, not `in~`
- Use `hassuffix_cs`, not `hassuffix`

For faster results, if you're testing for the presence of a symbol or alphanumeric word that is bound by non-alphanumeric characters, or the start or end of a field, use `has` or `in`. `has` works faster than `contains`, `startswith`, or `endswith`.

To search for IPv4 addresses or their prefixes, use one of special operators on IPv4 addresses, which are optimized for this purpose.

For more information, see Query best practices.

For example, the first of these queries will run faster:

Run the query

Kusto

```
StormEvents | where State has "North" | count;
StormEvents | where State contains "nor" | count
```

# Operators on IPv4 addresses

The following group of operators provide index accelerated search on IPv4 addresses or their prefixes.

| Operator | Description | Example (yields `true`) |
|---|---|---|
| has_ipv4 | LHS contains IPv4 address represented by RHS | `has_ipv4("Source address is 10.1.2.3:1234", "10.1.2.3")` |
| has_ipv4_prefix | LHS contains an IPv4 address that matches a prefix represented by RHS | `has_ipv4_prefix("Source address is 10.1.2.3:1234", "10.1.2.")` |
| has_any_ipv4 | LHS contains one of IPv4 addresses provided by RHS | `has_any_ipv4("Source address is 10.1.2.3:1234", dynamic(["10.1.2.3", "127.0.0.1"]))` |
| has_any_ipv4_prefix | LHS contains an IPv4 address that matches one of prefixes provided by RHS | `has_any_ipv4_prefix("Source address is 10.1.2.3:1234", dynamic(["10.1.2.", "127.0.0."]))` |

# Feedback

# contains operator

Article • 03/16/2023

Filters a record set for data containing a case-insensitive string. `contains` searches for arbitrary sub-strings rather than terms.

The following table compares the `contains` 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`) |
|---|---|---|---|
| contains | RHS occurs as a subsequence of LHS | No | `"FabriKam" contains "BRik"` |
| !contains | RHS doesn't occur in LHS | No | `"Fabrikam" !contains "xyz"` |
| contains_cs | RHS occurs as a subsequence of LHS | Yes | `"FabriKam" contains_cs "Kam"` |
| !contains_cs | RHS doesn't occur in LHS | Yes | `"Fabrikam" !contains_cs "Kam"` |

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 contains_cs - a case-sensitive version of the operator.

If you're looking for a term, use `has` for faster results.

## Syntax

*T* `|` `where` *col* `contains_cs` `(`*string*`)`

# Parameters

| Name | Type | Required | Description |
|---|---|---|---|
| *T* | string | ✓ | The tabular input whose records are to be filtered. |
| *col* | string | ✓ | The name of the column to check for *string*. |
| *string* | string | ✓ | The case-sensitive string by which to filter the data. |

# Returns

Rows in *T* for which *string* is in *col*.

# Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State contains "enn"
| where event_count > 10
| project State, event_count
| render table
```

### Output

| State | event_count |
|---|---|
| PENNSYLVANIA | 1687 |
| TENNESSEE | 1125 |

# Feedback

# contains_cs operator

Article • 03/16/2023

Filters a record set for data containing a case-sensitive string. `contains_cs` searches for arbitrary sub-strings rather than terms.

The following table compares the `contains` 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`) |
|---|---|---|---|
| contains | RHS occurs as a subsequence of LHS | No | `"FabriKam" contains "BRik"` |
| !contains | RHS doesn't occur in LHS | No | `"Fabrikam" !contains "xyz"` |
| contains_cs | RHS occurs as a subsequence of LHS | Yes | `"FabriKam" contains_cs "Kam"` |
| !contains_cs | RHS doesn't occur in LHS | Yes | `"Fabrikam" !contains_cs "Kam"` |

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**.

If you're looking for a term, use `has_cs` for faster results.

## Syntax

*T* `|` `where` *col* `contains_cs` `(`*string*`)`

# Parameters

| Name | Type | Required | Description |
|---|---|---|---|
| *T* | string | ✓ | The tabular input whose records are to be filtered. |
| *col* | string | ✓ | The name of the column to check for *string*. |
| *string* | string | ✓ | The case-sensitive string by which to filter the data. |

# Returns

Rows in *T* for which *string* is in *col*.

# Example

Run the query

Kusto

```
StormEvents
| summarize event_count=count() by State
| where State contains_cs "AS"
```

### Output

| Count |
|---|
| 8 |

# Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

# !contains operator

Article • 03/12/2023

Filters a record set for data that doesn't include a case-sensitive string. `!contains` searches for characters rather than terms of three or more characters. The query scans the values in the column, which is slower than looking up a term in a term index.

The following table compares the `contains` 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`) |
|---|---|---|---|
| contains | RHS occurs as a subsequence of LHS | No | `"FabriKam" contains "BRik"` |
| !contains | RHS doesn't occur in LHS | No | `"Fabrikam" !contains "xyz"` |
| contains_cs | RHS occurs as a subsequence of LHS | Yes | `"FabriKam" contains_cs "Kam"` |
| !contains_cs | RHS doesn't occur in LHS | Yes | `"Fabrikam" !contains_cs "Kam"` |

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 !contains_cs.

Use `!has` if you're looking for a term.

## Syntax

## Case insensitive syntax

*T* `|` `where` *Column* `!contains` `(` *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 !contains "kan"
| where event_count > 3000
| project State, event_count
```

### Output

| State | event_count |
|---|---|
| TEXAS | 4701 |

# Feedback

Was this page helpful? Yes No

# !contains_cs operator

Article • 03/12/2023

Filters a record set for data that doesn't include a case-sensitive string. `!contains_cs` searches for characters rather than terms of three or more characters. The query scans the values in the column, which is slower than looking up a term in a term index.

The following table compares the `contains` 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`) |
|---|---|---|---|
| contains | RHS occurs as a subsequence of LHS | No | `"FabriKam" contains "BRik"` |
| !contains | RHS doesn't occur in LHS | No | `"Fabrikam" !contains "xyz"` |
| contains_cs | RHS occurs as a subsequence of LHS | Yes | `"FabriKam" contains_cs "Kam"` |
| !contains_cs | RHS doesn't occur in LHS | Yes | `"Fabrikam" !contains_cs "Kam"` |

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**.

If you're looking for a term, use `!has_cs` for faster results.

# Syntax

## Case-sensitive syntax

*T* `|` `where` *Column* `!contains_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 !contains_cs "AS"
| count
```

### Output

| Count |
|---|
| 59 |

Run the query

Kusto

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

Output

| State | event_count |
| --- | --- |
| KANSAS | 3,166 |

# Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

# endswith operator

Article • 01/30/2023

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

The following table compares the `endswith` 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`) |
|---|---|---|---|
| endswith | RHS is a closing subsequence of LHS | No | `"Fabrikam" endswith "Kam"` |
| !endswith | RHS isn't a closing subsequence of LHS | No | `"Fabrikam" !endswith "brik"` |
| endswith_cs | RHS is a closing subsequence of LHS | Yes | `"Fabrikam" endswith_cs "kam"` |
| !endswith_cs | RHS isn't a closing subsequence of LHS | Yes | `"Fabrikam" !endswith_cs "brik"` |

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**.

For faster results, use the case-sensitive version of an operator. For example, use `endswith_cs` instead of `endswith`.

## Syntax

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

# Parameters

| Name | Type | Required | Description |
|---|---|---|---|
| *T* | string | ✓ | The tabular input whose records are to be filtered. |
| *col* | string | ✓ | The column to filter. |
| *expression* | string | ✓ | The expression used to filter. |

# Returns

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

# Example

Run the query

Kusto

```
StormEvents
| summarize Events=count() by State
| where State endswith "sas"
| where Events > 10
| project State, Events
```

### Output

| State | Events |
|---|---|
| KANSAS | 3166 |
| ARKANSAS | 1028 |

# Feedback

# endswith_cs operator

Article • 01/30/2023

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

The following table compares the `endswith` 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`) |
|---|---|---|---|
| endswith | RHS is a closing subsequence of LHS | No | `"Fabrikam" endswith "Kam"` |
| !endswith | RHS isn't a closing subsequence of LHS | No | `"Fabrikam" !endswith "brik"` |
| endswith_cs | RHS is a closing subsequence of LHS | Yes | `"Fabrikam" endswith_cs "kam"` |
| !endswith_cs | RHS isn't a closing subsequence of LHS | Yes | `"Fabrikam" !endswith_cs "brik"` |

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* `endswith_cs` `(`*expression*`)`

## Parameters

| Name | Type | Required | Description |
|---|---|---|---|
| *T* | string | ✓ | The tabular input whose records are to be filtered. |
| *col* | string | ✓ | The column to filter. |
| *expression* | string | ✓ | The expression used to filter. |

# Returns

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

# Example

Run the query

Kusto

```
StormEvents
| summarize Events = count() by State
| where State endswith_cs "NA"
```

### Output

| State | Events |
|---|---|
| NORTH CAROLINA | 1721 |
| MONTANA | 1230 |
| INDIANA | 1164 |
| SOUTH CAROLINA | 915 |
| LOUISIANA | 463 |
| ARIZONA | 340 |

# Feedback

# !endswith operator

Article • 01/30/2023

Filters a record set for data that excludes a case-insensitive ending string.

The following table compares the `endswith` 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`) |
|---|---|---|---|
| endswith | RHS is a closing subsequence of LHS | No | `"Fabrikam" endswith "Kam"` |
| !endswith | RHS isn't a closing subsequence of LHS | No | `"Fabrikam" !endswith "brik"` |
| endswith_cs | RHS is a closing subsequence of LHS | Yes | `"Fabrikam" endswith_cs "kam"` |
| !endswith_cs | RHS isn't a closing subsequence of LHS | Yes | `"Fabrikam" !endswith_cs "brik"` |

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 !endswith_cs.

## Syntax

*T* `|` `where` *col* `!endswith` `(` *expression* `)`

# Parameters

| Name | Type | Required | Description |
|---|---|---|---|
| *T* | string | ✓ | The tabular input whose records are to be filtered. |
| *col* | string | ✓ | The column to filter. |
| *expression* | string | ✓ | The expression used to filter. |

# Returns

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

# Example

Run the query

Kusto

```
StormEvents
| summarize Events=count() by State
| where State !endswith "is"
| where Events > 2000
| project State, Events
```

### Output

| State | Events |
|---|---|
| TEXAS | 4701 |
| KANSAS | 3166 |
| IOWA | 2337 |
| MISSOURI | 2016 |

# Feedback

Was this page helpful? Yes No

# !endswith_cs operator

Article • 01/30/2023

Filters a record set for data that doesn't contain a case-insensitive ending string.

The following table compares the `endswith` 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`) |
|---|---|---|---|
| endswith | RHS is a closing subsequence of LHS | No | `"Fabrikam" endswith "Kam"` |
| !endswith | RHS isn't a closing subsequence of LHS | No | `"Fabrikam" !endswith "brik"` |
| endswith_cs | RHS is a closing subsequence of LHS | Yes | `"Fabrikam" endswith_cs "kam"` |
| !endswith_cs | RHS isn't a closing subsequence of LHS | Yes | `"Fabrikam" !endswith_cs "brik"` |

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* `!endswith_cs` `(`*expression*`)`

## Parameters

| Name | Type | Required | Description |
|---|---|---|---|
| *T* | string | ✓ | The tabular input whose records are to be filtered. |
| *col* | string | ✓ | The column to filter. |
| *expression* | string | ✓ | The expression used to filter. |

# Returns

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

# Example

Run the query

Kusto

```
StormEvents
| summarize Events=count() by State
| where State !endswith_cs "A"
```

The following table only shows the first 10 results. To see the full output, run the query.

| State | Events |
|---|---|
| TEXAS | 4701 |
| KANSAS | 3166 |
| ILLINOIS | 2022 |
| MISSOURI | 2016 |
| WISCONSIN | 1850 |
| NEW YORK | 1750 |
| COLORADO | 1654 |
| MICHIGAN | 1637 |
| KENTUCKY | 1391 |
| OHIO | 1233 |

# Feedback

**Was this page helpful?** Yes No

Provide product feedback | Get help at Microsoft Q&A

# == (equals) operator

Article • 03/29/2023

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

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

| Operator | Description | Case-Sensitive | Example (yields `true`) |
|---|---|---|---|
| == | Equals | Yes | `"aBc" == "aBc"` |
| != | Not equals | Yes | `"abc" != "ABC"` |
| =~ | Equals | No | `"abc" =~ "ABC"` |
| !~ | Not equals | No | `"aBc" !~ "xyz"` |

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* `==` `(`*expression*`,` ... `)`

## Parameters

| Name | Type | Required | Description |
|---|---|---|---|
| *T* | string | ✓ | The tabular input whose records are to be filtered. |
| *col* | string | ✓ | The column to filter. |
| *expression* | string | ✓ | The expression used to filter. |

# Returns

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

# Example

Run the query

```
Kusto

StormEvents
| where State == "kansas"
| count
```

| Count |
| --- |
| 0 |

Run the query

```
Kusto

StormEvents
| where State == "KANSAS"
| count
```

| Count |
| --- |
| 3,166 |

# Feedback

# =~ (equals) operator

Article • 01/30/2023

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

The following table provides a comparison of the `==` (equals) operators:

| Operator | Description | Case-Sensitive | Example (yields `true`) |
|---|---|---|---|
| == | Equals | Yes | `"aBc" == "aBc"` |
| != | Not equals | Yes | `"abc" != "ABC"` |
| =~ | Equals | No | `"abc" =~ "ABC"` |
| !~ | Not equals | No | `"aBc" !~ "xyz"` |

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 == - a case-sensitive version of the operator.

## Syntax

*T* `|` `where` *col* `=~` `(`*expression*`)`

## Parameters

| Name | Type | Required | Description |
|---|---|---|---|
| *T* | string | ✓ | The tabular input whose records are to be filtered. |
| *col* | string | ✓ | The column to filter. |

| Name | Type | Required | Description |
|---|---|---|---|
| *expression* | string | ✓ | The expression used to filter. |

# Returns

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

# Example

The `State` values in the `StormEvents` table are capitalized. The following query matches columns with the value "KANSAS".

Run the query

Kusto

```
StormEvents
| where State =~ "kansas"
| project EventId, State
```

The following table only shows the first 10 results. To see the full output, run the query.

| EventId | State |
|---|---|
| 70787 | KANSAS |
| 43450 | KANSAS |
| 43451 | KANSAS |
| 38844 | KANSAS |
| 18463 | KANSAS |
| 18464 | KANSAS |
| 18495 | KANSAS |
| 43466 | KANSAS |
| 43467 | KANSAS |
| 43470 | KANSAS |

# Feedback

**Was this page helpful?** Yes No

Provide product feedback | Get help at Microsoft Q&A

# != (not equals) operator

Article • 03/12/2023

Filters a record set for data that doesn't match a case-sensitive string.

The following table provides a comparison of the `==` (equals) operators:

| Operator | Description | Case-Sensitive | Example (yields `true`) |
|---|---|---|---|
| == | Equals | Yes | `"aBc" == "aBc"` |
| != | Not equals | Yes | `"abc" != "ABC"` |
| =~ | Equals | No | `"abc" =~ "ABC"` |
| !~ | Not equals | No | `"aBc" !~ "xyz"` |

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* `!=` `(` *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 != "FLORIDA") and (event_count > 4000)
| project State, event_count
```

### Output

| State | event_count |
|---|---|
| TEXAS | 4,701 |

# Feedback

# !~ (not equals) operator

Article • 03/12/2023

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

The following table provides a comparison of the `==` (equals) operators:

| Operator | Description | Case-Sensitive | Example (yields `true`) |
|---|---|---|---|
| == | Equals | Yes | `"aBc" == "aBc"` |
| != | Not equals | Yes | `"abc" != "ABC"` |
| =~ | Equals | No | `"abc" =~ "ABC"` |
| !~ | Not equals | No | `"aBc" !~ "xyz"` |

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 !=.

## Syntax

*T* `|` `where` *column* `!~` `(`*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. |

| Name | Type | Required | Description |
|---|---|---|---|
| *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 !~ "texas") and (event_count > 3000)
| project State, event_count
```

### Output

| State | event_count |
|---|---|
| KANSAS | 3,166 |

# Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A