dcount_intersect plugin

Article • 12/28/2022

Calculates intersection between N sets based on `hll` values (N in range of [2..16]), and returns N `dcount` values. The plugin is invoked with the evaluate operator.

Syntax

T `| evaluate` `dcount_intersect(`*hll_1*, *hll_2*, [`,` *hll_3*`,` ...]`)`

Parameters

Name	Type	Required	Description
T	string	✓	The input tabular expression.
hll_i	The values of set S_i calculated with the hll() function.		

Returns

Returns a table with N `dcount` values (per column, representing set intersections). Column names are s0, s1, ... (until n-1).

Given sets S_1, S_2, .. S_n return values will be representing distinct counts of:
S_1,
$S_1 \cap S_2$,
$S_1 \cap S_2 \cap S_3$,
... ,
$S_1 \cap S_2 \cap ... \cap S_n$

Examples

Run the query

Kusto

```
// Generate numbers from 1 to 100
range x from 1 to 100 step 1
```

```
| extend isEven = (x % 2 == 0), isMod3 = (x % 3 == 0), isMod5 = (x % 5 == 0)
// Calculate conditional HLL values (note that '0' is included in each of
them as additional value, so we will subtract it later)
| summarize hll_even = hll(iif(isEven, x, 0), 2),
            hll_mod3 = hll(iif(isMod3, x, 0), 2),
            hll_mod5 = hll(iif(isMod5, x, 0), 2)
// Invoke the plugin that calculates dcount intersections
| evaluate dcount_intersect(hll_even, hll_mod3, hll_mod5)
| project evenNumbers = s0 - 1,             //
100 / 2 = 50
          even_and_mod3 = s1 - 1,           // gcd(2,3) = 6, therefor:
100 / 6 = 16
          even_and_mod3_and_mod5 = s2 - 1   // gcd(2,3,5) is 30, therefore:
100 / 30 = 3
```

Output

evenNumbers	even_and_mod3	even_and_mod3_and_mod5
50	16	3

infer_storage_schema plugin

Article • 03/12/2023

This plugin infers schema of external data, and returns it as CSL schema string. The string can be used when creating external tables. The plugin is invoked with the evaluate operator.

Authentication and authorization

In the properties of the request, you specify storage connection strings to access. Each storage connection string specifies the authorization method to use for access to the storage. Depending on the authorization method, the principal may need to be granted permissions on the external storage to perform the schema inference.

The following table lists the supported authentication methods and any required permissions by storage type.

Authentication method	Azure Blob Storage / Data Lake Storage Gen2	Data Lake Storage Gen1
Impersonation	Storage Blob Data Reader	Reader
Shared Access (SAS) token	List + Read	This authentication method isn't supported in Gen1.
Azure AD access token		
Storage account access key		This authentication method isn't supported in Gen1.

Syntax

`evaluate` `infer_storage_schema(` *Options* `)`

Parameters

Name	Type	Required	Description
Options	dynamic	✓	A property bag specifying the properties of the request.

Properties of the request

Name	Type	Required	Description
StorageContainers	dynamic	✓	An array of storage connection strings that represent prefix URI for stored data artifacts.
DataFormat	string	✓	One of the supported data formats.
FileExtension	string		If specified, the function will only scan files ending with this file extension. Specifying the extension may speed up the process or eliminate data reading issues.
FileNamePrefix	string		If specified, the function will only scan files starting with this prefix. Specifying the prefix may speed up the process.
Mode	string		The schema inference strategy. A value of: `any`, `last`, `all`. The function infers the data schema from the first found file, from the last written file, or from all files respectively. The default value is `last`.

Returns

The `infer_storage_schema` plugin returns a single result table containing a single row/column holding CSL schema string.

Note

- Storage container URI secret keys must have the permissions for *List* in addition to *Read*.
- Schema inference strategy 'all' is a very "expensive" operation, as it implies reading from *all* artifacts found and merging their schema.
- Some returned types may not be the actual ones as a result of wrong type guess (or, as a result of schema merge process). This is why you should review the result carefully before creating an external table.

Example

Kusto

```
let options = dynamic({
  'StorageContainers': [
    h@'https://storageaccount.blob.core.windows.net/MobileEvents;secretKey'
  ],
  'FileExtension': '.parquet',
  'FileNamePrefix': 'part-',
  'DataFormat': 'parquet'
});
evaluate infer_storage_schema(options)
```

Output

CslSchema
app_id:string, user_id:long, event_time:datetime, country:string, city:string, device_type:string, device_vendor:string, ad_network:string, campaign:string, site_id:string, event_type:string, event_name:string, organic:string, days_from_install:int, revenue:real

Use the returned schema in external table definition:

Kusto

```
.create external table MobileEvents(
    app_id:string, user_id:long, event_time:datetime, country:string,
city:string, device_type:string, device_vendor:string, ad_network:string,
campaign:string, site_id:string, event_type:string, event_name:string,
organic:string, days_from_install:int, revenue:real
)
kind=blob
partition by (dt:datetime = bin(event_time, 1d), app:string = app_id)
pathformat = ('app=' app '/dt=' datetime_pattern('yyyyMMdd', dt))
dataformat = parquet
(
    h@'https://storageaccount.blob.core.windows.net/MovileEvents;secretKey'
)
```

Feedback

ipv4_lookup plugin

Article • 01/10/2023

The `ipv4_lookup` plugin looks up an IPv4 value in a lookup table and returns rows with matched values. The plugin is invoked with the evaluate operator.

Syntax

T `|` `evaluate` `ipv4_lookup(` *LookupTable* `,` *SourceIPv4Key* `,` *IPv4LookupKey* [`,` *ExtraKey1* [.. `,` *ExtraKeyN* [`,` *return_unmatched*]]] `)`

Parameters

Name	Type	Required	Description
T	string	✓	The tabular input whose column *SourceIPv4Key* will be used for IPv4 matching.
LookupTable	string	✓	Table or tabular expression with IPv4 lookup data, whose column *LookupKey* will be used for IPv4 matching. IPv4 values can be masked using IP-prefix notation.
SourceIPv4Key	string	✓	The column of *T* with IPv4 string to be looked up in *LookupTable*. IPv4 values can be masked using IP-prefix notation.
IPv4LookupKey	string	✓	The column of *LookupTable* with IPv4 string that is matched against each *SourceIPv4Key* value.
ExtraKey1 .. *ExtraKeyN*	string		Additional column references that are used for lookup matches. Similar to `join` operation: records with equal values will be considered matching. Column name references must exist both is source table `T` and `LookupTable`.
return_unmatched	bool		A boolean flag that defines if the result should include all or only matching rows (default: `false` - only matching rows returned).

IP-prefix notation

IP-prefix notation (also known as CIDR notation) is a concise way of representing an IP address and its associated network mask. The format is `<base IP>/<prefix length>`, where the prefix length is the number of leading 1 bits in the netmask. The prefix length determines the range of IP addresses that belong to the network.

For IPv4, the prefix length is a number between 0 and 32. So the notation 192.168.2.0/24 represents the IP address 192.168.2.0 with a netmask of 255.255.255.0. This netmask has 24 leading 1 bits, or a prefix length of 24.

For IPv6, the prefix length is a number between 0 and 128. So the notation fe80::85d:e82c:9446:7994/120 represents the IP address fe80::85d:e82c:9446:7994 with a netmask of ffff:ffff:ffff:ffff:ffff:ffff:ffff:ff00. This netmask has 120 leading 1 bits, or a prefix length of 120.

Returns

The `ipv4_lookup` plugin returns a result of join (lookup) based on IPv4 key. The schema of the table is the union of the source table and the lookup table, similar to the result of the lookup operator.

If the *return_unmatched* argument is set to `true`, the resulting table will include both matched and unmatched rows (filled with nulls).

If the *return_unmatched* argument is set to `false`, or omitted (the default value of `false` is used), the resulting table will have as many records as matching results. This variant of lookup has better performance compared to `return_unmatched=true` execution.

ⓘ **Note**

- This plugin covers the scenario of IPv4-based join, assuming a small lookup table size (100K-200K rows), with the input table optionally having a larger size.
- The performance of the plugin will depend on the sizes of the lookup and data source tables, the number of columns, and number of matching records.

Examples

IPv4 lookup - matching rows only

Run the query

Kusto

```
// IP lookup table: IP_Data
// Partial data from: https://raw.githubusercontent.com/datasets/geoip2-ipv4/master/data/geoip2-ipv4.csv
let IP_Data = datatable(network:string, continent_code:string ,continent_name:string, country_iso_code:string,
country_name:string)
[
  "111.68.128.0/17","AS","Asia","JP","Japan",
  "5.8.0.0/19","EU","Europe","RU","Russia",
  "223.255.254.0/24","AS","Asia","SG","Singapore",
  "46.36.200.51/32","OC","Oceania","CK","Cook Islands",
  "2.20.183.0/24","EU","Europe","GB","United Kingdom",
];
let IPs = datatable(ip:string)
[
  '2.20.183.12',   // United Kingdom
  '5.8.1.2',       // Russia
  '192.165.12.17', // Unknown
];
IPs
| evaluate ipv4_lookup(IP_Data, ip, network)
```

Output

ip	network	continent_code	continent_name	country_iso_code	country_name
2.20.183.12	2.20.183.0/24	EU	Europe	GB	United Kingdom
5.8.1.2	5.8.0.0/19	EU	Europe	RU	Russia

IPv4 lookup - return both matching and non-matching rows

Run the query

Kusto

```
// IP lookup table: IP_Data
// Partial data from:
// https://raw.githubusercontent.com/datasets/geoip2-ipv4/master/data/geoip2-ipv4.csv
let IP_Data = datatable(network:string,continent_code:string ,continent_name:string ,country_iso_code:string
,country_name:string )
[
    "111.68.128.0/17","AS","Asia","JP","Japan",
    "5.8.0.0/19","EU","Europe","RU","Russia",
    "223.255.254.0/24","AS","Asia","SG","Singapore",
    "46.36.200.51/32","OC","Oceania","CK","Cook Islands",
    "2.20.183.0/24","EU","Europe","GB","United Kingdom",
];
let IPs = datatable(ip:string)
[
    '2.20.183.12',   // United Kingdom
    '5.8.1.2',       // Russia
    '192.165.12.17', // Unknown
];
IPs
| evaluate ipv4_lookup(IP_Data, ip, network, return_unmatched = true)
```

Output

ip	network	continent_code	continent_name	country_iso_code	country_name
2.20.183.12	2.20.183.0/24	EU	Europe	GB	United Kingdom
5.8.1.2	5.8.0.0/19	EU	Europe	RU	Russia
192.165.12.17					

IPv4 lookup - using source in external_data()

Run the query

Kusto

```
let IP_Data = external_data(network:string,geoname_id:long,continent_code:string,continent_name:string
,country_iso_code:string,country_name:string,is_anonymous_proxy:bool,is_satellite_provider:bool)
    ['https://raw.githubusercontent.com/datasets/geoip2-ipv4/master/data/geoip2-ipv4.csv'];
let IPs = datatable(ip:string)
[
    '2.20.183.12',   // United Kingdom
    '5.8.1.2',       // Russia
    '192.165.12.17', // Sweden
];
IPs
| evaluate ipv4_lookup(IP_Data, ip, network, return_unmatched = true)
```

Output

ip	network	geoname_id	continent_code	continent_name	country_iso_code	country_name	is_anonymous_proxy	is_satellite_pr
2.20.183.12	2.20.183.0/24	2635167	EU	Europe	GB	United Kingdom	0	0
5.8.1.2	5.8.0.0/19	2017370	EU	Europe	RU	Russia	0	0
192.165.12.17	192.165.8.0/21	2661886	EU	Europe	SE	Sweden	0	0

IPv4 lookup - using extra columns for matching

Run the query

Kusto

```
let IP_Data = external_data(network:string,geoname_id:long,continent_code:string,continent_name:string
,country_iso_code:string,country_name:string,is_anonymous_proxy:bool,is_satellite_provider:bool)
    ['https://raw.githubusercontent.com/datasets/geoip2-ipv4/master/data/geoip2-ipv4.csv'];
let IPs = datatable(ip:string, continent_name:string, country_iso_code:string)
[
    '2.20.183.12',   'Europe', 'GB', // United Kingdom
    '5.8.1.2',       'Europe', 'RU', // Russia
    '192.165.12.17', 'Europe', '',   // Sweden is 'SE' - so it won't be matched
];
IPs
| evaluate ipv4_lookup(IP_Data, ip, network, continent_name, country_iso_code)
```

Output

ip	continent_name	country_iso_code	network	geoname_id	continent_code	country_name	is_anonymous_proxy	is_satellite_provi
2.20.183.12	Europe	GB	2.20.183.0/24	2635167	EU	United Kingdom	0	0
5.8.1.2	Europe	RU	5.8.0.0/19	2017370	EU	Russia	0	0

Feedback

Was this page helpful? Yes No

Provide product feedback | Get help at Microsoft Q&A

preview plugin

Article • 01/26/2023

Returns a table with up to the specified number of rows from the input record set, and the total number of records in the input record set.

Syntax

T `|` `evaluate` `preview(`*NumberOfRows*`)`

Parameters

Name	Type	Required	Description
T	string	✓	The table to preview.
NumberOfRows	int	✓	The number of rows to preview from the table.

Returns

The `preview` plugin returns two result tables:

- A table with up to the specified number of rows. For example, the sample query above is equivalent to running `T | take 50`.
- A table with a single row/column, holding the number of records in the input record set. For example, the sample query above is equivalent to running `T | count`.

Tip

If `evaluate` is preceded by a tabular source that includes a complex filter, or a filter that references most of the source table columns, prefer to use the **materialize** function. For example:

Example

Run the query

Kusto

```
StormEvents | evaluate preview(5)
```

Table1

The following output table only includes the first 6 columns. To see the full result, run the query.

StartTime	EndTime	EpisodeId	EventId	State	EventType	...
2007-12-30T16:00:00Z	2007-12-30T16:05:00Z	11749	64588	GEORGIA	Thunderstorm Wind	...
2007-12-20T07:50:00Z	2007-12-20T07:53:00Z	12554	68796	MISSISSIPPI	Thunderstorm Wind	...
2007-09-29T08:11:00Z	2007-09-29T08:11:00Z	11091	61032	ATLANTIC SOUTH	Waterspout	...
2007-09-20T21:57:00Z	2007-09-20T22:05:00Z	11078	60913	FLORIDA	Tornado	...
2007-09-18T20:00:00Z	2007-09-19T18:00:00Z	11074	60904	FLORIDA	Heavy Rain	...

Table2

Count
59066

Feedback

schema_merge plugin

Article • 01/31/2023

Merges tabular schema definitions into a unified schema.

Schema definitions are expected to be in the format produced by the getschema operator.

The `schema merge` operation joins columns in input schemas and tries to reduce data types to common ones. If data types can't be reduced, an error is displayed on the problematic column.

The plugin is invoked with the evaluate operator.

Syntax

`T` `|` `evaluate` `schema_merge(`*PreserveOrder*`)`

Parameters

Name	Type	Required	Description
PreserveOrder	bool		When set to `true`, directs the plugin to validate the column order as defined by the first tabular schema that is kept. If the same column is in several schemas, the column ordinal must be like the column ordinal of the first schema that it appeared in. Default value is `true`.

Returns

The `schema_merge` plugin returns output similar to what getschema operator returns.

Examples

Merge with a schema that has a new column appended.

Run the query

Kusto

```
let schema1 = datatable(Uri:string, HttpStatus:int)[] | getschema;
let schema2 = datatable(Uri:string, HttpStatus:int, Referrer:string)[] |
getschema;
union schema1, schema2 | evaluate schema_merge()
```

Output

ColumnName	ColumnOrdinal	DataType	ColumnType
Uri	0	System.String	string
HttpStatus	1	System.Int32	int
Referrer	2	System.String	string

Merge with a schema that has different column ordering (`HttpStatus` ordinal changes from `1` to `2` in the new variant).

Run the query

Kusto

```
let schema1 = datatable(Uri:string, HttpStatus:int)[] | getschema;
let schema2 = datatable(Uri:string, Referrer:string, HttpStatus:int)[] |
getschema;
union schema1, schema2 | evaluate schema_merge()
```

Output

ColumnName	ColumnOrdinal	DataType	ColumnType
Uri	0	System.String	string
Referrer	1	System.String	string
HttpStatus	-1	ERROR(unknown CSL type:ERROR(columns are out of order))	ERROR(columns are out of order)

Merge with a schema that has different column ordering, but with `PreserveOrder` set to `false`.

Run the query

Kusto

```
let schema1 = datatable(Uri:string, HttpStatus:int)[] | getschema;
let schema2 = datatable(Uri:string, Referrer:string, HttpStatus:int)[] |
getschema;
union schema1, schema2 | evaluate schema_merge(PreserveOrder = false)
```

Output

ColumnName	ColumnOrdinal	DataType	ColumnType
Uri	0	System.String	string
Referrer	1	System.String	string
HttpStatus	2	System.Int32	int