sumo logic

SEARCH OPERATORS

PARSING

parse regex

```
| parse regex "[0-9A-Za-z-]+\.(?<domain>
[A-Za-z-]+\.(?:co\.uk|com|com\.au))/.*"
Field Option
| parse regex field=url "[0-9A-Za-z-]+\.(?<domain>
[A-Za-z-]+\.(?:co\.uk|com|com\.au))/.*"
```

parse (anchor)

```
| parse "User=*:" as user
Field Option
| parse field=xyz "User=*:" as user
Nodrop Option
| parse "a=*," as a nodrop
```

JSON

```
| json field=jsonobject "sessionId"
or
| json auto
```

csv

```
| csv _raw extract 1 as user
```

split

```
| split text delim=':' extract 1 as user,
2 as account_id, 3 as session_id, 4 as result
```

xml

```
| parse XML "/af/@type"
```

parseHex

```
| parseHex("12D230") as decimalValue
```

keyvalue

```
| keyvalue infer "module", "thread"
```

parseDate

```
| parseDate(strDate, dateFormat)
| parseDate(strDate, dateFormat, timeZone)
```

AGGREGATING

avg

```
| avg(request_received) by hour
```

first and last

```
| first(error_message) by hostname
| last(error_message) by fieldname
```

pct

```
| pct(value, 95) as value_95pct
```

count, count_distinct, and count_frequent

```
| count by url
| count_distinct(username) by hostname
| count_frequent srcIP, url
```

fillmissing

```
| fillmissing values("backend", "database",
"webapp") in _sourceCategory
```

stddev

```
| stddev(request_received) by hour
```

most_recent and least_recent

```
| parse ... as status | withtime status | most_recent(status_withtime)
by _sourcehost
| parse ... as status | withtime status | least_recent(status_withtime)
by _sourcehost
```

min and max

```
| max(request_received) by hour
| min(request_received) by hour
```

sum

```
| sum(bytes_received) by hostname
```

SEARCH OPERATORS

accum

```
| count as requests by _timeslice,cs_username
| sort by _timeslice asc,cs_username
| accum requests as running_total
```

(as) operator

```
| parse "* - - " as ip_addr
| ip_addr as src_ip
```

backshift

```
| count by _timeslice
| sort + _timeslice
| backshift _count,10 as size
```

base64Encode

```
| base64Encode ("hello world")
as base64
```

base64Decode

```
| base64Decode("aHR0cDovL2NvZGVjLmFw
YWNoZS5vcmcvY29tbW1vbnM=") as V
```

CIDR

```
| where compareCIDRPrefix("10.10.1.32", ip_address, toInt(27))
| where getCIDRPrefix("10.10.1.35", toInt(24))`
| maskFromCIDR(toInt(24)) as s`
```

concat

```
| concat(field1, field2, field3) as new_string
```

contains

```
| contains("hello world", "hello") as containing`
```

decToHex

```
| decToHex("4919") as V
```

diff

```
| diff bytes as diff_bytes
```

fields

```
| fields method, status_code
```

filter

```
_sourceCategory=HttpServers | timeslice 1m | count by _timeslice, _sourceHost
| filter _sourcehost in (outlier _count by _sourceHost | where _count_viola-
tion > 0)  | transpose row _timeslice column _sourcehost
```

format

```
| parse "fiveMinuteRate=*," as rate
| format("%s : %s","Five Minute Rate is" , rate) as formattedVal
```

formatDate

```
| formatDate(now(), "YYYY-MM-dd") as today
```

geo lookup

```
| parse "remote_ip=*]" as remote_ip
| lookup latitude, longitude, country_code, country_name, region, city,
postal_code, area_code, metro_code from geo://default on ip = remote_ip
| count by latitude, longitude, country_code, country_name, region, city,
postal_code, area_code, metro_code
| sort _count
```

haversine

```
| haversine(39.04380, -77.48790, 45.73723, -119.81143) as distanceKMs
```

hexToDec

| hexToDec("0000000000001337") as V

if

```
| if(status_code matches "5*", 1, 0) as server_error
Or
| status_code matches "5*" ? 1 : 0 as server_error
```

?

```
| status_code matches "5*"
? 1 : 0 as server_error
```

in

```
| if (status_code in ("500", "501", "502",
"503", "504", "505", "506", "401", "402",
"403", "404"), "Error", "OK") as status_code_type
```

ipv4ToNumber

```
| ipv4ToNumber(ip) as num
```

isBlank

```
| where isBlank(user)
```

isEmpty

```
| if(isEmpty(src_ip),1,0)
as null_ip_counts
```

isNull

```
| where isNull(src_ip)
```

isNumeric

```
| where isNumeric(num)
```

isPrivateIP

```
| where isPrivateIP(src_ip)
```

isPublicIP

```
| where isPublicIP(src_ip)
```

isValidIP

```
| where isValidIP(src_ip)
```

join

```
("starting stream from" OR "starting search")
| join
(parse "starting stream from *" AS a) AS T1,
(parse "starting search * from parent stream *"
AS b, c) AS T2 on T1.a = T2.c
```

length

```
| where length(query) <= 20
```

limit

```
| count by _sourceCategory
| sort by _count
| limit 5
```

lookup

```
| parse "name=*, phone number=*," as (name,
phone)
| count by name, phone
| lookup email from https://company.com/us-
erTable.csv
 on name=userName, phone=cell
```

logcompare

```
| logcompare timeshift -24h
```

logreduce

```
| logreduce
```

luhn

```
| "6666-7777-6666-8888" as b
| luhn(b) as d`
```

matches

```
| if (agent matches "*MSIE*","Internet Explorer","Other") as Browser
| if (agent matches "*Firefox*","Firefox",Browser) as Browser
```

now

```
| now() as current_date
```

num

```
| num(duration)
```

outlier

```
_sourceCategory=IIS/Access
| parse regex "\d+-\d+-\d+ \d+:\d+:\d+ (?<server_ip>\S+) (?<method>\S+)
(?<cs_uri_stem>/\S+?) \S+ \d+ (?<user>\S+) (?<client_ip>[\.\d]+) "
| parse regex "\d+ \d+ \d+ (?<response_time>\d+)$"
| timeslice 1m
| max(response_time) as response_time by _timeslice
| outlier response_time window=5,threshold=3,consecutive=2,direction=+-
```

predict

```
| count by _timeslice
| toDouble(_count)
| predict _count by 1m forecast=5
```

queryEndTime()

```
| queryEndTime() as endtime
```

queryStartTime()

```
| queryStartTime() as starttime
```

queryTimeRange ()

```
| queryTimeRange() as timerange
```

replace

```
| replace(query, ".","->") as query
```

rollingstd

```
| rollingstd _count,1 by _sourcehost
```

save

```
| save /shared/lookups/daily_users
```

sessionize

```
| sessionize "id=*" as requestId, "sessionId=* , rId=$requestId" as
sessionId
```

smooth

```
| smooth _count,1 by _sourcehost
```

sort

```
| count as page_hits by _sourceHost
| sort by page_hits asc
```

substring

```
| substring("Hello world!", 6)
```

timeslice

```
| timeslice 1h
| count by _timeslice
```

toLowerCase

```
| toLowerCase(username) as username
```

toUpperCase

```
| toUpperCase(_sourceHost) as _sourceHost
```

top

```
| top 5 _sourcecategory
```

topk

```
| topk(2,_count) by _sourceHost
```

total

```
| total gbytes as total_memory
```

tourl

```
| tourl("https://www.sumologic.net/ui/#section/search/H10KM-
VHzntXo9PrFAumuFemdU27f2iqU7bA3U7Lq", "Akamai Denials by Host") as AkamaiQuery
```

trace

```
|  trace "ID=( [0-9a-fA-F] {4} )" "7F92"
```

transpose

```
_sourceCategory=service
| parse "Successful login for user '*', organization: '*'" as user, org_id
| timeslice 1d
| count _timeslice, user
| transpose row _timeslice column user
```

trim

```
| trim(" Hello World  ") as greet
```

urldecode

```
| urlencode(url) as url
```

where

```
| where status_code matches "4*"
| where !(status_code matches "2*")
```

MATH EXPRESSIONS

BASIC

abs

```
| abs(-1.5) as v
```

round

```
| round((bytes/1024)/1024) as v
```

ceil

```
| ceil(1.5) as v
```

floor

```
| floor(1.5) as v
```

max

```
| max(1, 2) as v
```

min

```
| min(1, 2) as v
```

sqrt

```
| sqrt(4) as v
```

cbrt

```
| cbrt(8) as v
```

TRIGONOMIC

sin

```
| sin(1) as v
```

asin

```
| asin(1) as v
```

atan2

```
| atan2(0, -1) as v
```

tanh

```
| tanh(x) as v
```

cosh

```
| cosh(x) as v
```

cos

```
| cos(1) as v
```

acos

```
| acos(x) as v
```

sinh

```
| sinh(x) as v
```

tan

```
| tan(1) as v
```

atan

```
| atan(x) as v
```

EXPONENTS & LOGS

exp

```
| exp(1) as v
```

expm1

```
| expm1(0.1) as v
```

log

```
| log(2) as v
```

log10

```
| log10(2) as v
```

log1p

```
| log1p(0.1) as v
```

ADVANCED

hypot

```
| hypot(1, 0) as v
```

toDegrees

```
| toDegrees(asin(1)) as v
```

toRadians

```
| toRadians(180) as v
```

TRANSACTION ANALITICS

transaction

```
| transaction on sessionid fringe=10m
with "Starting session *" as init,
with "Initiating countdown *" as countdown_start,
with "Countdown reached *" as countdown_done,
with "Launch *" as launch
results by transaction
```

transactionize

```
| parse "[system=001] [sessionId=*]" as system1Id nodrop
| parse "[system=002][sessionId=*]" as system2Id nodrop
| parse "[system=003][sessionId=*]" as system3Id nodrop
| parse "system=001 with sessionId=*" as system1Id nodrop
| transactionize system1Id, system2Id, system3Id
```

merge

```
| parse "BytesSentPersec =
\"*\"" as BytesPersec
| merge BytesPersec join with
"--", _messageTime takeLast
```