

# 검색 명령어

데이터 나열, 변환	table, rename, fields, dedup, sort
통계 계산	stats, top, rare, len(x)
차트 시각화	timechart, chart
비교분석	eval, case, cidrmatch, ff, line, match
다중 문자열과 시간	mvindex, split, substr, round, ruldecode, strftime, strptime, now

# top

- **지정한 필드에서 가장 많이 나오는 값을 보여주는 명령어**

| top limit=<숫자> [showperc=T/F] [showcount=T/F] [useother=T/F] 필드1, 필드2 by 필드

- 검색 결과를 파이프로 입력 받아서 계산 한 후에 결과를 반환
- showperc와 showcount의 기본값이 T이므로 지정하지 않더라도 비율과 개수를 보여줌

limit	반환되는 결과 개수를 지정 숫자를 지정하지 않으면 기본적으로 10개로 선정 전체 값을 모두 보려면 limit=0을 설정
showperc	해당 값이 차이 하는 비율을 보여줌 기본 값이 T
showcount	해당 값의 개수를 보여줌 기본 값이 T
useother	Top10 외 다른 숫자의 크기를 알아보려면 T를 입력

# table

- 필드명과 결합해 검색 결과를 테이블 형태로 보여줌
- table 명령어 다음에 확인하고 싶은 필드명 적음
- 필드명을 잘못 적으면 해당 필드에 공란이 보임
  - 다른 필드에는 값이 보이는데 특정 필드가 공백이면 필드명 입력오류 의심
- 필드명은 대소문자를 구분
- 여러 개의 필드는 쉼표(,) 또는 띄어쓰기로 구분

table <<필드1>> <<필드2>> ...<<필드n>>

index=main sourcetype=access\_combined\_wcookie  
| table clientip, method, productId, status

새로운 검색

다른 이름으로 저장 ▼   테이블 보기 만들기   닫기

index=main sourcetype=access\_combined\_wcookie  
| table clientip method productId status

최근 7일 ▼

✓ 29,014개의 이벤트 (22/07/02 15:00:00.000 ~ 22/07/09 15:22:13.000)   이벤트 샘플링 없음 ▼   작업 ▼   ||   ■ ▼   →   🖨   ⬇   🗨 상세 모드 ▼

이벤트 (29,014)   패턴   통계 (29,014)   시각화

페이지당 20개 ▼   ✎ 형식   미리보기 ▼

clientip ▼ ✎	method ▼ ✎	productId ▼ ✎	status ▼ ✎
91.205.189.15	GET		200
91.205.189.15	GET		200
182.236.164.11	GET	BS-AG-G09	200
182.236.164.11	POST	SF-BVS-G01	408
182.236.164.11	GET		200
182.236.164.11	POST		200
182.236.164.11	POST	MB-AG-G07	200
182.236.164.11	GET		200
182.236.164.11	GET	WC-SH-G04	200

# rename

- 필드명을 다른 이름으로 변경
- 필드명을 띄어쓰기로 구분하고 싶으면 원하는 필드명을 따옴표로 표시

rename <<원래 필드명>> AS <<변경 필드명>>

index=main sourcetype=access\_combined\_wcookie


| table clientip, action, productId, status

| **rename** action AS “Customer Action”, productId AS ProductID, status AS “HTTP Status”

새로운 검색

다른 이름으로 저장 ▼ 테이블 보기 만들기 닫기

index=main sourcetype=access\_combined\_wcookie  
| table clientip, action, productId, status  
| rename action AS "Customer Action", productId AS "ProductID", status AS "HTTP Status"

최근 7일 ▼ 

✓ 29,014개의 이벤트 (22/07/02 15:00:00.000 ~ 22/07/09 15:37:09.000) 이벤트 샘플링 없음 ▼

작업 ▼ || ■ → 🖨️ ⬇️ 🔍 상세 모드 ▼

이벤트 (29,014) 패턴 통계 (29,014) 시각화

페이지당 20개 ▼ ✂️ 형식 미리보기 ▼

< 이전 1 2 3 4 5 6 7 8 ... 다음 >

clientip ⇅	Customer Action ⇅	ProductID ⇅	HTTP Status ⇅
91.205.189.15			200
91.205.189.15			200
182.236.164.11	addtocart	BS-AG-G09	200
182.236.164.11		SF-BVS-G01	408
182.236.164.11			200
182.236.164.11	purchase		200
182.236.164.11	purchase	MB-AG-G07	200
182.236.164.11			200

# dedup

- 검색 결과에서 중복을 제거

dedup << 필드명1>>, << 필드명2>>

- 중복 제거는 지정한 필드를 기준으로 실행
- 두 개 이상의 필드에서 중복을 제거하려면 쉼표(,)로 구분
- 중복을 제거하면 해당 필드의 유일한 값을 기준으로 결과를 보여줌
- 중복을 제거하면 다른 필드의 값은 중복이 아닌데도 중복된 결과를 제거하면서 사라짐
- dedup은 상세 분석이 아니라 해당 필드에 존재하는 값을 확인하는 정도로 사용

index=main sourcetype=access\_combined\_wcookie status=404

## 새로운 검색

index=main sourcetype=access\_combined\_wcookie status=404

✓ 507개의 이벤트 (22/07/02 15:00:00.000 ~ 22/07/09 15:55:21.000) 이벤트 샘플링 없음 ▼

이벤트 (507) 패턴 통계 시각화

index=main sourcetype=access\_combined\_wcookie status=404 | **dedup host**

## 새로운 검색

index=main sourcetype=access\_combined\_wcookie status=404 | dedup host

✓ 1개의 이벤트 (22/07/02 15:00:00.000 ~ 22/07/09 15:56:26.000) 이벤트 샘플링 없음 ▼

이벤트 (1) 패턴 통계 시각화



# sort

- 검색 결과를 정렬

sort (+|-) << 필드명1>>

- 기본 값은 오름차순 정렬
- 내림차순으로 정렬하려면 필드 앞에 마이너스(-)를 붙임

index=main sourcetype=access\_combined\_wcookie


| table clientip, action, productId, status

| | **sort** action, -productId





새로운 검색

다른 이름으로 저장 ▼   테이블 보기 만들기   닫기


index=main sourcetype=access\_combined\_wcookie  
| table clientip, action, productId, status  
| sort action, -productId





최근 7일 ▼ 

✓ 28,737개의 이벤트 (22/07/02 16:00:00.000 ~ 22/07/09 16:02:48.000)   이벤트 샘플링 없음 ▼

작업 ▼   ||               상세 모드 ▼

이벤트 (28,737)   패턴   통계 (10,000)   시각화

페이지당 20개 ▼    형식   미리보기 ▼

clientip ▼ 	action ▼ 	productId ▼ 	status ▼ 
74.53.23.135	addtocart	WC-SH-T02	200
74.53.23.135	addtocart	WC-SH-T02	200
12.130.60.5	addtocart	WC-SH-T02	200
12.130.60.5	addtocart	WC-SH-T02	200
193.33.170.23	addtocart	WC-SH-T02	200
203.223.0.20	addtocart	WC-SH-T02	200
109.169.32.135	addtocart	WC-SH-T02	200
125.17.14.100	addtocart	WC-SH-T02	200
201.122.42.235	addtocart	WC-SH-T02	200

index=main sourcetype=access\_combined\_wcookie

| table clientip, action, productId, status

| | **sort** -ip(clientip)

\*IP주소를 정렬 할 경우 함수 ip() 또는 num() 사용

새로운 검색

다른 이름으로 저장 ▼   테이블 보기 만들기   닫기

index=main sourcetype=access\_combined\_wcookie  
| table clientip, action, productId, status  
| sort -ip(clientip)

최근 7일 ▼ 

✓ 28,737개의 이벤트 (22/07/02 16:00:00.000 ~ 22/07/09 16:06:02.000)   이벤트 샘플링 없음 ▼    작업 ▼                상세 모드 ▼

이벤트 (28,737)   패턴   통계 (10,000)   시각화

페이지당 20개 ▼    형식   미리보기 ▼

clientip ▼ 	action ▼ 	productId ▼ 	status ▼ 
233.77.49.94	addtocart	WC-SH-A02	200
233.77.49.94			200
233.77.49.94		WC-SH-A02	200
233.77.49.94	view	SC-MG-G10	200
233.77.49.94	view	FI-AG-G08	200
233.77.49.94	view	MB-AG-T01	200
233.77.49.94			200

# fields

- 검색 결과에서 특정 필드를 포함시키거나 제거 시 사용

fields + << 필드명>>

fields - << 필드명>>

- +는 기본값이므로 반드시 명시할 필요는 없음
- 필드를 추출하기 전에 동작하므로 검색 결과에서 원하는 필드만 선택 시 전체

검색 성능을 높일 수 있음

# stats

- 각종 통계 함수를 이용해서 데이터 계산

| stats [count|dc|sum|avg|list|values] by [필드명]

함수명	설명
count(x)	X필드의 개수를 반환
dc(x)	X 필드의 중복을 제거한 개수 반환
sum(x)	X필드의 총합 반환
avg(x)	X필드의 평균 반환
list(x)	X필드의 목록으로 만들어 반환
values(x)	X필드의 중복을 제거한 목록반환
max(x) / median(x) / min(x)	X필드의 최대/중앙/최소값 반환
var(x)	X필드의 분산값 반환
stdev(x)	X필드의 표준편차 반환

```
index=main sourcetype="access_combined_wcookie"
```

```
| stats sum(bytes), avg(bytes), max(bytes), median(bytes), min(bytes) by clientip
```

\* 사용자마다 외부로 전송하는 바이트의 총합, 최대, 중간, 최소값을 검색

새로운 검색 다른 이름으로 저장 ▼ 테이블 보기 만들기 닫기

index=main sourcetype="access\_combined\_wcookie"

| stats sum(bytes), avg(bytes), max(bytes), median(bytes), min(bytes) by clientip

전체 시간

✓ 118,596개의 이벤트 (22/07/20 20:09:52.000 이전) 이벤트 샘플링 없음 ▼ 작업 ▼ || ■ ↶ ↷ ⬇ ⬆ 상세 모드 ▼

이벤트 (118,596) 패턴 통계 (182) 시각화

페이지당 20개 ▼ ✎ 형식 미리보기 ▼ < 이전 1 2 3 4 5 6 7 8 ... 다음 >

clientip	sum(bytes)	avg(bytes)	max(bytes)	median(bytes)	min(bytes)
107.3.146.207	2408067	2090.3359375	3996	2086	203
108.65.113.83	1590204	2181.349794238683	3963	2261	218
109.169.32.135	2677236	2114.7203791469196	3995	2162	203
110.138.30.229	1088547	2073.422857142857	3994	2018	208
110.159.208.78	1588872	2068.84375	3999	2051	208
111.161.27.20	1350723	2175.07729468599	4000	2243	234

# rare

- top과 반대의 결과인 빈도가 적은 값의 순서를 추출

| rare limit=<숫자> [showperc=T/F] [showcount=T/F] [useother=T/F] 필드1, 필드2 by 필드

\* | rare useother=T clientip by method

- method 필드를 기준으로 하위 10개의 clientip를 보여줌

새로운 검색

다른 이름으로 저장 ▼ 테이블 보기 만들기 닫기

\* | rare useother=T clientip by method전체 시간 🔍

✓ 329,592개의 이벤트 (22/07/20 20:55:23.000 이전) 이벤트 샘플링 없음 ▼작업 ▼ ⏸ ■ ↶ 🖨 ⬇ 🗒 상세 모드 ▼

이벤트 (329,592)패턴통계 (22)시각화

페이지당 20개 ▼ ✎ 형식 미리보기 ▼< 이전 1 2 다음 >

method ⚙	clientip ⚙	count ⚙	percent ⚙
GET	142.233.200.21	177	0.237272
GET	212.58.253.71	183	0.245315
GET	131.178.233.243	204	0.273466
GET	207.36.232.245	225	0.301617
GET	59.36.99.70	225	0.301617
GET	223.205.219.67	231	0.309660
GET	86.9.190.90	234	0.313681



# len(x)

- 문자열의 길이를 양의 정수값으로 돌려줌
- 문자열의 길이를 활용해서 비정상적인 값을 추출

# 차트 시각화

- 차트와 같은 데이터 시각화는 원본 로그 데이터로 차트를 그리는 것이 불가능
  - 데이터를 차트로 만들려면 먼저 관련 데이터를 추출
  - 시각화 원본 데이터를 통계 테이블로 변환
- 차트 관련 명령어 : `chart`, `timechart`

# timechart

- 시간에 따른 통계 테이블 생성

timechart span=[시간범위] 통계함수 by [필드명]

- 시간 필드가 X축, 실제데이터가 Y축에 표시되는 차트 시각화 형식으로 나타냄
- 시간에 따른 통계의 추세를 표시할 때 많이 사용
- Span : 시간 계산단위 설정
- Count : 전체 개수를 계산하는데 12시간 단위로 총 숫자를 구해서 보여줌

```
index=main sourcetype="access_combined_wcookie"
| timechart span=12h count(clientip) as "Access Count"
```

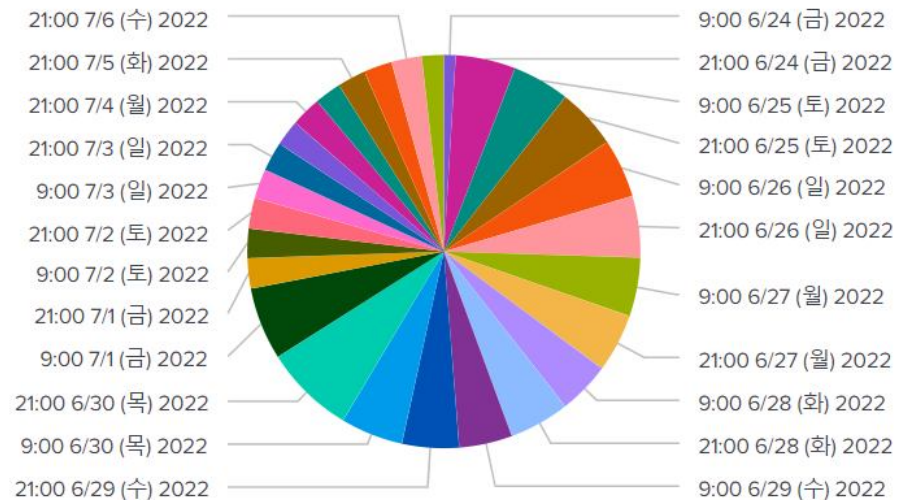
## 새로운 검색

```
index=main sourcetype="access_combined_wcookie"
| timechart span=12h count(clientip) as "Access Count"
```

✓ 118,596개의 이벤트 (22/07/20 21:07:59.000 이전) 이벤트 샘플링 없음 ▼

이벤트 (118,596)   패턴   통계 (27)   시각화

📊 Pie Chart   ✎ 형식   📏 격자



# chart

- Timechart와 유사한 방식으로 동작

chart [통계함수] over X축 by [기준필드]

- Timechart는 span 옵션을 사용해서 시간 단위를 지정 할 수 있지만 char명령어는 해당 옵션을 사용할 수 없음

```
index=main sourcetype="access_combined_wcookie"  
| chart dc(clientip) as "Unique Count" over date_wday
```

## 새로운 검색

```
index=main sourcetype="access_combined_wcookie"  
| chart dc(clientip) as "Unique Count" over date_wday
```

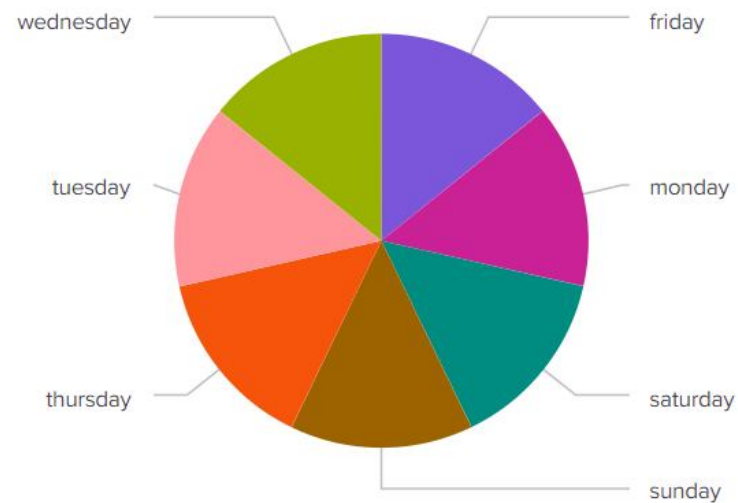
✓ 118,596개의 이벤트 (22/07/20 21:22:08.000 이전) 이벤트 샘플링 없음 ▼

이벤트 (118,596)   패턴   통계 (7)   시각화

● Pie Chart

✎ 형식

≡ 격자



# eval

- 검사 결과 값의 변환, 검증을 수행하며 **함수 실행 결과 값을 반환**

| eval [반환값\_저장변수] = 함수(인자1, 인자2..)

```
sourcetype=access_*
```

```
| eval status_code=if(status==200, "OK", "Error")
```

```
| table clientip status_code
```

```
index=httplog sourcetype=httplog
```

```
| eval list="mozilla"
```

```
| `ut_parse_extended(url,list)`
```

```
| table ut_netloc, ut_domain, ut_subdomain, ut_domain_without_tld, ut_tld
```

```
| dedup ut_netloc
```

## **case(X,"Y",...)**

- 여러 개의 조건을 검증할 때 사용
- 두 개의 인자가 한 그룹으로 동작
- 첫 번째 인자가 참인 경우 두 번째 인자의 내용이 반환, 세번째 인자가 참이면  
네 번째 인자가 수행

```
index=httplog sourcetype=httplog
```

```
| eval description=case(error==404, "Not found", error==500, "internal Server Error")
```

```
| table clientip description
```

```
| eval quarter=case(date_month=="January", "1Q", date_month=="April", "2Q")
```



## cidrmatch(“X”,Y)

- IP주소 Y가 네트워크 범위 X에 존재하는지 확인
- 반환 값은 참 또는 거짓이며 두 개의 인자가 사용
- 첫 번째 CIDR 형식의 네트워크 주소 범위, 두 번째는 검사를 위한 IP 주소가 입력

```
| eval local=cidrmatch(“10.0.0.0/8”, “10.10.0.100”)
```

**\* IP 주소 10.10.0.100이 10.0.0.0/8 대역에 포함되면 true 아니면 false 반환**

- cidrmatch 함수는 검색 필터로 사용할 수 있음

```
| where (cidrmatch(“10.0.0.0/8”, ip) OR  
        (cidrmatch(“172.16.0.0/12”, ip) OR  
        (cidrmatch(“192.16.0.0/16”, ip)
```

## if(X,Y,Z)

- X가 참이면 Y를 실행하고, X가 거짓이면 Z 실행

```
* | eval ip1="10.10.0.100", ip2="100.10.0.100"  
| eval network1=if(cidrmatch("10.10.0.0/24", ip1),"local", "external"),  
network2=if(cidrmatch("10.10.0.0/24", ip2),"local", "external")  
| table ip1, network1, ip2, network2
```

- IP 필드 값이 10.10.0.100이라면 10.10.0.0/24 네트워크에 포함되므로 network1 필드에 “local” 문자열이 저장
- IP 필드 값이 100.10.0.100이라면 network2 필드에는 “external” 이 할당

## 새로운 검색

다른 이름으로 저장 ▼

```
* | eval ip1="10.10.0.100", ip2="100.10.0.100"
| eval network1=if(cidrmatch("10.10.0.0/24", ip1),"local", "external"), network2=if(cidrmatch("10.10.0.0/24", ip2),"local", "external")
| table ip1, network1, ip2, network2
```

✓ 329,592개의 이벤트 (22/07/20 22:10:32.000 이전) 이벤트 샘플링 없음 ▼

작업 ▼ || ■

이벤트 (329,592) 패턴 통계 (329,592) 시각화

페이지당 20개 ▼ ✎ 형식 미리보기 ▼

< 이전 1 2 3 4

ip1 ⇅ ✎	network1 ⇅ ✎	ip2 ⇅ ✎	network2 ⇅
10.10.0.100	local	100.10.0.100	external
10.10.0.100	local	100.10.0.100	external
10.10.0.100	local	100.10.0.100	external
10.10.0.100	local	100.10.0.100	external
10.10.0.100	local	100.10.0.100	external
10.10.0.100	local	100.10.0.100	external
10.10.0.100	local	100.10.0.100	external
10.10.0.100	local	100.10.0.100	external

## like(X,“Y”)

- Like 함수의 X필드에서 일부 문자열인 Y를 검색
- 첫 번째 인자는 대상 필드, 두 번째 인자는 정규 표현식의 탐색 패턴
- X에서 Y를 찾을 수 있으면 참을 반환
- like 함수의 와일드 카드 문자열은 ‘%’

```
...| where like(field, “add%”)
```

- field 변수가 addr로 시작하는지 검사

## match(X, “Y”)

- Like 함수가 일부라도 맞는 값을 찾는다면 match함수는 함수명과 같이 정확한 일치여부를 비교

```
match(filename, “malicious.exe”)
```

- Filename이 malicious.exe와 정확히 같으면 참, 그렇지 않으면 거짓을 반환
- 문자열 비교에 대소문자는 구분하지 않음

```
match(filename, “(.jpg|.gif|.png)$”)
```

- \$는 종결자로서 ‘\$’ 앞의 문자로 단어가 끝난다는 의미
- ‘|’는 다중 선택을 의미
- Filename 필드 값이 .jpg, gif, .png로 종결(\$)하는지 검사
- 파일 확장자가 jpg, gif, png 여부 검사

```
match(method, "(GET|POST|-)")
```

method가 GET, POST, 또는 “-” 인지 검사

```
NOT match(method, "(GET|POST|-)")
```

method가 GET, POST, 또는 “-” 가 아닌 메소드들을 검사

```
index=httplog sourcetype=httplog  
| where NOT match(method, "(GET|POST|-)")  
| stats count(src) as src_count by method  
| sort - src_count
```

- ❶ method가 GET, POST, 또는 “-” 가 아닌 것들을 메소드들을 검사
- ❷ 검사된 메소드들 별로 송신지 주소 개수를 세서 결과 반환
- ❸ 개수 Count(src)를 src\_count에 저장
- ❹ 저장된 src\_count 값을 기준으로 내림 차순 정렬

## split(X,"Y")

- 구분자 Y를 이용해서 X를 분할해 다중값 형식으로 변환
- 구분자로 분리한 문자열은 여러 개의 토큰이 발생하므로 주로 `mvindex()`에서 사용
- 이벤트에서 특정 값을 추출할 때 사용

```
/data/utility/tool/GoogleToolbar.exe
```

```
split(uri,"/")
```

```
data
```

```
[0]
```

```
utility
```

```
[1]
```

```
tool
```

```
[2]
```

```
GoogleToolbar.exe
```

```
[3]
```

## mvindex(X,Y,Z)

- 필드 X에 있는 Y 번째 값은 반환 (Z 생략가능)
- Y는 인덱스 번호 ( 0 : 첫 번째 값, -1 : 인덱스를 뒤에서 시작, -2 : 끝에서 두 번째)
- 세 번째 인자인 Z는 선택적으로 사용
  - Z값을 지정하면 함수는 Y부터 Z까지의 값을 반환

```
/data/utility/tool/GoogleToolbar.exe
```

```
mvindex(split(uri,"/"), -1)
```

data	utility	tool	GoogleToolbar.exe
[0]	[1]	[2]	[3]
0		-2	-1



# substr(X,Y,Z)

- 세 번째 인자인 Z가 없다면 필드 X의 Y부터 시작해서 문자열 끝까지 반환
- Z가 주어지면 Y부터 Z개의 문자열을 반환

```
*| eval passwd_str="lightdm:x:107:117:Light Display M a n a g e r :/var/lib/lightdm:/bin/false"  
| eval uid=mvindex(split(passwd_str,:),0)  
| eval subuid1=substr(uid,2)  
| eval subuid2=substr(uid,2,4)  
| table uid, subuid1, subuid2
```

lightdm:x:107:117:Light Display M a n a g e r :/var/lib/lightdm:/bin/false

split(passwd\_str,":")

lightdm

x

107

117

Light Display M a n a g e r

:/var/lib/lightdm

/bin/false

[0]

```
uid=mvindex(split(passwd_str,:),0)  
uid=lightdm
```

첫번째 인덱스 문자열

```
subuid1=substr(lightdm,2)  
subuid1=ighdm
```

두 번째 글자부터~끝까지

```
subuid1=substr(lightdm,2,4)  
subuid1=igh
```

두 번째 글자를 포함해서~4개의 문자

```
*| eval passwd_str="lightdm:x:107:117:Light Display Manager:/var/lib/lightdm:/bin/false"
| eval uid=mvindex(split(passwd_str,:),0)
| eval subuid1=substr(uid,2)
| eval subuid2=substr(uid,2,4)
| table uid, subuid1, subuid2
```

## 새로운 검색

```
*
| eval passwd_str="lightdm:x:107:117:Light Disply Manager:/var/lib/lightdm:/bin/false"
| eval uid=mvindex(split(passwd_str,:),0)
| eval subuid1=substr(uid,2)
| eval subuid2=substr(uid,2,4)
| table uid, subuid1, subuid2
```

✓ 329,592개의 이벤트 (22/07/20 22:46:48.000 이전) 이벤트 샘플링 없음 ▼

이벤트 (329,592) 패턴 통계 (329,592) 시각화

페이지당 20개 ▼

형식

미리보기 ▼

< 이전

uid ↕	subuid1 ↕	subuid2 ↕
lightdm	ightdm	ight
lightdm	ightdm	ight
lightdm	ightdm	ight

## **round(X,Y)**

- X를 Y 자리 수 기준으로 반올림
- 나누기 계산을 할 경우 소수점 자리가 급격히 늘어나는 것을 방지

## **urldecode(X)**

- URL 인코딩이 있는 X를 디코딩해 반환
- 웹 주소에 한글이 사용되는 경우 대부분 URL인코딩이 되어 바로 확인이 힘들
- 인코딩 문자열을 디코딩해서 한글이 있는 경우라도 바로 확인 할 수 있음

## **strftime(X,Y)**

- 유닉스 타임 X를 지정한 Y형식으로 출력
- 주로 사용자가 읽기 편한 형식으로 변환 할 때 사용
- 유닉스 타임(에포크 타임) 계산법은 1970년 1월 1일 0시를 기준으로 초를 계산

## **strptime(X,Y)**

- strftime과 반대로 Y형식으로 된 X 시간 문자열을 입력받아서 유닉스 타임을 반환

```
index=main sourcetype="access_combined_wcookie"
```

```
| eval unixtime=strptime(req_time,"%d/%B/%Y:%H:%M:%S")
```

```
| eval humantime=strftime(unixtime,"%Y-%m-%d %H:%M:%S")
```

```
| table req_time, unixtime, humantime
```

## 새로운 검색

[다른 이름으로 저장](#)

```
index=main sourcetype="access_combined_wcookie"
| eval unixtime=strptime(req_time,"%d/%B/%Y:%H:%M:%S")
| eval humantime=strftime(unixtime,"%Y-%m-%d %H:%M:%S")
| table req_time, unixtime, humantime
```

✓ 118,596개의 이벤트 (22/07/20 23:03:47.000 이전) 이벤트 샘플링 없음 ▼

작업 ▼ ||

이벤트 (118,596) 패턴 통계 (118,596) 시각화

페이지당 20개 ▼ ✎ 형식 미리보기 ▼

< 이전 1 2 3

req_time ↕ ✎	unixtime ↕ ✎	humantime ↕
07/Jul/2022:18:22:16	1657185736.000000	2022-07-07 18:22:16
07/Jul/2022:18:22:15	1657185735.000000	2022-07-07 18:22:15
07/Jul/2022:18:20:56	1657185656.000000	2022-07-07 18:20:56
07/Jul/2022:18:20:55	1657185655.000000	2022-07-07 18:20:55

```

index=main sourcetype="access_combined_wcookie"
| eval unixtime=strptime(req_time,"%d/%B/%Y:%H:%M:%S")
| eval date_diff=round((now()-unixtime)/86400,0)
| table req_time, date_diff

```

## 새로운 검색

다른 이름으로 저장 ▼ 테

```

index=main sourcetype="access_combined_wcookie"
| eval unixtime=strptime(req_time,"%d/%B/%Y:%H:%M:%S")
| eval date_diff=round((now()-unixtime)/86400,0)
| table req_time, date_diff

```

✓ 118,596개의 이벤트 (22/07/20 23:08:44.000 이전) 이벤트 샘플링 없음 ▼

작업 ▼ || ■ →

이벤트 (118,596) 패턴 통계 (118,596) 시각화

페이지당 20개 ▼ 형식 미리보기 ▼

< 이전 1 2 3 4 5 6 7 8 ... 다음 >

req_time ↕	date_diff ↕
07/Jul/2022:18:22:16	13
07/Jul/2022:18:22:15	13
07/Jul/2022:18:20:56	13
07/Jul/2022:18:20:55	13
07/Jul/2022:18:20:54	13
07/Jul/2022:18:20:54	13
07/Jul/2022:18:20:54	13
07/Jul/2022:18:20:53	13

- 시간 관련 검색 예약어를 사용해서 설정

earliest=<시간연산자>	검색 시작 시간을 지정
latest=<시간연산자>	검색 끝 시간을 지정

- 시간은 절대 시간과 상대 시간으로 시간 범위를 설정

- 절대 시간 표시 방법 : %m/%d/%Y:%H:%M:%S

(예) 2019년 11월 31일 오후 1시 부터 2019년 12월 1일 오후 1시까지

earliest=11/31/2019:13:0000 latest=12/1/2019:13:00:00

- 상대 시간 범위 지정
  - Splunk 검색에서는 절대 시간보다 상대 시간을 많이 사용
- 상대 시간은 검색어를 실행시키는 시간을 기준으로 과거 또는 미래를 지정

빼기(-)	이전 시간 지정
더하기(+)	이후 시간 지정
s, sec, secs, second, seconds	초
m, min, minute, minutes	분
h, hr, hrs, hour, hours	시간
d, day, days	일
w, week, weeks	주
mon, month, months	월
q, qtr, qtrs, quarter, quarters	분기
y, yr, yrs, year, years	년

\* 숫자를 지정하지 않고 단위만 사용하면 1이 포함된 것으로 간주



- 상대 시간을 사용하면 시간 차이를 이용해서 검색 시간을 지정 할 수 있음
- 맞추기 시간 단위는 사용자가 지정하는 시간 길이에 가장 가까운 시간이나 시간 길이의 종료 시간으로 반내림
- @ 문자로 시간을 지정하면 반내림이 됨

[ + | - ] <시간\_정수><시간단위>@<시간단위>

earliest=-2h latest=now	2시간 이전 부터 현재까지 로그 검색 * 현재 시간이 4시 30분이면 2시 30분으로 설정하여 검색
earliest=-2h@ latest=now	@ 시간 한정지시자가 설정 되어 있으므로 2시 30분이 아니라 가장 가까운 시간으로 반내림(오후 2시가 검색 시작 시간이 됨)
earliest=-mon@mon latest=@mon	이전 달의 모든 이벤트를 검색
earliest=-w@w latest=@w	이전 주의 모든 이벤트를 검색
earliest=-d@d latest=@d	직전 일의 이벤트를 검색