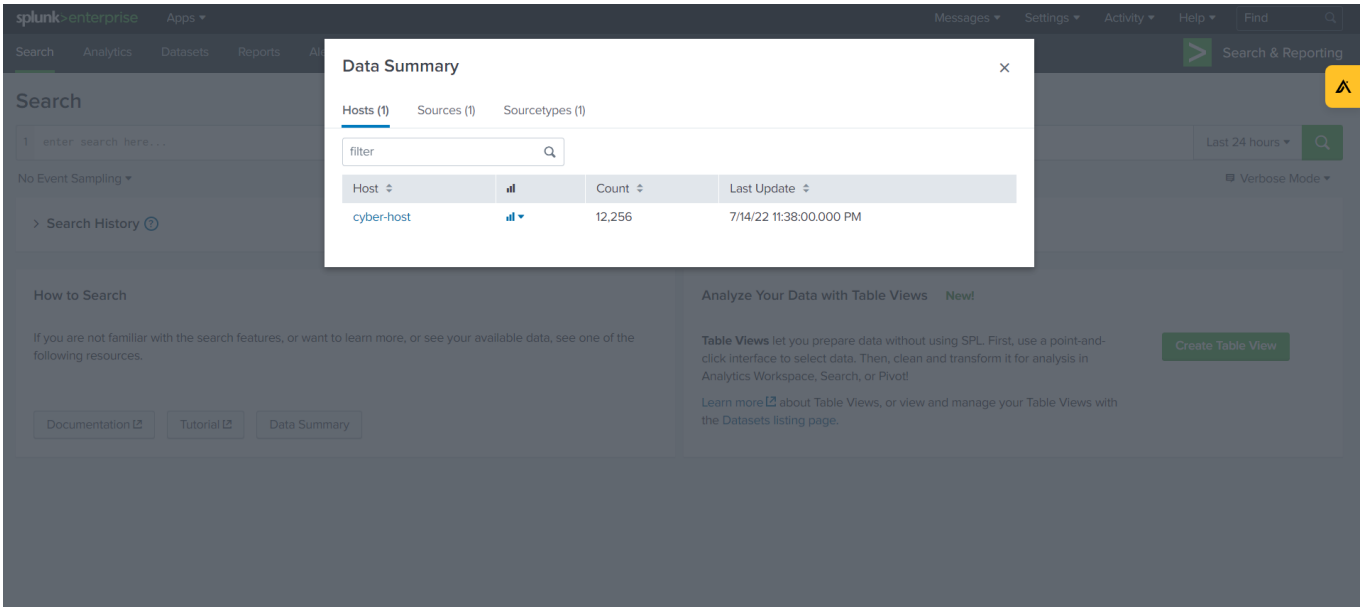
Splunk is a powerful SIEM solution that provides the ability to search and explore machine data. **Search Processing Language (SPL)** is used to make the search more effective. It comprises various functions and commands used together to form complex yet effective search queries to get optimized results.

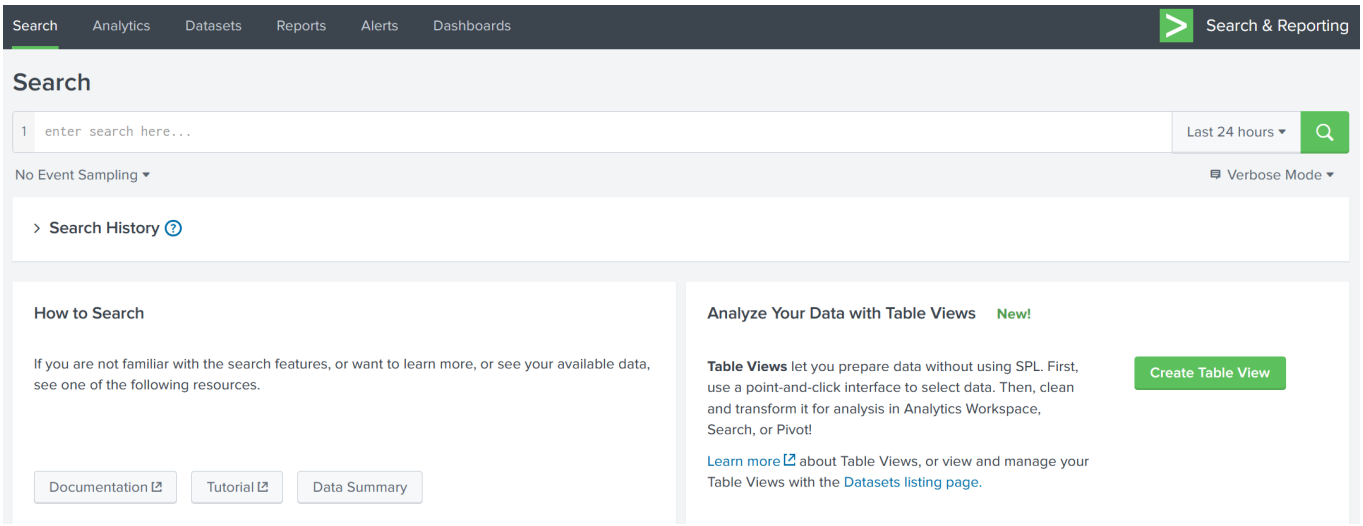
We go to the Serach and data summary. We can see it's cyber host.




What is the name of the host in the Data Summary tab?

cyber-host ✓ Correct Answer Hint

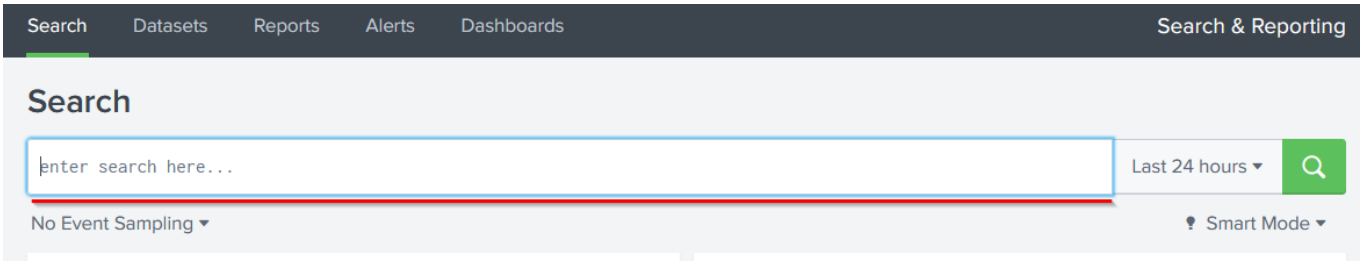
**Search & Reporting App** is the default interface used to search and analyze the data on the Splunk Home page. It has various functionalities that assist analysts in improving the search experience.



Some important functionalities present in the search App are explained below:

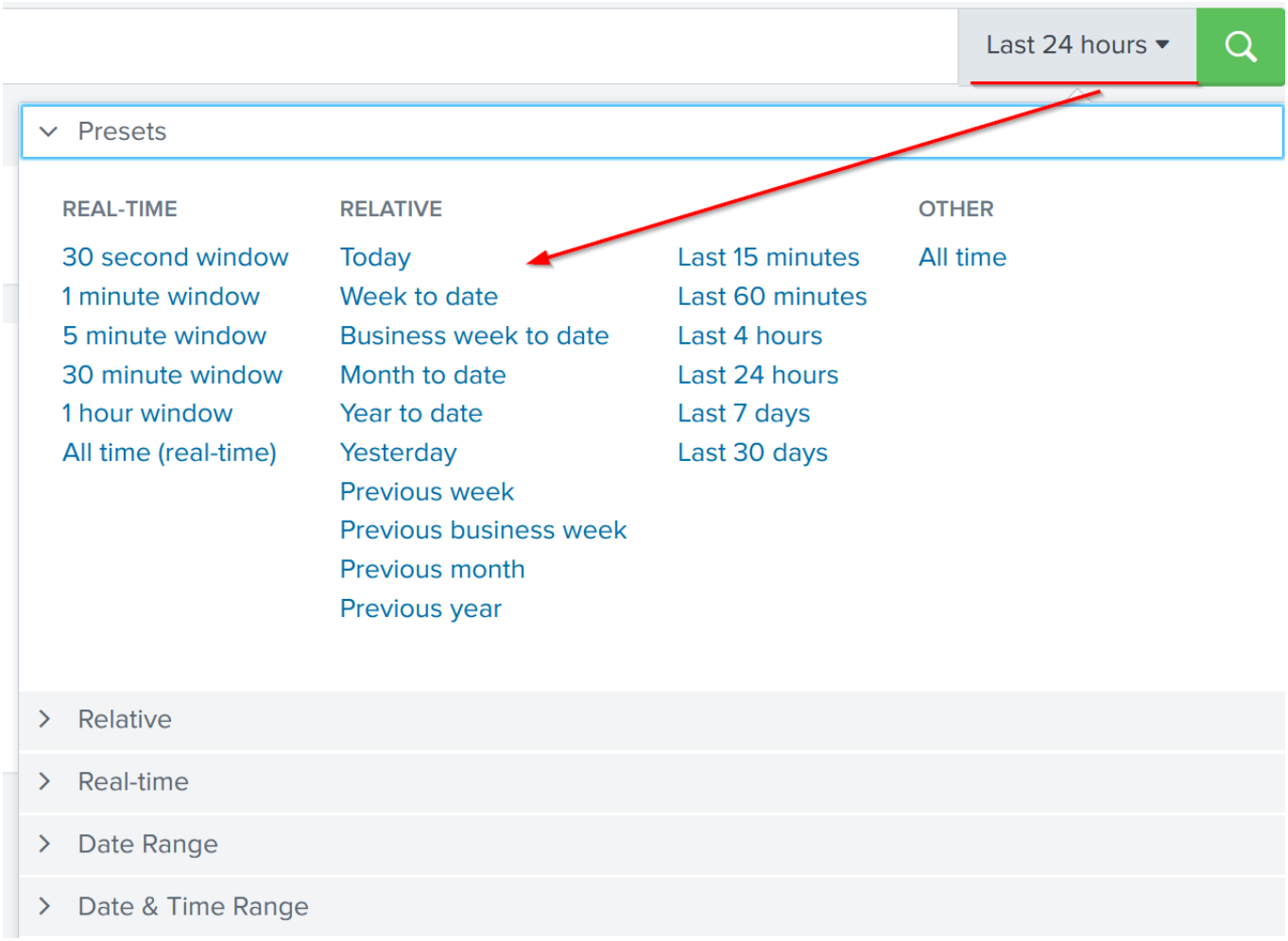
**1) Search Head:**

Search Head is where we use search processing language queries to look for the data.



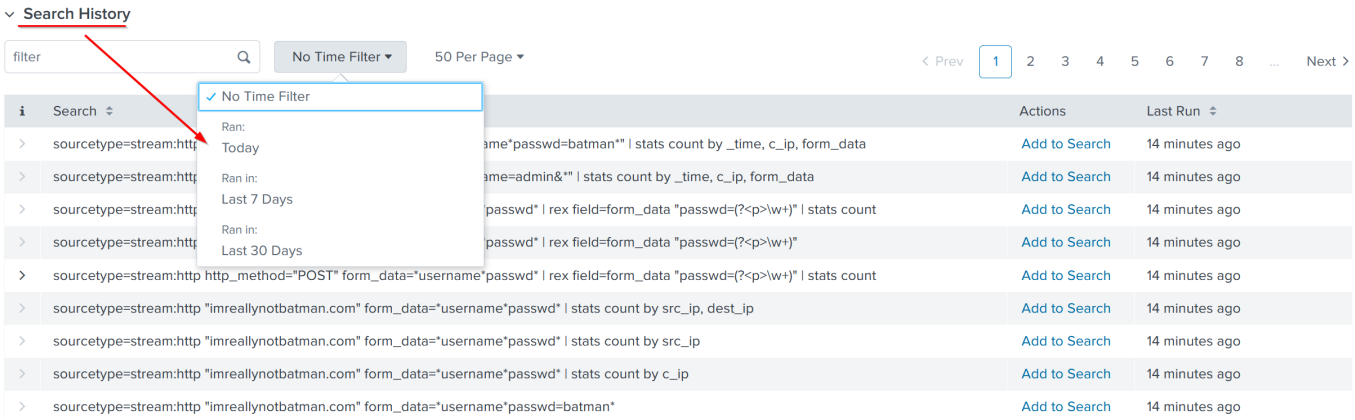

**2) Time Duration:**

This tab option provides multiple options to select the time duration for the search. **All-time** will display the events in real-time. Similarly, the **last 60 minutes** will display all the events captured in the last hour.



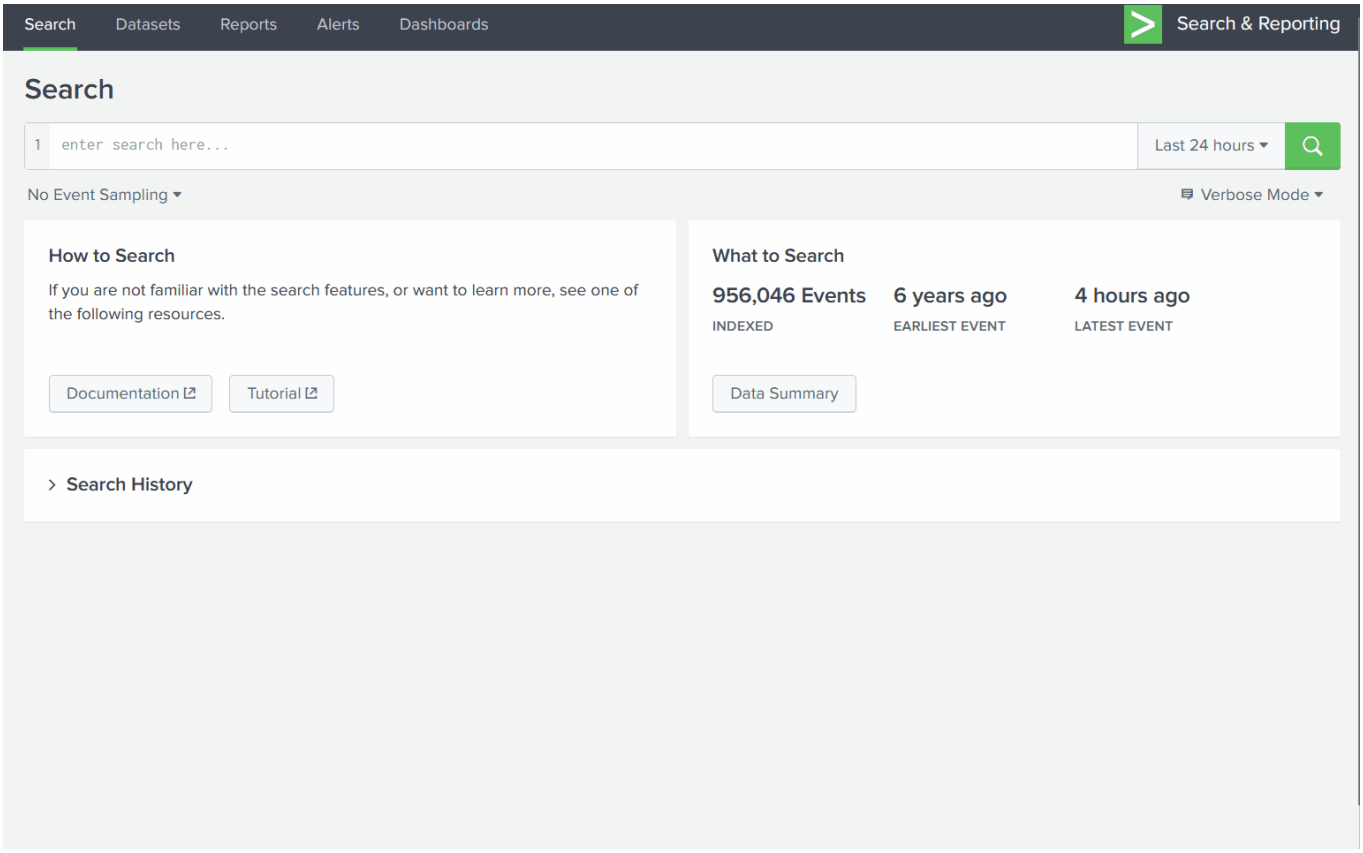

**3) Search History:**

This tab saves the search queries that the user has run in the past along with the time when it was run. It lets the user click on the past searches and look at the result. The filter option is used to search for the particular query based on the term.



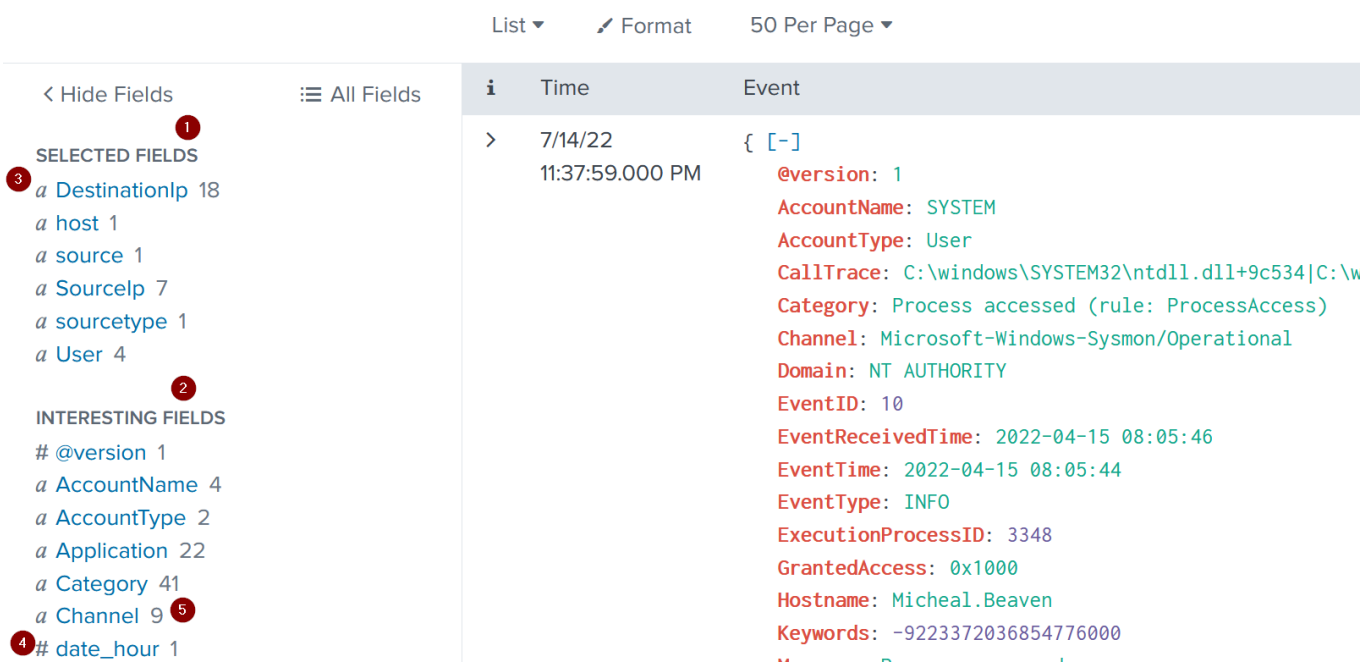

**4) Data Summary:**

This tab provides a summary of the data type, the data source, and the hosts that generated the events as shown below. This tab is very important feature used to get a brief idea about the network visibility.



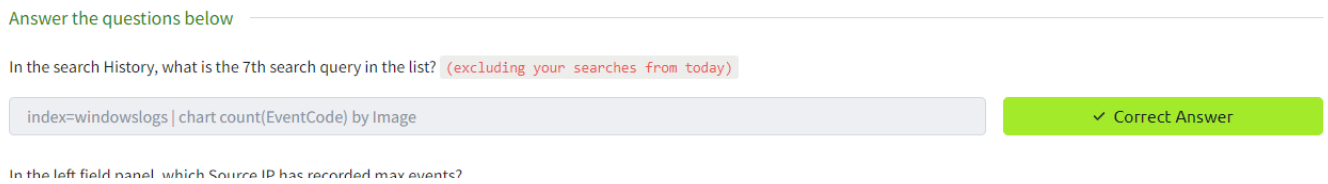

**5) Field Sidebar:**

The Field Sidebar can be found on the left panel of Splunk search. This sidebar has two sections showing selected fields and interesting fields. It also provides quick results, such as top values and raw values against each field.

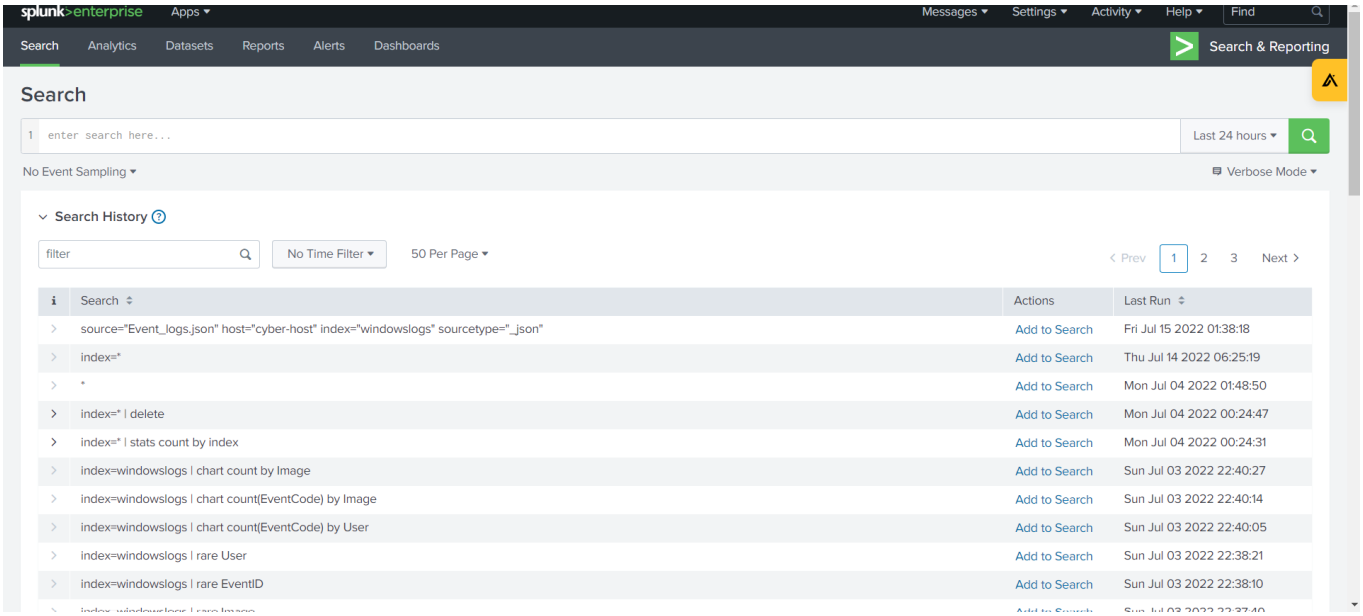



Some important points to understand about the sidebar are explained below:

| | |
|---|---|
| **1- Selected Fields** | Splunk extracts the default fields like source, sourcetype, and host, which appear in each event, and places them under the selected fields column. We can select other fields that seem essential and add them to the list. |
| **2- Interesting Fields** | Pulls all the interesting fields it finds and displays them in the left panel to further explore. |
| **3- Alpha-numeric fields 'α'** | This alpha symbol shows that the field contains text values. |
| **4- Numeric fields '#'** | This symbol shows that this field contains numerical values. |
| **5- Count** | The number against each field shows the number of events captured in that timeframe. |

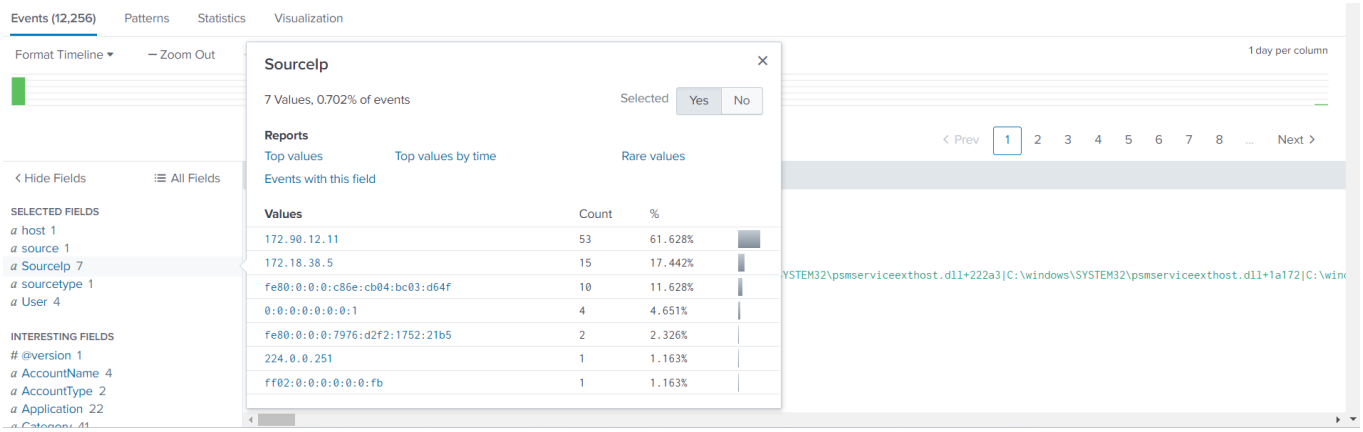



We can find the answer here:




In the left field panel, which Source IP has recorded max events?

172.90.12.11 ✓ Correct Answer Hint




How many events are returned when we apply the time filter to display events on 04/15/2022 and Time from 08:05 AM to 08:06 AM?

134 ✓ Correct Answer

Splunk Search Processing Language comprises of multiple functions, operators and commands that are used together to form a simple to complex search and get the desired results from the ingested logs. Main components of SPL are explained below:

### Search Field Operators

Splunk field operators are the building blocks used to construct any search query. These field operators are used to filter, remove, and narrow down the search result based on the given criteria. Common field operators are Comparison operators, wildcards, and boolean operators.

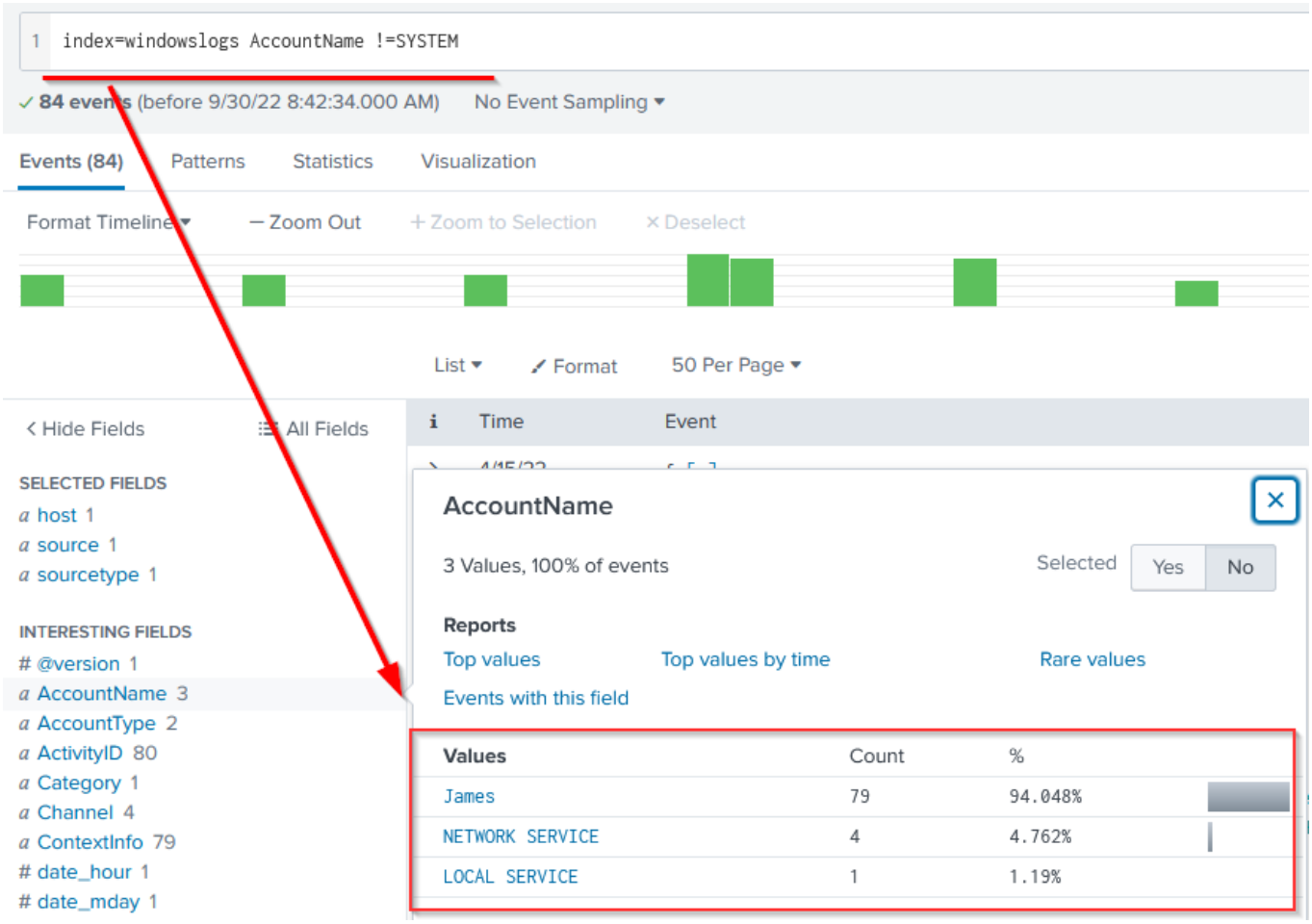
Comparison Operators

These operators are used to compare the values against the fields. Some common comparisons operators are mentioned below:

| Field Name | Operator | Example | Explanation |
|---|---|---|---|
| **Equal** | = | UserName=Mark | This operator is used to match values against the field. In this example, it will look for all the events, where the value of the field UserName is equal to Mark. |
| **Not Equal to** | != | UserName!=Mark | This operator returns all the events where the UserName value does not match Mark. |
| **Less than** | < | Age < 10 | Showing all the events with the value of Age less than 10. |
| **Less than or Equal to** | <= | Age <= 10 | Showing all the events with the value of Age less than or equal to 10. |
| **Greater than** | > | Outbound_traffic > 50 MB | This will return all the events where the Outbound traffic value is over 50 MB. |
| **Greater Than or Equal to** | >= | Outbound_traffic >= 50 MB | This will return all the events where the Outbound traffic value is greater or equal to 50 MB. |

Lets use the comparison operator to display all the event logs from the index "windowslogs", where AccountName is not Equal to "System"

**Search Query:** `index=windowslogs AccountName !=SYSTEM`



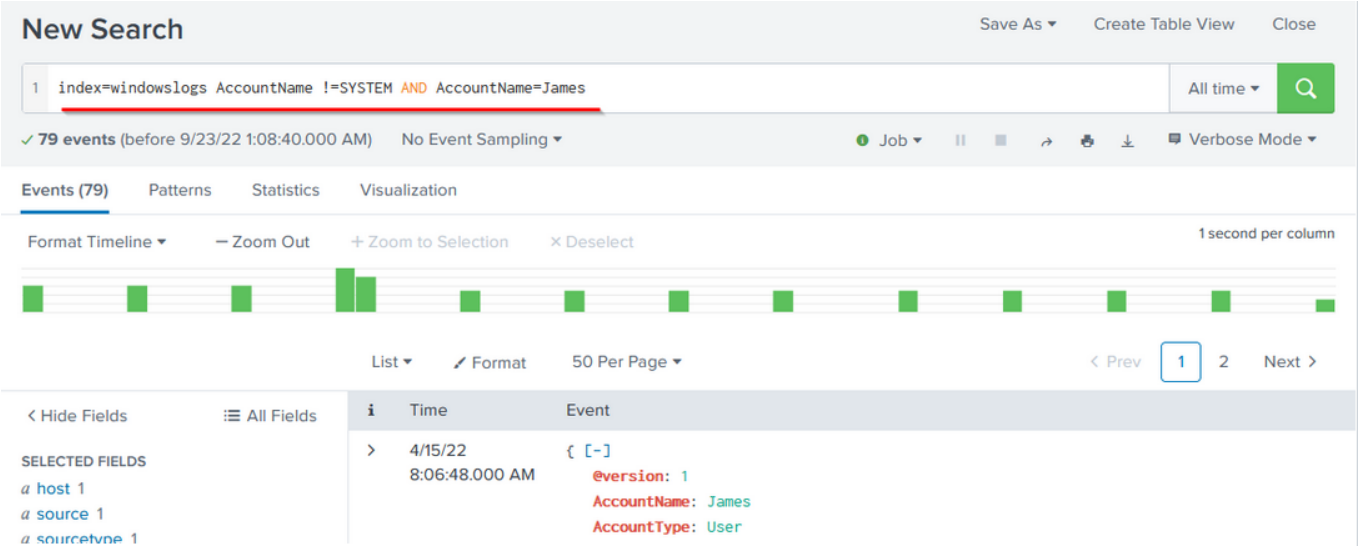

Boolean Operators

Splunk supports the following Boolean operators, which can be very handy in searching/filtering and narrowing down results.

| Operator | Syntax | Explanation |
|---|---|---|
| **NOT** | field_A **NOT** value | Ignore the events from the result where field_A contain the specified value. |
| **OR** | field_A=value1 **OR** field_A=value2 | Return all the events in which field_A contains either value1 or value2. |
| **AND** | field_A=value1 **AND** field_B=value2 | Return all the events in which field_A contains value1 and field_B contains value2. |

To understand how boolean operator works in SPL, lets add the condition to show the events from the James account.

**Search Query:** `index=windowslogs AccountName !=SYSTEM **AND** AccountName=James`



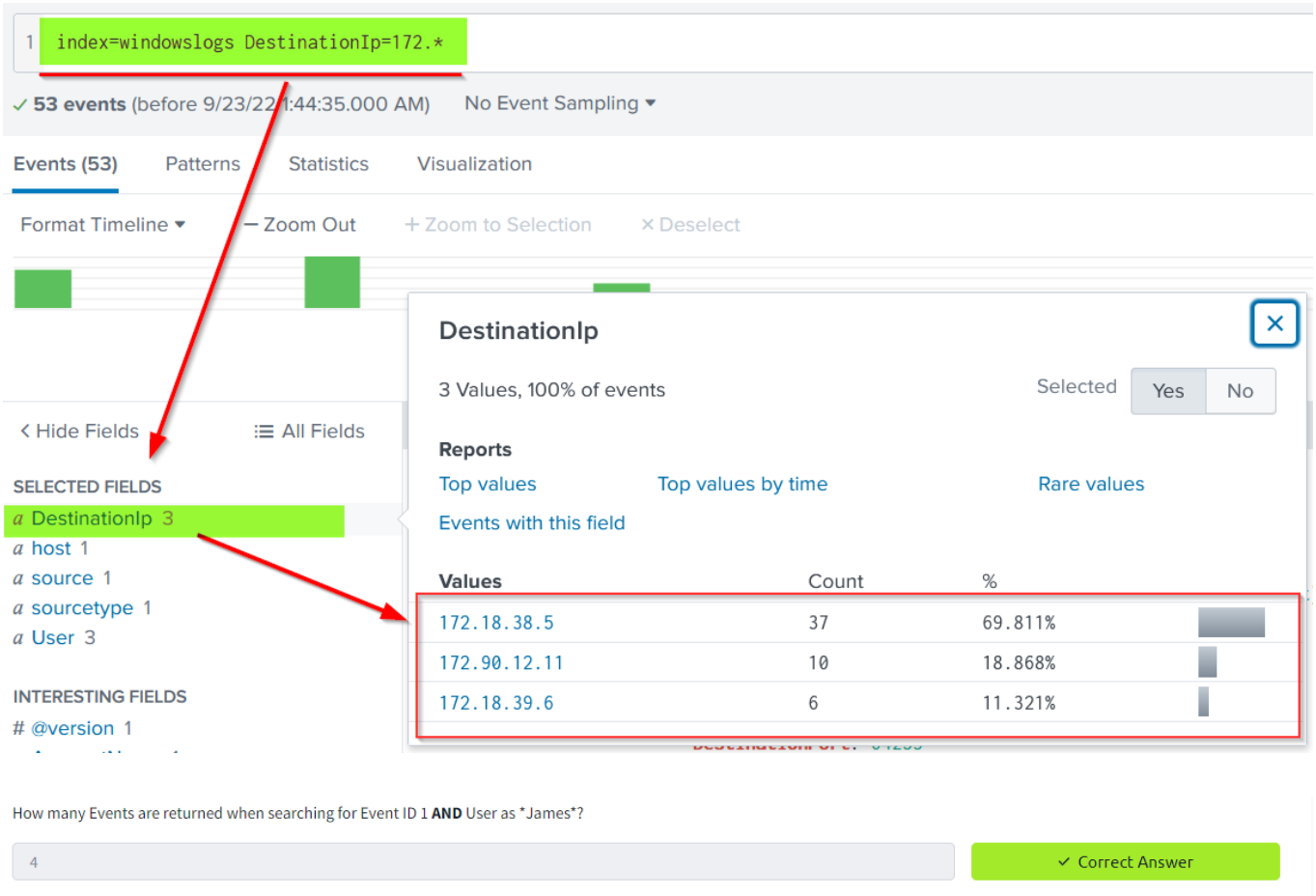

Wild Card

Splunk supports wildcards to match the characters in the strings.

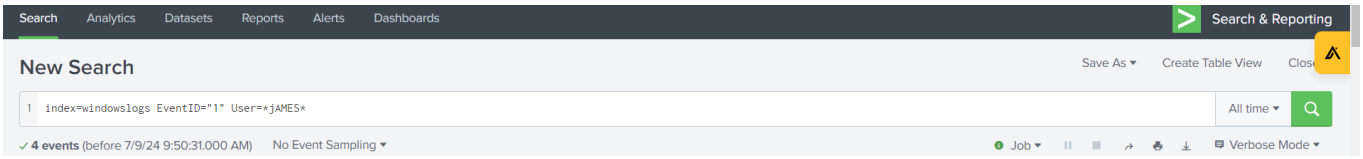
| Wildcard symbol | Example | Explanation |
|---|---|---|
| * | status=fail* | It will return all the results with values like<br><br>status=failed<br><br>status=failure |

In the events, there are multiple DestinationIPs reported. Let's use the wildcard only to show the **DestinationIP** starting from 172.*

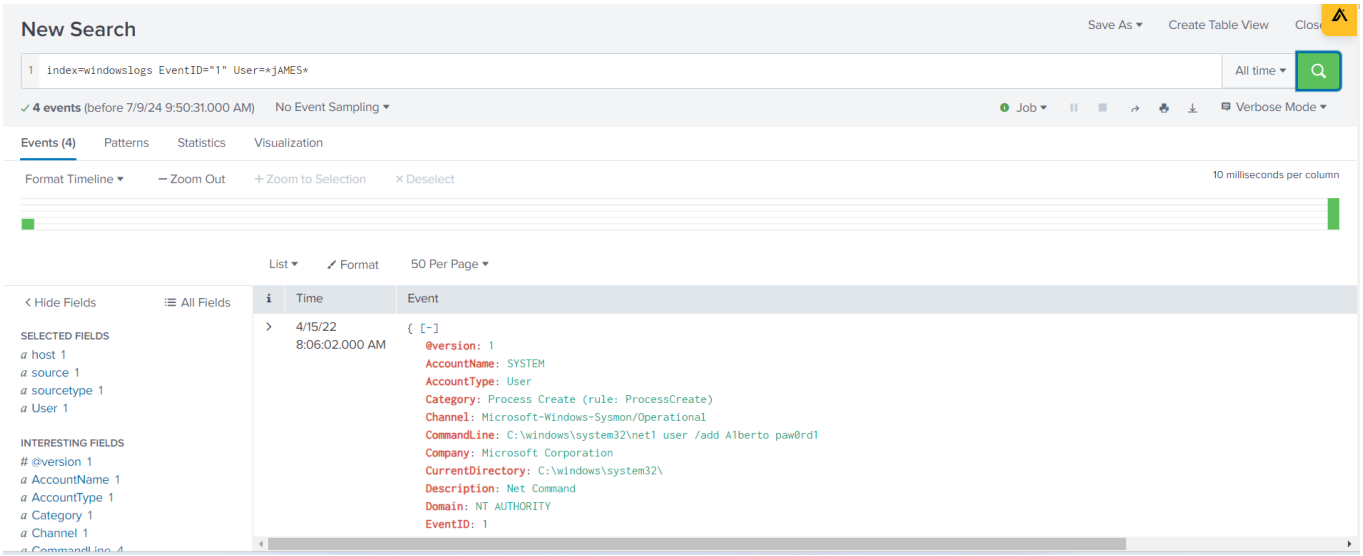
**Search Query:** `index=windowslogs DestinationIp=172.*`




So first in the search we type this



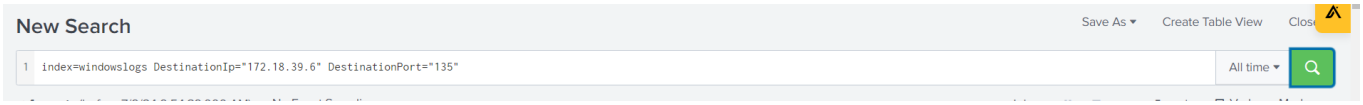

from this we get number of events like shown in the next photo:



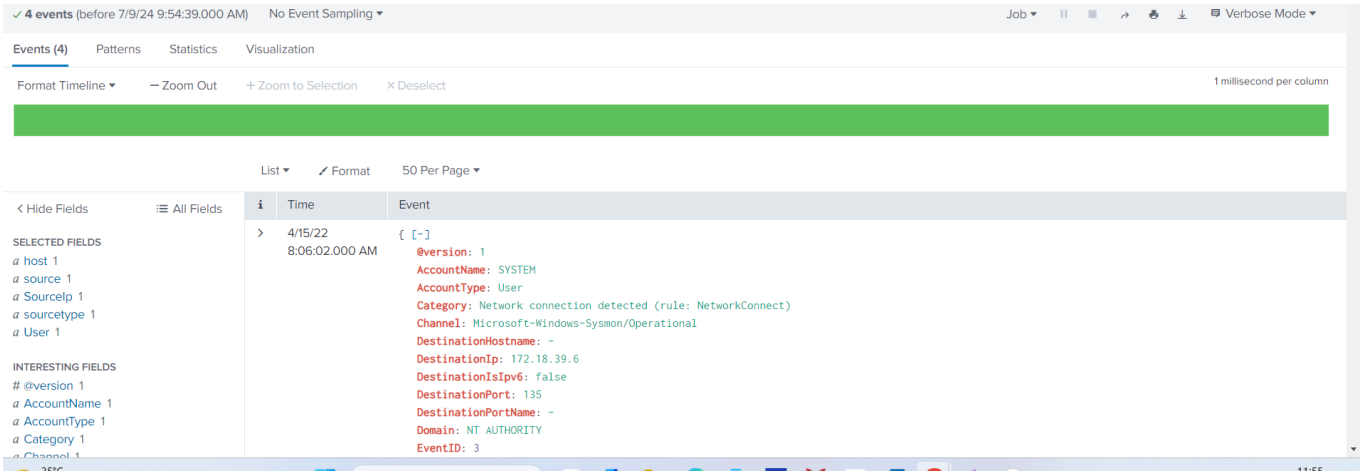
How many events are observed with Destination IP 172.18.39.6 AND destination Port 135?

4 ✓ Correct Answer Hint

For this answer in search we typed in this




and we got 4 events

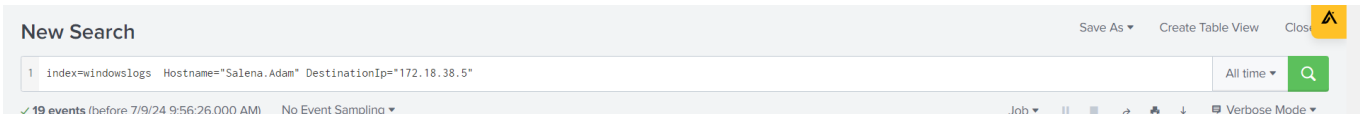



so that's why the answer is 4

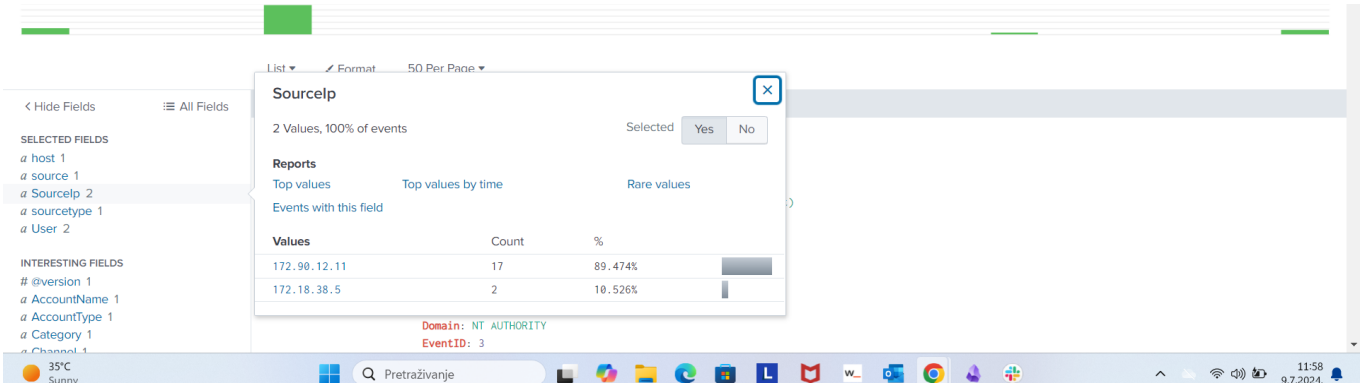
What is the Source IP with highest count returned with this Search query?
Search Query: index=windowslogs Hostname="Salena.Adam" DestinationIp="172.18.38.5"

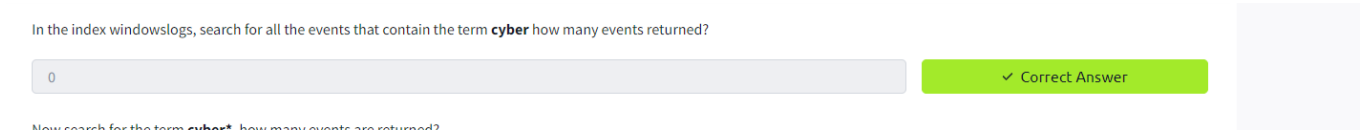
172.90.12.11 ✓ Correct Answer

For this question in the search bar we had to type in this:

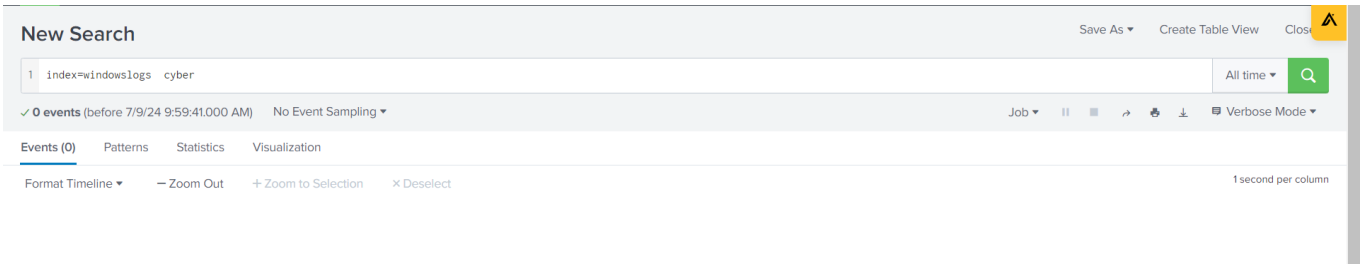



Then here we can see answer to our question

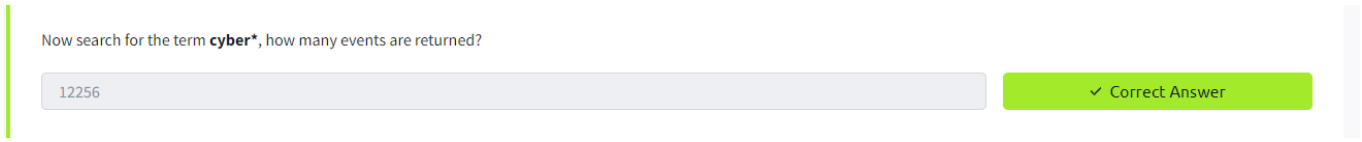




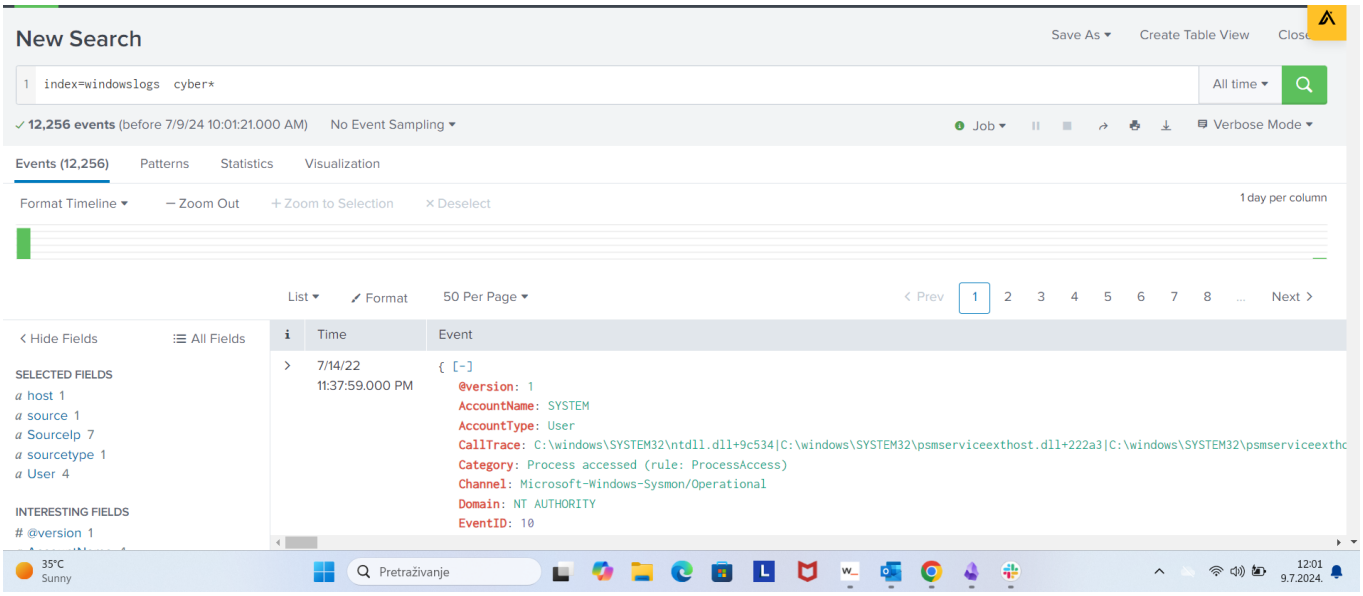

In the search bar we type in this




and we can see that the output is 0




In the search bar we type this




and we can see there are 12256 generated events so that is the answer to our question.

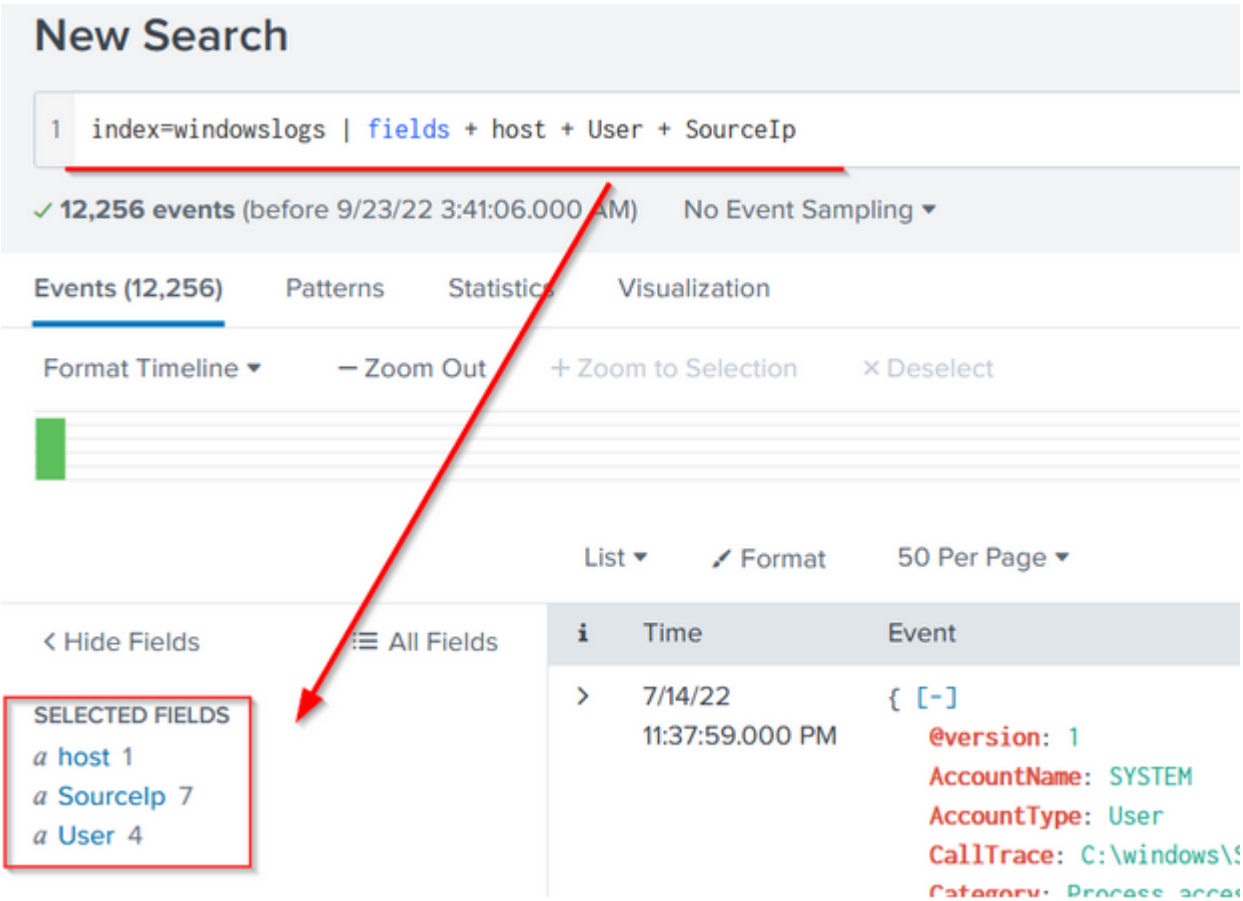
Our network generates thousands of logs each minute, all ingesting into our SIEM solution. It becomes a daunting task to search for any anomaly without using filters. SPL allows us to use **Filters** to narrow down the result and only show the important events that we are interested in. We can add or remove certain data from the result using filters. The following commands are useful in applying filters to the search results.

**Fields**

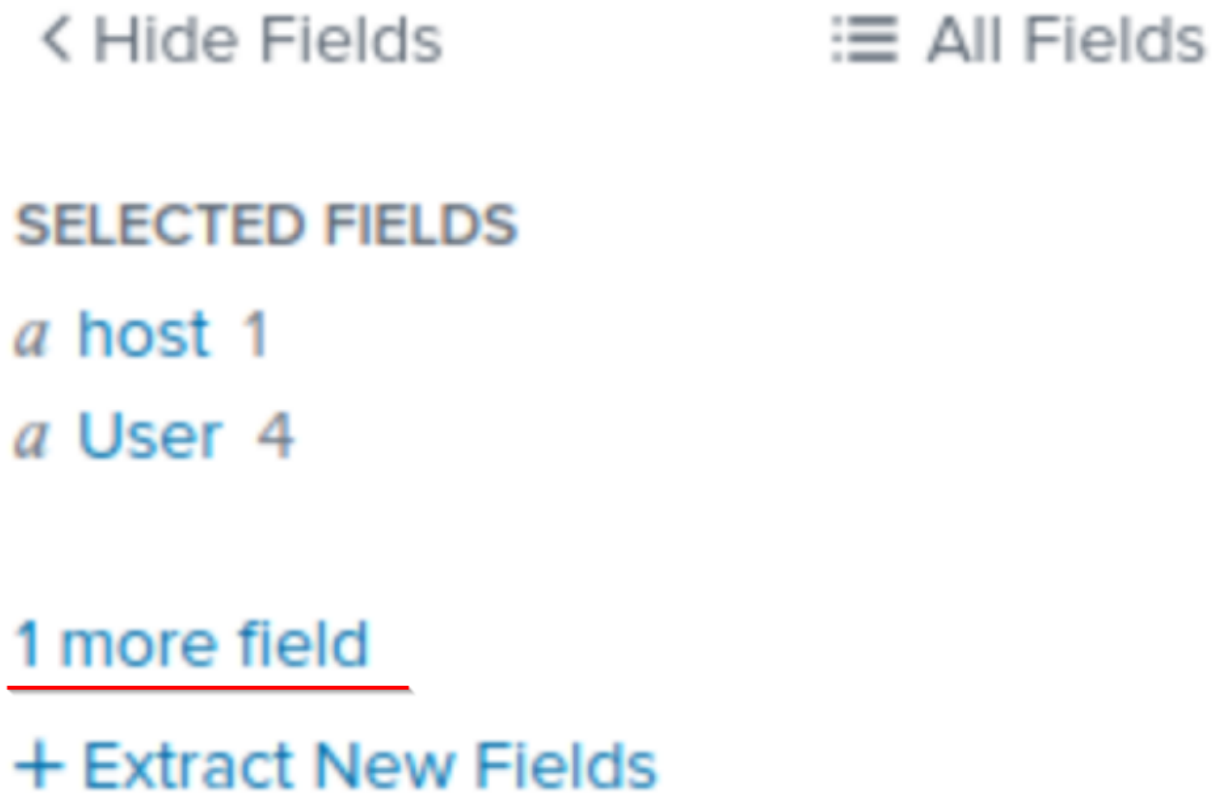
| | |
|---|---|
| **Command** | **fields** |
| **Explanation** | Fields command is used to add or remove mentioned fields from the search results. To remove the field, minus sign ( - ) is used before the fieldname and plus ( + ) is used before the fields which we want to display. |
| **Syntax** | \| fields <field_name1>  <field_name2> |
| **Example** | \| `fields + HostName - EventID` |

Let's use the fields command to only display host, User, and SourceIP fields using the following syntax.

**Search Query:** `index=windowslogs | fields + host + User + SourceIp`




**Note:** Click on the **More field** to display the fields if some fields are not visible.

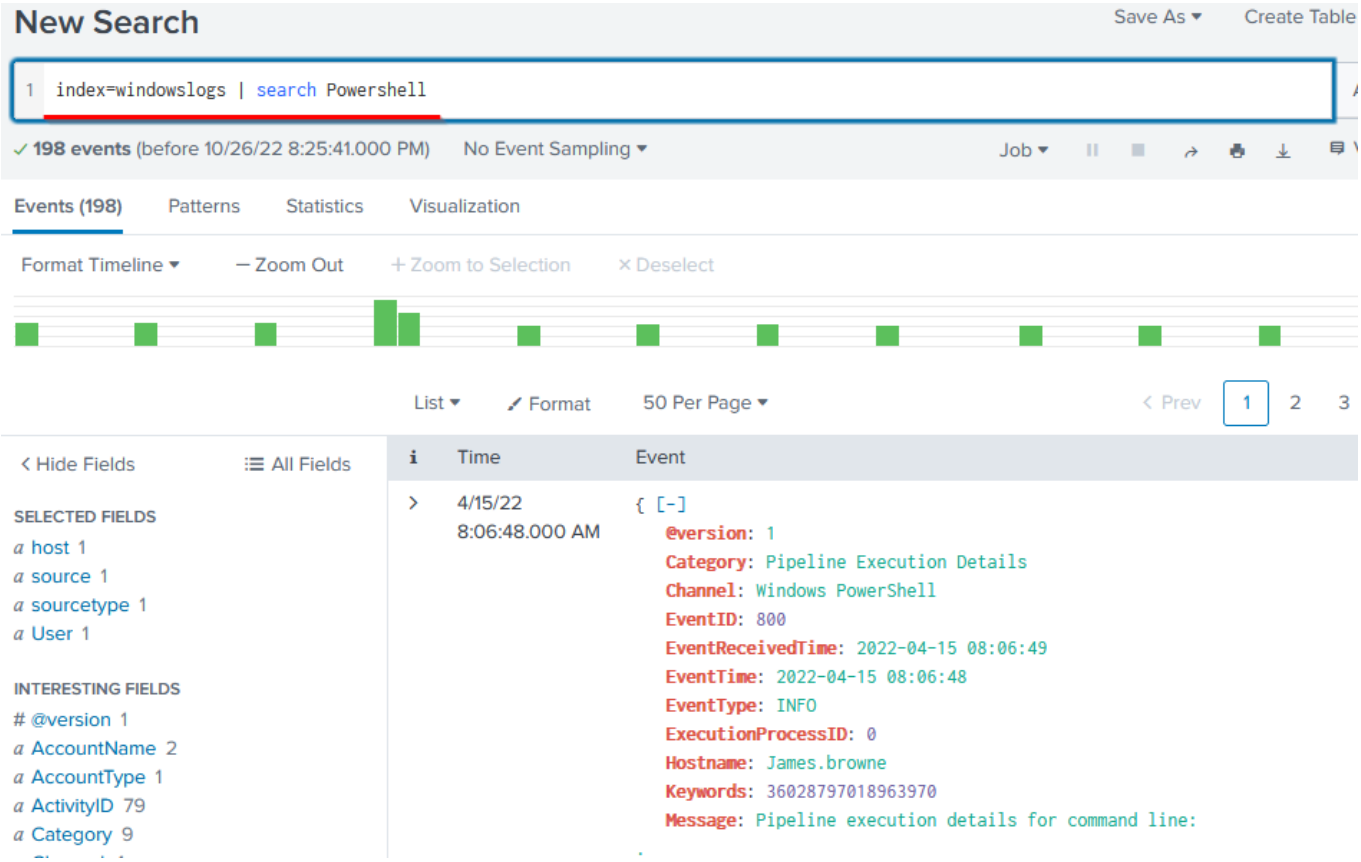



**Search**

| | |
|---|---|
| **Command** | **search** |
| **Explanation** | This command is used to search for the raw text while using the chaining command `\|` |
| **Syntax** | \| search <search_keyword> |
| **Example** | \| `search "Powershell"` |

Use the search command to show all the events containing the term Powershell. This will return all the events that contain the term "**Powershell**".

**Search Query:** `index=windowslogs | search Powershell`

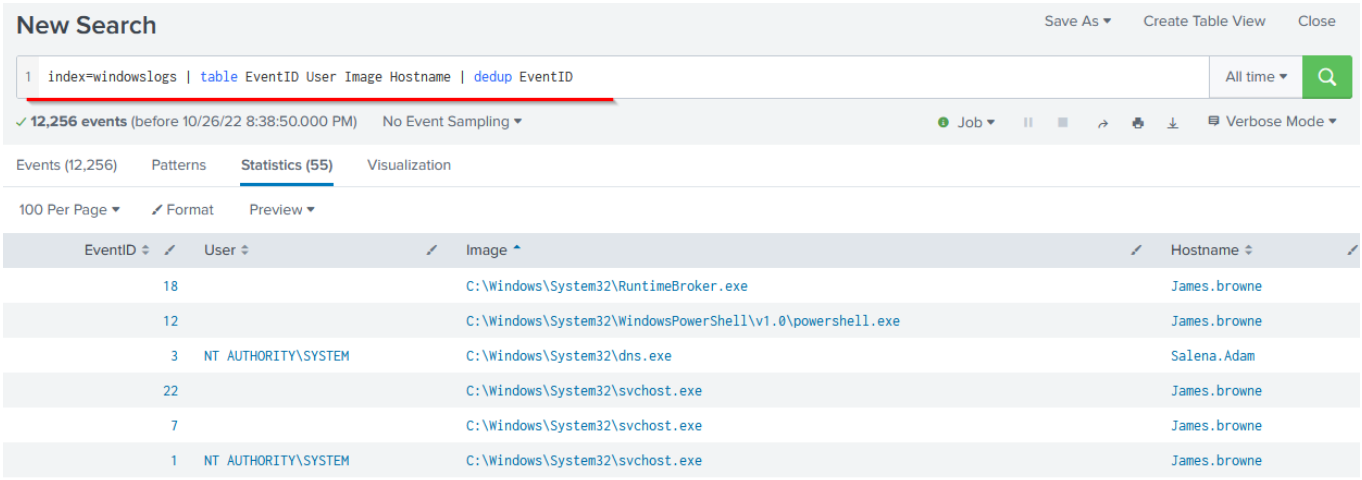



### Dedup

| | |
|---|---|
| **Command** | **dedup** |
| **Explanation** | Dedup is the command used to remove duplicate fields from the search results. We often get the results with various fields getting the same results. These commands remove the duplicates to show the unique values. |
| **Syntax** | \| dedup |
| **Example** | \| `dedup EventID` |

We can use the dedup command to show the list of unique **EventIDs** from a particular hostname.

**Search Query:** `index=windowslogs | table EventID User Image Hostname | dedup EventID`

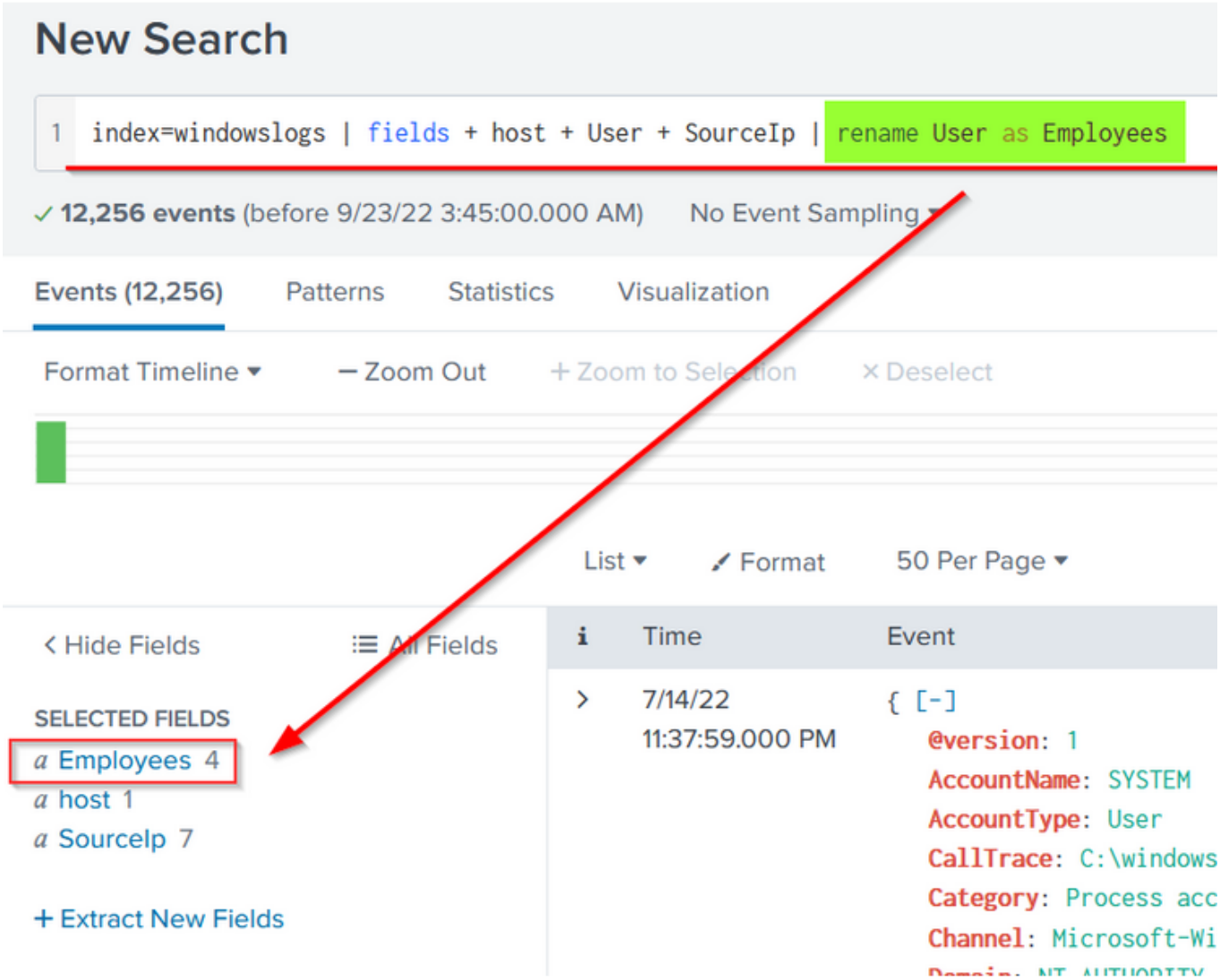



### Rename

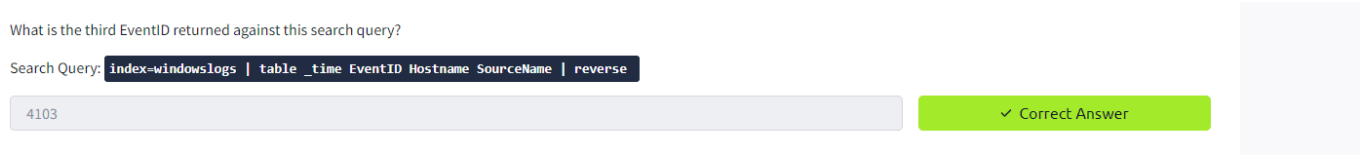
| | |
|---|---|
| **Command** | **rename** |
| **Explanation** | It allows us to change the name of the field in the search results. It is useful in a scenario when the field name is generic or log, or it needs to be updated in the output. |
| **Syntax** | \| rename |
| **Example** | \| `rename User as Employees` |

Let's rename the User field to Employees using the following search query.

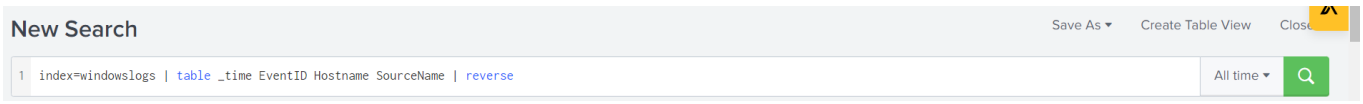
**Search Query**: `index=windowslogs | fields + host + User + SourceIp | rename User as Employees`



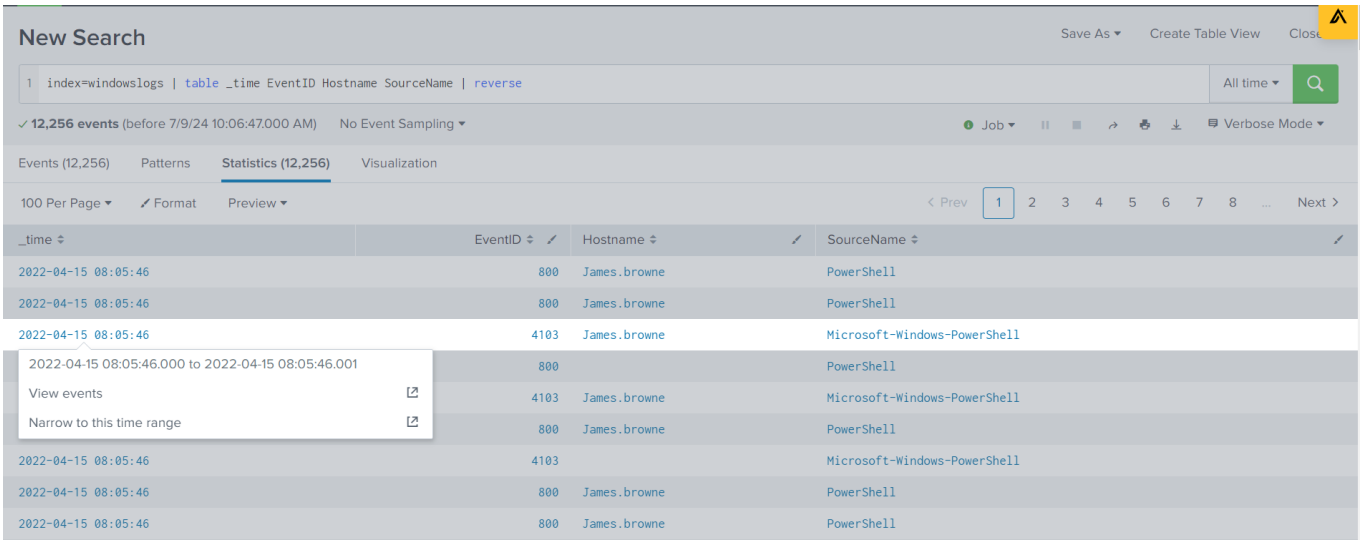

Answer the questions below

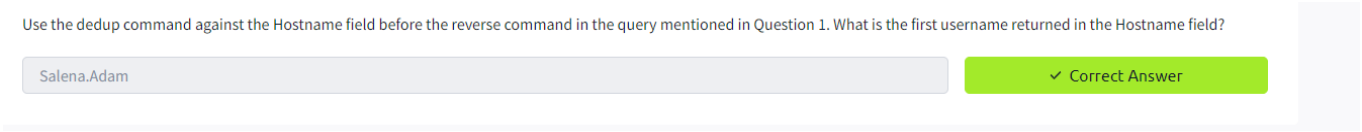



We first type in this to the search query

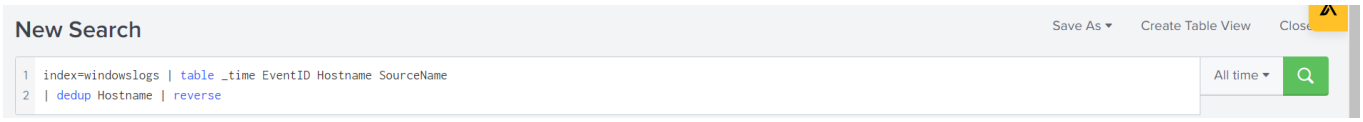



and we can see answer from the output

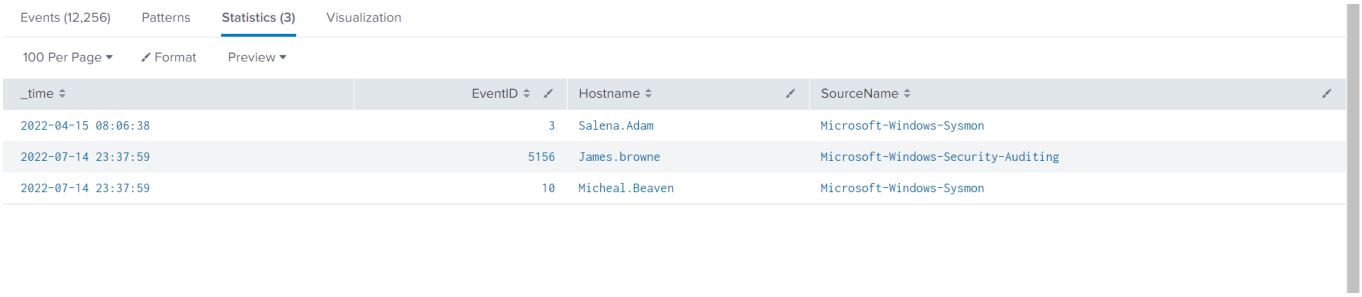



We type in this to the search bar




and we get the output like this




So first generated username is Selena.Adam

SPL provides various commands to bring structure or order to the search results. These sorting commands like `head`, `tail`, and `sort` can be very useful during logs investigation. These ordering commands are explained below:

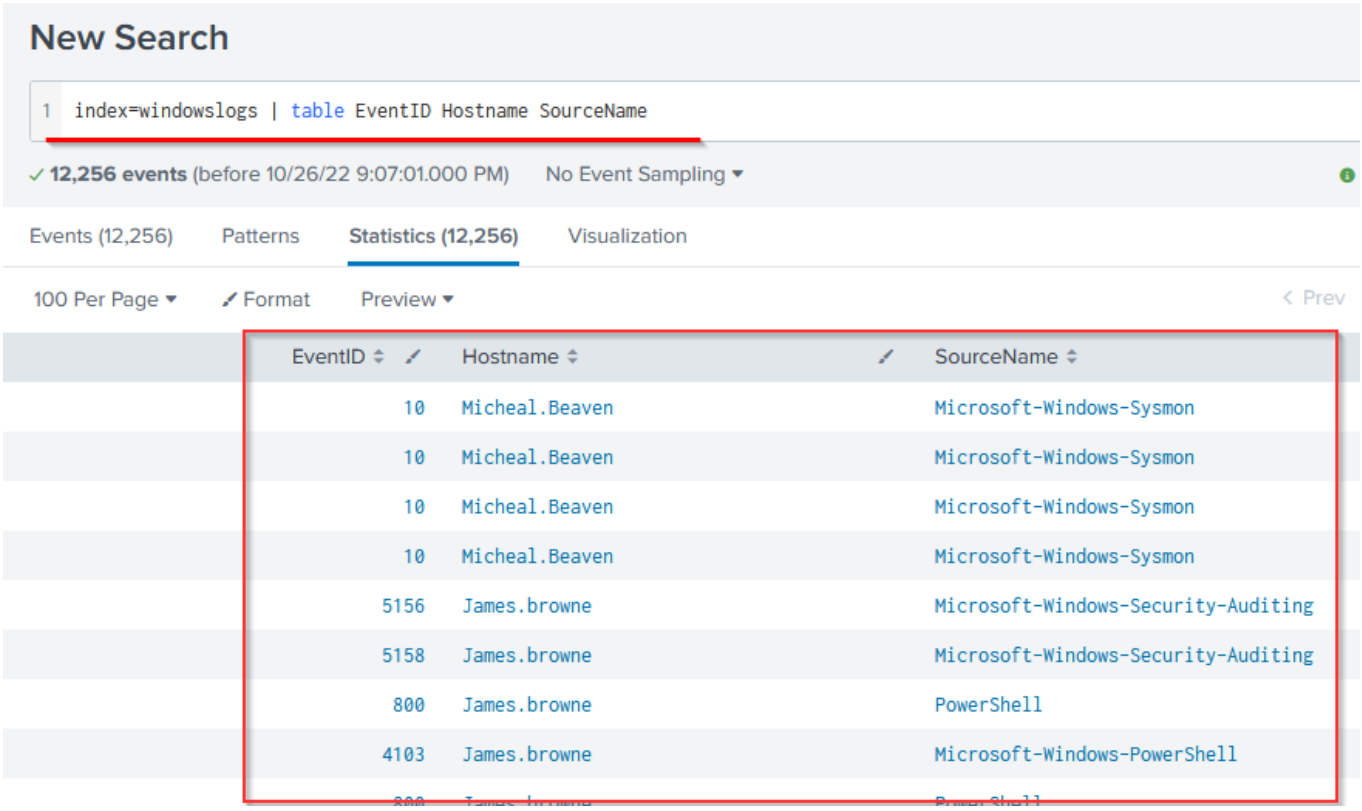
Table

| | |
|---|---|
| **Explanation** | Each event has multiple fields, and not every field is important to display. The Table command allows us to create a table with selective fields as columns. |
| **Syntax** | \| table <field_name1> <fieldname_2> |

| | |
|---|---|
| **Example** | \| `table`<br><br>\| `head 20` # will return the top 20 events from the result list. |

This search query will create a table with three columns selected and ignore all the remaining columns from the display.

**Search Query:** `index=windowslogs | table EventID Hostname SourceName`

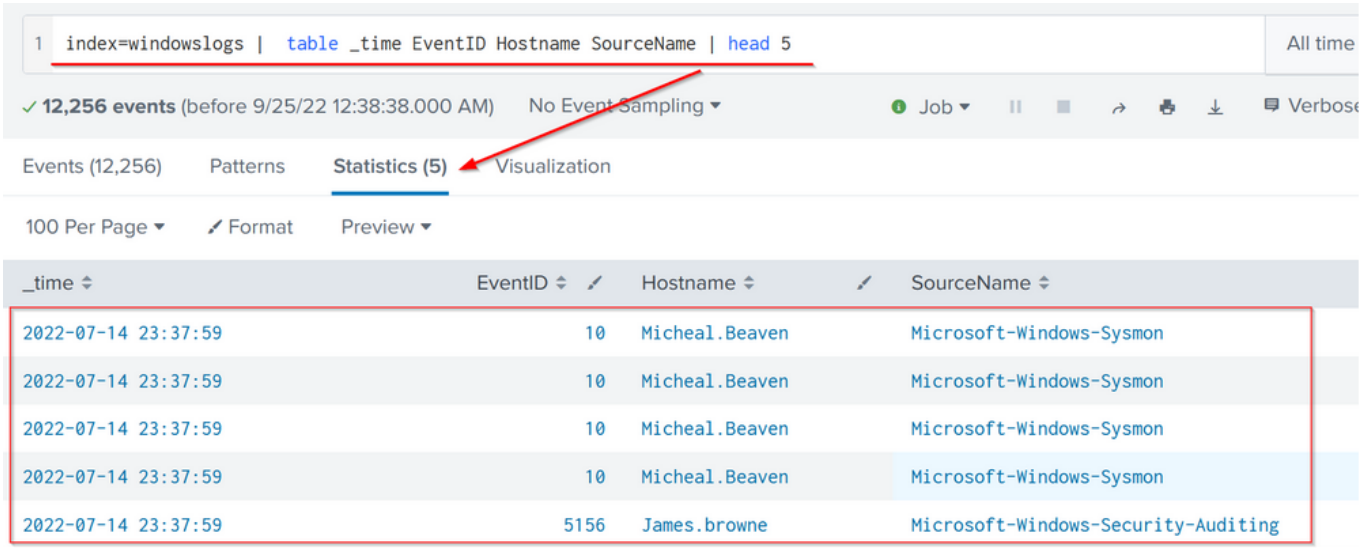



Head

| | |
|---|---|
| **Explanation** | The **head** command returns the first 10 events if no number is specified. |
| **Syntax** | \| head |
| **Example** | \| `head` # will return the top 10 events from the result list<br><br>\| `head 20` # will return the top 20 events from the result list |

The following search query will show the table containing the mentioned fields and display only the top 5 entries.

**Search Query:** `index=windowslogs | table _time EventID Hostname SourceName | **head 5**`

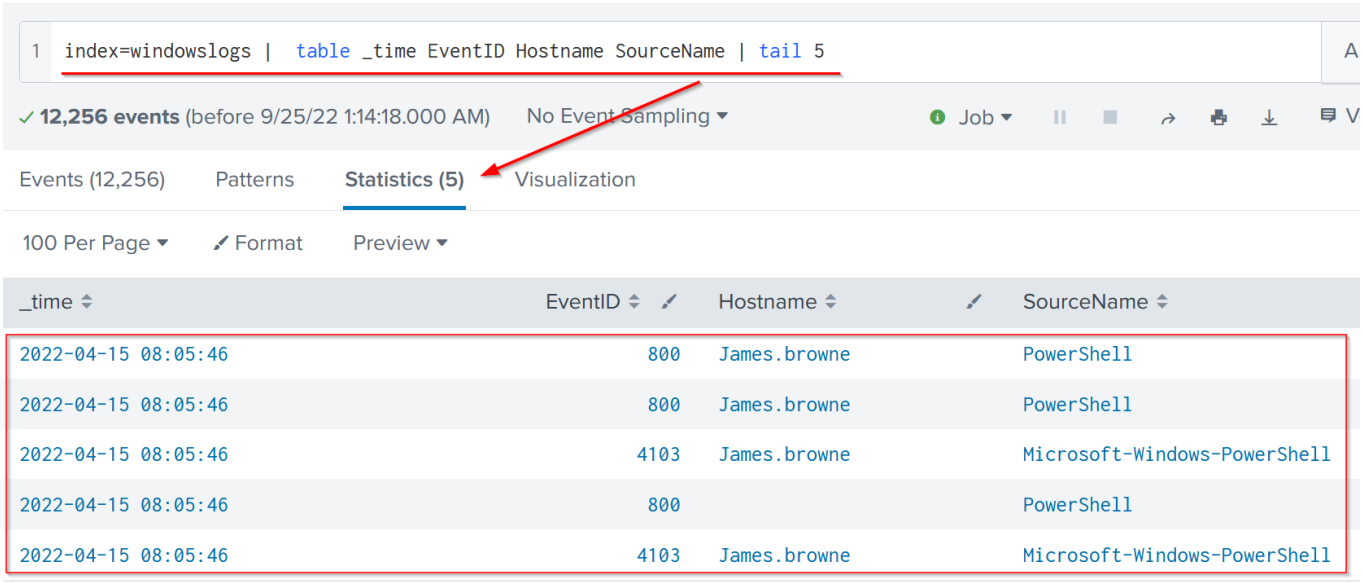



Tail

| | |
|---|---|
| **Explanation** | The **Tail** command returns the last 10 events if no number is specified. |
| **Syntax** | \| tail |
| **Example** | \| `tail` # will return the last 10 events from the result list<br><br>\| `tail 20` # will return the last 20 events from the result list |

The following search query will show the table containing the mentioned fields and display only 5 entries from the bottom of the list.

**Search Query:** `index=windowslogs | table _time EventID Hostname SourceName | tail 5`

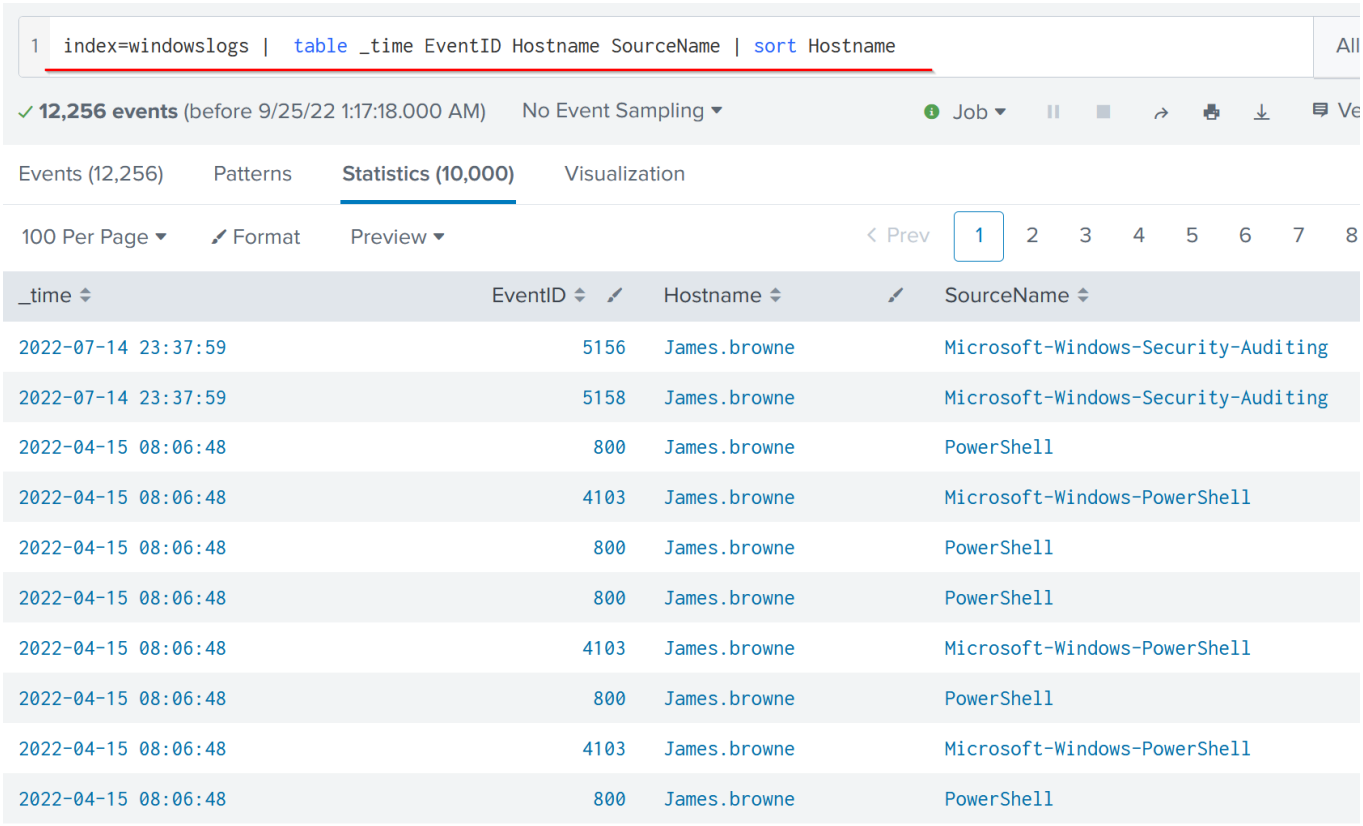



sort

| | |
|---|---|
| **Explanation** | The **Sort** command allows us to order the fields in ascending or descending order. |
| **Syntax** | \| `sort` <field_name> |
| **Example** | \| `sort Hostname` # This will sort the result in Ascending order. |

The following search query will sort the results based on the Hostname field.

**Search Query:** `index=windowslogs | table _time EventID Hostname SourceName | sort Hostname`

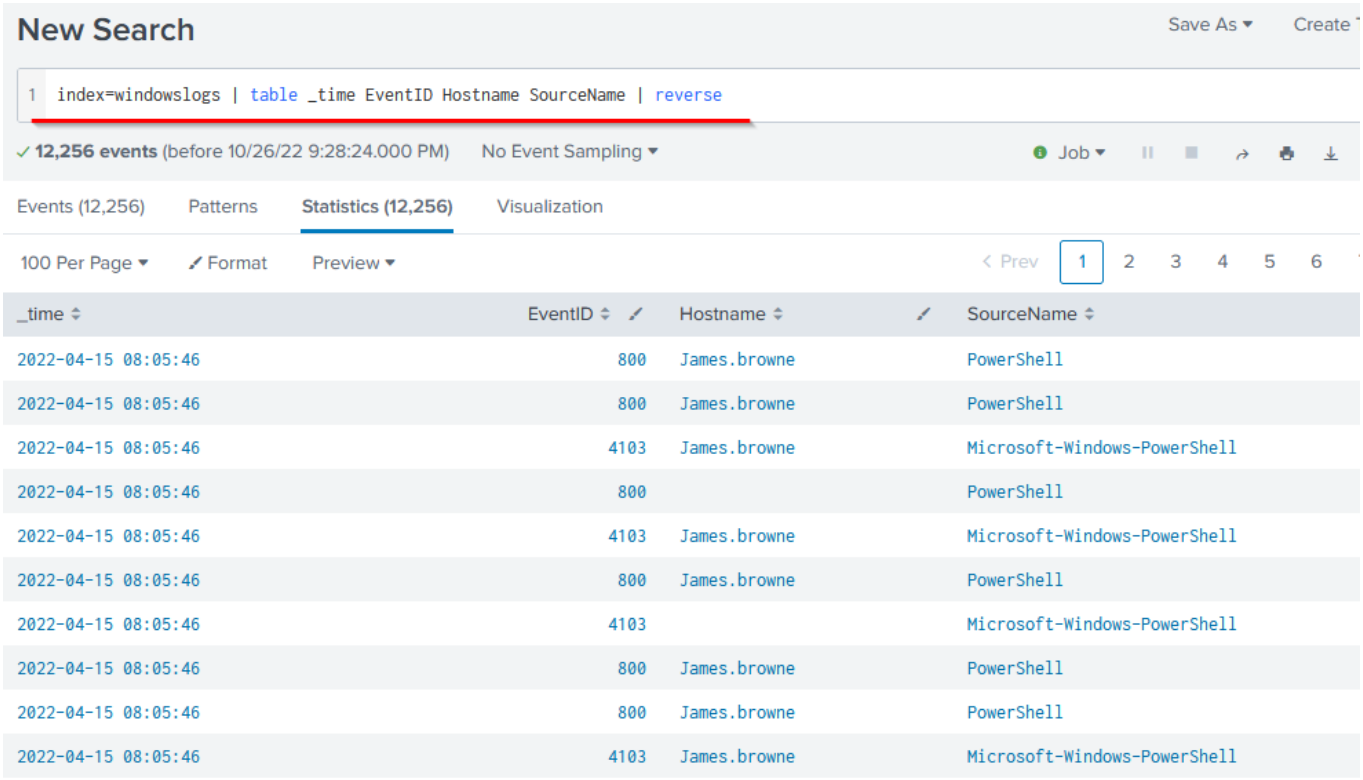


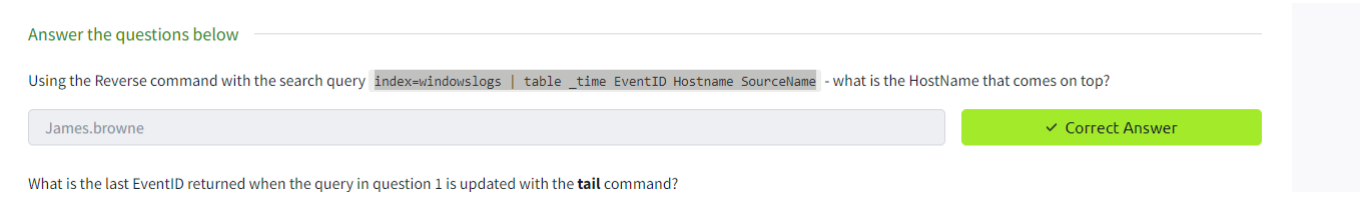

| _time | EventID | Hostname | SourceName |
|---|---|---|---|
| 2022-07-14 23:37:59 | 5156 | James.browne | Microsoft-Windows-Security-Auditing |
| 2022-07-14 23:37:59 | 5158 | James.browne | Microsoft-Windows-Security-Auditing |
| 2022-04-15 08:06:48 | 800 | James.browne | PowerShell |
| 2022-04-15 08:06:48 | 4103 | James.browne | Microsoft-Windows-PowerShell |
| 2022-04-15 08:06:48 | 800 | James.browne | PowerShell |
| 2022-04-15 08:06:48 | 800 | James.browne | PowerShell |
| 2022-04-15 08:06:48 | 4103 | James.browne | Microsoft-Windows-PowerShell |
| 2022-04-15 08:06:48 | 800 | James.browne | PowerShell |
| 2022-04-15 08:06:48 | 4103 | James.browne | Microsoft-Windows-PowerShell |
| 2022-04-15 08:06:48 | 800 | James.browne | PowerShell |

Reverse

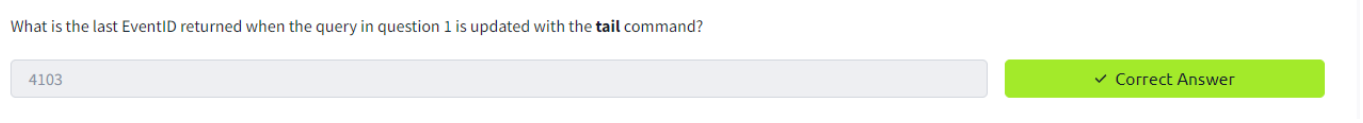
| | |
|---|---|
| **Explanation** | The reverse command simply reverses the order of the events. |
| **Syntax** | \| reverse |
| **Example** | `<Search Query> \| reverse` |

**Search Query:** `index=windowslogs | table _time EventID Hostname SourceName | reverse`

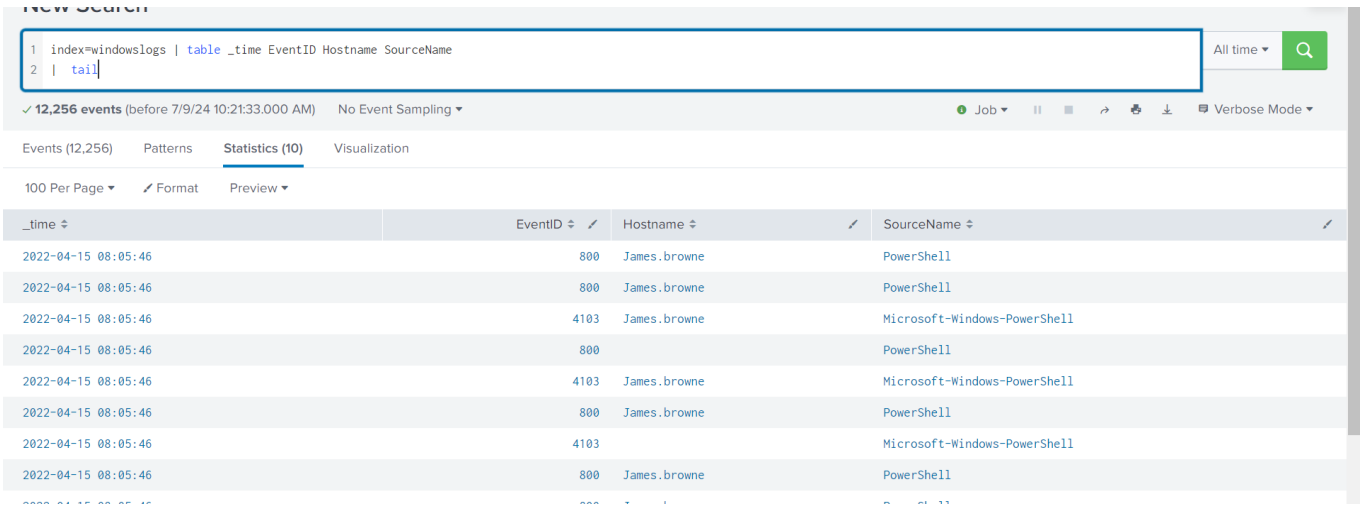


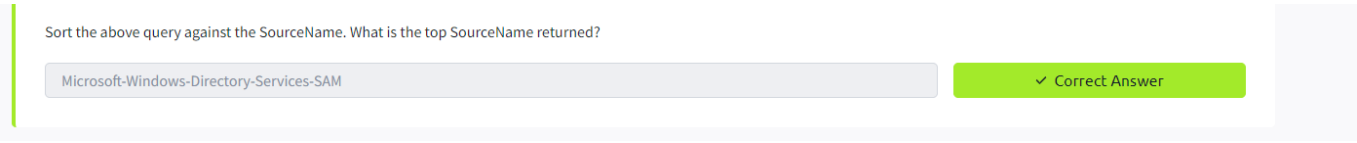




We can see the answer is James.browne

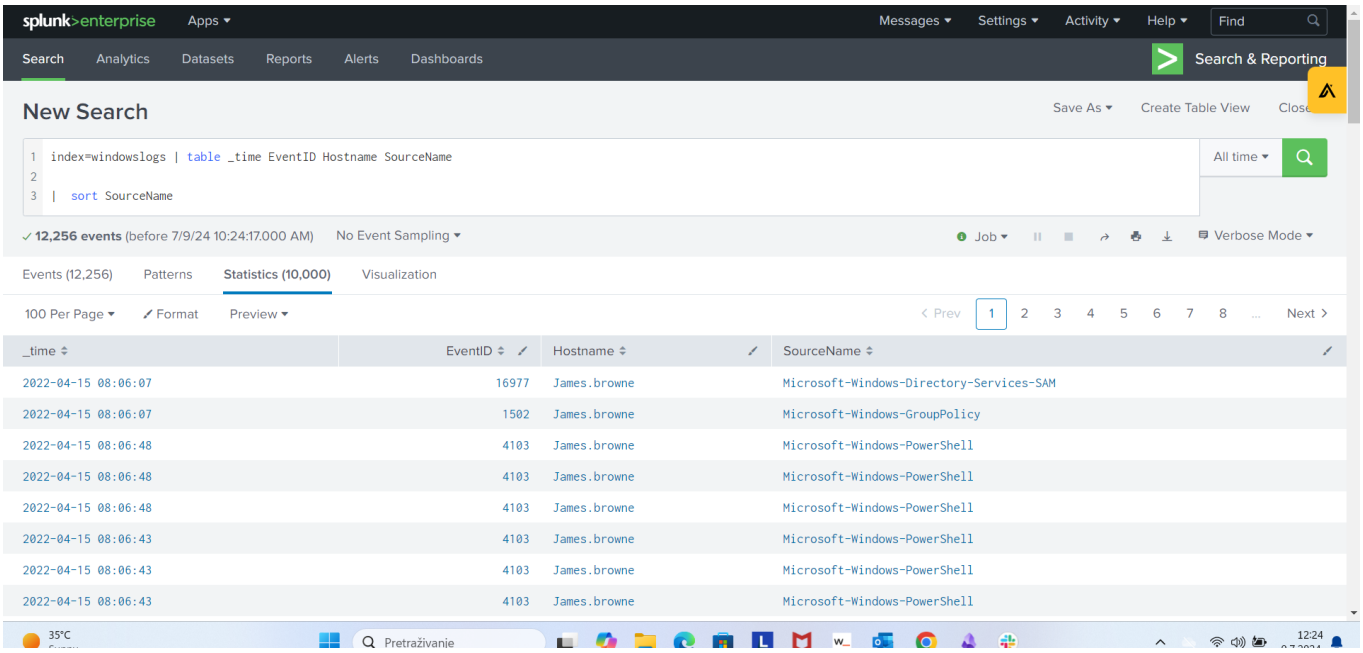



When we type this into the search bar we can see what was the last EventID






We can see the answer is this




Transformational commands are those commands that change the result into a data structure from the field-value pairs. These commands simply transform specific values for each event into numerical values which can easily be utilized for statistical purposes or turn the results into visualizations. Searches that use these transforming commands are called transforming searches. Some of the most used transforming commands are explained below.

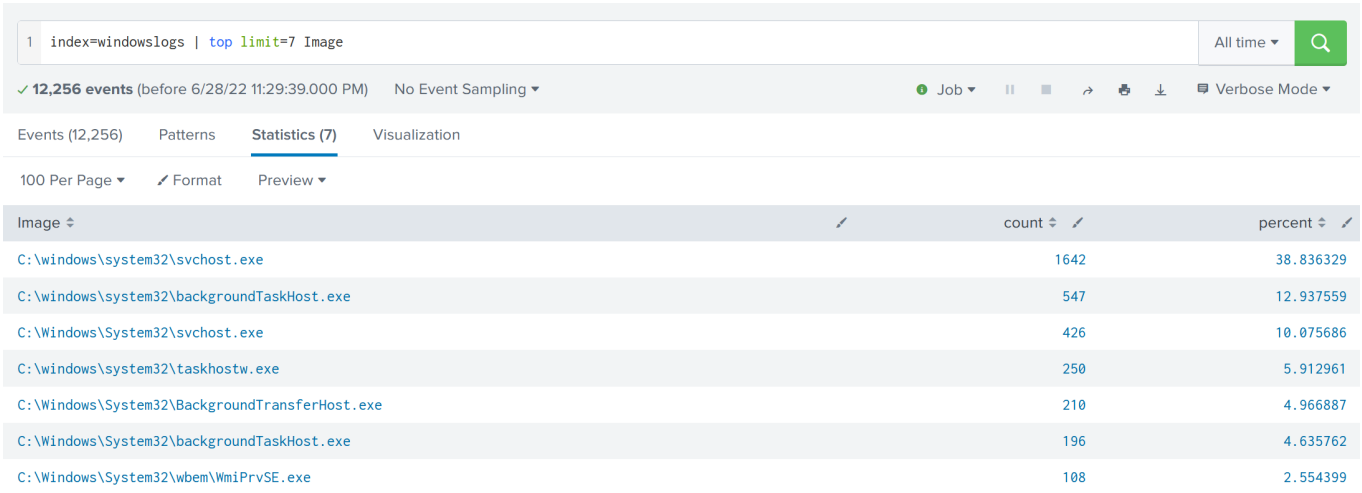
General Transformational Commands

Top

| **Command** | **top** |
| --- | --- |
| **Explanation** | This command returns frequent values for the top 10 events. |
| **Syntax** | \| top  <field_name><br><br>\| top limit=6 <field_name> |
| **Example** | `top limit=3 EventID` |

The following command will display the top 7 Image ( representing Processes) captured.

**Search Query:** `index=windowslogs | top limit=7 Image`



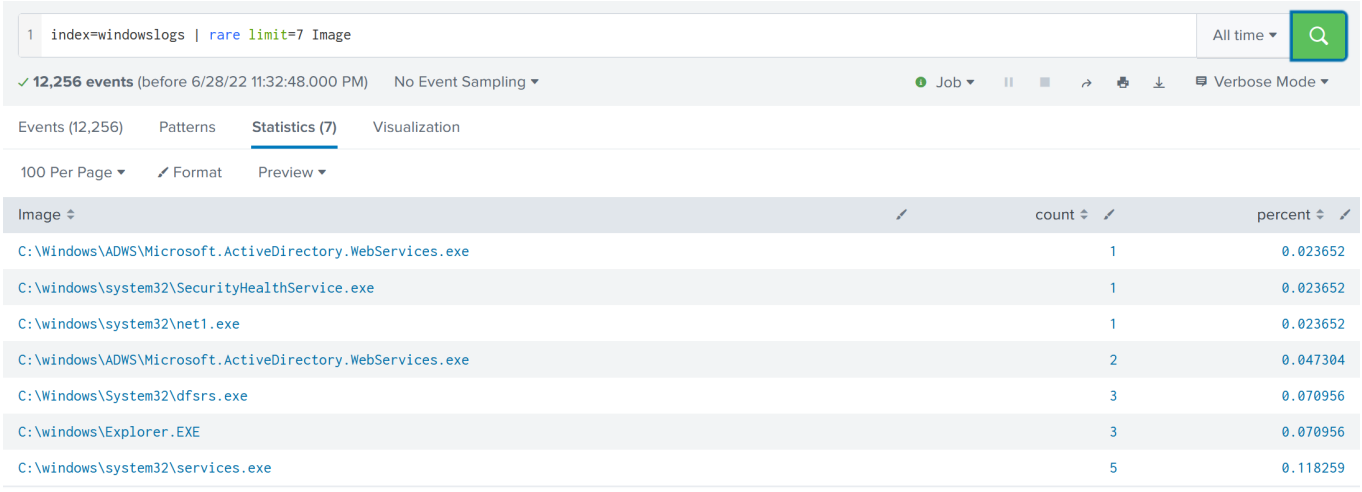

| Image | count | percent |
|---|---|---|
| C:\windows\system32\svchost.exe | 1642 | 38.836329 |
| C:\windows\system32\backgroundTaskHost.exe | 547 | 12.937559 |
| C:\Windows\System32\svchost.exe | 426 | 10.075686 |
| C:\windows\system32\taskhostw.exe | 250 | 5.912961 |
| C:\Windows\System32\BackgroundTransferHost.exe | 210 | 4.966887 |
| C:\Windows\System32\backgroundTaskHost.exe | 196 | 4.635762 |
| C:\Windows\System32\wbem\WmiPrvSE.exe | 108 | 2.554399 |

## Rare

| | |
|---|---|
| **Command** | **rare** |
| **Explanation** | This command does the opposite of top command as it returns the least frequent values or bottom 10 results. |
| **Syntax** | \| rare <field_name><br><br>\| rare limit=6 <field_name> |
| **Example** | `rare limit=3 EventID` |

The following command will display the rare 7 Image (Processes) captured.

**Search Query:** `index=windowslogs | rare limit=7 Image`




| Image | count | percent |
|---|---|---|
| C:\Windows\ADWS\Microsoft.ActiveDirectory.WebServices.exe | 1 | 0.023652 |
| C:\windows\system32\SecurityHealthService.exe | 1 | 0.023652 |
| C:\windows\system32\net1.exe | 1 | 0.023652 |
| C:\windows\ADWS\Microsoft.ActiveDirectory.WebServices.exe | 2 | 0.047304 |
| C:\Windows\System32\dfsrs.exe | 3 | 0.070956 |
| C:\windows\Explorer.EXE | 3 | 0.070956 |
| C:\windows\system32\services.exe | 5 | 0.118259 |

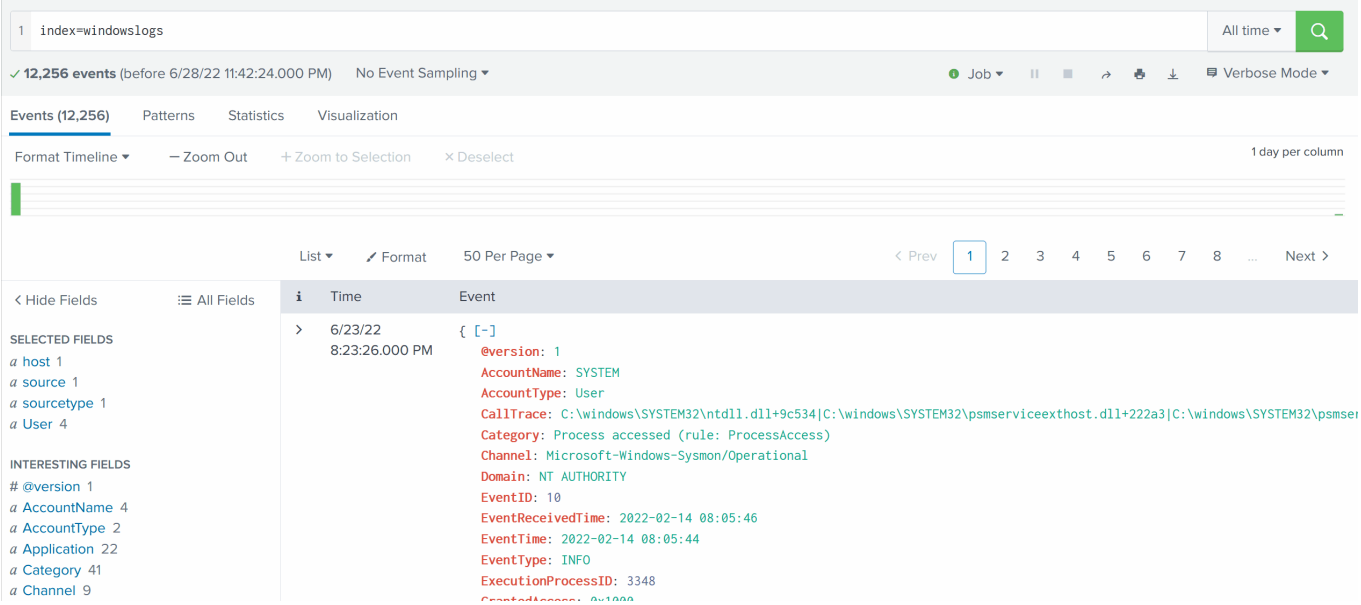
## Highlight

| | |
|---|---|
| **Command** | **highlight** |

| | |
|---|---|
| **Explanation** | The highlight command shows the results in raw events mode with fields highlighted. |
| **Syntax** | highlight <field_name1> <field_name2> |
| **Example** | `highlight User, host, EventID, Image` |

The following command will highlight the three mentioned fields in the raw logs

**Search Query:** `index=windowslogs | highlight User, host, EventID, Image`




STATS Commands

SPL supports various stats commands that help in calculating statistics on the values. Some common stat commands are:

| Command | Explanation | Syntax | Example |
|---|---|---|---|
| **Average** | This command is used to calculate the average of the given field. | stats avg(field_name) | stats avg(product_price) |
| **Max** | It will return the maximum value from the specific field. | stats max(field_name) | stats max(user_age) |
| **Min** | It will return the minimum value from the specific field. | stats min(field_name) | stats min(product_price) |
| **Sum** | It will return the sum of the fields in a specific value. | stats sum(field_name) | stats sum(product_cost) |
| **Count** | The count command returns the number of data occurrences. | stats count(function) AS new_NAME | stats count(source_IP) |

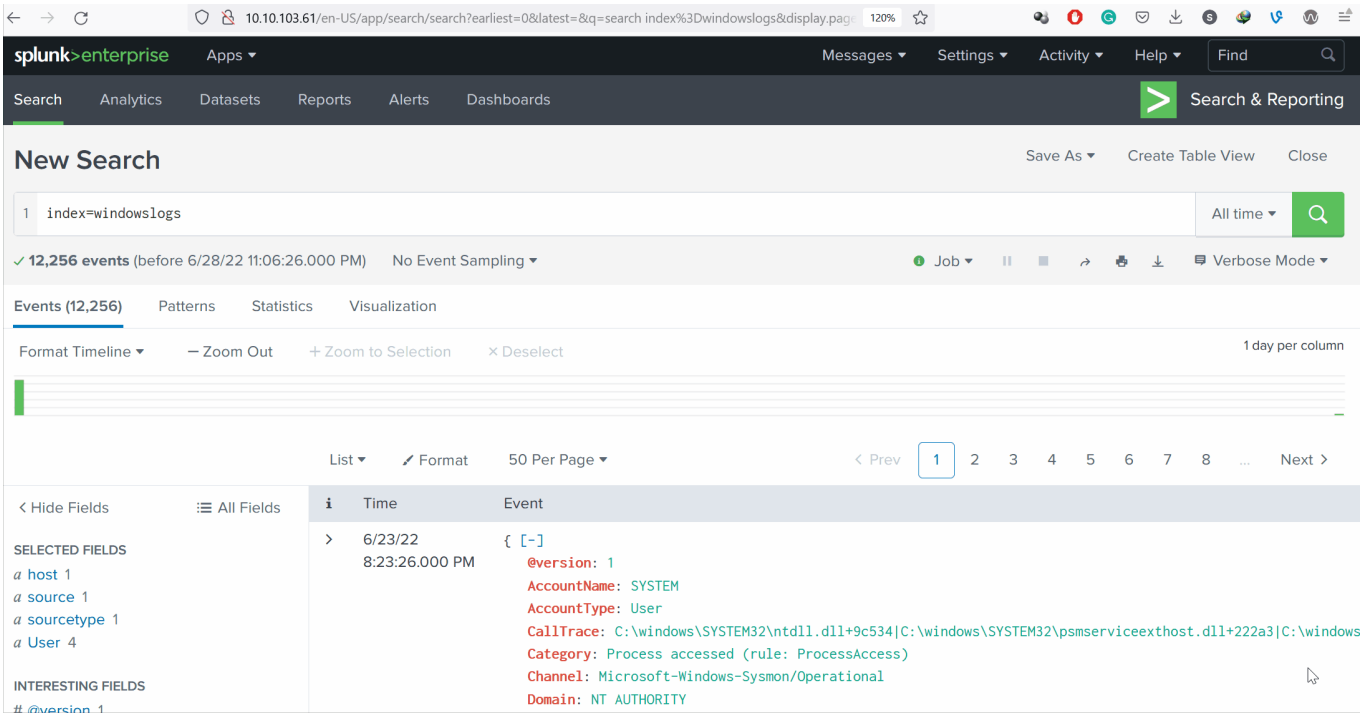
### Splunk Chart Commands

These are very important types of transforming commands that are used to present the data in table or visualization form. Most of the chart commands utilize various stat commands.

Chart

| | |
|---|---|
| **Command** | **chart** |
| **Explanation** | The chart command is used to transform the data into tables or visualizations. |
| **Syntax** | \| chart |
| **Example** | \| `chart count by User` |

**Search Query:** `index=windowslogs | chart count by User`




Timechart

| | |
|---|---|
| **Command** | **timechart** |
| **Explanation** | The timechart command returns the time series chart covering the field following the function mentioned. Often combined with STATS commands. |
| **Syntax** | \| timechart function <field_name> |
| **Example** | \| `timechart count by Image` |

The following query will display the Image chart based on the time.

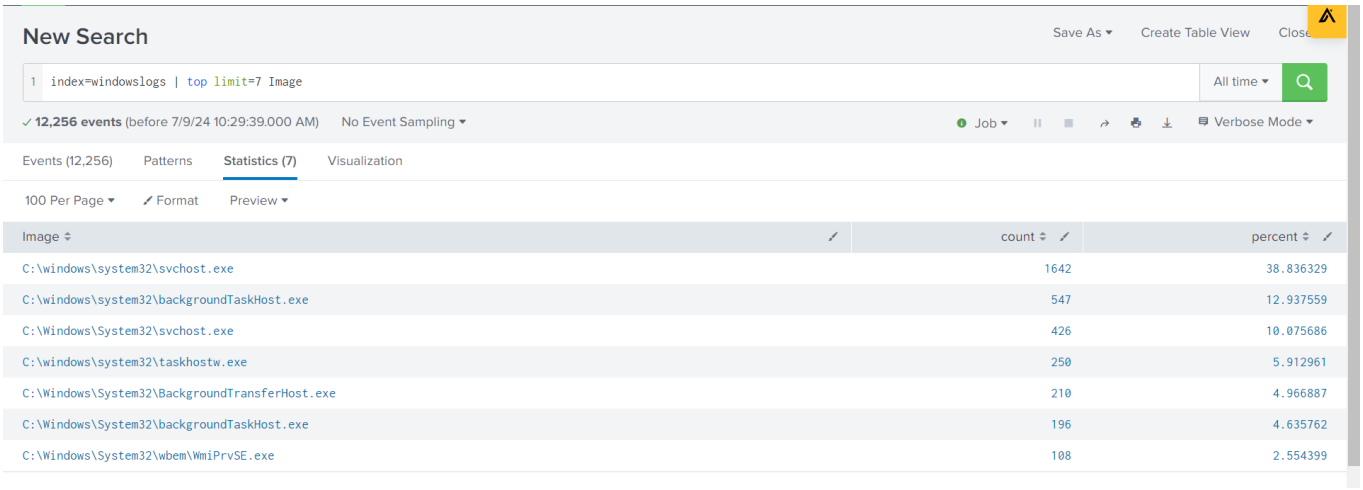
**Search Query:** `index=windowslogs | timechart count by Image`

Answer the questions below

List the top 8 Image processes using the top command - what is the total count of the 6th Image?

196 ✓ Correct Answer

We can see from the output that the total count of the 6th image is this

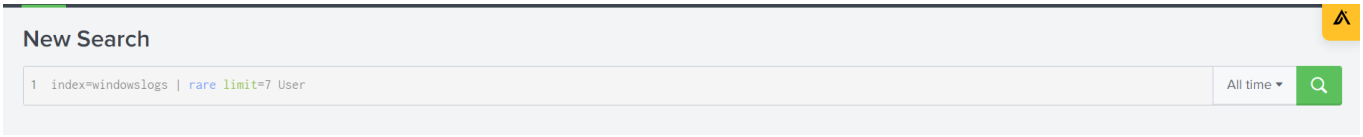



| Image | count | percent |
|---|---|---|
| C:\windows\system32\svchost.exe | 1642 | 38.836329 |
| C:\windows\system32\backgroundTaskHost.exe | 547 | 12.937559 |
| C:\Windows\System32\svchost.exe | 426 | 10.075686 |
| C:\windows\system32\taskhostw.exe | 250 | 5.912961 |
| C:\Windows\System32\BackgroundTransferHost.exe | 210 | 4.966887 |
| C:\Windows\System32\backgroundTaskHost.exe | 196 | 4.635762 |
| C:\Windows\System32\wbem\WmiPrvSE.exe | 108 | 2.554399 |

Using the rare command, identify the user with the least number of activities captured?

James ✓ Correct Answer

When we type this into search we can see the user is James




Create a pie-chart using the chart command - what is the count for the conhost.exe process?

70 ✓ Correct Answer