Mastering KQL

Interactive Learning and Practical Application

Attention

- MMS sessions are 60-75 minutes followed by LONG Q&A
- Please hold detailed questions until that Q&A starts
- Feel free to ask clarification questions
- Remind the speakers to use Zoom It when you can't see

Speakers

Rod Trent

Senior Program Manager, Microsoft

X @rodtrent

Sergey Chubarov

Ethical Hacker

Can't see our slides? Can't hear? Need to repeat the question? Call us out!

Session Learning Objectives

At the end of this session, you will have…

- A foundational understanding of Log Analytics and the Kusto Query Language
- An appreciation of the purpose of the most common KQL operations and functions
- The ability to construct security-related queries using these common operators and functions

What is KQL?

Let’s start here

What is KQL and What does it stand for?

Kusto Query Language
- Built for Performance
- Read-only

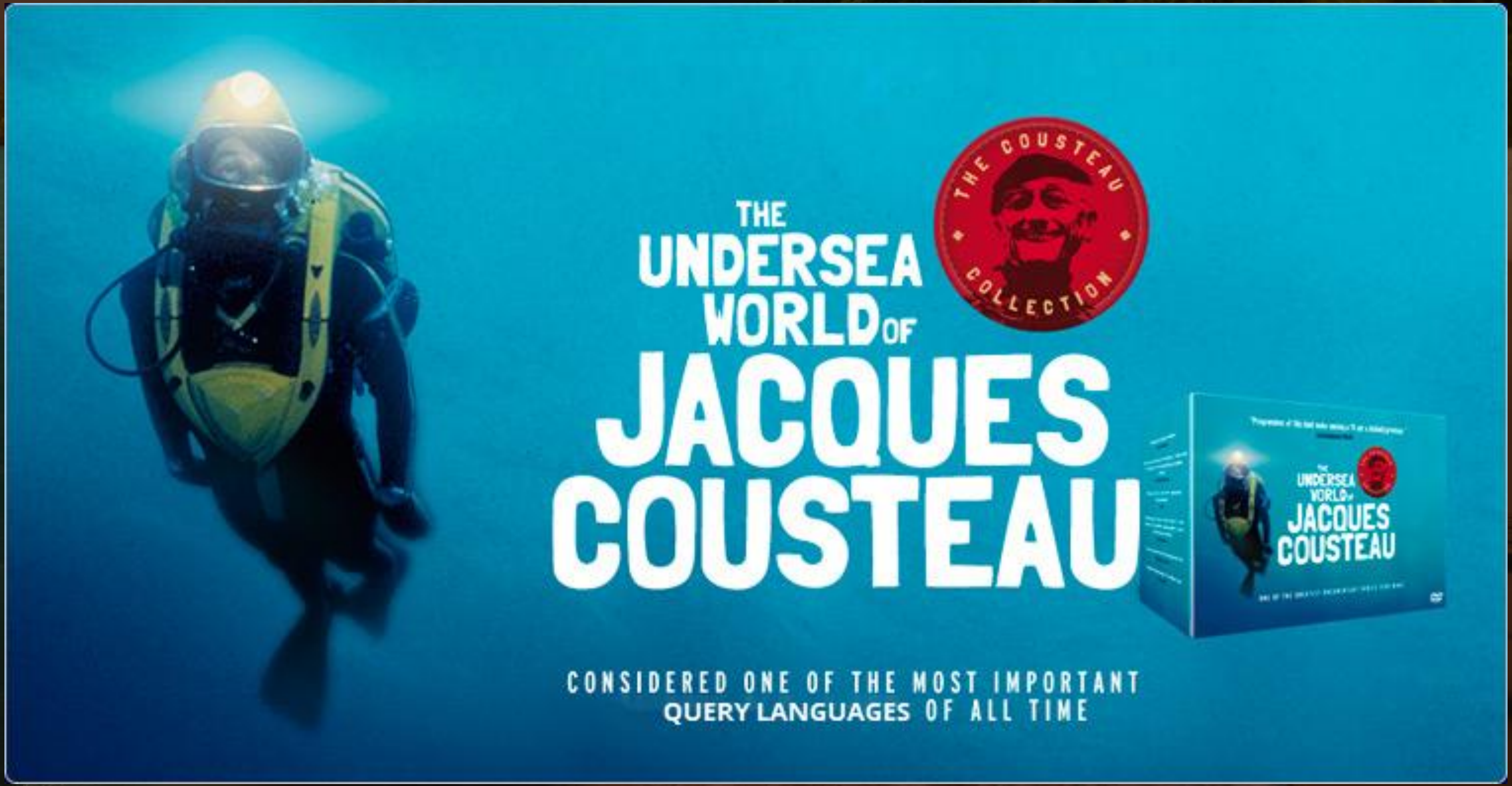
Importance:
- Log Analytics
- Azure Monitor
- Azure Security services
- Across all of Azure

It’s the language used to query the Azure log databases: Azure Monitor Logs, Azure Monitor Application Insights and others.

What’s Kusto?

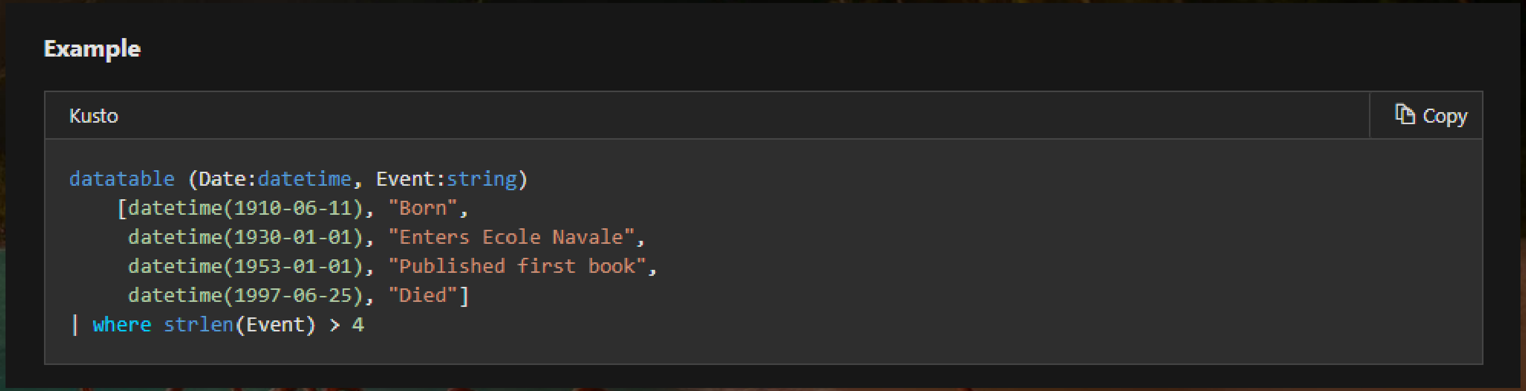
Kusto was the original codename for the Azure Application Insights platform that Azure Monitor is now based on and…

…is named after the famous explorer.


THE UNDERSEA WORLD OF JACQUES COUSTEAU
THE COUSTEAU COLLECTION
CONSIDERED ONE OF THE MOST IMPORTANT QUERY LANGUAGES OF ALL TIME

Jacques Cousteau

Actual references to **Jacques** in the Kusto documentation



Example

Kusto

```
datatable (Date:datetime, Event:string)
    [datetime(1910-06-11), "Born",
     datetime(1930-01-01), "Enters Ecole Navale",
     datetime(1953-01-01), "Published first book",
     datetime(1997-06-25), "Died"]
| where strlen(Event) > 4
```

https://docs.microsoft.com/en-us/azure/kusto/query/datatableoperator?pivots=azuremonitor

The Basics

Base Jumping

Log Analytics Workspace Overview

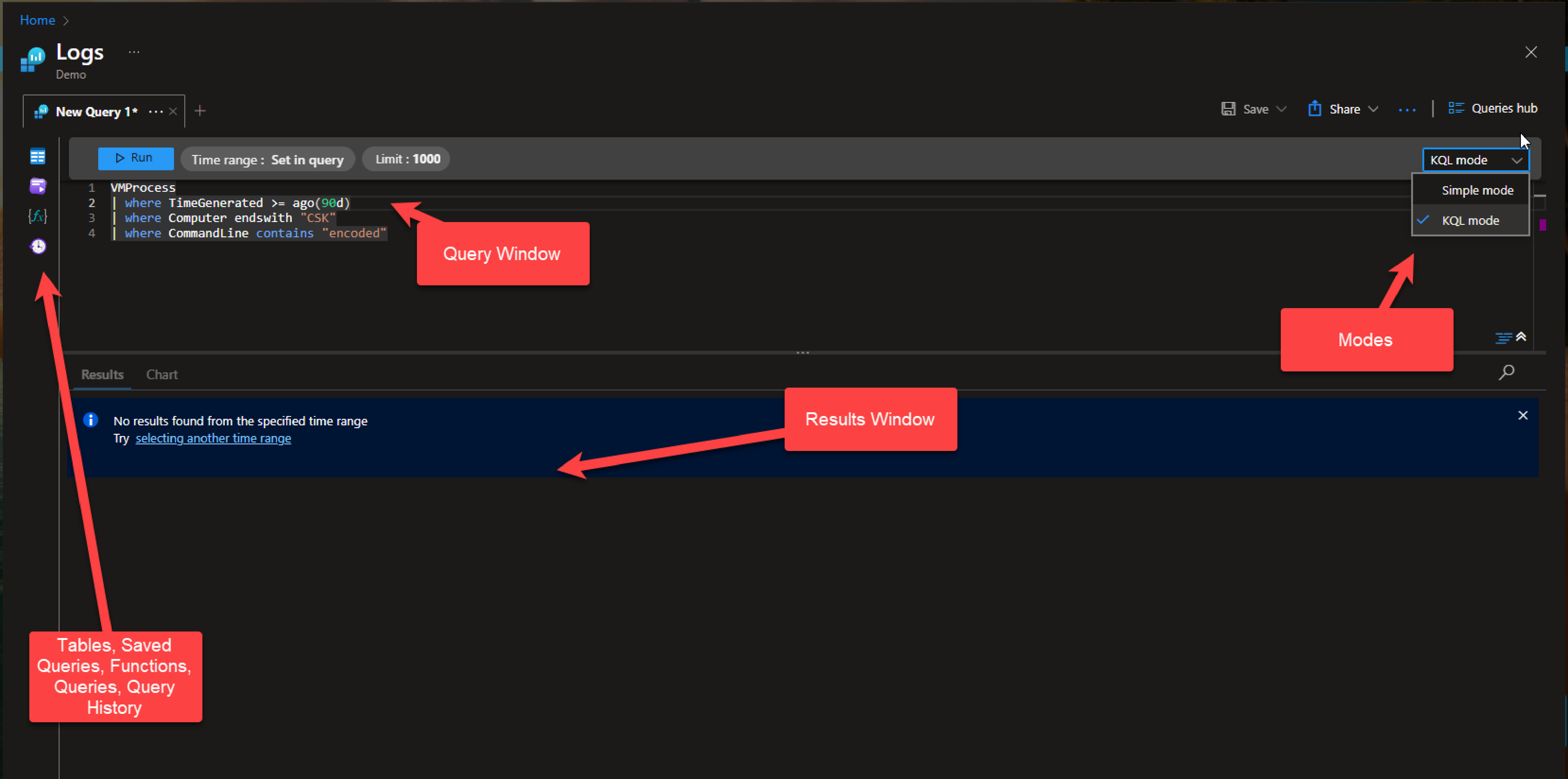
Follow along……

aka.ms/LADemo

aka.ms/MMSKQL

(Or using the CybersecuritySOC demo)

Query Window Overview

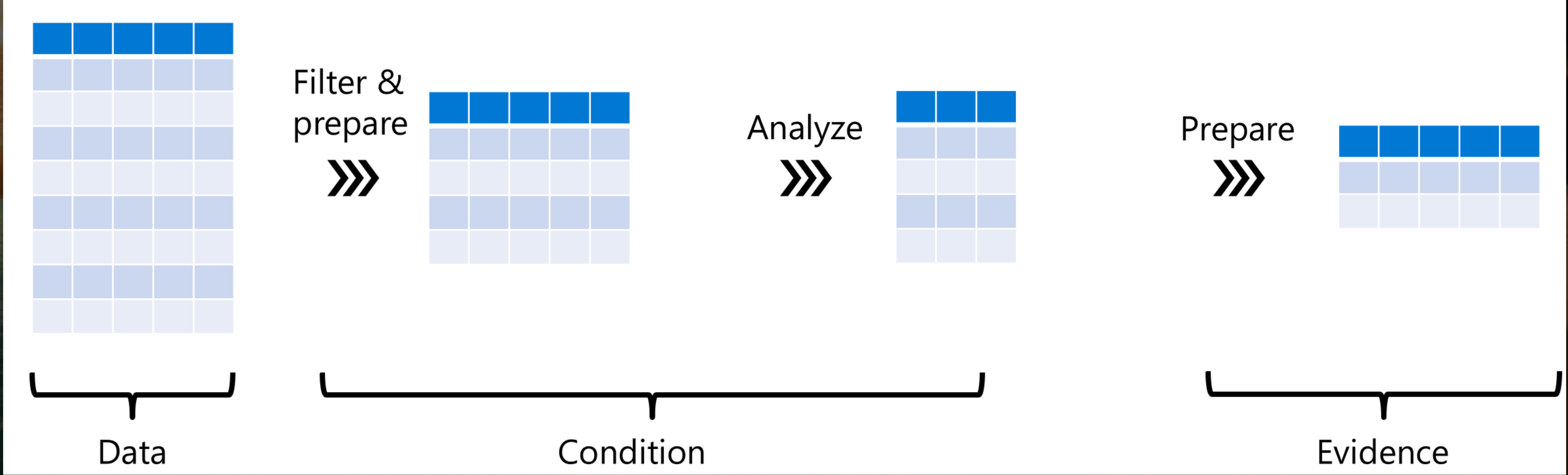


```
VMProcess
| where TimeGenerated >= ago(90d)
| where Computer endswith "CSK"
| where CommandLine contains "encoded"
```

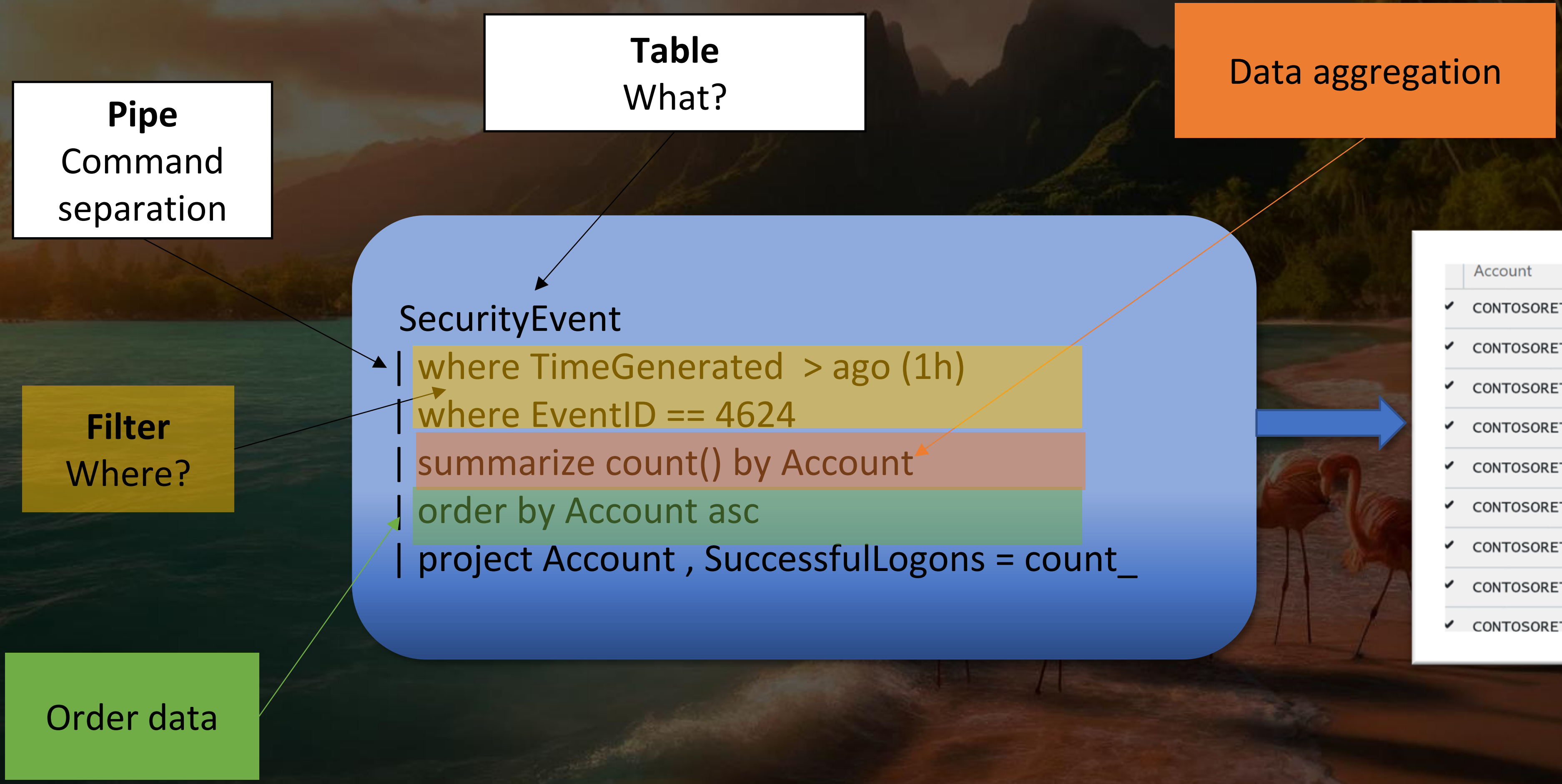
No results found from the specified time range
Try selecting another time range

Understanding the pipe

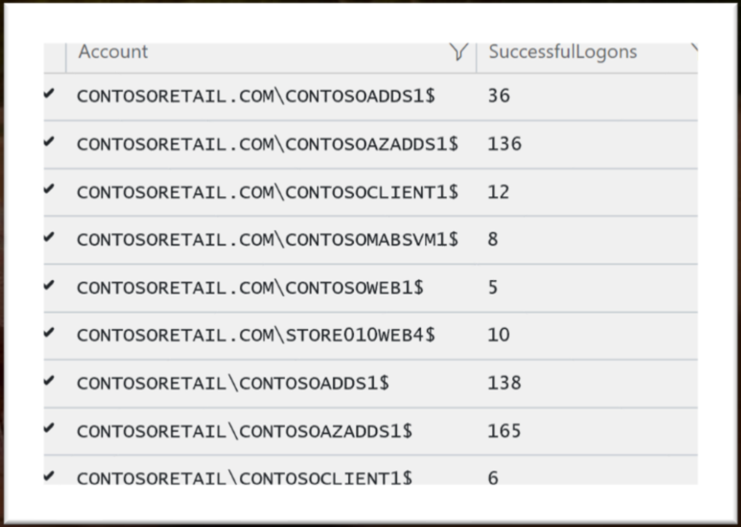
```
SecurityEvent | where EventID == "4624" | summarize count() by Account | top 10 by _count
```



(The Essence) Structure of KQL Queries



```
SecurityEvent
| where TimeGenerated  > ago (1h)
| where EventID == 4624
| summarize count() by Account
| order by Account asc
| project Account , SuccessfulLogons = count_
```



Account	SuccessfulLogons
CONTOSORETAIL.COM\CONTOSOADDS1$	36
CONTOSORETAIL.COM\CONTOSOAZADDS1$	136
CONTOSORETAIL.COM\CONTOSOCLIENT1$	12
CONTOSORETAIL.COM\CONTOSOMABSVM1$	8
CONTOSORETAIL.COM\CONTOSOWEB1$	5
CONTOSORETAIL.COM\STORE010WEB4$	10
CONTOSORETAIL\CONTOSOADDS1$	138
CONTOSORETAIL\CONTOSOAZADDS1$	165
CONTOSORETAIL\CONTOSOCLIENT1$	6

Operators and Functions

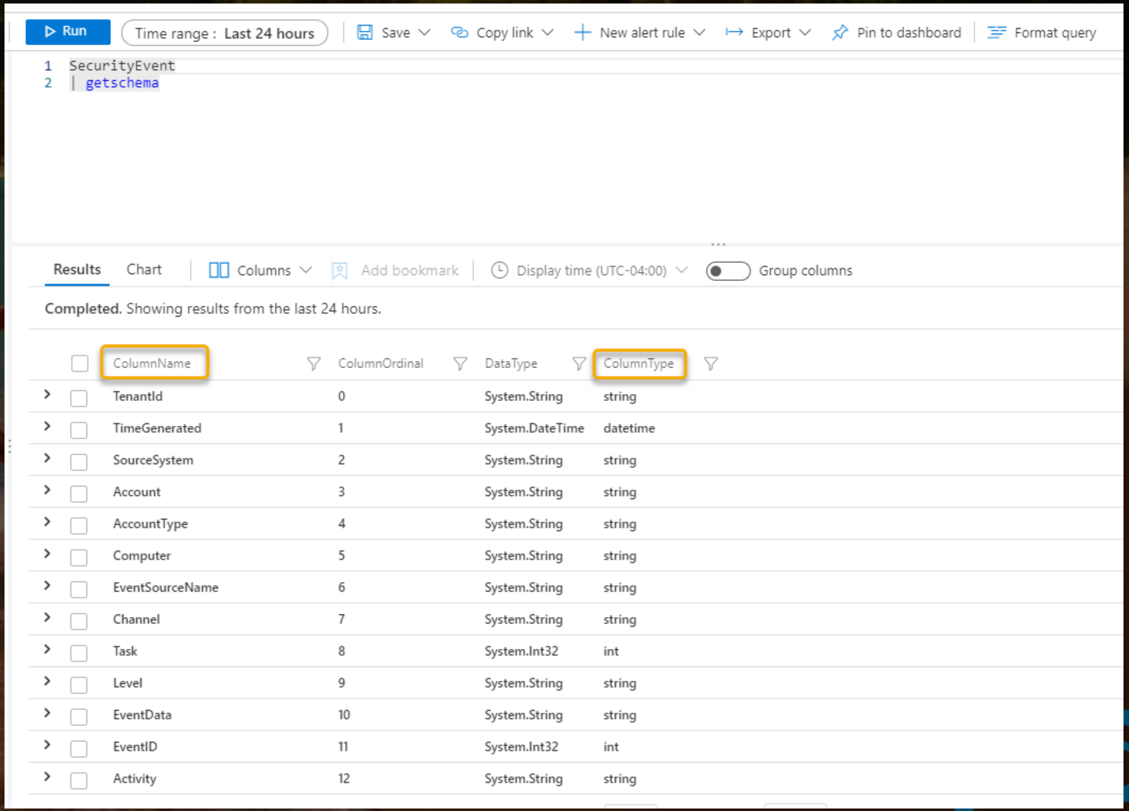
‘getschema’ operator

Shows the schema (Data-Types) for the table that you are working with.

Syntax: *Table | getschema*

Examples: *SecurityEvent | getschema*

- This shows what columns and data types are available to work with.
- This information is also viewable from the “Tables” pane in the UI



```
SecurityEvent
| getschema
```

Completed. Showing results from the last 24 hours.

ColumnName	ColumnOrdinal	DataType	ColumnType
TenantId	0	System.String	string
TimeGenerated	1	System.DateTime	datetime
SourceSystem	2	System.String	string
Account	3	System.String	string
AccountType	4	System.String	string
Computer	5	System.String	string
EventSourceName	6	System.String	string
Channel	7	System.String	string
Task	8	System.Int32	int
Level	9	System.String	string
EventData	10	System.String	string
EventID	11	System.Int32	int
Activity	12	System.String	string

‘search’ operator

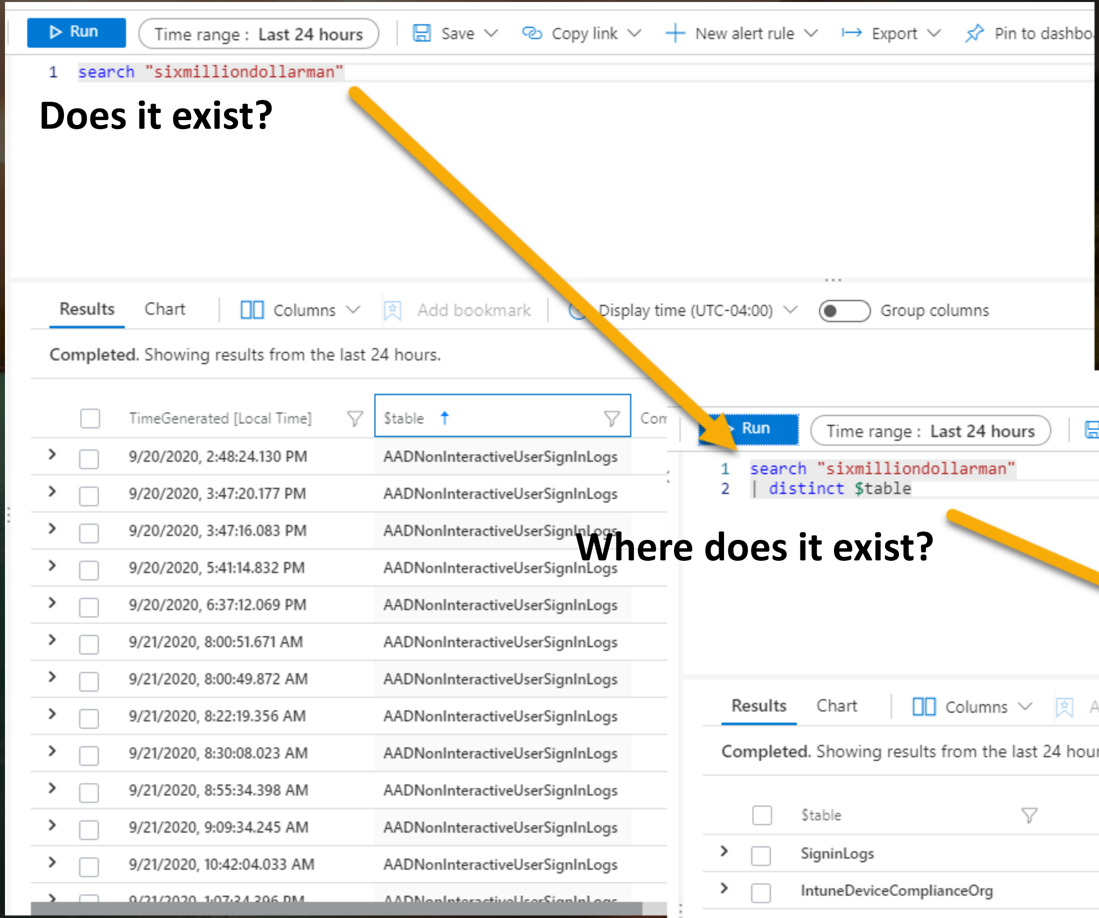
Easy to use. Use interactively for threat hunting but not in analytic rules.
Filter first to avoid long search times.

Syntax:	*Table | search*
Examples:	*search “administrator”*

- “T |” and “in (Tables)” are optional. If no table is specified, it will search all tables.
- Filter first before search to save time.
- The result set will include a “$table” field and will indicate the table name in the output, if more than one table is searched.
- **If you are going to use a file-path in your search, you will need to “@” before the file-path ..i.e.(@”C:/Windows/System32/hacked.dll”)**

Multi-Table Searches –3 Questions To Ask

The *search* operator provides a multi-table/multi-column search experience.

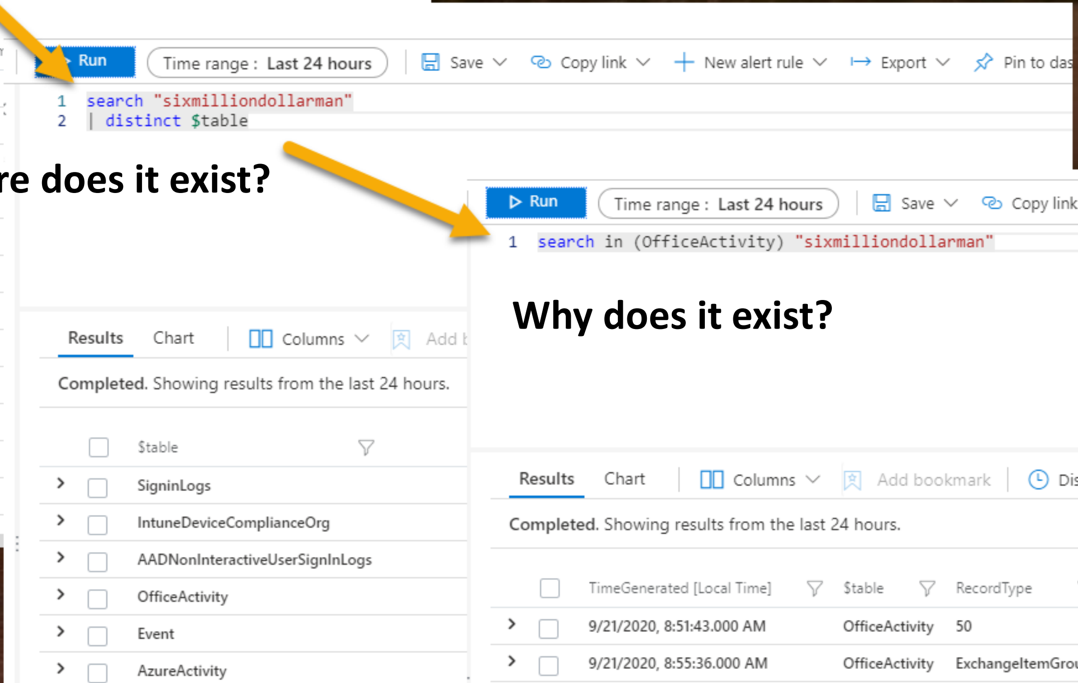


```
search "sixmilliondollarman"
```

Does it exist?

Completed. Showing results from the last 24 hours.

TimeGenerated [Local Time]	$table
9/20/2020, 2:48:24.130 PM	AADNonInteractiveUserSignInLogs
9/20/2020, 3:47:20.177 PM	AADNonInteractiveUserSignInLogs
9/20/2020, 3:47:16.083 PM	AADNonInteractiveUserSignInLogs
9/20/2020, 5:41:14.832 PM	AADNonInteractiveUserSignInLogs
9/20/2020, 6:37:12.069 PM	AADNonInteractiveUserSignInLogs
9/21/2020, 8:00:51.671 AM	AADNonInteractiveUserSignInLogs
9/21/2020, 8:00:49.872 AM	AADNonInteractiveUserSignInLogs
9/21/2020, 8:22:19.356 AM	AADNonInteractiveUserSignInLogs
9/21/2020, 8:30:08.023 AM	AADNonInteractiveUserSignInLogs
9/21/2020, 8:55:34.398 AM	AADNonInteractiveUserSignInLogs
9/21/2020, 9:09:34.245 AM	AADNonInteractiveUserSignInLogs
9/21/2020, 10:42:04.033 AM	AADNonInteractiveUserSignInLogs

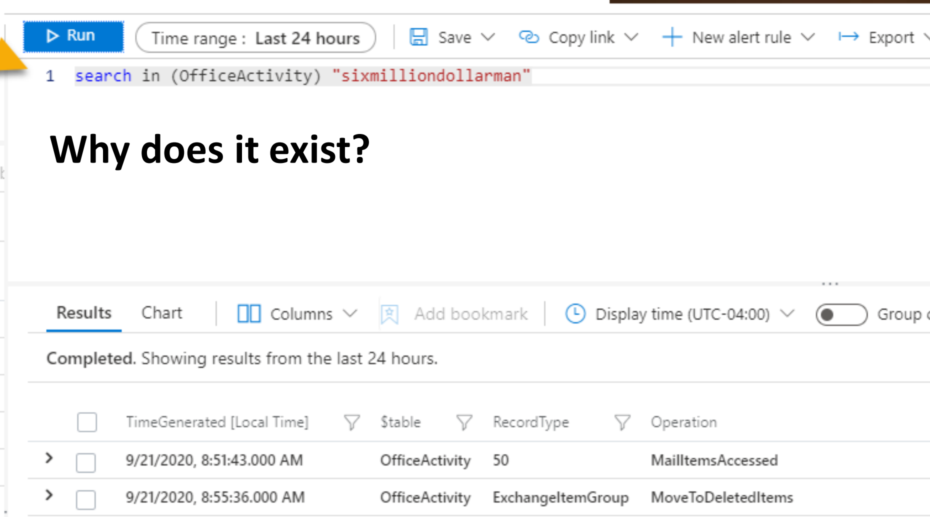


```
search "sixmilliondollarman"
| distinct $table
```

Where does it exist?

Completed. Showing results from the last 24 hours.

$table
SigninLogs
IntuneDeviceComplianceOrg
AADNonInteractiveUserSignInLogs
OfficeActivity
Event
AzureActivity



```
search in (OfficeActivity) "sixmilliondollarman"
```

Why does it exist?

Completed. Showing results from the last 24 hours.

TimeGenerated [Local Time]	$table	RecordType	Operation
9/21/2020, 8:51:43.000 AM	OfficeActivity	50	MailItemsAccessed
9/21/2020, 8:55:36.000 AM	OfficeActivity	ExchangeItemGroup	MoveToDeletedItems

Filtering Data

‘**where**’ operator

Filters a table to the subset of rows that satisfy a predicate.

Syntax: *Table | where Predicate*

Examples:

```
SecurityEvent
| where EventID == 1102
| summarize  LogClearedCount = count() by Computer
```

- **String**: ==, has, contains, startswith, endswith, matches regex, etc
- **Numeric/Date**: ==, !=, <, >, <=, >=
- **Lookup**: in ,!in, has_any
- **Empty**: isempty(), notempty(), isnull(), notnull()

’where’ example

```
SecurityEvent
| where TimeGenerated > ago(1h)
| where EventID == 4624 // Successful logon
| where AccountType =~ "user" // case insensitive
```

‘Count’ operator

Returns the number of records in the input record set.

Syntax:	*T | count*
Example:	*SecurityEvent | count*

‘count’ example

```
SecurityEvent
| where TimeGenerated > ago(1h)
| where EventID == 4624
| count
```

‘limit’ / ‘take’ operators

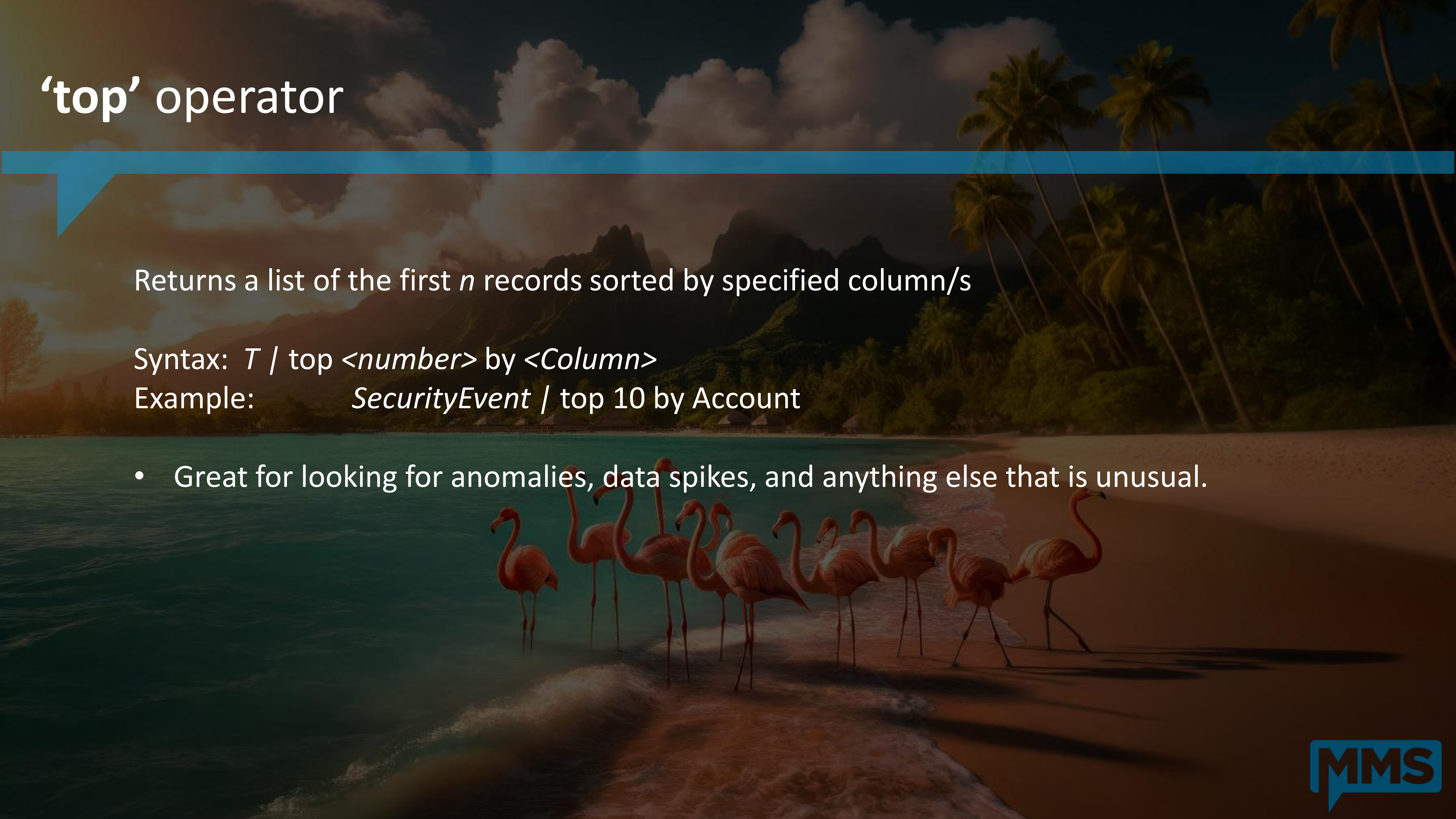
Return up to the specified number of rows.

Syntax: *T | limit <number>*
Example: *SecurityEvent | limit 5*

- Sort is not guaranteed to be preserved.
- Consistent result is not guaranteed (when running the same query twice)
- Very useful when trying out new queries.
- Default limit is 30,000.

‘limit/take’ example

```
SecurityEvent
| limit 10


SecurityEvent
| where TimeGenerated > ago(1h)
| where EventID == 4624
| where AccountType =~ "user"
| take 10
```



‘top’ operator

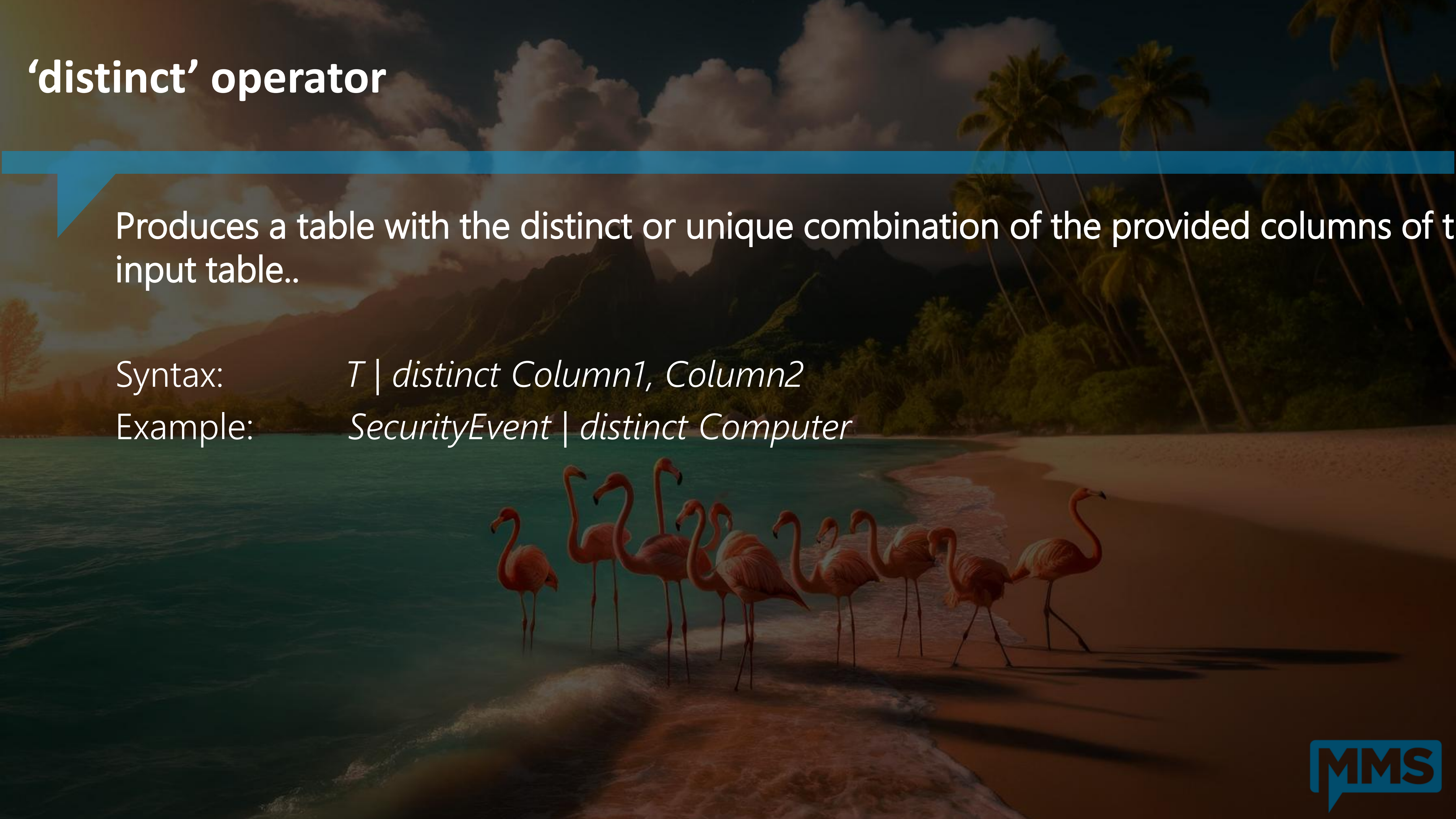
Returns a list of the first *n* records sorted by specified column/s

Syntax: *T* | top *<number>* by *<Column>*
Example: *SecurityEvent* | top 10 by Account

- Great for looking for anomalies, data spikes, and anything else that is unusual.

‘top’ example

```
SecurityEvent
| where TimeGenerated > ago(1h)
| where EventID == 4624
| summarize count() by Account
| top 10 by count_
```



‘distinct’ operator

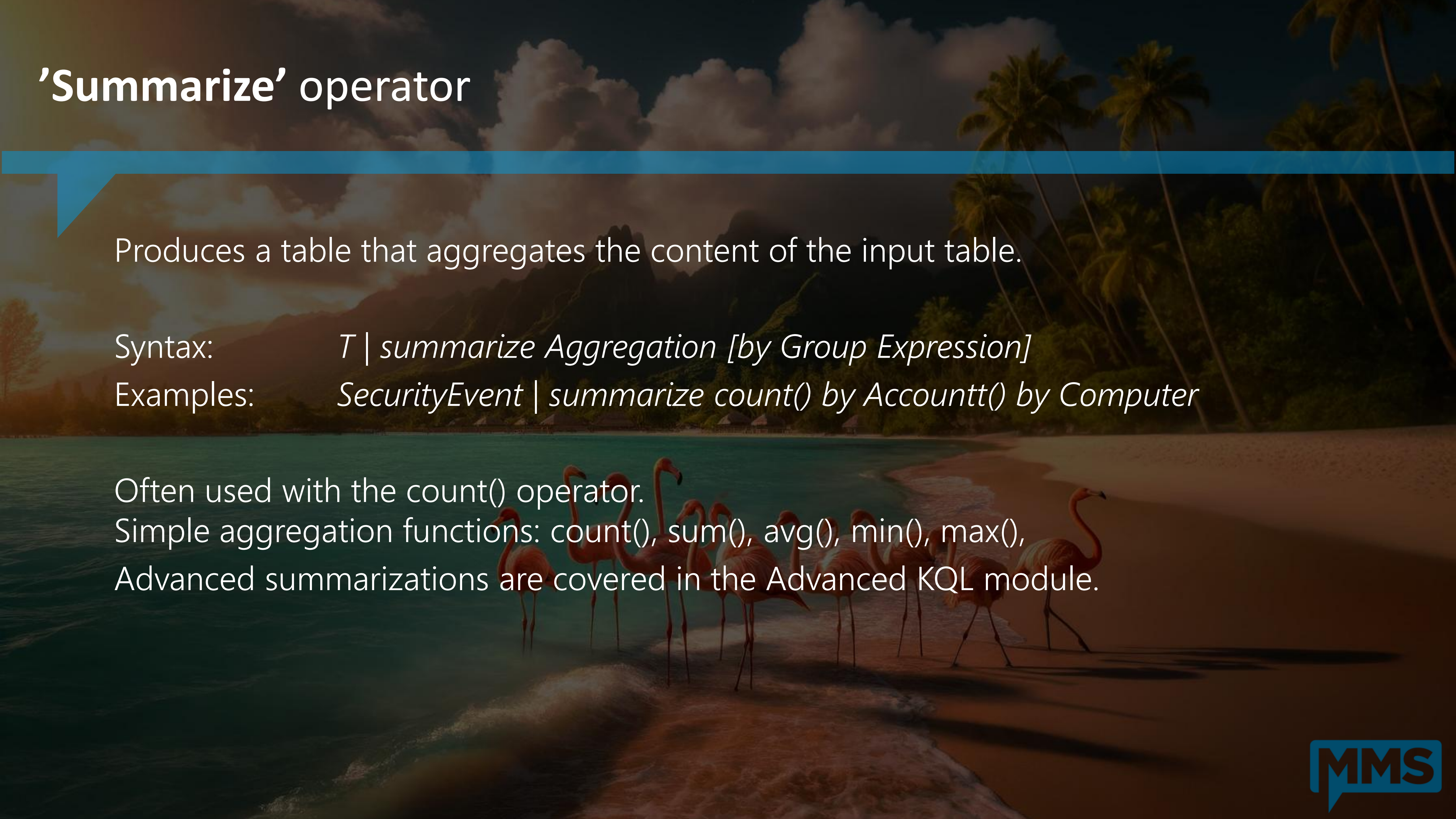
Produces a table with the distinct or unique combination of the provided columns of t
input table..

Syntax:	*T | distinct Column1, Column2*
Example:	*SecurityEvent | distinct Computer*

‘distinct’ example

```
SecurityEvent
| where EventID = 4624
| distinct Computer


ThreatIntelligenceIndicator
| where ThreatType contains “BotNet”
| distinct NetworkSourceIP
```

Analyzing Data



’Summarize’ operator

Produces a table that aggregates the content of the input table.

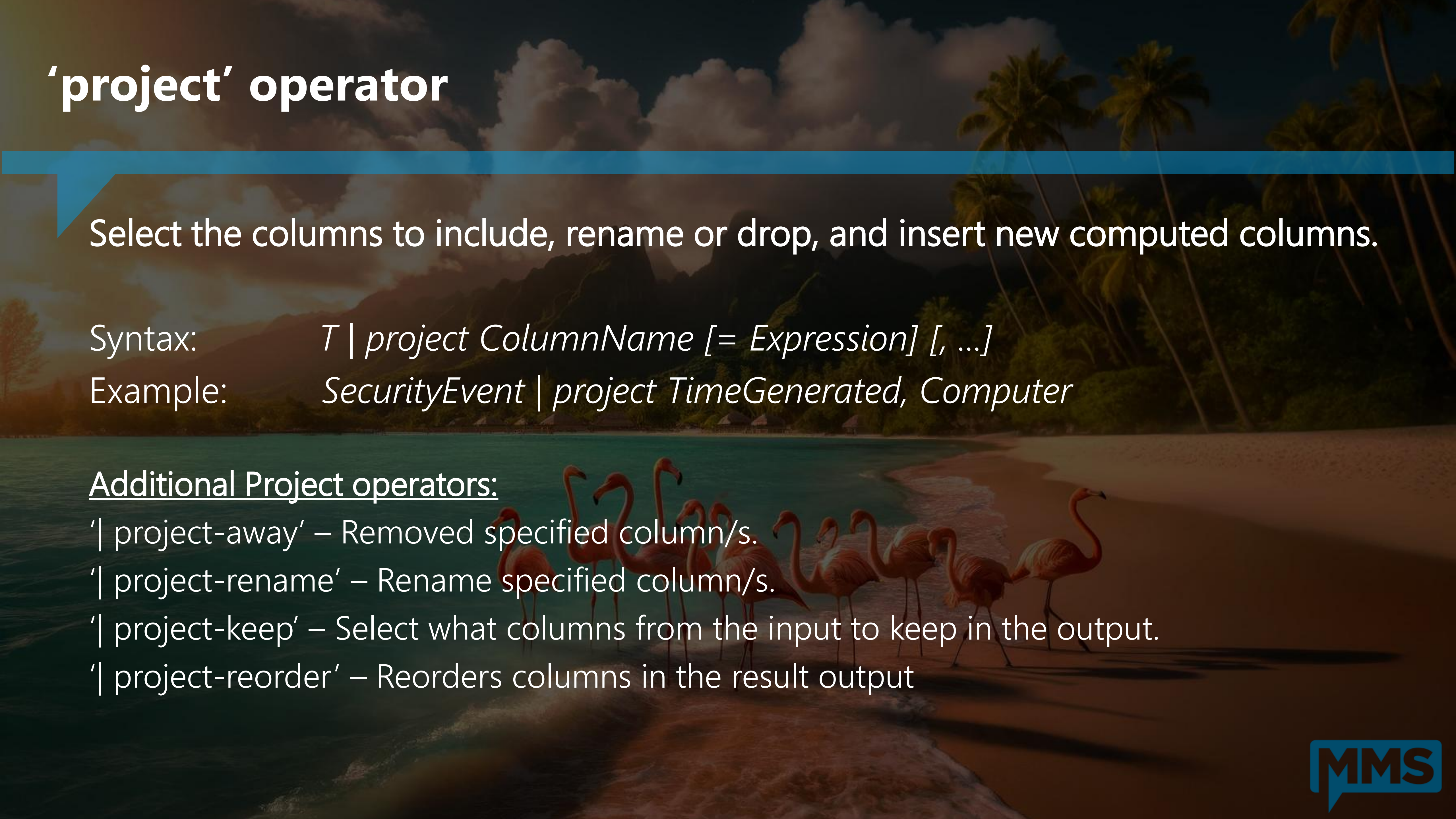
Syntax:	*T | summarize Aggregation [by Group Expression]*
Examples:	*SecurityEvent | summarize count() by Accountt() by Computer*

Often used with the count() operator.
Simple aggregation functions: count(), sum(), avg(), min(), max(),
Advanced summarizations are covered in the Advanced KQL module.

‘summarize’ example

```
//Logons with clear text password by target account
SecurityEvent
| where EventID == 4624 and LogonType == 8
| summarize count() by Computer
```

Presenting Data



‘project’ operator

Select the columns to include, rename or drop, and insert new computed columns.

Syntax: *T | project ColumnName [= Expression] [, ...]*

Example: *SecurityEvent | project TimeGenerated, Computer*

<u>Additional Project operators:</u>

‘| project-away’ – Removed specified column/s.

‘| project-rename’ – Rename specified column/s.

‘| project-keep’ – Select what columns from the input to keep in the output.

‘| project-reorder’ – Reorders columns in the result output

‘project’ example

```
Perf
| where CounterName == "Free Megabytes"
| project Computer, CounterName
```

‘extend’ operator

Create calculated columns and append them to the result set.

Syntax: *T | extend ColumnName [= Expression] [, ...]*
Example: *SecurityEvent | extend ComputerNameLength = strlen(Computer)*

- The new added column is not indexed.
- To only change a column name, use ‘project-rename’.
- Useful functions for ‘extend’: iff, extract

‘extend’ example

```
Perf
| where CounterName == "Free Megabytes"
| extend FreeKB = CounterValue * 1024
| extend FreeGB = CounterValue / 1024
| extend FreeMB = CounterValue
| project Computer, CounterName, FreeGB, FreeMB,
FreeKB
```

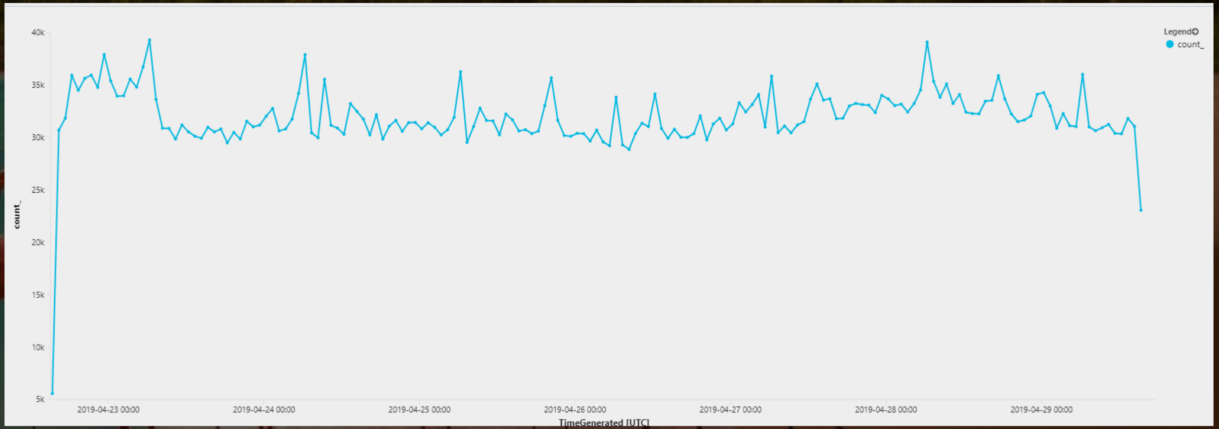
'Summarize' : bin and time series

A very useful summarize operation is creating time series:

SecurityEvent | summarize count() by bin(TimeGenerated, 1h) | render timechart

[by Group Expression]

() by Computer



Other time measurements: 1h, 5d, 10m (defaults to 1h)

Can create multiple legends by aggregating additional field

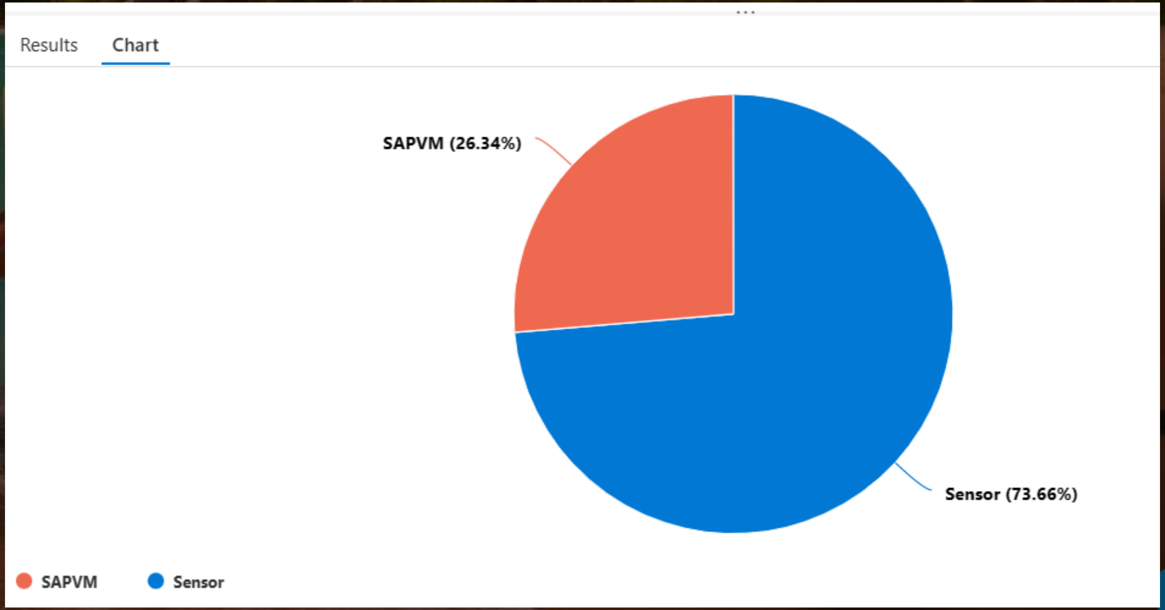
Render Operator

Generates a visualization of the query results.

Syntax: *T* | render *Visualization [with (PropertyName = PropertyValue [, ...])]*

Supported visualizations:

- Areachart
- Barchart
- Columnchart
- Piechart
- Scatterchart
- timechart



‘bin’ and ‘render’ example

```
SecurityEvent
| where TimeGenerated > ago(7d)
| summarize count() by bin(TimeGenerated, 1d)
| render barchart
```

‘let’ statement

The **let** command creates a temporary variable that can hold a single value, list, or table

- Used as a traditional variable before a complex query
- Used for allow, deny, and exclusion lists
- Used as a temporary container for a table

For good examples of **let** variables and complex queries, check out the Sentinel scheduled rule templates

‘let’ statement example

```
let suspiciousAccounts = datatable(account: string) [
    @"\administrator",
    @"NT AUTHORITY\SYSTEM“
];
SecurityEvent | where Account in (suspiciousAccounts)
```

Merging Data

‘union’ operator

Takes two or more tables and returns the rows of all of them.

Example: *SecurityEvent| union (SecurityAlert | where AlertSeverity == "high")*

- kind=inner(common columns), outer (all columns- default)
- Supports wildcard to union multiple tables (union Security*)
- Can union between tables from different clusters (or workspaces)

'union' example

```
SecurityEvent
| union Heartbeat
| summarize count() by Computer
```

‘join’ operator

Merge the rows of two tables to form a new table by matching values of the specified column(s) from ea
table.

Syntax: LeftTable | join [JoinParameters] (RightTable) on Attributes

Example: *SecurityEvent | join (SecurityAlert | where AlertSeverity == "high") on Status*

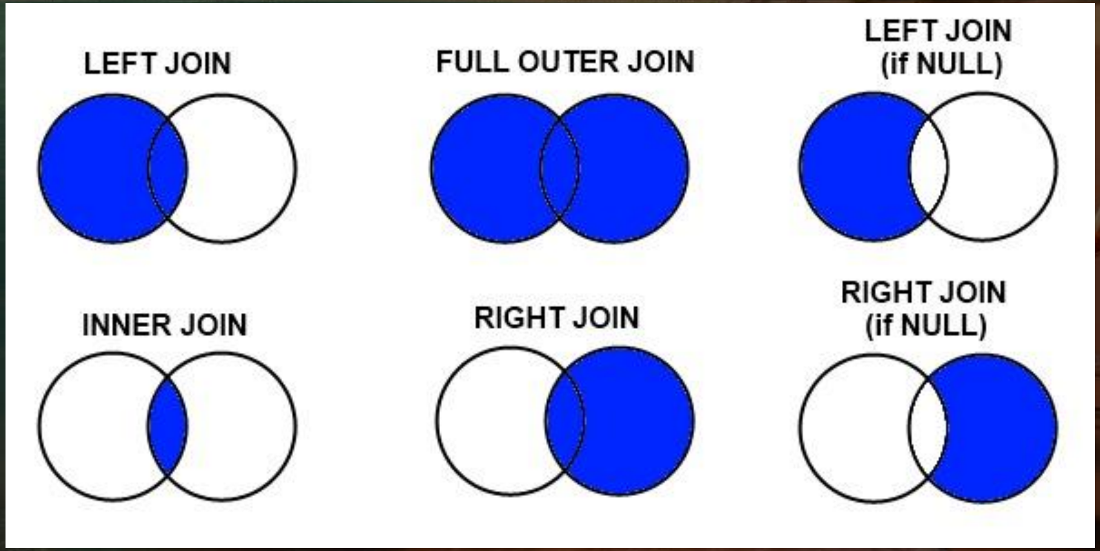


Table1 | join (Table2) on CommonColumn, $left.Col1 == $right.Col2

‘join’ example

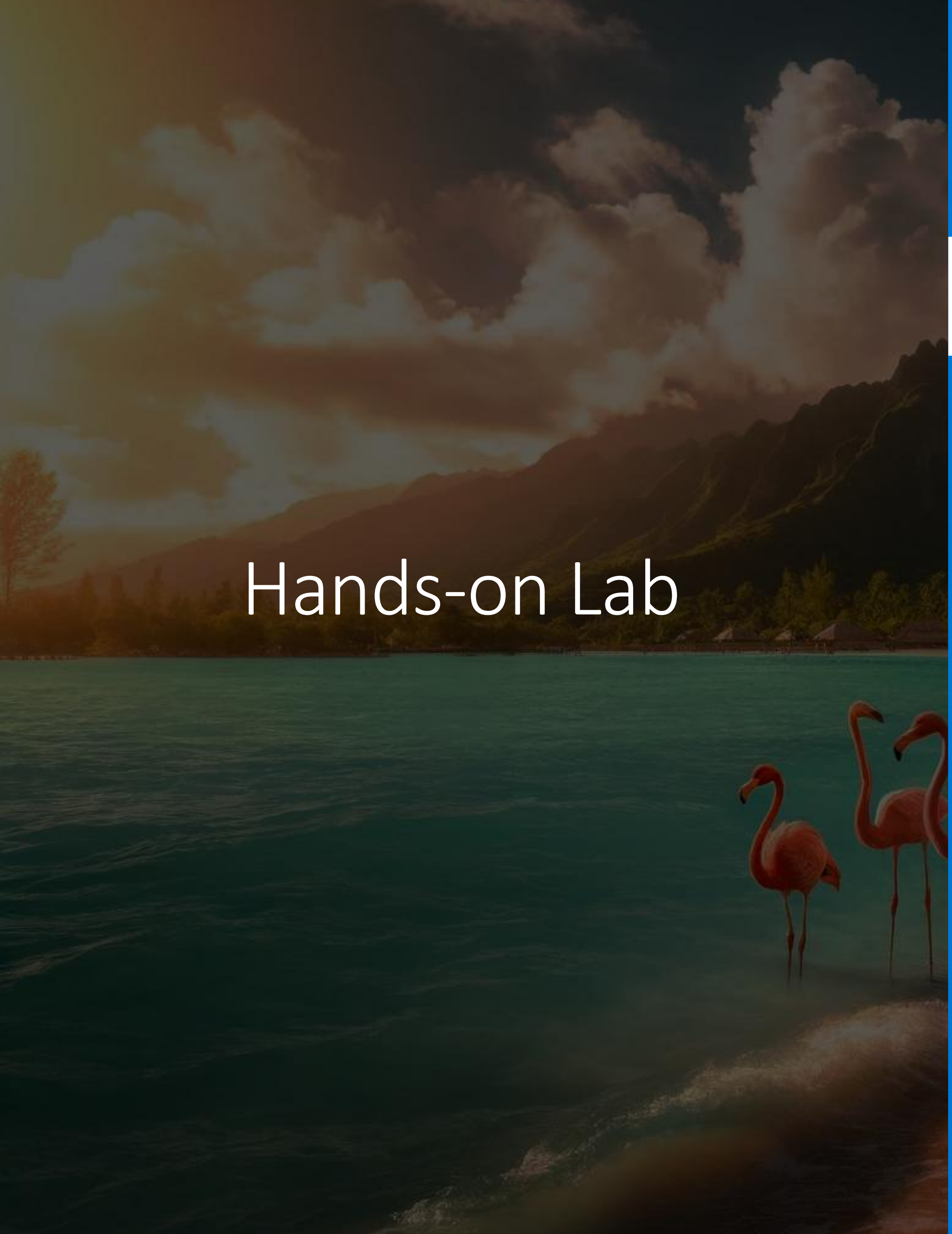
```
SecurityEvent
| join Heartbeat on Computer
| where EventID == "4688"
| project Computer, OSType, OSMajorVersion,
Version
```

Syntax Review

- Each command is separated by a pipe "|" (like PowerShell).
- KQL queries are case-insensitive by default.
- Lines are not delimited, no (;) to mark the end of a line. (Except for Let statements)
- Command are typically 'stacked' one per line (though single line is acceptable).
- Extra spaces between commands are ignored.
- Data types include basic, int (long), Boolean, string, datetime, timestamp, or dynamic (JSON).
- Operators include ==, !=, contains, has, !in, startswith, <=, >=, and many more.
- In-line commenting "\\" are supported.
- Anything in quoted is treated as a string.
- Numbers with no quotes are treated as a long integer.

You do you

Lighten up, Francis



Hands-on Lab

1st Scenario

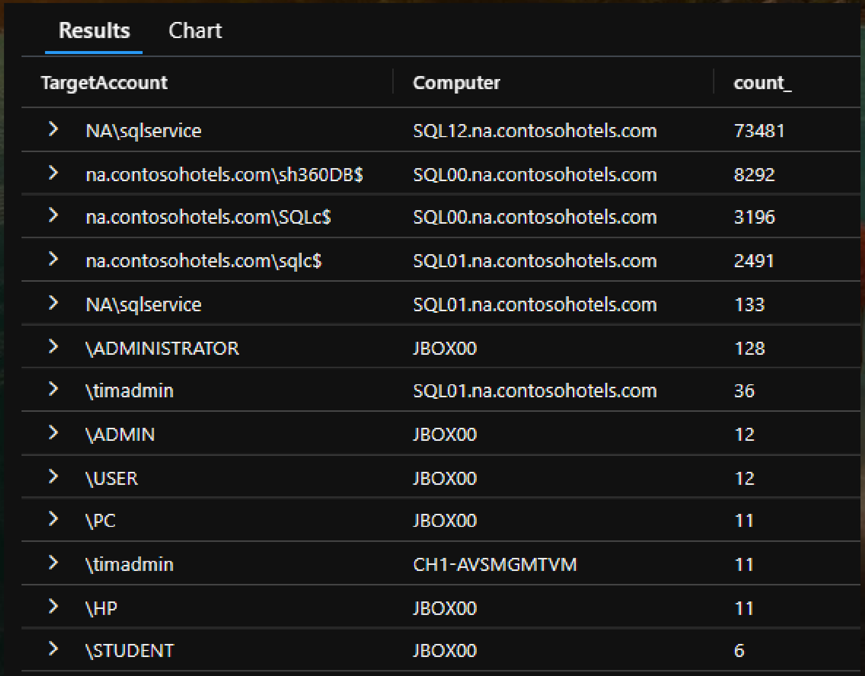
Your SOC has just discovered that a hacker has been using brute force attacks against your network for the past 7 days.
You need to find the count of failed logins for each user account and computer being attacked during that period.

Hints and guidelines:

- Use the SecurityEvent table to begin your search.
- How would you find the EventID for failed logins using KQL?
- What column would have the account name?

1st Lab Exercise

```
// Find the count of failed logins by Account Name
// run parts of the query, adding a line at the time, to learn more

SecurityEvent
| where TimeGenerated <= ago(7d)
| where EventID == 4625
| summarize count() by TargetAccount, Computer
```



TargetAccount	Computer	count_
NA\sqlservice	SQL12.na.contosohotels.com	73481
na.contosohotels.com\sh360DB$	SQL00.na.contosohotels.com	8292
na.contosohotels.com\SQLc$	SQL00.na.contosohotels.com	3196
na.contosohotels.com\sqlc$	SQL01.na.contosohotels.com	2491
NA\sqlservice	SQL01.na.contosohotels.com	133
\ADMINISTRATOR	JBOX00	128
\timadmin	SQL01.na.contosohotels.com	36
\ADMIN	JBOX00	12
\USER	JBOX00	12
\PC	JBOX00	11
\timadmin	CH1-AVSMGMTVM	11
\HP	JBOX00	11
\STUDENT	JBOX00	6

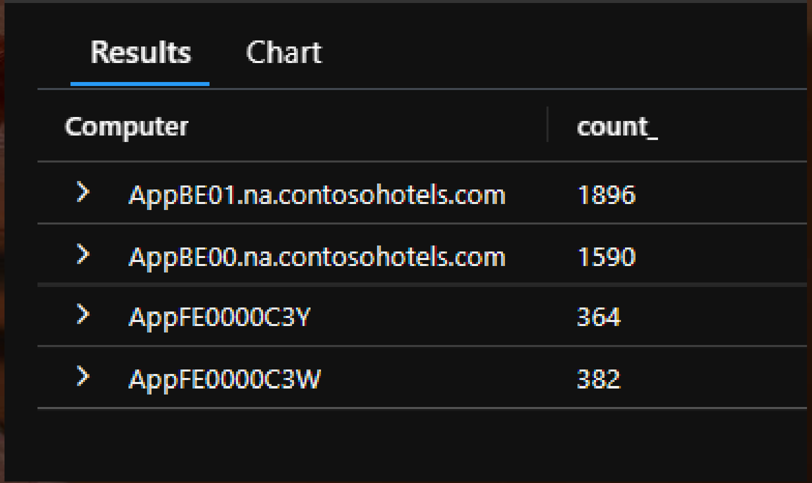
2nd Scenario

A hacker has compromised your network and has successfully logged into your network. To find the intruder, you need to find all Windows logon events starting 2 weeks ago until 1 week ago that occurred on a computer with a name which starts with “App” .

- Hints and guidelines:
- Windows security events are stored in the table “SecurityEvent”
- The logon event id is 4624. What is the name of the field which contains the event ID?
- What is the name of the field which represents the computer name?
- What should be the order of the commands for better performance?
- **Bonus:** Can you find the count per computer as well?

2nd Lab Exercise

```
// Find all Windows logon events starting 2 weeks ago until 1 week ago that occurred on a
computer with name which starts with “App”

SecurityEvent | limit 100 // Find relevant fields: Activity, EventID, Computer

SecurityEvent | summarize by Activity // find the Event signaling login

SecurityEvent
| where TimeGenerated between (ago(14d)..ago(7d))   // start with the time filter
| where EventID == "4624"
| where Computer startswith "App" // case insensitive
        // This is the solution, but there are so many results

SecurityEvent
| where TimeGenerated between (ago(14d)..ago(7d))
| where EventID == "4624"
| where Computer startswith "App"
| summarize count() by Computer
        // so let’s count per computer
```



Computer	count_
AppBE01.na.contosohotels.com	1896
AppBE00.na.contosohotels.com	1590
AppFE0000C3Y	364
AppFE0000C3W	382

3rd Scenario

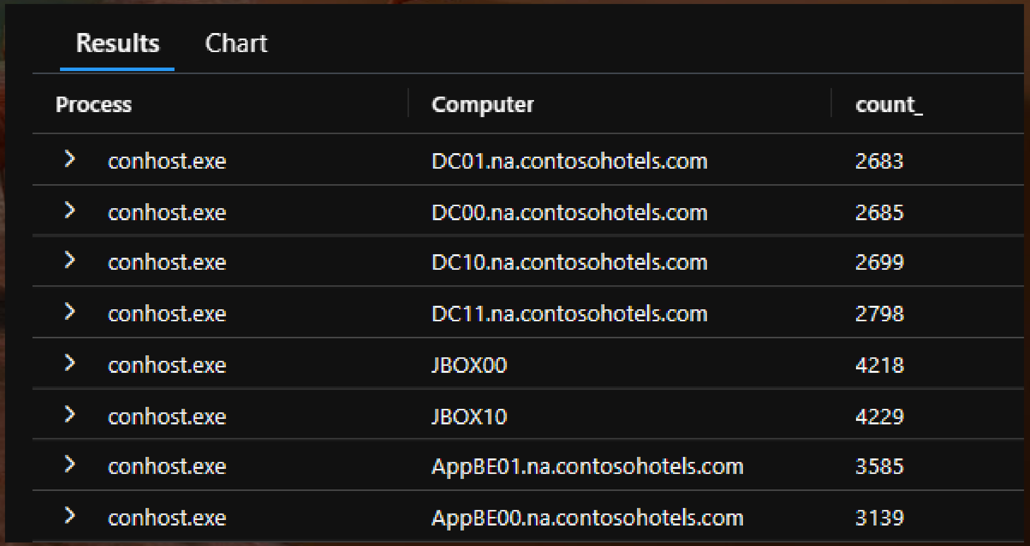
An APT has installed malware on your network .
In order to find the traces of malware, you need to find out how many times each process ran per computer.

Hints and guidelines:

- Event 4688 logs process creation.
- Which column represents the processes created?
- Which computer was it run on?

3rd Lab Exercise

```
// Find how many times each process ran per computer

SecurityEvent | summarize by Activity // Let’s find the event that includes
process names

SecurityEvent | where EventID == "4688" | limit 10
      // find the relevant field, in this case "Process"

SecurityEvent
| where EventID == "4688"
| summarize count() by Process, Computer
```



Process	Computer	count_
conhost.exe	DC01.na.contosohotels.com	2683
conhost.exe	DC00.na.contosohotels.com	2685
conhost.exe	DC10.na.contosohotels.com	2699
conhost.exe	DC11.na.contosohotels.com	2798
conhost.exe	JBOX00	4218
conhost.exe	JBOX10	4229
conhost.exe	AppBE01.na.contosohotels.com	3585
conhost.exe	AppBE00.na.contosohotels.com	3139

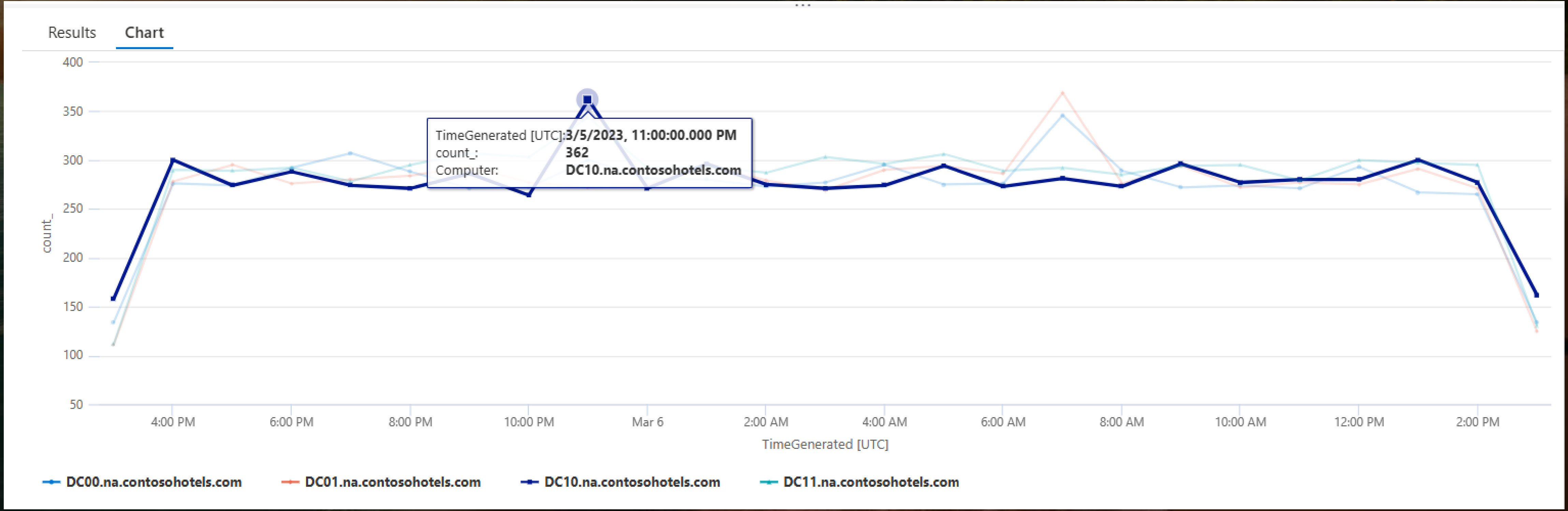
4th Scenario

Your SOC has just discovered a Crypto-Mining Agent has been installed on one of your domain controllers.
You need to chart the rate of process creation on all domain controllers in order to discover which DC has been compromised .

- Hints and guidelines:
- Process creation is Windows event 4688
- All Domain controller names start with "DC"
- This will be a time chart. (Think bin…)

4th Lab Exercise

```
// Chart the rate of process creation on all domain controllers.

SecurityEvent
| where Computer startswith "DC"
| where EventID == "4688" | summarize count() by Computer, bin(TimeGenerated, 1h)
| render timechart
```



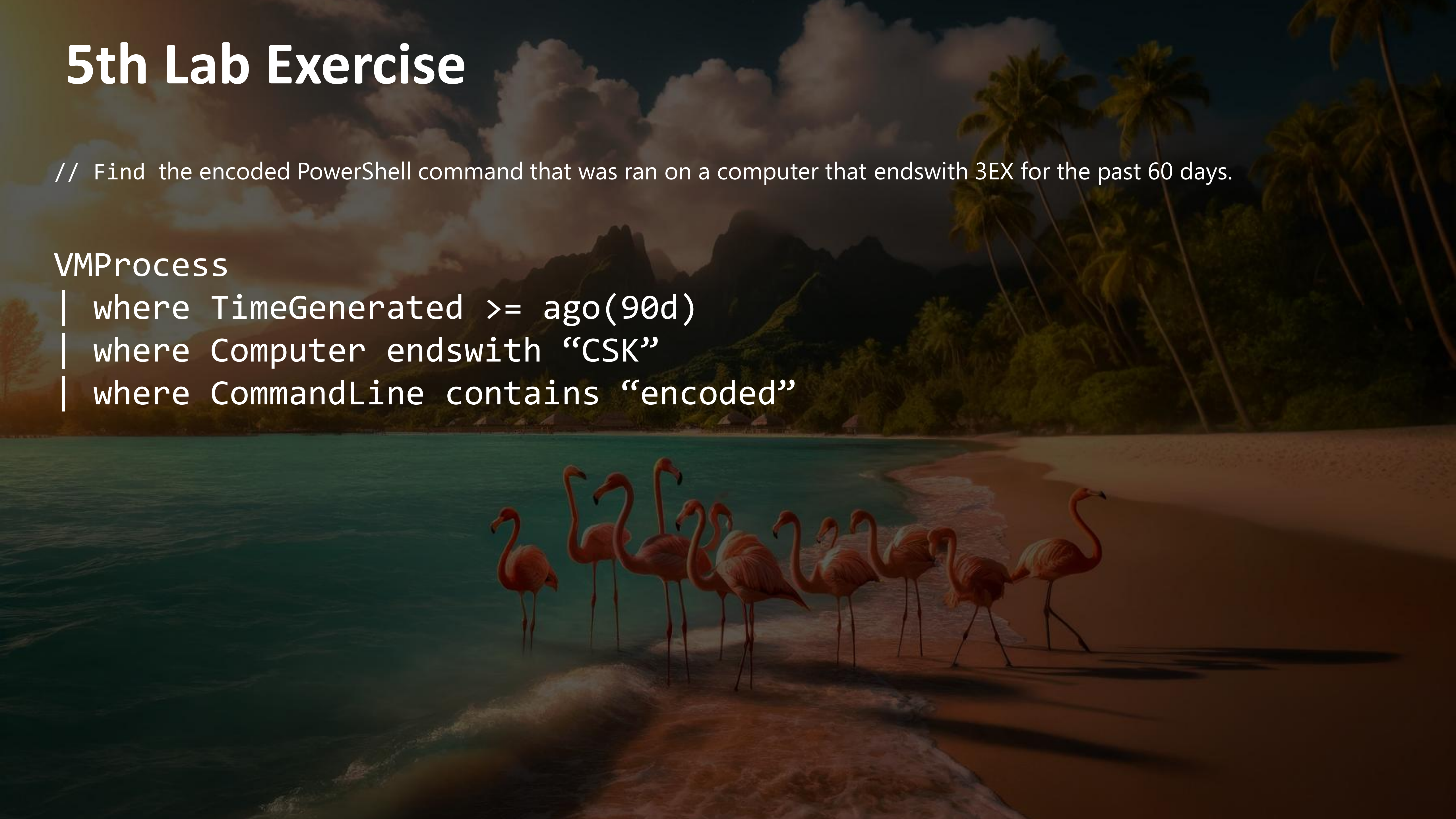
```
TimeGenerated [UTC]: 3/5/2023, 11:00:00.000 PM
count_:              362
Computer:            DC10.na.contosohotels.com
```

5th Scenario

A hacker has run an encoded PowerShell command on your network and has started exfiltrating data. You need to find the encoded PowerShell command that was ran on a computer that endswith CSK that was ran in the past 90 days.

Hints and Guidelines:

- Use the VMProcess table for your hunt.
- There *should* only be one result.



5th Lab Exercise

// Find the encoded PowerShell command that was ran on a computer that endswith 3EX for the past 60 days.

```
VMProcess
| where TimeGenerated >= ago(90d)
| where Computer endswith “CSK”
| where CommandLine contains “encoded”
```

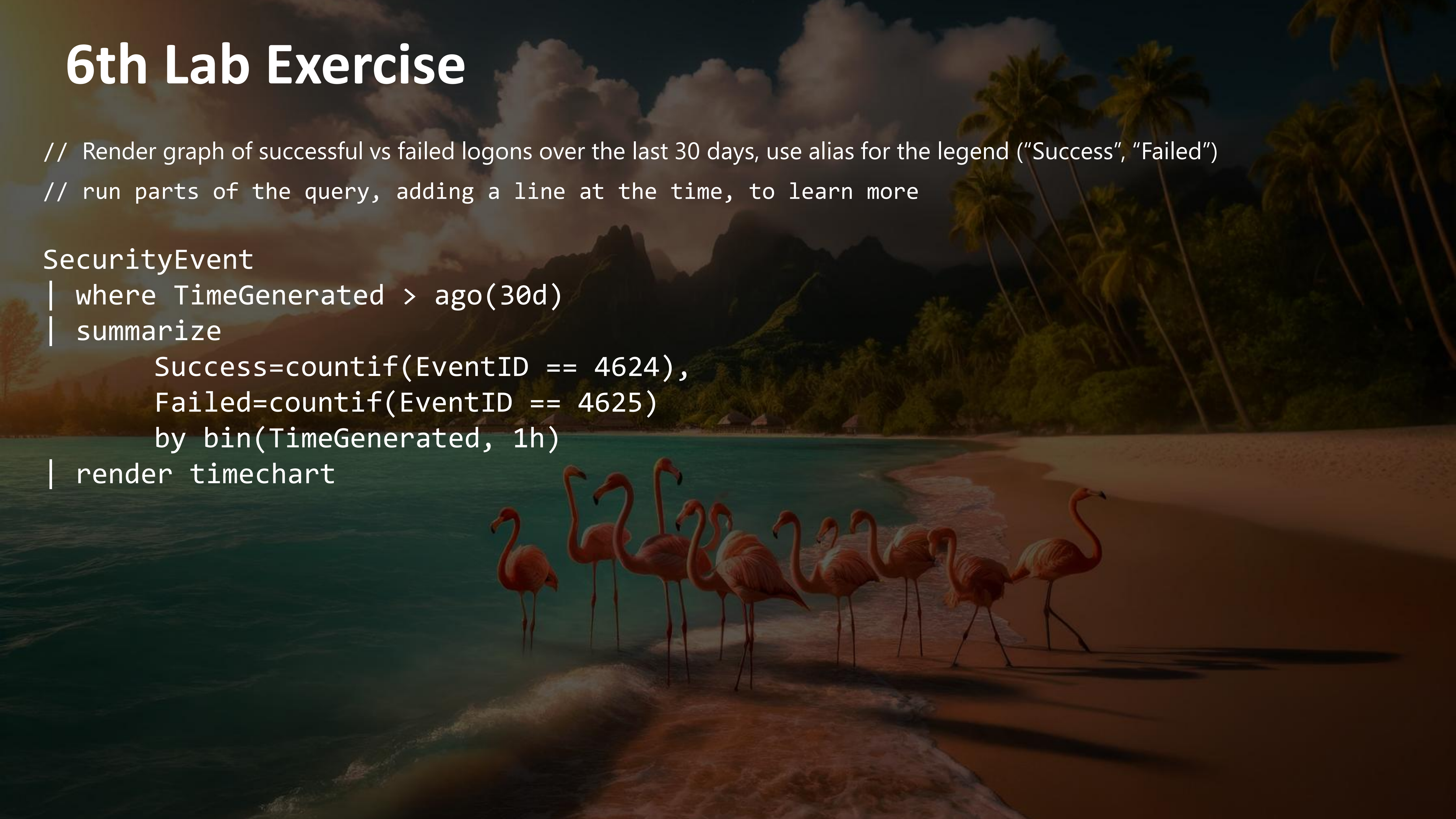
6th Scenario

As a part of your post incident response, you need to compare the successful and failed logons to determine what day the password spray took place.

You will need to render a graph of successful vs failed logons over the last 30 days, use alias for the legend (“Success”, “Failed”) to find your answer.

Hints and guideline:

- Utilize Countif for each EventID
- Remember this is a time chart.



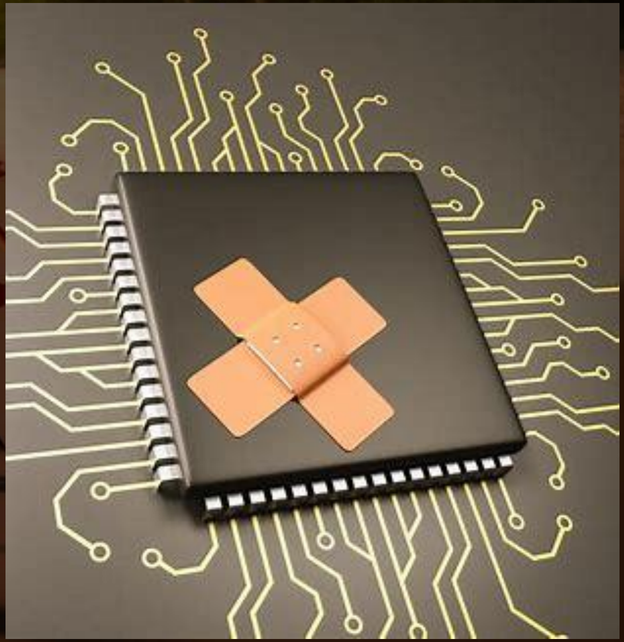
6th Lab Exercise

```
//  Render graph of successful vs failed logons over the last 30 days, use alias for the legend (“Success”, “Failed”)
// run parts of the query, adding a line at the time, to learn more

SecurityEvent
| where TimeGenerated > ago(30d)
| summarize
       Success=countif(EventID == 4624),
       Failed=countif(EventID == 4625)
       by bin(TimeGenerated, 1h)
| render timechart
```

7th Scenario

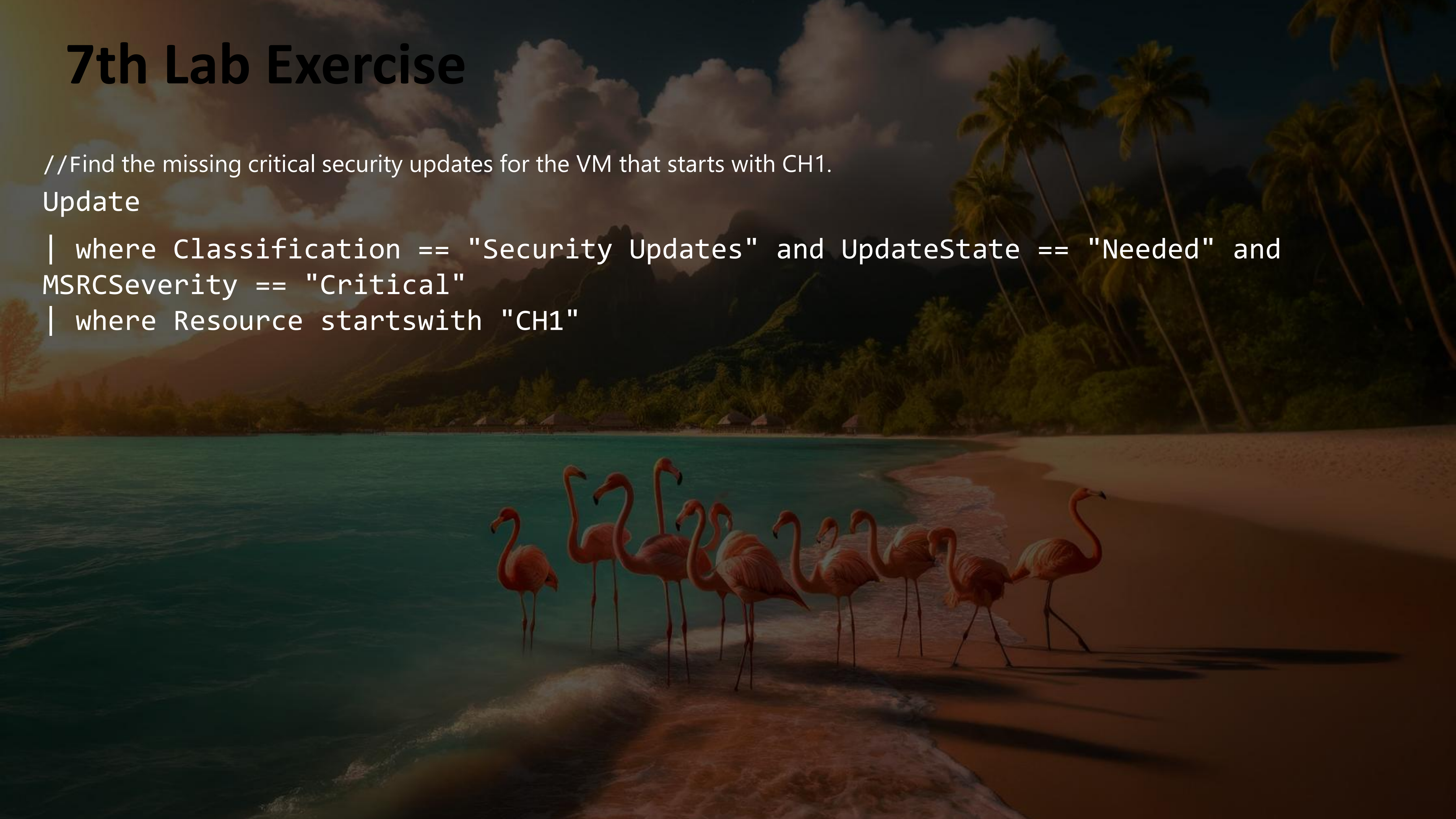
Your company has hired a Pentester to audit your security environment, the results of the test determined that missing patches have resulted in a successful attack against your systems.

You need to find the missing critical security updates for the VM that starts with CH1.



Hints and guideline:

- Use the Updates table to begin your hunt.
- How would you find patch information?



7th Lab Exercise

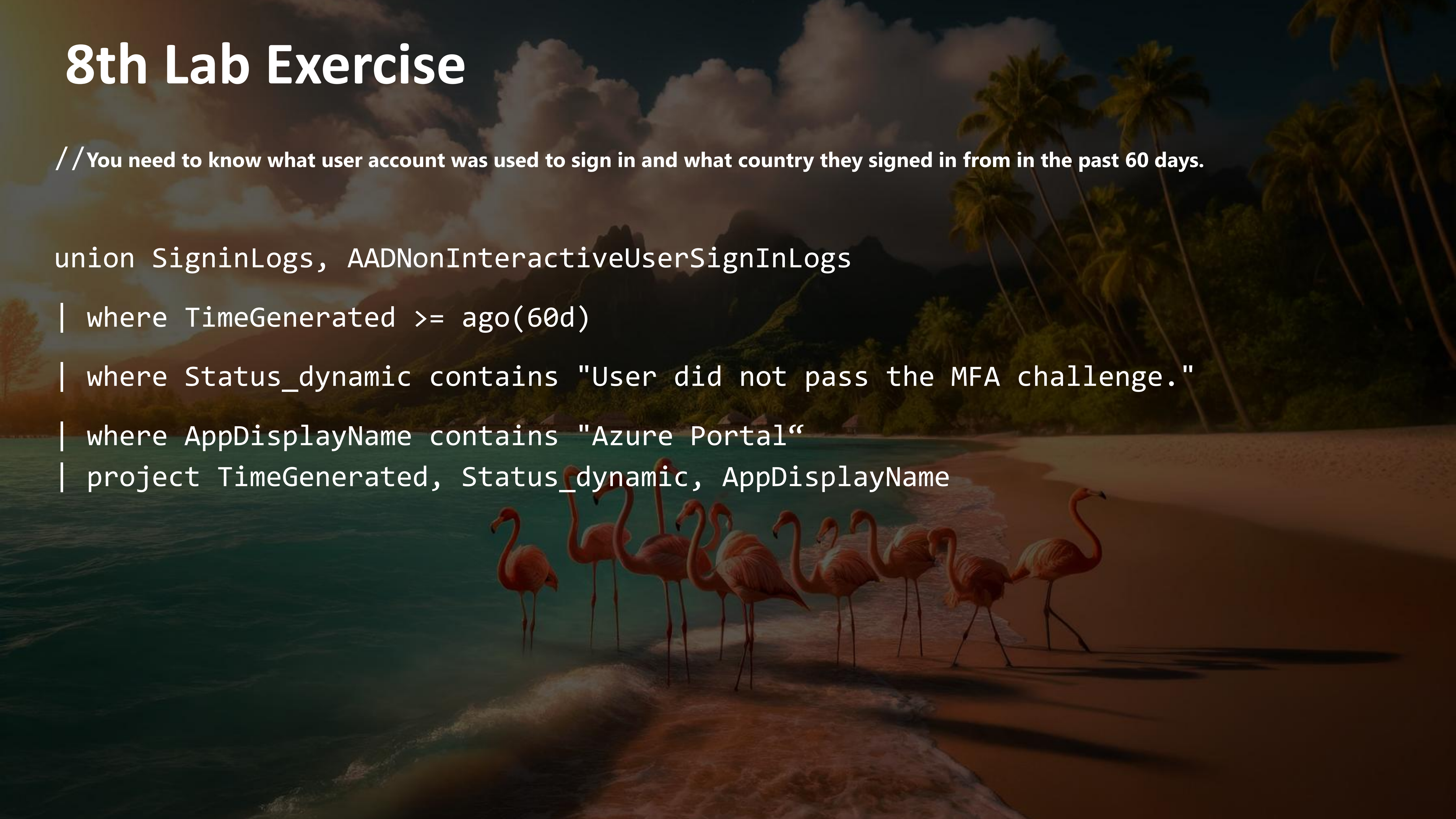
```
//Find the missing critical security updates for the VM that starts with CH1.
Update
| where Classification == "Security Updates" and UpdateState == "Needed" and
MSRCSeverity == "Critical"
| where Resource startswith "CH1"
```

8th Scenario

Your Azure Environment has suffered an attack with multiple brute force attacks. No compromise took place; however, you need to know what user account was used and what country the attacker attempted to log in from. The results should only show the timestamp, Status, Username and the app name.

Hints and guideline:

- This will involve more than one table.
- The company did have MFA enabled.
- Limit your search to the past 60 days.



8th Lab Exercise

//You need to know what user account was used to sign in and what country they signed in from in the past 60 days.

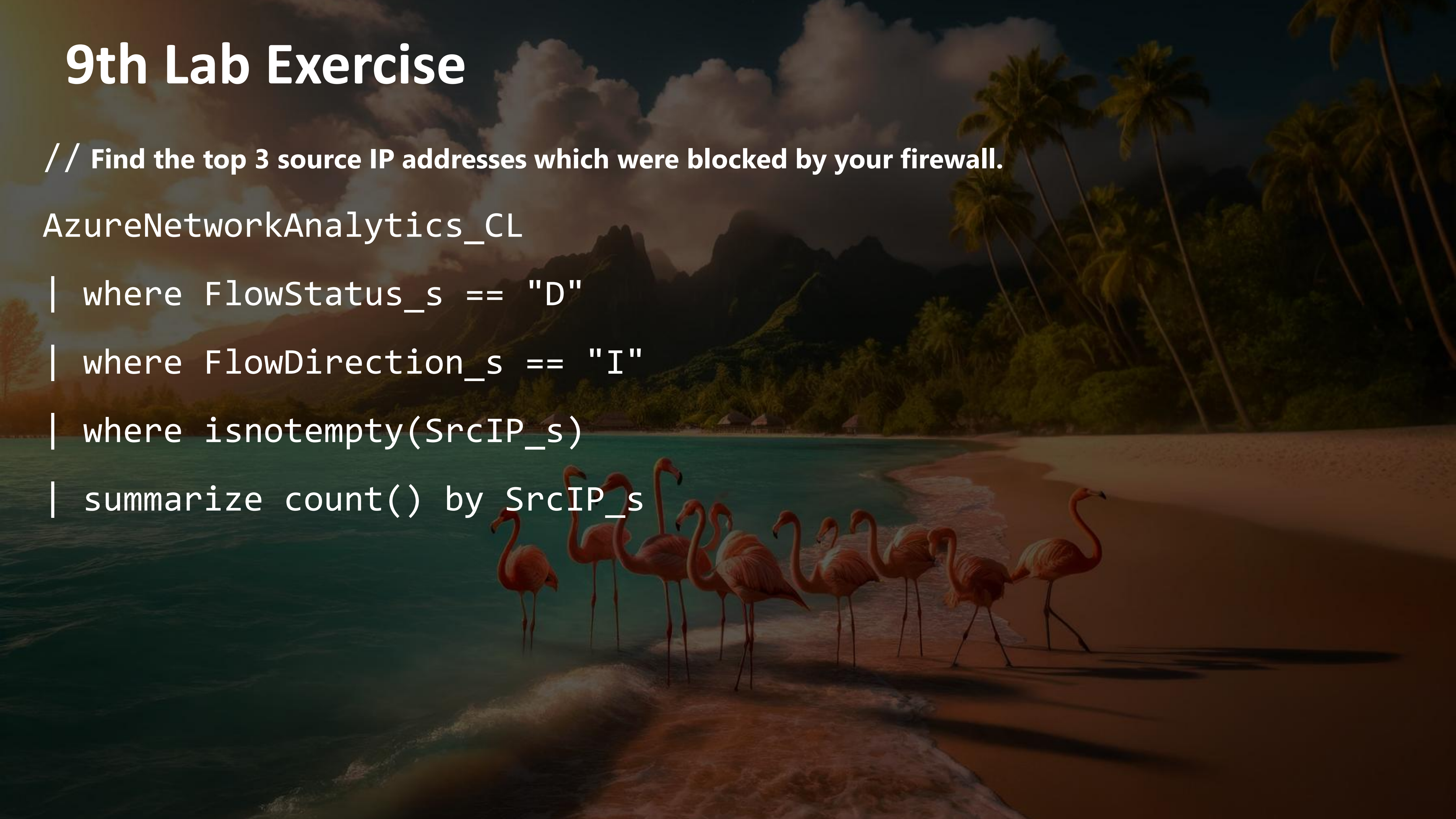
```
union SigninLogs, AADNonInteractiveUserSignInLogs

| where TimeGenerated >= ago(60d)

| where Status_dynamic contains "User did not pass the MFA challenge."

| where AppDisplayName contains "Azure Portal“
| project TimeGenerated, Status_dynamic, AppDisplayName
```

9th Scenario

Your Azure Environment has successfully defended an attack from an outside entity. As a part of your IR Report, you need to find the top 3 source IP addresses which were blocked by your firewall.

Hints and guideline:

- Which table would you use?
- What way does the data “flow” ?



9th Lab Exercise

```
// Find the top 3 source IP addresses which were blocked by your firewall.

AzureNetworkAnalytics_CL

| where FlowStatus_s == "D"

| where FlowDirection_s == "I"

| where isnotempty(SrcIP_s)

| summarize count() by SrcIP_s
```

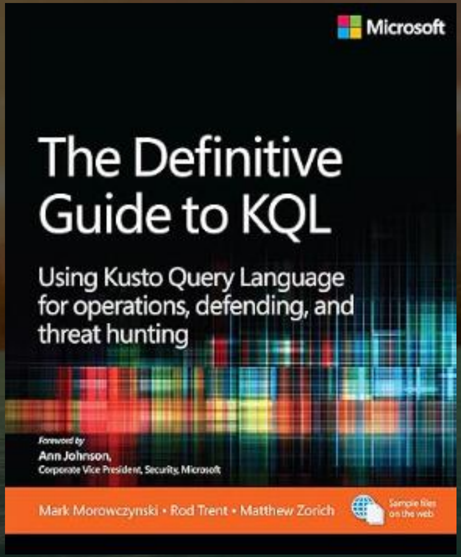
Do it yourself!

aka.ms/LADemo

aka.ms/MMSKQL

(Or using the CybersecuritySOC demo)

Go Deeper!



Microsoft

The Definitive Guide to KQL

Using Kusto Query Language for operations, defending, and threat hunting

Foreword by Ann Johnson, Corporate Vice President, Security, Microsoft

Mark Morowczynski • Rod Trent • Matthew Zorich

Sample files on the web

https://aka.ms/KQLMSPress

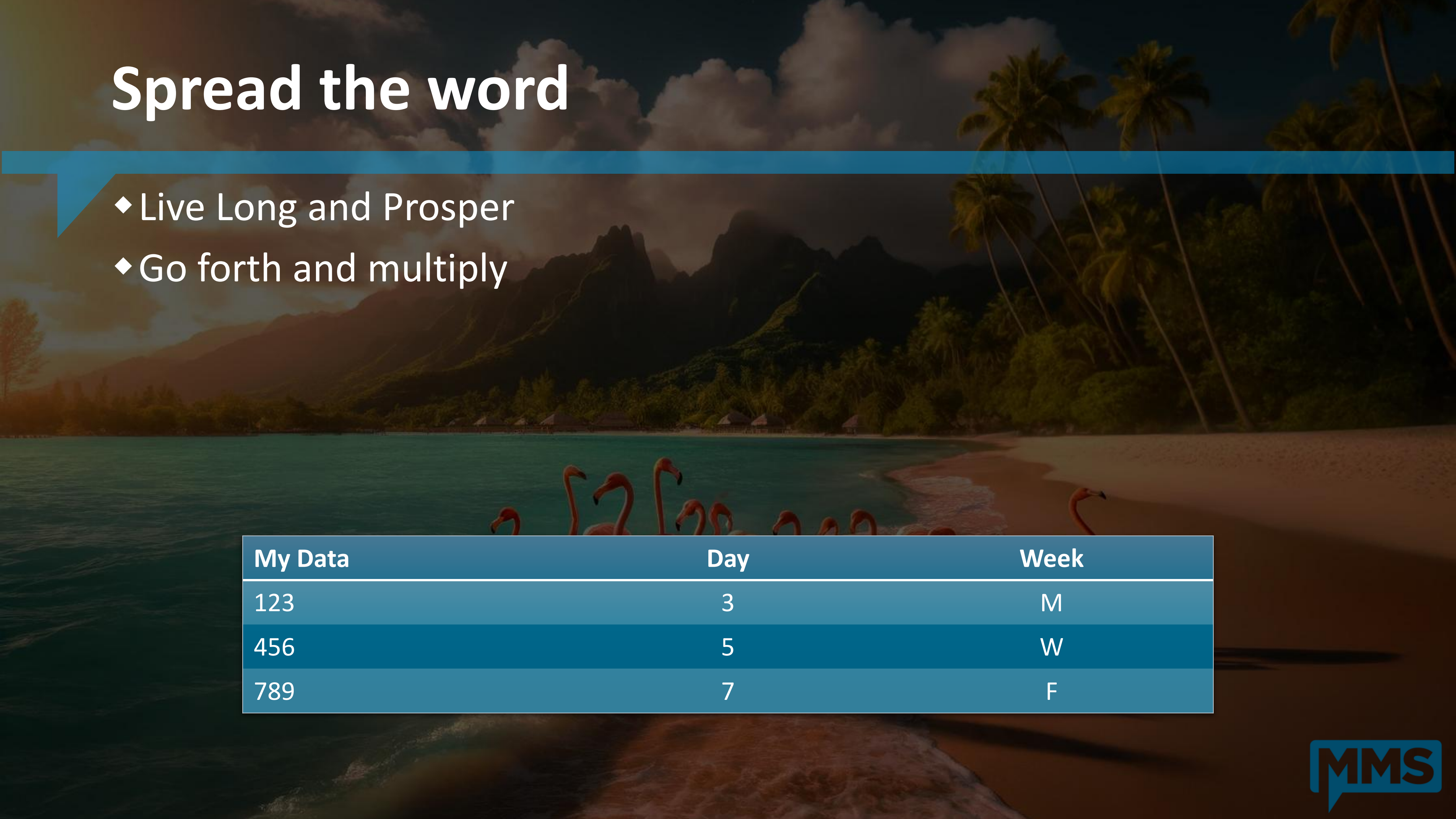
MUST LEARN KQL

THE SERIES

Empower every person and every organization on the planet to learn KQL

ROD TRENT
CLOUD SECURITY ADVOCATE

https://aka.ms/mustlearnkql



Spread the word

- Live Long and Prosper
- Go forth and multiply

My Data	Day	Week
123	3	M
456	5	W
789	7	F

Save the Dates

May 4-8, 2025

Oct 12-15, 2025

May 3-7, 2026

Extended Q&A

PATCH MY PC

ninjaOne

2Pint

RECAST SOFTWARE

1E

eido

ZERO TOUCH

VIAMONSTRA ONLINE ACADEMY

systemcenterdudes

MMS 10 YEARS