Kusto Query Language

KQL- Used to query Azure log databases: Azure Monitor Logs, Azure Monitor Application Insights/ Log analytics.

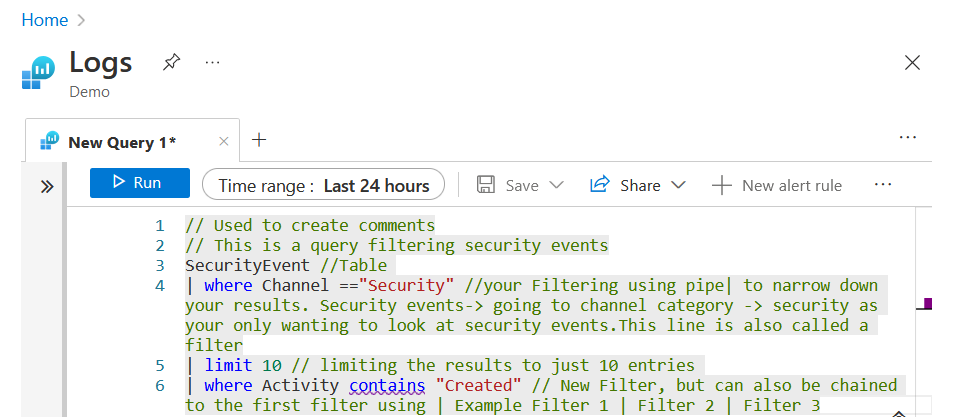
Log Analytics = Log Database and pricing depends on how large the database is and how long your wanting to store it for. Note: You are only wanting to configure Log Analytics with the logs your going to be storing, Collecting, and Analyzing on a daily basis as again it effects your pricing and you don’t want to be waste less in spending.

For a Log analytics Demo go to <https://aka.ms/lademo>. Make sure you have a have an active azure account/subscription

on in order to view it.

Tips: When your creating your query, you can either hit run or just hit shift + enter to execute the query.

Lab is from Teachjing YouTube video link is under resources.



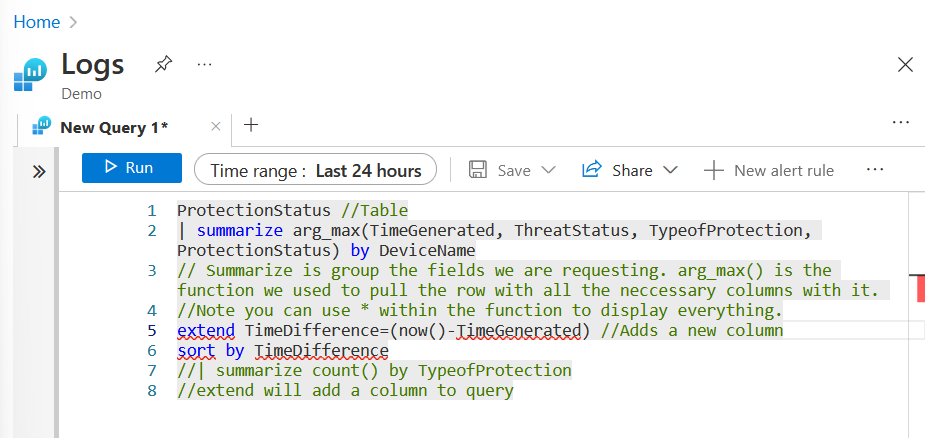
ISP had a outage and didn’t want to redo the above, but to filter for how many different type of events, or objects you use the **distinct** function. Line 7: | distinct Activity // This is going to filter the category Activity for all the different type of event alerts discovered.

To count how many events there is for each different event alert. you have to use the summarize function.

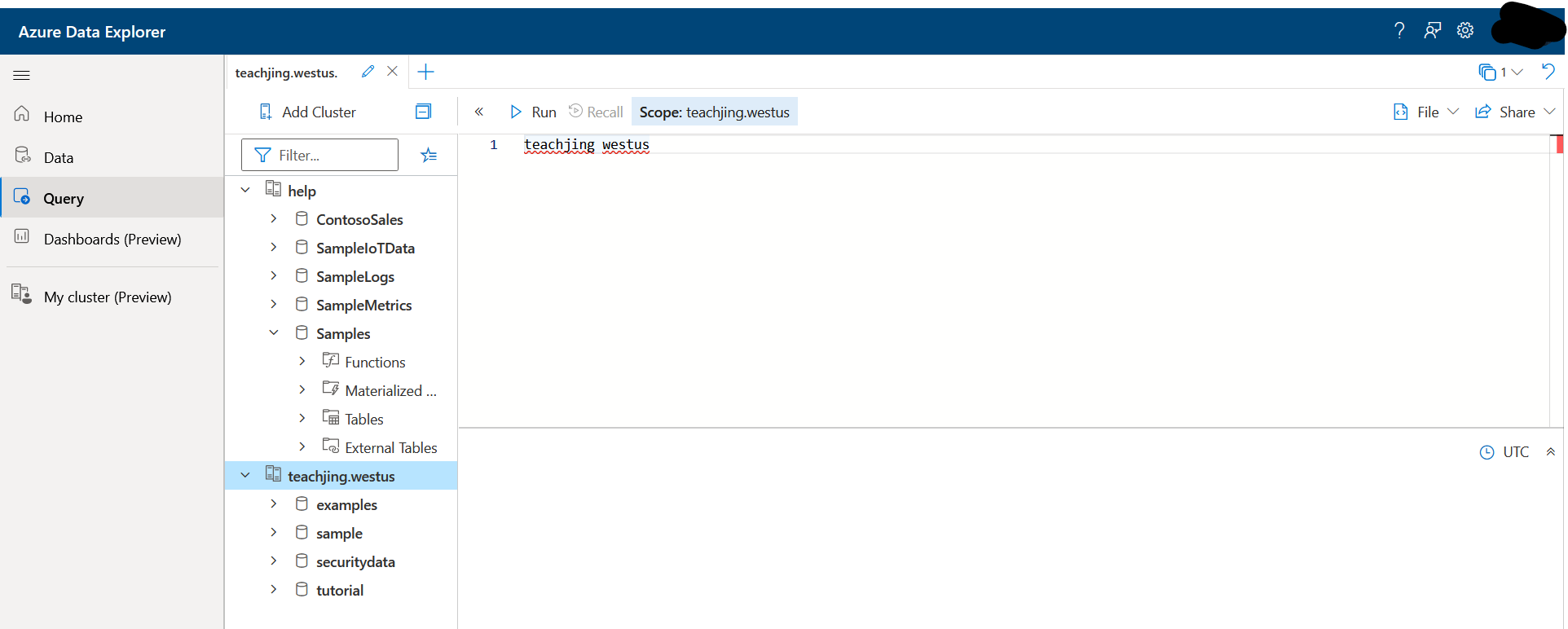
line 8: | Summarize Count() by Activity

Example: I was told I need to Analyze the protectionstatus and create a report. The report should list the machines with the must current time generated showing the following fields: Threat Status, Protection Status and Type of Protection.

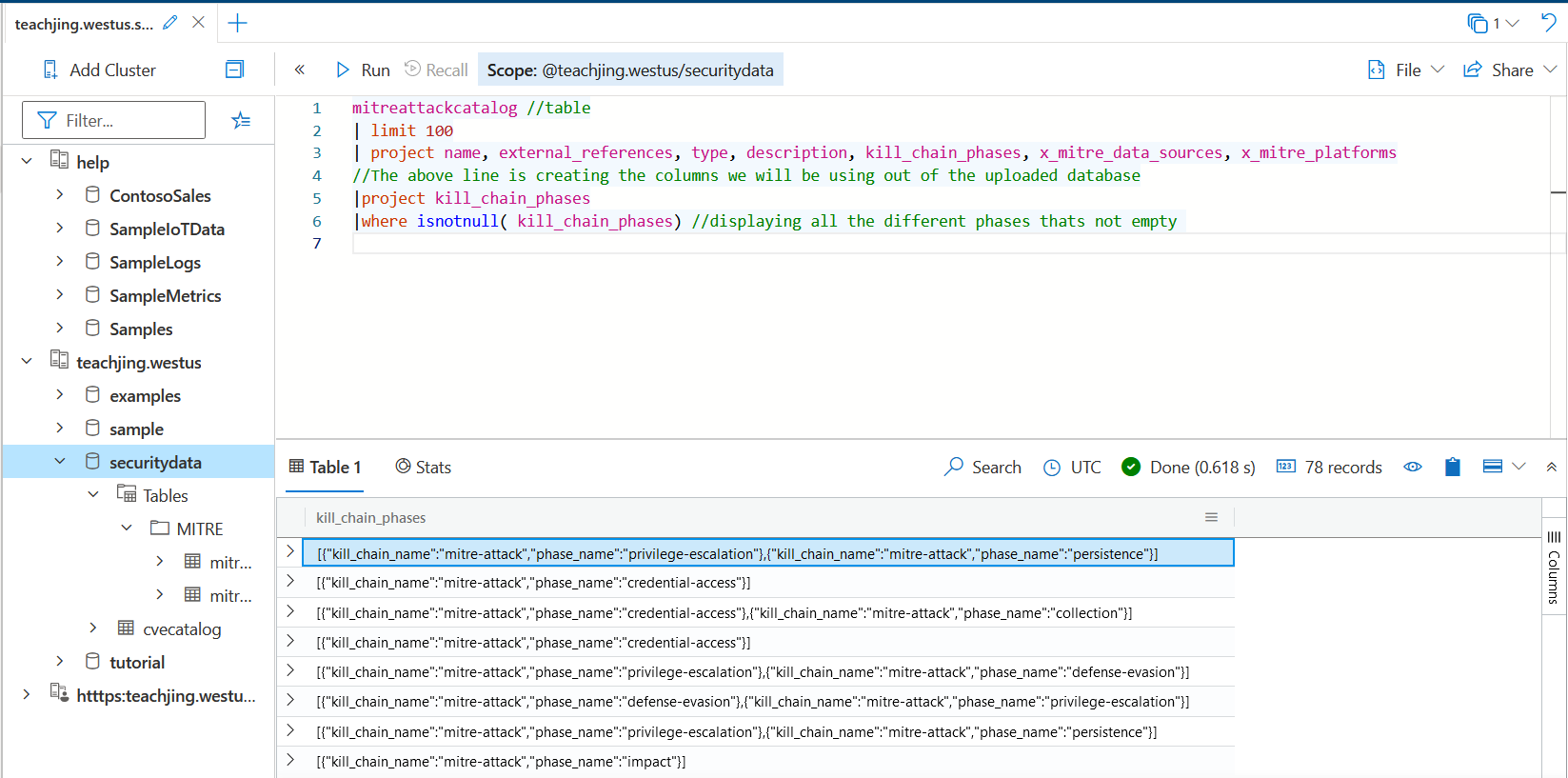
Note: KQL Operator documentation will be vital for this scenario if you’re a beginner. <https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/>



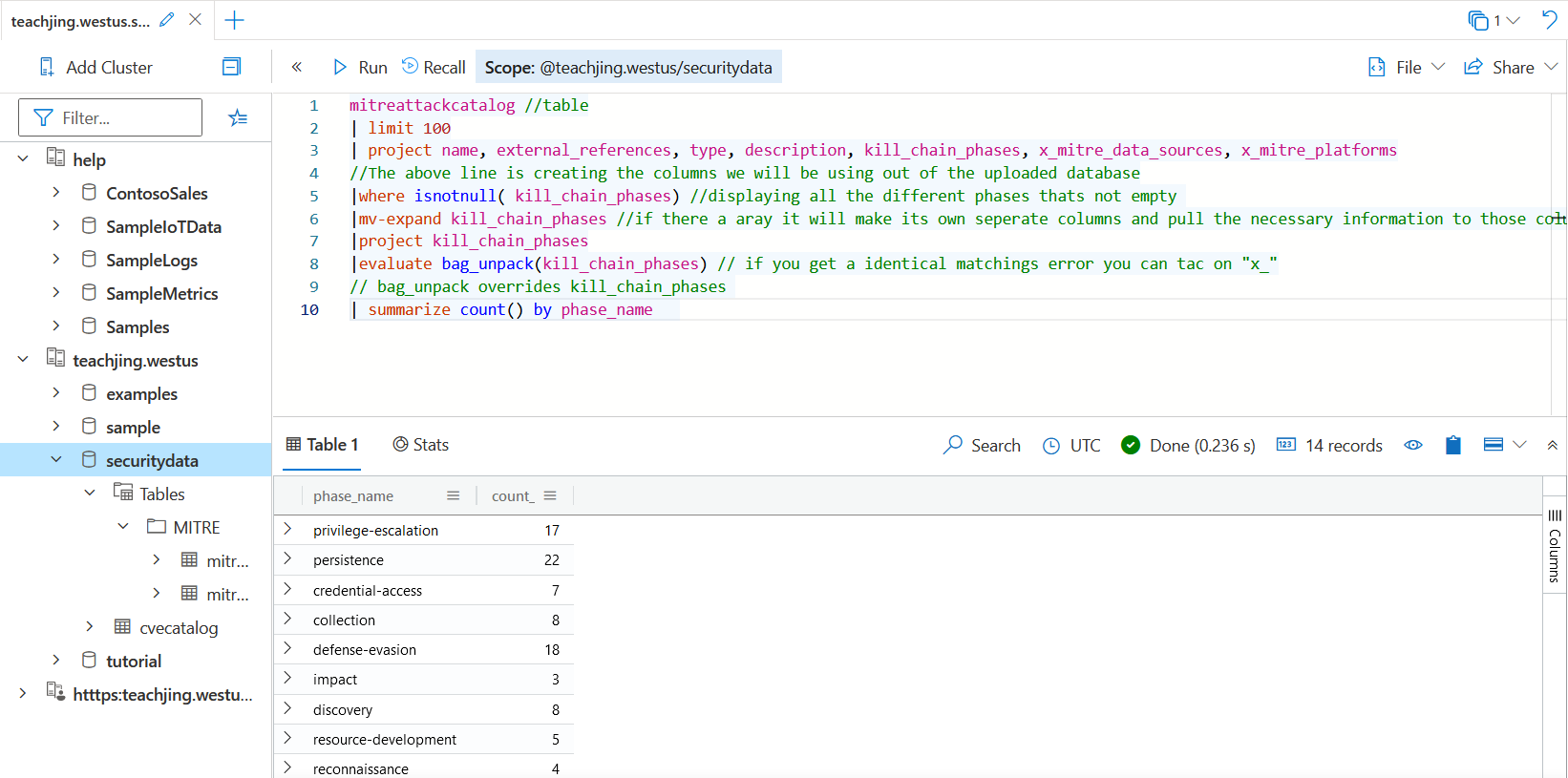
Taking one step further, I am going to make a MITRE dashboard. Start off to Azure Data Explorer <https://dataexplorer.azure.com/clusters/help/databases/Samples>

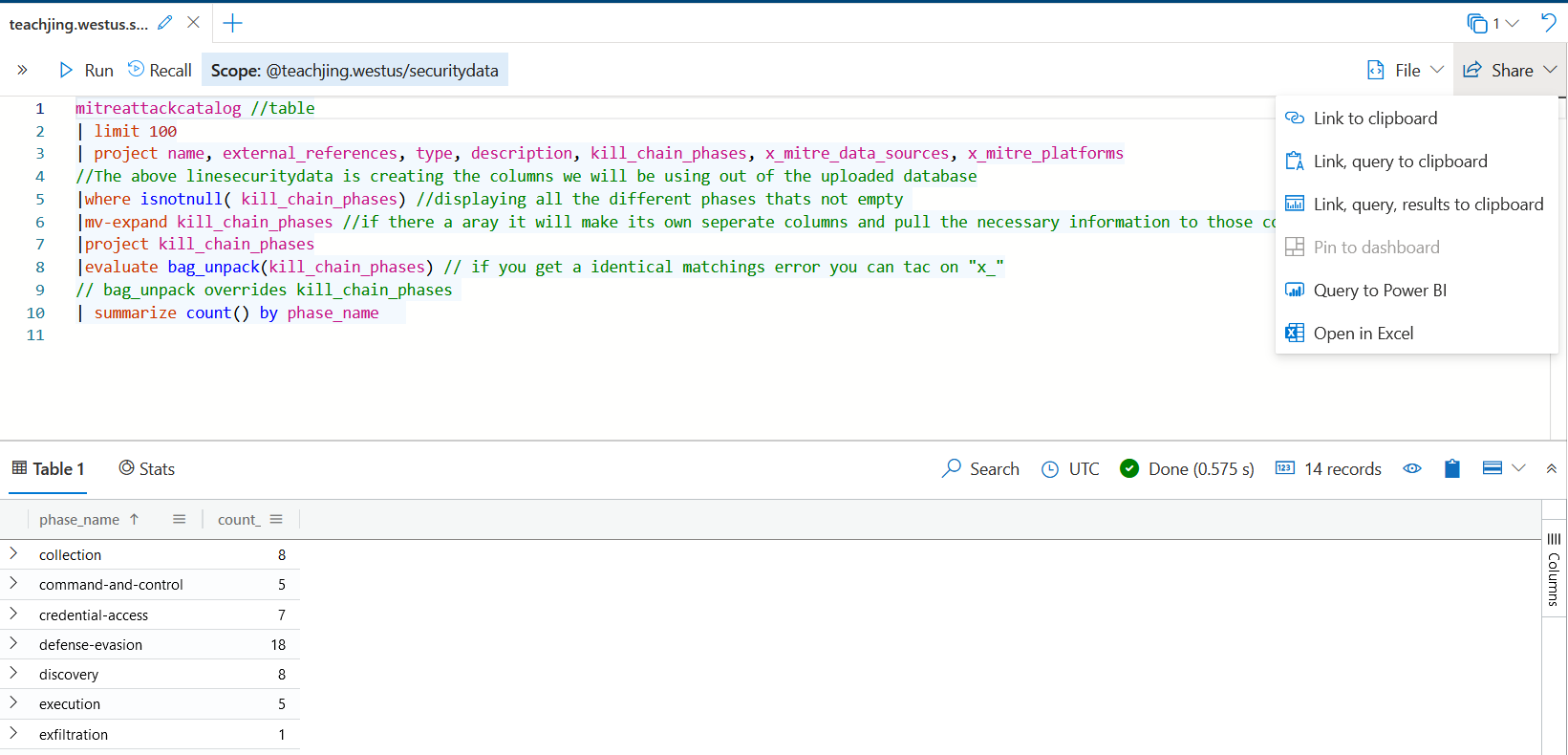


I had to connect to the cluster that Im using for this training lab. I selected Add Cluster --> URL <https://teachjing.us.kusto.windows.net> --> display name is teachjing.westus

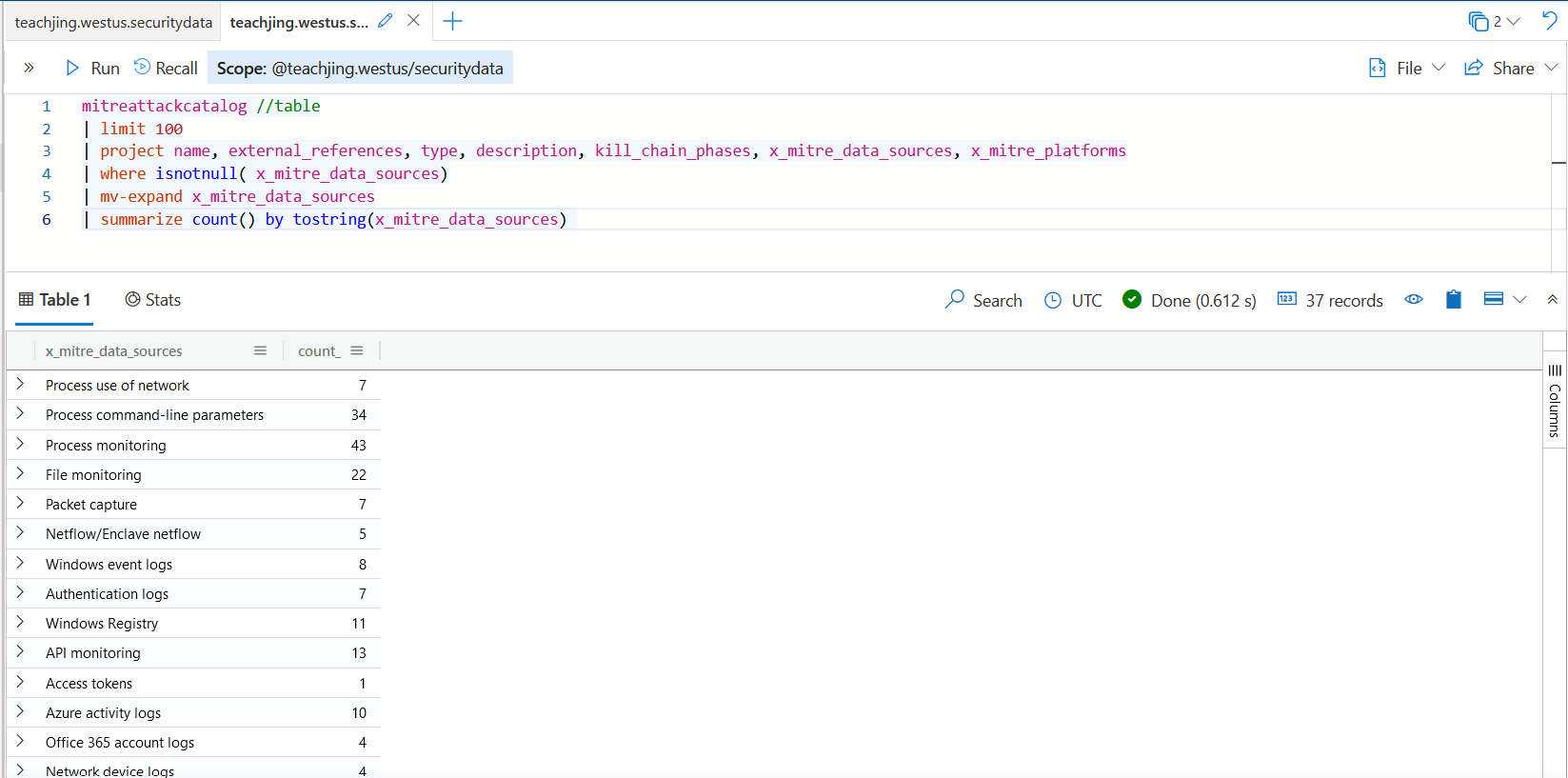


I’m beginning to create my GUI dashboard. This is the first step of getting to the graphical visual charts. I’m currently configuring my columns and analyzing my output ensuring that I’m not making a mistake. To read the output its listing the category and then a subcategory for example: Privilege-Escalation --> persistence

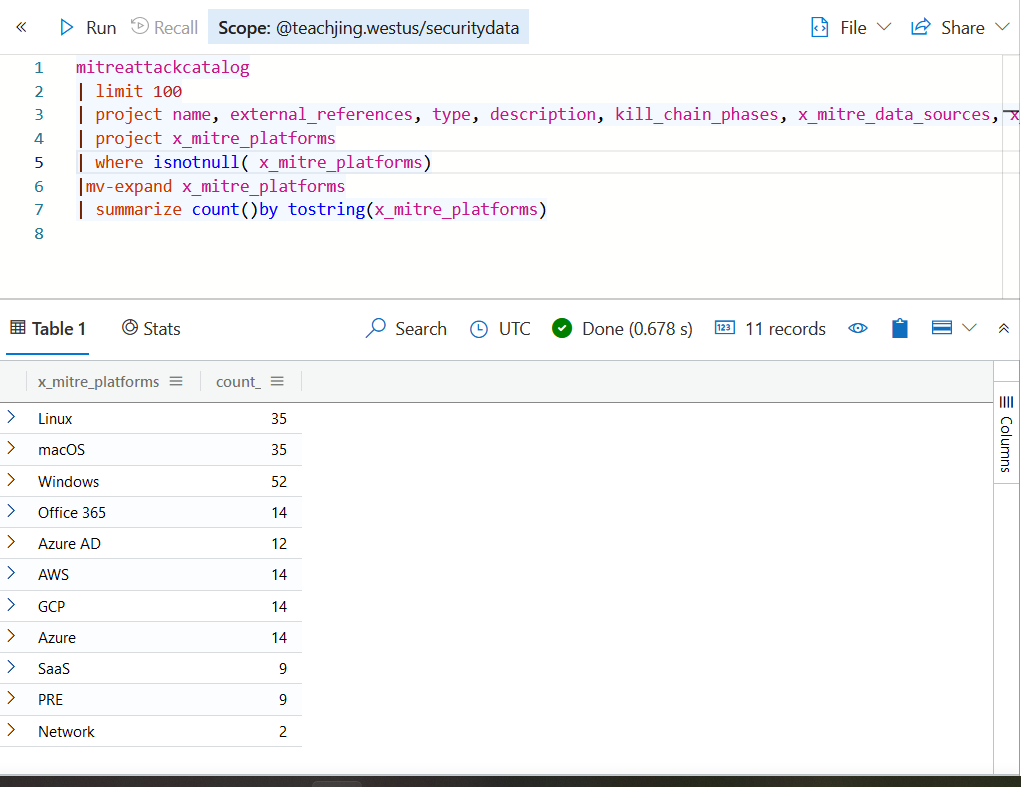




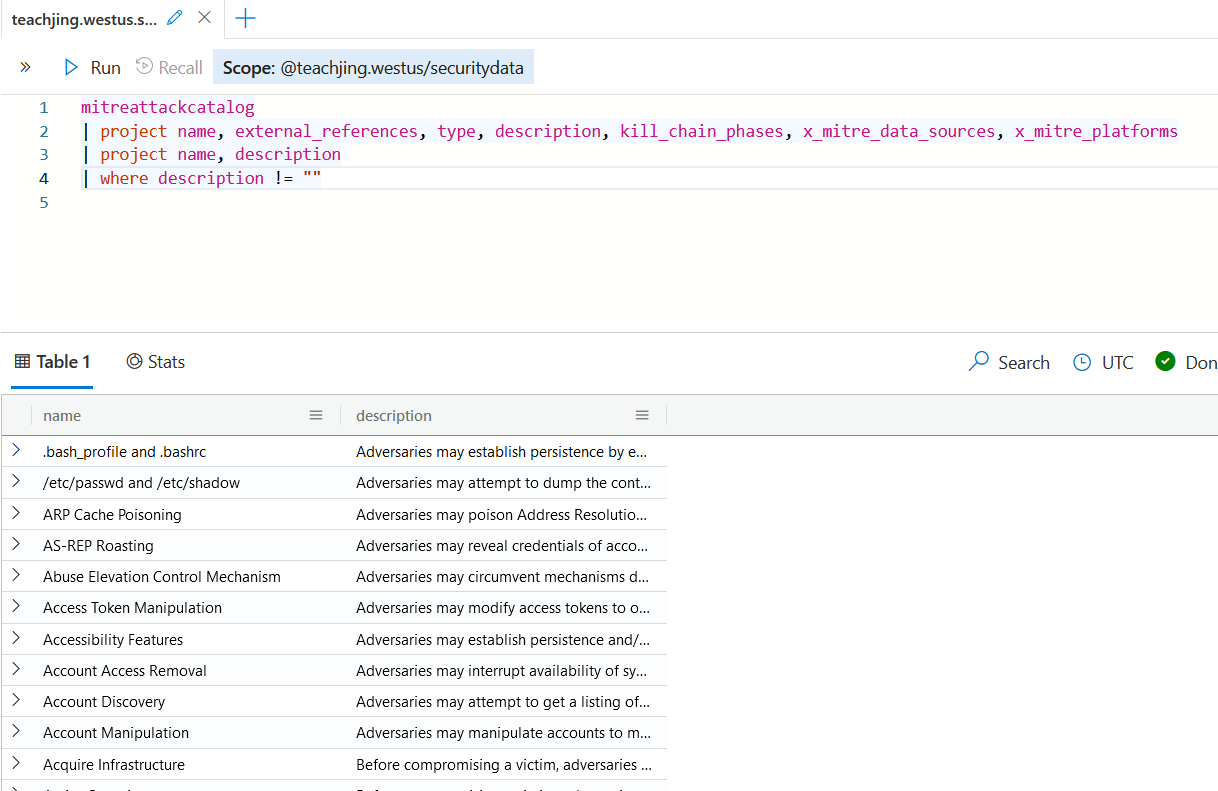
I attempted to Export /Pin query to dashboard like training video stated to do, but for some reason it won’t allow me to. After spending 30 minutes troubleshooting this issue, I haven’t been able to find documentation on to why it won’t grant me that option. I suspect it is due to permissions that the owner of this lab query has in place. If I was allowed to execute the pin to dashboard option, the dashboard will create and you can see from a visual perspective of what’s occurring in your network.



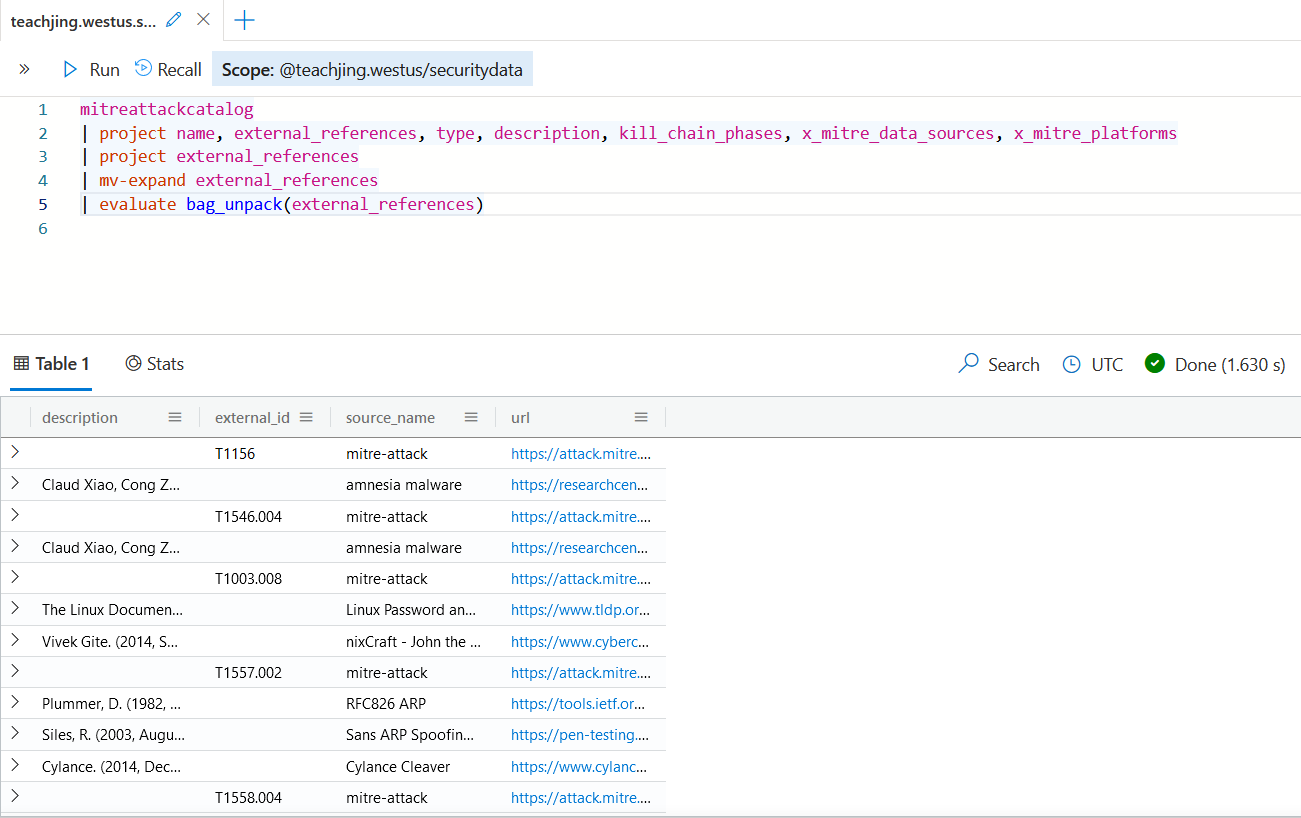
This query is displaying the different data sources. Still can’t pin to dashboard. If I had the permission to continue with the motion in generating the dashboard all that would be left to do is to edit the dashboard selecting what type of chart I want to use and labeling then save it.



Platform Dashboard

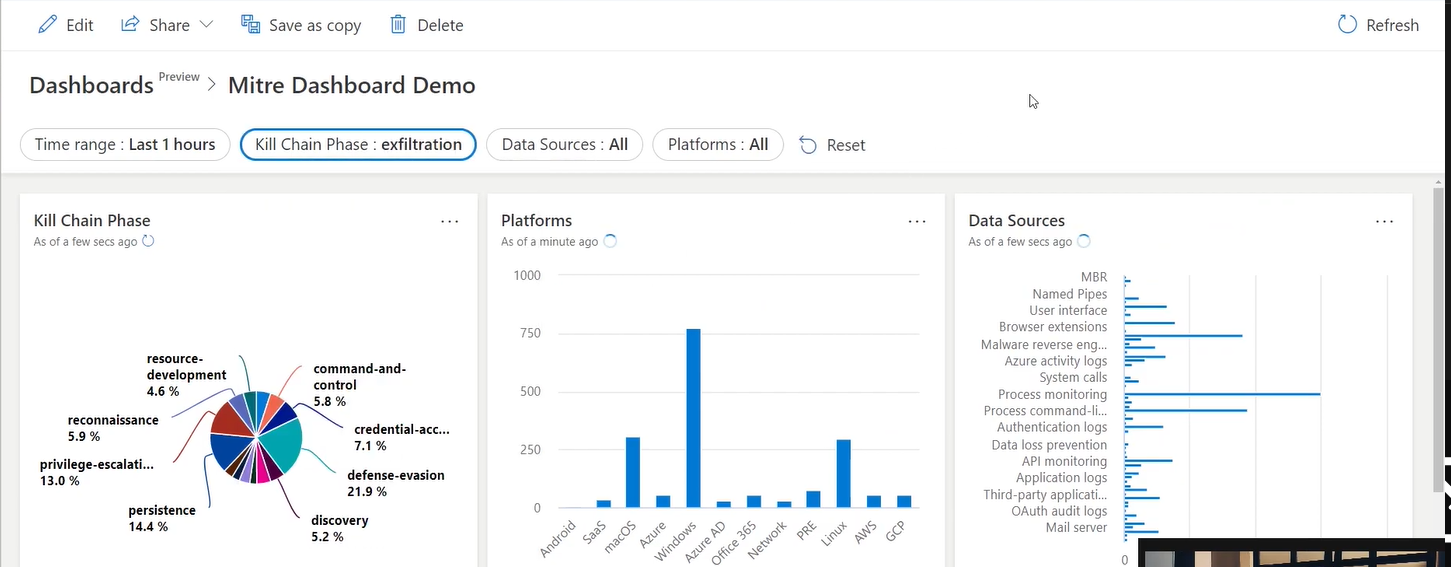


Attack Dashboard



External Links Dashboard

Note: To filter your dashboards you have to go into your dashboard, Select Parameters --> New Parameter. In the fields just fill it out using the query created above and hit done



This is a snippet from the YouTube training video of what my dashboard would have looked like if Azure gave me the option to pin to dashboard.

This portion I will be working with Joins.

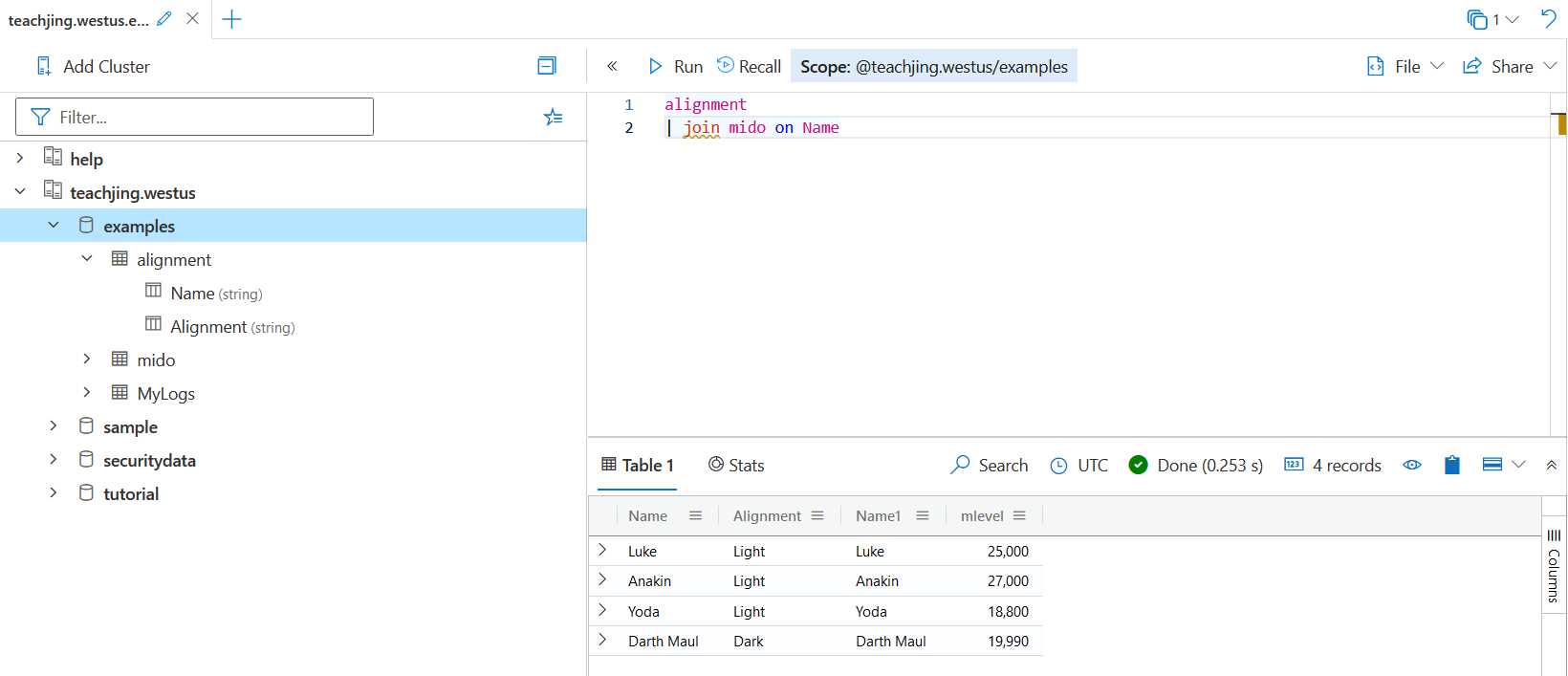
Inner unique:

- Default Join Flavor with left side deduplication. It gets all records, but ignores multiple records on the key value residing on the left table

- Example: Table1 | join(Table2) on CommonColumn :

Name: Mike | Vehicle Type:Car, Name:Jim | Vehicle Type: Truck |||| Name: Mike | Miles: 50k | Name Jim | Miles = 75k . == Name Mike, Jim | Vehicle Car, Truck | Miles: 50k, 75k

YouTube GUI Script Example:

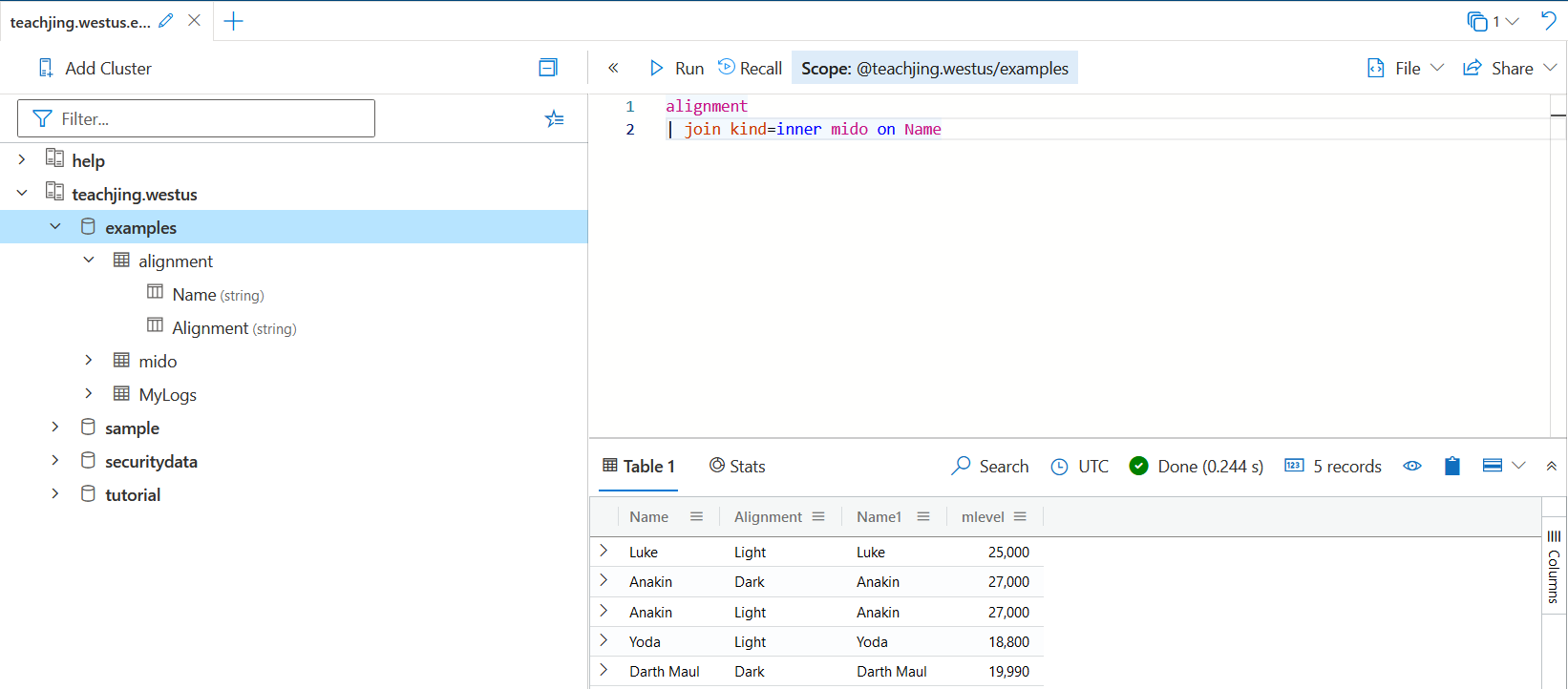


Inner:

- Gets all Records where the common column matches

- From example above Jim would also appear twice one as shown below, but another copy of Jim. other words mike will show he has the same properties as Jim.

YouTube Script GUI Example:

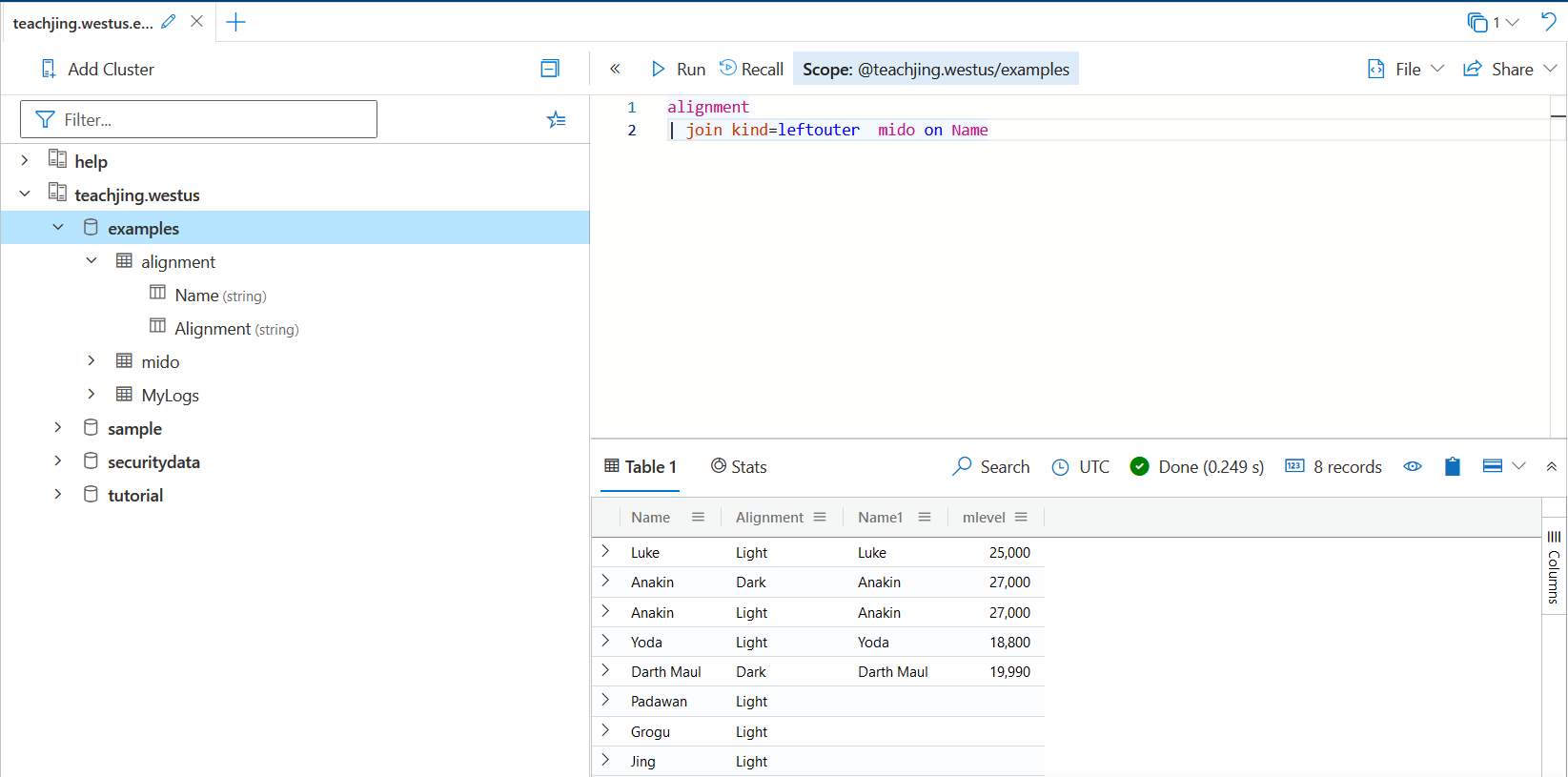


Outer:

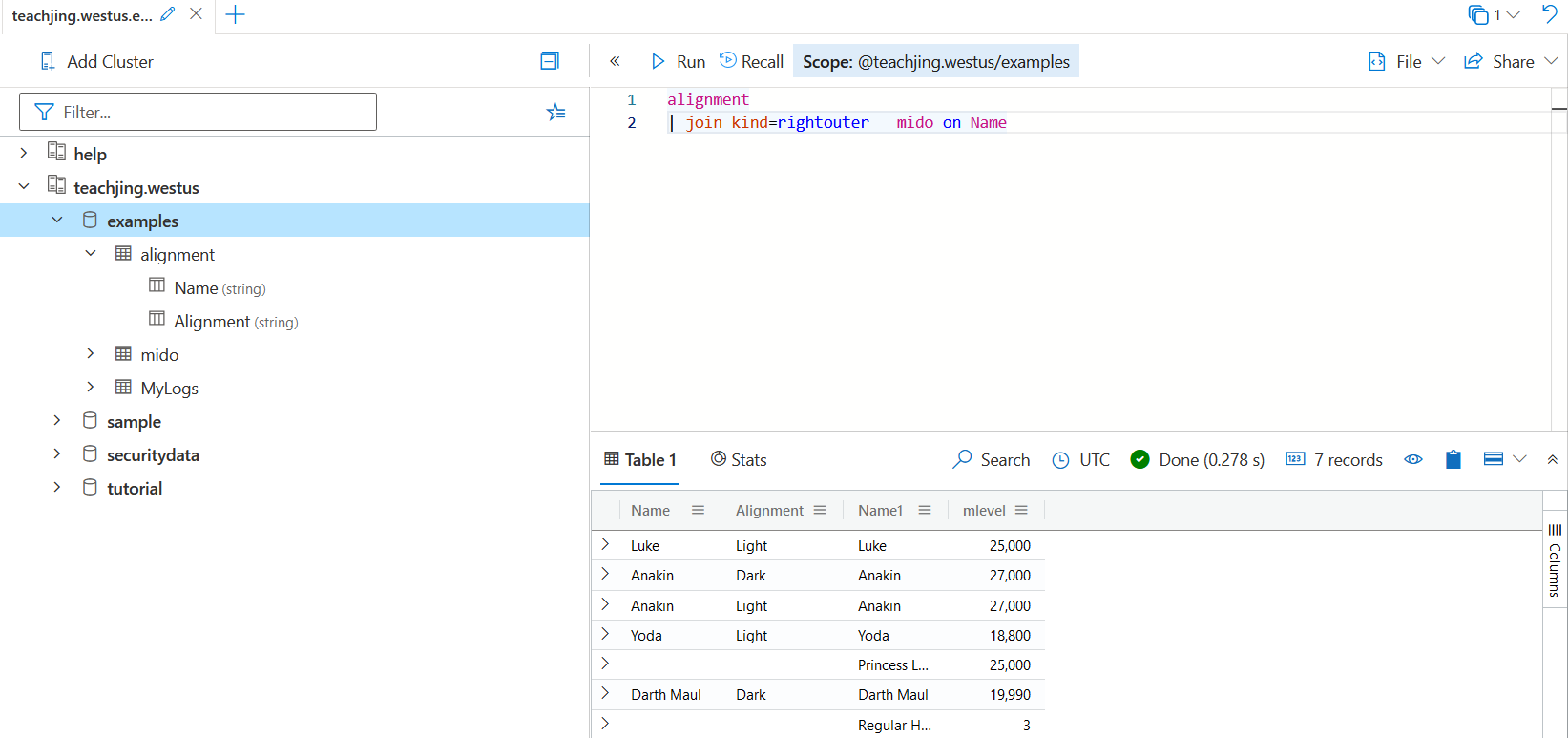
- Results of outer join will ALWAYS contain all records of the defended kind even if the join condition doesn’t find any matching records.

YouTube Script Gui Example:

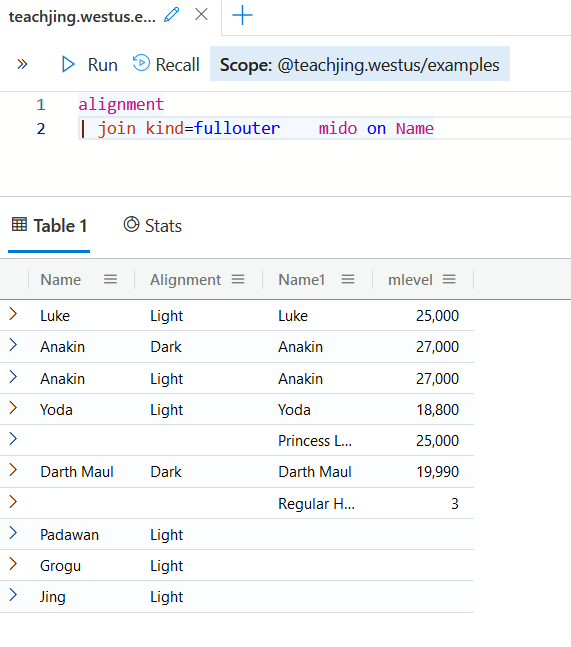
Left outer:



Right outer:



Full outer:



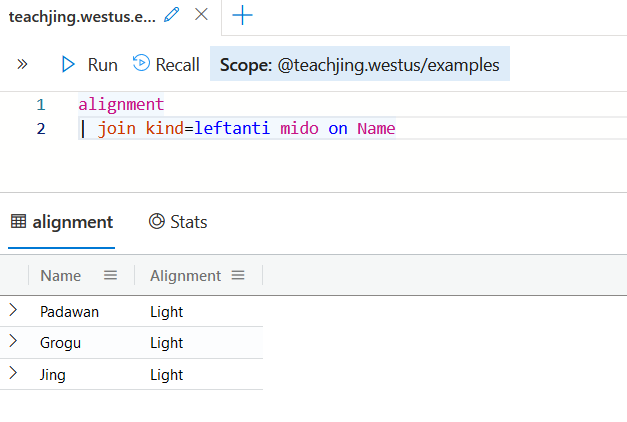
-------------------------------------------

Anti-Join:

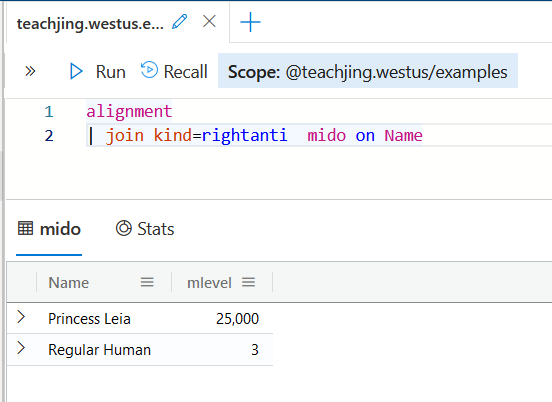
- Returns all records from the defined kind that doesn’t match any record from the other side.

- Looking from object linkage that’s not the same that will display.

Left-Anti:



Right-Anti:

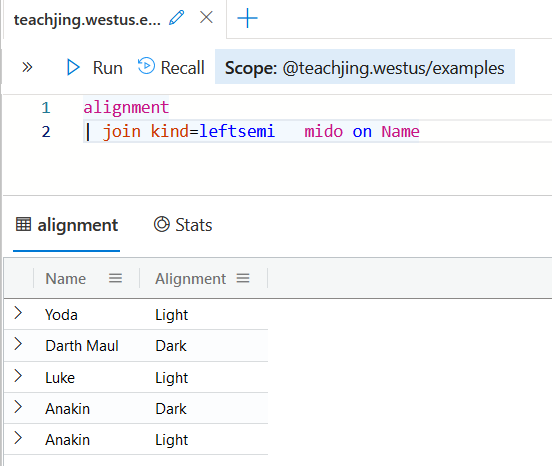


----------------------------------------------

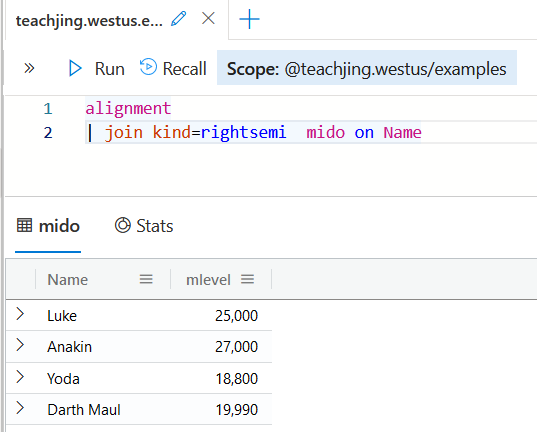
Semi:

- Returns all records from the defined kind matching a record from the other side. Only the columns from the defined kind are returned

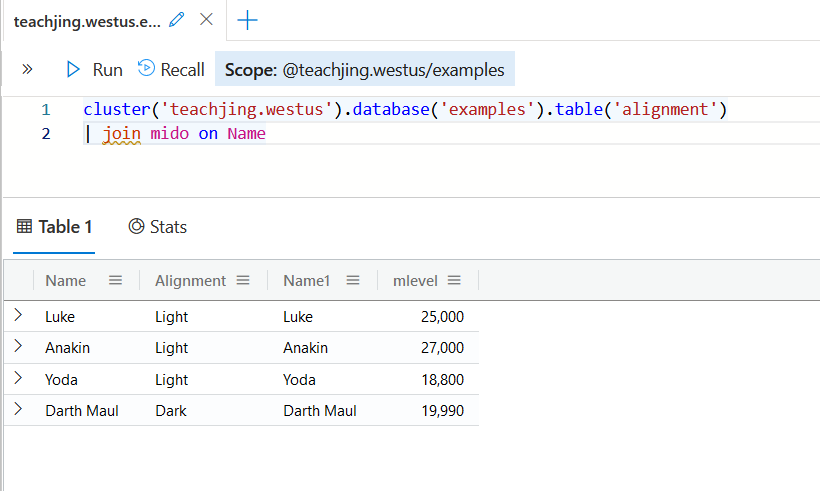
Left-Semi:



Right-Semi:



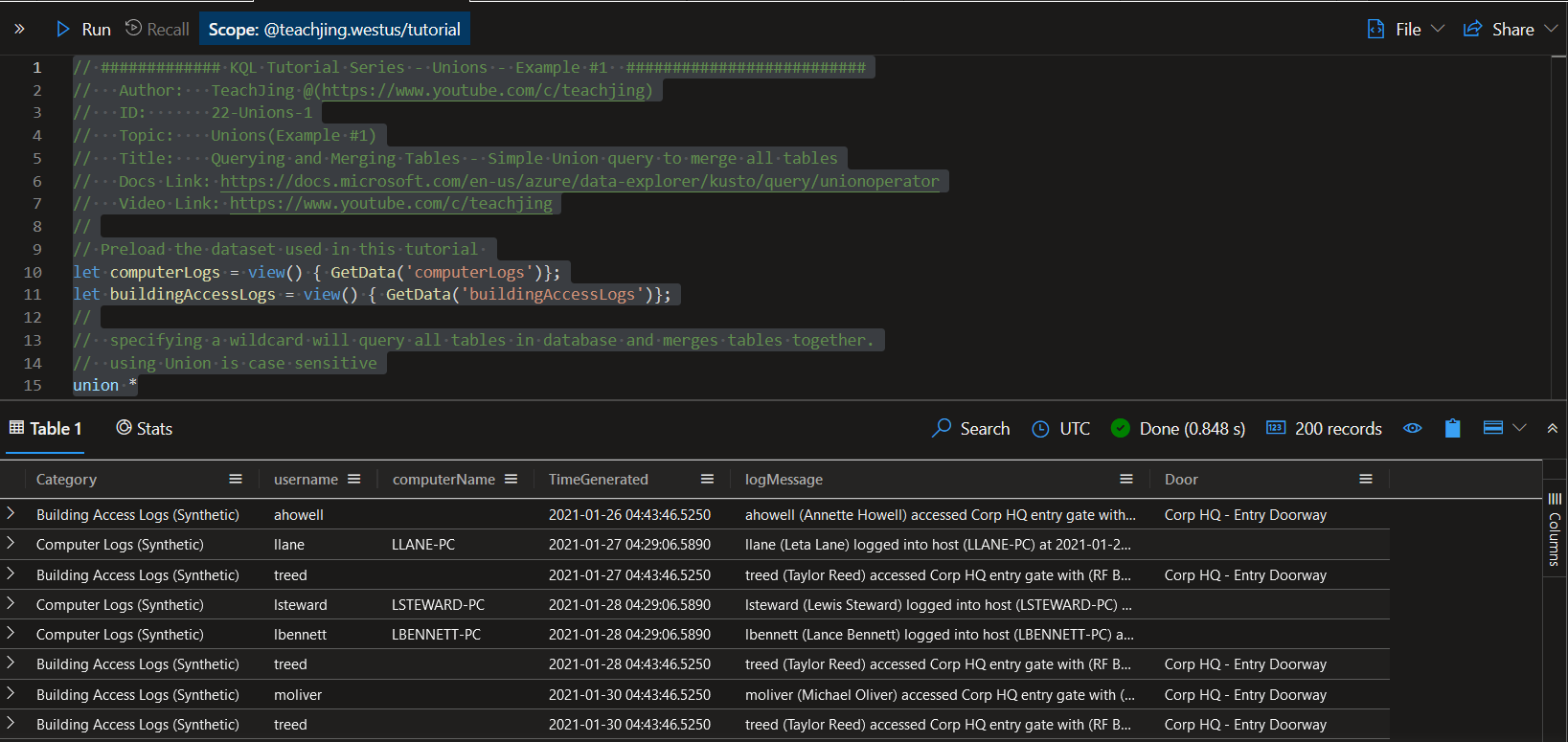
Other:



Here we are joining tables across clusters. first you have the specify the cluster that’s holding your database then you would call the table where you would join with a table from another cluster.

Unions and Lab

Unions take data from two or more tables and displays the results together. This from learning about joins sounds the same, but the difference is that join produces specific results through joining rows of just two tables using matching of the values of specified columns.



So, what we are looking at here is a pre-defined sample dataset that is using the Union operator. As mention unions combines 2 or more tables together and outputs the results. \* is being used in the most general sense of the definition of union as the results are pulling together the table information from computerLogs and buildingAccessLogs and adding it together. Unions are case sensitive, therefore, “L” in Logs must be capitalized or a error will generate.

Using the above with slight changes for the next few examples:

If we are using Syntax: union \*Logs.

// This combines all the tables that ends with the word log. The wildcard "\*" indicates all characters before log are acceptable to include.

// Your searching for all words ending with the word "Logs among the two tables: computerLogs and buildingAccessLogs"

---------------------------------------------------------------------

// This union query will directly specify which tables you want to query. In this case its computerLogs and buildingAccessLogs

Syntax: union computerLogs, buildingAccessLogs

---------------------------------------------------------------------

// This union query will use the "isfuzzy=true" parameter to ignore tables that don't exist or result in error.

//In this case randomLog and anotherLog actually don’t exist and is ignored when searching the different tables to pull the data together.

Syntax: union isfuzzy=true computerLogs, buildingAccessLogs, randomLog, anotherLog

------------------------------------------------------------------------

// This query will search in all the tables where any row contains the word "Synthetic" across all columns.

Syntax: union \*

| where \* contains "Synthetic"

Note: This is just like early labs where we are using “where” tells the search exactly what to look for. In this case, we are simply looking for all the logs among the two tables that includes the word “Synthetic”

Note: You can also specify which columns. Example: Where username contains “Tyler”

---------------------------------------------------------------------------

// This query will search all tables. "kind=inner" ensures only similar columns(Columns thats in both tables only) are outputted in the results. "withsource" adds another column that specifies what table the row entry came originated from (Helpful to locate what table)

Syntax: union withsource=SourceTable kind=inner \*

Note: You are assigning the source of the pull data to a variable called withsource and displaying the withsource column containing the pulled data.

// Uncomment this to try an outer kind

Syntax: union withsource=SourceTable kind=outer \*

Note: Using Outer will include all the log columns from both tables regardless if its matching or not.

KQL Functions

Function: Set of instructions bundled together to achieve a specific outcome. Functions are good ways of having repeating blocks of code in a program.

Types Of Functions: Built-in, Local (Query-defined) , and Global Functions (Stored)

Categories:

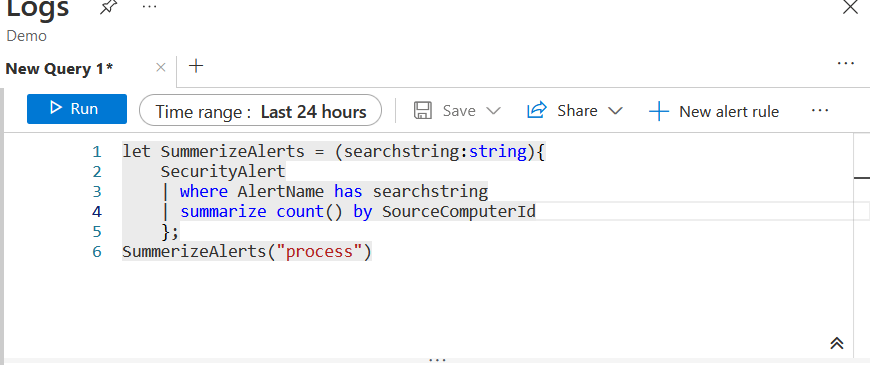
- Scalar Functions Note: In respect to KQL think of singluar output)

- let players = (playerstoadd: word) { playerstoadd + 10};

- Tabular Functions Note: output will be in the form of a table

Note: in order to save functions in Azure you can’t be in a test environment.

Examples:



Here I created a function summarizing the security alerts. first, I summarized all to get all the alerts that contains the word process, but that still didn’t narrow it down. Therefore, I added Line 4 | summarize count () by SourceComputerId. This changes the function to continue looking for logs that contains the word “process”, but now to include which machines and the number of logs that includes the word “process”

Resources

KQL Cheat Sheet: [KQL/kql\_cheat\_sheet.pdf at master · marcusbakker/KQL · GitHub](https://github.com/marcusbakker/KQL/blob/master/kql_cheat_sheet.pdf)

Teachjing YouTube Channel: [KQL Tutorial Series - YouTube](https://www.youtube.com/playlist?list=PLM3TOIlrnaI4hwmXTxrYGE665q-9fyTfB)

KQL Functions: <https://docs.microsoft.com/en-us/azure/data-explorer/kusto/query/functions/user-defined-functions>