# Lab 05 - SQL DW Management and Monitoring

Contents

[Lab 05 - SQL DW Management and Monitoring 1](#_Toc18905517)

[Overview 1](#_Toc18905518)

[Pre-requisites 1](#_Toc18905519)

[Resource Monitoring in Azure Monitor 1](#_Toc18905520)

[Azure Data Studio SQLDW Dashboard (Azure SQL Data Warehouse Insights) 2](#_Toc18905521)

[Azure SQL Data Warehouse Table and Statistics Queries 3](#_Toc18905522)

[Create User-defined Restore Points 8](#_Toc18905523)

[Maintenance Window Scheduling, Service Health, Service Health Alerts 14](#_Toc18905524)

[Querying ADW Diagnostic Logs using Azure Monitor 18](#_Toc18905525)

[If you did not configure Diagnostics earlier: 19](#_Toc18905526)

[Review Diagnostics Logs: 21](#_Toc18905527)

## Overview

This module will walk you through a variety of tools and techniques for monitoring and managing your Azure Data Warehouse.

## Pre-requisites

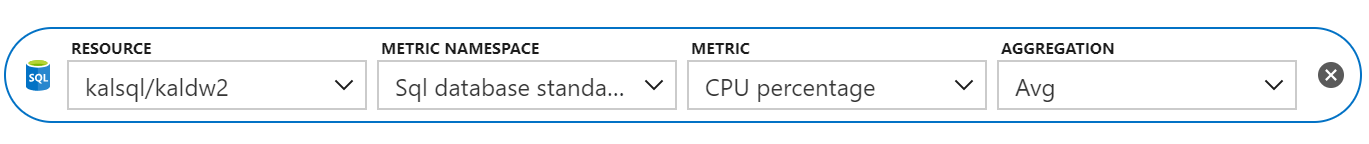
* Azure Portal Access
* Azure SQL Data Warehouse
* SQL Server Management Studio
* Azure Data Studio

## Resource Monitoring in Azure Monitor

1. Logon to Azure Portal (portal.azure.com) using your credentials
2. Click on Monitor from the list of services.



1. Click on ‘Explore Metrics’
2. Click on ‘Add metric’ and select Subscription and Resource Group names from the drop-down.
3. Select your DW instance from the list of resources displayed.
4. Select ‘CPU percentage’ as the metric from the drop-down as shown below:



1. This will display the CPU percentage (Avg) chart for the last 24 hours.
2. Click on ‘Add metric’ again and ‘Data IO Percentage’ to the chart.
3. Repeat the process to add more metrics to the chart.
4. Click on the Time Range drop-down and change the time range from ‘Last 24 hours’ to ‘Last hour’. Observe the chart data range change to display last hour data.
5. Click on ‘Add chart’ and select Subscription, Resource Group, Resource and Metric as before. You will observe a new chart created displaying the data for the metric you selected.

## Azure Data Studio SQLDW Dashboard (Azure SQL Data Warehouse Insights)

1. Open Azure Data Studio application.

Azure Data Studio (ADS) is a cross-platform database tool for monitoring and managing various data platforms, including Azure SQL DW.

1. Click on ‘Extensions’  icon from the list of icons on the left side ribbon of the window.
2. Select ‘Azure SQL Data Warehouse Insights’ from the list of extensions. (Note: Enter ‘SQL’ in the search window, if this extension is not showing up in the list)
3. Click ‘Install’ and wait for the extension installation to complete. (Note: Review the details of the extension to understand all the reports displayed by this extension. You can also click on the github link available to review the sql code for each of these reports)
4. Click on the ‘Servers’  icon on the left and click on ‘Add Connection’.
5. Expand ‘Databases’ node in the explorer window and right-click on your database and click ‘Manage. This will show Database Dashboard with details on your DW instance and tables.
6. Click on ‘SQL DW Dashboard’ tab in the details pane. Observe the reports on various metrics viz. Latest Backup date, User Activities, Data Distribution etc. (Note: Some reports maybe empty if there are no results to display.)
7. Observe the Data Distribution report. This shows data size in each of the 60 distributions. What can you infer from this report?

## Azure SQL Data Warehouse Table and Statistics Queries

1. Open SQL Server Management Studio (SSMS). Connect to the your SQL DW instance.
2. Expand Databases node, right-click on your data warehouse database and click ‘New Query’. Execute the following query in your query window to create a useful View.

CREATE VIEW dbo.vTableSizes

AS

WITH base

AS

(

SELECT

GETDATE() AS [execution\_time]

, DB\_NAME() AS [database\_name]

, s.name AS [schema\_name]

, t.name AS [table\_name]

, QUOTENAME(s.name)+'.'+QUOTENAME(t.name) AS [two\_part\_name]

, nt.[name] AS [node\_table\_name]

, ROW\_NUMBER() OVER(PARTITION BY nt.[name] ORDER BY (SELECT NULL)) AS [node\_table\_name\_seq]

, tp.[distribution\_policy\_desc] AS [distribution\_policy\_name]

, c.[name] AS [distribution\_column]

, nt.[distribution\_id] AS [distribution\_id]

, i.[type] AS [index\_type]

, i.[type\_desc] AS [index\_type\_desc]

, nt.[pdw\_node\_id] AS [pdw\_node\_id]

, pn.[type] AS [pdw\_node\_type]

, pn.[name] AS [pdw\_node\_name]

, di.name AS [dist\_name]

, di.position AS [dist\_position]

, nps.[partition\_number] AS [partition\_nmbr]

, nps.[reserved\_page\_count] AS [reserved\_space\_page\_count]

, nps.[reserved\_page\_count] - nps.[used\_page\_count] AS [unused\_space\_page\_count]

, nps.[in\_row\_data\_page\_count]

+ nps.[row\_overflow\_used\_page\_count]

+ nps.[lob\_used\_page\_count] AS [data\_space\_page\_count]

, nps.[reserved\_page\_count]

- (nps.[reserved\_page\_count] - nps.[used\_page\_count])

- ([in\_row\_data\_page\_count]

+ [row\_overflow\_used\_page\_count]+[lob\_used\_page\_count]) AS [index\_space\_page\_count]

, nps.[row\_count] AS [row\_count]

from

sys.schemas s

INNER JOIN sys.tables t

ON s.[schema\_id] = t.[schema\_id]

INNER JOIN sys.indexes i

ON t.[object\_id] = i.[object\_id]

AND i.[index\_id] <= 1

INNER JOIN sys.pdw\_table\_distribution\_properties tp

ON t.[object\_id] = tp.[object\_id]

INNER JOIN sys.pdw\_table\_mappings tm

ON t.[object\_id] = tm.[object\_id]

INNER JOIN sys.pdw\_nodes\_tables nt

ON tm.[physical\_name] = nt.[name]

INNER JOIN sys.dm\_pdw\_nodes pn

ON nt.[pdw\_node\_id] = pn.[pdw\_node\_id]

INNER JOIN sys.pdw\_distributions di

ON nt.[distribution\_id] = di.[distribution\_id]

INNER JOIN sys.dm\_pdw\_nodes\_db\_partition\_stats nps

ON nt.[object\_id] = nps.[object\_id]

AND nt.[pdw\_node\_id] = nps.[pdw\_node\_id]

AND nt.[distribution\_id] = nps.[distribution\_id]

LEFT OUTER JOIN (select \* from sys.pdw\_column\_distribution\_properties where distribution\_ordinal = 1) cdp

ON t.[object\_id] = cdp.[object\_id]

LEFT OUTER JOIN sys.columns c

ON cdp.[object\_id] = c.[object\_id]

AND cdp.[column\_id] = c.[column\_id]

)

, size

AS

(

SELECT

[execution\_time]

, [database\_name]

, [schema\_name]

, [table\_name]

, [two\_part\_name]

, [node\_table\_name]

, [node\_table\_name\_seq]

, [distribution\_policy\_name]

, [distribution\_column]

, [distribution\_id]

, [index\_type]

, [index\_type\_desc]

, [pdw\_node\_id]

, [pdw\_node\_type]

, [pdw\_node\_name]

, [dist\_name]

, [dist\_position]

, [partition\_nmbr]

, [reserved\_space\_page\_count]

, [unused\_space\_page\_count]

, [data\_space\_page\_count]

, [index\_space\_page\_count]

, [row\_count]

, ([reserved\_space\_page\_count] \* 8.0) AS [reserved\_space\_KB]

, ([reserved\_space\_page\_count] \* 8.0)/1000 AS [reserved\_space\_MB]

, ([reserved\_space\_page\_count] \* 8.0)/1000000 AS [reserved\_space\_GB]

, ([reserved\_space\_page\_count] \* 8.0)/1000000000 AS [reserved\_space\_TB]

, ([unused\_space\_page\_count] \* 8.0) AS [unused\_space\_KB]

, ([unused\_space\_page\_count] \* 8.0)/1000 AS [unused\_space\_MB]

, ([unused\_space\_page\_count] \* 8.0)/1000000 AS [unused\_space\_GB]

, ([unused\_space\_page\_count] \* 8.0)/1000000000 AS [unused\_space\_TB]

, ([data\_space\_page\_count] \* 8.0) AS [data\_space\_KB]

, ([data\_space\_page\_count] \* 8.0)/1000 AS [data\_space\_MB]

, ([data\_space\_page\_count] \* 8.0)/1000000 AS [data\_space\_GB]

, ([data\_space\_page\_count] \* 8.0)/1000000000 AS [data\_space\_TB]

, ([index\_space\_page\_count] \* 8.0) AS [index\_space\_KB]

, ([index\_space\_page\_count] \* 8.0)/1000 AS [index\_space\_MB]

, ([index\_space\_page\_count] \* 8.0)/1000000 AS [index\_space\_GB]

, ([index\_space\_page\_count] \* 8.0)/1000000000 AS [index\_space\_TB]

FROM base

)

SELECT \*

FROM size

;

GO

1. Execute the following query in the query window. This will show table space summary information.

SELECT

database\_name

, schema\_name

, table\_name

, distribution\_policy\_name

, distribution\_column

, index\_type\_desc

, COUNT(distinct partition\_nmbr) as nbr\_partitions

, SUM(row\_count) as table\_row\_count

, SUM(reserved\_space\_GB) as table\_reserved\_space\_GB

, SUM(data\_space\_GB) as table\_data\_space\_GB

, SUM(index\_space\_GB) as table\_index\_space\_GB

, SUM(unused\_space\_GB) as table\_unused\_space\_GB

FROM

dbo.vTableSizes

GROUP BY

database\_name

, schema\_name

, table\_name

, distribution\_policy\_name

, distribution\_column

, index\_type\_desc

ORDER BY

table\_reserved\_space\_GB desc

1. Click ‘New Query’. Execute the following query in your query window. This will list tables with >10% skew.

select two\_part\_name

,[distribution\_column]

,(max(row\_count \* 1.000) - min(row\_count \* 1.000))/max(row\_count \* 1.000) as 'skew'

,max(row\_count) as 'largest dist'

,min(row\_count) as 'smallest dist'

,avg(row\_count) as 'average dist'

,sum(row\_count) as 'total row\_count'

from dbo.vTableSizes

where row\_count > 0

AND distribution\_policy\_name = 'HASH'

group by two\_part\_name, [distribution\_column]

having (max(row\_count \* 1.000) - min(row\_count \* 1.000))/max(row\_count \* 1.000) >= .10

order by 3 DESC -- two\_part\_name

1. Expand Databases node, right-click on your data warehouse database and click ‘New Query’. Execute the following query in your query window to create another useful View.

CREATE VIEW dbo.vStats\_Columns

AS

SELECT

sm.[name] AS [schema\_name]

, tb.[name] AS [table\_name]

, st.[name] AS [stats\_name]

, st.[filter\_definition] AS [stats\_filter\_defiinition]

, st.[has\_filter] AS [stats\_is\_filtered]

, STATS\_DATE(st.[object\_id],st.[stats\_id])

AS [stats\_last\_updated\_date]

, st.[user\_created] AS [user\_created]

, co.[name] AS [stats\_column\_name]

, ty.[name] AS [column\_type]

, co.[max\_length] AS [column\_max\_length]

, co.[precision] AS [column\_precision]

, co.[scale] AS [column\_scale]

, co.[is\_nullable] AS [column\_is\_nullable]

, co.[collation\_name] AS [column\_collation\_name]

, QUOTENAME(sm.[name])+'.'+QUOTENAME(tb.[name])

AS two\_part\_name

, QUOTENAME(DB\_NAME())+'.'+QUOTENAME(sm.[name])+'.'+QUOTENAME(tb.[name])

AS three\_part\_name

, QUOTENAME(sm.[name])+'.'+QUOTENAME(tb.[name])+'.'+QUOTENAME(st.[name])

AS full\_stats\_name

--, st.[stats\_generation\_method\_desc] AS [stats\_generation\_method\_desc]

FROM sys.objects AS ob

JOIN sys.stats AS st ON ob.[object\_id] = st.[object\_id]

JOIN sys.stats\_columns AS sc ON st.[stats\_id] = sc.[stats\_id]

AND st.[object\_id] = sc.[object\_id]

JOIN sys.columns AS co ON sc.[column\_id] = co.[column\_id]

AND sc.[object\_id] = co.[object\_id]

JOIN sys.types AS ty ON co.[user\_type\_id] = ty.[user\_type\_id]

JOIN sys.tables AS tb ON co.[object\_id] = tb.[object\_id]

JOIN sys.schemas AS sm ON tb.[schema\_id] = sm.[schema\_id]

;

GO

1. Execute the following query in the query window. This will show table statistics information.

SELECT \* FROM dbo.vStats\_Columns ORDER BY 1,2,3

## 

## Create User-defined Restore Points

Azure SQL Data Warehouse includes automated database snapshots that can be leveraged to recover or copy a data warehouse to a previous state. These snapshots support an eight-hour recovery point objective (RPO) and are available to be used for 7 days. If you require a faster RPO or you require your snapshot to be available for longer than 7 days, you can manually trigger a snapshot to save the current database state. This is good practice to follow before and after large modifications to your data warehouse - it allows quicker recovery times in the event of any workload interruptions or user errors.

We will create a user-defined restore point, but not actually perform a restore due to the limitations of working in this lab environment and time constraints.

1. Logon to Azure Portal (portal.azure.com) using your credentials
2. Navigate to the Overview page of your Azure Data Warehouse.
3. Select New Restore Point



1. Specify a name for your restore point



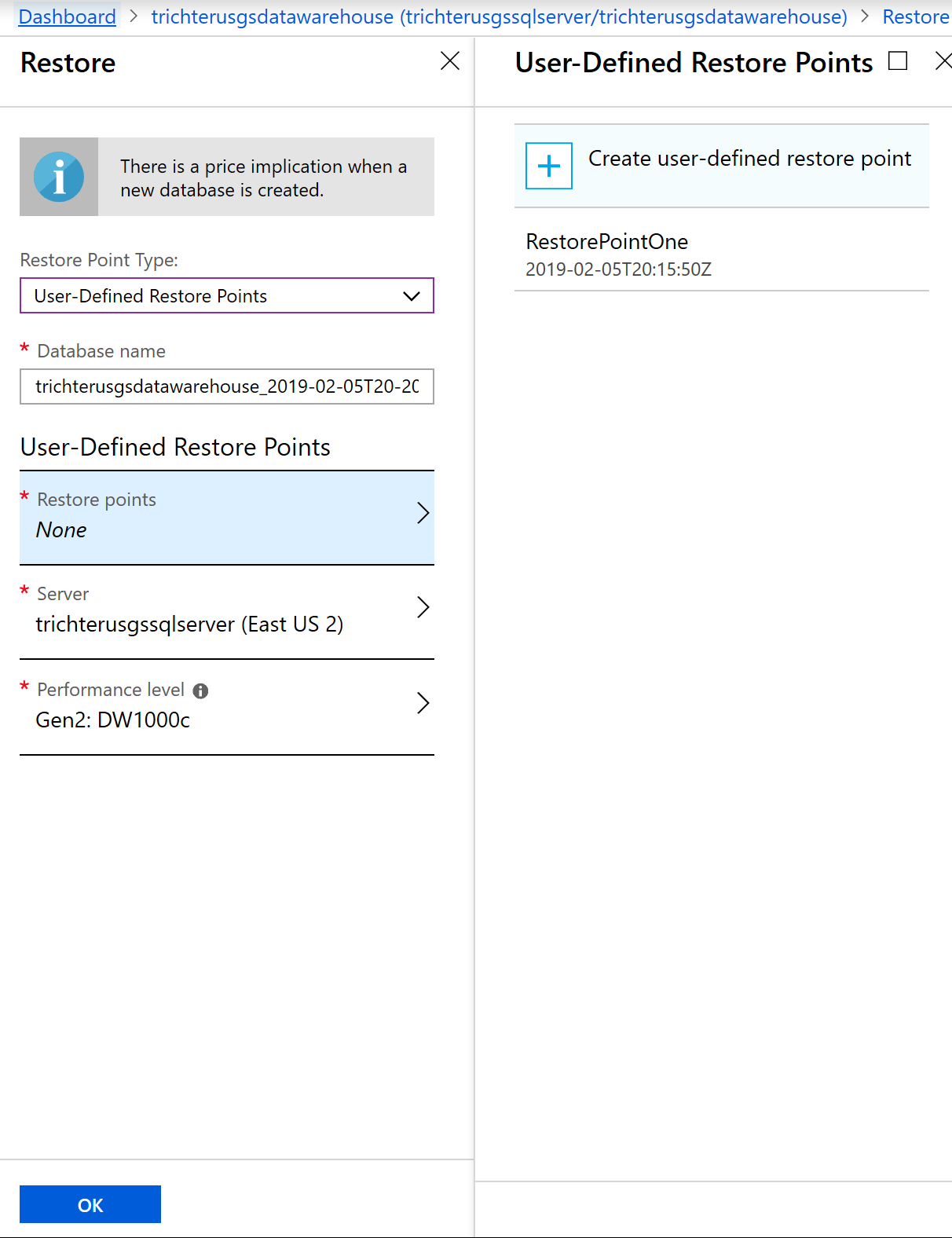
1. Click Ok
2. Navigate to the Overview page of your Azure Data Warehouse.
3. Select Restore



1. Select Restore Point Type: User-Defined Restore Points

Click Restore points to activate the User-Defined Restore Points blade

Notice your recently created user-defined restore point.



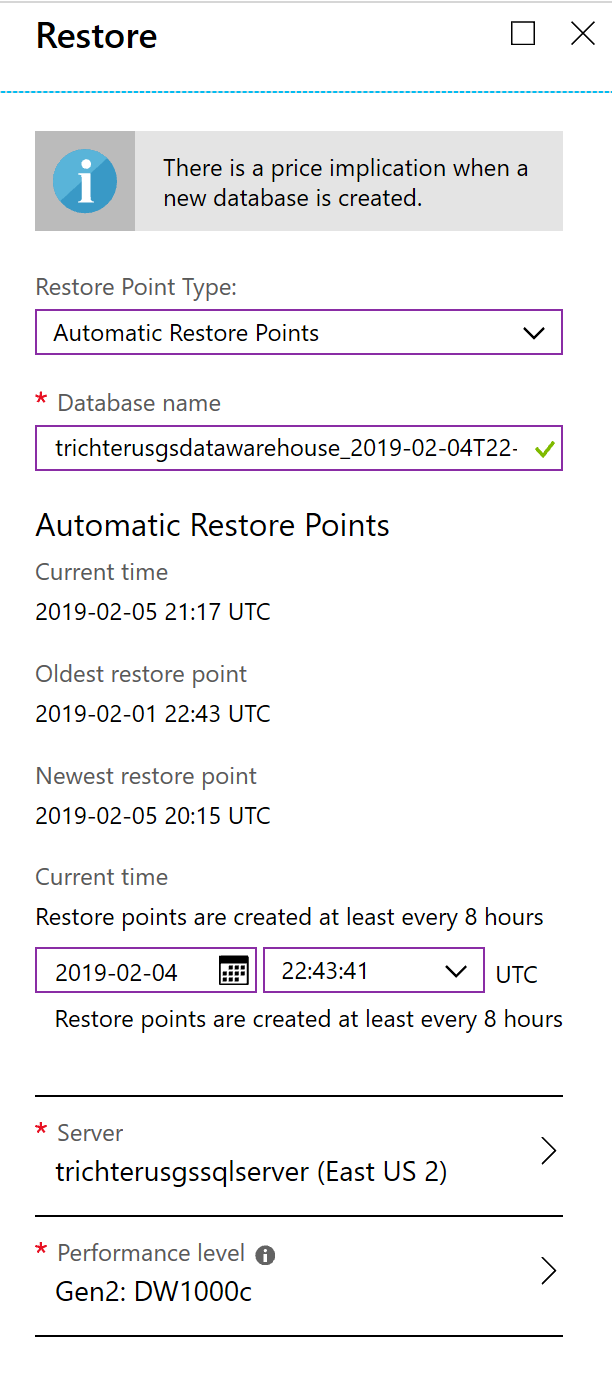
1. Click X to close the User-Defined Restore Points blade
2. Investigate the Automatic Restore Points

Select Restore Point Type: Automatic Restore Points

Notice the Oldest restore point and Newest restore point information

Select the calendar control and click a **bold** date

Notice that the time drop down is now populated with the restore point times available.



1. Click x to close the Restore blade.

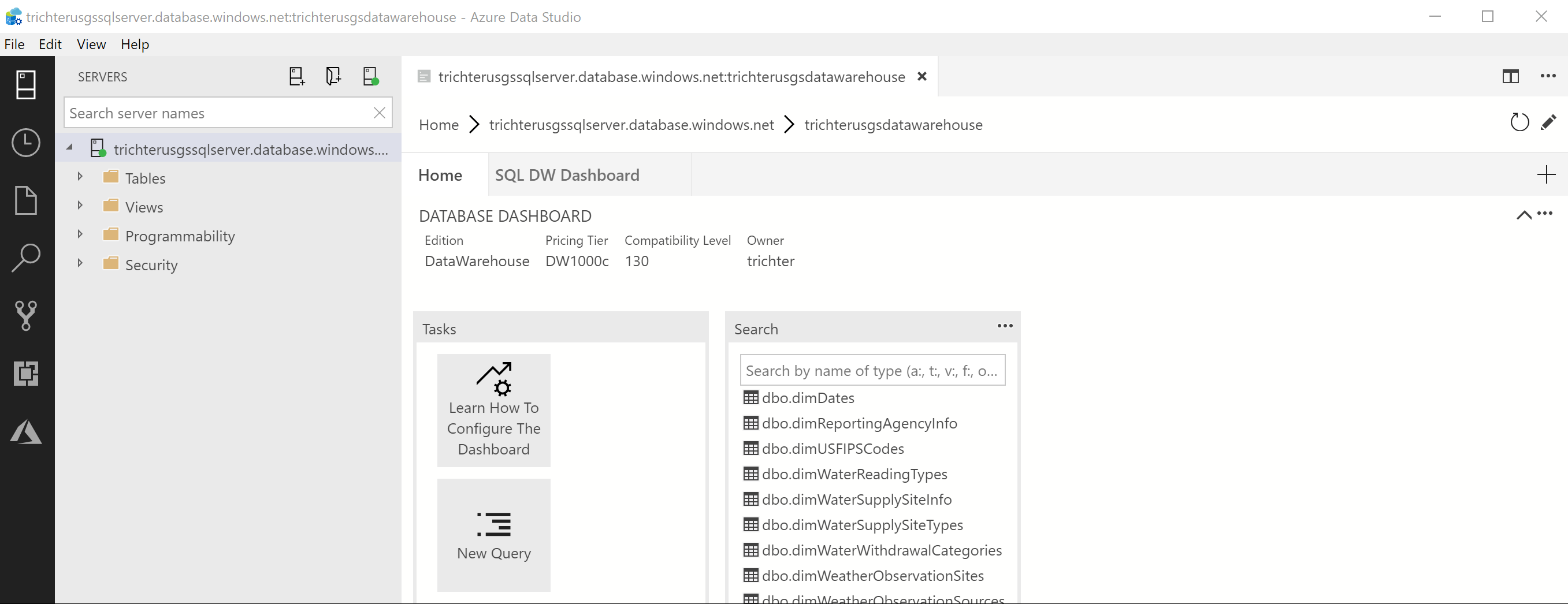
We will not perform an actual restore due to time and resource constraints of this lab environment.

1. Let’s see how restore points can be queried using Azure Data Studio

Open your Azure Data Studio application and connect to your Azure Data Warehouse

(You must have completed the prior module Azure SQL Data Studio SQLDW Dashboard)

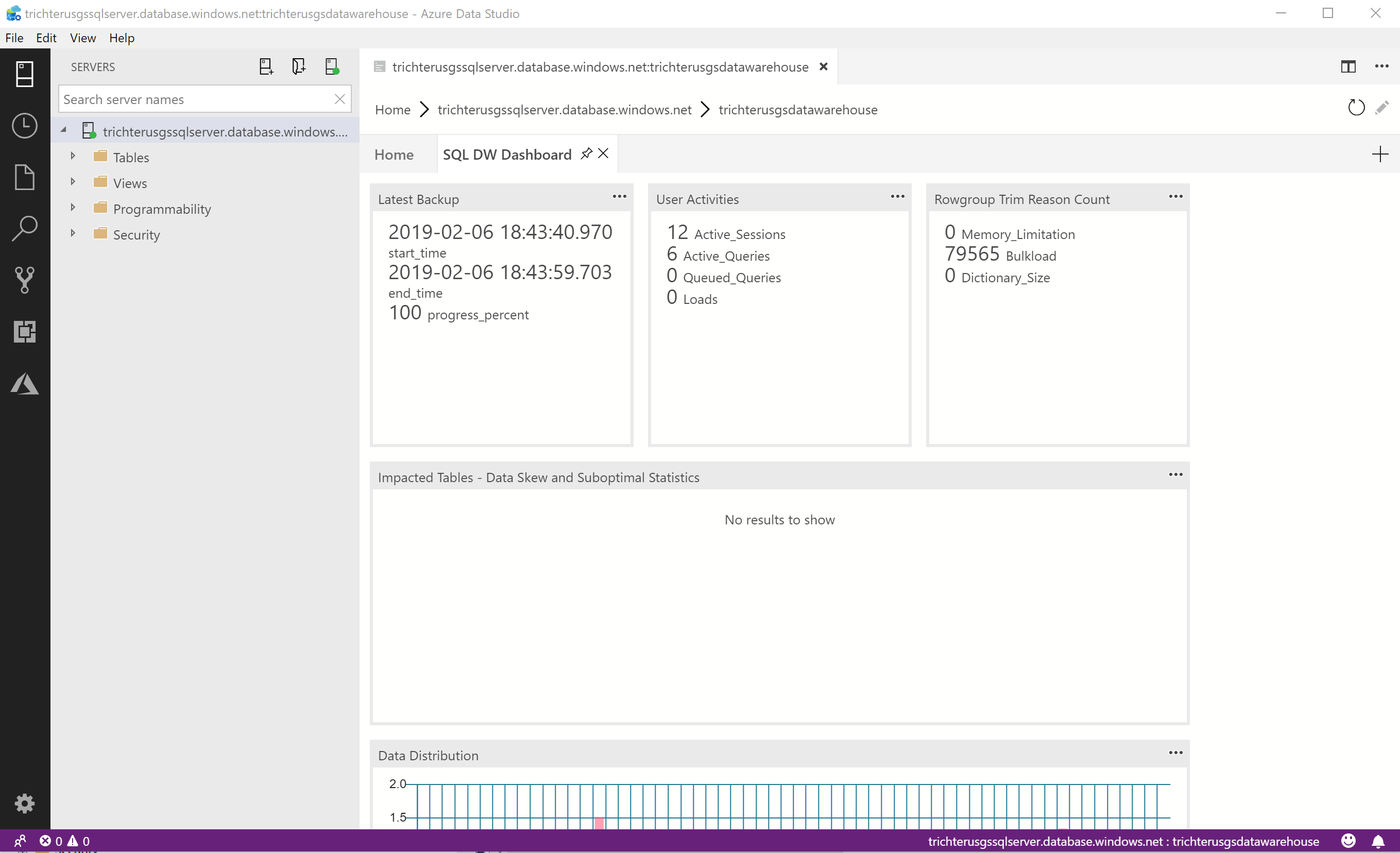
1. Right click your Azure Data Warehouse server and select Manage



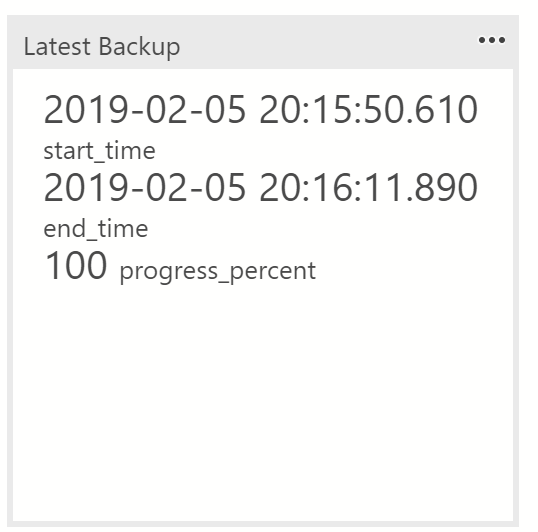
1. Select the SQL DW Dashboard tab

Notice the default summary information provided for user activities, CCI Health (Rowgroup Trim Reason Count), Tables that are being impacted by skew or by statistics issues, Etc.

The queries that run behind these visualizations are available to you. Let’s look into the Latest Backup information.

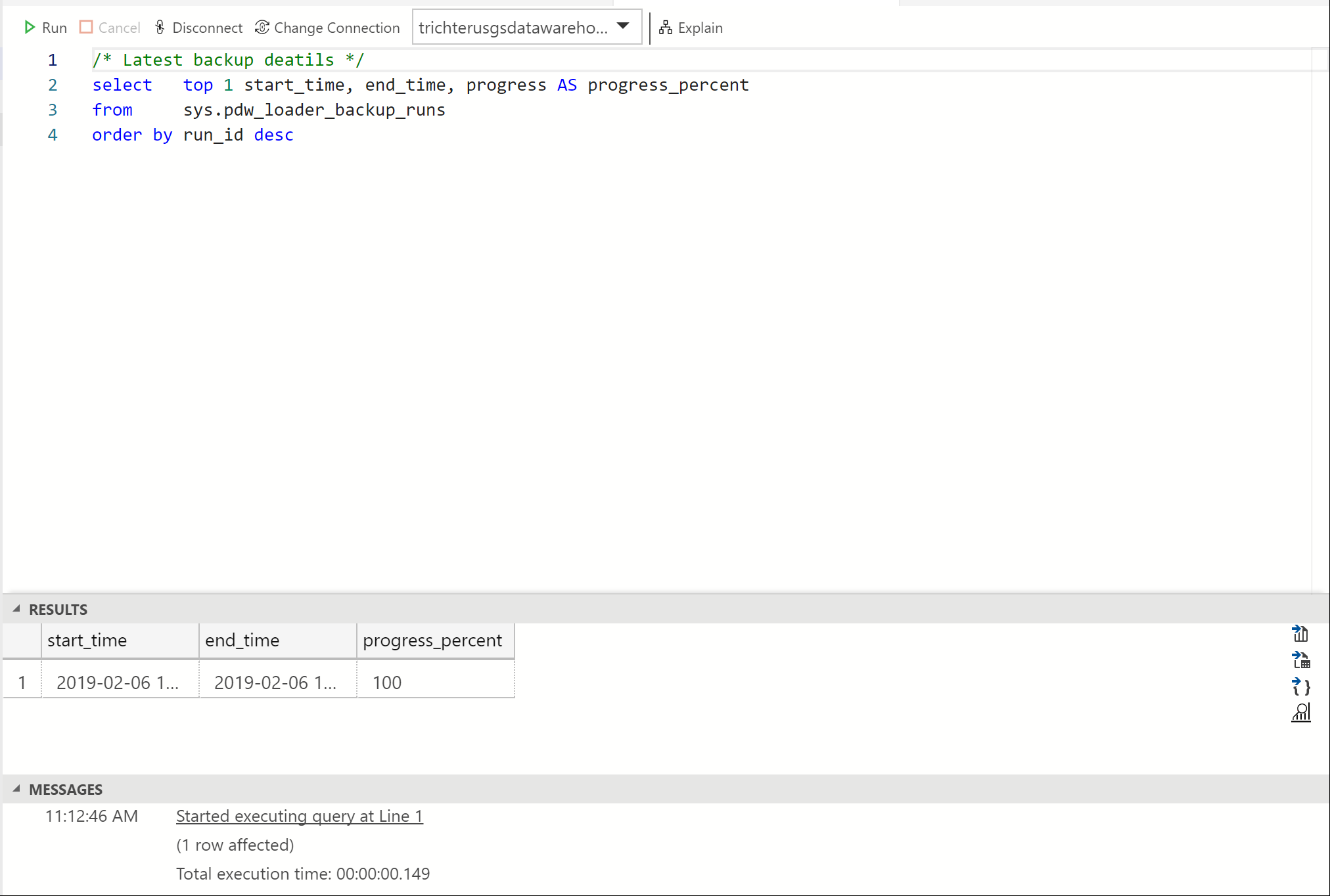


1. Select the ellipses (…) in the right-hand corner of the Latest Backup visualization

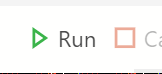


1. Select Run Query

A new tab will open and execute the query that populates the Latest Backup visualization and the result set will be displayed.



1. To review all of the backups that are available, modify the query by removing the top 1 clause from the select statement and click run

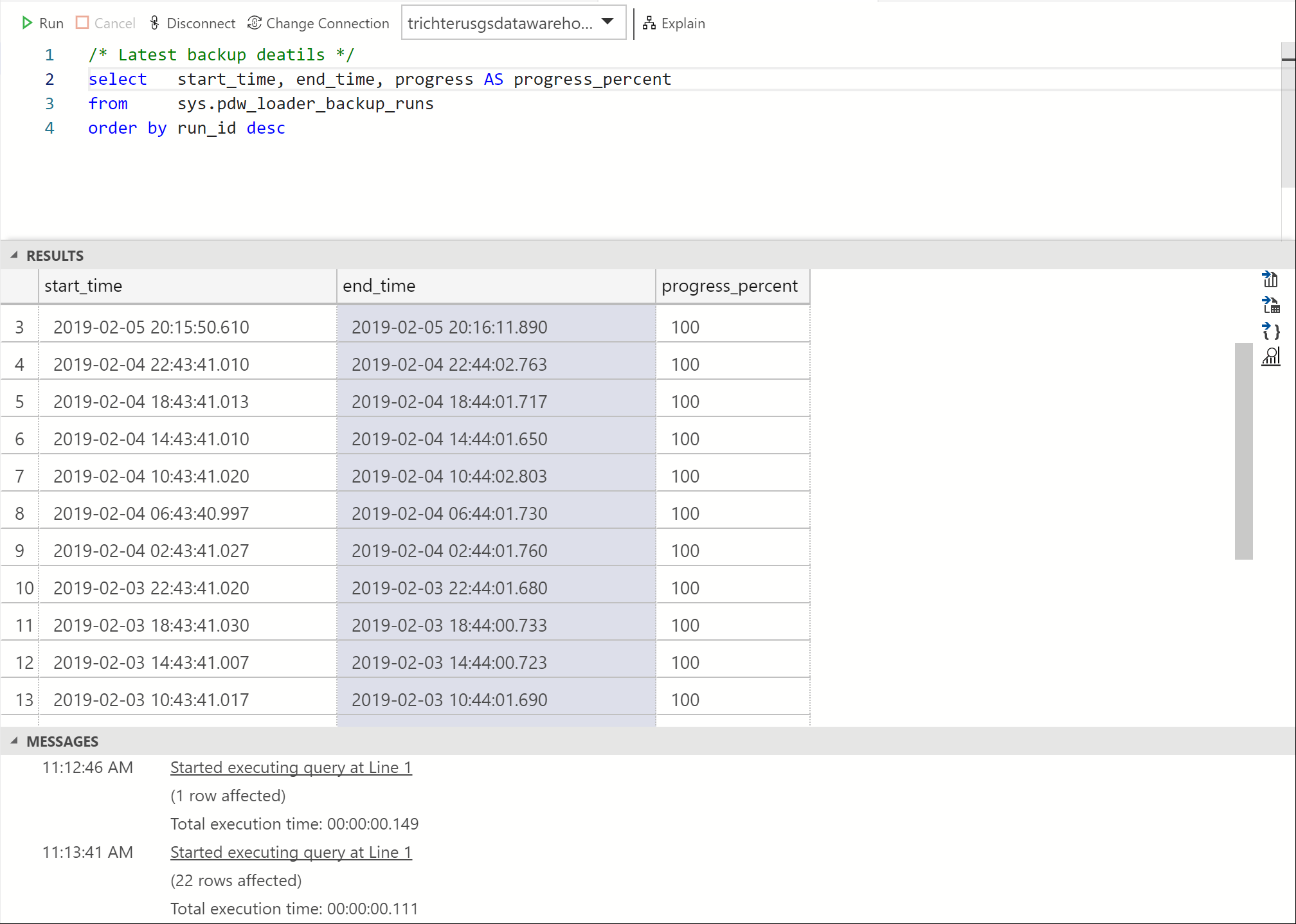


select start\_time, end\_time, progress AS progress\_percent

from sys.pdw\_loader\_backup\_runs

order by run\_id desc

1. All of the available restore points will be listed. Notice that they do not have friendly names. Your user-defined restore point is likely at the top, check the date and time.



1. Close the tab. Don’t Save your changes

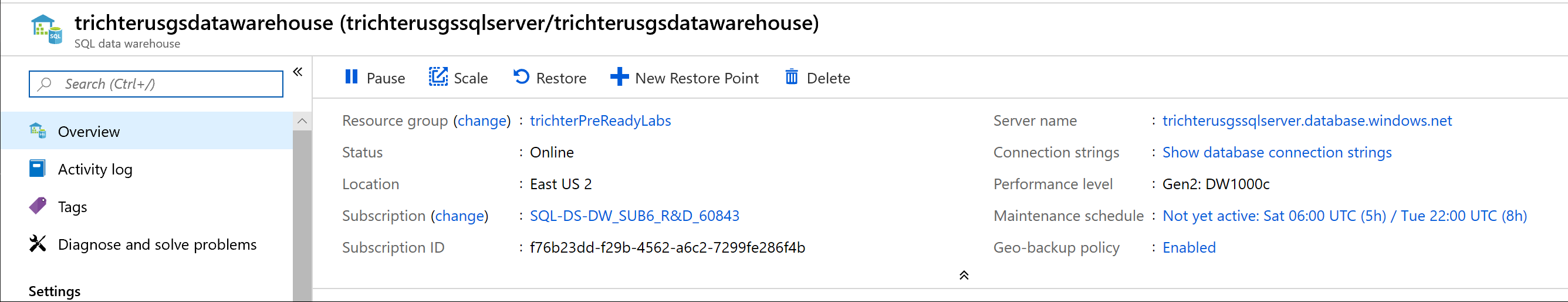
## Maintenance Window Scheduling, Service Health, Service Health Alerts

By default, all newly created Azure SQL Data Warehouse instances have an eight-hour primary and secondary maintenance window applied during deployment. You can change the windows as soon deployment is complete. No maintenance will take place outside the specified maintenance windows without prior notification.

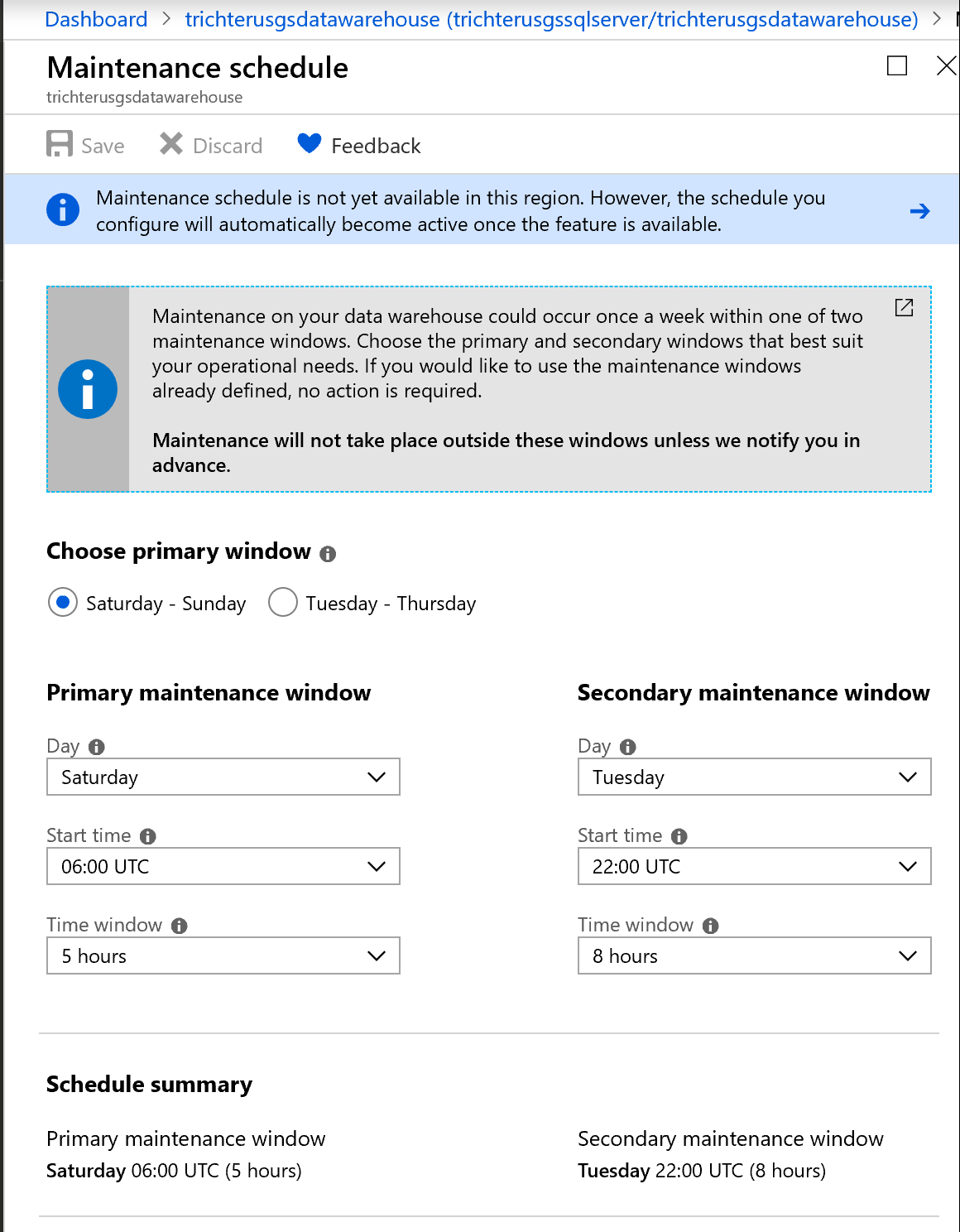
1. Logon to Azure Portal (portal.azure.com) using your credentials
2. Navigate to your data warehouse overview blade

Notice the Maintenance schedule text link

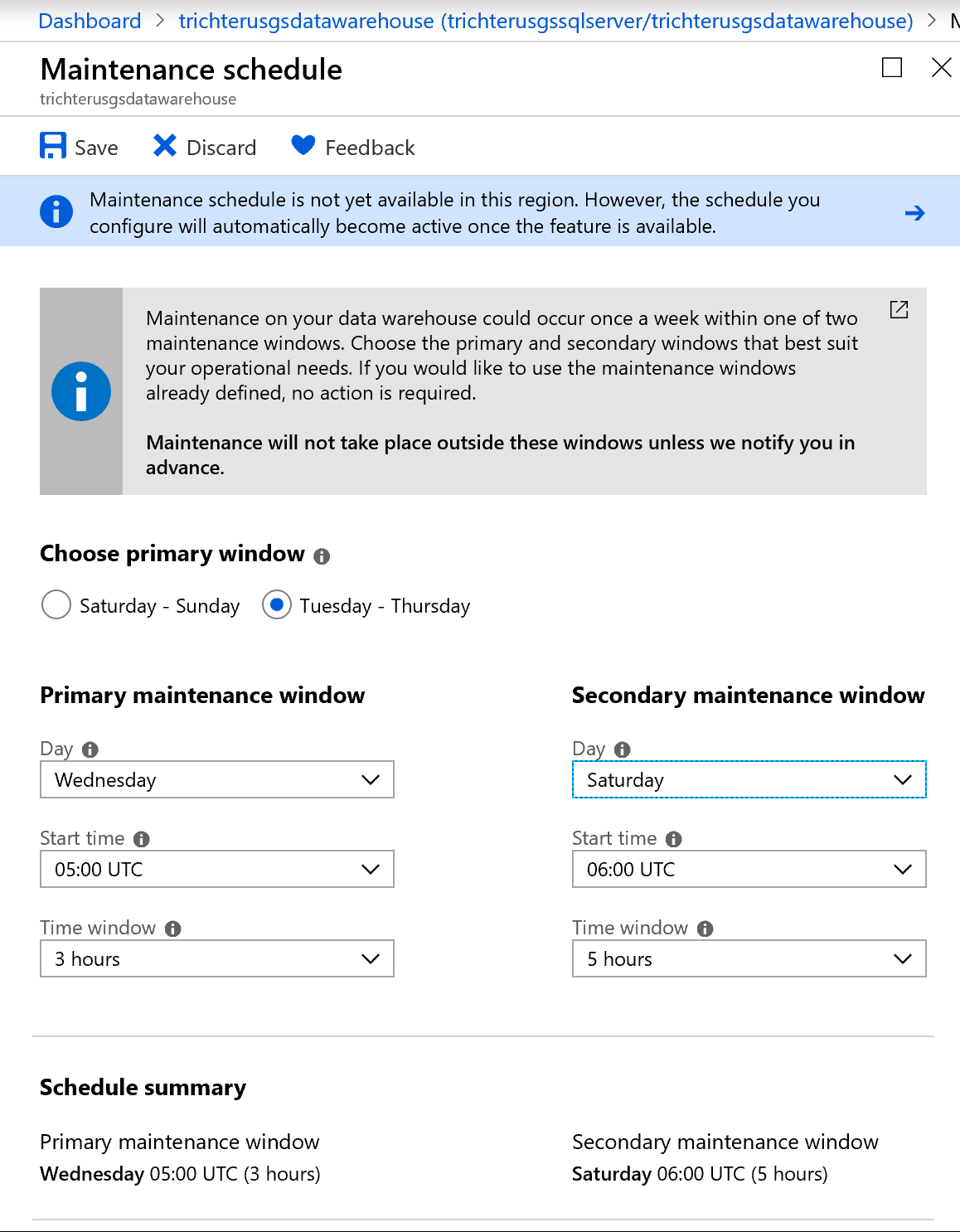
The state of any maintenance will be displayed in this area. “Not yet active” indicates that the current day and time are not within an active maintenance window.



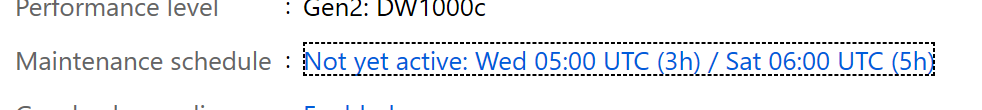
1. Click on Not Yet Active text link to activate the Maintenance schedule blade



1. Notice the various configuration options and configure a Primary and Secondary maintenance window

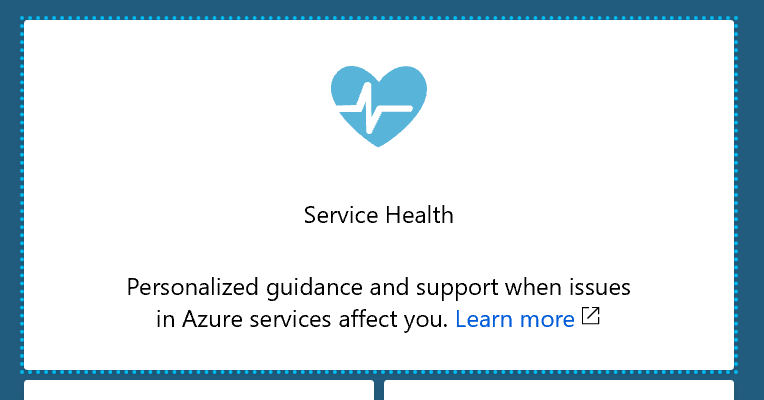


1. Click Save
2. Return to your Overview blade and notice the new days and times listed for Maintenance schedule



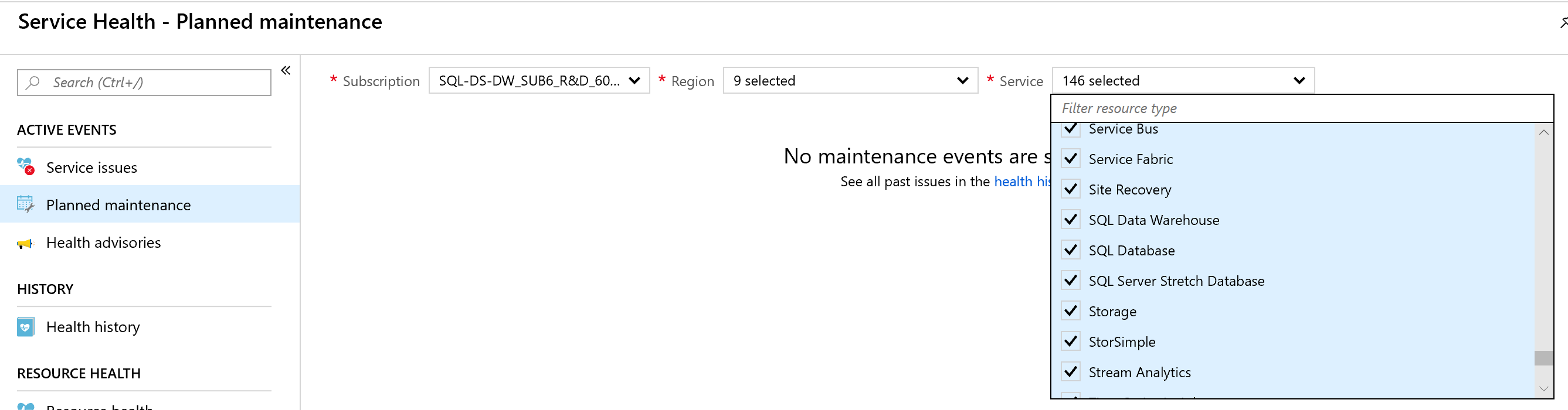
1. Planned Maintenance Activities will appear in Service Health

From the Dashboard select Service Health



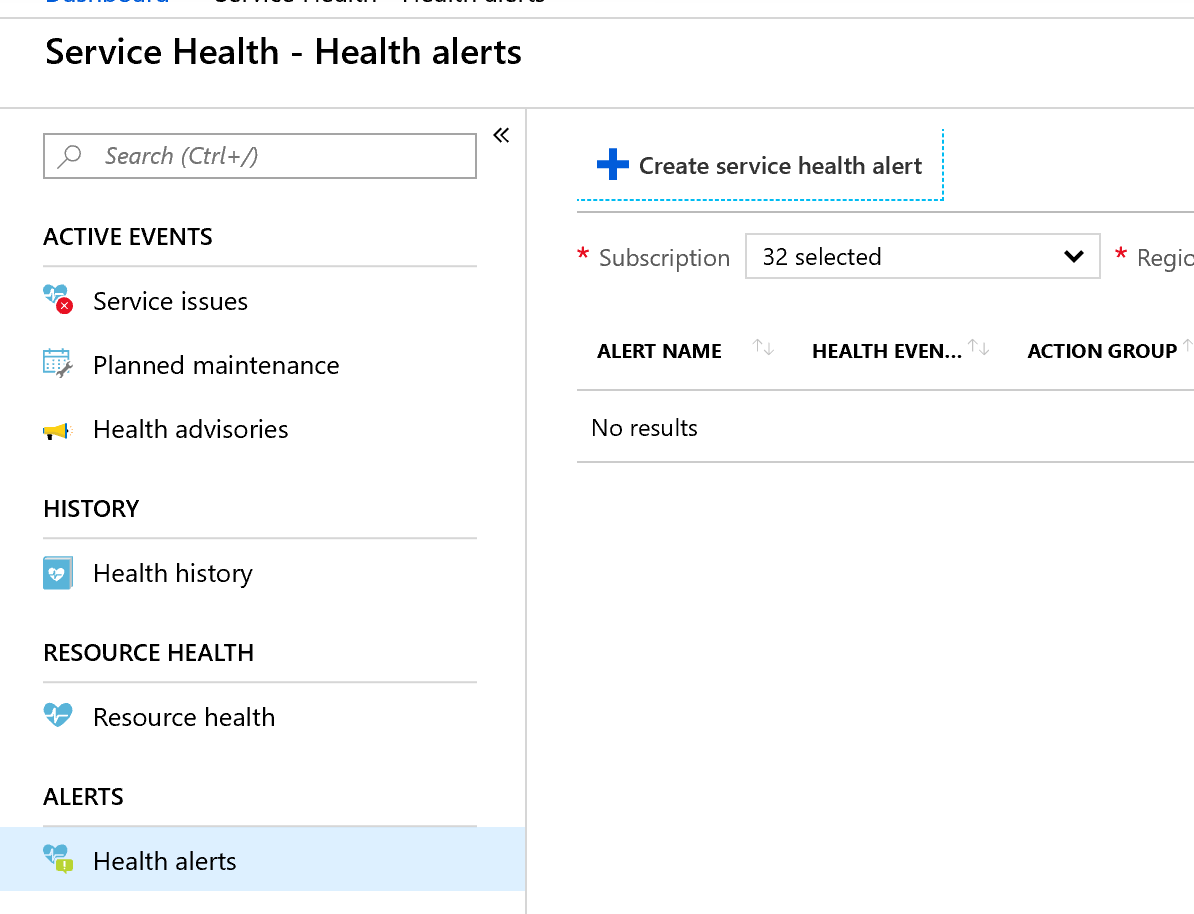
1. Click Planned Maintenance from the side menu to view Planned Maintenance events and notifications

Filter for SQL Data Warehouse

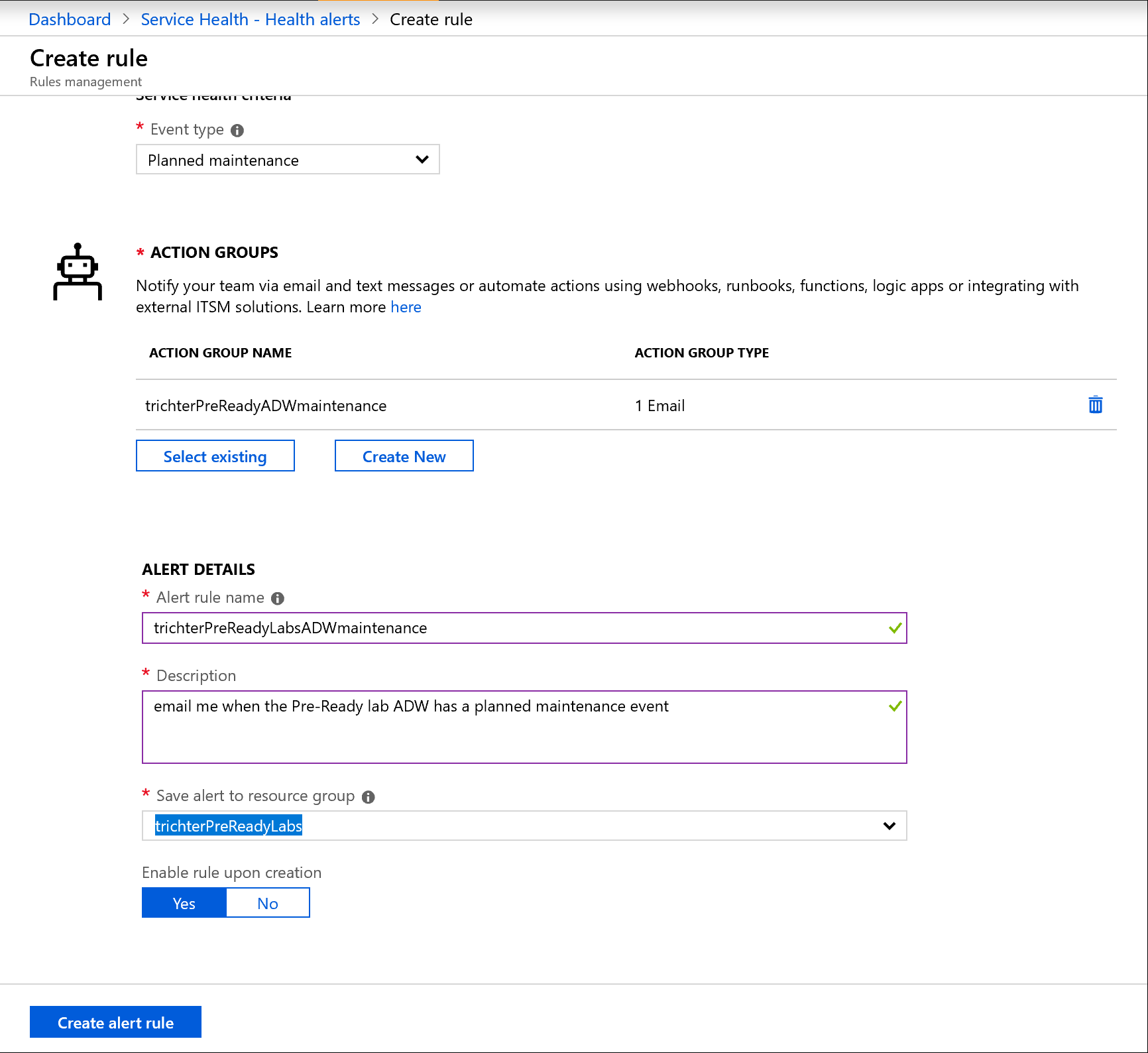


1. To be alerted for Planned SQL Data Warehouse planned maintenance

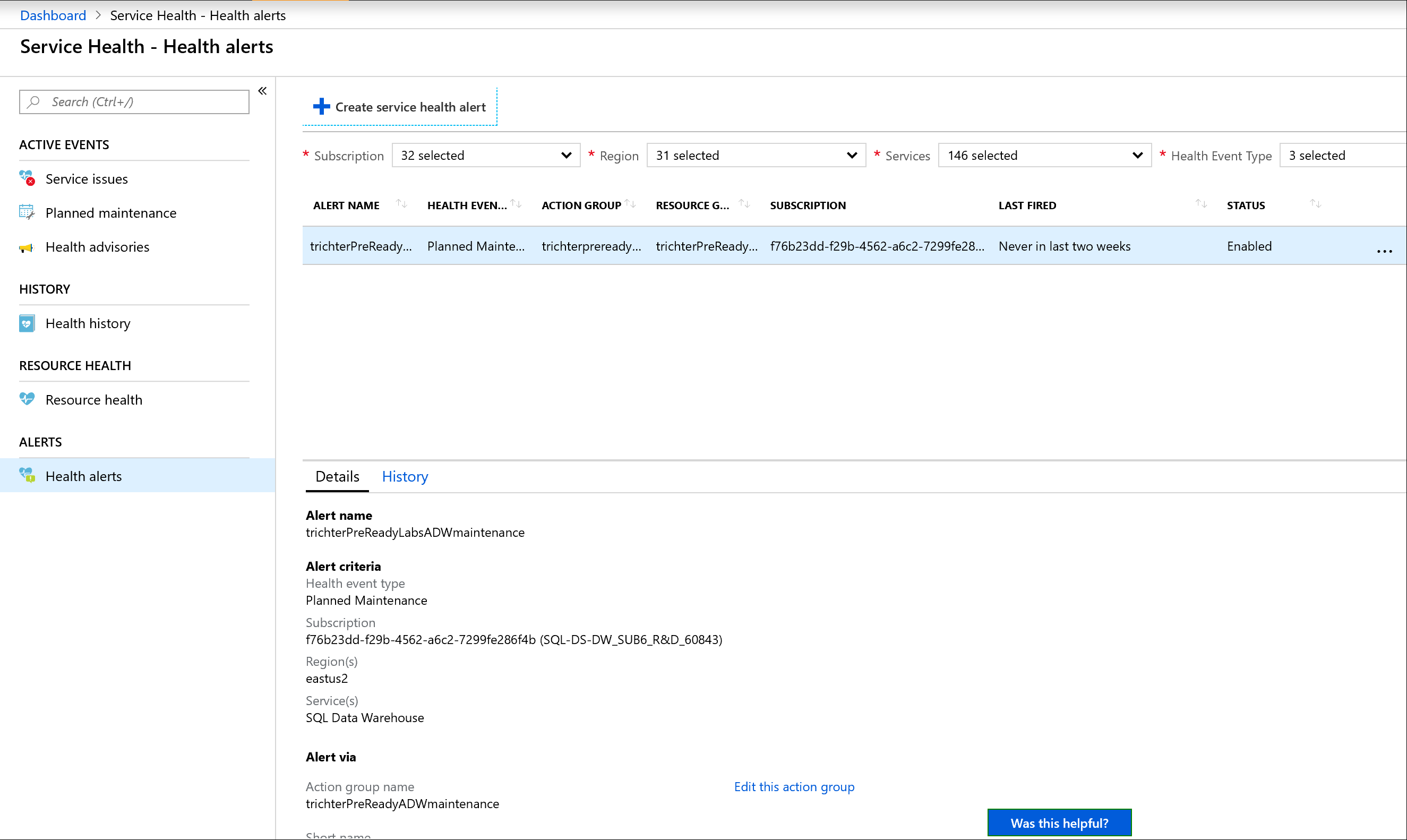
Select Health alerts and click Create service health alert



1. Configure a new rule



1. Click Create alert rule and refresh the Health alerts blade



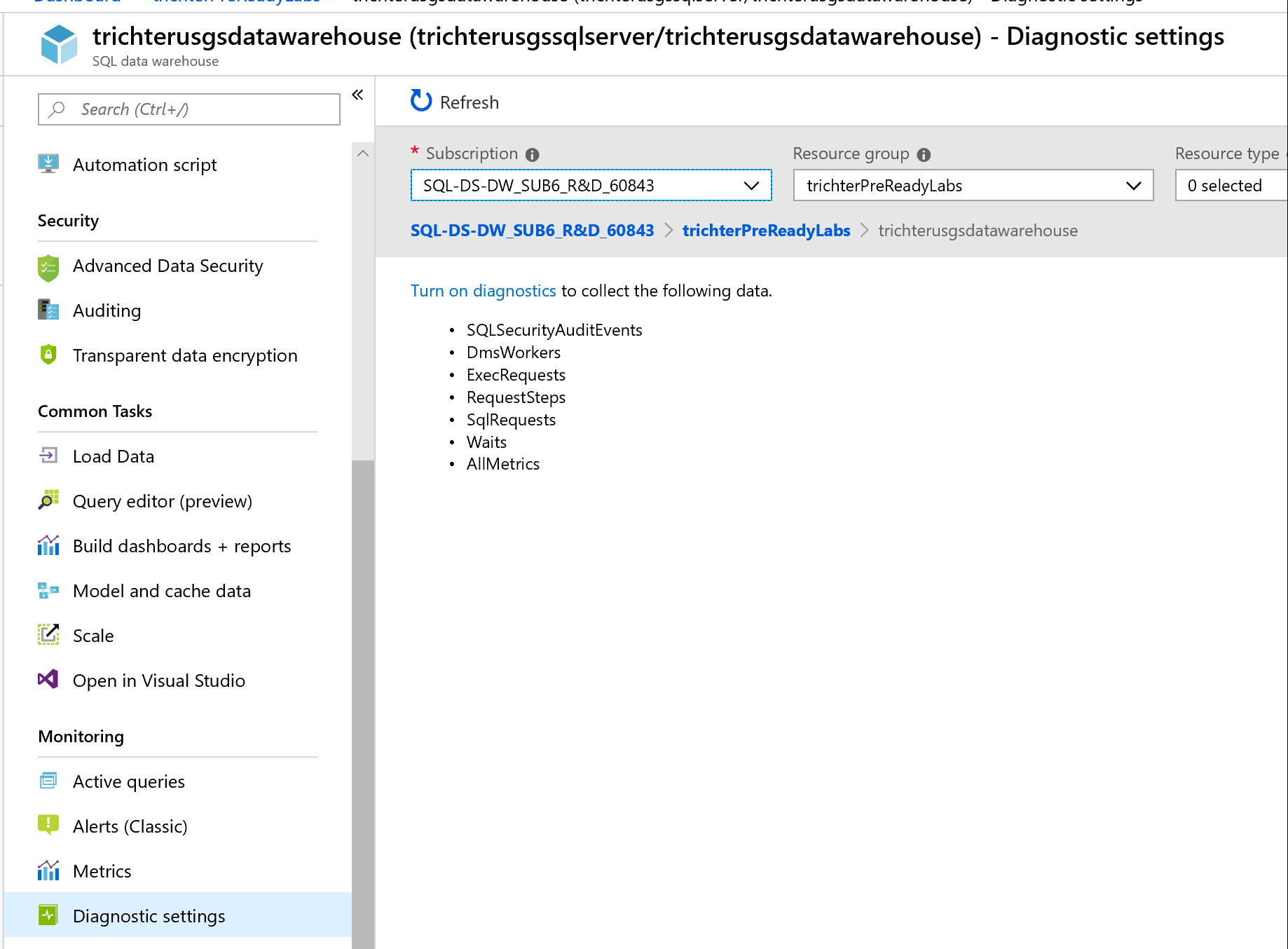
1. You will be able to view the details and history of this health alert from this blade.
2. Disable your alert by clicking the ellipses(…) and selecting Disable

## Querying ADW Diagnostic Logs using Azure Monitor

SQL Data Warehouse (SQL DW) now enables enhanced insights into analytical workloads by integrating directly with Microsoft Azure Monitor diagnostic logs. This new capability enables developers to analyze workload behavior over an extended time period and make informed decisions on query optimization or capacity management.

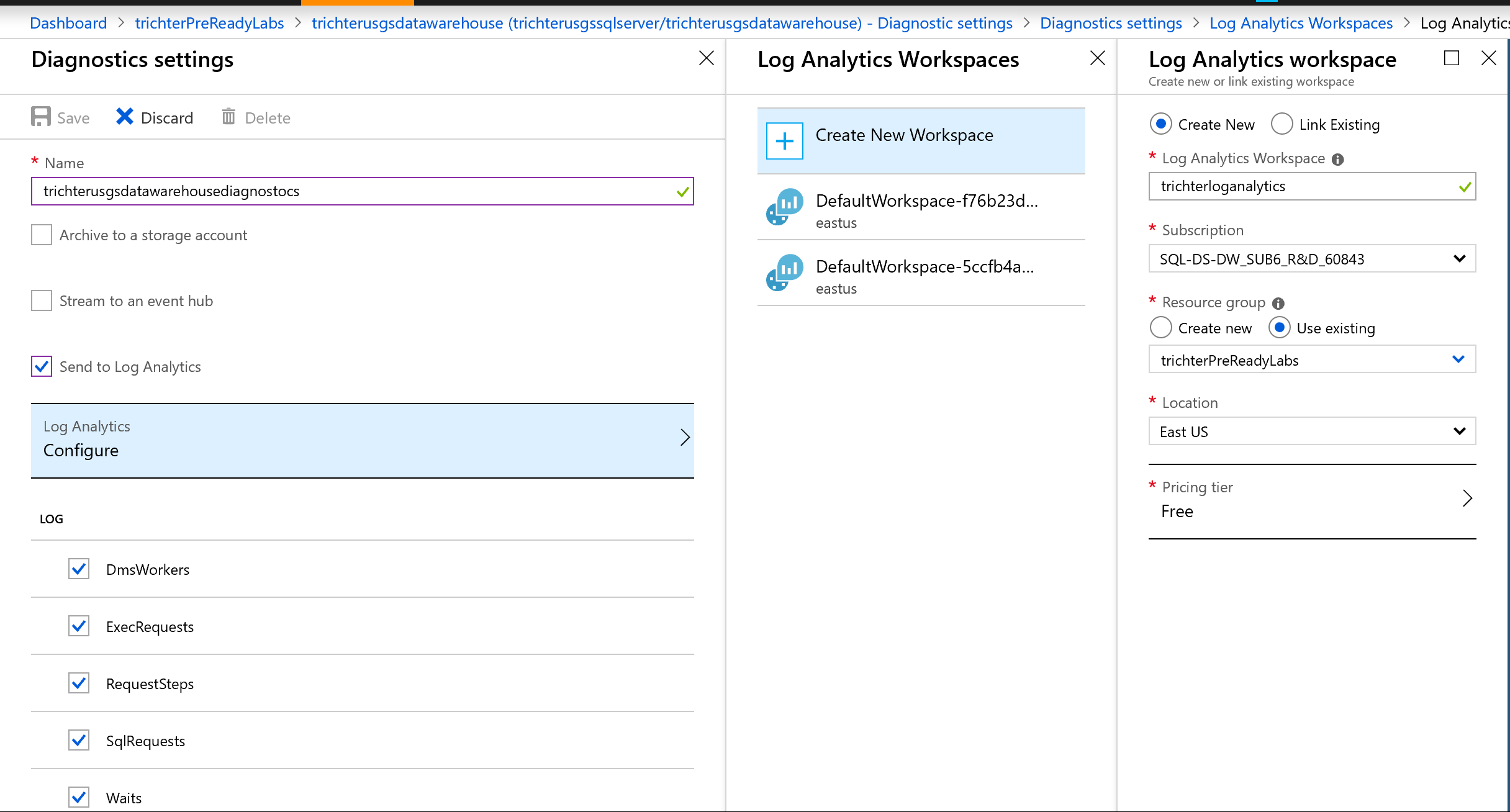
### If you did not configure Diagnostics earlier:

1. Logon to Azure Portal (portal.azure.com) using your credentials
2. Navigate to your Azure Data Warehouse
3. Click on Diagnostic Settings from the side menu

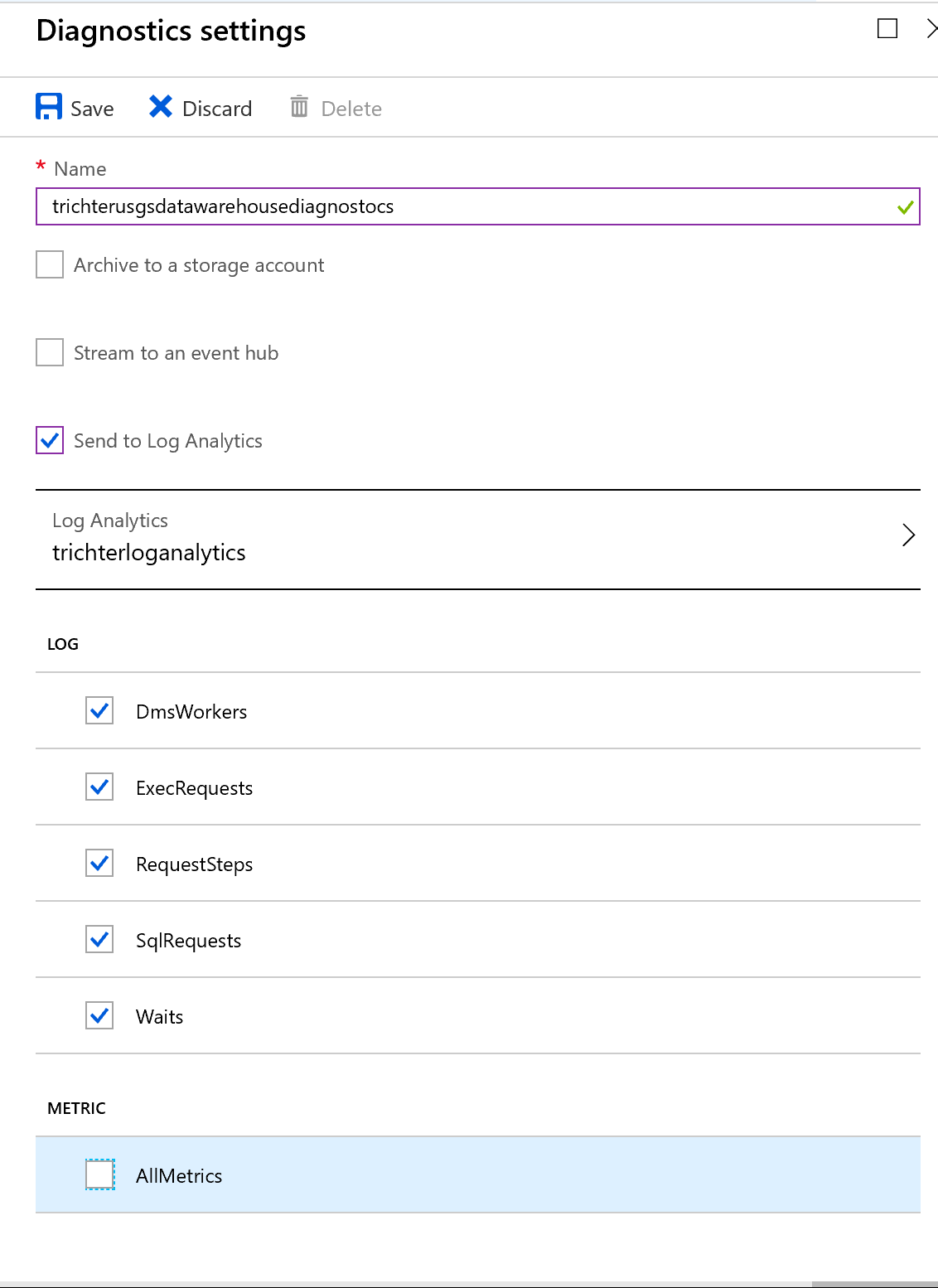


1. Click on the Turn on diagnostics text link in the blade
2. Provide a name for your diagnostics

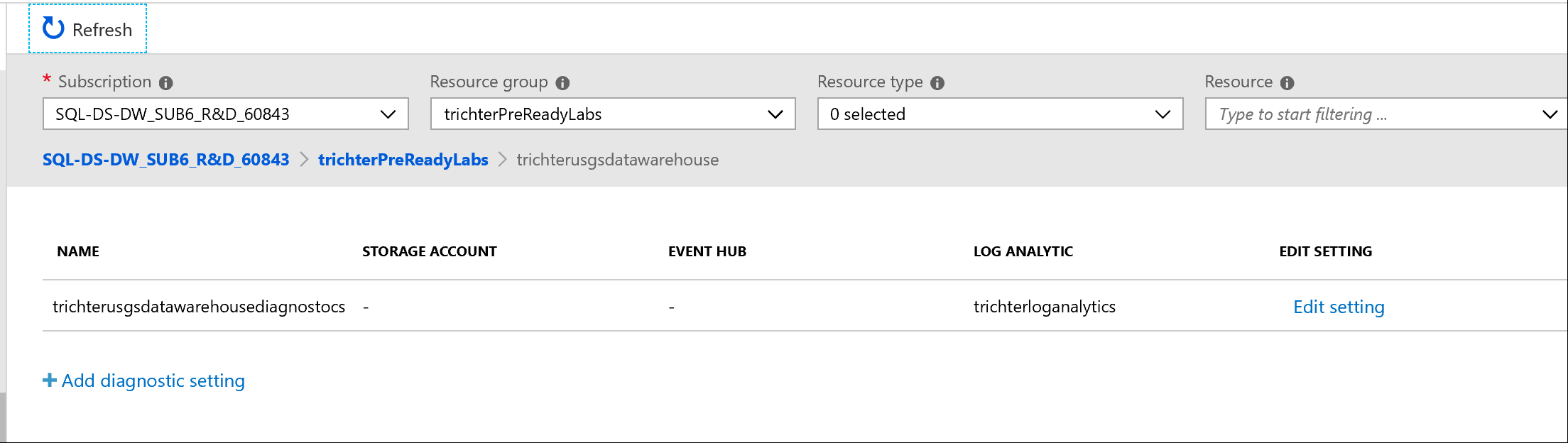
* Check Send to Log Analytics
* Select all options in LOG
* Click Log Analytics Configure to activate the Log Analytics Workspaces blade
  + Create a new Workspace in your resource group using one of the charged pricing tiers



1. Click Save on Diagnostics settings blade



1. Click refresh on Diagnostic settings blade to see your new diagnostics



### Review Diagnostics Logs:

1. To collect some data, using SSMS connect to your Azure Data Warehouse and execute some queries. These queries are in the 05\_SQLDWManagement\_Monitoring.sql file in section 2.

SELECT FIS.SalesAmount, DG.PostalCode, DC.YearlyIncome AS CustomerIncome, DD.FullDateAlternateKey AS OrderDate

FROM FactInternetSales AS FIS

JOIN DimCustomer AS DC

ON (FIS.CustomerKey = DC.CustomerKey)

JOIN DimDate AS DD

ON (FIS.OrderDateKey = DD.DateKey)

JOIN DimGeography AS DG

ON (DC.GeographyKey = DG.GeographyKey);

SELECT fis.SalesAmount, dst.Gender, dst.NumberCarsOwned, dst.YearlyIncome AS CustomerYearlyIncome, dst.TotalChildren

FROM FactInternetSales AS fis

LEFT OUTER JOIN DimCustomer AS dst

ON (fis.CustomerKey=dst.CustomerKey);

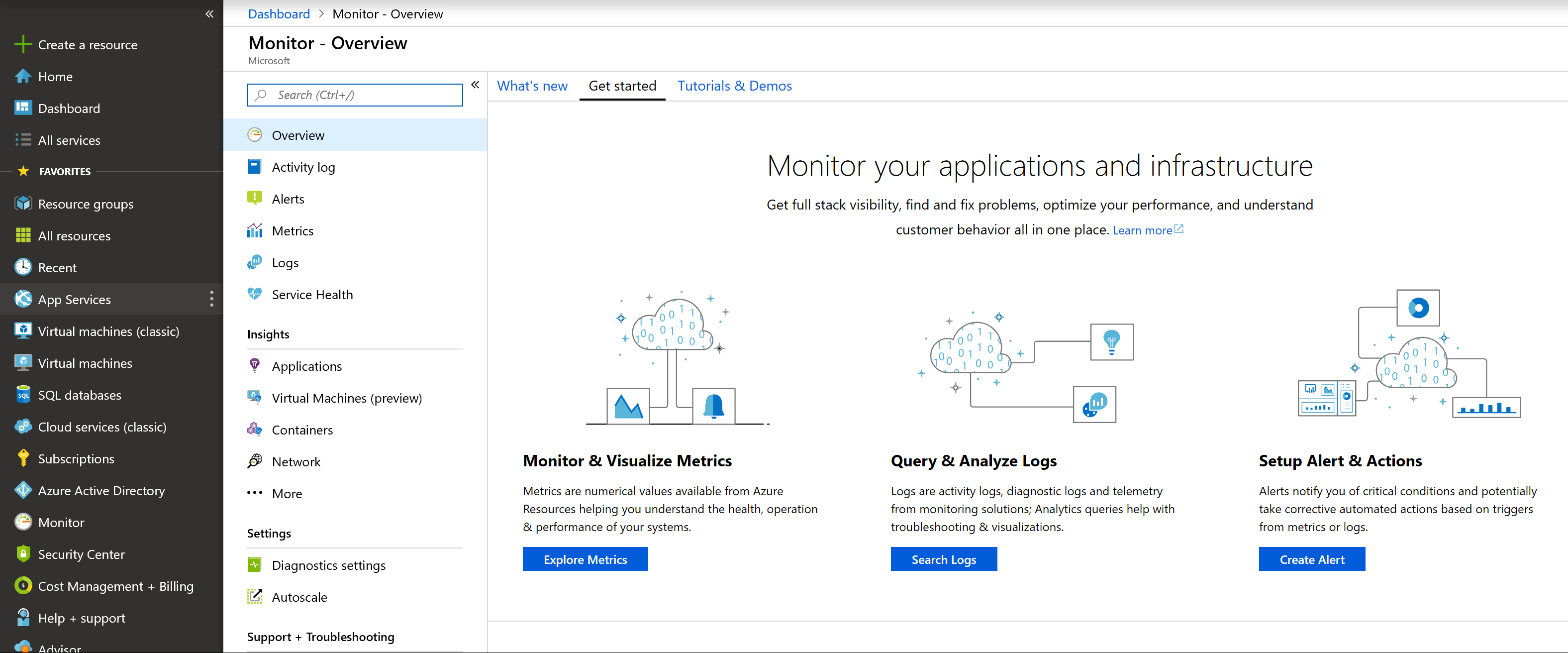
SELECT fis.SalesAmount, dst.ProductLine

FROM FactInternetSales AS fis

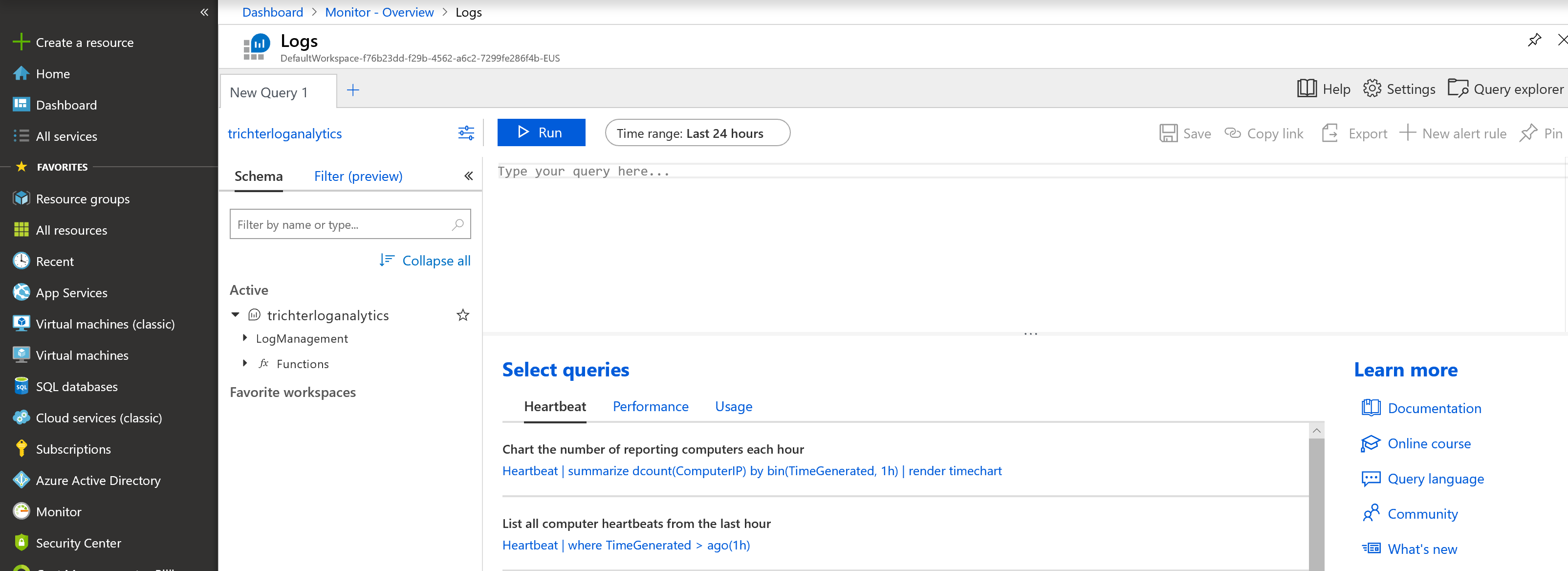
LEFT OUTER JOIN DimProduct AS dst

ON (fis.ProductKey=dst.ProductKey);

1. In the portal select Monitor from the sidebar to activate the Monitor – Overview blade



1. Click Search Logs



1. Execute these sample queries to get a feel for the information that can be retrieved from Azure Diagnostics.

Chart to determine the most active resource classes by request

AzureDiagnostics

| where Category contains "ExecRequests"

| where Status\_s == "Completed"

| summarize totalQueries = dcount(RequestId\_s) by ResourceClass\_s

| render barchart

Count of all queued queries

AzureDiagnostics

| where Category contains "waits"

| where Type\_s == "UserConcurrencyResourceType"

| summarize totalQueuedQueries = dcount(RequestId\_s)

Chart for top requests most impacted by data movement operations

AzureDiagnostics

| where Category == "RequestSteps"

| where OperationType\_s in ("ShuffleMoveOperation", "BroadcastMoveOperation", "PartitionMoveOperation", "RoundRobinMoveOperation", "SingleSourceRoundRobinMoveOperation", "MoveOperation", "TrimMoveOperation")

| where Status\_s == "Complete"

| project RequestId\_s, duration=datetime\_diff('millisecond',EndTime\_t, StartTime\_t)

| order by duration desc

| take 10

| render barchart