Microsoft

Monitoring and Tuning Azure SQL Database

Module 7

Learning Units covered in this Module

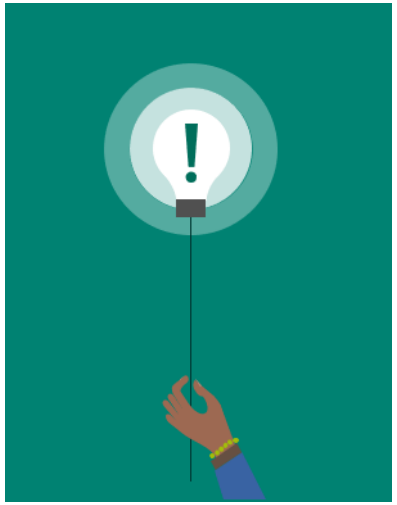
- Lesson 1: Using Query Performance Insight
- Lesson 2: Using Automatic Tuning
- Lesson 3: Using SSMS Built-In Reports
- Lesson 4: Using Metrics and Alerts

Lesson 1: Using Query Performance Insight

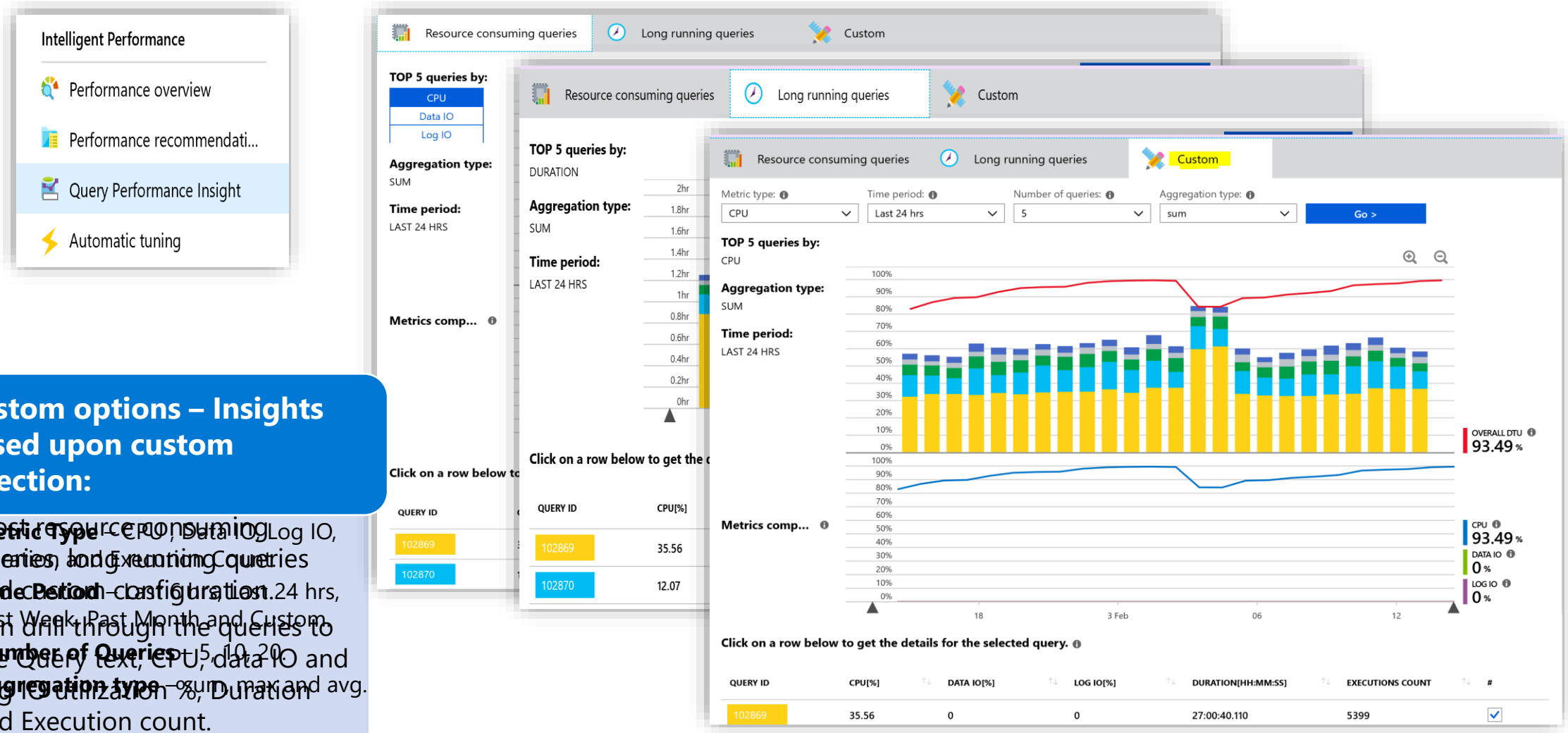
Objectives

After completing this learning, you will be able to:

- Know how to troubleshoot the performance of your queries by using Query Performance Insight.



Query Performance Insight



Custom options – Insights based upon custom selection:

- **Metric Type** – CPU, Data IO, Log IO, Duration and Execution Count
- **Time Period** – Last 1, 6, 12, 24 hrs, Past Week, Past Month and Custom
- **Number of Queries** – 5, 10, 20
- **Aggregation type** – sum, max and avg.

most resource consuming queries and Long running queries

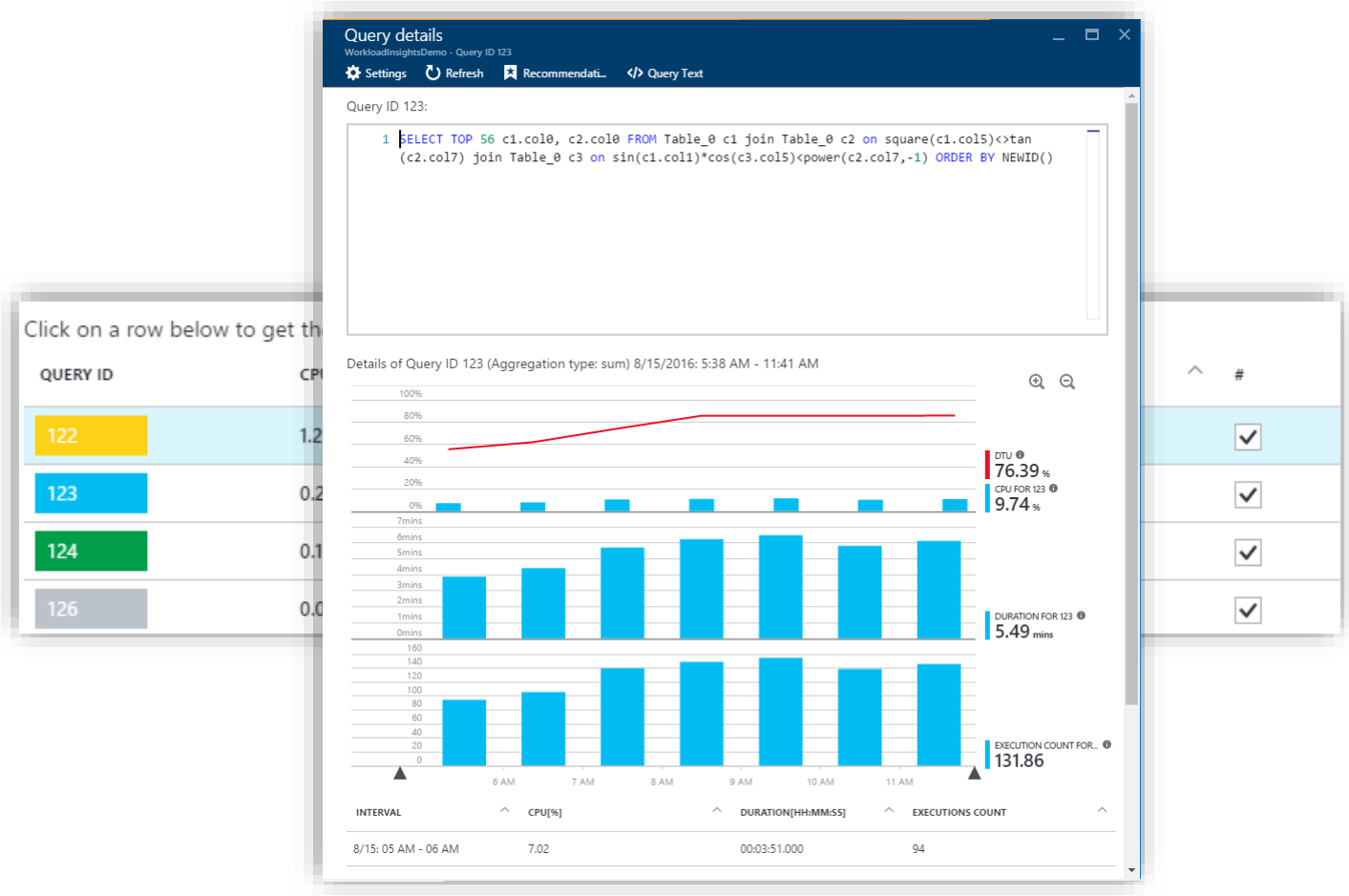
Can drill through the queries to see Query text, CPU, data IO and Log IO utilization %, Duration and Execution count.

Viewing individual query details

Get details for the individual queries

- CPU Consumption
- Duration
- Execution Count

It does not capture DDL queries

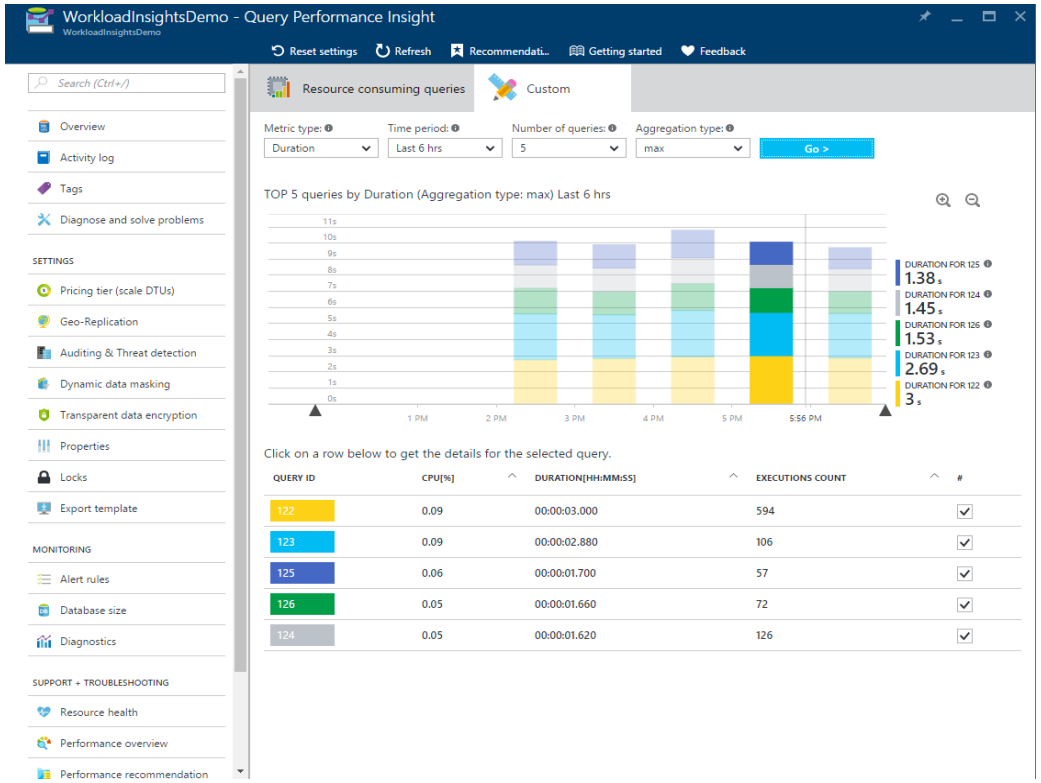


Review top queries per duration

Duration is one of the metrics showing potential bottleneck

Long-running queries has potential for:

- Longer locks
- Blocking other users
- Limiting scalability

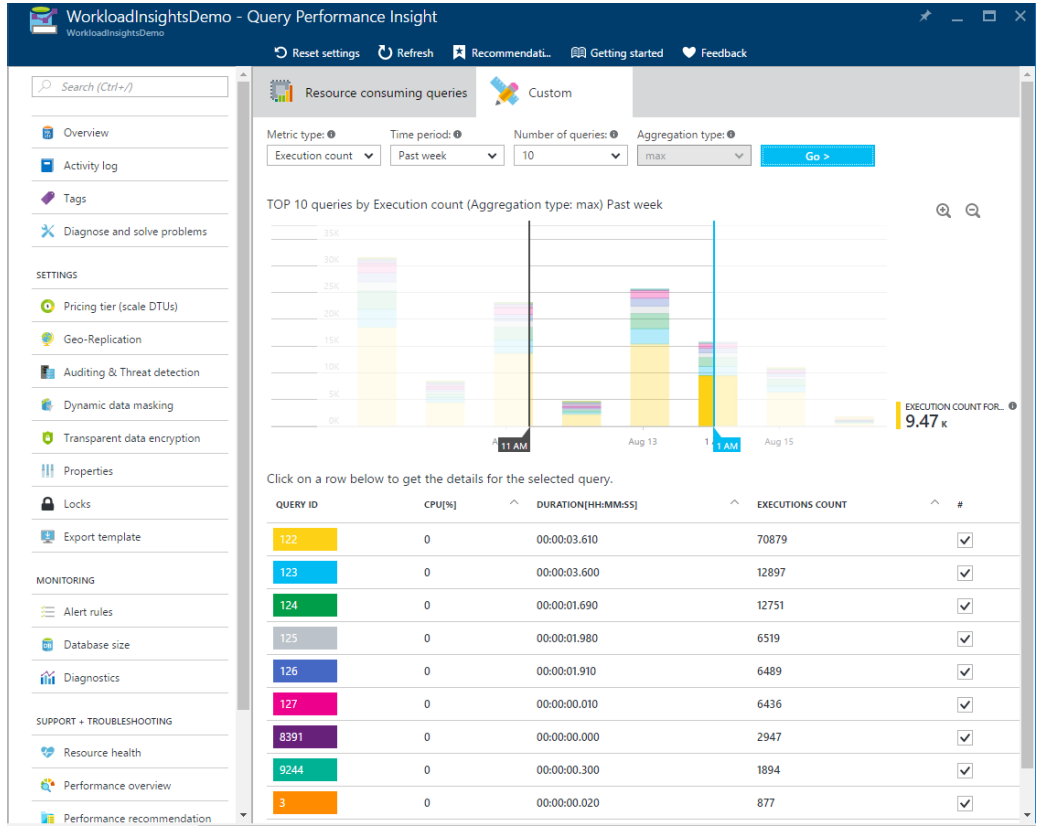


Review top queries per execution count

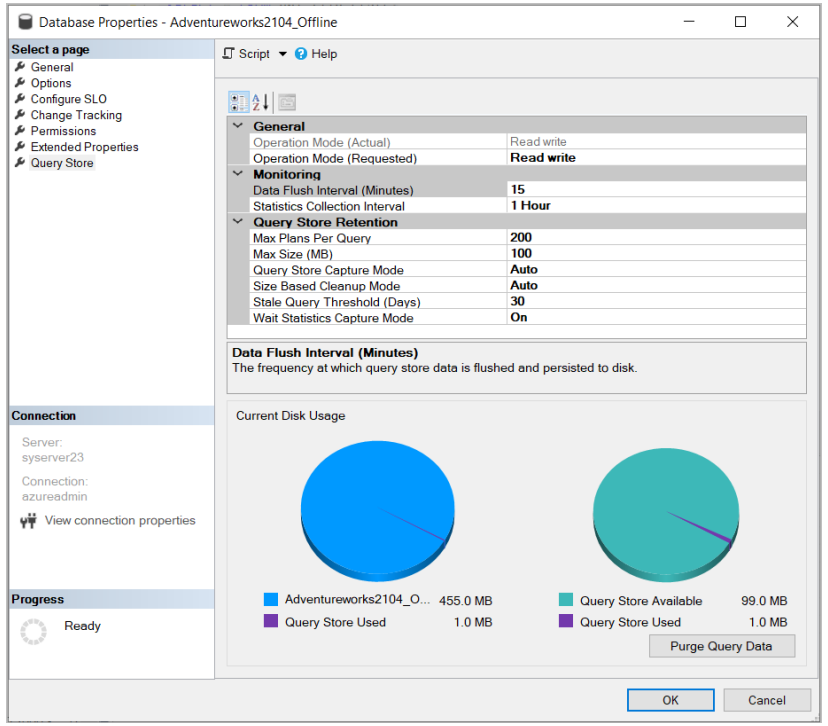
Execution count is one of the metrics showing potential bottleneck

High number og executions has potential for:

- Database performance
- Network latency
- Downstream server latency



Query Store



Retention Policy

- Size based – Auto cleanup when near max size.
- Time based – Default 30 days.
- Max Plans Per Query – Default 200.
- Wait Statistics Capture Mode – Default On.

Capture Policy

- All – Captures all queries.
- Auto – Infrequent queries are ignored.
- None – No queries are captured.
- Custom – Advanced Options

Demonstration

Query Performance Insight

- Analyze the Query Performance Insight output.

Questions?

Knowledge Check

What feature should be enabled on your Azure SQL Database before you can use Query Performance Insight?

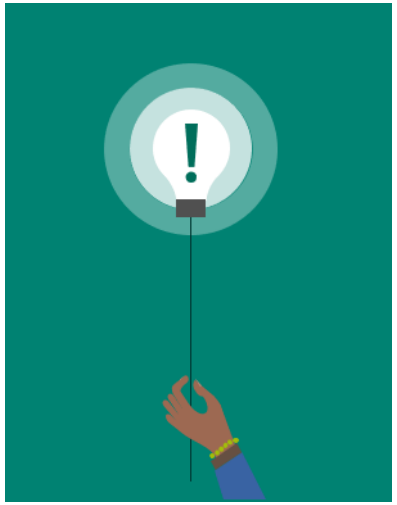
How can you view individual query details?

Lesson 2: Using Automatic Tuning

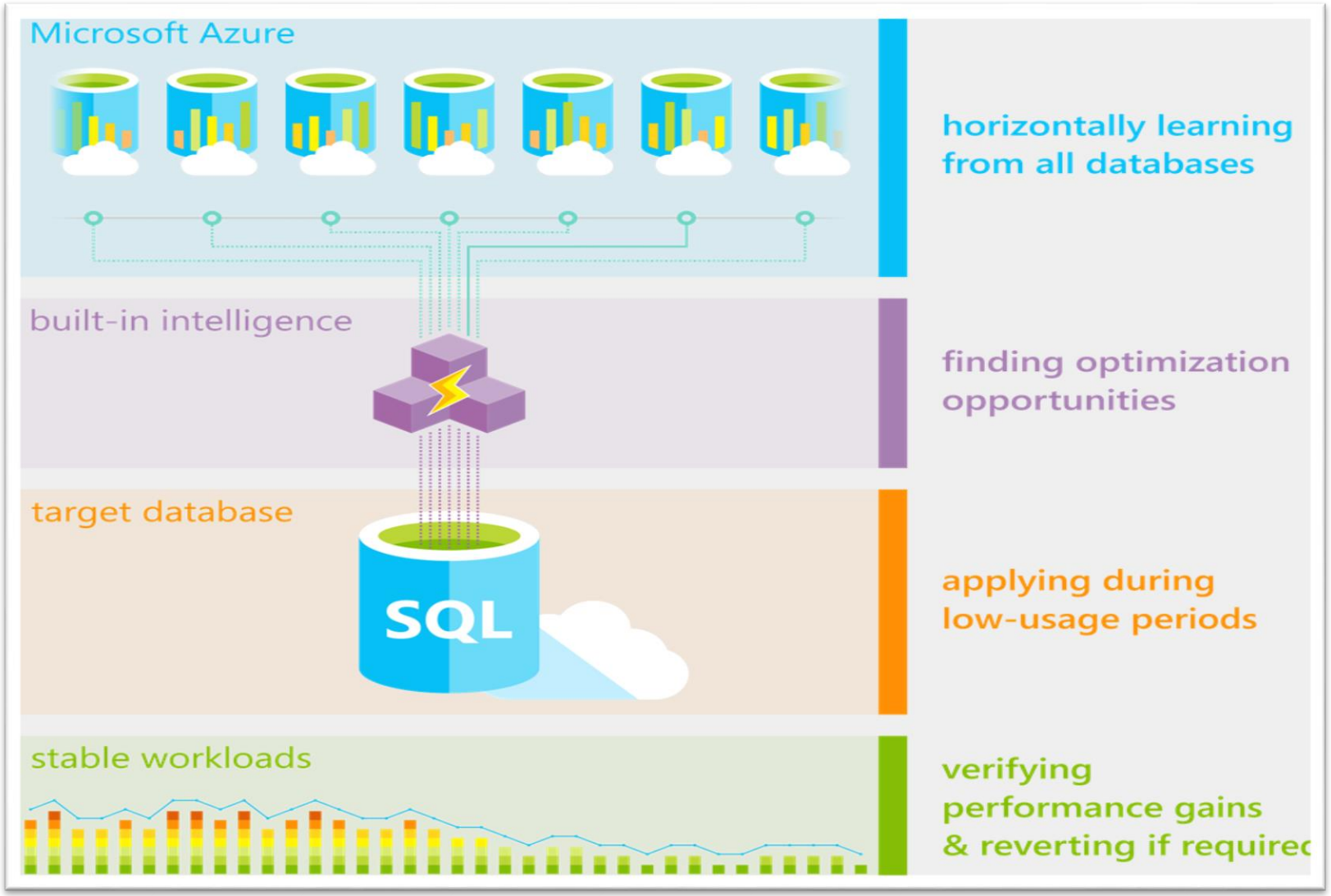
Objectives

After completing this learning, you will be able to:

- Know how Performance Recommendations can help to improve database performance.

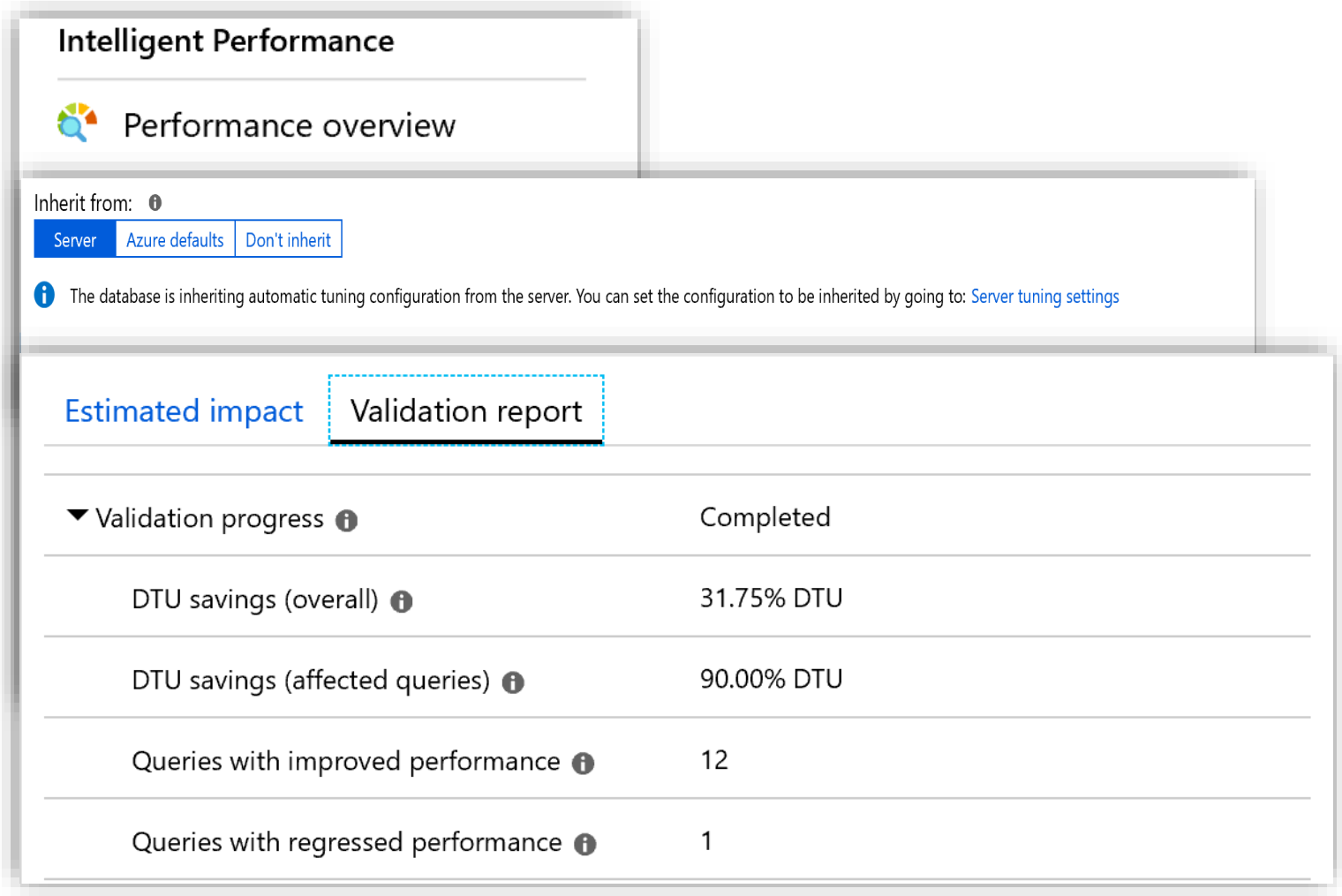


Automatic Tuning



[Performance recommendations for SQL Database]

Intelligent Performance – Automatic Tuning



http://automaticplancorrectiondemo.azurewebsites.net/index.html

Force Last Good Plan:

- Identifies regressed queries due to bad plan and replaces the bad plan with last Good Plan, validates performance improvements and reverts the change if performance does not improve.

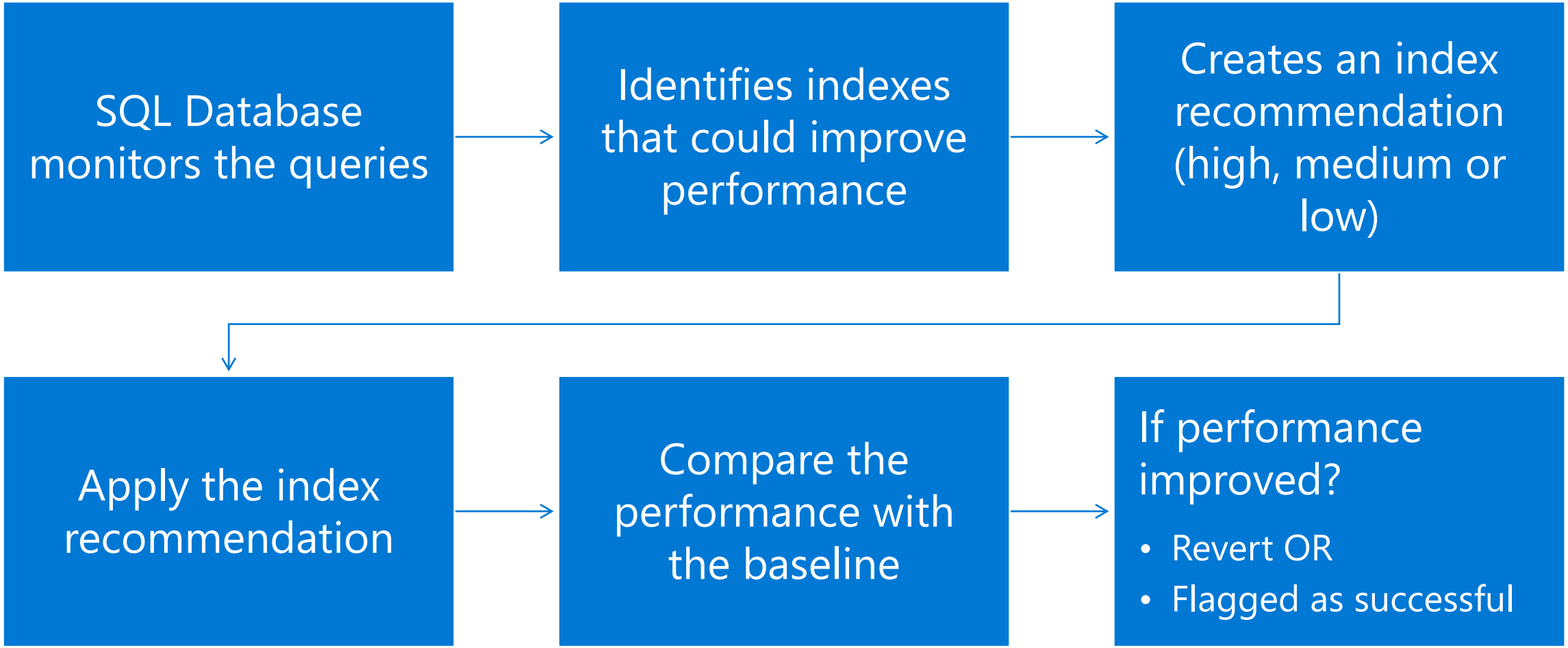
Create Index:

- Identifies and creates Indexes, validates performance improvements and reverts the change if performance degrades.

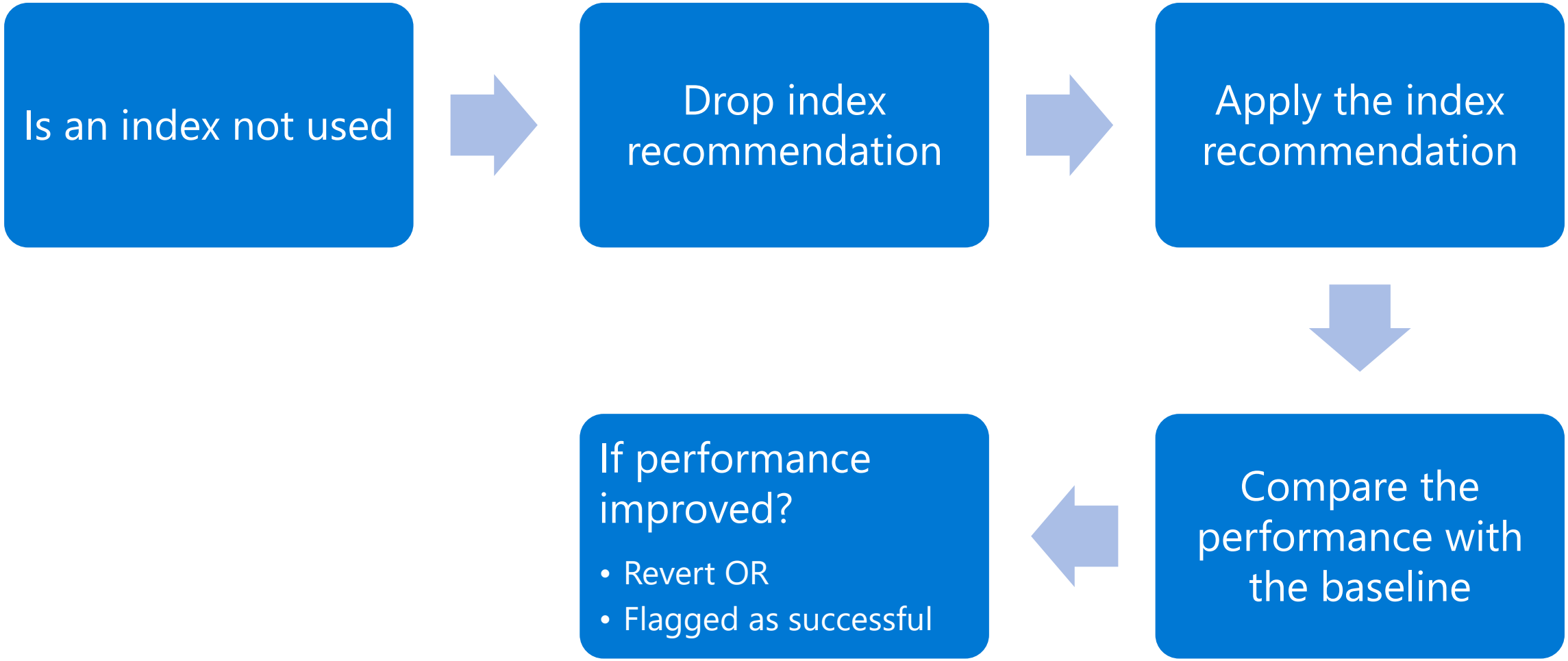
Drop Index:

- Identifies and drops unused Indexes, validates performance improvements and reverts the change if performance degrades.

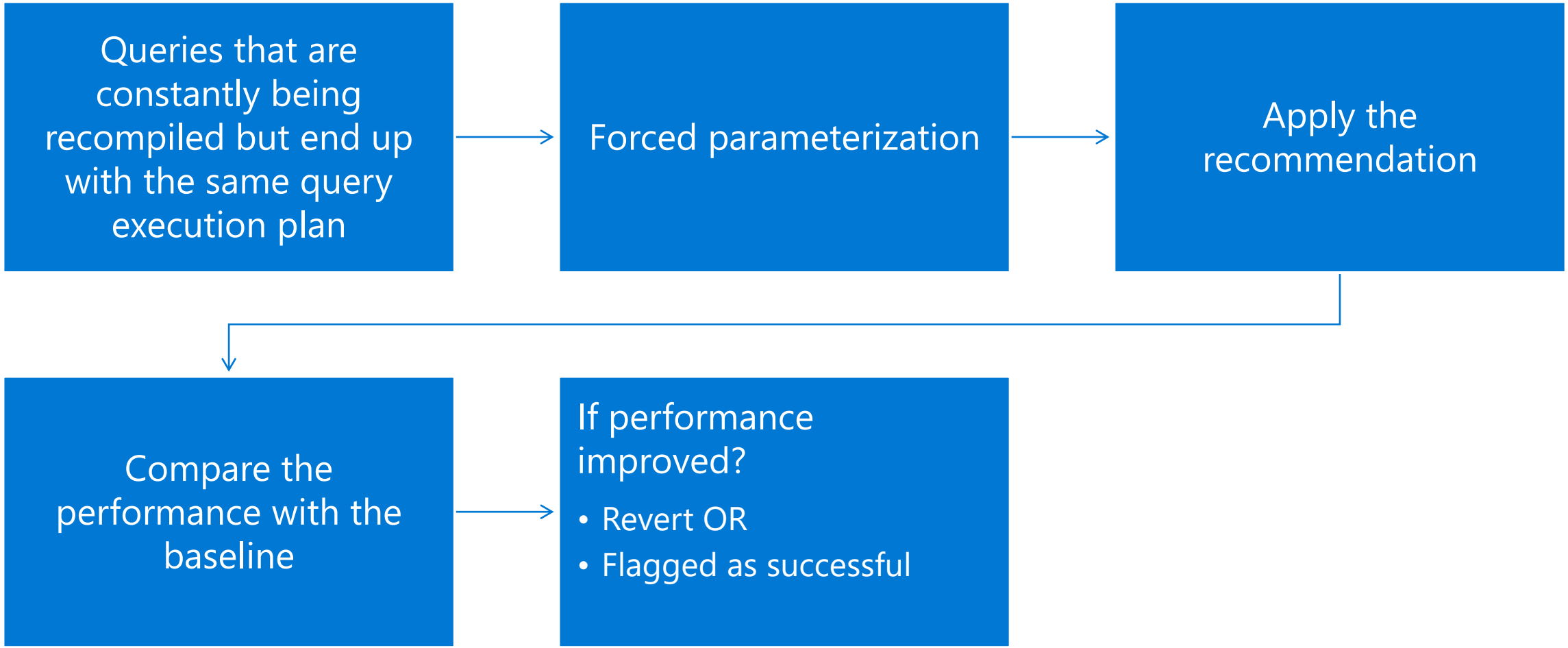
Automatic Tuning – Create Index



Automatic Tuning – Drop Index



Automatic Tuning – Parameterize Queries



Questions?

Knowledge Check

List three types of recommendations from Automatic Tuning.

What could be a reason to disable the automatic tuning option?

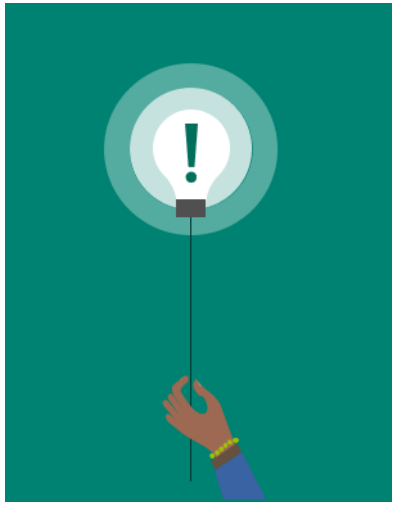
What technology is used for Automatic Tuning?

Lesson 3: Using SSMS Built-In Reports

Objectives

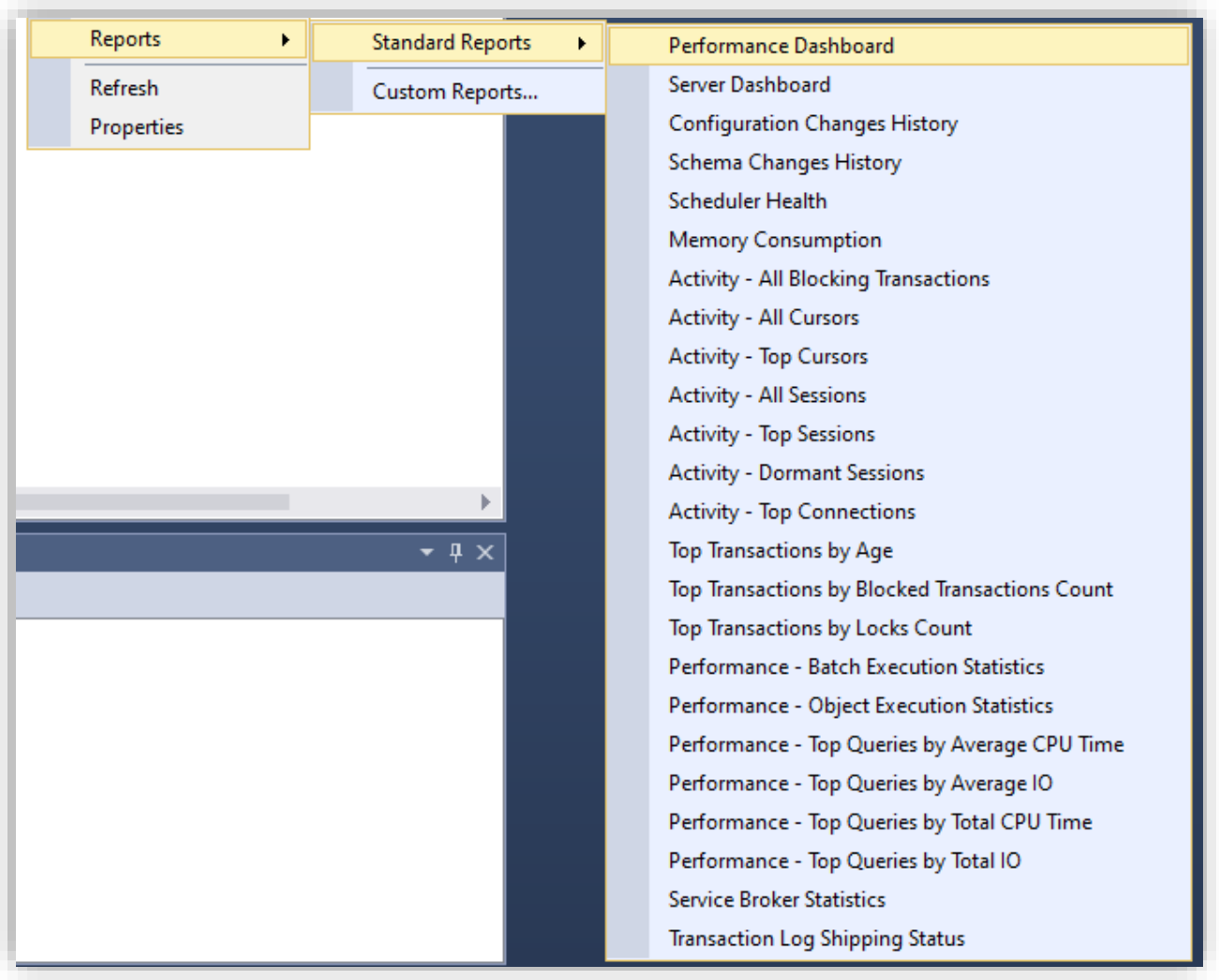
After completing this learning, you will be able to:

- Understand Built-In Instance Reports and Database Reports



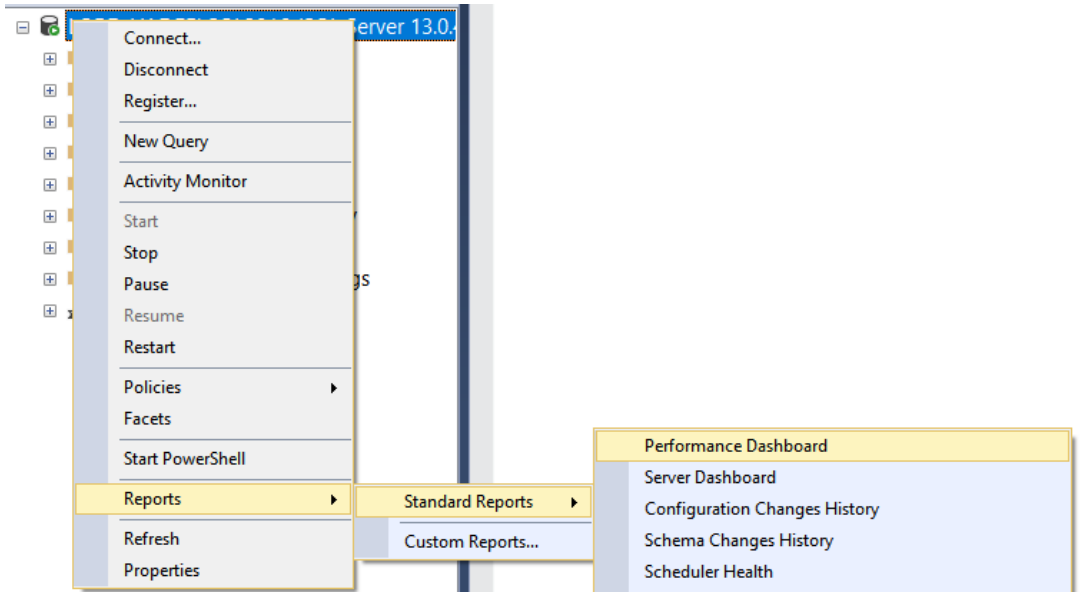
Instance Reports

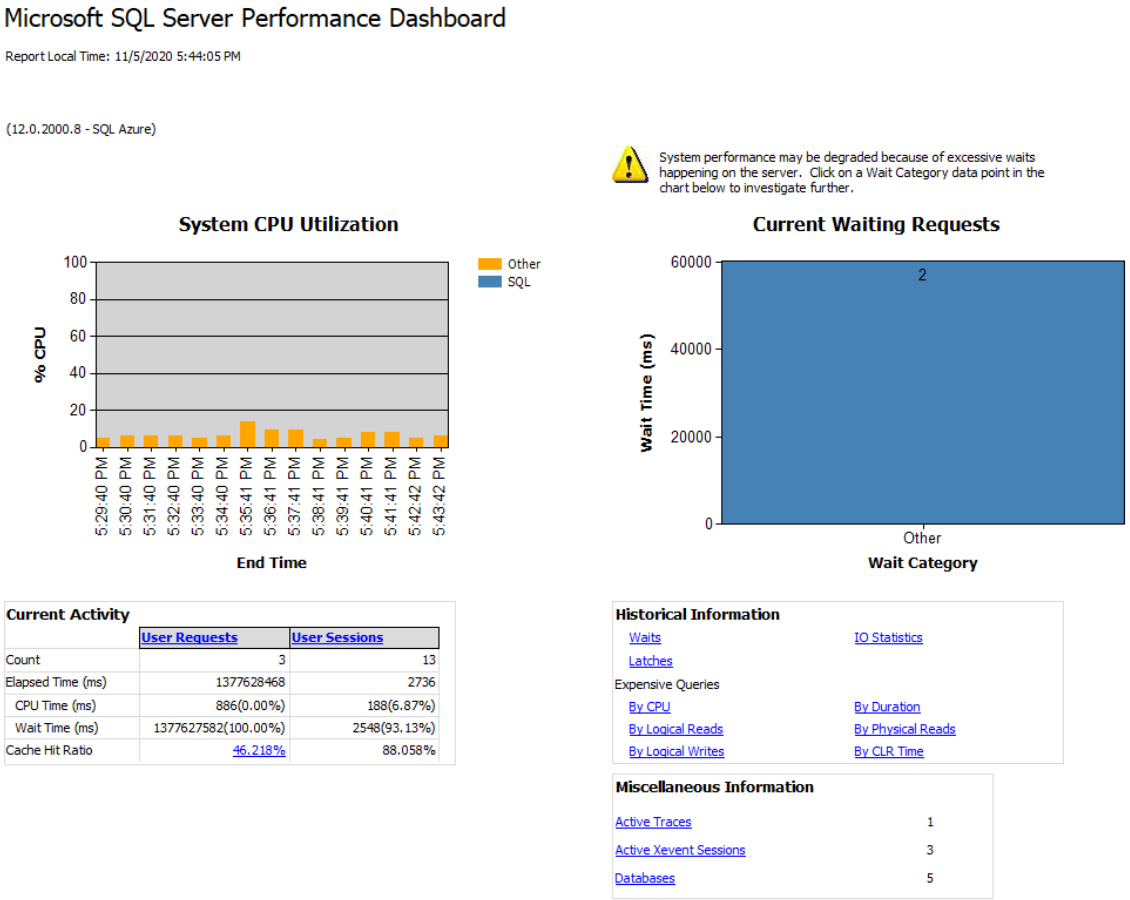
Built-in Instance Reports



Performance Dashboard

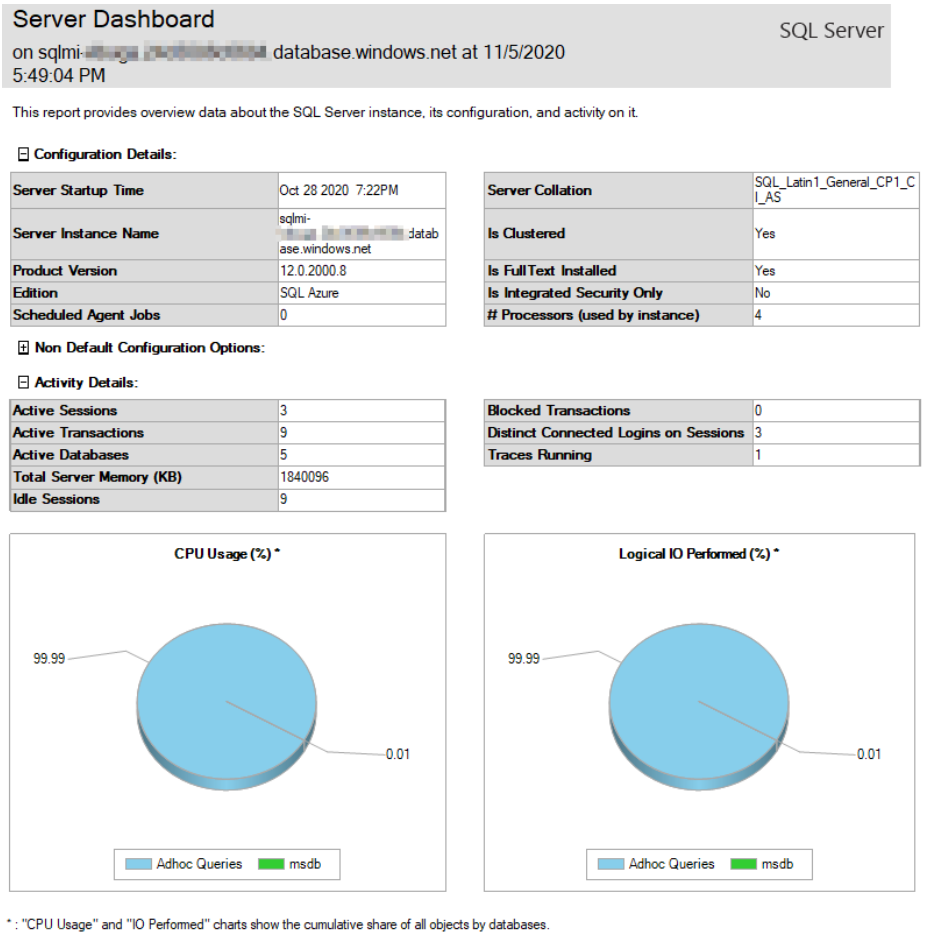
- To view the Performance Dashboard, right-click on the SQL Server instance name in Object Explorer, select **Reports**, **Standard Reports**, and click on **Performance Dashboard**.





Server Dashboard

- Report provides overview data about SQL Server Instance, configuration and activity on it.
  - Configuration Details
    - SQL Startup Time, Instance Name, Product Version
    - SQL Collation
    - Is Clustered, Is Integrated Security Only
    - # Processors
  - Non-Default Configuration Options
    - Traceflag, Run Value, Default Value
  - Activity Details
    - Active Sessions, Transaction, Databases
    - Total Server Memory, Idle Sessions
    - Blocked Transactions



Server Dashboard

SQL Server

on sqlmi- database.windows.net at 11/5/2020 5:49:04 PM

This report provides overview data about the SQL Server instance, its configuration, and activity on it.

Configuration Details:

Server Startup Time	Oct 28 2020 7:22PM	Server Collation	SQL_Latin1_General_CP1_CI_AS
Server Instance Name	sqlmi- .database.windows.net	Is Clustered	Yes
Product Version	12.0.2000.8	Is FullText Installed	Yes
Edition	SQL Azure	Is Integrated Security Only	No
Scheduled Agent Jobs	0	# Processors (used by instance)	4

Non Default Configuration Options:

Activity Details:

Active Sessions	3	Blocked Transactions	0
Active Transactions	9	Distinct Connected Logins on Sessions	3
Active Databases	5	Traces Running	1
Total Server Memory (KB)	1840096		
Idle Sessions	9		

* : "CPU Usage" and "IO Performed" charts show the cumulative share of all objects by databases.

Configuration Change History

Global Trace Flags – Configuration Options

- Provides a history of instance configuration and trace flag changes
- Reads history from **Default Trace**

Configuration Changes History

SQL Server

on [redacted].database.windows.net at 11/19/2020 1:21:24 PM

This report provides a history of all sp_configure and Trace Flag changes recorded by the Default Trace.

Configuration Changes History (Since 11/19/2020 1:21:10 PM).

Shows changes in server configuration and flags.

Configuration Option	Old Value	New Value	Time	User
Trace Flag (11024, -1)	--	on	11/19/2020 1:21:10 PM	AzureAdmin

Schema Change History

DDL Statement Commits

- Provides a history of schema changes

Schema Changes Hi...abase.windows.net | Schema Changes Hi...abase.windows.net

Schema Changes History

on [redacted].database.windows.net at 11/19/2020 4:52:30 PM

SQL Server

This report provides a history of all committed DDL statement executions recorded by the default trace.

Schema Change History (Since 11/19/2020 4:51:08 PM).

Shows changes made in the schema of the objects by DDL operations.

Database Name	Object Name	Type	DDL Operation	Time	Login Name
6cc8648e-d451-42ca-a103-dbcf85ab3dcb	Table_1	User Defined Table	ALTER	11/19/2020 4:52:21 PM	AzureAdmin
6cc8648e-d451-42ca-a103-dbcf85ab3dcb	Table_1	User Defined Table	CREATE	11/19/2020 4:52:21 PM	AzureAdmin

Scheduler Health

- Provides CPU consumption

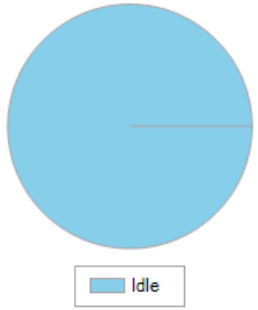
Scheduler Health

SQL Server

on sqlmi-vibuga.24d3086c6684.database.windows.net at 11/19/2020 5:10:07 PM

This report provides detailed activity data on each of the Instance's Schedulers.

Scheduler Status



Scheduler Details

Details of the workers, tasks & processes running under particular scheduler.

Scheduler ID	Status	CPU ID	# Preemptive Switches	# Context Switches	# Idle Switches	# Tasks	# Workers	Queue Length	# Pending IO Requests	Load Factor
⊞ 0	Idle	0	59,140	2,848,793	2,961,260	23	30	0	0	24
⊞ 1	Idle	1	529,526	7,295,761	8,924,272	22	29	0	0	23
⊞ 2	Idle	2	585,168	6,925,036	8,834,475	19	27	0	0	22
⊞ 3	Idle	3	25,522	4,881,099	5,015,779	23	30	0	0	23

Memory Consumption

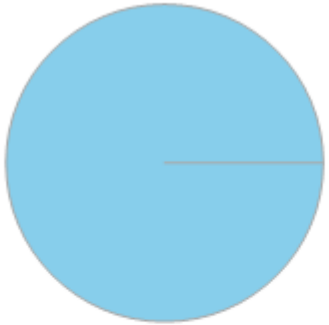
Consumption Components and Historical Data

- Provides memory consumption
- Reads history from **Default Trace**

Memory Grants Outstanding	0
Memory Grants Pending	0

Page life expectancy	1896786

Buffer Pages Distribution (# Pages)



Dirty (2058) Latched (0)

Top Memory Consuming Components

CACHESTORE_SQLCP (1075816 KB)
MEMORYCLERK_XE_BUFFER (672960 KB)
MEMORYCLERK_SQLBUFFERPOOL (231864 KB)
CACHESTORE_PHDR (143224 KB)
MEMORYCLERK_SOSNODE (118136 KB)
Others (472824 KB)
MEMORYCLERK_XTP (15672 KB)

Memory Changes Over Time (Last 7 Days)

There are no major changes in memory consumption or default trace is not enabled

Activity and Top Transaction Reports

Activity Reports

- All Blocking Transactions
- All Cursor
- Top Cursor
- All Sessions
- Top Sessions
- Dormant Sessions
- Top Connections

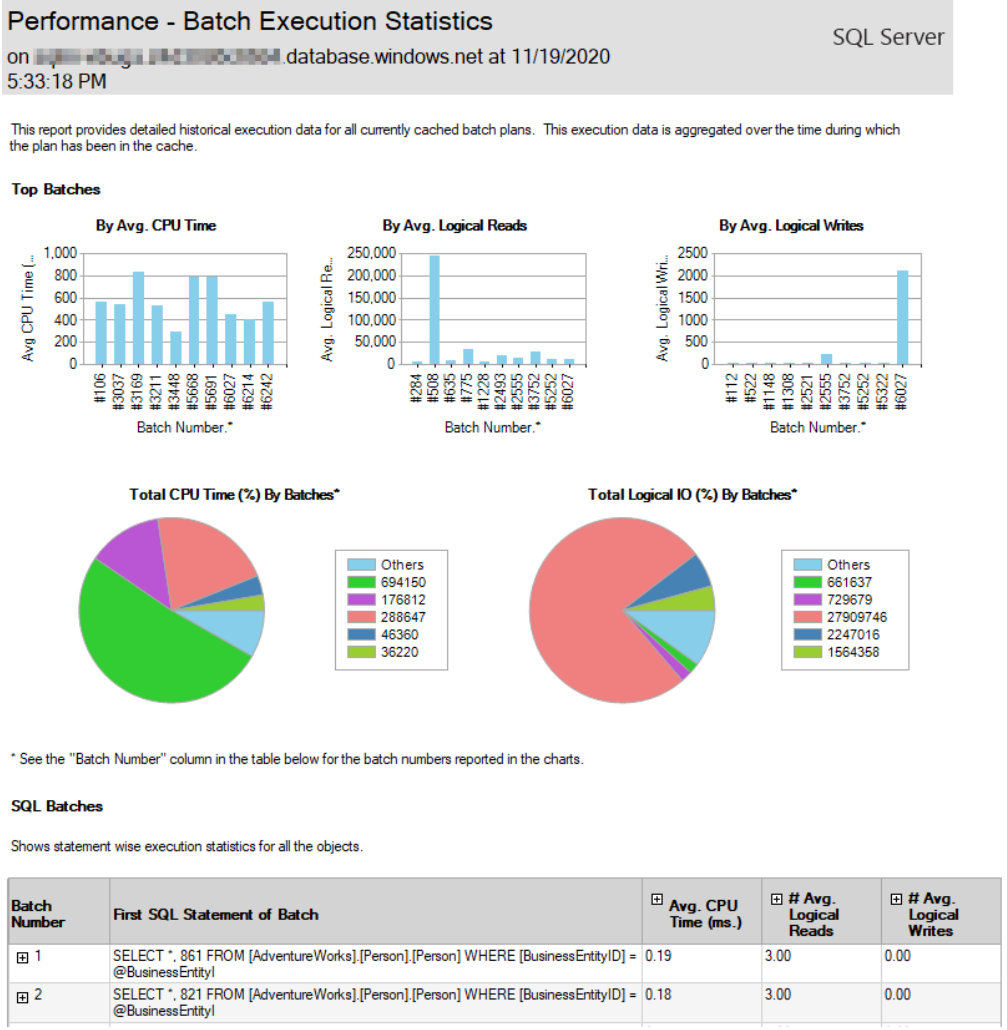
Top Transaction Reports

- Top Transaction by Age
- Top Transaction by Blocked Transaction Count
- Top Transactions by Locks Count

Performance Reports

Batch Execution Statistics

- Provides execution history data for all cached batch plans.



Performance - Batch Execution Statistics

on [redacted].database.windows.net at 11/19/2020 5:33:18 PM

SQL Server

This report provides detailed historical execution data for all currently cached batch plans. This execution data is aggregated over the time during which the plan has been in the cache.

Top Batches

* See the "Batch Number" column in the table below for the batch numbers reported in the charts.

SQL Batches

Shows statement wise execution statistics for all the objects.

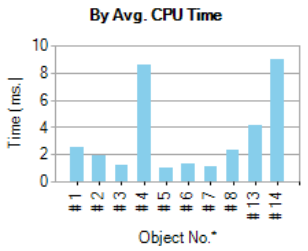
Batch Number	First SQL Statement of Batch	Avg. CPU Time (ms.)	# Avg. Logical Reads	# Avg. Logical Writes
1	SELECT *, 861 FROM [AdventureWorks].[Person].[Person] WHERE [BusinessEntityID] = @BusinessEntityI	0.19	3.00	0.00
2	SELECT *, 821 FROM [AdventureWorks].[Person].[Person] WHERE [BusinessEntityID] = @BusinessEntityI	0.18	3.00	0.00

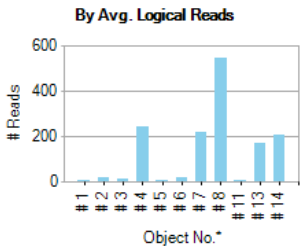
Performance Reports

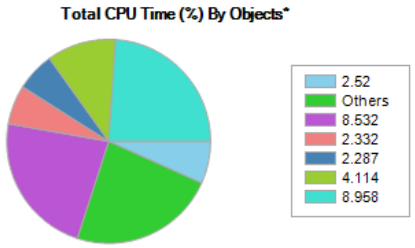
Object Execution Statistics

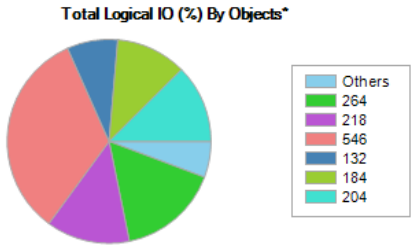
- Provides execution history data for all cached plans.

Top Executable Objects









All Executable Objects

Shows statement wise execution statistics for all the executable objects.

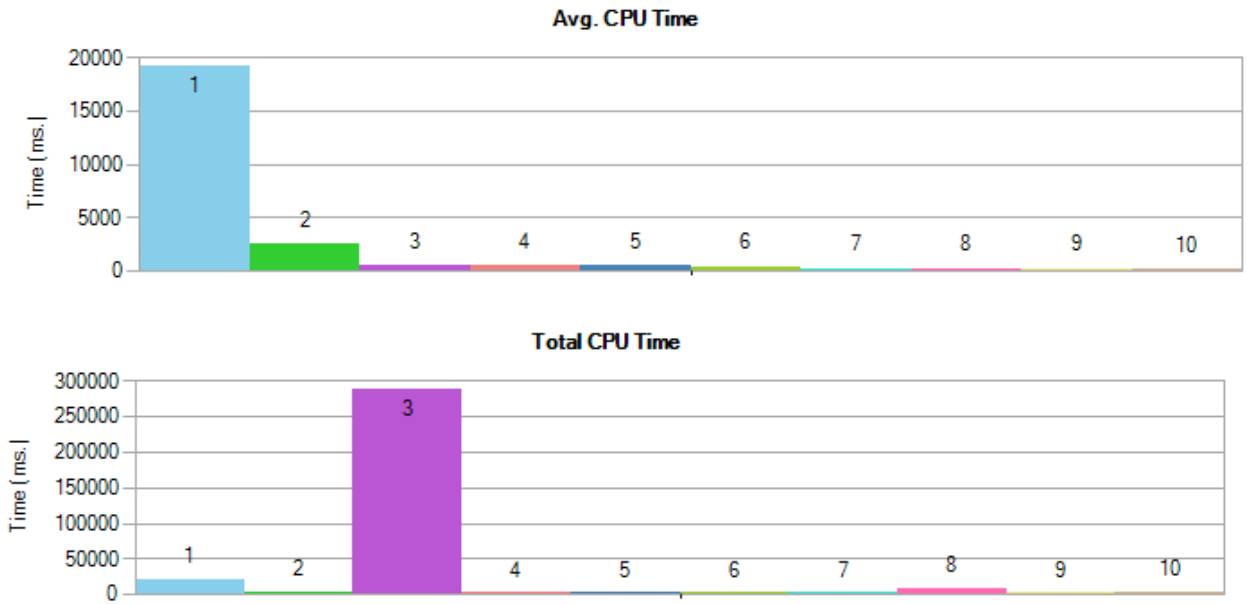
Object No.*	Database Name	Object Name	Object Type	Avg. CPU Time (ms.)	Total CPU Time (%)	# Avg. Logical Reads	# Avg. Logical Writes	# Avg. Logical IO	Total Logical IO (%)
1	msdb	managed_backup.fn_backup_db_config	SQL Table-valued-Function	2.52	6.73	4.00	0.00	4.00	0.24

SQL Statement	# Executions (With Last Plan)	# Plans Generated	Avg. CPU Time (ms.)		# Avg. Logical Reads	# Avg. Logical Writes	# Avg. Logical IO	
INSERT INTO @t SELECT aamd.db_name, aamd.db_guid, CASE WHEN aamd.group_db_guid IS NULL THEN CONVERT(BIT, 'False')	1	1	2.52		4.00	0.00	4.00	

Performance Reports

Top Queries by Average CPU Time

- Provides top queries by average CPU time for all cached plans.

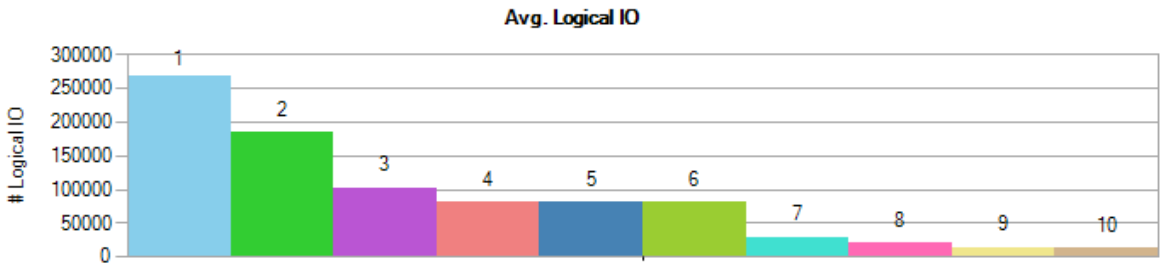


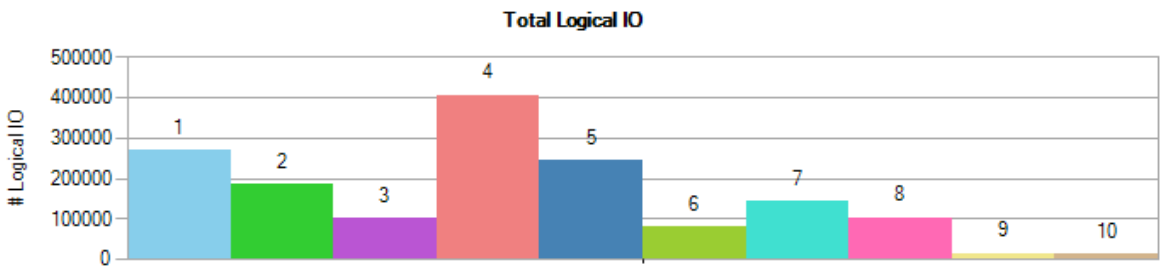
Query No.	Query Text	Database Name	Object ID	Avg. CPU Time (ms.)
1	select (s.t_SPRank)%2 as l1 , (dense_rank() over(order by s.t_SPRank,s.row_id))%2 as l2 , s.* , ob.cpu_rank as t_CPURank , ob.read_rank as t_ReadRank , ob.write_rank as t_WriteRank from @sql_handle_convert_table s join @objects ob on (s.t_SPRank = ob.obj_rank			19,204.53
2	insert into @sql_handle_convert_table Select sql_handle , sql_handle as chart_display_option , sql_handle as chart_display_optionIO , master.dbo.fn_varbintohexstr(sql_handle) , dense_rank() over (order by s1.sql_handle) as SPRank , dense_rank() over (partition by s1.sql_handle order by s1.statement_start_offset) as SPRank2 , (select top 1 substring(text,(s1.statement_start_offset+2)/2, (case when s1.statement_end_offset = -1 then len(convert (nvarchar(max),text))*2 else s1.statement_end_offset end - s1.statement_start_offset) /2) from sys.dm_exec_sql_text (s1.sql_handle)) as [SQL Statement] , execution_count , plan_generation_num , last_execution_time , ((total_worker_time+0.0)/execution_count)/1000 as [avg_worker_time] , total_worker_time/1000 , last_worker_time/1000 , min_worker_time/1000 , max_worker_time/1000 , ((total_logical_reads+0.0)/execution_count) as			2,552.09

Performance Reports

Top Queries by Average IO

- Provides top queries by average IO for all cached plans.





Query No.	Query Text	Database Name	Object ID	# Avg. Logical IO
1	select (s.t_SPRank)%2 as l1 , (dense_rank() over(order by s.t_SPRank,s.row_id))%2 as l2 , s.* , ob.cpu_rank as t_CPURank , ob.read_rank as t_ReadRank , ob.write_rank as t_WriteRank from @sql_handle_convert_table s join @objects ob on (s.t_SPRank = ob.obj_rank			268,351.00
2	select top 10 rank() over(order by (total_worker_time +0.0)/execution_count desc,sql_handle,statement_start_offset) as row_no , (rank() over(order by (total_worker_time +0.0)/execution_count desc,sql_handle,statement_start_offset))%2 as l1 , creation_time , last_execution_time , (total_worker_time+0.0)/1000 as total_worker_time , (total_worker_time+0.0)/(execution_count*1000) as [AvgCPUTime] , total_logical_reads as [LogicalReads] , total_logical_writes as [LogicalWrites] , execution_count , total_logical_reads+total_logical_writes as [AggIO] , (total_logical_reads+total_logical_writes)/(execution_count +0.0) as [AvgIO] , case when sql_handle IS NULL			184,550.00

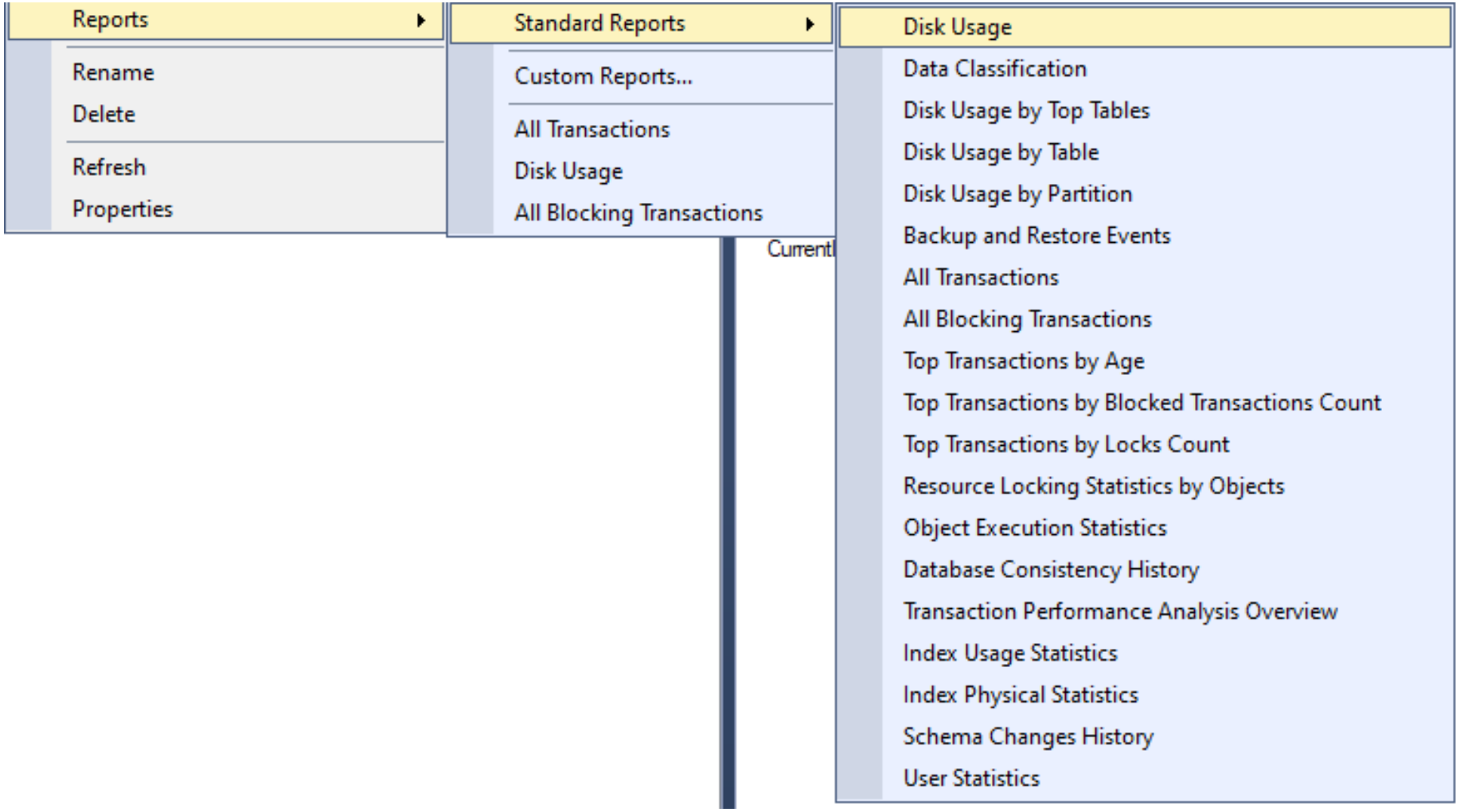
Demonstration

Server Reports

- SSMS Built-In Server Reports

Database Reports

Built-In Database Reports



Demonstration

Database Reports

- SSMS Built-In Database Reports

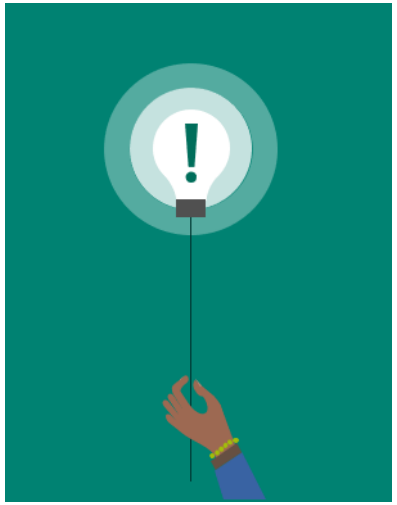
Questions?

Lesson 4: Using Metrics and Alerts

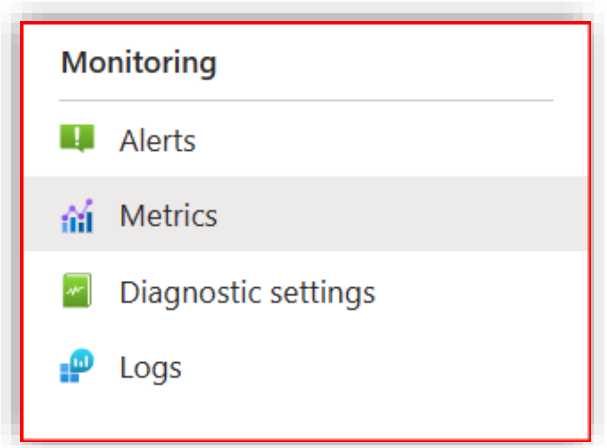
Objectives

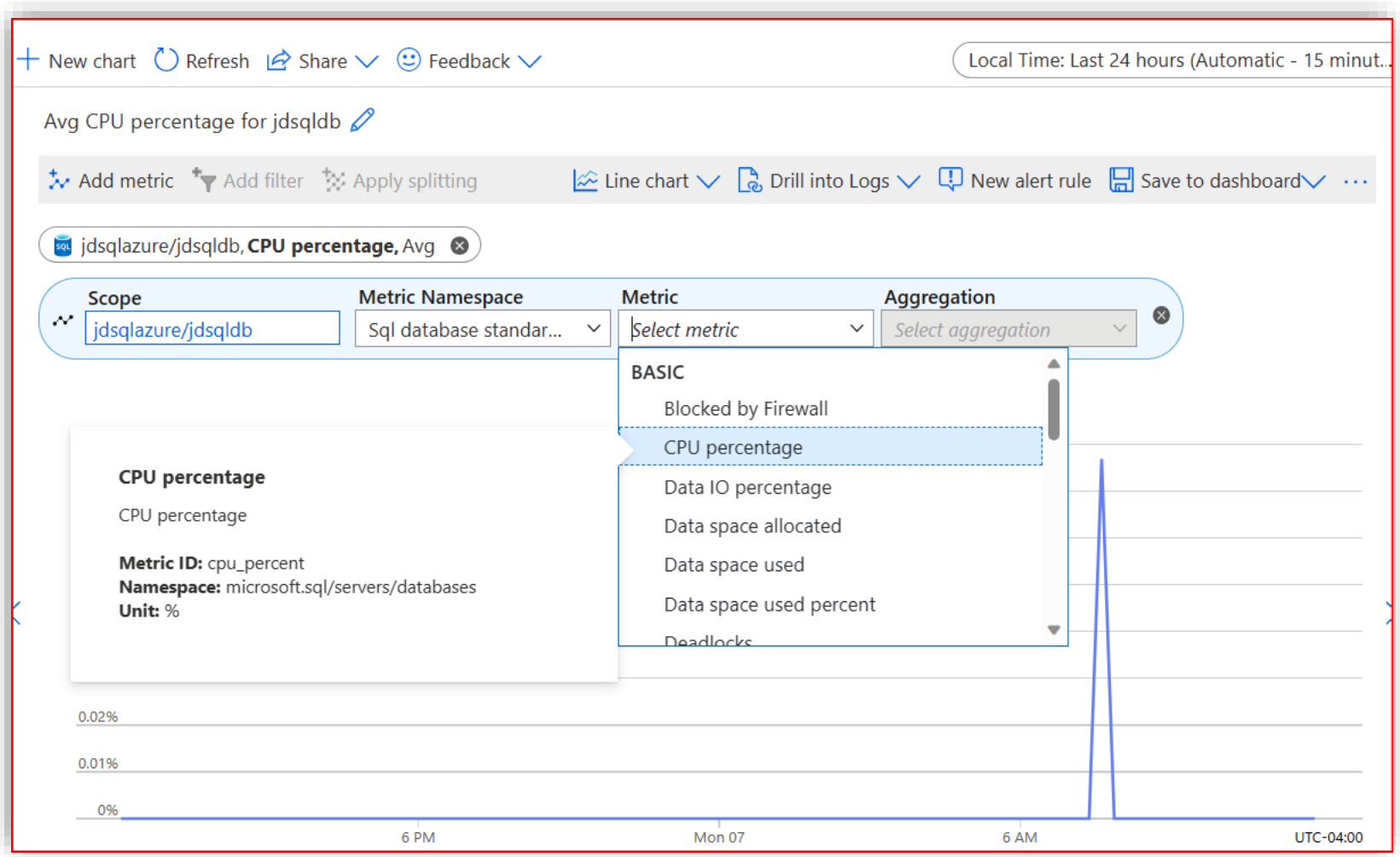
After completing this learning, you will be able to:

- Configure alerts using Azure Management Portal.



Metrics and Alerts





- Metrics enable you to see if a database is approaching the limits of CPU, memory, IO, or storage resources.
- High DTU, CPU or IO utilization may indicate that your workload needs more resources.

Purpose of Alerts for Azure SQL Database

Database alerts can help to proactively trigger various events related to database connectivity, high DTU usage or deadlocks, etc.

It helps to proactively resolve underlying issues to avoid application outages and improve user experience.

Receiving an alert based on monitoring metrics or events on

Metric values

- The alert triggers when the value of a specified metric crosses a threshold you assigned in either direction. It triggers when the condition is first met and then when that condition is no longer being met.

Activity log events

- An alert can trigger on every event, or, only when a certain number of events occur.

Purpose of Alerts for Azure SQL Database

You can configure an alert to do the following when it triggers:

- Send email notifications to the service administrator and co-administrators.
- Send email to additional emails that you specify.
- Call a webhook

You can configure and get information about alert rules using

- Azure portal
- PowerShell
- command-line interface (CLI).
- Azure Monitor REST API.

SQL Database alert values

Metric Name	Aggregation Type	Minimum Alert Time Window
CPU percentage	Average	5 minutes
Data IO percentage	Average	5 minutes
Log IO percentage	Average	5 minutes
DTU percentage	Average	5 minutes
Total database size	Maximum	30 minutes
Successful Connections	Total	10 minutes
Failed Connections	Total	10 minutes
Blocked by Firewall	Total	10 minutes
Deadlocks	Total	10 minutes
Database size percentage	Maximum	30 minutes
In-Memory OLTP storage percent(Preview)	Average	5 minutes
Workers percentage	Average	5 minutes
Sessions percent	Average	5 minutes
DTU limit	Average	5 minutes
DTU used	Average	5 minutes

Dankie Faleminderit **Shukran** Chnorakaloutioun Hvala Blagodaria

Děkuji **Tak** Dank u Tänan Kiitos **Merci** Danke Ευχαριστώ A dank

Mahalo תודה. **Dhanyavād** Köszönöm Takk Terima kasih **Grazie** Grazzi

Thank you!

감사합니다 Paldies Choukrane Ačiū **Благодарам** ありがとうございました

谢谢 Баярлалаа **Dziękuję** Obrigado Mulţumesc **Спасибо** Ngiyabonga

Ďakujem Tack Nandri Kop khun **Teşekkür ederim** Дякую Хвала Diolch

Demonstration

Configure Alerts through Azure Portal

- Configure alerts through Azure Portal.

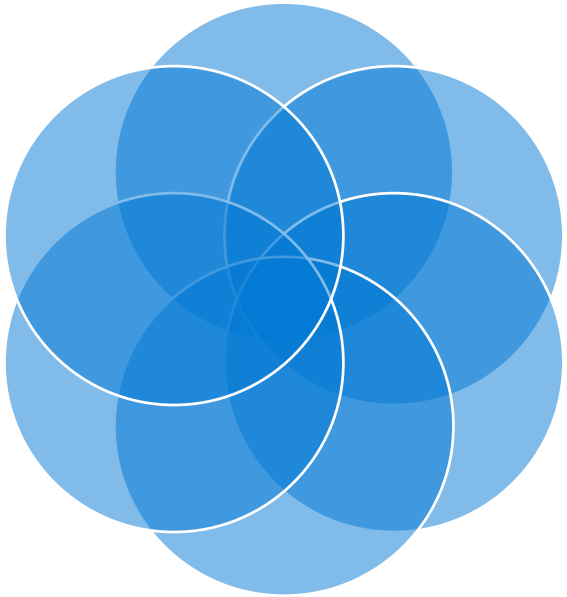
Questions?

Module Summary

Monitoring and Troubleshooting Azure SQL Database

Configure Alerts through Azure Portal

Monitoring Query Performance using Query Performance Insight



Monitoring Azure SQL Database Performance using Extended Events

Azure SQL Database Tuning using Automatic Tuning

Monitoring Azure SQL Database Performance using Intelligent Insights

Microsoft