



# Azure SQL Performance Tuning

Module 7

## Learning Units covered in this Module

- Lesson 1: Query Performance Insights
- Lesson 2: Automatic Tuning in Azure SQL
- Lesson 3: Metrics and Alerts
- Lesson 4: Intelligent Insights
- Lesson 5: Built-in SSMS Reports (MI)

# Lesson 1: Query Performance Insights

# Objectives

After completing this learning, you will be able to:

- Install the Query Performance Insights library
- Troubleshoot performance issues with QPI



# Objectives

After completing this learning, you will be able to:

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



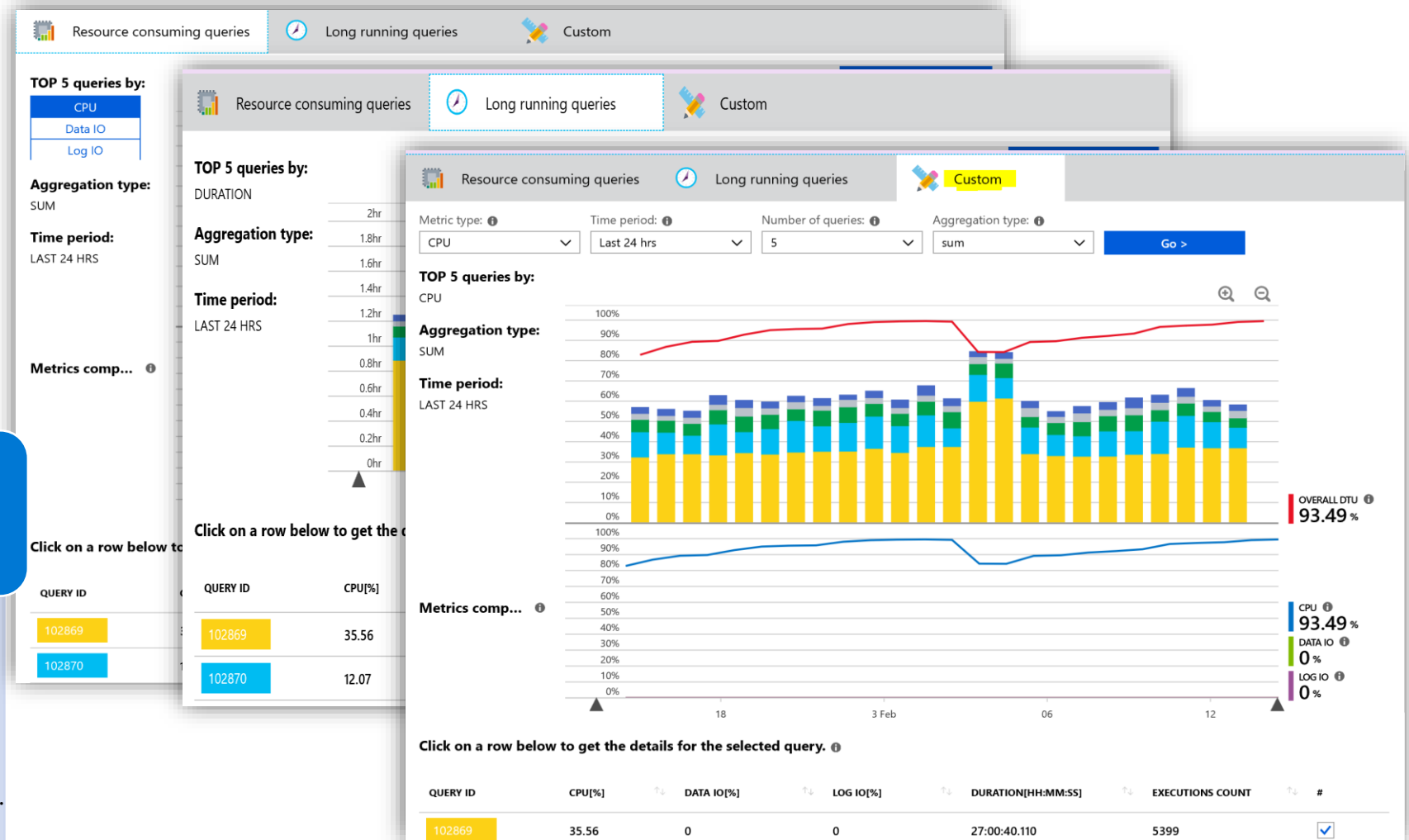
# Query Performance Insight

## Intelligent Performance

- Performance overview
- Performance recommendati...
- Query Performance Insight
- Automatic tuning

## Custom options – Insights based upon custom selection:

- Metric type – resource consuming queries, Long IO, Queries and Execution queries
- Time period – configuration: 24 hrs, Past Week, Past Month and Custom
- Can drill through the queries to see Query text, CPU, data IO and Log IO
- Number of Queries – 5, 10, 20
- Aggregation type – sum, max and avg.
- Log IO configuration – % Duration and Execution count.

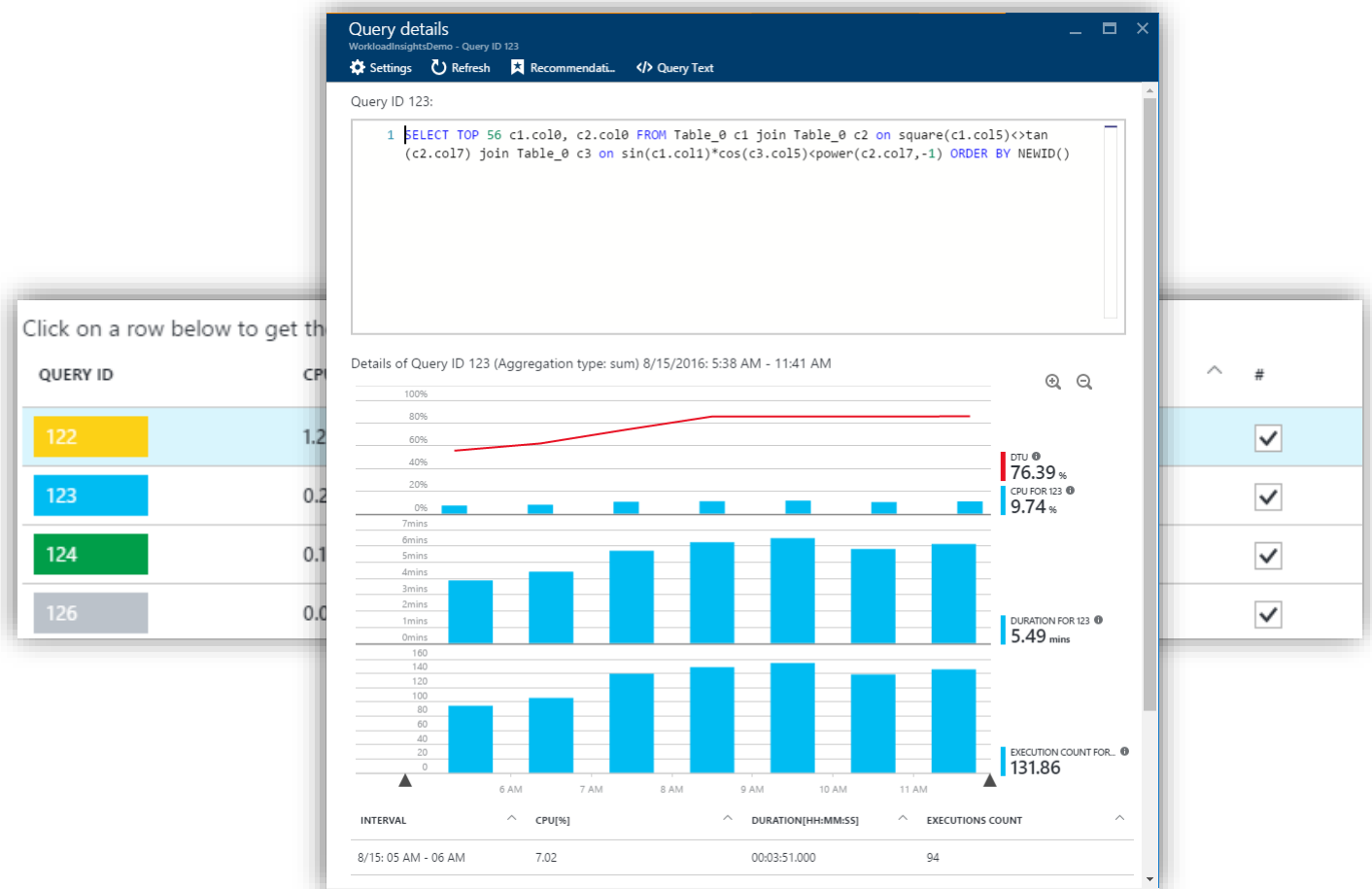


# Viewing individual query details

Get details for the individual queries

- CPU Consumption
- Duration
- Execution Count

It does not capture DDL queries

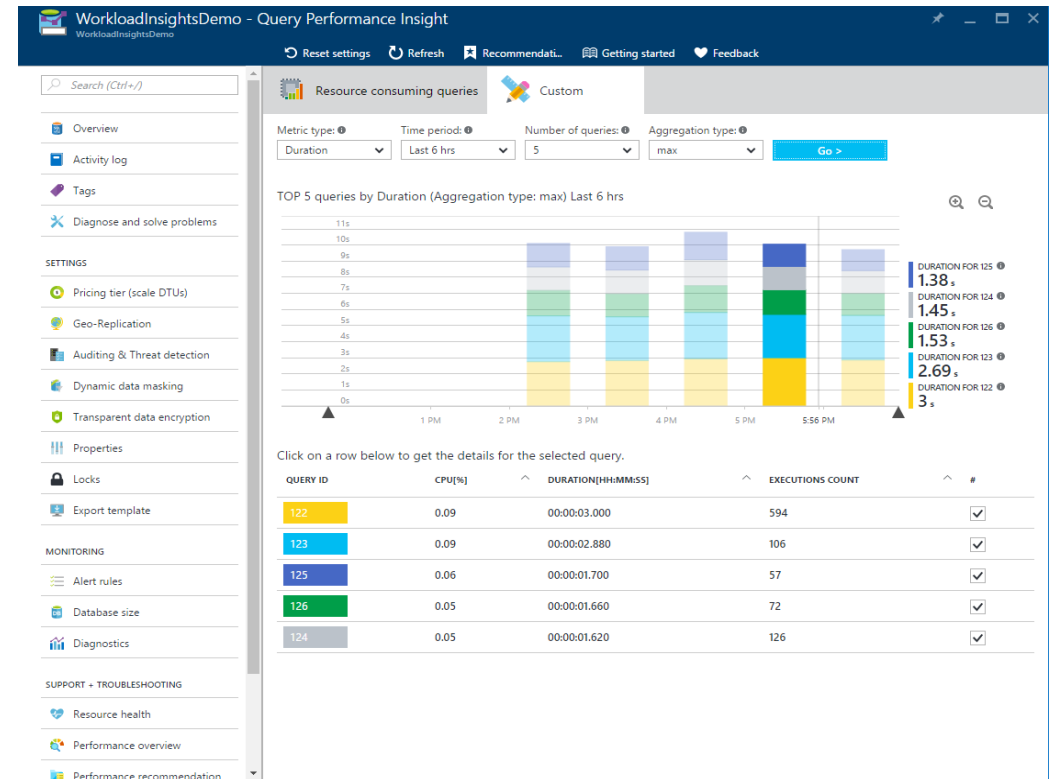


# Review top queries per duration

Duration is one of the metrics showing potential bottleneck

Long-running queries has potential for:

- Longer locks
- Blocking other users
- Limiting scalability



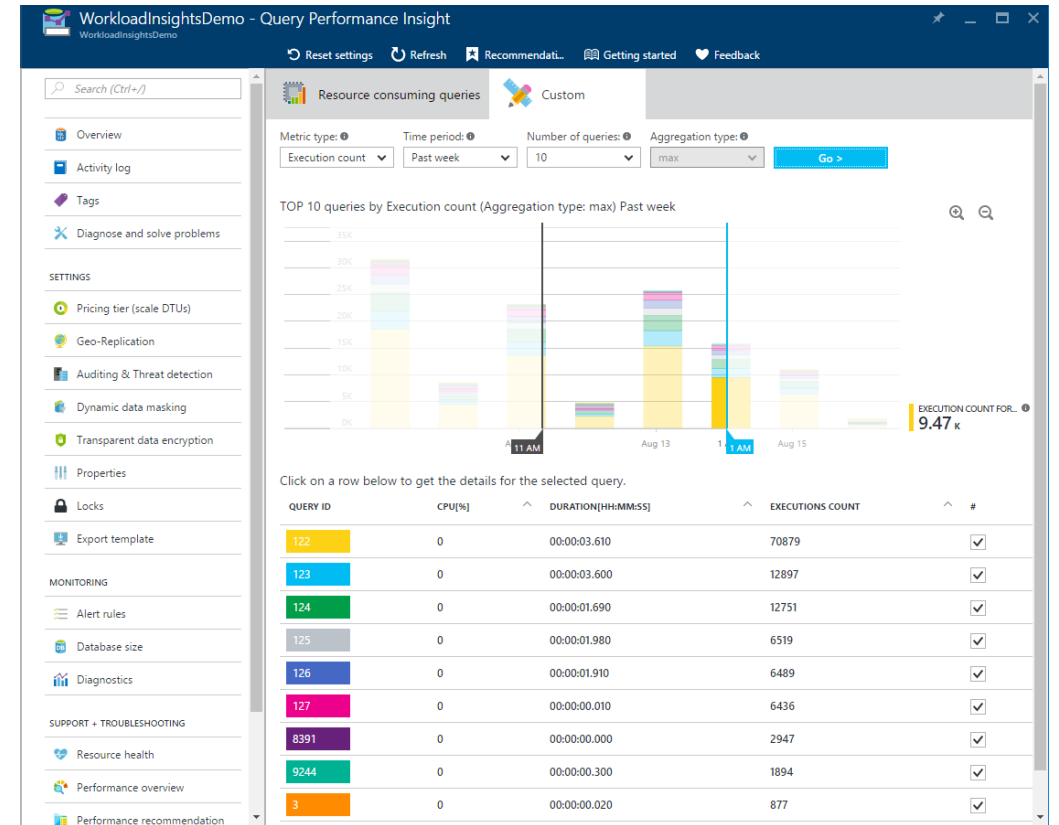


# Review top queries per execution count

Execution count is one of the metrics showing potential bottleneck

High number of executions has potential for:

- Database performance
- Network latency
- Downstream server latency



# Demonstration

## Query Performance Insight

- Analyze the Query Performance Insight output.



Questions?



## Lesson 2: Azure SQL Automatic Tuning

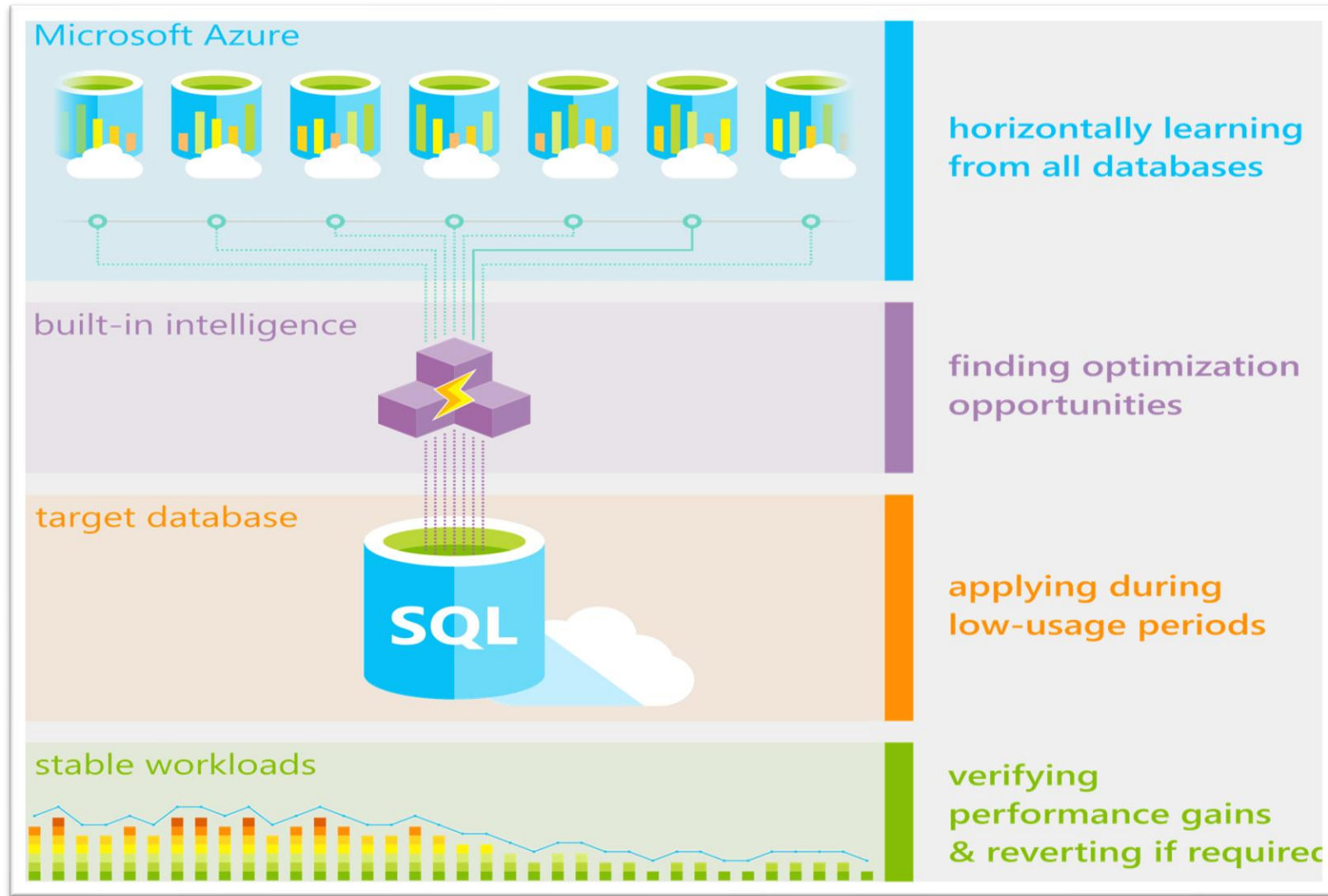
# Objectives

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






## Intelligent Performance

 Performance overview

Inherit from:  Server Azure defaults Don't inherit

 The database is inheriting automatic tuning configuration from the server. You can set the configuration to be inherited by going to: [Server tuning settings](#)

Estimated impact Validation report

▼ Validation progress 	Completed
DTU savings (overall) 	31.75% DTU
DTU savings (affected queries) 	90.00% DTU
Queries with improved performance 	12
Queries with regressed performance 	1

## 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:

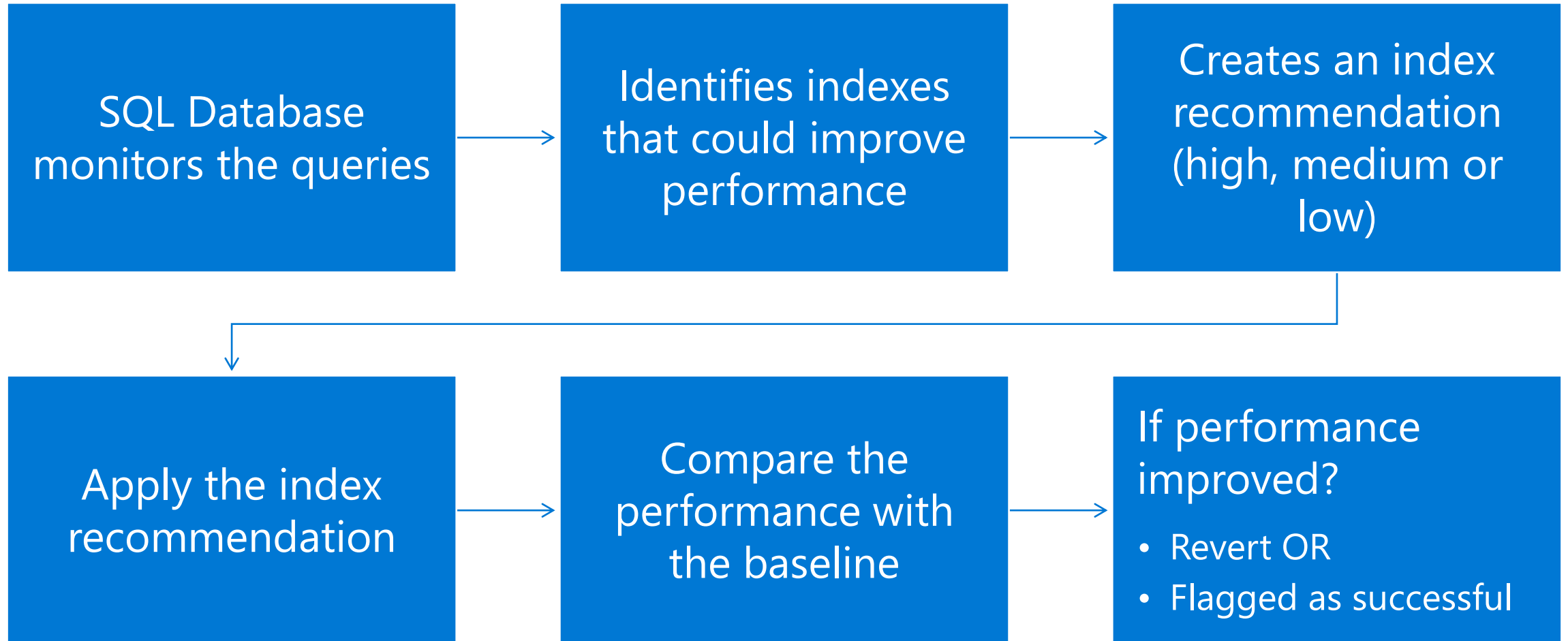
- Identifies and creates Indexes, validates performance improvements and reverts the change if performance degrades.

## Drop Index:

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

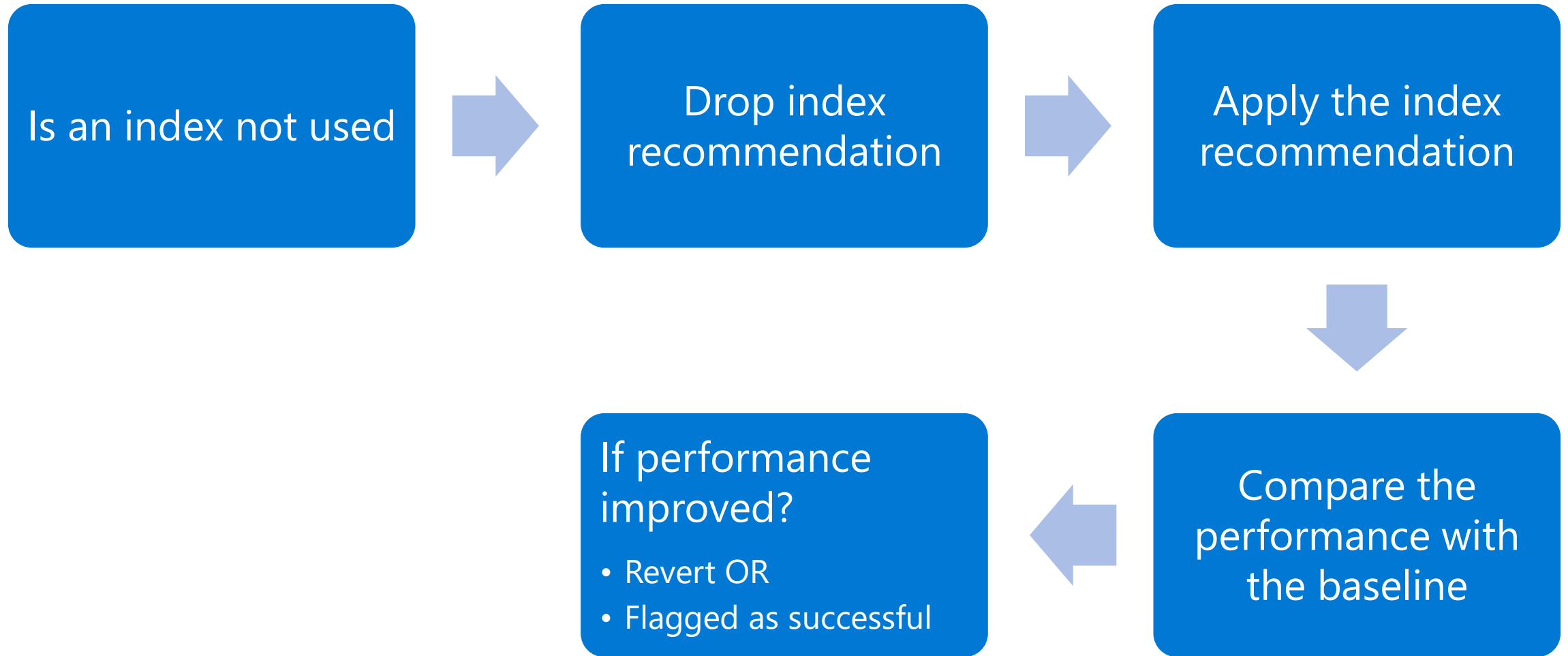
<http://automaticplan correctiondemo.azurewebsites.net/index.html>

# Automatic Tuning – Create Index

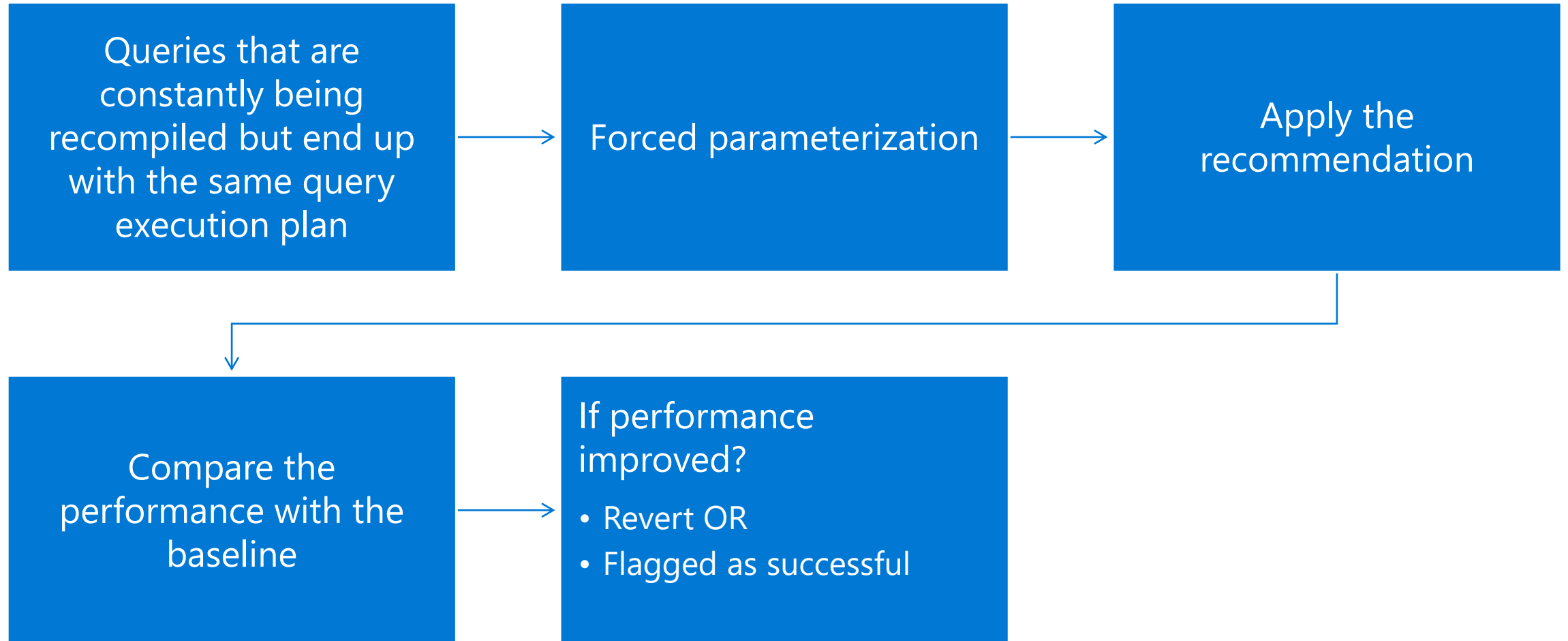




# Automatic Tuning – Drop Index



# Automatic Tuning – Parameterize Queries



Questions?



# Lesson 3: Metrics and Alerts through Azure Portal

# Objectives

After completing this learning, you will be able to:

- Configure alerts using Azure Management Portal.



# 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

# Lesson 4: Troubleshooting Azure SQL MI Performance with Intelligent Insights

# Objectives

After completing this learning, you will be able to:

- Describe and setup Intelligent Insights
- Consume and apply the Artificial Intelligence recommendations



# Activating the Power of Artificial Intelligence

Enable Azure SQL Analytics



Stream data to a data collector\*

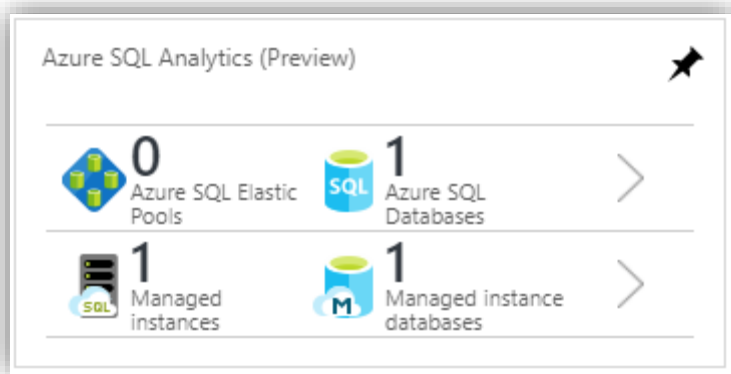


Detect database performance patterns

\* to [Azure Monitor logs](#), [Azure Event Hubs](#), [Azure Storage](#), or a third-party solution for custom DevOps alerting and reporting capabilities.

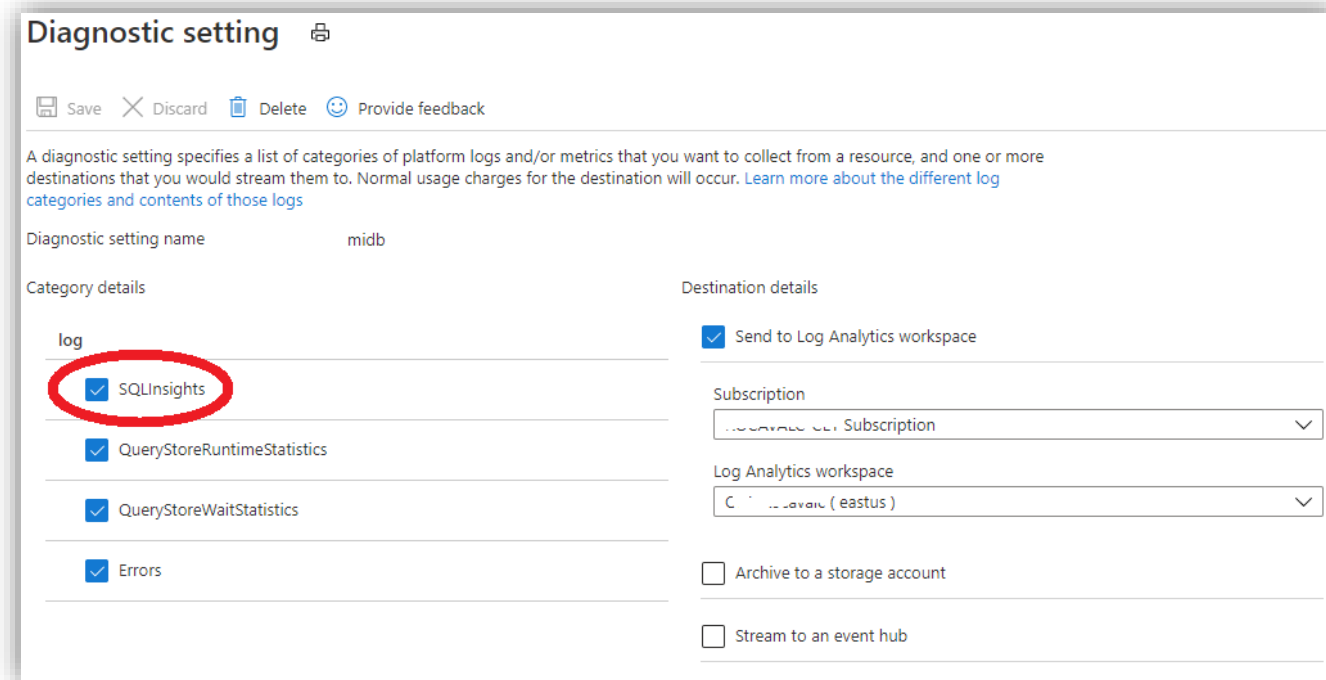
# Azure SQL Analytics


- Monitoring solution supporting streaming of diagnostics telemetry
- It provides hierarchical drill-down into Intelligent Insights into performance.







# Stream Data

- Configure Diagnostic settings of Azure SQL MI to stream to log analytics
- Push SQLInsights logs into a Log Analytics Workspace



**Diagnostic setting** 

 Save  Discard  Delete  Provide feedback

A diagnostic setting specifies a list of categories of platform logs and/or metrics that you want to collect from a resource, and one or more destinations that you would stream them to. Normal usage charges for the destination will occur. [Learn more about the different log categories and contents of those logs](#)

Diagnostic setting name: midb

Category details	Destination details
<p>log</p> <ul style="list-style-type: none"><li><input checked="" type="checkbox"/> SQLInsights</li><li><input checked="" type="checkbox"/> QueryStoreRuntimeStatistics</li><li><input checked="" type="checkbox"/> QueryStoreWaitStatistics</li><li><input checked="" type="checkbox"/> Errors</li></ul>	<ul style="list-style-type: none"><li><input checked="" type="checkbox"/> Send to Log Analytics workspace</li><li>Subscription: <input type="text" value="Subscription"/></li><li>Log Analytics workspace: <input type="text" value="C... (eastus)"/></li><li><input type="checkbox"/> Archive to a storage account</li><li><input type="checkbox"/> Stream to an event hub</li></ul>

# Lesson 5: Using SSMS Built-In Reports

# Objectives

After completing this learning, you will be able to:

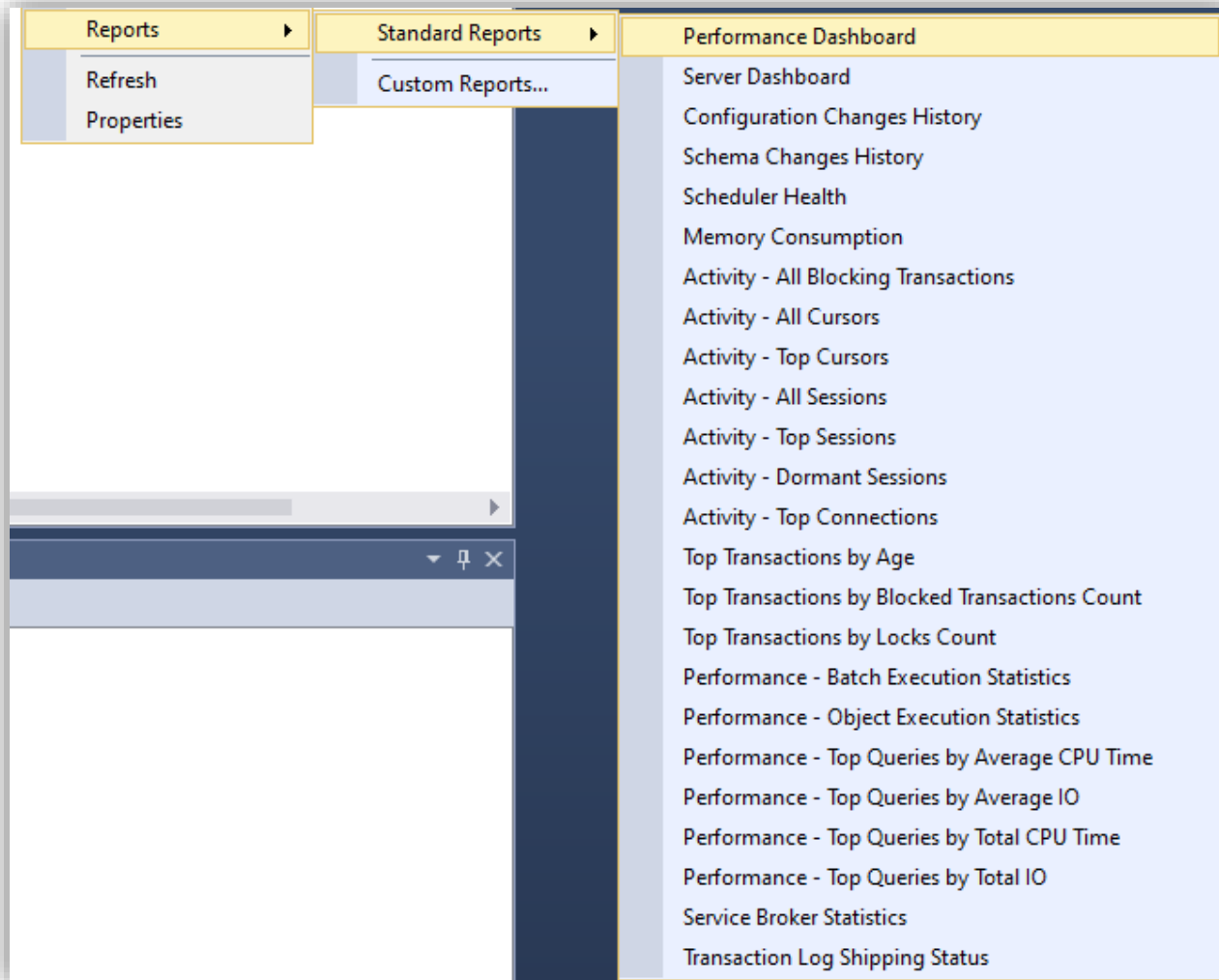
- Understand Built-In Instance Reports and Database Reports





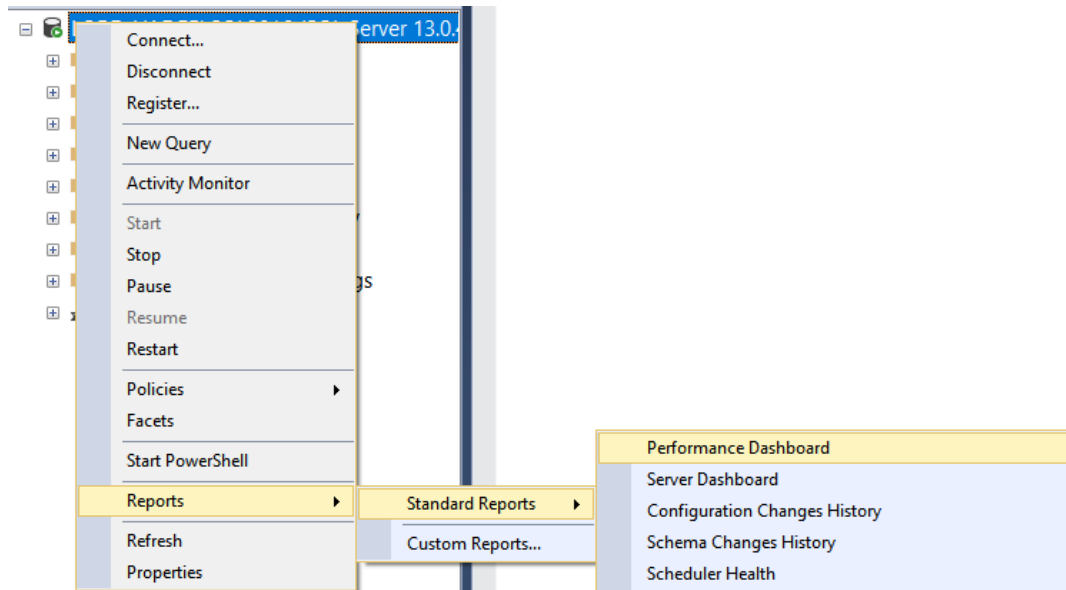
# Instance Reports

# Built-in Instance Reports



# Performance Dashboard

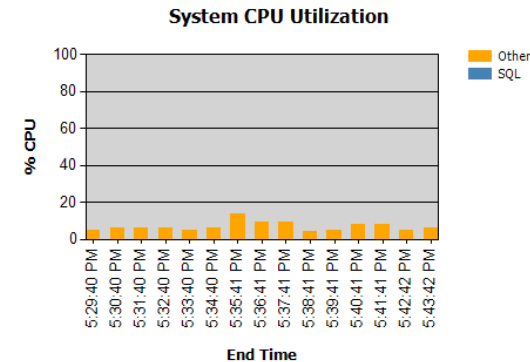
- To view the Performance Dashboard, right-click on the SQL Server instance name in Object Explorer, select **Reports, Standard Reports**, and click on **Performance Dashboard**.




## Microsoft SQL Server Performance Dashboard

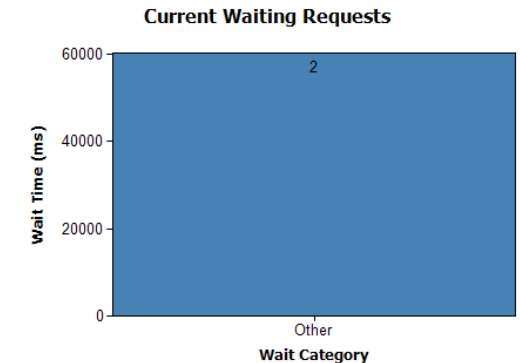
Report Local Time: 11/5/2020 5:44:05 PM

(12.0.2000.8 - SQL Azure)



Current Activity		
	User Requests	User Sessions
Count	3	13
Elapsed Time (ms)	1377628468	2736
CPU Time (ms)	886(0.00%)	188(6.87%)
Wait Time (ms)	1377627582(100.00%)	2548(93.13%)
Cache Hit Ratio	46.218%	88.058%

 System performance may be degraded because of excessive waits happening on the server. Click on a Wait Category data point in the chart below to investigate further.

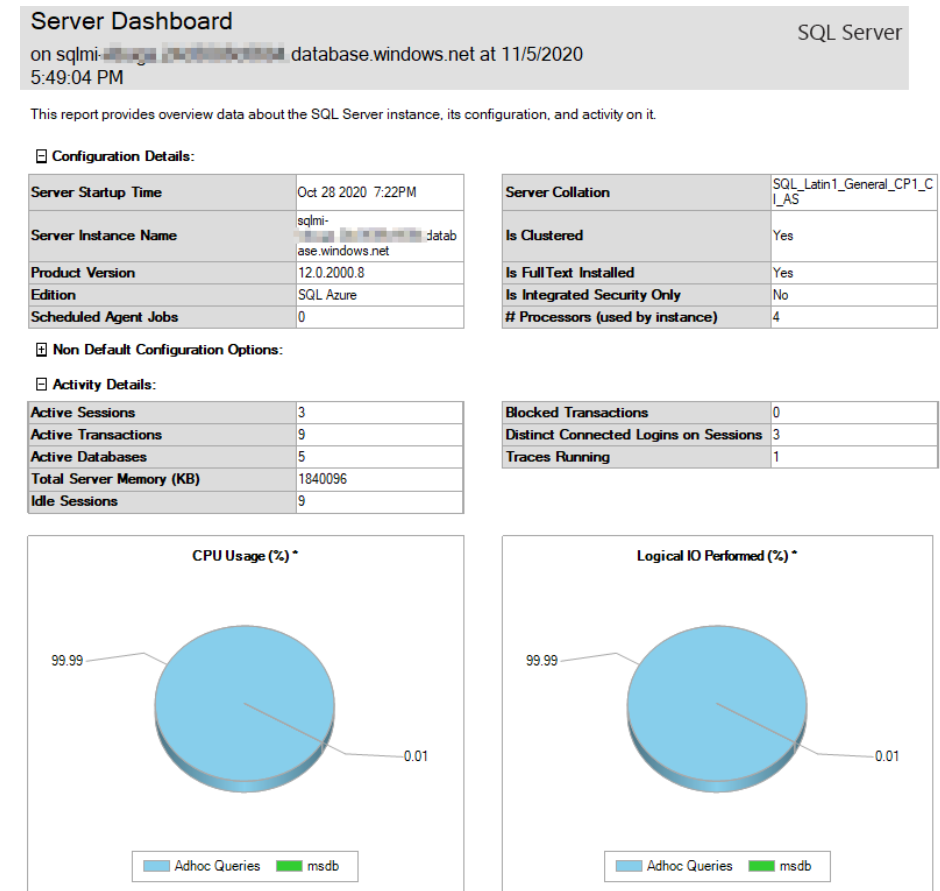


Historical Information	
<a href="#">Waits</a>	<a href="#">IO Statistics</a>
<a href="#">Latches</a>	
Expensive Queries	
<a href="#">By CPU</a>	<a href="#">By Duration</a>
<a href="#">By Logical Reads</a>	<a href="#">By Physical Reads</a>
<a href="#">By Logical Writes</a>	<a href="#">By CLR Time</a>

Miscellaneous Information	
<a href="#">Active Traces</a>	1
<a href="#">Active Xevent Sessions</a>	3
<a href="#">Databases</a>	5

# 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



\* : "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

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

SQL Server

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 X Schema Changes Hi...abase.windows.net

Schema Changes History

SQL Server

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

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

# Performance Reports

## Batch Execution Statistics

- Provides execution history data for all cached batch plans.

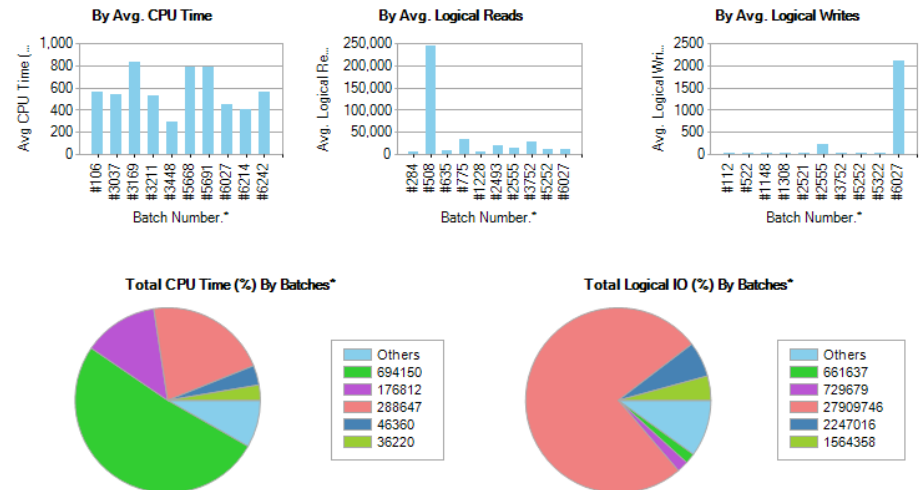
### Performance - Batch Execution Statistics

on [server-001.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.

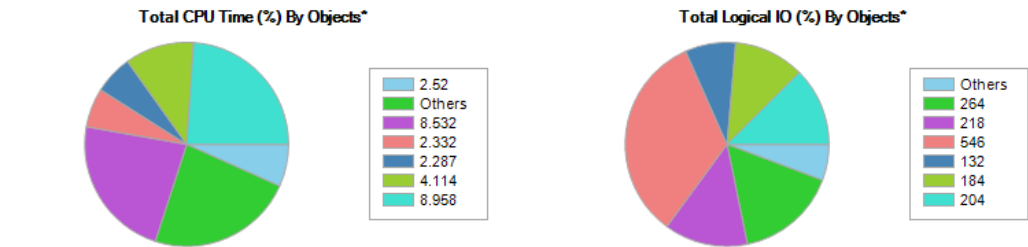
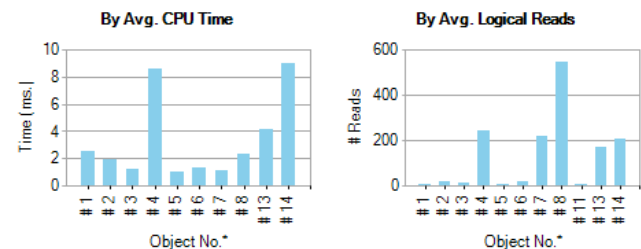
Batch Number	First SQL Statement of Batch	Avg. CPU Time (ms.)	# Avg. Logical Reads	# Avg. Logical Writes
1	SELECT *, 861 FROM [AdventureWorks].[Person].[Person] WHERE [BusinessEntityID] = @BusinessEntityI	0.19	3.00	0.00
2	SELECT *, 821 FROM [AdventureWorks].[Person].[Person] WHERE [BusinessEntityID]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

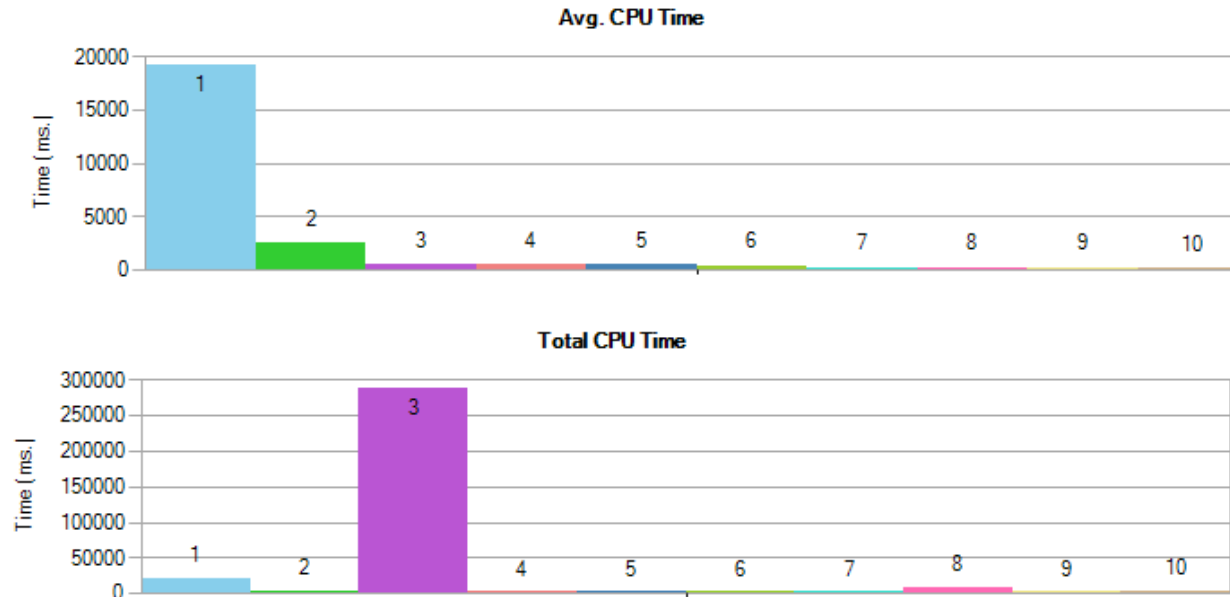
Object No.*	Database Name	Object Name	Object Type	Avg. CPU Time (ms.)	Total CPU Time (%)	# Avg. Logical Reads	# Avg. Logical Writes	# Avg. Logical IO	Total Logical IO (%)
1	msdb	managed_backup fn_backup_db_config	SQL Table-valued-Function	2.52	6.73	4.00	0.00	4.00	0.24
SQL Statement				# Executions (With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

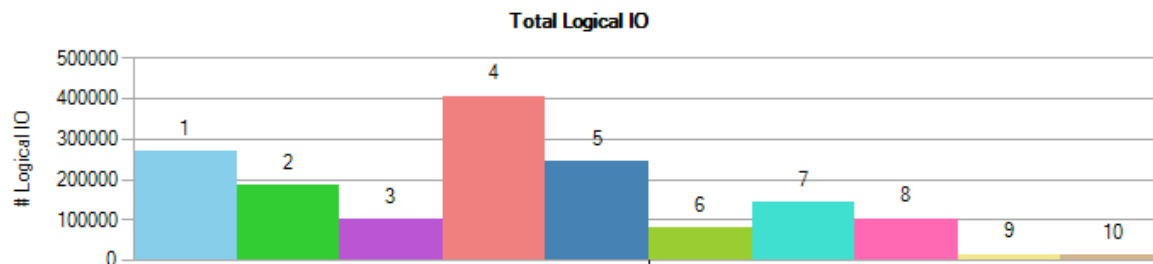
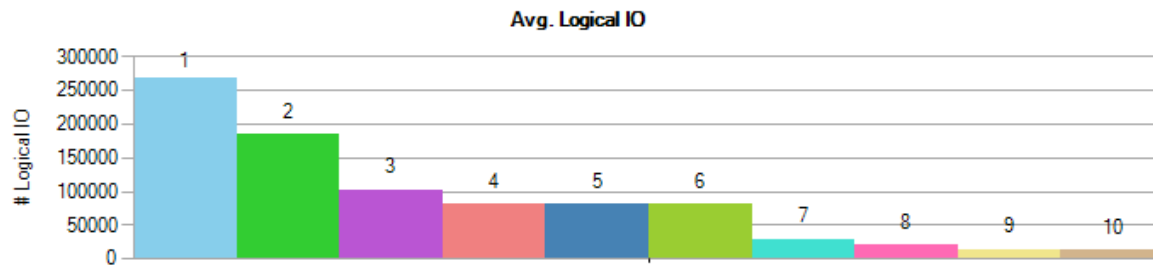


Query No.	Query Text	Database Name	Object ID	Avg. CPU Time (ms.)
1	<pre>select (s.t_SPRank)%2 as I1 , (dense_rank() over(order by s.t_SPRank,s.row_id))%2 as I2 , s.* , ob.cpu_rank as t_CPURank , ob.read_rank as t_ReadRank , ob.write_rank as t_WriteRank from @sql_handle_convert_table s join @objects ob on (s.t_SPRank = ob.obj_rank)</pre>			19,204.53
2	<pre>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</pre>			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	<pre>select (s.t_SPRank)%2 as I1 , (dense_rank() over(order by s.t_SPRank,s.row_id))%2 as I2 , 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</pre>			268,351.00
2	<pre>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 I1 , 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</pre>			184,550.00

# Database Reports

# Built-In Database Reports

Reports	Standard Reports	Disk Usage
Rename	Custom Reports...	Data Classification
Delete	All Transactions	Disk Usage by Top Tables
Refresh	Disk Usage	Disk Usage by Table
Properties	All Blocking Transactions	Disk Usage by Partition
		Backup and Restore Events
		All Transactions
		All Blocking Transactions
		Top Transactions by Age
		Top Transactions by Blocked Transactions Count
		Top Transactions by Locks Count
		Resource Locking Statistics by Objects
		Object Execution Statistics
		Database Consistency History
		Transaction Performance Analysis Overview
		Index Usage Statistics
		Index Physical Statistics
		Schema Changes History
		User Statistics

# Demonstration

## Server Reports

- SSMS Built-In Server Reports





Questions?



Dankie Faleminderit **Shukran** Chnorakaloutioun Hvala Blagodaria

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

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

**Thank you!**

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

谢谢 Баярлалаа **Dziękuję** Obrigado Mulțumesc **Спасибо** Ngiyabonga

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

