Microsoft

SQL Server 2016 CTP2 Query Store HOL

Contents

Overview and setup

Estimated time to complete lab is 15-25 minutes.

Overview

Query Store is a new SQL Server component that captures queries, query plans, runtime statistics, etc. in a persistent store inside the user database. You can think of it as a flight recorder, or black box, for your database. It can also enforce policies to direct the SQL Server query processor to compile some queries to be executed in a specific manner, such as forcing plans. It is a database-scoped persistent store of query workload history. Query Store is primarily targeting administrative scenarios for performance troubleshooting, identifying regressed workloads, or identifying issues with upgrades.

Query Store collects query texts and all relevant properties, as well as query plan choices and performance metrics. This collection process works across restarts or upgrades of the server and across recompilation of indexes. It provides many configurable options for customization. Query Store integrates with existing query execution statistics, plan forcing, and manageability tools.

This hands on lab will familiarize you with Query Store and show you how it can be used for troubleshooting and performance tuning. In particular, you will learn:

1. How to work with the Query Store user interface in SSMS;
2. How to identify queries based on several criteria, including those that generate more than one query plan;
3. How to force a query to use a specific plan;
4. How to modify Query Store settings on the database;
5. How to run custom queries based on the Query Store DMVs.

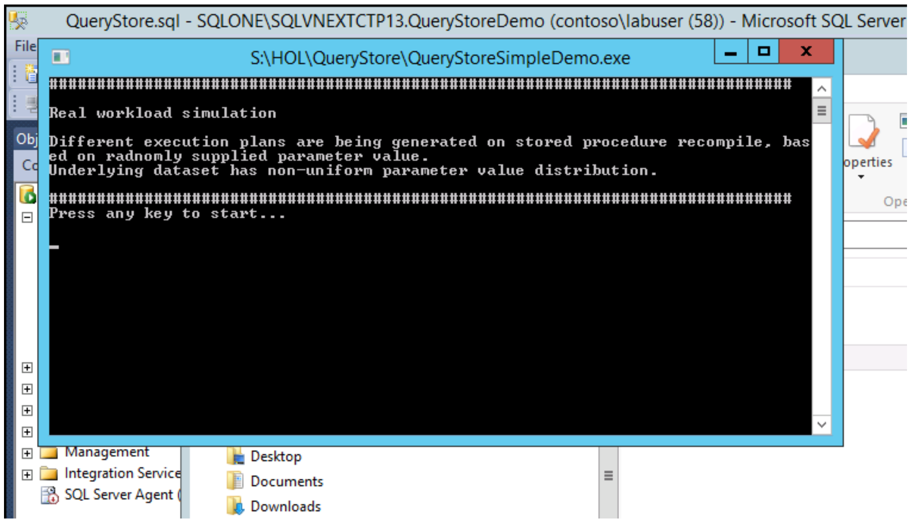
At the end of this lab, you will have worked through some of the most common scenarios involved with Query Store and how you can use this technology to run advanced troubleshooting and performance tuning in SQL Server 2016 CTP2.

Initial setup

1. During the setup you will need to record credentials and server locations. Open **notepad.exe** to keep track of information
2. Open SQL Server Management Studio on the lab machine.

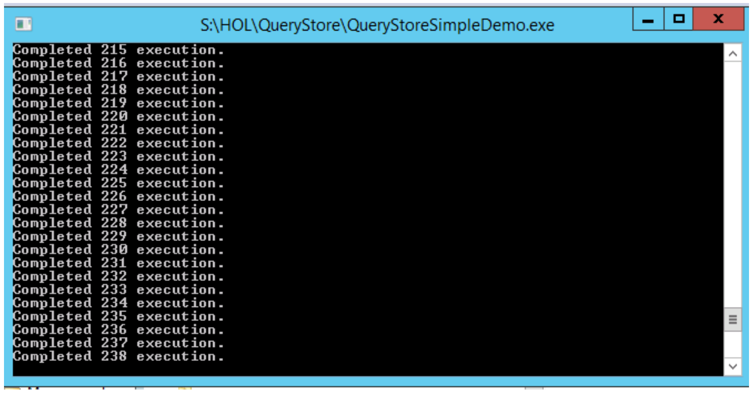
3. The script file that contains the scripts to be used in this lab are located at **C:\SQL Server 2016 CPT2 HOLs\Query Store\QueryStore.sql**. Open this script now in SSMS.

4. In the **C:\SQL Server 2016 CPT2 HOLs\Query Store** folder, double click on the file called **QueryStoreSimpleDemo.exe**. Once the executable program window opens, click the Enter key to start the workload. Let it run. It will remain running in the background for much of the rest of the lab.


QueryStore.sql - SQLONE\SQLVNEXTCTP13.QueryStoreDemo (contoso\labuser (58)) - Microsoft SQL Server
S:\HOL\QueryStore\QueryStoreSimpleDemo.exe
Management
Integration Service
SQL Server Agent
Desktop
Documents
Downloads

```
################################################################################
Real workload simulation

Different execution plans are being generated on stored procedure recompile, bas
ed on radnomly supplied parameter value.
Underlying dataset has non-uniform parameter value distribution.

################################################################################
Press any key to start...
```

5. Click **Enter** to run the load generator. The Visual Studio solution for this load generator is located in the **C:\SQL Server 2016 CPT2 HOLs\Query Store\QueryStoreSimpleDemo\QueryStoreSimpleDemo** folder.



```
Completed 215 execution.
Completed 216 execution.
Completed 217 execution.
Completed 218 execution.
Completed 219 execution.
Completed 220 execution.
Completed 221 execution.
Completed 222 execution.
Completed 223 execution.
Completed 224 execution.
Completed 225 execution.
Completed 226 execution.
Completed 227 execution.
Completed 228 execution.
Completed 229 execution.
Completed 230 execution.
Completed 231 execution.
Completed 232 execution.
Completed 233 execution.
Completed 234 execution.
Completed 235 execution.
Completed 236 execution.
Completed 237 execution.
Completed 238 execution.
```

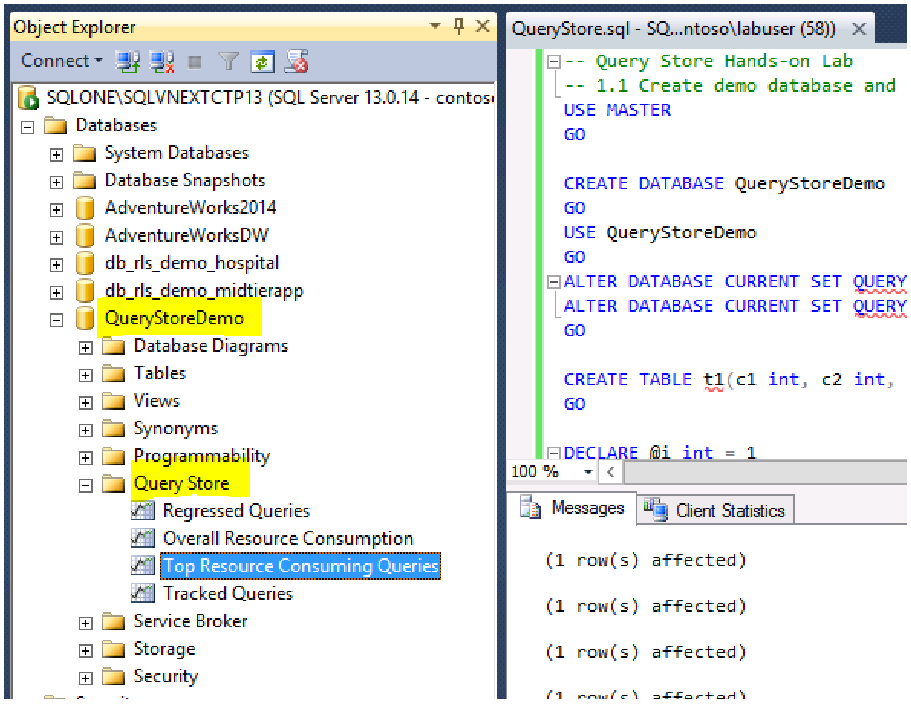
Explore the Query Store user interface in SSMS

In this lab, we will explore the Query Store tools in SSMS that help you conduct diagnostics and performance tuning. Let's get started!

Explore the Query Store UI in SQL Server Management Studio
Query Store data can be accessed through TSQL from the Query Store system views, but many important uses of the data can also be viewed through the rich toolset exposed in SSMS. We will explore the tools in SSMS first. At the end of this lab, we'll look at using TSQL and how you can leverage system views in querying this data.

1. Minimize the executing QueryStoreSimpleDemo program, but keep it running.

2. In the object explorer, refresh the database list and expand the new database we just created—QueryStoreDemo. Expand the Query Store node under that database.

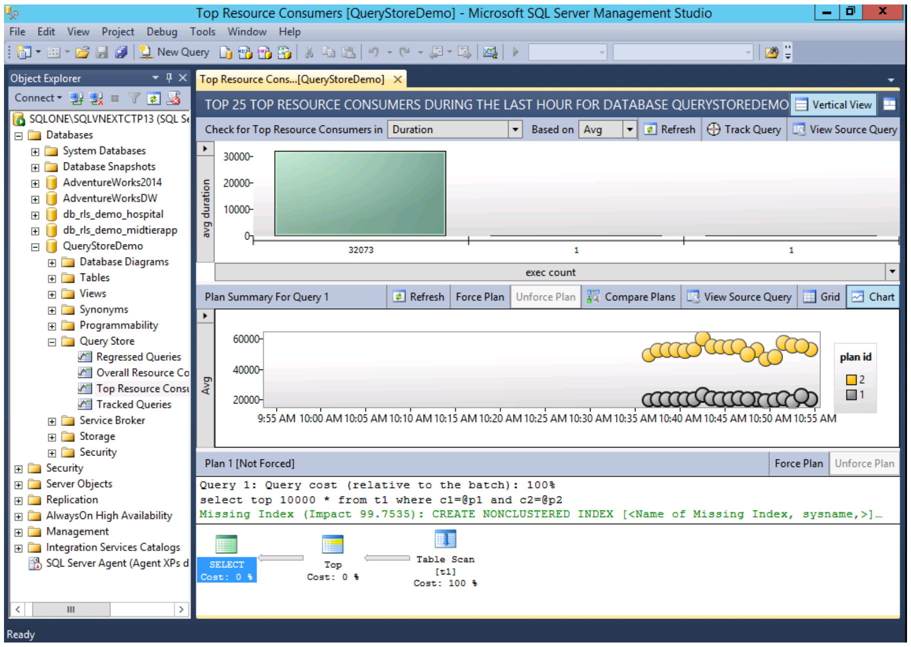


```
SQLONE\SQLVNEXTCTP13 (SQL Server 13.0.14 - contos
  Databases
    System Databases
    Database Snapshots
    AdventureWorks2014
    AdventureWorksDW
    db_rls_demo_hospital
    db_rls_demo_midtierapp
    QueryStoreDemo
      Database Diagrams
      Tables
      Views
      Synonyms
      Programmability
      Query Store
        Regressed Queries
        Overall Resource Consumption
        Top Resource Consuming Queries
        Tracked Queries
      Service Broker
      Storage
      Security
```

```
-- Query Store Hands-on Lab
-- 1.1 Create demo database and
USE MASTER
GO

CREATE DATABASE QueryStoreDemo
GO
USE QueryStoreDemo
GO
ALTER DATABASE CURRENT SET QUERY
ALTER DATABASE CURRENT SET QUERY
GO

CREATE TABLE t1(c1 int, c2 int,
GO

DECLARE @i int = 1
```

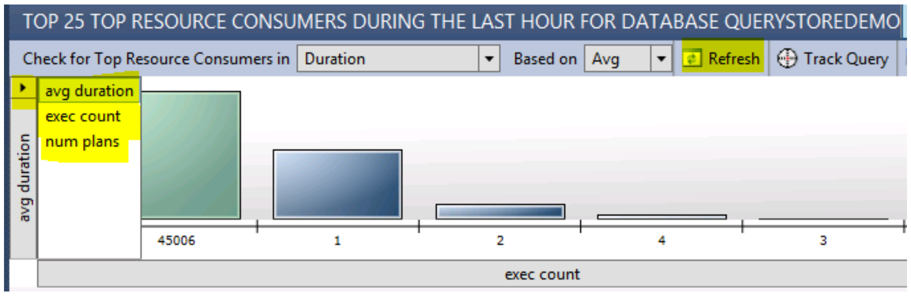
```
(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)

(1 row(s) affected)
```

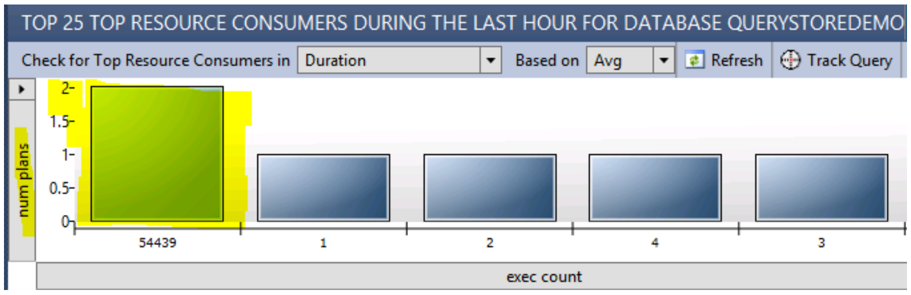
3. Double click on Top Resource Consuming Queries. There are three panes available. The top one allows you to view the top resource consumers based on various criteria. Click the refresh button in the top pane to refresh the view. Select the query that has the biggest impact on resources in the top pane.



```
Query 1: Query cost (relative to the batch): 100%
select top 10000 * from t1 where c1=@p1 and c2=@p2
Missing Index (Impact 99.7535): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]…
```

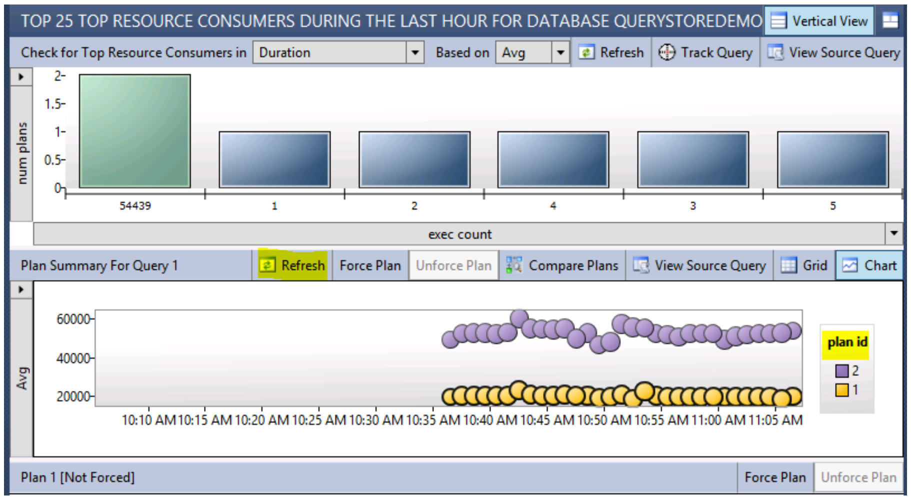
4. Notice that you can hover your mouse over a query in the top pane and see more details about the query ID, average duration, number of execution counts, and number of plans generated. Click refresh. Click on the left button labeled “avg duration” and change to “num plans.”


TOP 25 TOP RESOURCE CONSUMERS DURING THE LAST HOUR FOR DATABASE QUERYSTOREDEMO
Check for Top Resource Consumers in Duration
Based on Avg
Refresh
Track Query
avg duration
exec count
num plans
avg duration
45006
1
2
4
3
exec count


TOP 25 TOP RESOURCE CONSUMERS DURING THE LAST HOUR FOR DATABASE QUERYSTOREDEMO
Check for Top Resource Consumers in Duration
Based on Avg
Refresh
Track Query
num plans
2
1.5
1
0.5
0
54439
1
2
4
3
exec count

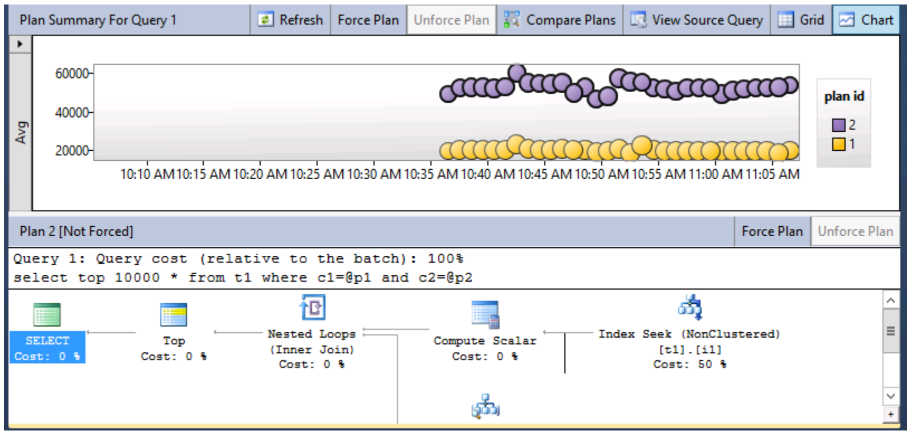
Note that this shows you that while several queries are running in the workload, one query has generated 2 plans. Click on that query that has 2 plans.

5. Look at the middle pane (Plan Summary for Query 1). You should see that two plans are running (with plan ids of 1 and 2) and that SQL Server is constantly alternating between them. Each dot represents an interval of aggregation, based on the interval length, which was set at 1 minute when we created the database. Click refresh in the middle pane to see how this changes over time.



The Query ID and Plan ID are the Query Store internal identifiers.

6. In the middle pane, click on the plan on top. It is generating longer average execution times (the values on the left axis represent time in milliseconds). When you click on either plan, you see the execution plan in the bottom pane.

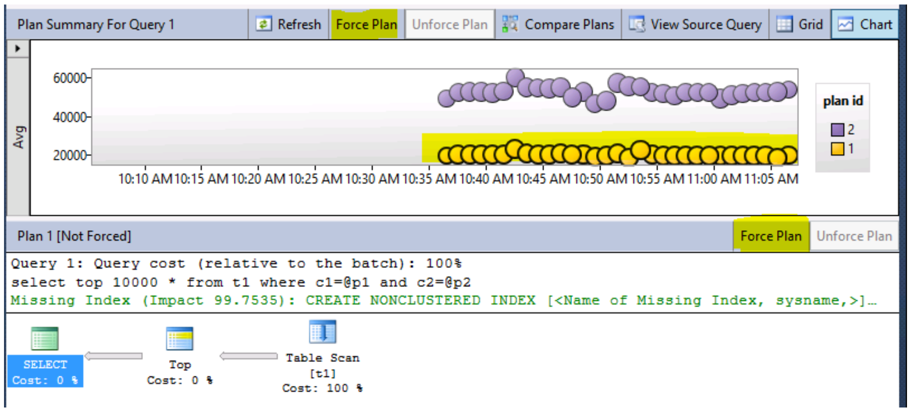


```
Query 1: Query cost (relative to the batch): 100%
select top 10000 * from t1 where c1=@p1 and c2=@p2
```

Force a query to use a certain execution plan

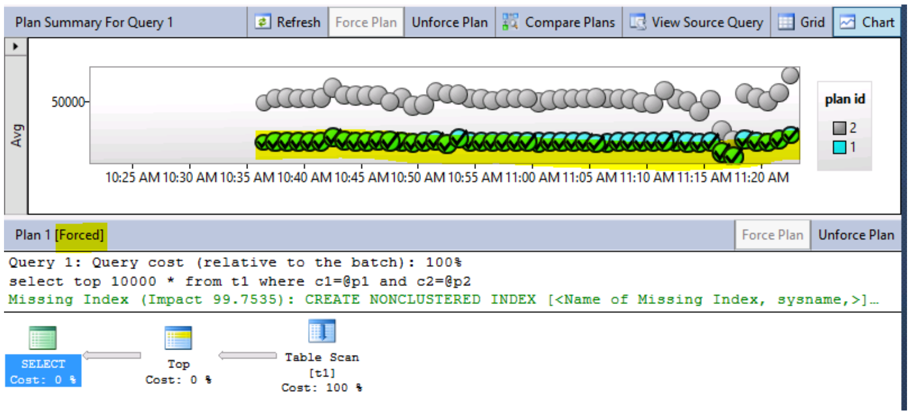
If you look at both the average execution times and the query execution plans for this query, you can see that one plan (the one on top) is taking much longer to run. We can use Query Store to force the query to always run with the better plan.

6. Select the bottom plan in the middle pane to select it. Notice also that hovering your mouse over the plan displays more information about it.



```
Query 1: Query cost (relative to the batch): 100%
select top 10000 * from t1 where c1=@p1 and c2=@p2
Missing Index (Impact 99.7535): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]…
```

7. Click on Force Plan (in either the middle or bottom pane). And click Yes on the confirmation.



```
Query 1: Query cost (relative to the batch): 100%
select top 10000 * from t1 where c1=@p1 and c2=@p2
Missing Index (Impact 99.7535): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>]…
```

8. The plan now contains check marks in each interval for the forced plan. Every time this query runs now, it will use the forced plan.

9. If you want to later allow SQL Server to choose between plans, you will need to click on the “Unforce Plan” button to the right of the disabled “Force Plan” button.

10. Refresh the pane again to verify.

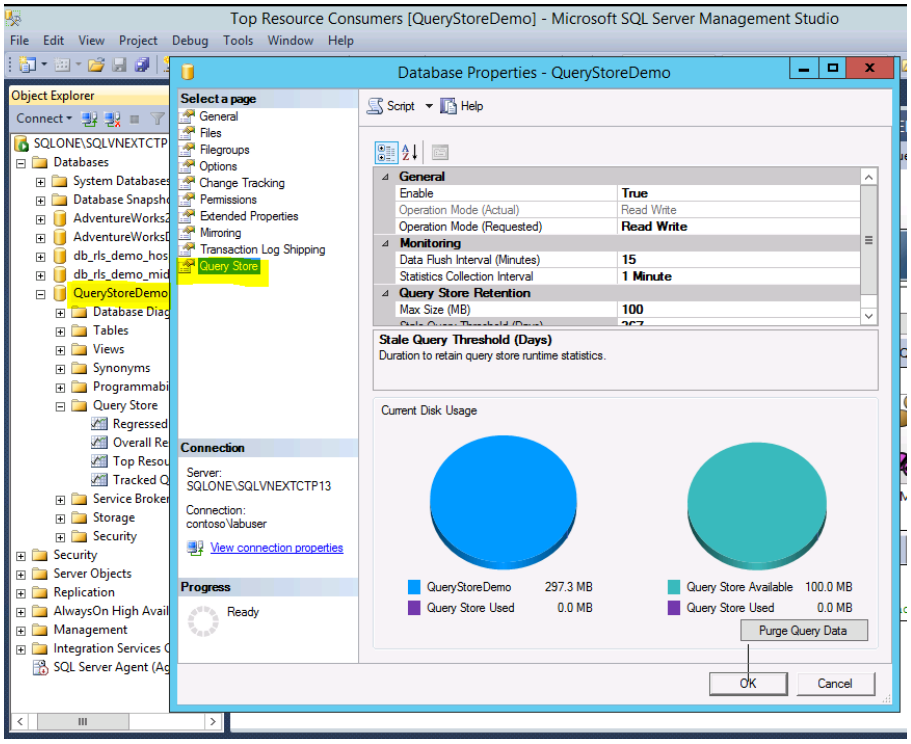
Explore Query Store database settings

In this lab we will look at how to adjust various settings at the database level for Query Store.

Explore database-level settings

Query Store has several settings you can adjust, depending on your needs.

1. In Object Explorer, right click on the database name (QueryStoreDemo) and select Properties. Notice that you can see a properties page for Query Store.



Top Resource Consumers [QueryStoreDemo] - Microsoft SQL Server Management Studio

Database Properties - QueryStoreDemo

General	
Enable	True
Operation Mode (Actual)	Read Write
Operation Mode (Requested)	Read Write
Monitoring	
Data Flush Interval (Minutes)	15
Statistics Collection Interval	1 Minute
Query Store Retention	
Max Size (MB)	100

Stale Query Threshold (Days)
Duration to retain query store runtime statistics.

Current Disk Usage

QueryStoreDemo	297.3 MB	Query Store Available	100.0 MB
Query Store Used	0.0 MB	Query Store Used	0.0 MB

Connection
Server:
SQLONE\SQLVNEXTCTP13
Connection:
contoso\labuser

2. Notice that you can adjust several settings here, including:
   - Enabling/Disabling Query Store
   - Operation Mode (Read Write v. Read only)
   - Data Flush Interval
   - Statistics Collection Interval
   - Max Size in megabytes for Query Store retention, and
   - Stale Query Threshold in days.

3. There are also graphics that show you how much the current disk usage is, as well as the ability to purge query data.

Explore Query Store data in TSQL

All data that Query Store stores is available through the following system views:

- sys.query_store_query_text
- sys.query_store_query
- sys.query_store_plan
- sys.query_context_settings
- sys.query_store_runtime_stats_interval
- sys.query_store_runtime_stats
- sys.database_query_store_options

You can use these views in your own queries to get insight about query execution and other matters for a period of time when Query Store was active. In this part of the lab, we're going to look at some common uses for these custom queries.

Execute each query. They are each in the SQL script file for this lab. Feel free to modify the TSQL to experiment with the results.

Find last *n* queries that were executed on the database

```
SELECT TOP 10 qt.query_sql_text, q.query_id,
qt.query_text_id, p.plan_id, rs.last_execution_time
FROM sys.query_store_query_text qt
JOIN sys.query_store_query q ON qt.query_text_id =
q.query_text_id
JOIN  sys.query_store_plan p ON q.query_id = p.query_id
JOIN  sys.query_store_runtime_stats rs ON p.plan_id =
rs.plan_id
ORDER BY rs.last_execution_time DESC
```

Find the count of executions for each query

```
SELECT q.query_id, qt.query_text_id, qt.query_sql_text,
SUM(rs.count_executions) AS total_execution_count
FROM  sys.query_store_query_text qt
JOIN  sys.query_store_query q ON qt.query_text_id =
q.query_text_id
JOIN  sys.query_store_plan p ON q.query_id = p.query_id
JOIN  sys.query_store_runtime_stats rs ON p.plan_id =
rs.plan_id
GROUP BY q.query_id, qt.query_text_id, qt.query_sql_text
ORDER BY total_execution_count DESC
```

Find *n* queries with the longest execution time in the last hour

```
SELECT TOP 10 qt.query_sql_text, q.query_id,
qt.query_text_id, p.plan_id,
getutcdate() as CurrentUTCTime, rs.last_execution_time,
rs.avg_duration
FROM  sys.query_store_query_text qt
JOIN  sys.query_store_query q ON qt.query_text_id =
q.query_text_id
JOIN  sys.query_store_plan p ON q.query_id = p.query_id
JOIN  sys.query_store_runtime_stats rs ON p.plan_id =
rs.plan_id
WHERE rs.last_execution_time > dateadd(hour, -1,
getutcdate())
ORDER BY rs.avg_duration DESC
```

Find *n* queries that had the biggest average physical IO reads in the last 24 hours, with row counts and execution counts

```
SELECT TOP 10 qt.query_sql_text, q.query_id,
qt.query_text_id, p.plan_id,
rs.runtime_stats_id, rsi.start_time, rsi.end_time,
rs.avg_physical_io_reads,
rs.avg_rowcount, rs.count_executions
FROM  sys.query_store_query_text qt
JOIN  sys.query_store_query q ON qt.query_text_id =
q.query_text_id
JOIN  sys.query_store_plan p ON q.query_id = p.query_id
JOIN  sys.query_store_runtime_stats rs ON p.plan_id =
rs.plan_id
JOIN  sys.query_store_runtime_stats_interval rsi ON
rsi.runtime_stats_interval_id =
rs.runtime_stats_interval_id
WHERE rsi.start_time >= dateadd(hour, -24, getutcdate())
ORDER BY rs.avg_physical_io_reads DESC
```

Find queries with multiple plans

```
WITH Query_MultPlans
AS
(
      SELECT COUNT(*) AS cnt, q.query_id
      FROM sys.query_store_query_text qt
      JOIN sys.query_store_query q
      ON qt.query_text_id = q.query_text_id
      JOIN sys.query_store_plan p
      ON p.query_id = q.query_id
      GROUP BY q.query_id
      HAVING COUNT(DISTINCT plan_id) > 1
)
SELECT q.query_id, OBJECT_NAME(object_id) AS
ContainingObject,
query_sql_text, plan_id, p.query_plan AS plan_xml,
p.last_compile_start_time, p.last_execution_time
FROM Query_MultPlans qm
JOIN sys.query_store_query q
ON qm.query_id = q.query_id
JOIN sys.query_store_plan p
ON q.query_id = p.query_id
JOIN sys.query_store_query_text qt
ON qt.query_text_id = q.query_text_id
ORDER BY query_id, plan_id
```

Find all queries that regressed in performance, where execution time was doubled within the last 48 hours

```
SELECT qt.query_sql_text, q.query_id, qt.query_text_id,
p1.plan_id AS plan1, rsi1.start_time AS
runtime_stats_interval_1, rs1.runtime_stats_id AS
runtime_stats_id_1, rs1.avg_duration AS avg_duration_1,
p2.plan_id AS plan2, rsi2.start_time AS
runtime_stats_interval_2, rs2.runtime_stats_id AS
runtime_stats_id_2, rs2.avg_duration AS plan2
FROM  sys.query_store_query_text qt
JOIN  sys.query_store_query q ON qt.query_text_id =
q.query_text_id
JOIN  sys.query_store_plan p1 ON q.query_id =
p1.query_id
JOIN  sys.query_store_runtime_stats rs1 ON p1.plan_id =
rs1.plan_id
JOIN  sys.query_store_runtime_stats_interval rsi1 ON
rsi1.runtime_stats_interval_id =
rs1.runtime_stats_interval_id
JOIN  sys.query_store_plan p2 ON q.query_id =
p2.query_id
JOIN  sys.query_store_runtime_stats rs2 ON p2.plan_id =
rs2.plan_id
JOIN  sys.query_store_runtime_stats_interval rsi2 ON
rsi2.runtime_stats_interval_id =
rs2.runtime_stats_interval_id
WHERE rsi1.start_time > dateadd(hour, -48, getutcdate())
AND   rsi2.start_time > rsi1.start_time AND
      rs2.avg_duration > 2*rs1.avg_duration
```

You can now close the lab environment.

Terms of use

© 2015 Microsoft Corporation. All rights reserved.
By using this Hands-on Lab, you agree to the following terms:

The technology/functionality described in this Hands-on Lab is provided by Microsoft Corporation in a “sandbox” testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the Hands-on Lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this Hands-on Lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS HANDS-ONLAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCITONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK. If you give feedback about the technology features, functionality and/or concepts described in this Hands-on Lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB , INCLUDING ALL WARRANTIES AND

CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

DISCLAIMER

This lab contains only a portion of new features and enhancements in Microsoft SQL Server 2016 CTP2. Some of the features might change in future releases of the product. In this lab, you will learn about some, but not all, new features.