About Me

sleek technologies

TALLAN

Microsoft Most Valuable Professional

pluralsight

Microsoft CERTIFIED AZURE COSMOS DB DEVELOPER SPECIALTY

Leonard Lobel

- CTO & Co-Founder
  - Sleek Technologies, Inc.
- Principal Consultant
  - Tallan, Inc.
- Microsoft MVP
  - Data Platform
- Trainer/Speaker/Author
- Programming since 1979

Contact

- Email: lenni.lobel@sleektech.com
- Blog: lennilobel.wordpress.com
- Twitter: @lennilobel

Download Slides and Code

http://bit.ly/vslaustin2022_sqlworkshop

(all lower case!)

Schedule

- Workshop Begins
  - 8:00 AM
- Morning Break (15 minutes)
  - 10:00 AM
- Lunch (1 hour)
  - 12:00 PM
- Afternoon Break (15 minutes)
  - 3:00 PM
- Workshop Ends
  - 5:00 PM

Overview

Overview

- Part 1
  - Linux, Docker, and Containers
- Part 2
  - Intelligent Query Processing
- Part 3
  - Modern Development Platform
- Part 4
  - Security Features
- Part 5
  - Data Virtualization

Part 1

Linux, Docker, and Containers

SQL Server without Windows

- It's not just for Windows anymore
- SQL Server 2017/2019 also runs on:
  - Linux
  - Docker Engine
    - Run inside Windows containers on a PC
    - Run inside Linux containers on a PC or Mac

What's Not Supported on Linux (2017)?

- Database Engine
  - Transactional Replication
  - Merge Replication
  - Change Data Capture (CDC)
  - Stretch DB
  - PolyBase
  - Distributed query w/3rd party connections
  - Linked servers to non-SQL Server sources
  - System extended stored procedures (XPs)
  - FILESTREAM / FileTable
  - Unsafe CLR assemblies
  - Buffer Pool Extension
- SQL Server Agent
  - Alerts
  - Log Reader Agent
  - Managed Backup
- High Availability
  - Database mirroring
- Security
  - Extensible Key Management
  - AD authentication for Linked Servers
  - AD authentication for Availability Groups
- Services
  - SQL Server Browser
  - R (machine learning) Services
  - StreamInsight
  - Analysis Services (SSAS)
  - Reporting Services (SSRS)
  - Data Quality Services (DQS)
  - Master Data Services (MDS)

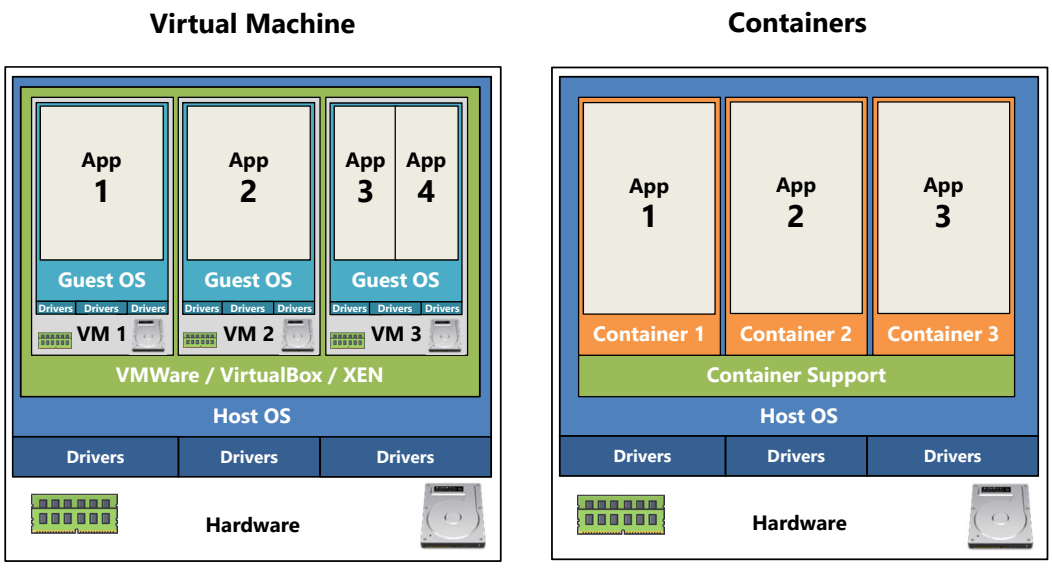
What's Not Supported on Linux (2019)?

- Database Engine
  - Transactional Replication
  - Merge Replication
  - Change Data Capture (CDC)
  - Stretch DB
  - PolyBase
  - Distributed query w/3rd party connections
  - Linked servers to non-SQL Server sources
  - System extended stored procedures (XPs)
  - FILESTREAM / FileTable
  - Unsafe CLR assemblies
  - Buffer Pool Extension
- SQL Server Agent
  - Alerts
  - Log Reader Agent
  - Managed Backup
- High Availability
  - Database mirroring
- Security
  - Extensible Key Management
  - AD authentication for Linked Servers
  - AD authentication for Availability Groups
- Services
  - SQL Server Browser
  - R (machine learning) Services
  - StreamInsight
  - Analysis Services (SSAS)
  - Reporting Services (SSRS)
  - Data Quality Services (DQS)
  - Master Data Services (MDS)

Docker vs. VM

- Docker is not a Virtual Machine
  - VMs import a guest OS on top of a host OS
    - Run multiple operating systems on the same machine
  - Fully allocated hardware resources
    - All of a VMs memory is carved out of the host machine
- Containers
  - OS virtualization with process isolation
    - Guest OS utilizes the host OS (and drivers)
  - Shared hardware resources and OS libraries
    - Each container competes for the memory that it needs
  - Isolated container storage
    - Host OS provides isolated storage for each container's file system
  - No Windows GUI / No RDP
    - CLI only (command line, PowerShell)

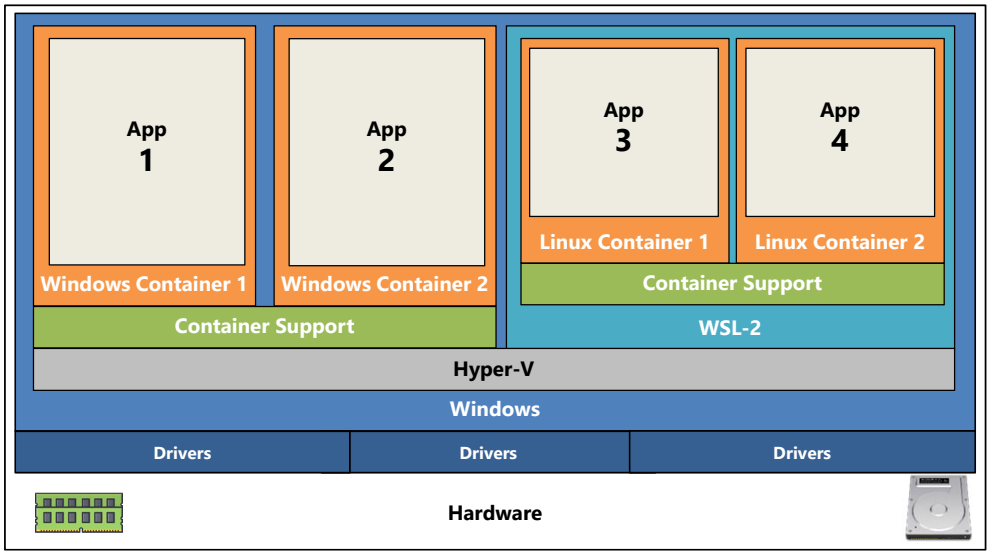
Docker Containers vs. Virtual Machines



Linux on Windows?

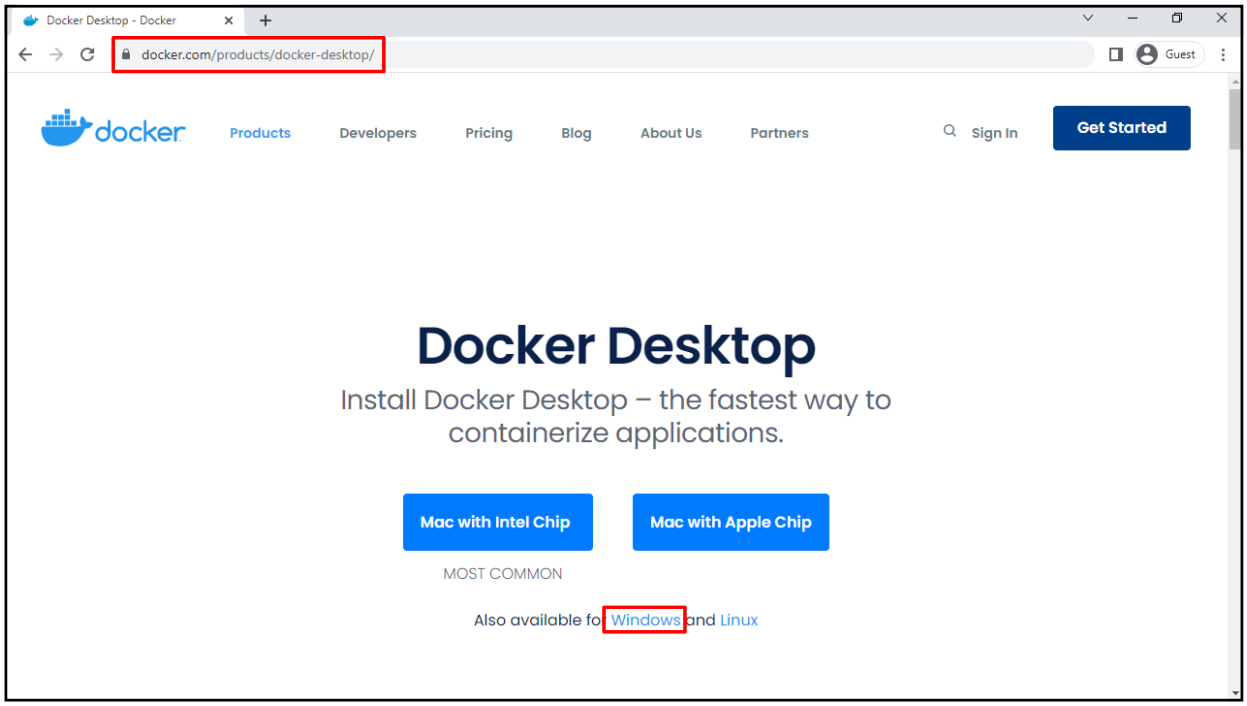
- Hyper-V and WSL 2 (Windows Subsystem for Linux)
  - Hypervisor in Windows 10 and Windows Server 2019
  - WSL 2 provides Linux kernel to support Linux containers
  - Run multiple Linux containers on Windows
- Windows and Linux containers can run concurrently
  - But they must be managed separately

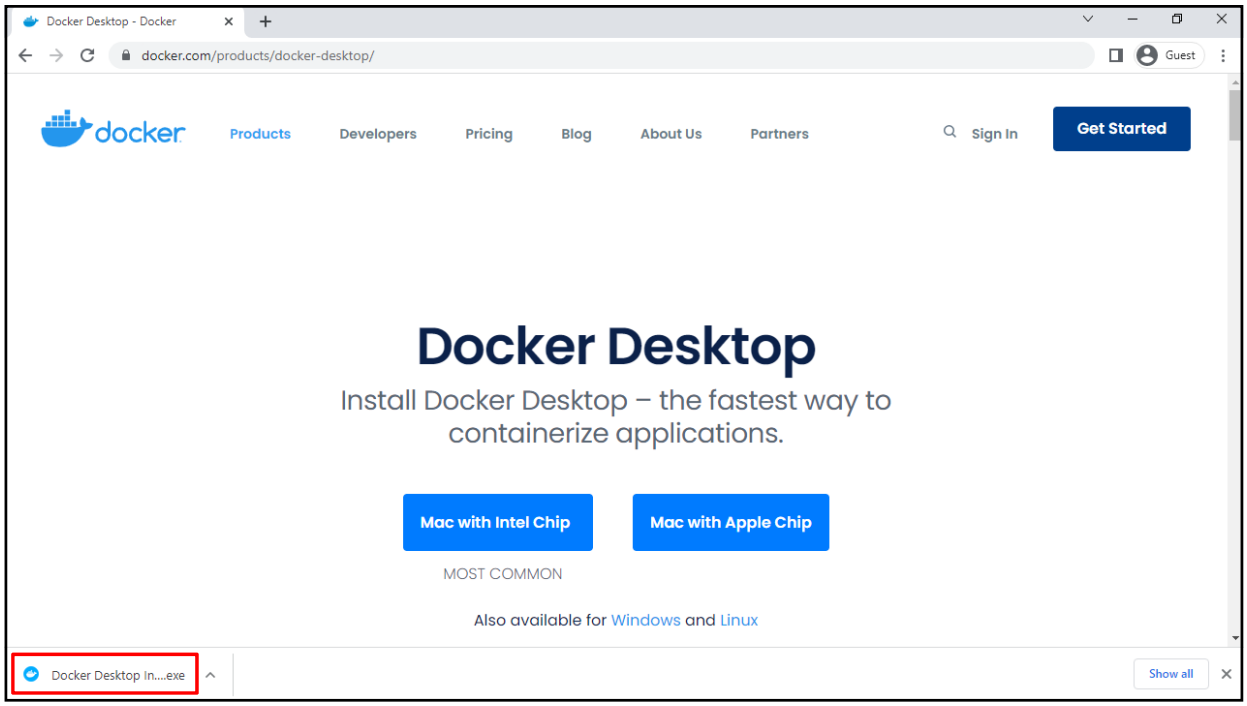
Linux on Windows (WSL-2)

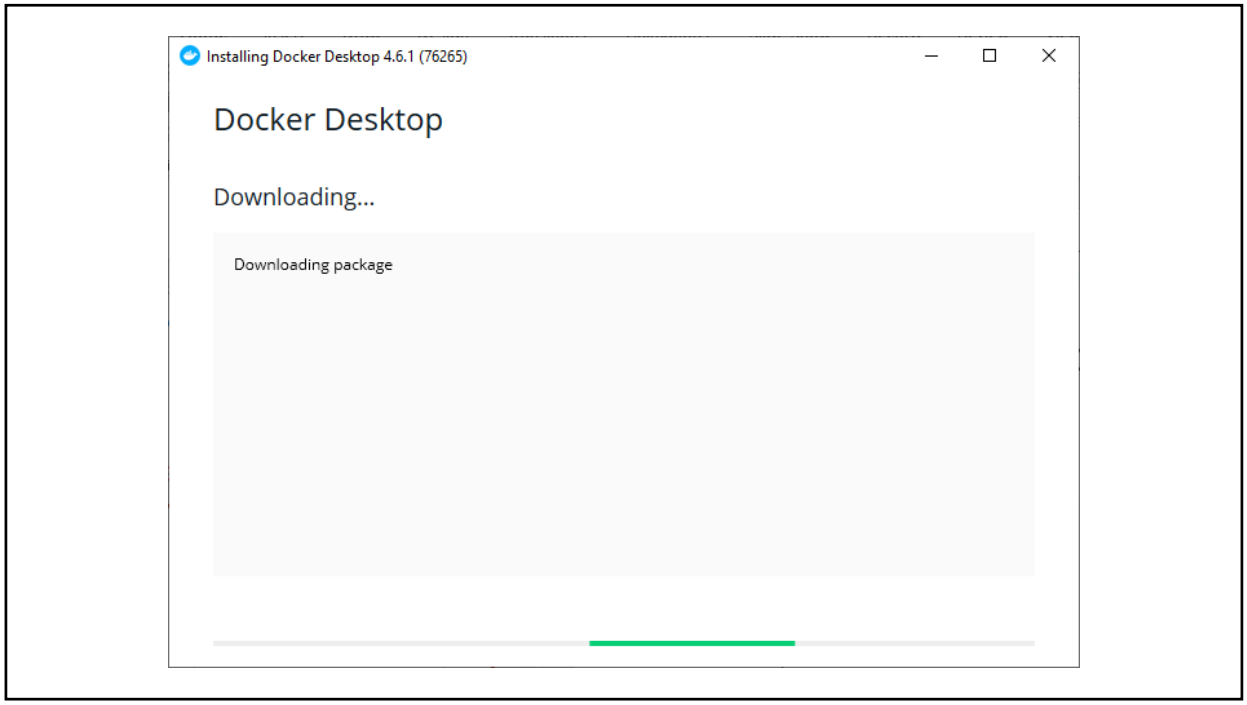


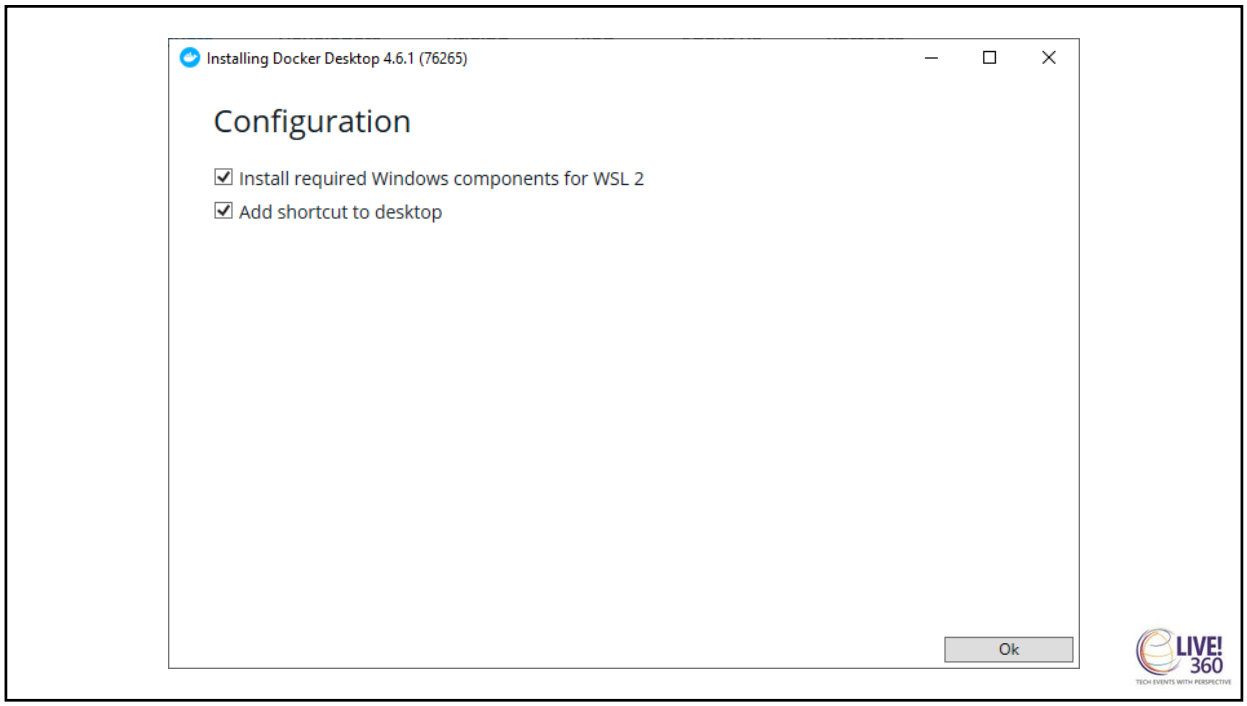
Docker on Windows Checklist

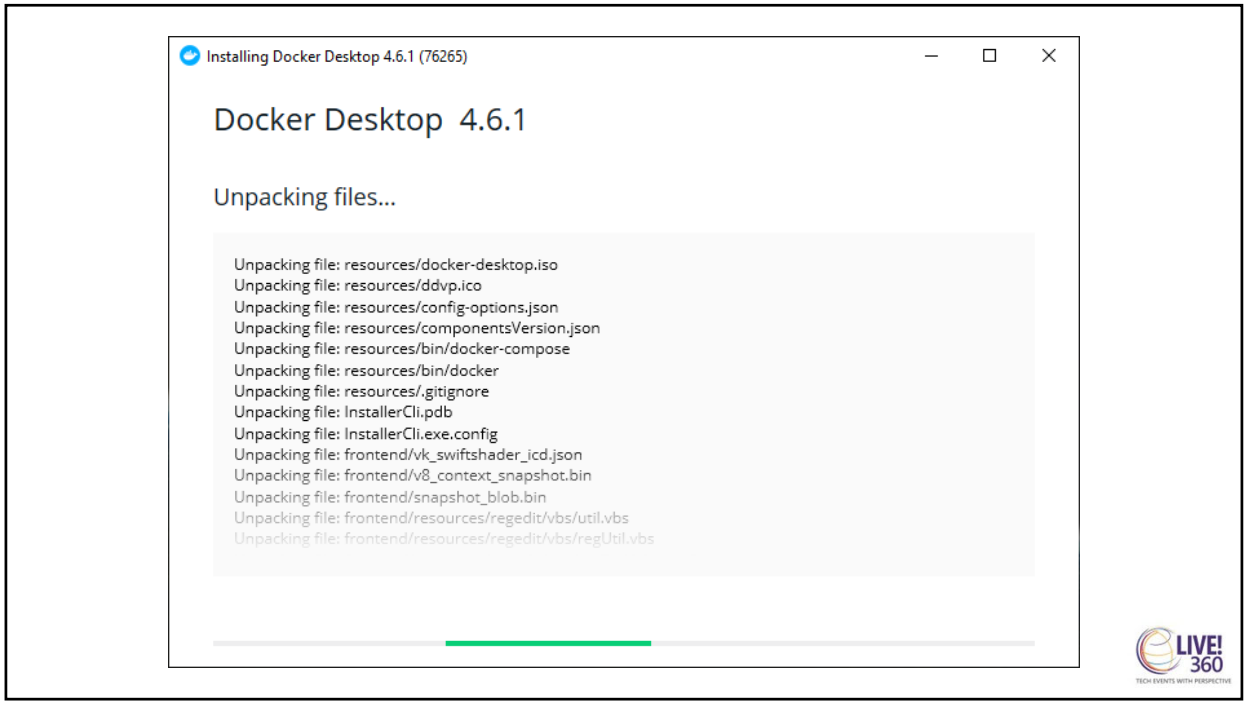
- Download Docker Desktop for Windows
  - Installs the tooling to work with containers
- Download desired Docker images
  - Pull base layer images (for either Windows or Linux)
- Run your Windows and/or Linux containers
  - Fully isolated GUI-less environment (CLI only, no RDP)

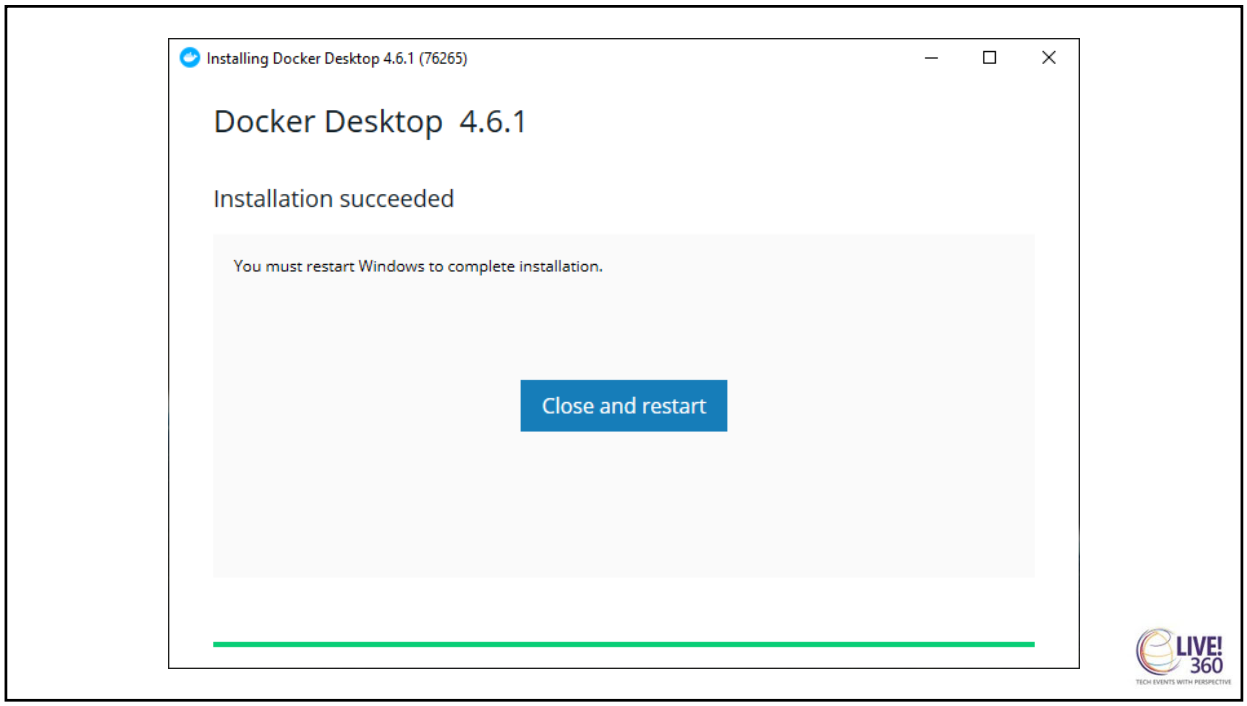


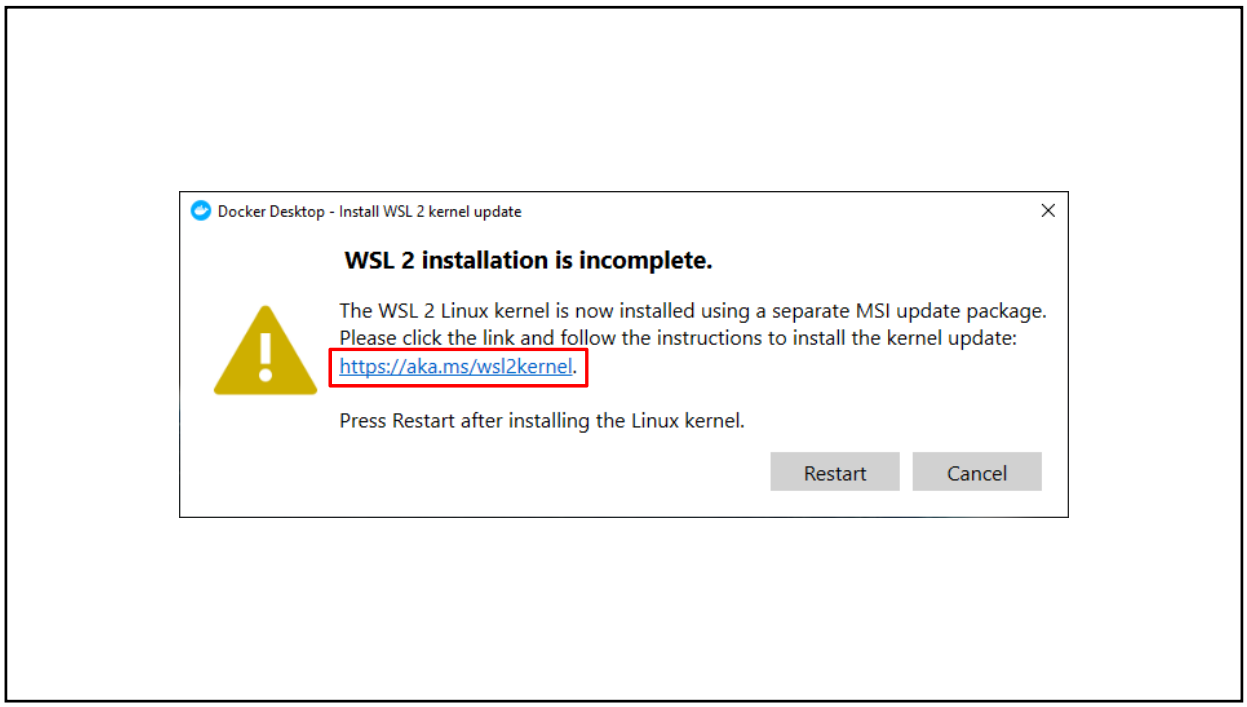

Docker Desktop - Docker
docker.com/products/docker-desktop/
Guest
docker
Products
Developers
Pricing
Blog
About Us
Partners
Sign In
Get Started
Docker Desktop
Install Docker Desktop – the fastest way to containerize applications.
Mac with Intel Chip
Mac with Apple Chip
MOST COMMON
Also available for Windows and Linux
Docker Desktop In....exe
Show all

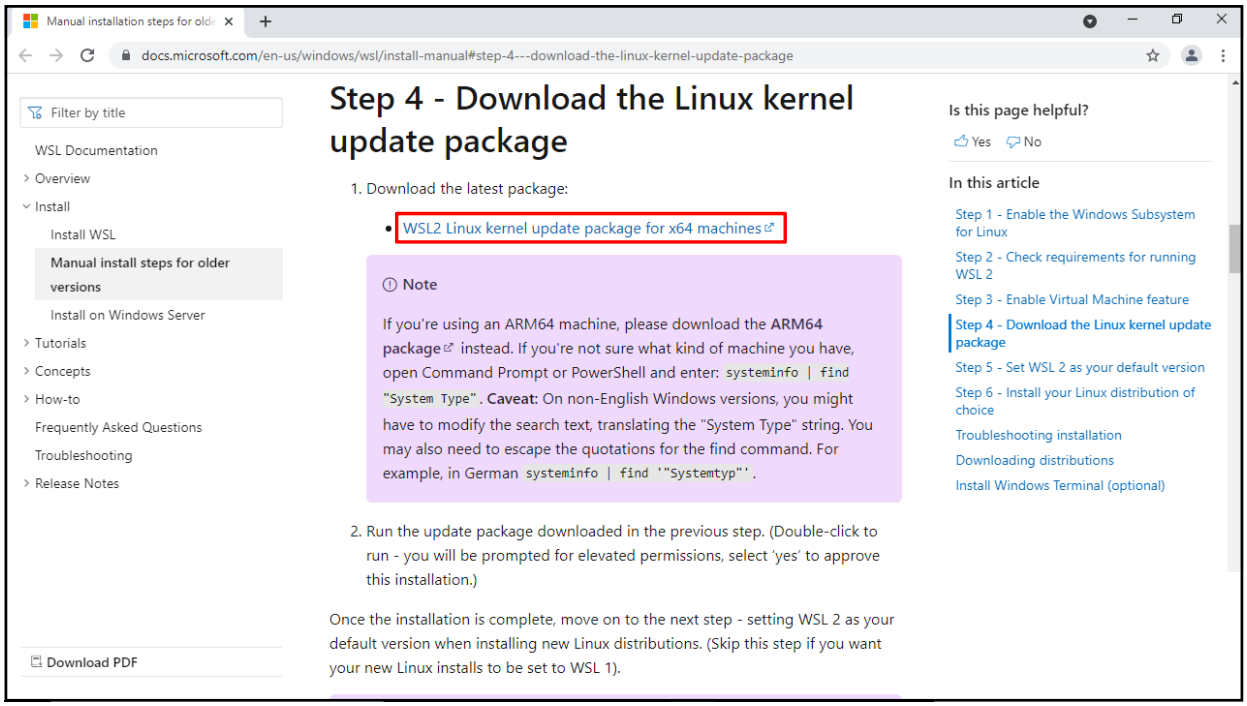

Installing Docker Desktop 4.6.1 (76265)
Docker Desktop
Downloading...
Downloading package

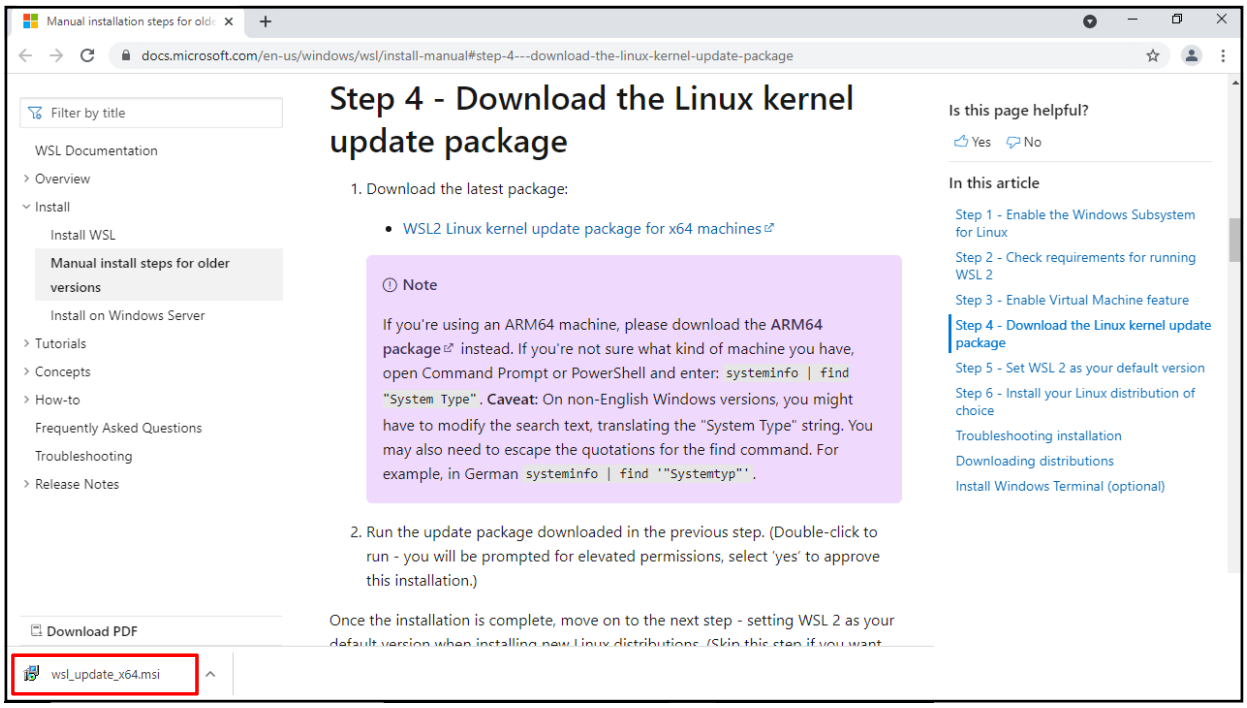

Installing Docker Desktop 4.6.1 (76265)
Configuration
Install required Windows components for WSL 2
Add shortcut to desktop
Ok
LIVE! 360

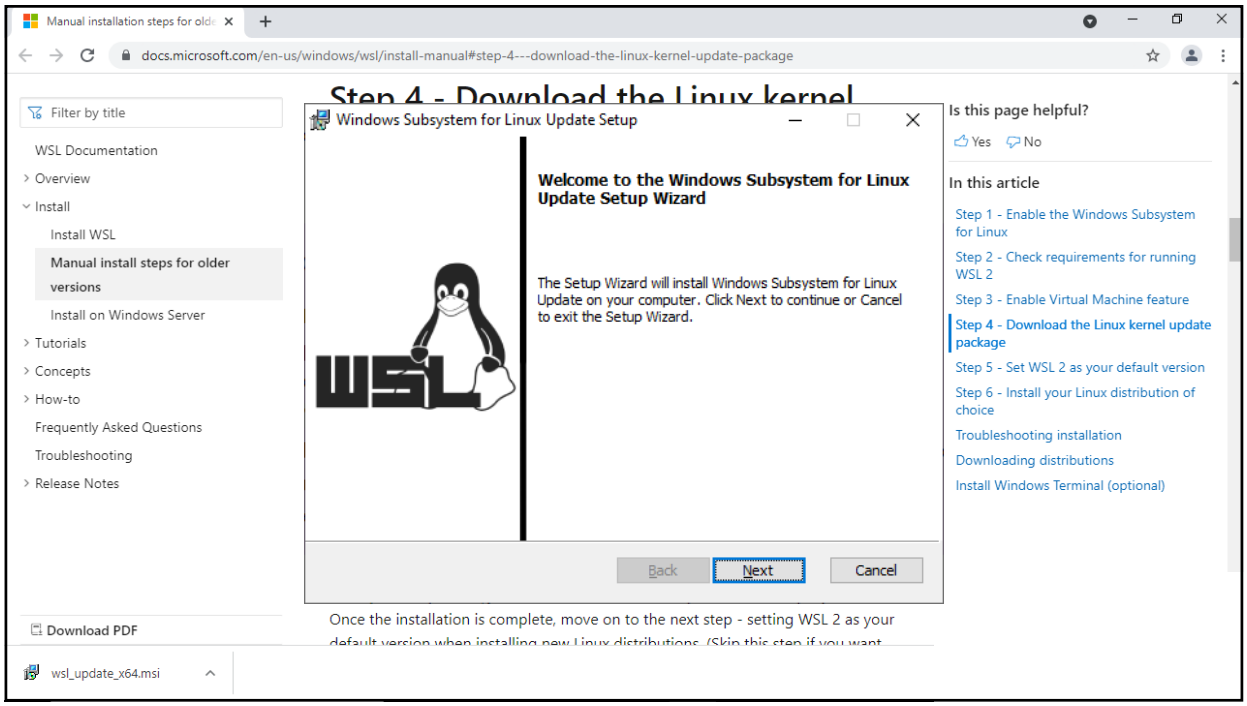

Installing Docker Desktop 4.6.1 (76265)
Docker Desktop 4.6.1
Unpacking files...
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin
Unpacking file: frontend/resources/regedit/vbs/util.vbs
Unpacking file: frontend/resources/regedit/vbs/regUtil.vbs
LIVE! 360

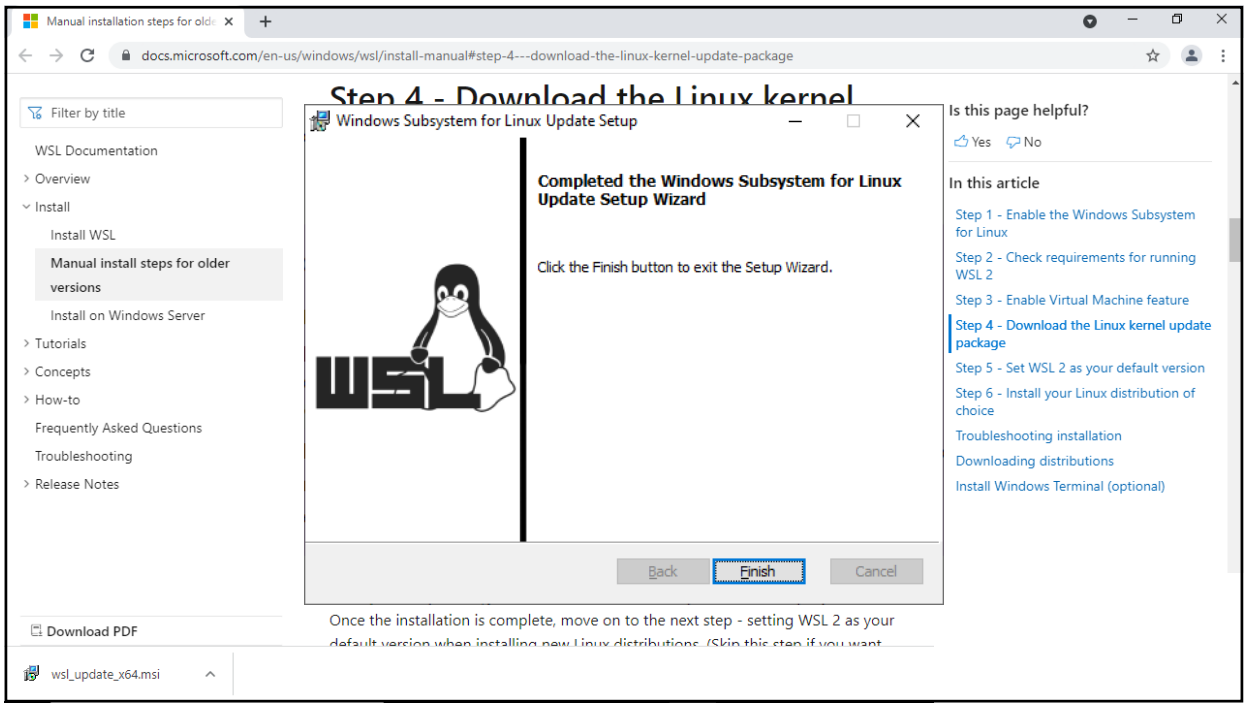

Installing Docker Desktop 4.6.1 (76265)
Docker Desktop 4.6.1
Installation succeeded
You must restart Windows to complete installation.
Close and restart
LIVE! 360


Docker Desktop - Install WSL 2 kernel update
WSL 2 installation is incomplete.
The WSL 2 Linux kernel is now installed using a separate MSI update package.
Please click the link and follow the instructions to install the kernel update:
https://aka.ms/wsl2kernel.
Press Restart after installing the Linux kernel.
Restart
Cancel

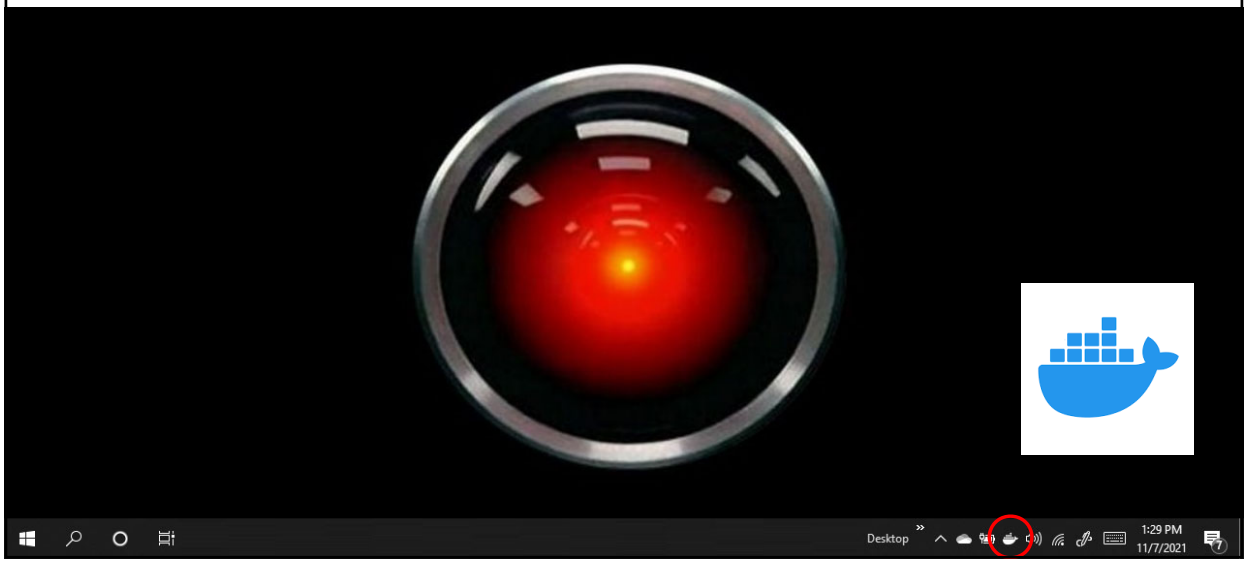

Manual installation steps for olde
docs.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package
Filter by title
WSL Documentation
Overview
Install
Install WSL
Manual install steps for older versions
Install on Windows Server
Tutorials
Concepts
How-to
Frequently Asked Questions
Troubleshooting
Release Notes
Download PDF
Step 4 - Download the Linux kernel update package
1. Download the latest package:
WSL2 Linux kernel update package for x64 machines
Note
If you're using an ARM64 machine, please download the ARM64 package instead. If you're not sure what kind of machine you have, open Command Prompt or PowerShell and enter: systeminfo | find "System Type". Caveat: On non-English Windows versions, you might have to modify the search text, translating the "System Type" string. You may also need to escape the quotations for the find command. For example, in German systeminfo | find '"Systemtyp"'.
2. Run the update package downloaded in the previous step. (Double-click to run - you will be prompted for elevated permissions, select 'yes' to approve this installation.)
Once the installation is complete, move on to the next step - setting WSL 2 as your default version when installing new Linux distributions. (Skip this step if you want your new Linux installs to be set to WSL 1).
Is this page helpful?
Yes
No
In this article
Step 1 - Enable the Windows Subsystem for Linux
Step 2 - Check requirements for running WSL 2
Step 3 - Enable Virtual Machine feature
Step 4 - Download the Linux kernel update package
Step 5 - Set WSL 2 as your default version
Step 6 - Install your Linux distribution of choice
Troubleshooting installation
Downloading distributions
Install Windows Terminal (optional)


Manual installation steps for olde
docs.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package
Filter by title
WSL Documentation
Overview
Install
Install WSL
Manual install steps for older versions
Install on Windows Server
Tutorials
Concepts
How-to
Frequently Asked Questions
Troubleshooting
Release Notes
Download PDF
wsl_update_x64.msi
Step 4 - Download the Linux kernel update package
1. Download the latest package:
WSL2 Linux kernel update package for x64 machines
Note
If you're using an ARM64 machine, please download the ARM64 package instead. If you're not sure what kind of machine you have, open Command Prompt or PowerShell and enter: systeminfo | find "System Type". Caveat: On non-English Windows versions, you might have to modify the search text, translating the "System Type" string. You may also need to escape the quotations for the find command. For example, in German systeminfo | find '"Systemtyp"'.
2. Run the update package downloaded in the previous step. (Double-click to run - you will be prompted for elevated permissions, select 'yes' to approve this installation.)
Once the installation is complete, move on to the next step - setting WSL 2 as your
Is this page helpful?
Yes
No
In this article
Step 1 - Enable the Windows Subsystem for Linux
Step 2 - Check requirements for running WSL 2
Step 3 - Enable Virtual Machine feature
Step 4 - Download the Linux kernel update package
Step 5 - Set WSL 2 as your default version
Step 6 - Install your Linux distribution of choice
Troubleshooting installation
Downloading distributions
Install Windows Terminal (optional)

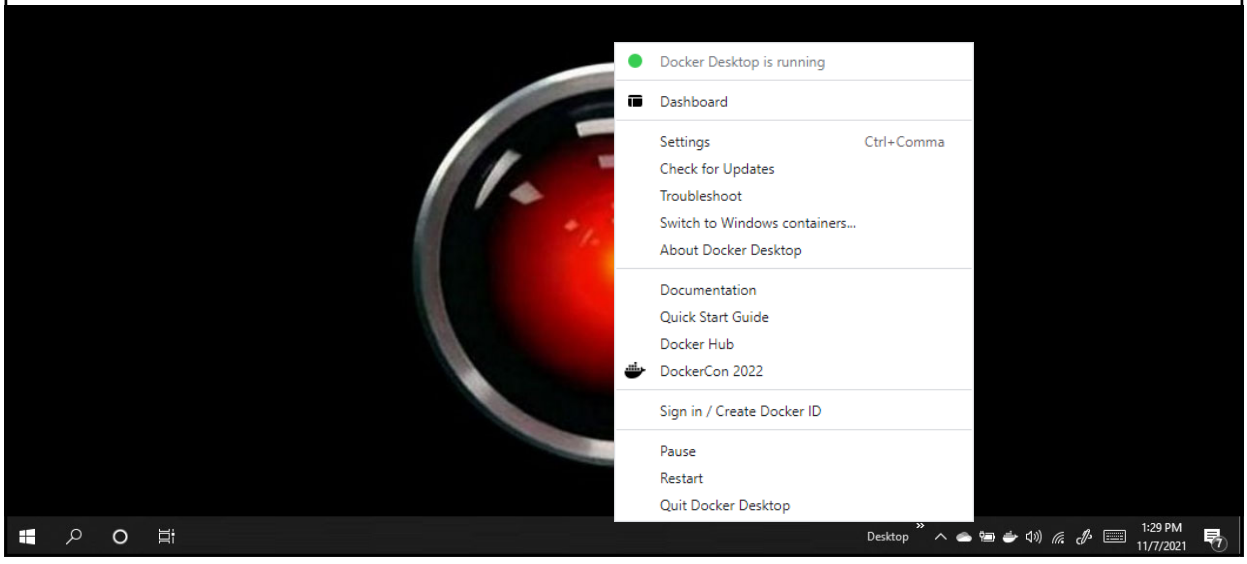

Manual installation steps for olde
docs.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package
Step 4 - Download the Linux kernel
Filter by title
WSL Documentation
Overview
Install
Install WSL
Manual install steps for older versions
Install on Windows Server
Tutorials
Concepts
How-to
Frequently Asked Questions
Troubleshooting
Release Notes
Download PDF
Windows Subsystem for Linux Update Setup
Welcome to the Windows Subsystem for Linux Update Setup Wizard
The Setup Wizard will install Windows Subsystem for Linux Update on your computer. Click Next to continue or Cancel to exit the Setup Wizard.
Back
Next
Cancel
Once the installation is complete, move on to the next step - setting WSL 2 as your
Is this page helpful?
Yes
No
In this article
Step 1 - Enable the Windows Subsystem for Linux
Step 2 - Check requirements for running WSL 2
Step 3 - Enable Virtual Machine feature
Step 4 - Download the Linux kernel update package
Step 5 - Set WSL 2 as your default version
Step 6 - Install your Linux distribution of choice
Troubleshooting installation
Downloading distributions
Install Windows Terminal (optional)
wsl_update_x64.msi


Manual installation steps for olde
docs.microsoft.com/en-us/windows/wsl/install-manual#step-4---download-the-linux-kernel-update-package
Step 4 - Download the Linux kernel
Filter by title
WSL Documentation
Overview
Install
Install WSL
Manual install steps for older versions
Install on Windows Server
Tutorials
Concepts
How-to
Frequently Asked Questions
Troubleshooting
Release Notes
Download PDF
Windows Subsystem for Linux Update Setup
Completed the Windows Subsystem for Linux Update Setup Wizard
Click the Finish button to exit the Setup Wizard.
Back
Finish
Cancel
Once the installation is complete, move on to the next step - setting WSL 2 as your
Is this page helpful?
Yes
No
In this article
Step 1 - Enable the Windows Subsystem for Linux
Step 2 - Check requirements for running WSL 2
Step 3 - Enable Virtual Machine feature
Step 4 - Download the Linux kernel update package
Step 5 - Set WSL 2 as your default version
Step 6 - Install your Linux distribution of choice
Troubleshooting installation
Downloading distributions
Install Windows Terminal (optional)
wsl_update_x64.msi

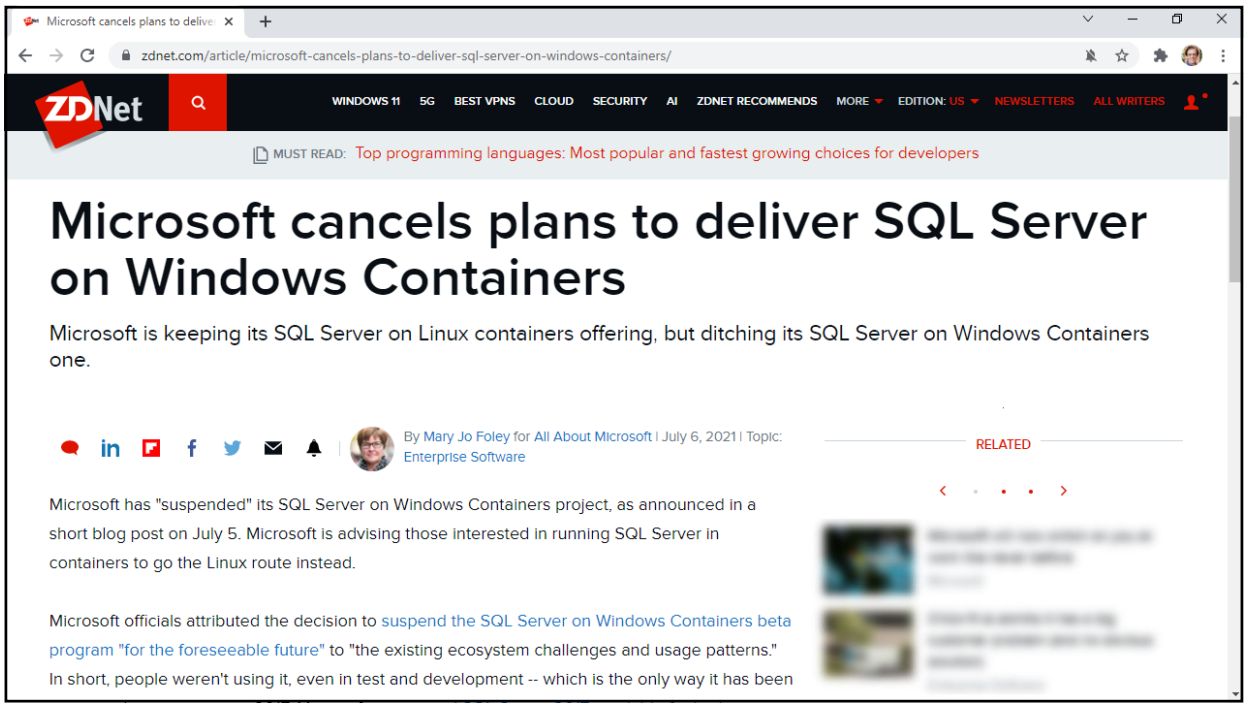
Switch to Windows Containers



Switch to Windows Containers



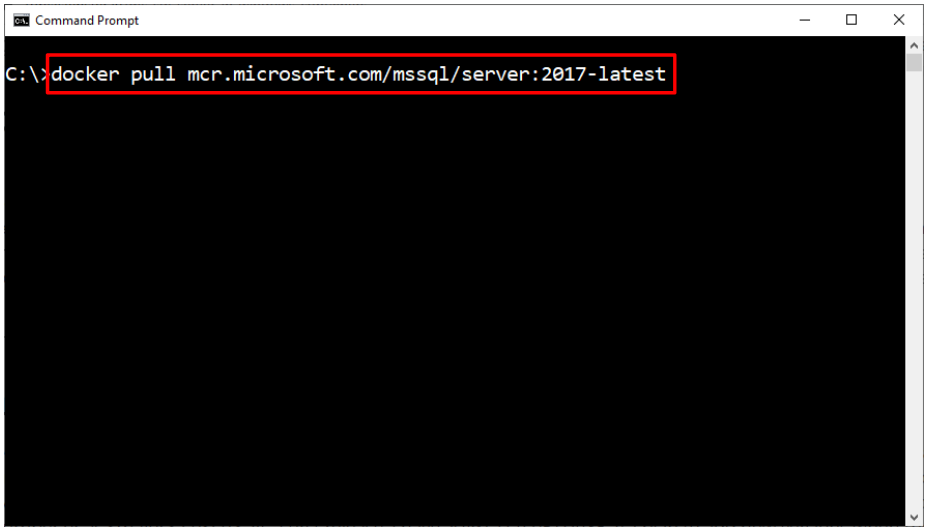

Switch to Windows Containers
Switch to Windows containers
Switch to Windows containers
You are about to switch to Windows containers. Existing containers will continue to run, but you will not be able to manage them until you switch back to linux containers. No data will be lost otherwise.
Do you want to continue?
Don't show this message again
Switch
Cancel
Desktop
1:29 PM
11/7/2021


Microsoft cancels plans to deliver SQL Server on Windows Containers
zdnet.com/article/microsoft-cancels-plans-to-deliver-sql-server-on-windows-containers/
ZDNet
WINDOWS 11
5G
BEST VPNS
CLOUD
SECURITY
AI
ZDNET RECOMMENDS
MORE
EDITION: US
NEWSLETTERS
ALL WRITERS
MUST READ: Top programming languages: Most popular and fastest growing choices for developers
Microsoft cancels plans to deliver SQL Server on Windows Containers
Microsoft is keeping its SQL Server on Linux containers offering, but ditching its SQL Server on Windows Containers one.
By Mary Jo Foley for All About Microsoft | July 6, 2021 | Topic: Enterprise Software
RELATED
Microsoft has "suspended" its SQL Server on Windows Containers project, as announced in a short blog post on July 5. Microsoft is advising those interested in running SQL Server in containers to go the Linux route instead.
Microsoft officials attributed the decision to suspend the SQL Server on Windows Containers beta program "for the foreseeable future" to "the existing ecosystem challenges and usage patterns." In short, people weren't using it, even in test and development -- which is the only way it has been

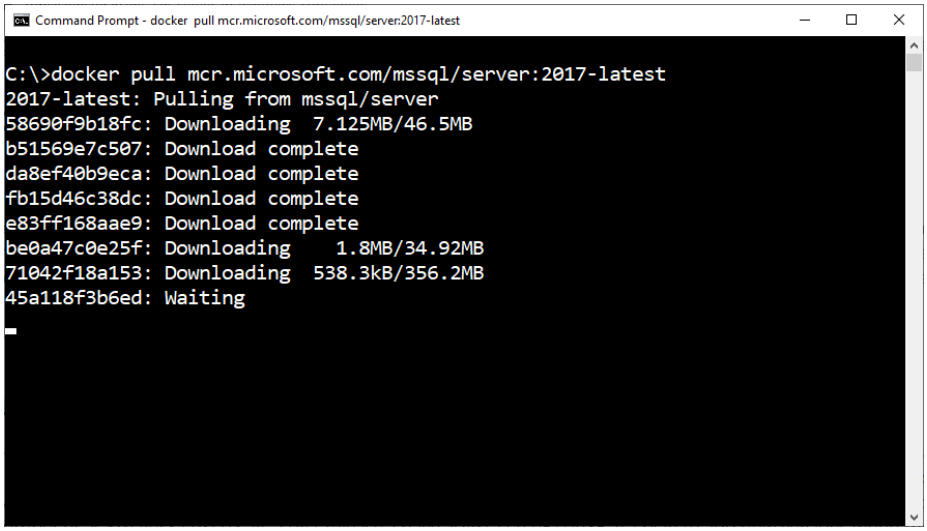
Common Docker Commands

Command	Description
`docker ps`	Show containers
`docker images`	Show images
`docker search`	Search registry for images
`docker pull`	Pull images from registry
`docker run image`	Run command on image in new container
`docker exec container`	Run command on image in running container
`docker logs container`	Show container logs (captures STDERR)
`docker stop container`	Stop a running container
`docker start container`	Start a stopped container
`docker rename container name`	Assign a friendly name to a container
`docker commit container`	Create a new image from a container
`docker tag image name`	Assign a friendly name to an image

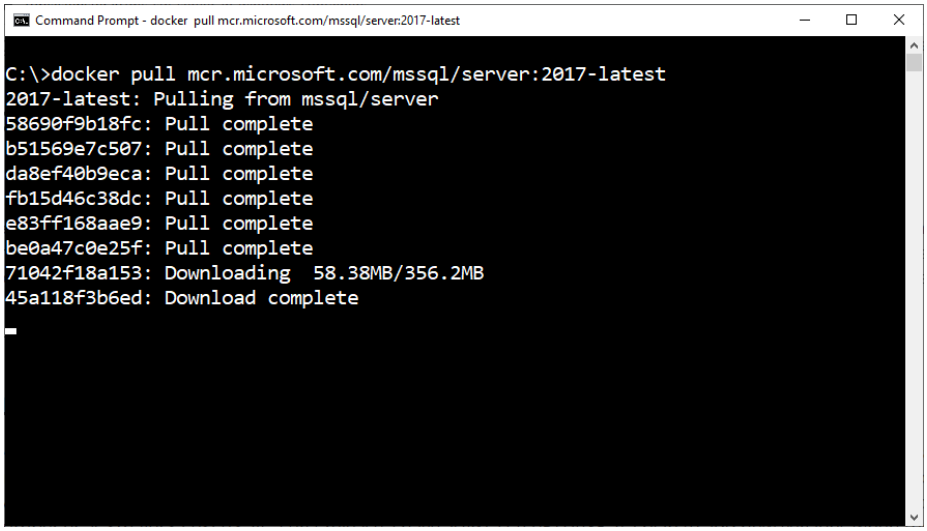
Downloading the Base Images



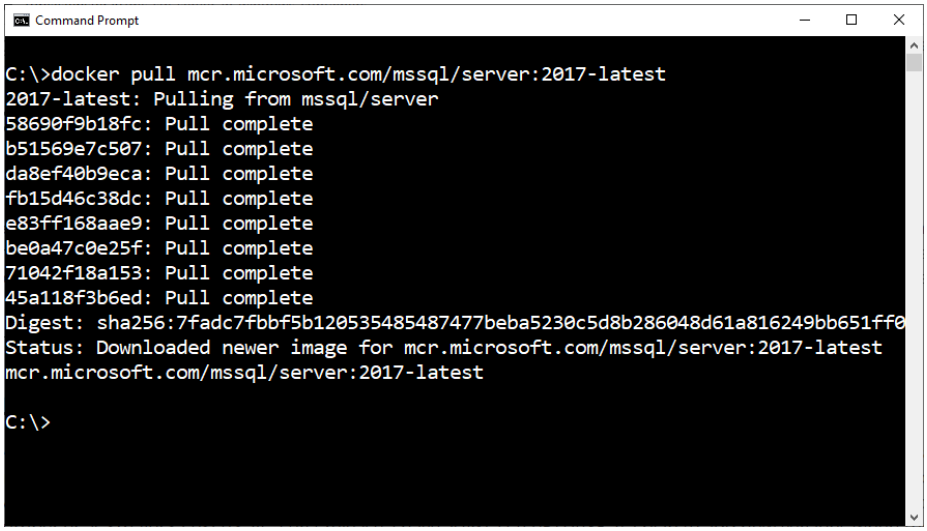
Downloading the Base Images



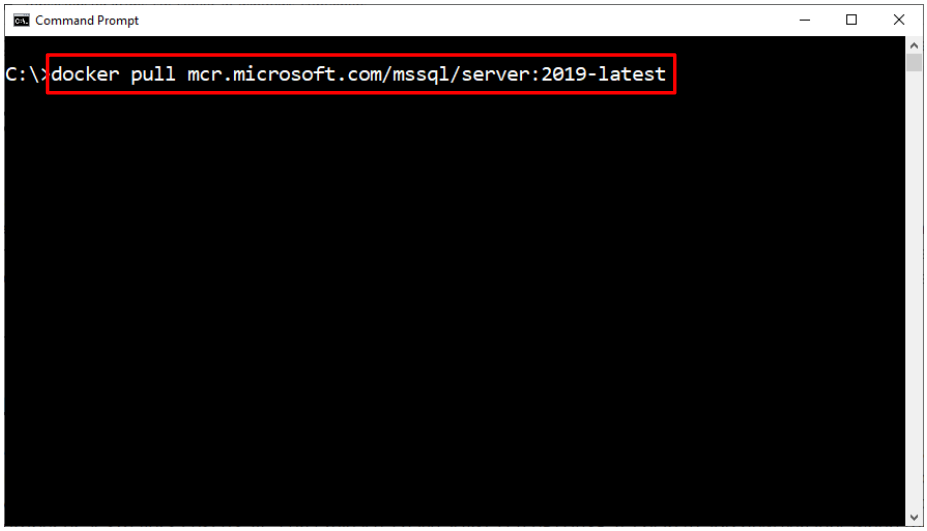
Downloading the Base Images



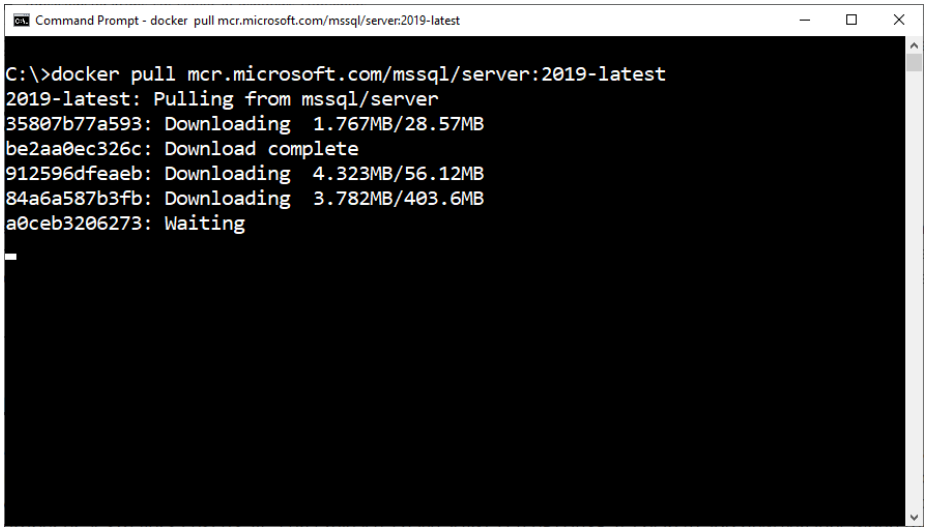
Downloading the Base Images



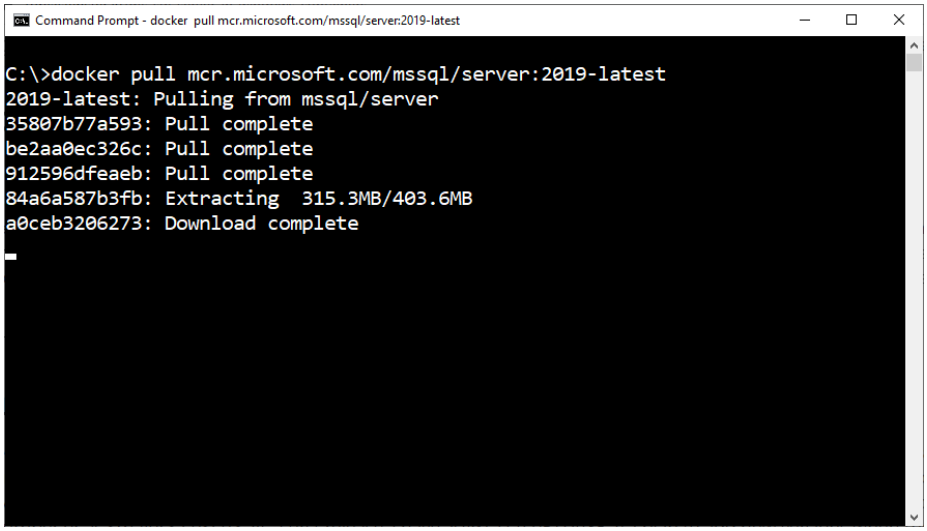
Downloading the Base Images



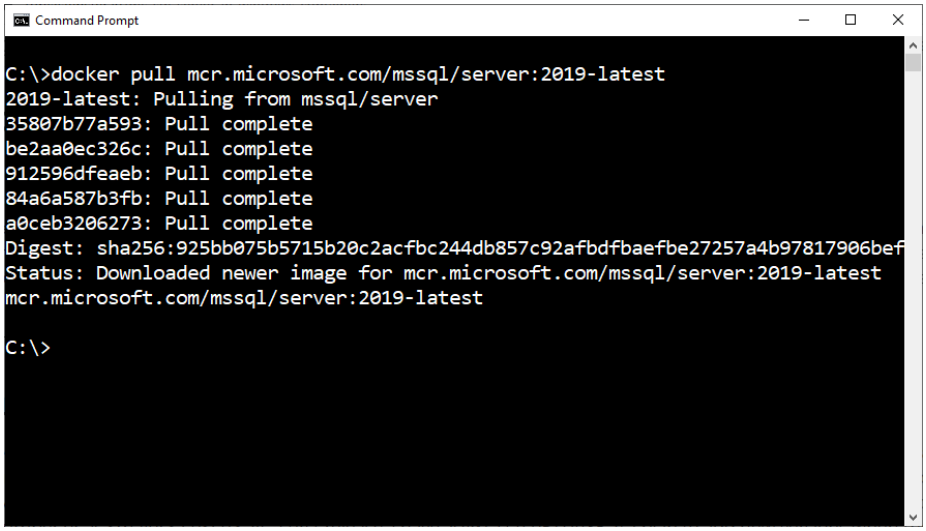
Downloading the Base Images



Downloading the Base Images

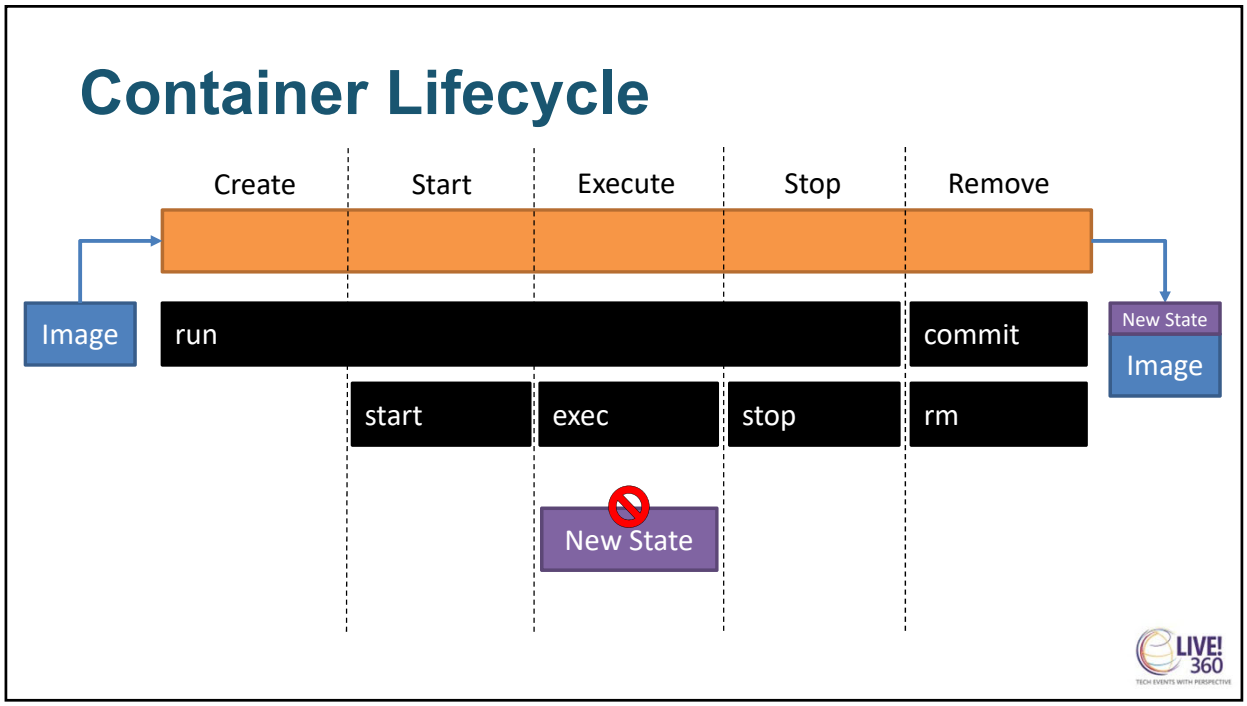


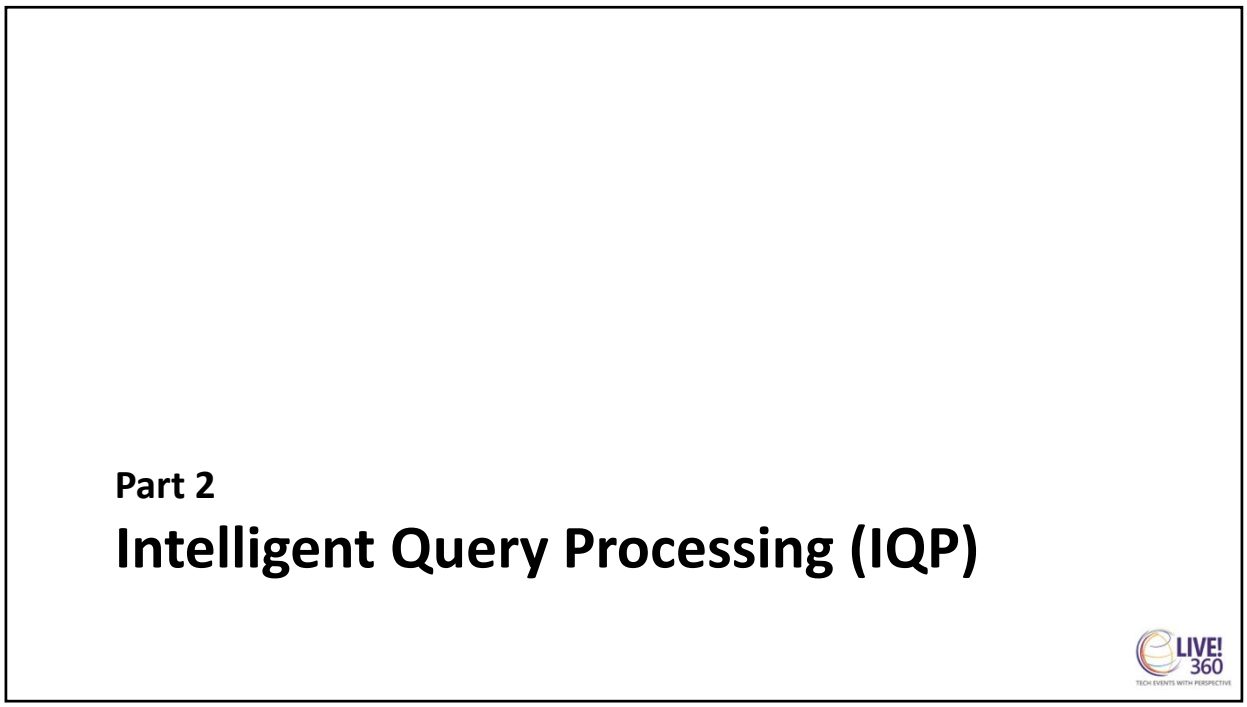
Downloading the Base Images



Docker Live!

demo


Container Lifecycle
Create
Start
Execute
Stop
Remove
Image
run
commit
New State
Image
start
exec
stop
rm
New State


Part 2
Intelligent Query Processing (IQP)

The Intelligent Database

- SQL Server 2019 has groundbreaking performance enhancements
- Intelligent Query Processing
  - Family of features to improve performance of existing workloads
  - Requires minimal (or zero) effort to implement
- Critical parallel workloads improve when running at scale
  - While remaining adaptive to the constantly data flow
  - Helps maintain a predictable level of performance
- Available by default
  - Just flip the compatibility switch (level 150)
  - No need to refactor your code, in most cases

Query Processing Fundamentals

Physical storage

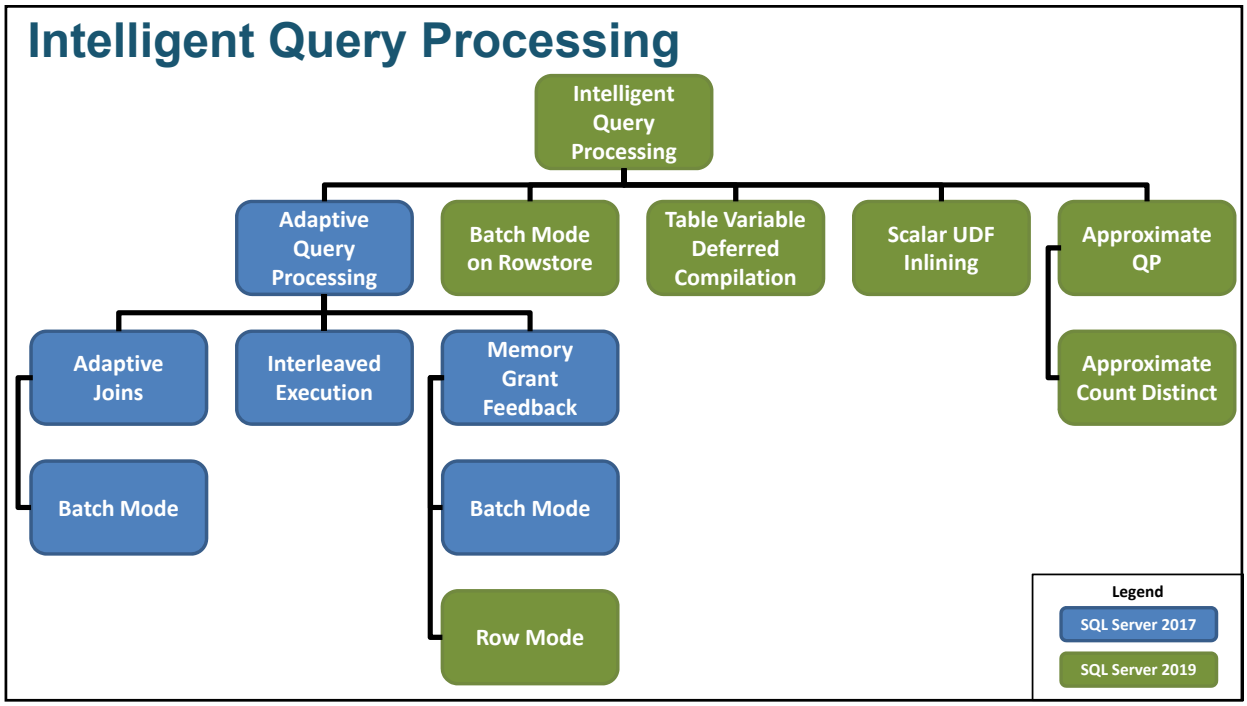
Rowstore	Columnstore
Traditional row-based storage	Columnar storage (compression)
Transactional workloads	Analytic workloads
Read/write fewer rows (more columns)	Read many rows (fewer columns)

Execution mode

Row mode	Batch mode
Process one row at a time	Process multiple rows at a time
Rowstore and columnstore	Columnstore only (pre 2019)

Join type

Nested loops	Hash match
Better for smaller sets of rows	Better for larger sets of rows



Adaptive Query Processing

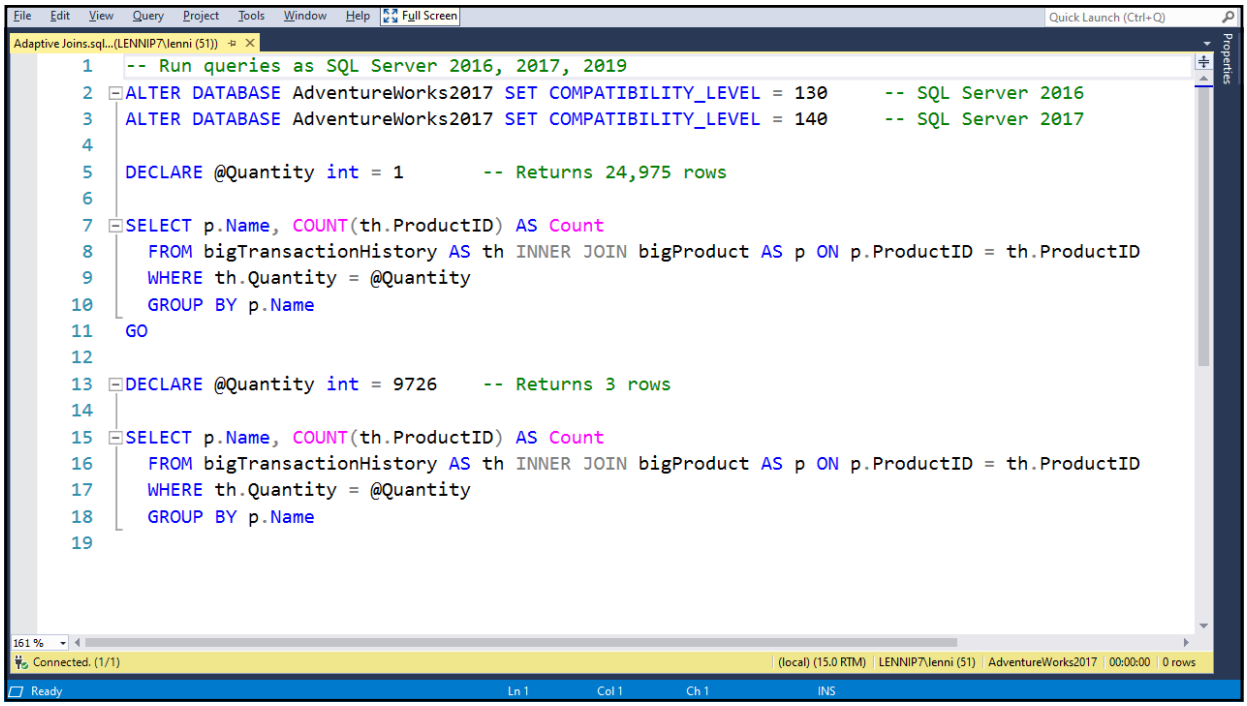
- Bad estimations lead to poor assumptions
  - Prior to SQL Server 2017, the cached query plan will not change
- With AQP
  - Cached query plan adapts optimizations to the runtime workload
- Three features
  - Adaptive joins (batch mode)
  - Interleaved execution
  - Memory grant feedback (batch mode)
- And in SQL Server 2019
  - Memory grant feedback (row mode)

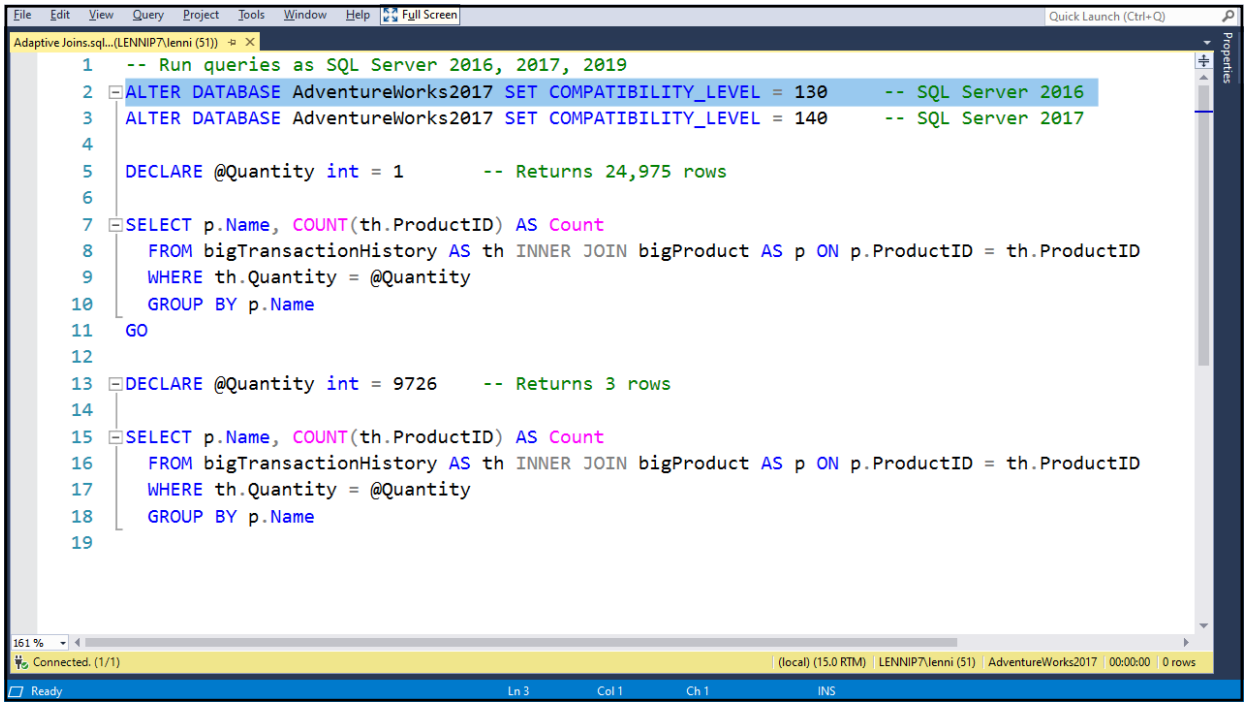
Adaptive Joins

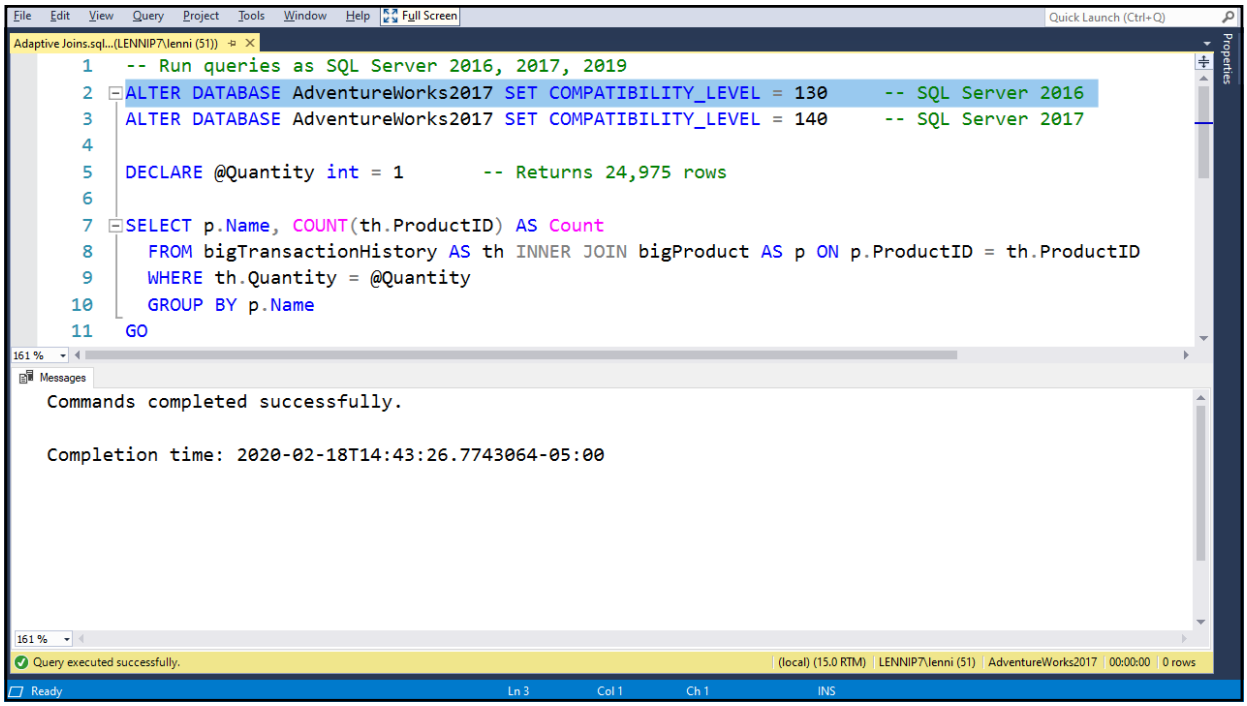
- Dynamically chooses a **hash match** or **nested loops** join at runtime
- How does it work?
  - Examines the statistics of tables being joined
  - Determines a row target
- Based on the target, chooses a hash or loops join
  - If there are too few, uses a loops join
  - If there are too many , uses a hash join
- Available only for batch mode
  - In SQL Server 2017, batch mode requires columnstore
  - In SQL Server 2019, batch mode also available for rowstore

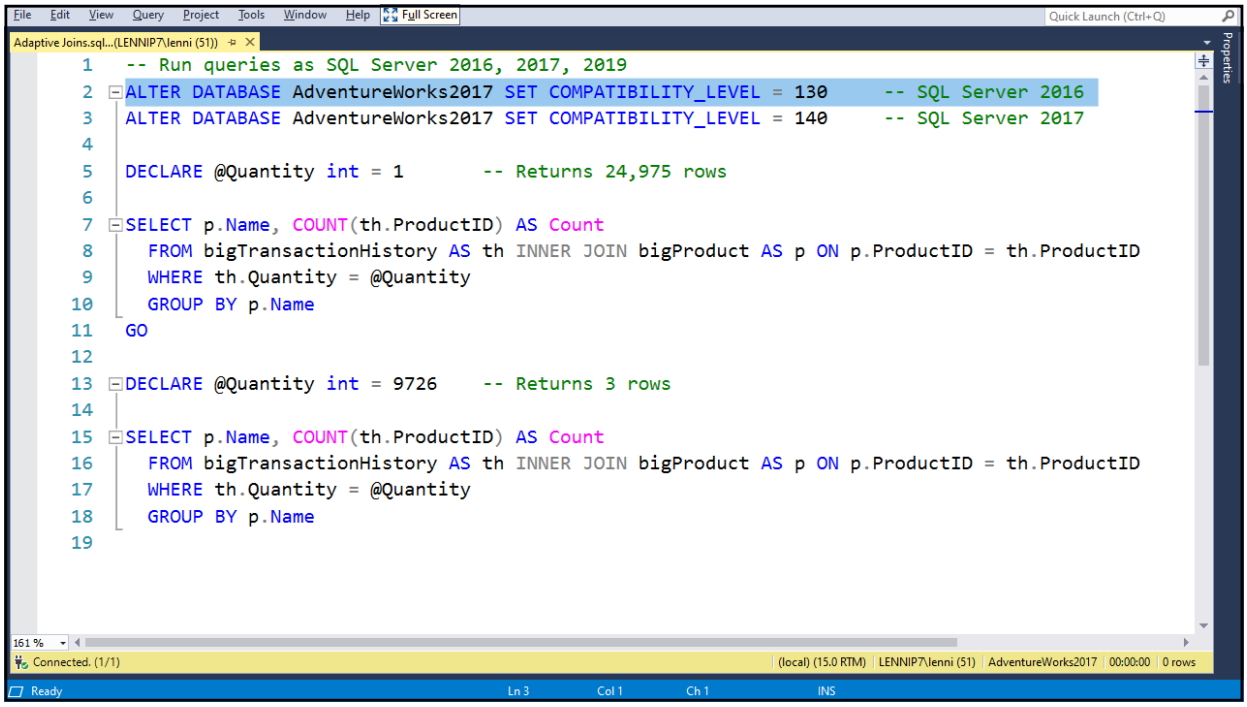
Adaptive Joins

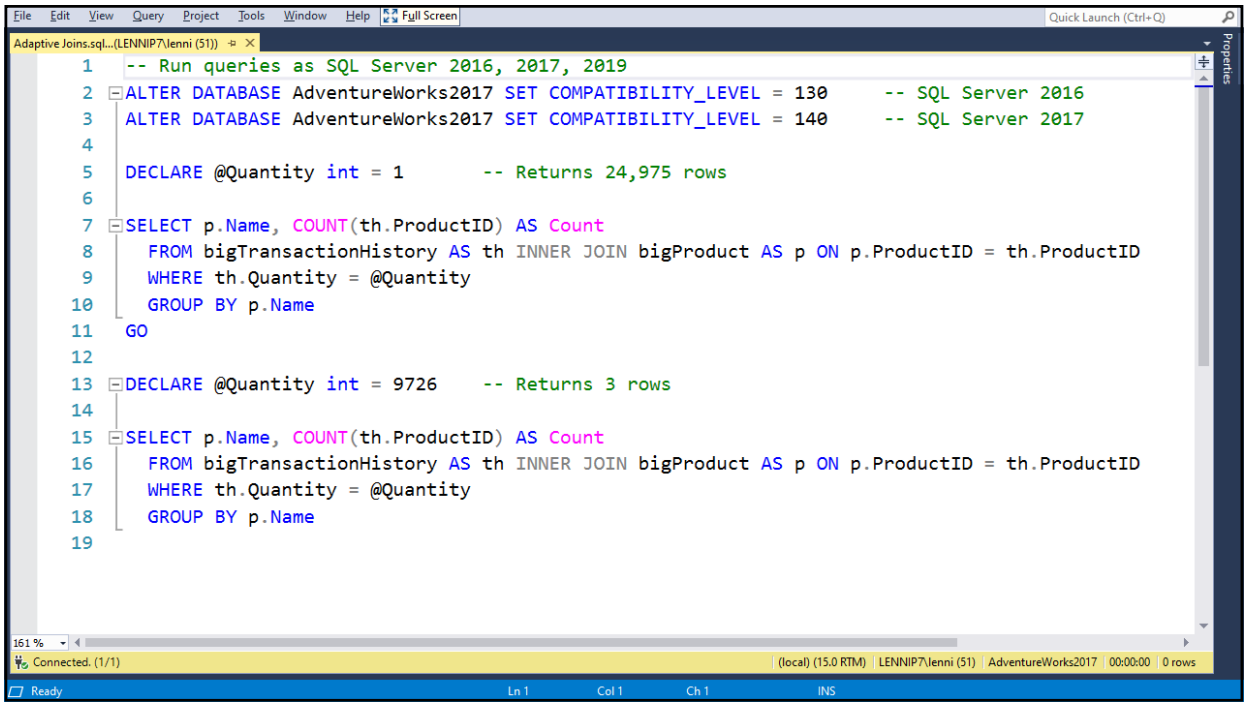
demo

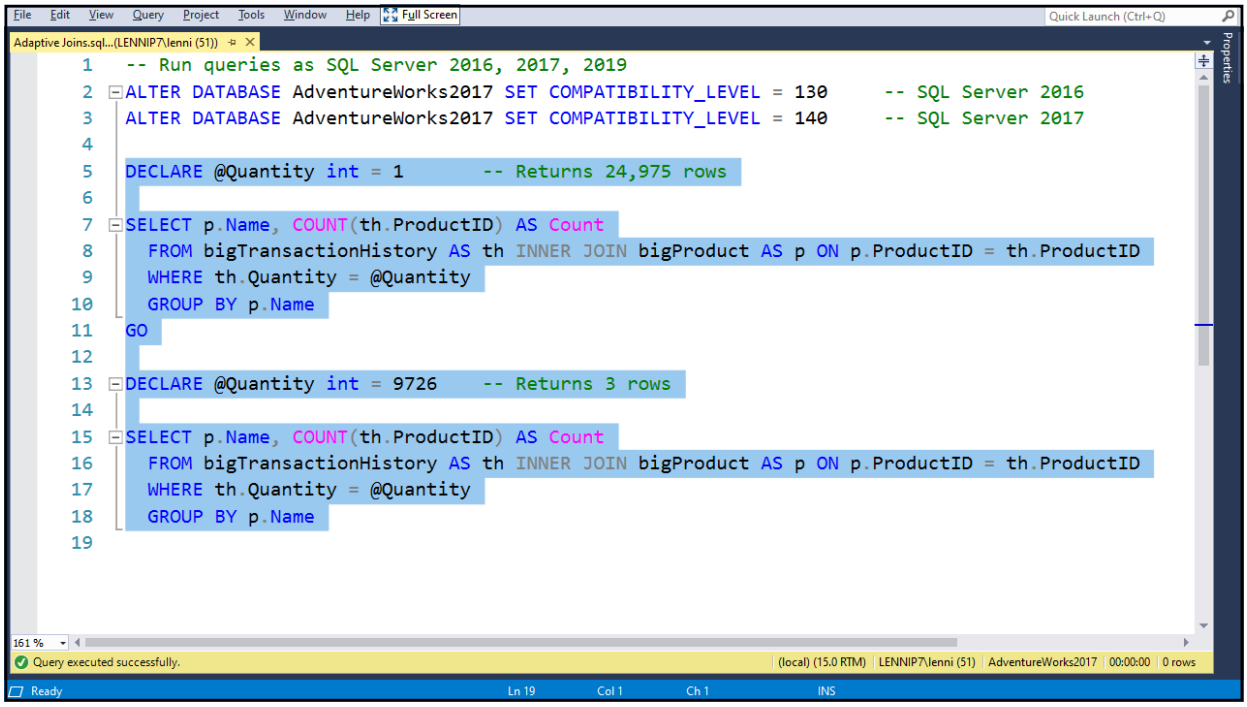

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
Properties
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
5 DECLARE @Quantity int = 1 -- Returns 24,975 rows
6
7 SELECT p.Name, COUNT(th.ProductID) AS Count
8 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
9 WHERE th.Quantity = @Quantity
10 GROUP BY p.Name
11 GO
12
13 DECLARE @Quantity int = 9726 -- Returns 3 rows
14
15 SELECT p.Name, COUNT(th.ProductID) AS Count
16 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
17 WHERE th.Quantity = @Quantity
18 GROUP BY p.Name
19
161 %
Connected. (1/1)
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 0 rows
Ready Ln 1 Col 1 Ch 1 INS

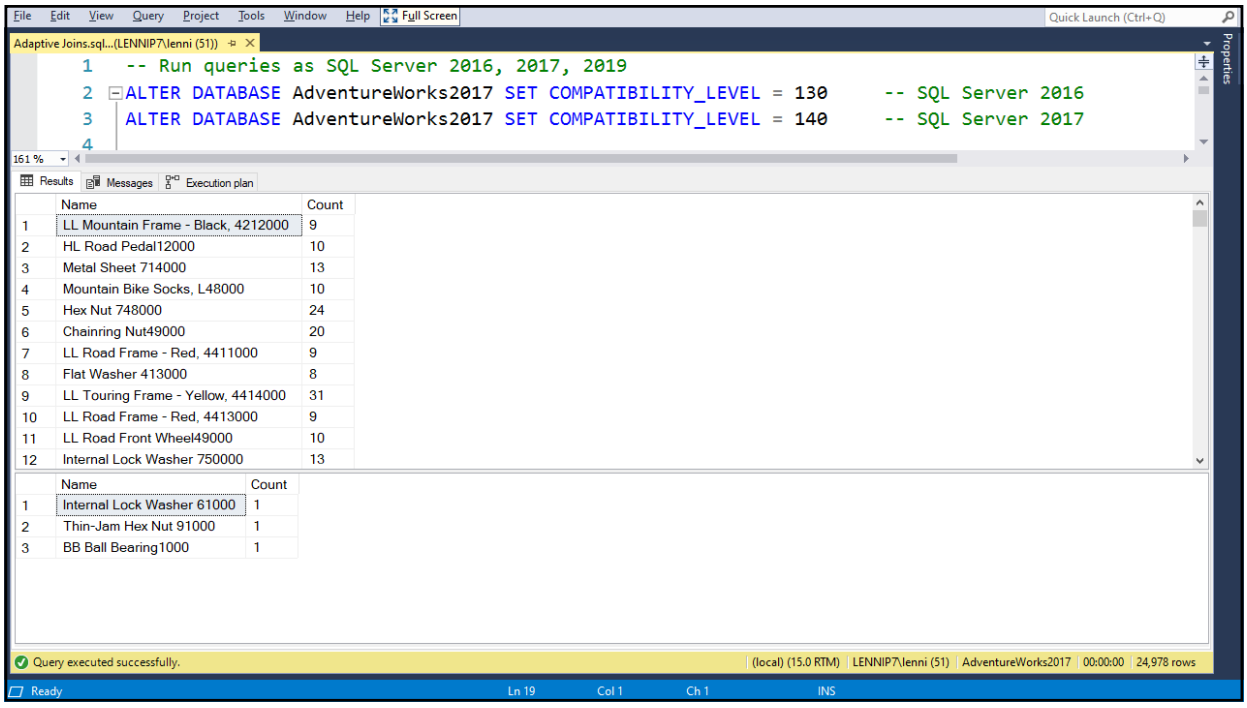

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
Properties
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
5 DECLARE @Quantity int = 1 -- Returns 24,975 rows
6
7 SELECT p.Name, COUNT(th.ProductID) AS Count
8 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
9 WHERE th.Quantity = @Quantity
10 GROUP BY p.Name
11 GO
12
13 DECLARE @Quantity int = 9726 -- Returns 3 rows
14
15 SELECT p.Name, COUNT(th.ProductID) AS Count
16 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
17 WHERE th.Quantity = @Quantity
18 GROUP BY p.Name
19
161 %
Connected. (1/1)
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 0 rows
Ready Ln 3 Col 1 Ch 1 INS

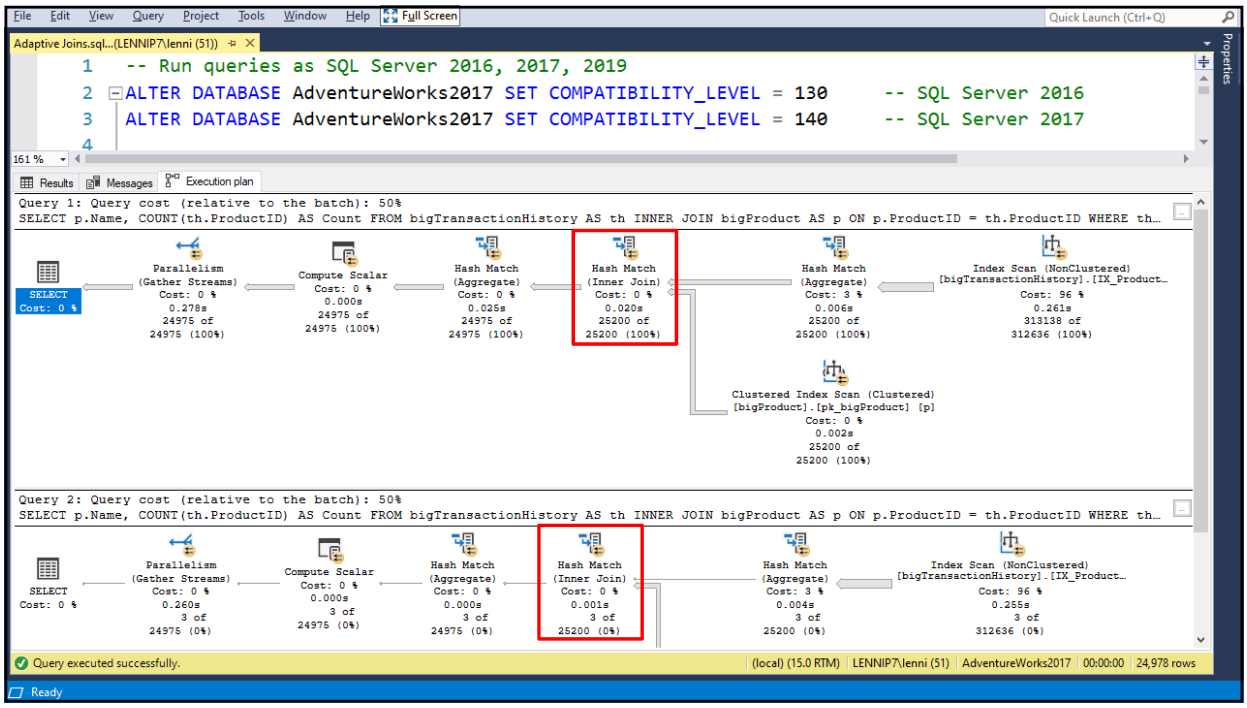

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
Properties
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
5 DECLARE @Quantity int = 1 -- Returns 24,975 rows
6
7 SELECT p.Name, COUNT(th.ProductID) AS Count
8 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
9 WHERE th.Quantity = @Quantity
10 GROUP BY p.Name
11 GO
161 %
Messages
Commands completed successfully.
Completion time: 2020-02-18T14:43:26.7743064-05:00
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 0 rows
Ready Ln 3 Col 1 Ch 1 INS

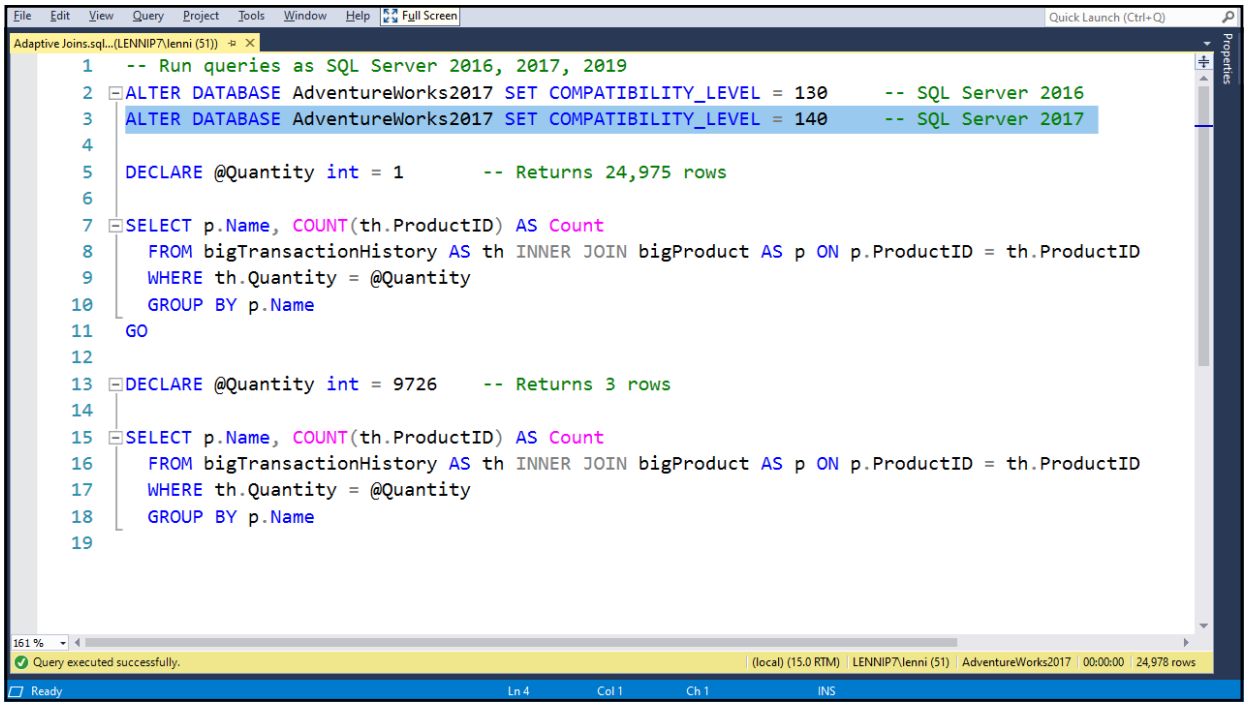

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
Properties
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
5 DECLARE @Quantity int = 1 -- Returns 24,975 rows
6
7 SELECT p.Name, COUNT(th.ProductID) AS Count
8 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
9 WHERE th.Quantity = @Quantity
10 GROUP BY p.Name
11 GO
12
13 DECLARE @Quantity int = 9726 -- Returns 3 rows
14
15 SELECT p.Name, COUNT(th.ProductID) AS Count
16 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
17 WHERE th.Quantity = @Quantity
18 GROUP BY p.Name
19
161 %
Connected. (1/1)
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 0 rows
Ready Ln 3 Col 1 Ch 1 INS

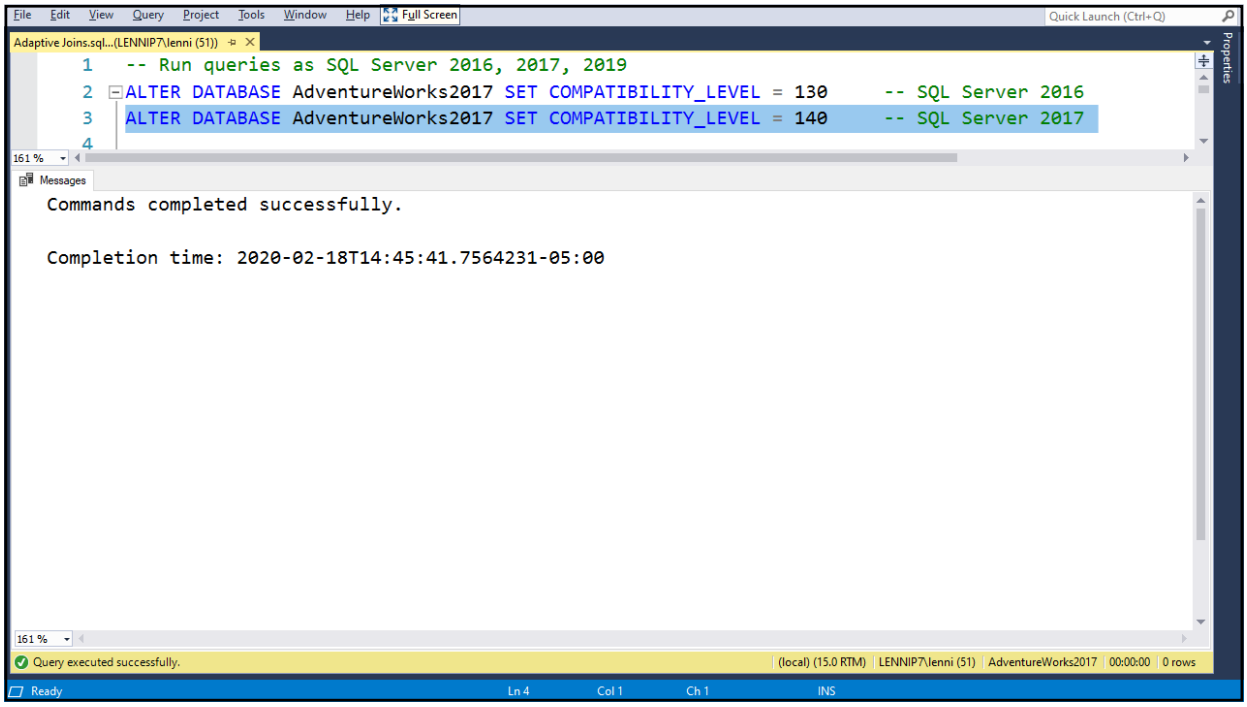

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
Properties
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
5 DECLARE @Quantity int = 1 -- Returns 24,975 rows
6
7 SELECT p.Name, COUNT(th.ProductID) AS Count
8 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
9 WHERE th.Quantity = @Quantity
10 GROUP BY p.Name
11 GO
12
13 DECLARE @Quantity int = 9726 -- Returns 3 rows
14
15 SELECT p.Name, COUNT(th.ProductID) AS Count
16 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
17 WHERE th.Quantity = @Quantity
18 GROUP BY p.Name
19
161 %
Connected. (1/1)
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 0 rows
Ready Ln 1 Col 1 Ch 1 INS

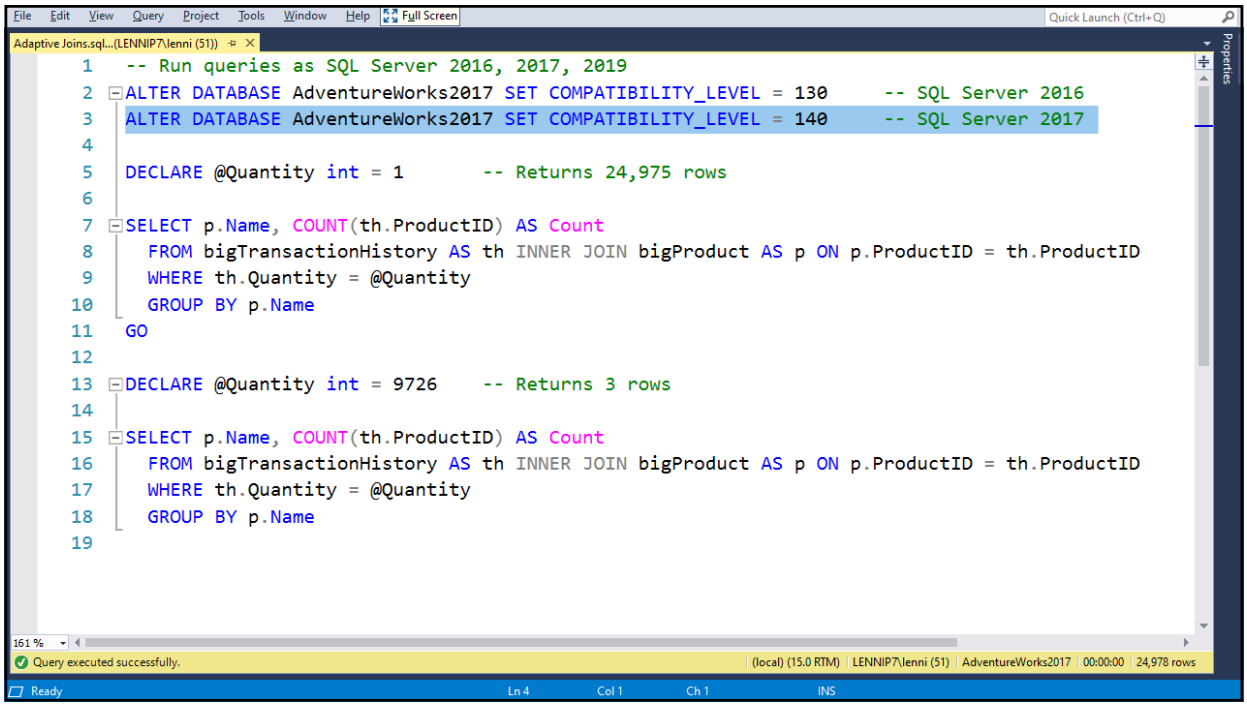

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
Properties
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
5 DECLARE @Quantity int = 1 -- Returns 24,975 rows
6
7 SELECT p.Name, COUNT(th.ProductID) AS Count
8 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
9 WHERE th.Quantity = @Quantity
10 GROUP BY p.Name
11 GO
12
13 DECLARE @Quantity int = 9726 -- Returns 3 rows
14
15 SELECT p.Name, COUNT(th.ProductID) AS Count
16 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
17 WHERE th.Quantity = @Quantity
18 GROUP BY p.Name
19
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 0 rows
Ready Ln 19 Col 1 Ch 1 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Name Count
1 LL Mountain Frame - Black, 4212000 9
2 HL Road Pedal12000 10
3 Metal Sheet 714000 13
4 Mountain Bike Socks, L48000 10
5 Hex Nut 748000 24
6 Chainring Nut49000 20
7 LL Road Frame - Red, 4411000 9
8 Flat Washer 413000 8
9 LL Touring Frame - Yellow, 4414000 31
10 LL Road Frame - Red, 4413000 9
11 LL Road Front Wheel49000 10
12 Internal Lock Washer 750000 13
Name Count
1 Internal Lock Washer 61000 1
2 Thin-Jam Hex Nut 91000 1
3 BB Ball Bearing1000 1
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready Ln 19 Col 1 Ch 1 INS
Properties

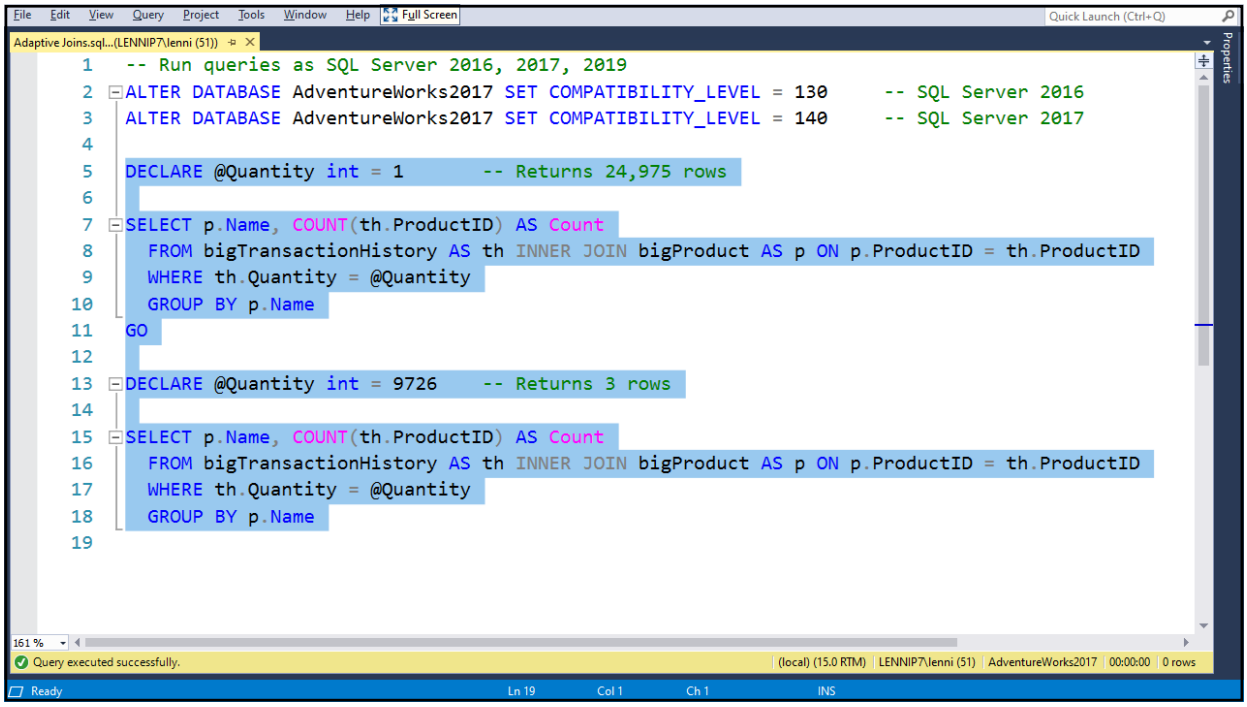

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th...
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.278s 24975 of 24975 (100%)
Compute Scalar Cost: 0 % 0.000s 24975 of 24975 (100%)
Hash Match (Aggregate) Cost: 0 % 0.025s 24975 of 24975 (100%)
Hash Match (Inner Join) Cost: 0 % 0.020s 25200 of 25200 (100%)
Hash Match (Aggregate) Cost: 3 % 0.006s 25200 of 25200 (100%)
Index Scan (NonClustered) [bigTransactionHistory].[IX_Product... Cost: 96 % 0.261s 313138 of 312636 (100%)
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 0 % 0.002s 25200 of 25200 (100%)
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th...
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.260s 3 of 24975 (0%)
Compute Scalar Cost: 0 % 0.000s 3 of 24975 (0%)
Hash Match (Aggregate) Cost: 0 % 0.000s 3 of 24975 (0%)
Hash Match (Inner Join) Cost: 0 % 0.001s 3 of 25200 (0%)
Hash Match (Aggregate) Cost: 3 % 0.004s 3 of 25200 (0%)
Index Scan (NonClustered) [bigTransactionHistory].[IX_Product... Cost: 96 % 0.255s 3 of 312636 (0%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready
Properties


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
Properties
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
5 DECLARE @Quantity int = 1 -- Returns 24,975 rows
6
7 SELECT p.Name, COUNT(th.ProductID) AS Count
8 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
9 WHERE th.Quantity = @Quantity
10 GROUP BY p.Name
11 GO
12
13 DECLARE @Quantity int = 9726 -- Returns 3 rows
14
15 SELECT p.Name, COUNT(th.ProductID) AS Count
16 FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
17 WHERE th.Quantity = @Quantity
18 GROUP BY p.Name
19
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready Ln 4 Col 1 Ch 1 INS

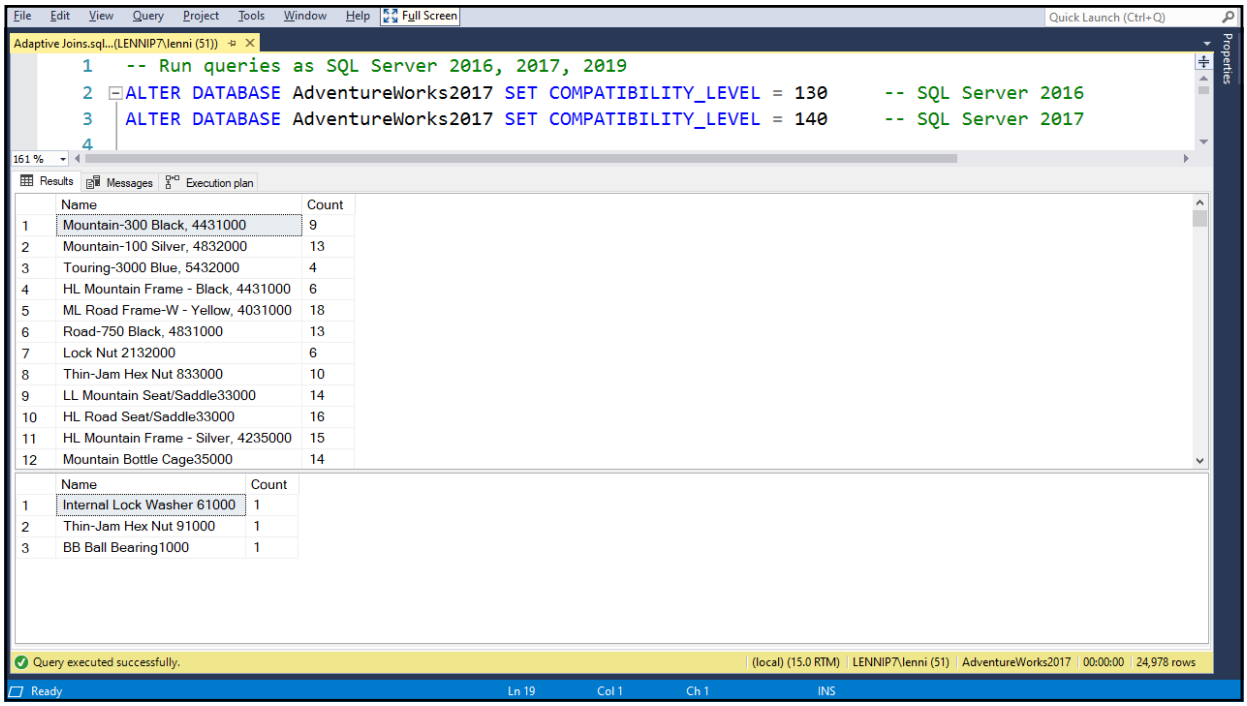

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
Properties
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Messages
Commands completed successfully.
Completion time: 2020-02-18T14:45:41.7564231-05:00
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 0 rows
Ready Ln 4 Col 1 Ch 1 INS

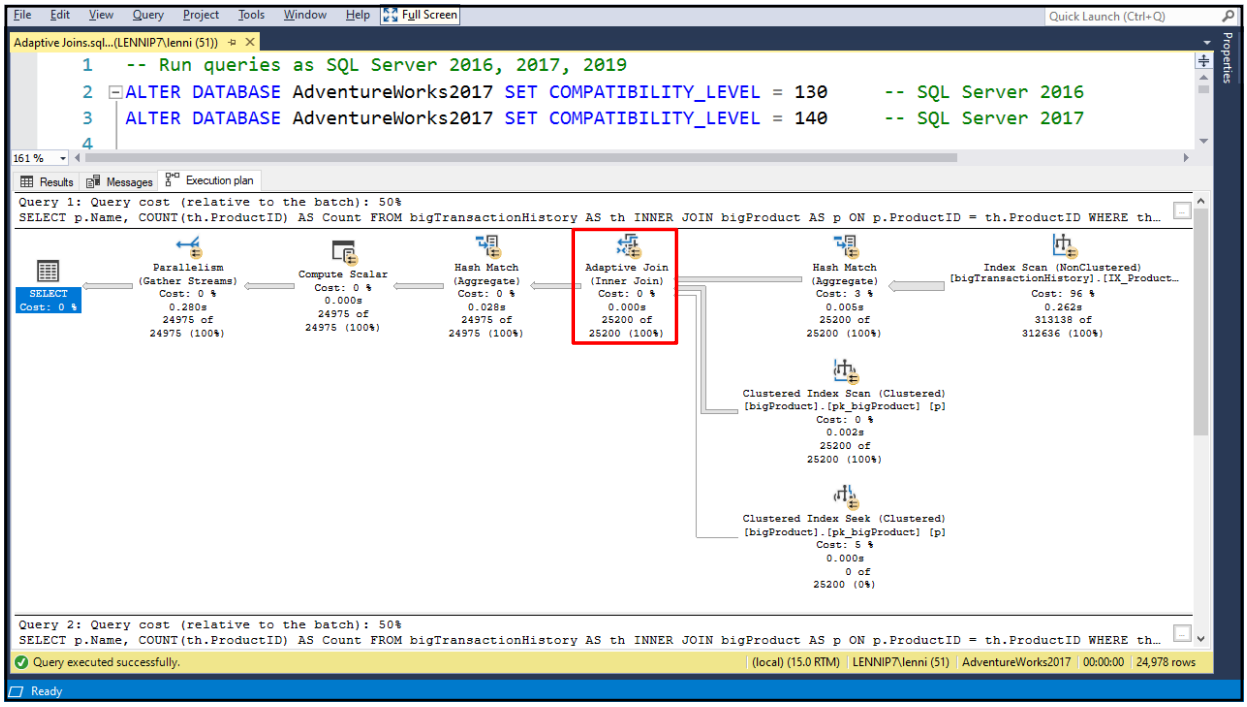


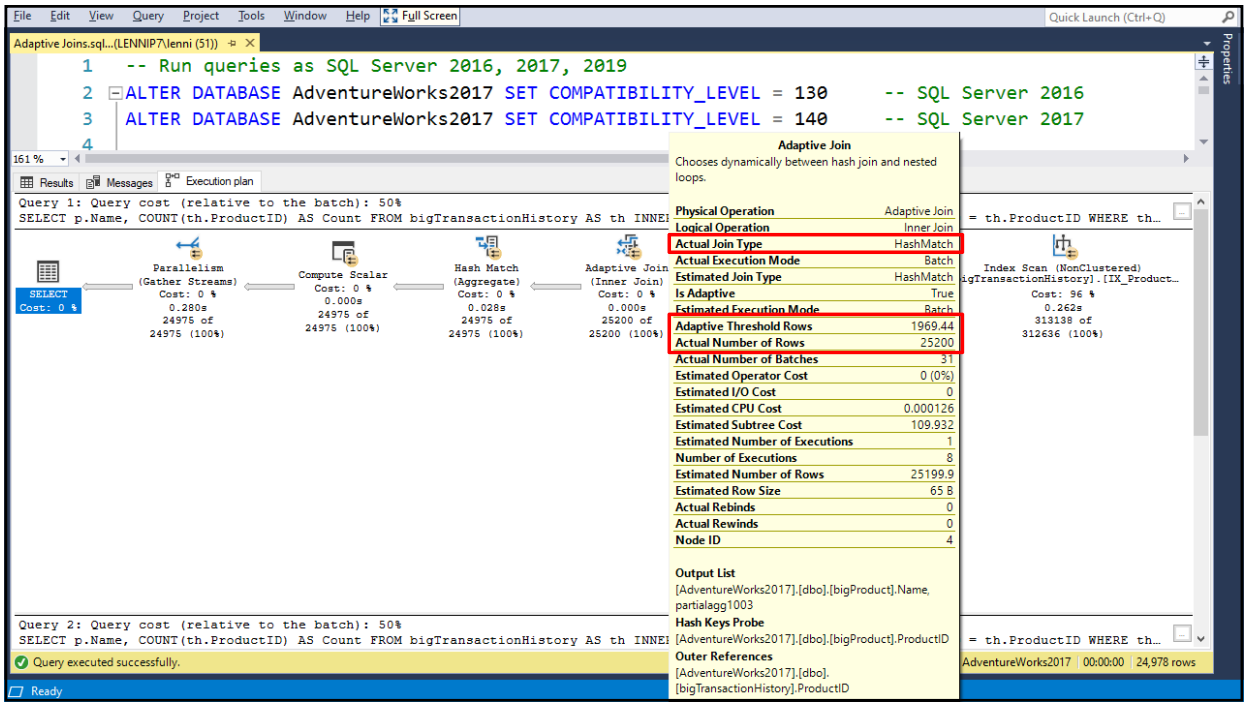
```
-- Run queries as SQL Server 2016, 2017, 2019
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130     -- SQL Server 2016
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140     -- SQL Server 2017

DECLARE @Quantity int = 1       -- Returns 24,975 rows

SELECT p.Name, COUNT(th.ProductID) AS Count
  FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
  WHERE th.Quantity = @Quantity
  GROUP BY p.Name
GO

DECLARE @Quantity int = 9726    -- Returns 3 rows

SELECT p.Name, COUNT(th.ProductID) AS Count
  FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
  WHERE th.Quantity = @Quantity
  GROUP BY p.Name
```

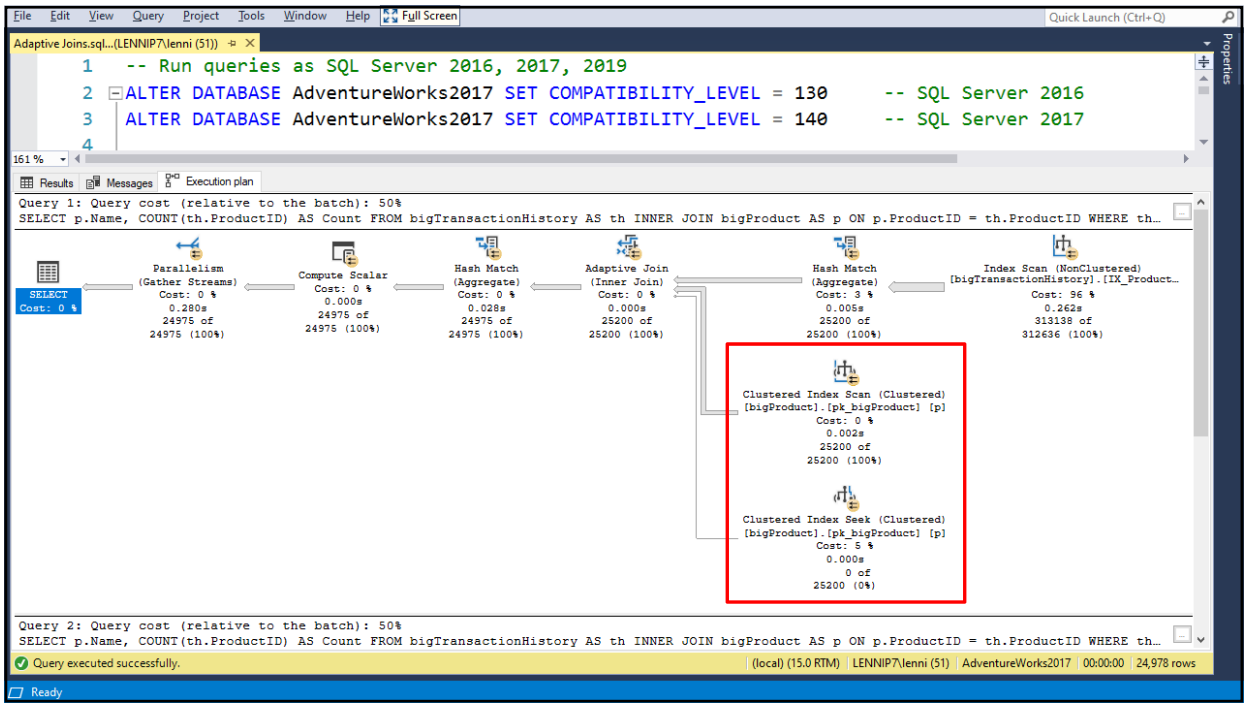


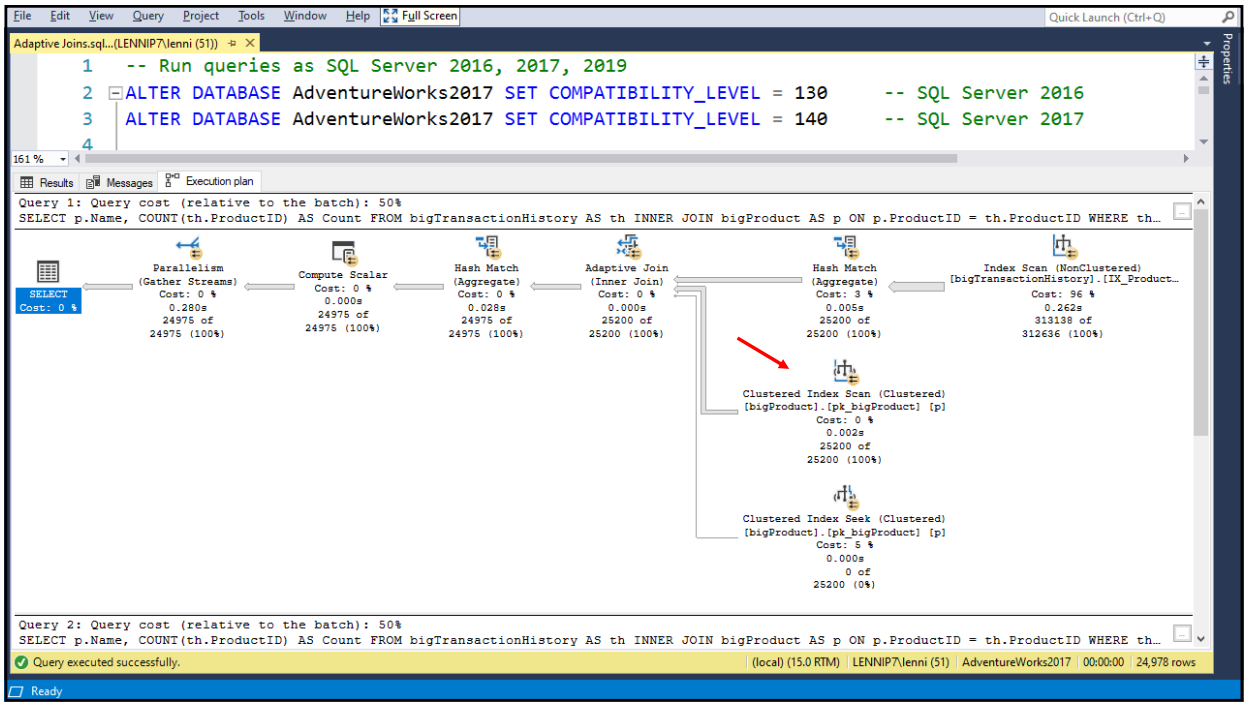
```
-- Run queries as SQL Server 2016, 2017, 2019
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130     -- SQL Server 2016
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140     -- SQL Server 2017

DECLARE @Quantity int = 1       -- Returns 24,975 rows

SELECT p.Name, COUNT(th.ProductID) AS Count
  FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
  WHERE th.Quantity = @Quantity
  GROUP BY p.Name
GO

DECLARE @Quantity int = 9726    -- Returns 3 rows

SELECT p.Name, COUNT(th.ProductID) AS Count
  FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
  WHERE th.Quantity = @Quantity
  GROUP BY p.Name
```

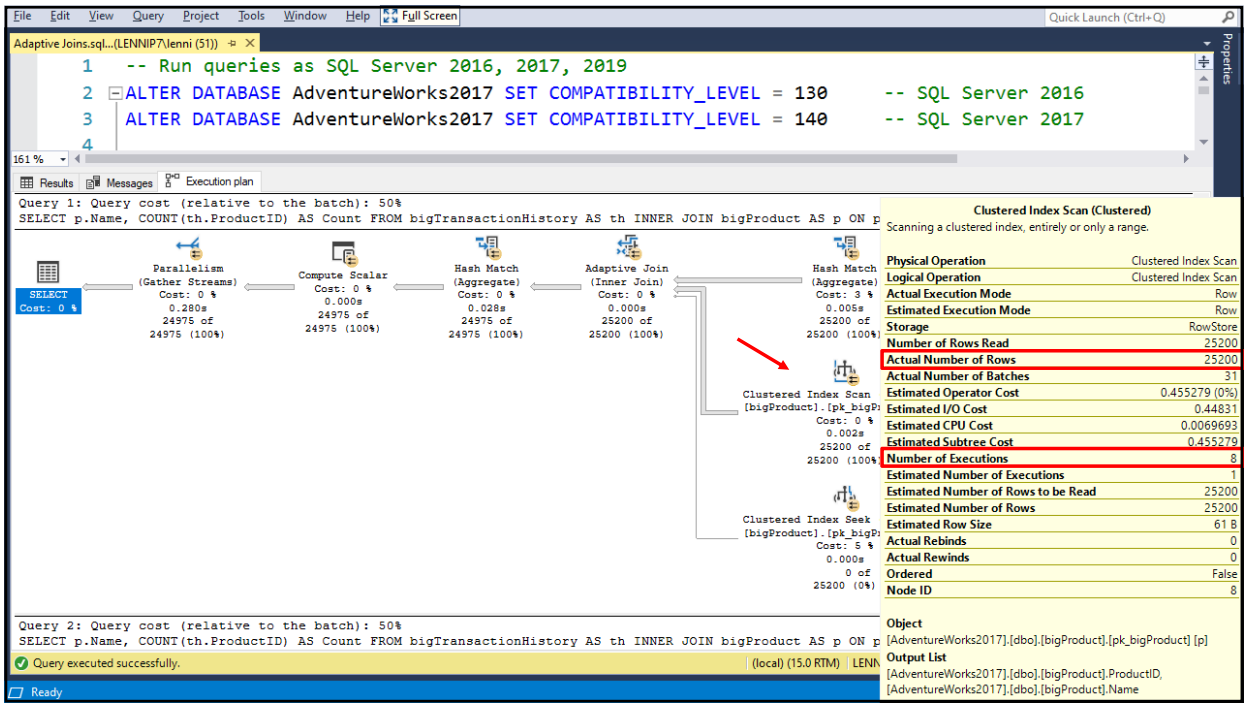

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Name Count
1 Mountain-300 Black, 4431000 9
2 Mountain-100 Silver, 4832000 13
3 Touring-3000 Blue, 5432000 4
4 HL Mountain Frame - Black, 4431000 6
5 ML Road Frame-W - Yellow, 4031000 18
6 Road-750 Black, 4831000 13
7 Lock Nut 2132000 6
8 Thin-Jam Hex Nut 833000 10
9 LL Mountain Seat/Saddle33000 14
10 HL Road Seat/Saddle33000 16
11 HL Mountain Frame - Silver, 4235000 15
12 Mountain Bottle Cage35000 14
Name Count
1 Internal Lock Washer 61000 1
2 Thin-Jam Hex Nut 91000 1
3 BB Ball Bearing1000 1
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready Ln 19 Col 1 Ch 1 INS
Properties

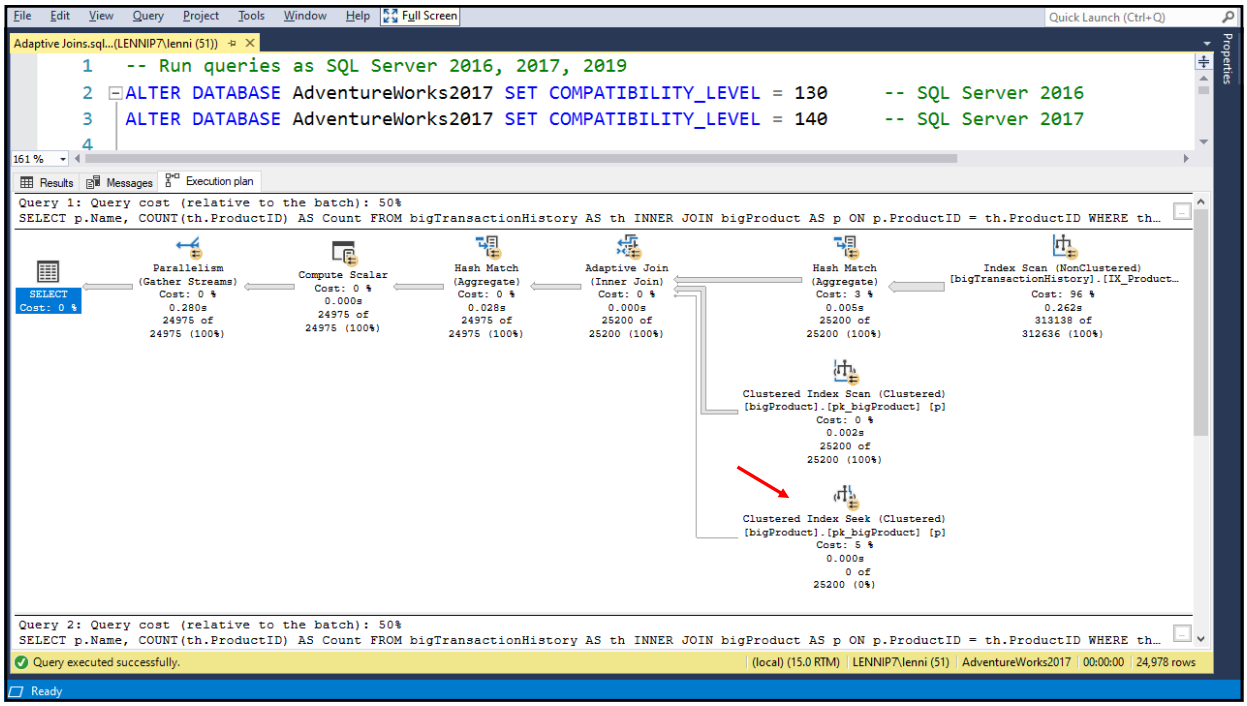

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th...
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.280s 24975 of 24975 (100%)
Compute Scalar Cost: 0 % 0.000s 24975 of 24975 (100%)
Hash Match (Aggregate) Cost: 0 % 0.028s 24975 of 24975 (100%)
Adaptive Join (Inner Join) Cost: 0 % 0.000s 25200 of 25200 (100%)
Hash Match (Aggregate) Cost: 3 % 0.005s 25200 of 25200 (100%)
Index Scan (NonClustered) [bigTransactionHistory].[IX_Product... Cost: 96 % 0.262s 313138 of 312636 (100%)
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 0 % 0.002s 25200 of 25200 (100%)
Clustered Index Seek (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 5 % 0.000s 0 of 25200 (0%)
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th...
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready
Properties

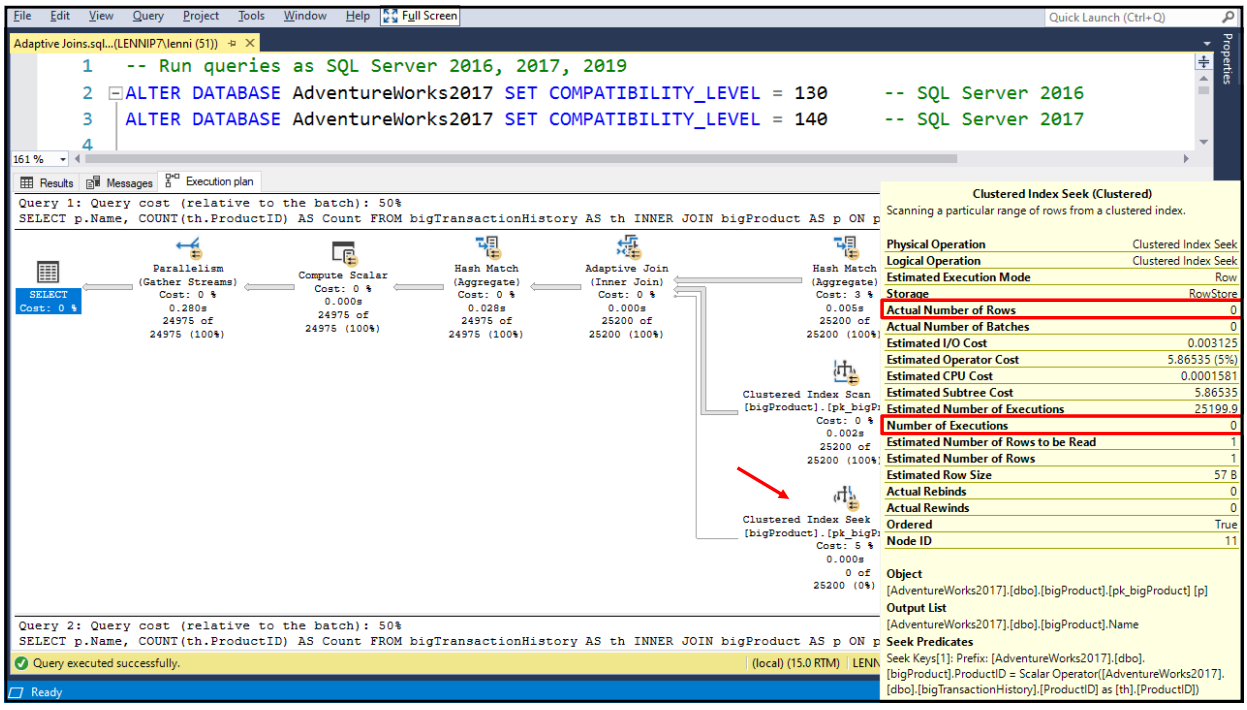

-- Run queries as SQL Server 2016, 2017, 2019
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
Adaptive Join
Chooses dynamically between hash join and nested loops.
Physical Operation Adaptive Join
Logical Operation Inner Join
Actual Join Type HashMatch
Actual Execution Mode Batch
Estimated Join Type HashMatch
Is Adaptive True
Estimated Execution Mode Batch
Adaptive Threshold Rows 1969.44
Actual Number of Rows 25200
Actual Number of Batches 31
Estimated Operator Cost 0 (0%)
Estimated I/O Cost 0
Estimated CPU Cost 0.000126
Estimated Subtree Cost 109.932
Estimated Number of Executions 1
Number of Executions 8
Estimated Number of Rows 25199.9
Estimated Row Size 65 B
Actual Rebinds 0
Actual Rewinds 0
Node ID 4
Output List
[AdventureWorks2017].[dbo].[bigProduct].Name, partialagg1003
Hash Keys Probe
[AdventureWorks2017].[dbo].[bigProduct].ProductID
Outer References
[AdventureWorks2017].[dbo].[bigTransactionHistory].ProductID
Query executed successfully.
Ready

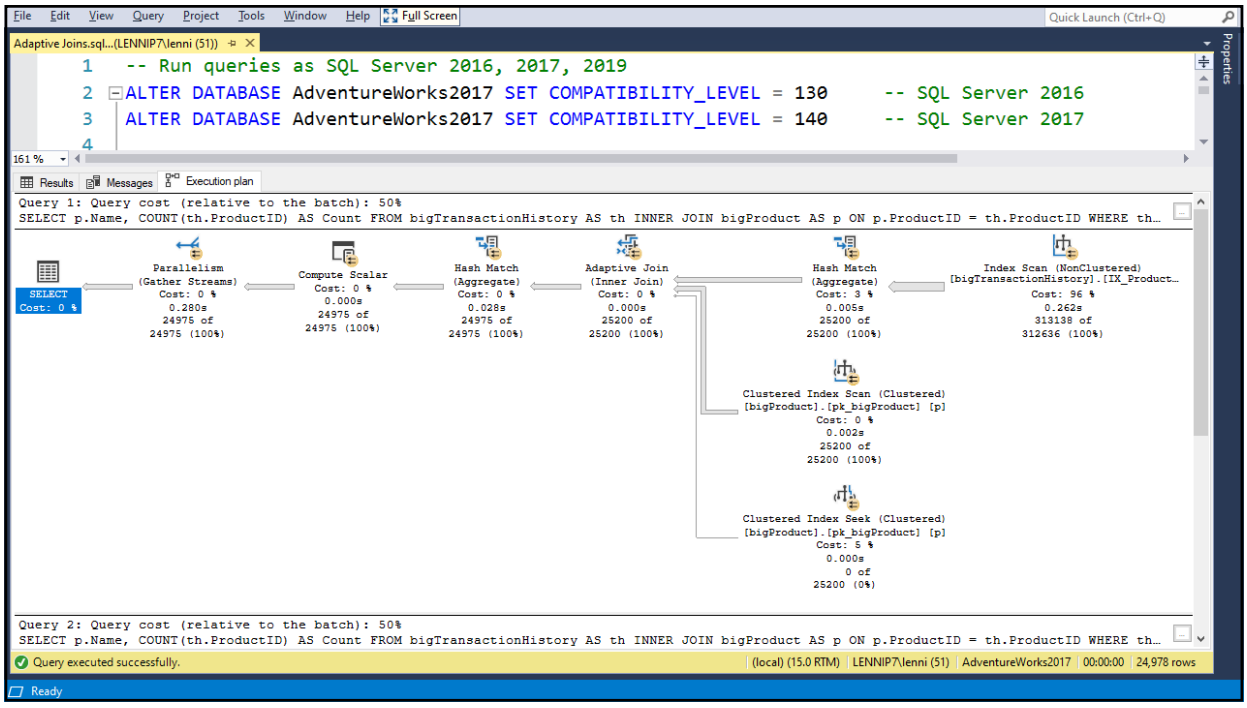

-- Run queries as SQL Server 2016, 2017, 2019
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p]
Clustered Index Seek (Clustered) [bigProduct].[pk_bigProduct] [p]
Query executed successfully.
Ready

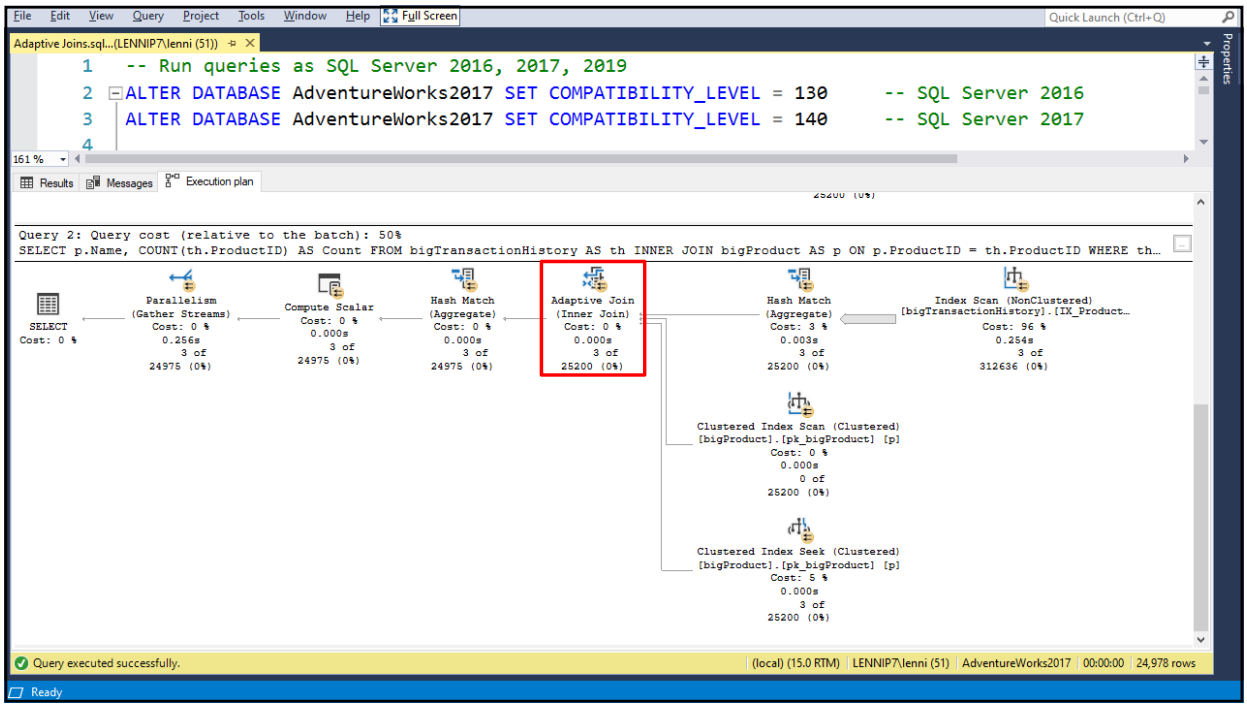

Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
Results
Messages
Execution plan
Query 1: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th…
SELECT
Parallelism (Gather Streams)
Compute Scalar
Hash Match (Aggregate)
Adaptive Join (Inner Join)
Hash Match (Aggregate)
Index Scan (NonClustered) [bigTransactionHistory].[IX_Product…
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p]
Clustered Index Seek (Clustered) [bigProduct].[pk_bigProduct] [p]
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th…
Query executed successfully.
(local) (15.0 RTM)
LENNIP7\lenni (51)
AdventureWorks2017
00:00:00
24,978 rows
Ready

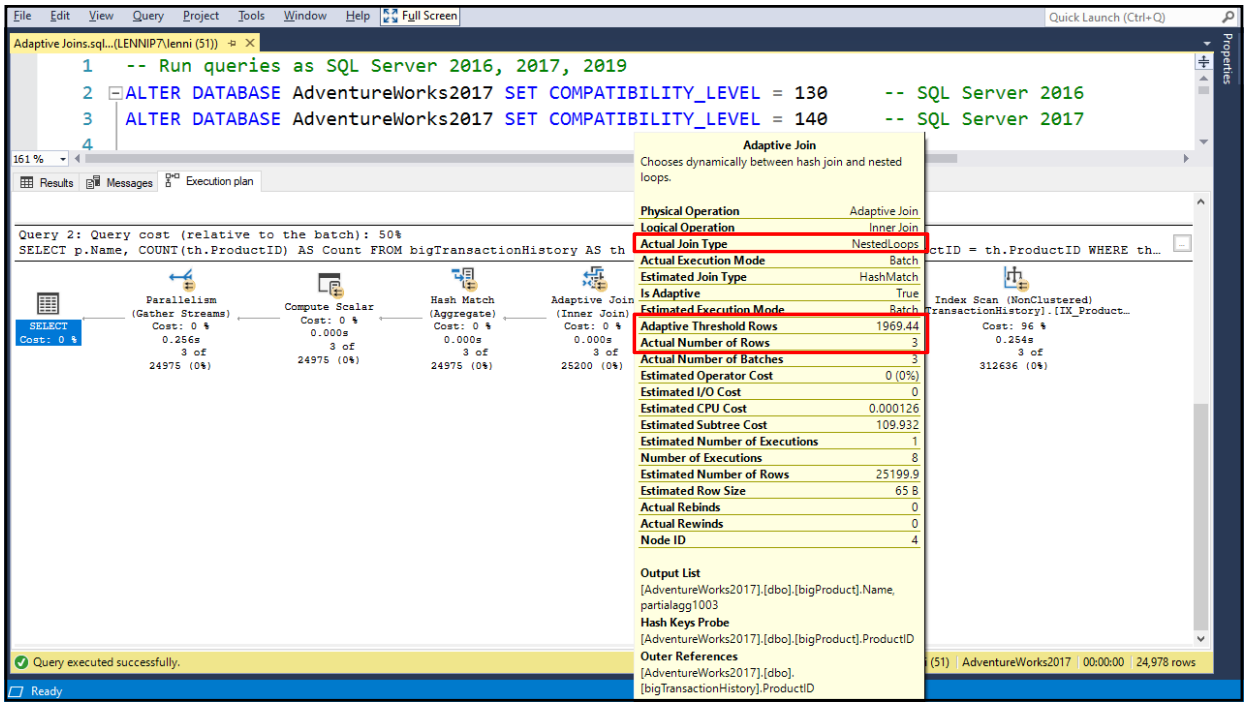

Clustered Index Scan (Clustered)
Scanning a clustered index, entirely or only a range.
Physical Operation: Clustered Index Scan
Logical Operation: Clustered Index Scan
Actual Execution Mode: Row
Estimated Execution Mode: Row
Storage: RowStore
Number of Rows Read: 25200
Actual Number of Rows: 25200
Actual Number of Batches: 31
Estimated Operator Cost: 0.455279 (0%)
Estimated I/O Cost: 0.44831
Estimated CPU Cost: 0.0069693
Estimated Subtree Cost: 0.455279
Number of Executions: 8
Estimated Number of Executions: 1
Estimated Number of Rows to be Read: 25200
Estimated Number of Rows: 25200
Estimated Row Size: 61 B
Actual Rebinds: 0
Actual Rewinds: 0
Ordered: False
Node ID: 8
Object
[AdventureWorks2017].[dbo].[bigProduct].[pk_bigProduct] [p]
Output List
[AdventureWorks2017].[dbo].[bigProduct].ProductID,
[AdventureWorks2017].[dbo].[bigProduct].Name

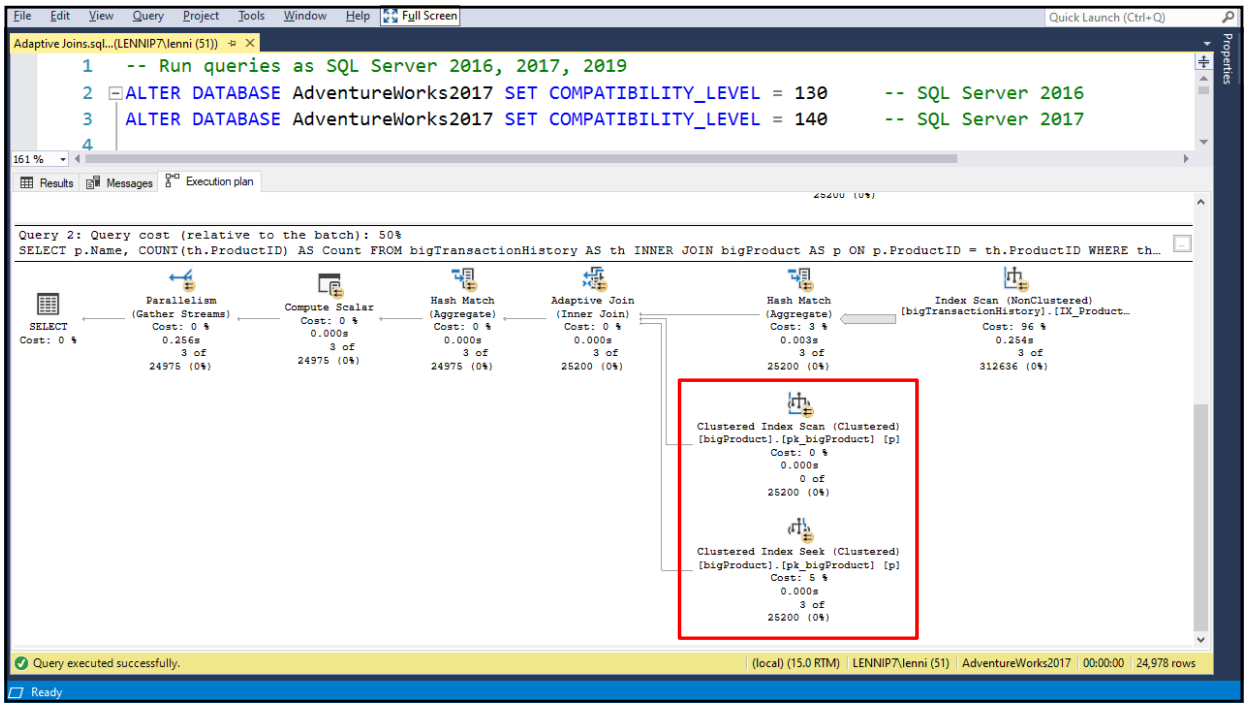

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th…
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.280s 24975 of 24975 (100%)
Compute Scalar Cost: 0 % 0.000s 24975 of 24975 (100%)
Hash Match (Aggregate) Cost: 0 % 0.028s 24975 of 24975 (100%)
Adaptive Join (Inner Join) Cost: 0 % 0.000s 25200 of 25200 (100%)
Hash Match (Aggregate) Cost: 3 % 0.005s 25200 of 25200 (100%)
Index Scan (NonClustered) [bigTransactionHistory].[IX_Product… Cost: 96 % 0.262s 313138 of 312636 (100%)
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 0 % 0.002s 25200 of 25200 (100%)
Clustered Index Seek (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 5 % 0.000s 0 of 25200 (0%)
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th…
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready
Properties

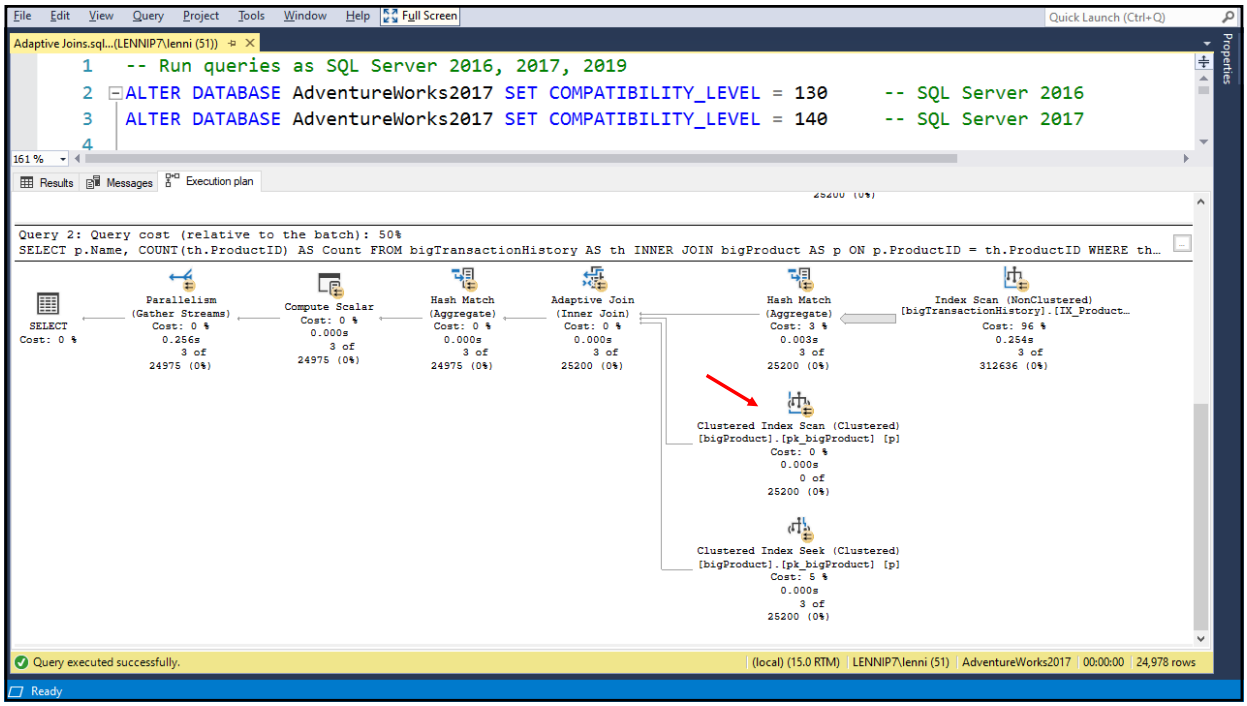

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.280s 24975 of 24975 (100%)
Compute Scalar Cost: 0 % 0.000s 24975 of 24975 (100%)
Hash Match (Aggregate) Cost: 0 % 0.028s 24975 of 24975 (100%)
Adaptive Join (Inner Join) Cost: 0 % 0.000s 25200 of 25200 (100%)
Hash Match (Aggregate) Cost: 3 % 0.005s 25200 of 25200 (100%)
Clustered Index Scan [bigProduct].[pk_bigP… Cost: 0 % 0.002s 25200 of 25200 (100%)
Clustered Index Seek [bigProduct].[pk_bigP… Cost: 5 % 0.000s 0 of 25200 (0%)
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p
Query executed successfully.
(local) (15.0 RTM) LENN
Ready
Properties
Clustered Index Seek (Clustered)
Scanning a particular range of rows from a clustered index.
Physical Operation Clustered Index Seek
Logical Operation Clustered Index Seek
Estimated Execution Mode Row
Storage RowStore
Actual Number of Rows 0
Actual Number of Batches 0
Estimated I/O Cost 0.003125
Estimated Operator Cost 5.86535 (5%)
Estimated CPU Cost 0.0001581
Estimated Subtree Cost 5.86535
Estimated Number of Executions 25199.9
Number of Executions 0
Estimated Number of Rows to be Read 1
Estimated Number of Rows 1
Estimated Row Size 57 B
Actual Rebinds 0
Actual Rewinds 0
Ordered True
Node ID 11
Object
[AdventureWorks2017].[dbo].[bigProduct].[pk_bigProduct] [p]
Output List
[AdventureWorks2017].[dbo].[bigProduct].Name
Seek Predicates
Seek Keys[1]: Prefix: [AdventureWorks2017].[dbo].[bigProduct].ProductID = Scalar Operator([AdventureWorks2017].[dbo].[bigTransactionHistory].[ProductID] as [th].[ProductID])

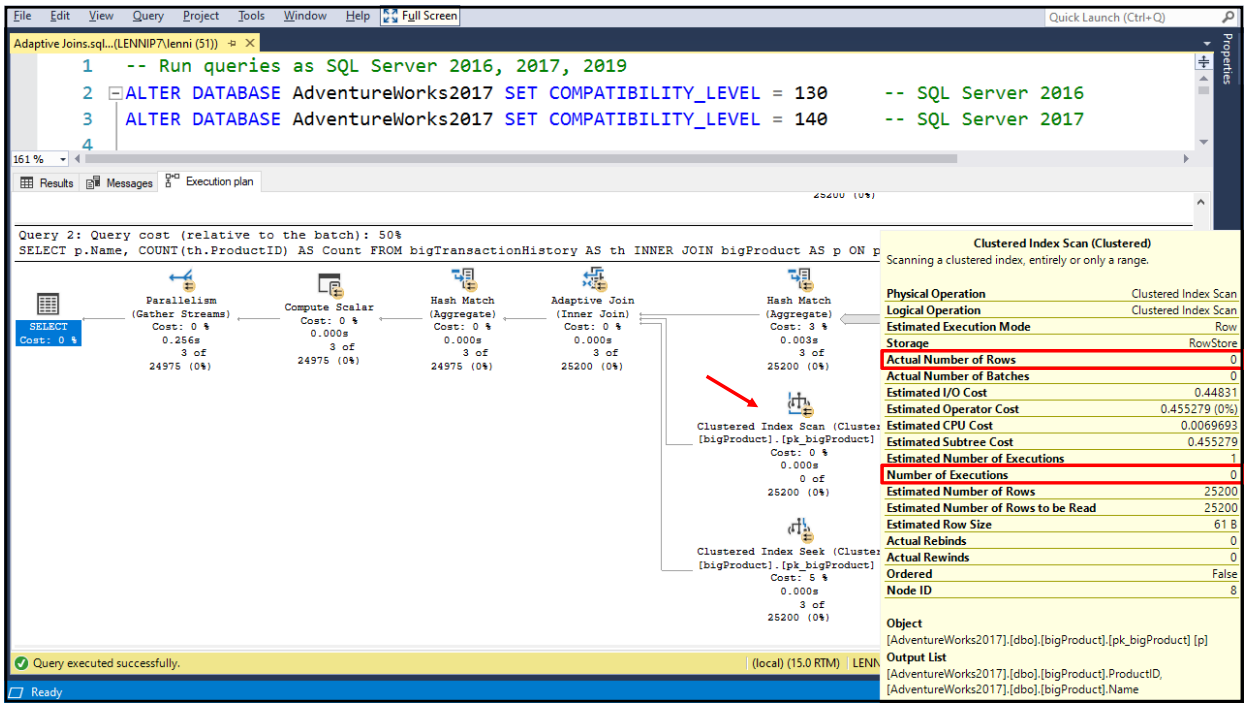

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th…
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.280s 24975 of 24975 (100%)
Compute Scalar Cost: 0 % 0.000s 24975 of 24975 (100%)
Hash Match (Aggregate) Cost: 0 % 0.028s 24975 of 24975 (100%)
Adaptive Join (Inner Join) Cost: 0 % 0.000s 25200 of 25200 (100%)
Hash Match (Aggregate) Cost: 3 % 0.005s 25200 of 25200 (100%)
Index Scan (NonClustered) [bigTransactionHistory].[IX_Product… Cost: 96 % 0.262s 313138 of 312636 (100%)
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 0 % 0.002s 25200 of 25200 (100%)
Clustered Index Seek (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 5 % 0.000s 0 of 25200 (0%)
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th…
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready
Properties

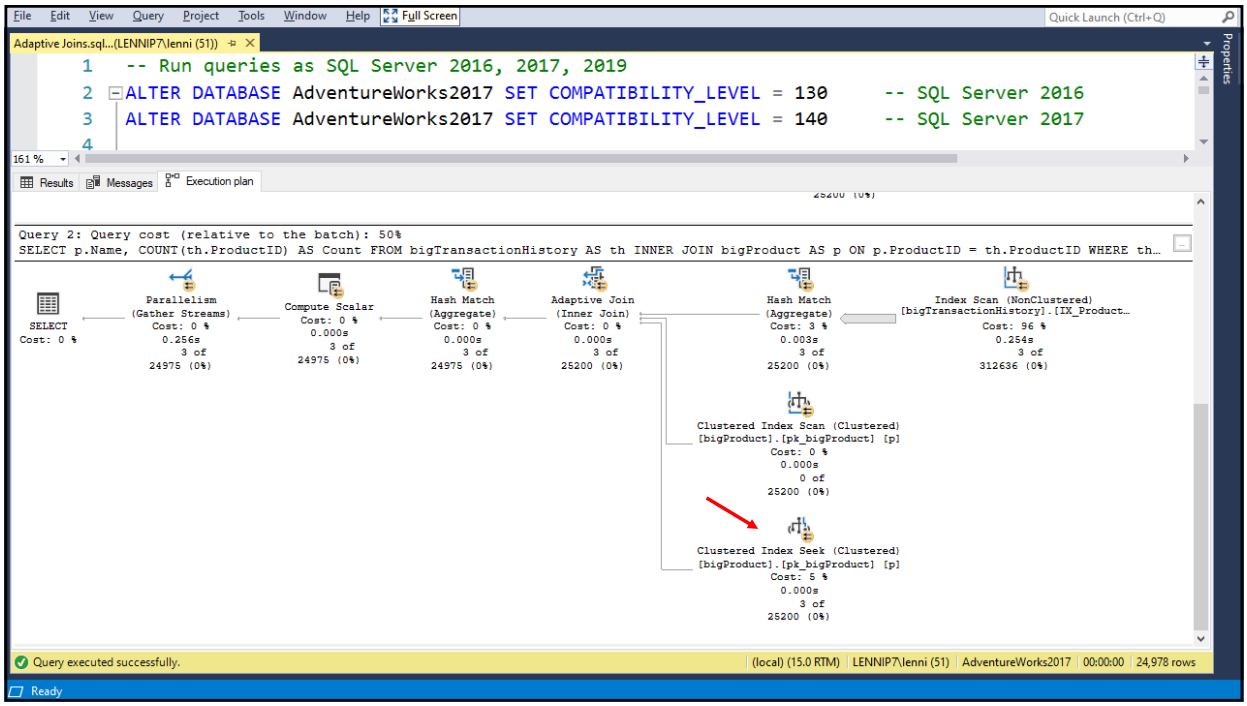

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th…
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.256s 3 of 24975 (0%)
Compute Scalar Cost: 0 % 0.000s 3 of 24975 (0%)
Hash Match (Aggregate) Cost: 0 % 0.000s 3 of 24975 (0%)
Adaptive Join (Inner Join) Cost: 0 % 0.000s 3 of 25200 (0%)
Hash Match (Aggregate) Cost: 3 % 0.003s 3 of 25200 (0%)
Index Scan (NonClustered) [bigTransactionHistory].[IX_Product… Cost: 96 % 0.254s 3 of 312636 (0%)
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 0 % 0.000s 0 of 25200 (0%)
Clustered Index Seek (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 5 % 0.000s 3 of 25200 (0%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready
Properties

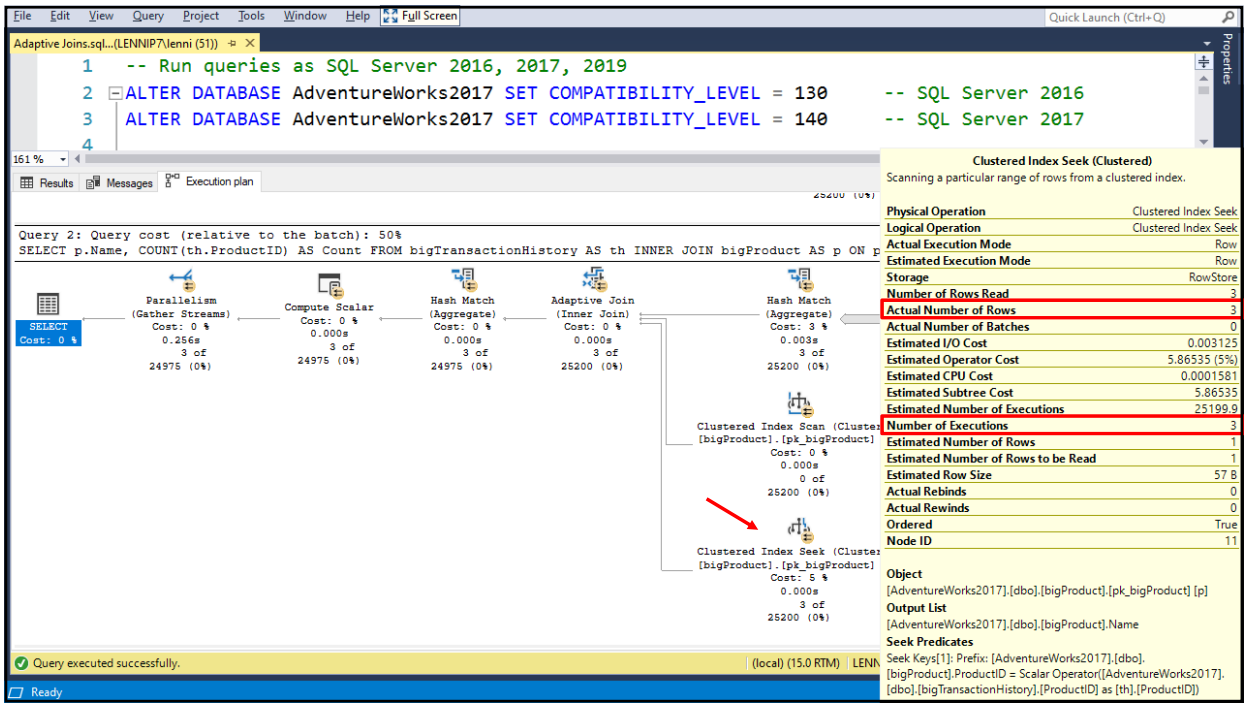

-- Run queries as SQL Server 2016, 2017, 2019
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
Adaptive Join
Chooses dynamically between hash join and nested loops.
Physical Operation: Adaptive Join
Logical Operation: Inner Join
Actual Join Type: NestedLoops
Actual Execution Mode: Batch
Estimated Join Type: HashMatch
Is Adaptive: True
Adaptive Threshold Rows: 1969.44
Actual Number of Rows: 3
Actual Number of Batches: 3
Estimated Operator Cost: 0 (0%)
Estimated I/O Cost: 0
Estimated CPU Cost: 0.000126
Estimated Subtree Cost: 109.932
Estimated Number of Executions: 1
Number of Executions: 8
Estimated Number of Rows: 25199.9
Estimated Row Size: 65 B
Actual Rebinds: 0
Actual Rewinds: 0
Node ID: 4
Output List
[AdventureWorks2017].[dbo].[bigProduct].Name, partialagg1003
Hash Keys Probe
[AdventureWorks2017].[dbo].[bigProduct].ProductID
Outer References
[AdventureWorks2017].[dbo].[bigTransactionHistory].ProductID
Query executed successfully.
Ready

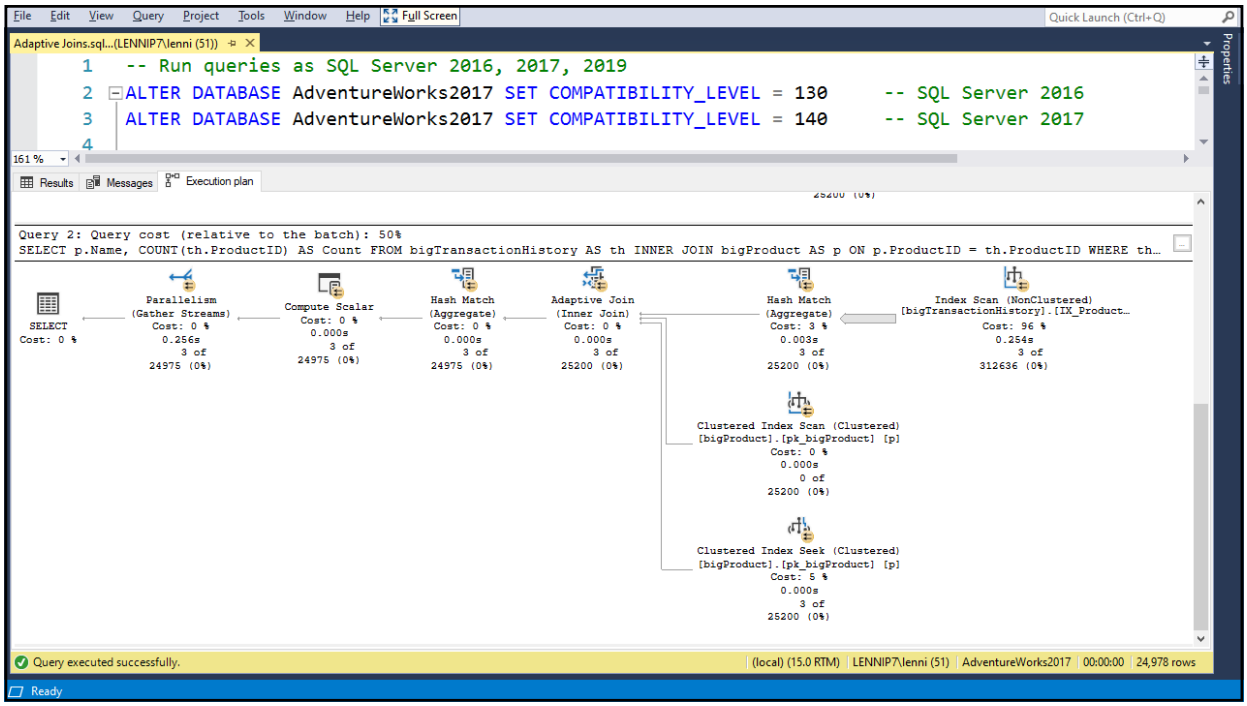

-- Run queries as SQL Server 2016, 2017, 2019
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th...
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p]
Clustered Index Seek (Clustered) [bigProduct].[pk_bigProduct] [p]
Query executed successfully.
Ready


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th...
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.256s 3 of 24975 (0%)
Compute Scalar Cost: 0 % 0.000s 3 of 24975 (0%)
Hash Match (Aggregate) Cost: 0 % 0.000s 3 of 24975 (0%)
Adaptive Join (Inner Join) Cost: 0 % 0.000s 3 of 25200 (0%)
Hash Match (Aggregate) Cost: 3 % 0.003s 3 of 25200 (0%)
Index Scan (NonClustered) [bigTransactionHistory].[IX_Product... Cost: 96 % 0.254s 3 of 312636 (0%)
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 0 % 0.000s 0 of 25200 (0%)
Clustered Index Seek (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 5 % 0.000s 3 of 25200 (0%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready
Properties


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.256s 3 of 24975 (0%)
Compute Scalar Cost: 0 % 0.000s 3 of 24975 (0%)
Hash Match (Aggregate) Cost: 0 % 0.000s 3 of 24975 (0%)
Adaptive Join (Inner Join) Cost: 0 % 0.000s 3 of 25200 (0%)
Hash Match (Aggregate) Cost: 3 % 0.003s 3 of 25200 (0%)
Clustered Index Scan (Cluste [bigProduct].[pk_bigProduct] Cost: 0 % 0.000s 0 of 25200 (0%)
Clustered Index Seek (Cluste [bigProduct].[pk_bigProduct] Cost: 5 % 0.000s 3 of 25200 (0%)
Clustered Index Scan (Clustered)
Scanning a clustered index, entirely or only a range.
Physical Operation Clustered Index Scan
Logical Operation Clustered Index Scan
Estimated Execution Mode Row
Storage RowStore
Actual Number of Rows 0
Actual Number of Batches 0
Estimated I/O Cost 0.44831
Estimated Operator Cost 0.455279 (0%)
Estimated CPU Cost 0.0069693
Estimated Subtree Cost 0.455279
Estimated Number of Executions 1
Number of Executions 0
Estimated Number of Rows 25200
Estimated Number of Rows to be Read 25200
Estimated Row Size 61 B
Actual Rebinds 0
Actual Rewinds 0
Ordered False
Node ID 8
Object
[AdventureWorks2017].[dbo].[bigProduct].[pk_bigProduct] [p]
Output List
[AdventureWorks2017].[dbo].[bigProduct].ProductID,
[AdventureWorks2017].[dbo].[bigProduct].Name
Query executed successfully.
(local) (15.0 RTM) LENN
Ready
Properties


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID WHERE th...
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.256s 3 of 24975 (0%)
Compute Scalar Cost: 0 % 0.000s 3 of 24975 (0%)
Hash Match (Aggregate) Cost: 0 % 0.000s 3 of 24975 (0%)
Adaptive Join (Inner Join) Cost: 0 % 0.000s 3 of 25200 (0%)
Hash Match (Aggregate) Cost: 3 % 0.003s 3 of 25200 (0%)
Index Scan (NonClustered) [bigTransactionHistory].[IX_Product... Cost: 96 % 0.254s 3 of 312636 (0%)
Clustered Index Scan (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 0 % 0.000s 0 of 25200 (0%)
Clustered Index Seek (Clustered) [bigProduct].[pk_bigProduct] [p] Cost: 5 % 0.000s 3 of 25200 (0%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) AdventureWorks2017 00:00:00 24,978 rows
Ready
Properties


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Adaptive Joins.sql...(LENNIP7\lenni (51))
1 -- Run queries as SQL Server 2016, 2017, 2019
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
3 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
4
161 %
Results Messages Execution plan
Query 2: Query cost (relative to the batch): 50%
SELECT p.Name, COUNT(th.ProductID) AS Count FROM bigTransactionHistory AS th INNER JOIN bigProduct AS p ON p
SELECT Cost: 0 %
Parallelism (Gather Streams) Cost: 0 % 0.256s 3 of 24975 (0%)
Compute Scalar Cost: 0 % 0.000s 3 of 24975 (0%)
Hash Match (Aggregate) Cost: 0 % 0.000s 3 of 24975 (0%)
Adaptive Join (Inner Join) Cost: 0 % 0.000s 3 of 25200 (0%)
Hash Match (Aggregate) Cost: 3 % 0.003s 3 of 25200 (0%)
Clustered Index Scan (Cluste [bigProduct].[pk_bigProduct] Cost: 0 % 0.000s 0 of 25200 (0%)
Clustered Index Seek (Cluste [bigProduct].[pk_bigProduct] Cost: 5 % 0.000s 3 of 25200 (0%)
Clustered Index Seek (Clustered)
Scanning a particular range of rows from a clustered index.
Physical Operation Clustered Index Seek
Logical Operation Clustered Index Seek
Actual Execution Mode Row
Estimated Execution Mode Row
Storage RowStore
Number of Rows Read 3
Actual Number of Rows 3
Actual Number of Batches 0
Estimated I/O Cost 0.003125
Estimated Operator Cost 5.86535 (5%)
Estimated CPU Cost 0.0001581
Estimated Subtree Cost 5.86535
Estimated Number of Executions 25199.9
Number of Executions 3
Estimated Number of Rows 1
Estimated Number of Rows to be Read 1
Estimated Row Size 57 B
Actual Rebinds 0
Actual Rewinds 0
Ordered True
Node ID 11
Object
[AdventureWorks2017].[dbo].[bigProduct].[pk_bigProduct] [p]
Output List
[AdventureWorks2017].[dbo].[bigProduct].Name
Seek Predicates
Seek Keys[1]: Prefix: [AdventureWorks2017].[dbo].[bigProduct].ProductID = Scalar Operator([AdventureWorks2017].[dbo].[bigTransactionHistory].[ProductID] as [th].[ProductID])
Query executed successfully.
(local) (15.0 RTM) LENN
Ready
Properties

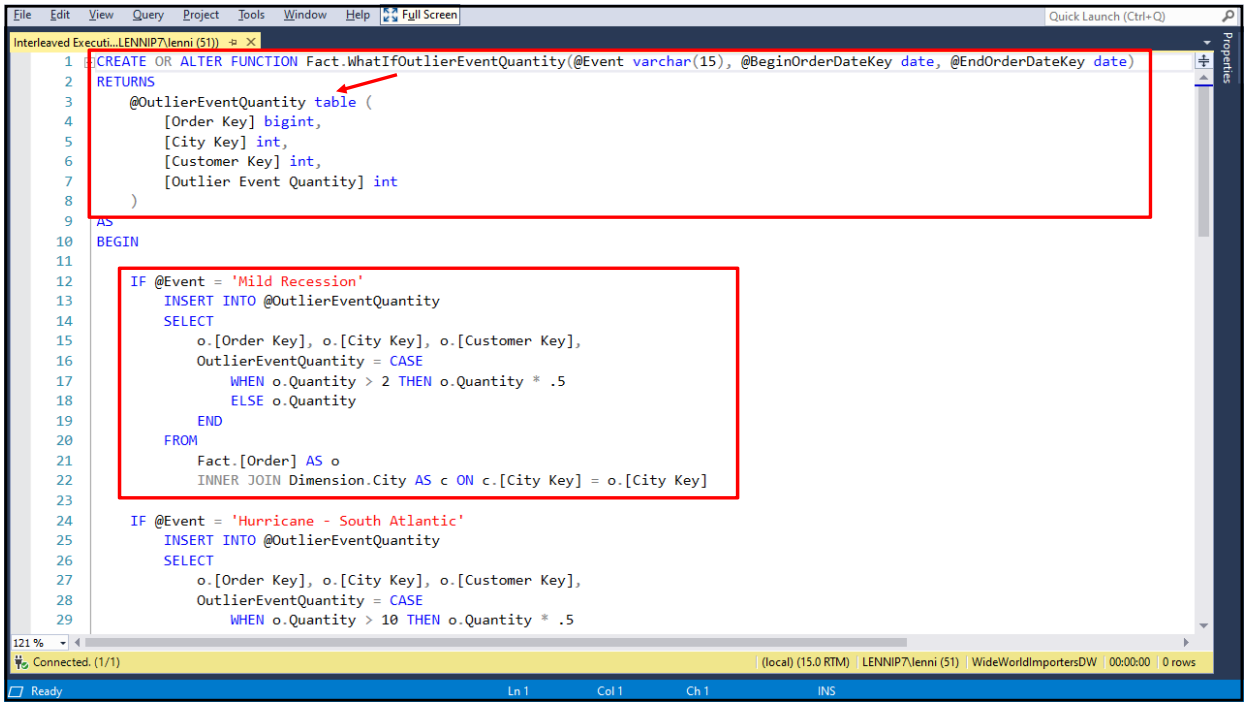


Interleaved Execution

- Optimizes queries that use MSTVFs
  - Multi-statement table-valued functions
- SQL Server 2012 and earlier
  - MSTVFs fixed cardinality = 1
- SQL Server 2014 and 2016
  - MSTVFs fixed cardinality = 100
- SQL Server 2017 and 2019
  - Pauses optimization when the MSTVF operation is encountered
  - Executes the MSTVF
  - Captures accurate cardinality estimates
  - Resumes optimization for downstream operations

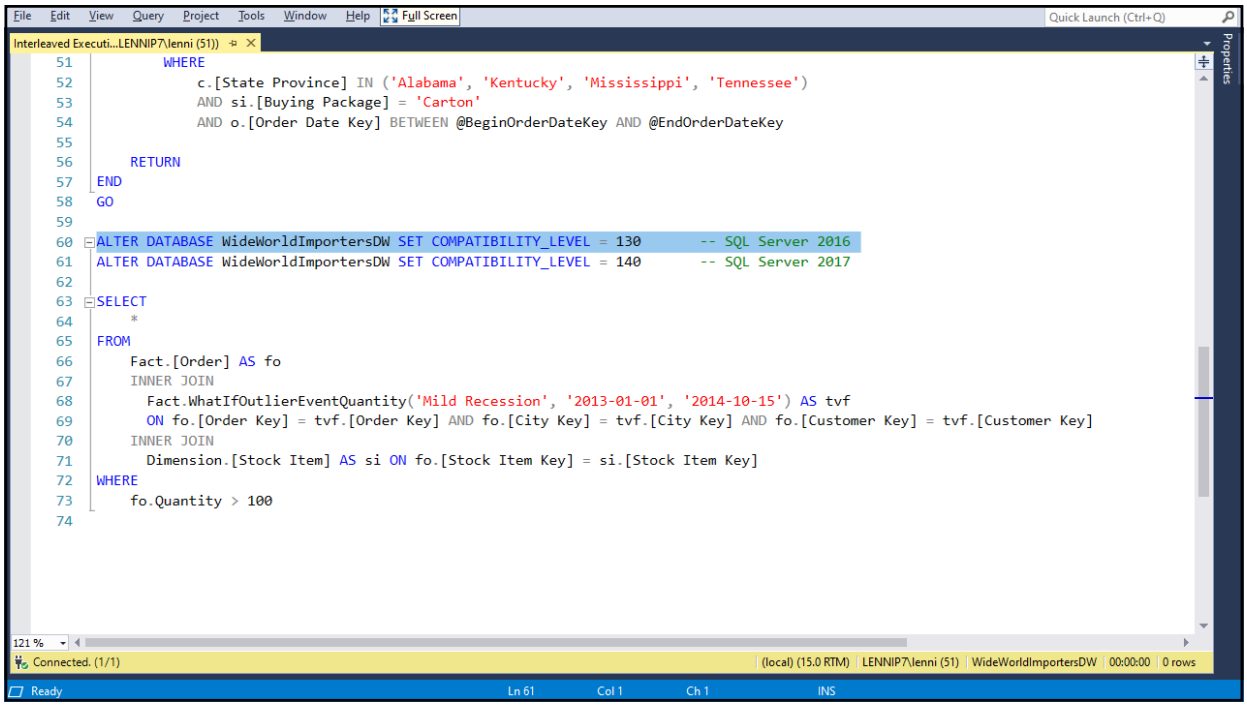
Interleaved Execution

demo

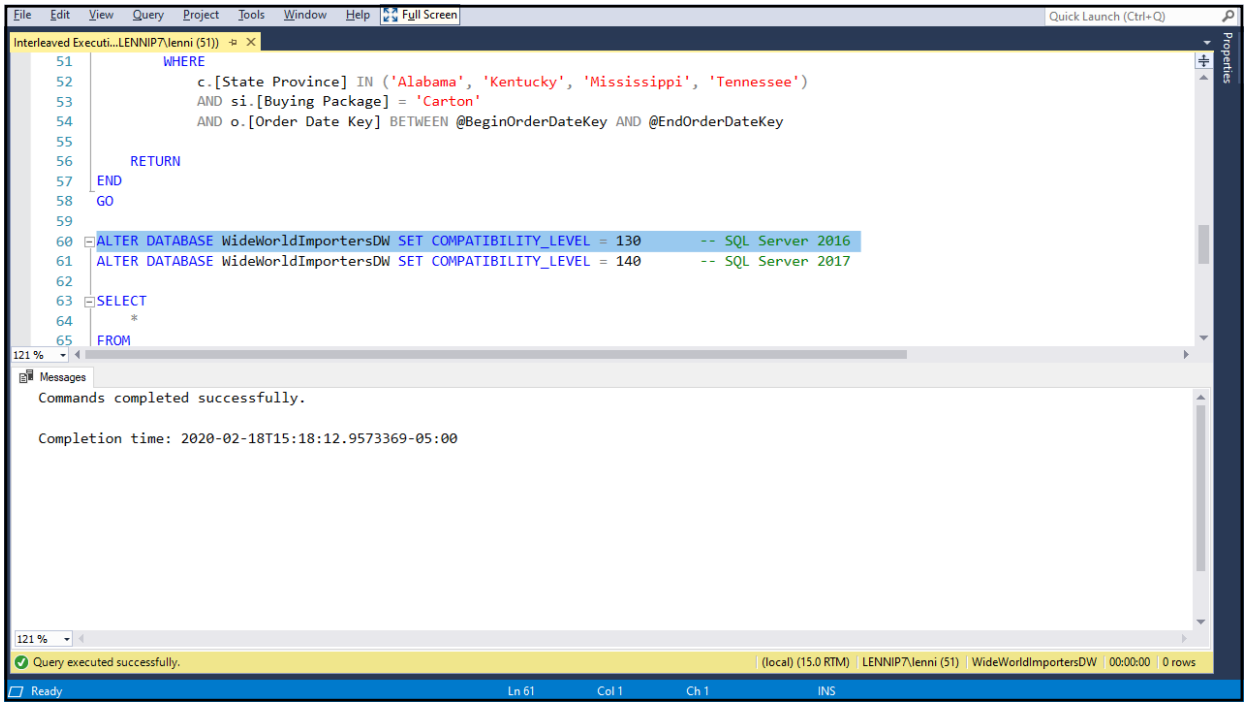


```
IF @Event = 'Hurricane - South Atlantic'
    INSERT INTO @OutlierEventQuantity
    SELECT
        o.[Order Key], o.[City Key], o.[Customer Key],
        OutlierEventQuantity = CASE
            WHEN o.Quantity > 10 THEN o.Quantity * .5
            ELSE o.Quantity
        END
    FROM
        Fact.[Order] AS o
        INNER JOIN Dimension.City AS c ON c.[City Key] = o.[City Key]
    WHERE
        c.[State Province] IN ('Florida', 'Georgia', 'Maryland', 'North Carolina', 'South Carolina', 'Virginia', 'West Virginia
        AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey

IF @Event = 'Hurricane - East South Central'
    INSERT INTO @OutlierEventQuantity
    SELECT
        o.[Order Key], o.[City Key], o.[Customer Key],
        CASE
            WHEN o.Quantity > 50 THEN o.Quantity * .5
            ELSE o.Quantity
        END
    FROM
        Fact.[Order] AS o
        INNER JOIN Dimension.City AS c ON c.[City Key] = o.[City Key]
        INNER JOIN Dimension.[Stock Item] AS si ON si.[Stock Item Key] = o.[Stock Item Key]
    WHERE
        c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
```

```
        WHERE
            c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
            AND si.[Buying Package] = 'Carton'
            AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey

    RETURN
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 130       -- SQL Server 2016
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017

SELECT
    *
FROM
    Fact.[Order] AS fo
    INNER JOIN
      Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf
      ON fo.[Order Key] = tvf.[Order Key] AND fo.[City Key] = tvf.[City Key] AND fo.[Customer Key] = tvf.[Customer Key]
    INNER JOIN
      Dimension.[Stock Item] AS si ON fo.[Stock Item Key] = si.[Stock Item Key]
WHERE
    fo.Quantity > 100
```

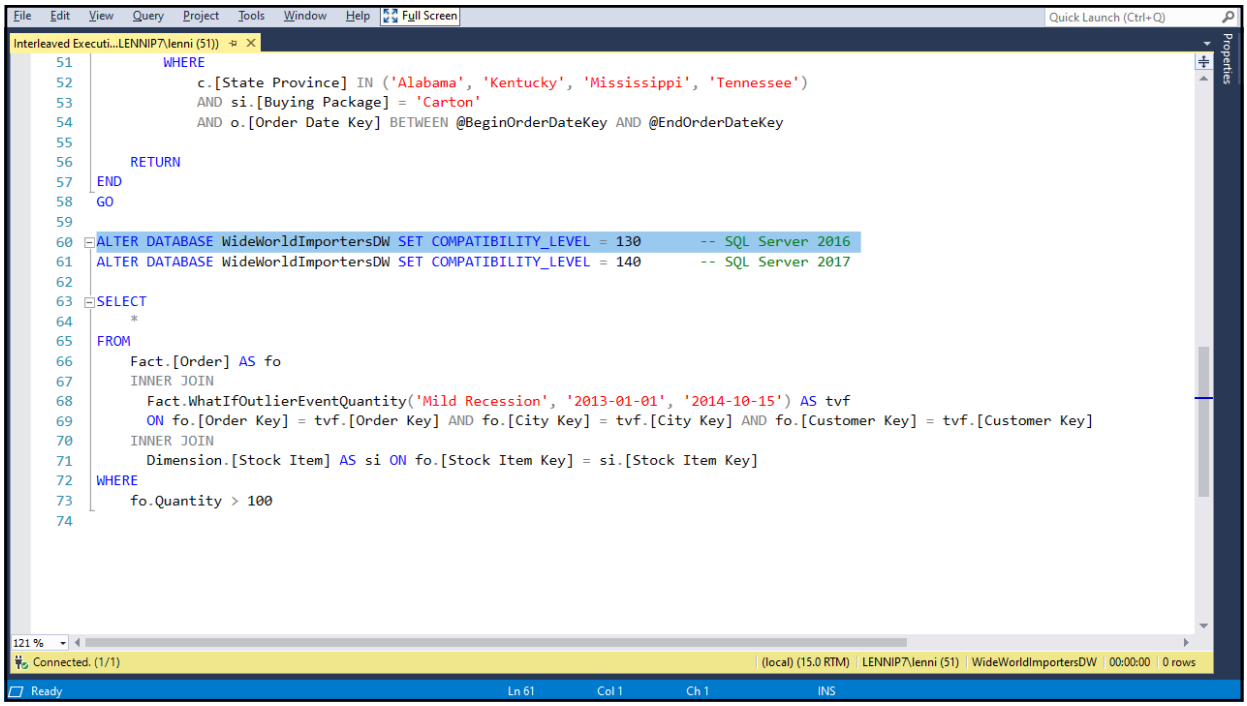


```
        WHERE
            c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
            AND si.[Buying Package] = 'Carton'
            AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey

    RETURN
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 130       -- SQL Server 2016
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017

SELECT
    *
FROM
    Fact.[Order] AS fo
    INNER JOIN
      Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf
      ON fo.[Order Key] = tvf.[Order Key] AND fo.[City Key] = tvf.[City Key] AND fo.[Customer Key] = tvf.[Customer Key]
    INNER JOIN
      Dimension.[Stock Item] AS si ON fo.[Stock Item Key] = si.[Stock Item Key]
WHERE
    fo.Quantity > 100
```

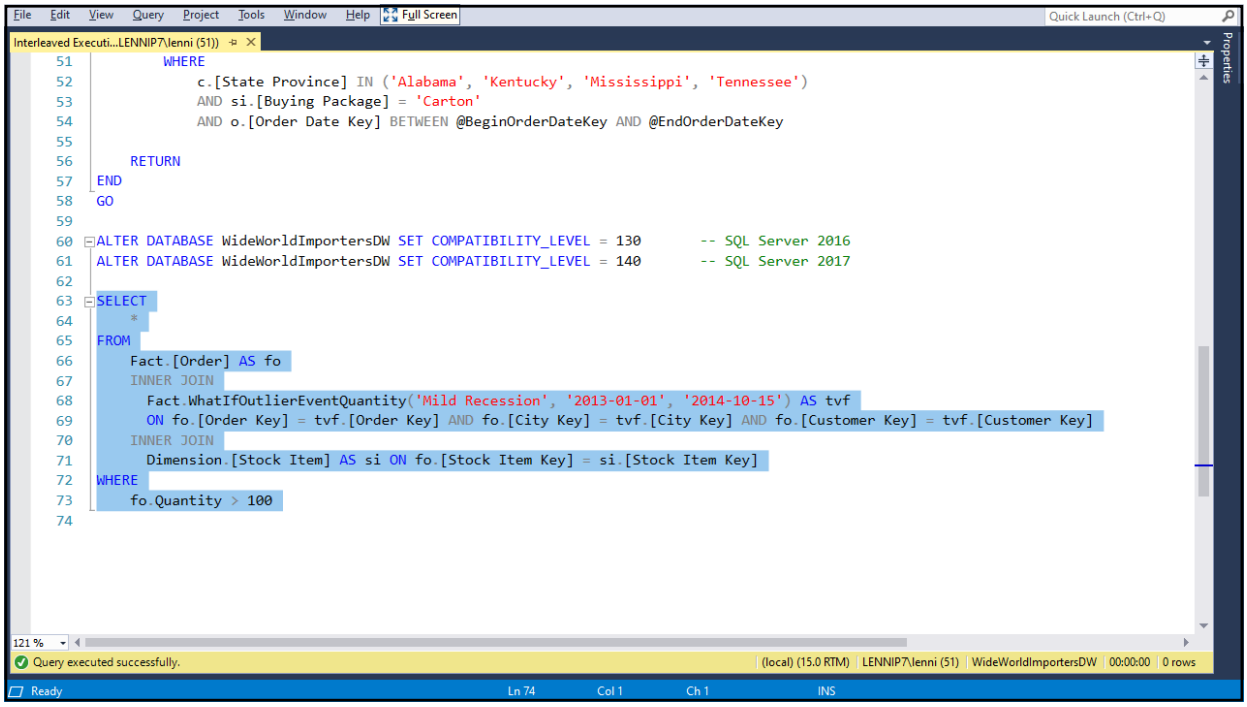


```
        WHERE
            c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
            AND si.[Buying Package] = 'Carton'
            AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey

    RETURN
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 130       -- SQL Server 2016
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017

SELECT
    *
FROM
```

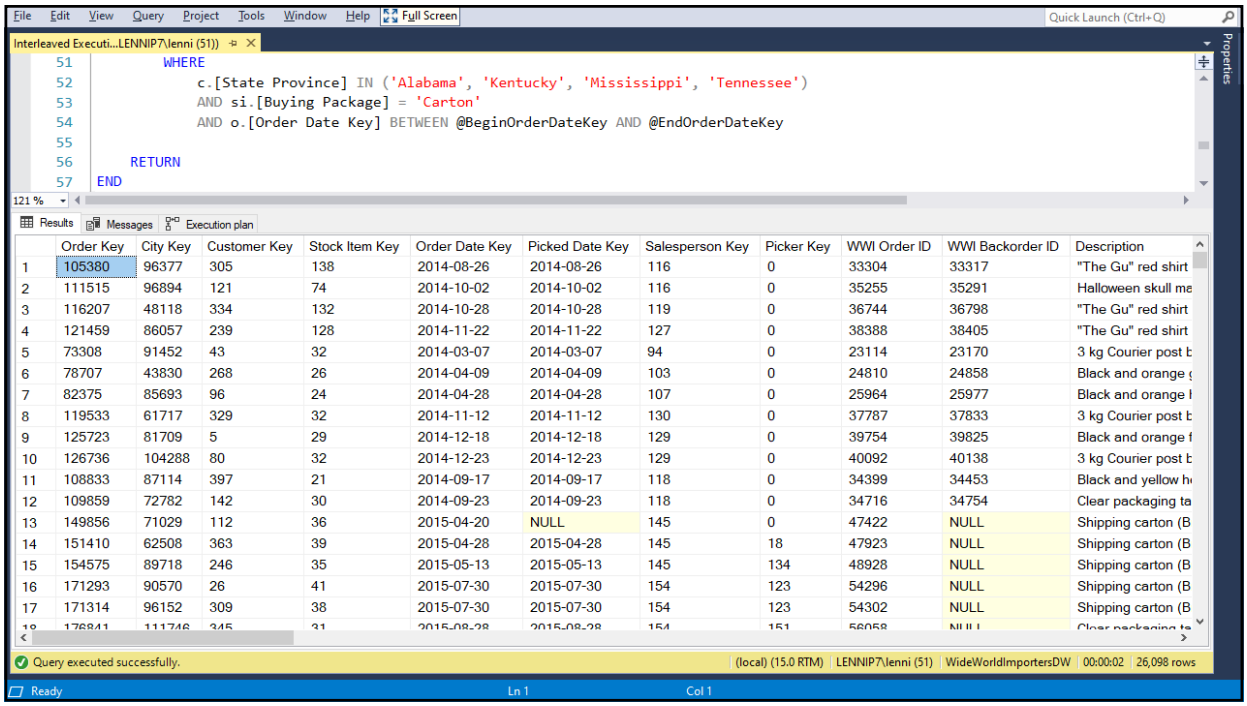
```
Commands completed successfully.

Completion time: 2020-02-18T15:18:12.9573369-05:00
```

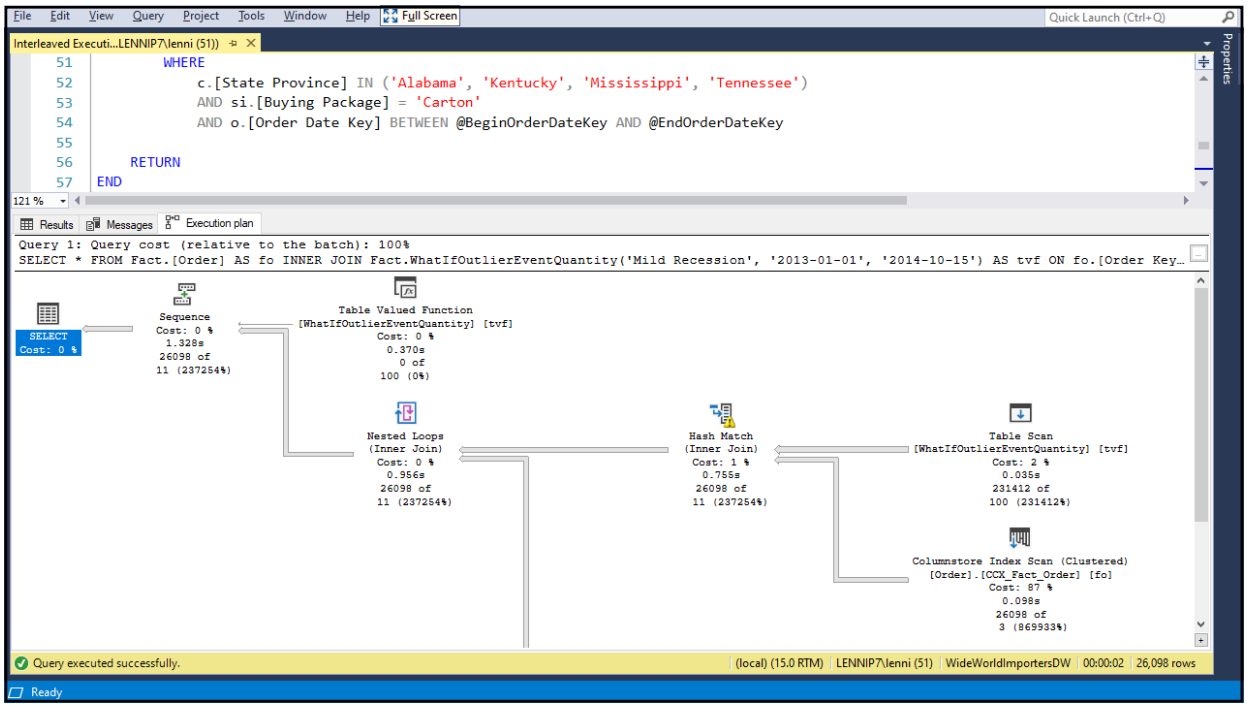


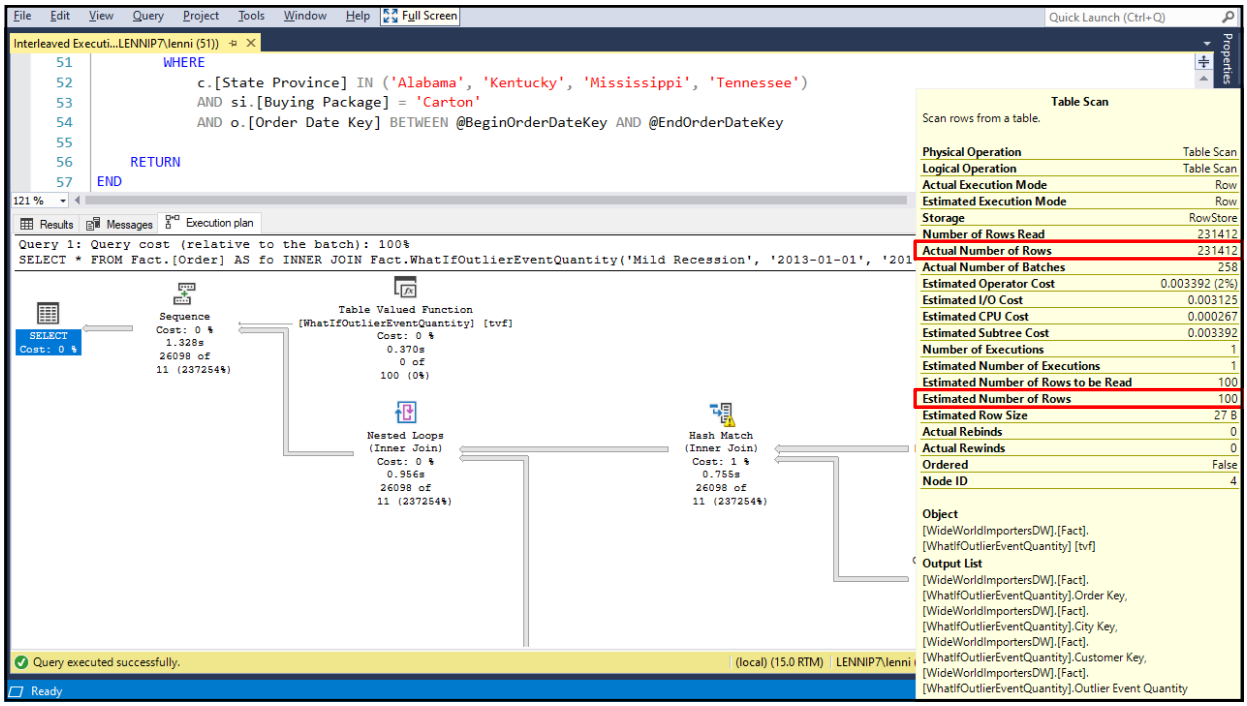
```
        WHERE
            c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
            AND si.[Buying Package] = 'Carton'
            AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey

    RETURN
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 130       -- SQL Server 2016
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017

SELECT
    *
FROM
    Fact.[Order] AS fo
    INNER JOIN
      Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf
      ON fo.[Order Key] = tvf.[Order Key] AND fo.[City Key] = tvf.[City Key] AND fo.[Customer Key] = tvf.[Customer Key]
    INNER JOIN
      Dimension.[Stock Item] AS si ON fo.[Stock Item Key] = si.[Stock Item Key]
WHERE
    fo.Quantity > 100
```

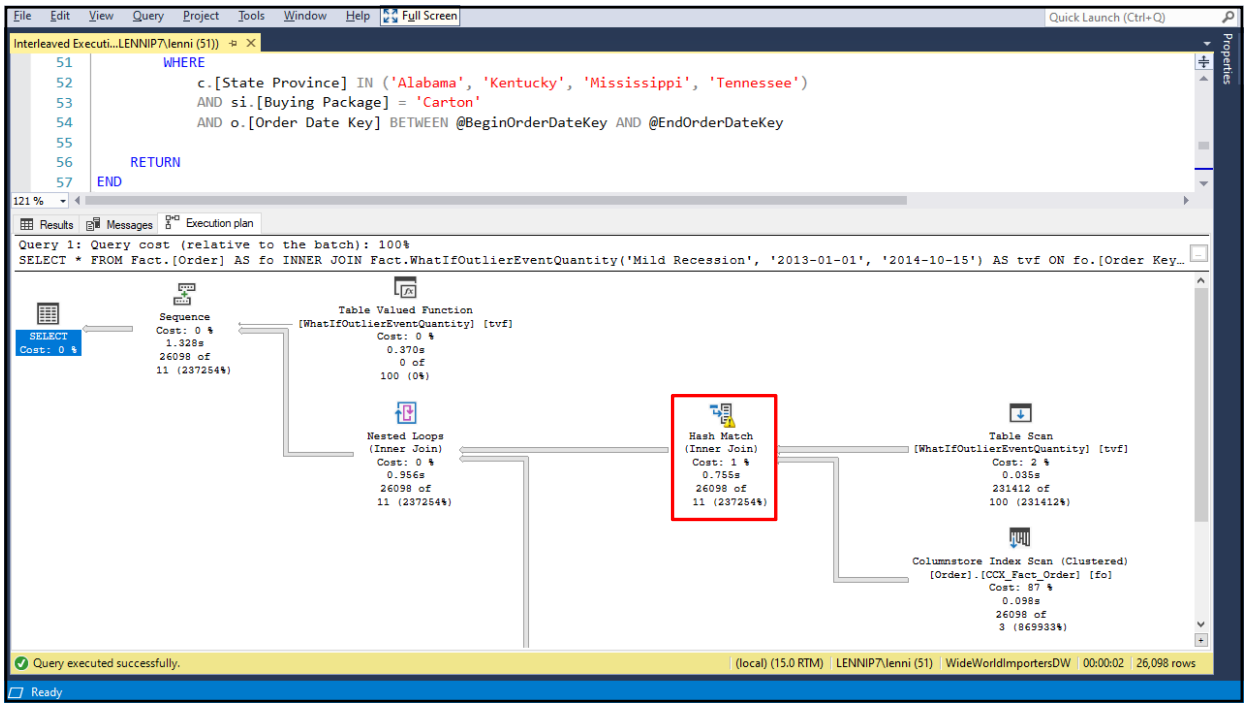


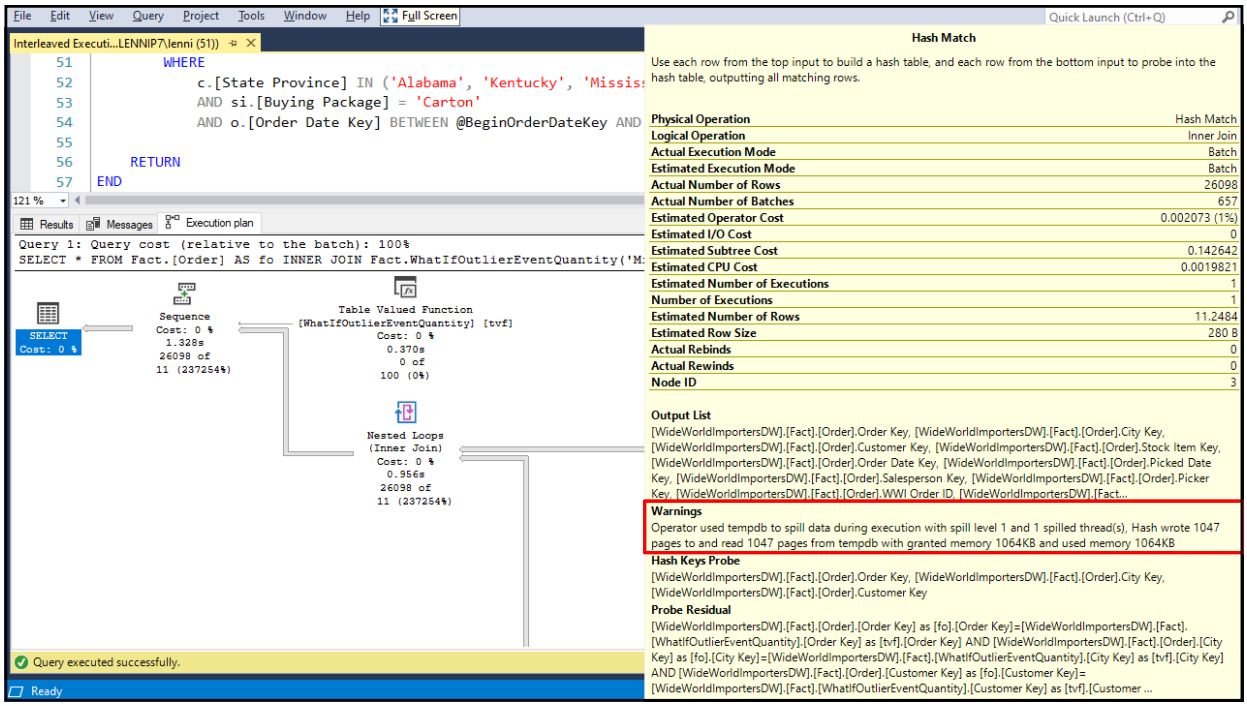
```
        WHERE
            c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
            AND si.[Buying Package] = 'Carton'
            AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey

    RETURN
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 130       -- SQL Server 2016
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017

SELECT
    *
FROM
    Fact.[Order] AS fo
    INNER JOIN
      Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf
      ON fo.[Order Key] = tvf.[Order Key] AND fo.[City Key] = tvf.[City Key] AND fo.[Customer Key] = tvf.[Customer Key]
    INNER JOIN
      Dimension.[Stock Item] AS si ON fo.[Stock Item Key] = si.[Stock Item Key]
WHERE
    fo.Quantity > 100
```

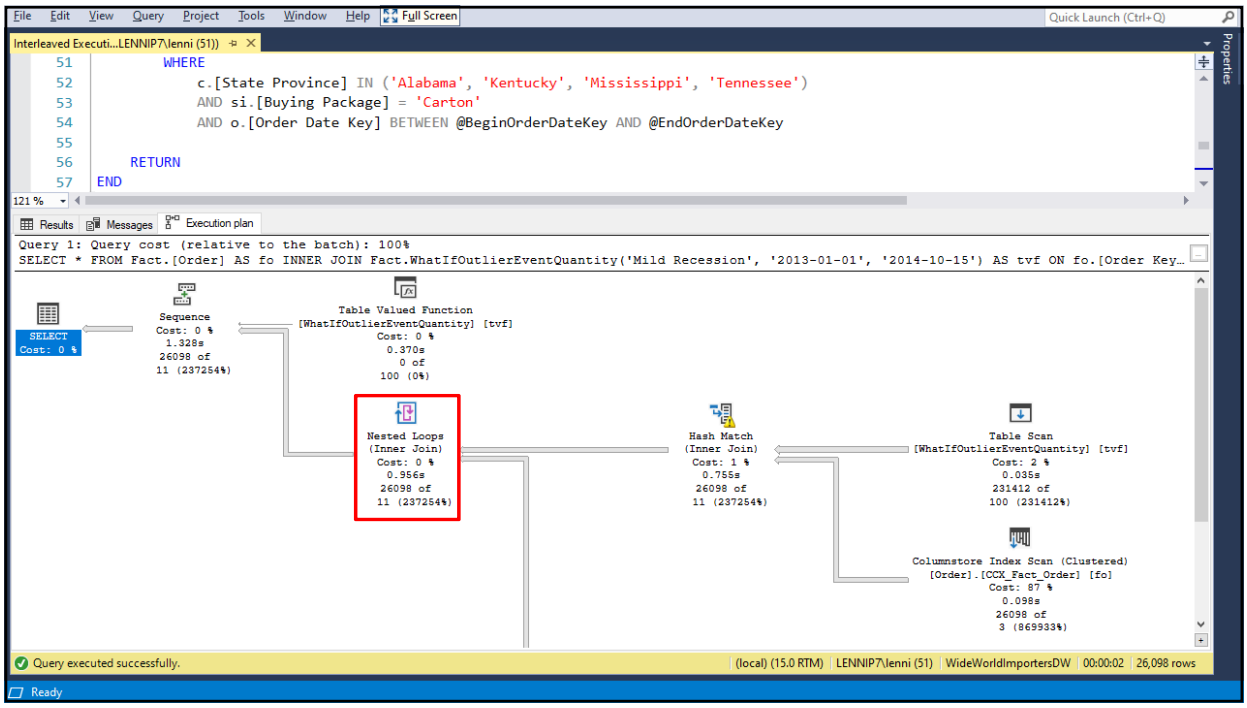

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Interleaved Executi...LENNIP7\lenni (51))
Properties
51 WHERE
52 c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53 AND si.[Buying Package] = 'Carton'
54 AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
55
56 RETURN
57 END
121 %
Results Messages Execution plan
Order Key | City Key | Customer Key | Stock Item Key | Order Date Key | Picked Date Key | Salesperson Key | Picker Key | WWI Order ID | WWI Backorder ID | Description
1 105380 96377 305 138 2014-08-26 2014-08-26 116 0 33304 33317 "The Gu" red shirt
2 111515 96894 121 74 2014-10-02 2014-10-02 116 0 35255 35291 Halloween skull ma
3 116207 48118 334 132 2014-10-28 2014-10-28 119 0 36744 36798 "The Gu" red shirt
4 121459 86057 239 128 2014-11-22 2014-11-22 127 0 38388 38405 "The Gu" red shirt
5 73308 91452 43 32 2014-03-07 2014-03-07 94 0 23114 23170 3 kg Courier post b
6 78707 43830 268 26 2014-04-09 2014-04-09 103 0 24810 24858 Black and orange
7 82375 85693 96 24 2014-04-28 2014-04-28 107 0 25964 25977 Black and orange
8 119533 61717 329 32 2014-11-12 2014-11-12 130 0 37787 37833 3 kg Courier post b
9 125723 81709 5 29 2014-12-18 2014-12-18 129 0 39754 39825 Black and orange
10 126736 104288 80 32 2014-12-23 2014-12-23 129 0 40092 40138 3 kg Courier post b
11 108833 87114 397 21 2014-09-17 2014-09-17 118 0 34399 34453 Black and yellow h
12 109859 72782 142 30 2014-09-23 2014-09-23 118 0 34716 34754 Clear packaging ta
13 149856 71029 112 36 2015-04-20 NULL 145 0 47422 NULL Shipping carton (B
14 151410 62508 363 39 2015-04-28 2015-04-28 145 18 47923 NULL Shipping carton (B
15 154575 89718 246 35 2015-05-13 2015-05-13 145 134 48928 NULL Shipping carton (B
16 171293 90570 26 41 2015-07-30 2015-07-30 154 123 54296 NULL Shipping carton (B
17 171314 96152 309 38 2015-07-30 2015-07-30 154 123 54302 NULL Shipping carton (B
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImportersDW 00:00:02 26,098 rows
Ready Ln 1 Col 1

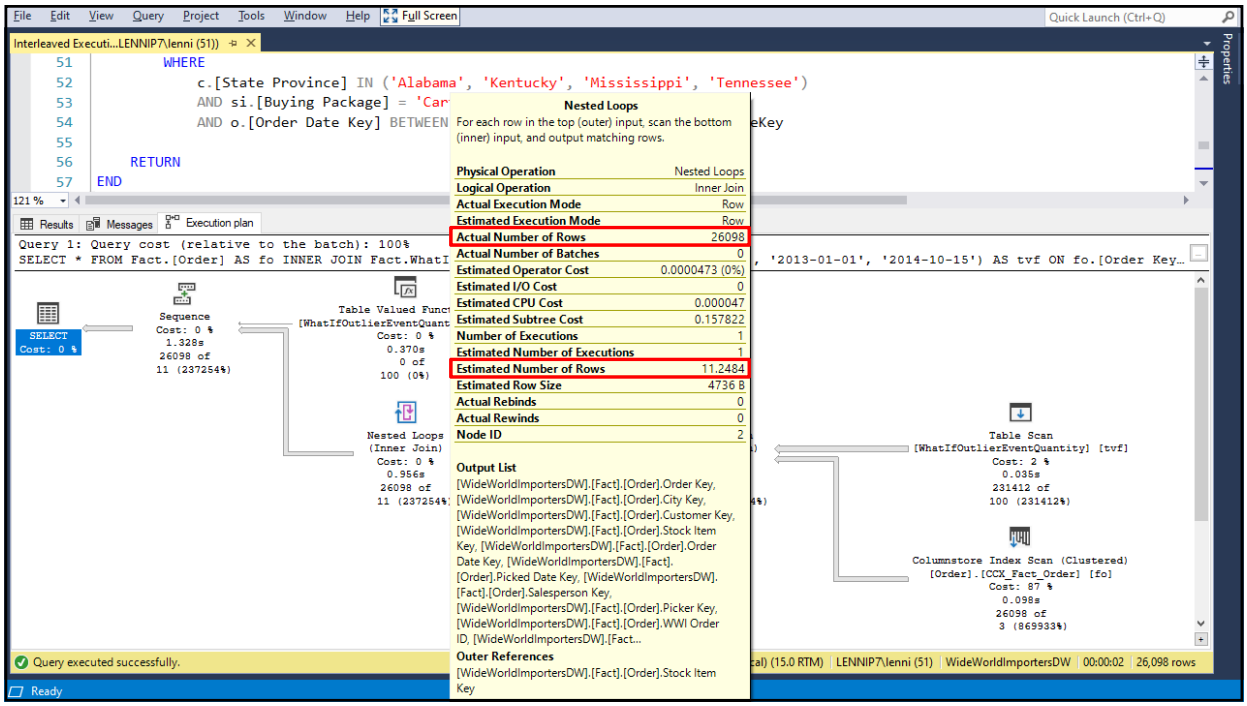

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Interleaved Executi...LENNIP7\lenni (51))
Properties
51 WHERE
52 c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53 AND si.[Buying Package] = 'Carton'
54 AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
55
56 RETURN
57 END
121 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Fact.[Order] AS fo INNER JOIN Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf ON fo.[Order Key...
SELECT Cost: 0 %
Sequence Cost: 0 % 1.328s 26098 of 11 (237254%)
Table Valued Function [WhatIfOutlierEventQuantity] [tvf] Cost: 0 % 0.370s 0 of 100 (0%)
Nested Loops (Inner Join) Cost: 0 % 0.956s 26098 of 11 (237254%)
Hash Match (Inner Join) Cost: 1 % 0.755s 26098 of 11 (237254%)
Table Scan [WhatIfOutlierEventQuantity] [tvf] Cost: 2 % 0.035s 231412 of 100 (231412%)
Columnstore Index Scan (Clustered) [Order].[CCX_Fact_Order] [fo] Cost: 87 % 0.098s 26098 of 3 (869933%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImportersDW 00:00:02 26,098 rows
Ready

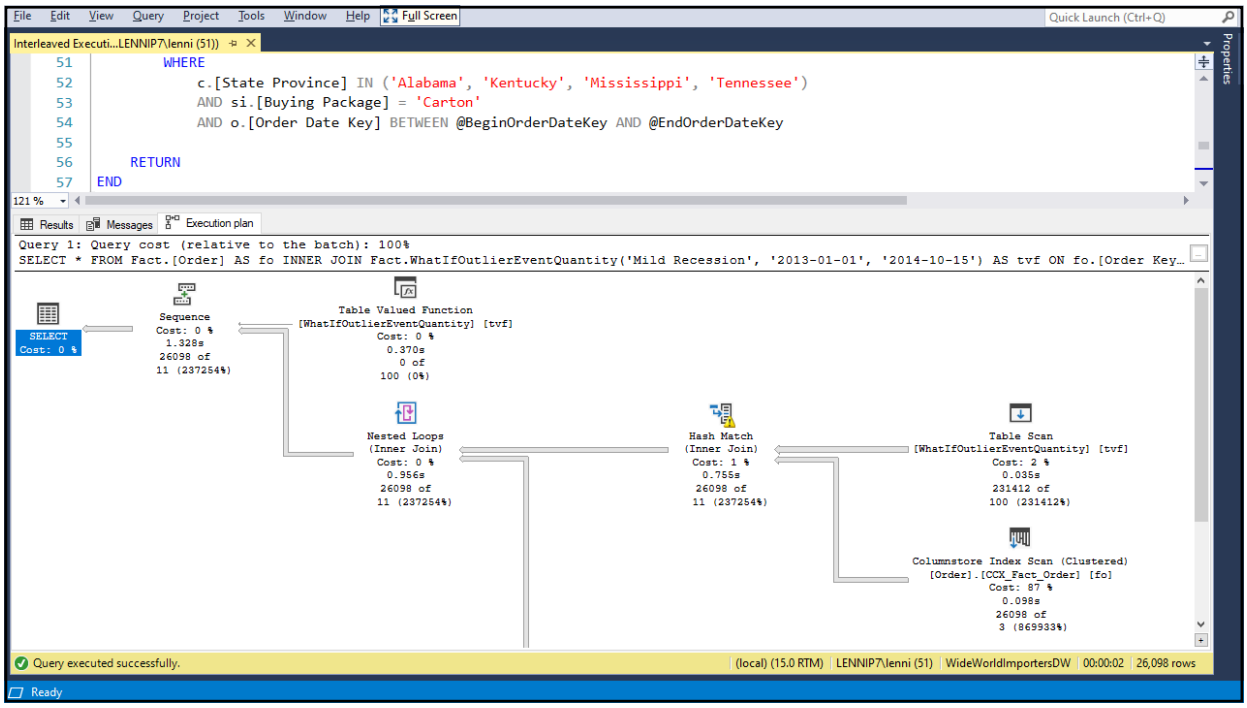

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Interleaved Executi...LENNIP7\lenni (51))
51 WHERE
52 c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53 AND si.[Buying Package] = 'Carton'
54 AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
55
56 RETURN
57 END
121 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Fact.[Order] AS fo INNER JOIN Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '201
SELECT Cost: 0 %
Sequence Cost: 0 % 1.328s 26098 of 11 (237254%)
Table Valued Function [WhatIfOutlierEventQuantity] [tvf] Cost: 0 % 0.370s 0 of 100 (0%)
Nested Loops (Inner Join) Cost: 0 % 0.956s 26098 of 11 (237254%)
Hash Match (Inner Join) Cost: 1 % 0.755s 26098 of 11 (237254%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni
Ready
Properties
Table Scan
Scan rows from a table.
Physical Operation Table Scan
Logical Operation Table Scan
Actual Execution Mode Row
Estimated Execution Mode Row
Storage RowStore
Number of Rows Read 231412
Actual Number of Rows 231412
Actual Number of Batches 258
Estimated Operator Cost 0.003392 (2%)
Estimated I/O Cost 0.003125
Estimated CPU Cost 0.000267
Estimated Subtree Cost 0.003392
Number of Executions 1
Estimated Number of Executions 1
Estimated Number of Rows to be Read 100
Estimated Number of Rows 100
Estimated Row Size 27 B
Actual Rebinds 0
Actual Rewinds 0
Ordered False
Node ID 4
Object
[WideWorldImportersDW].[Fact].
[WhatIfOutlierEventQuantity] [tvf]
Output List
[WideWorldImportersDW].[Fact].
[WhatIfOutlierEventQuantity].Order Key,
[WideWorldImportersDW].[Fact].
[WhatIfOutlierEventQuantity].City Key,
[WideWorldImportersDW].[Fact].
[WhatIfOutlierEventQuantity].Customer Key,
[WideWorldImportersDW].[Fact].
[WhatIfOutlierEventQuantity].Outlier Event Quantity

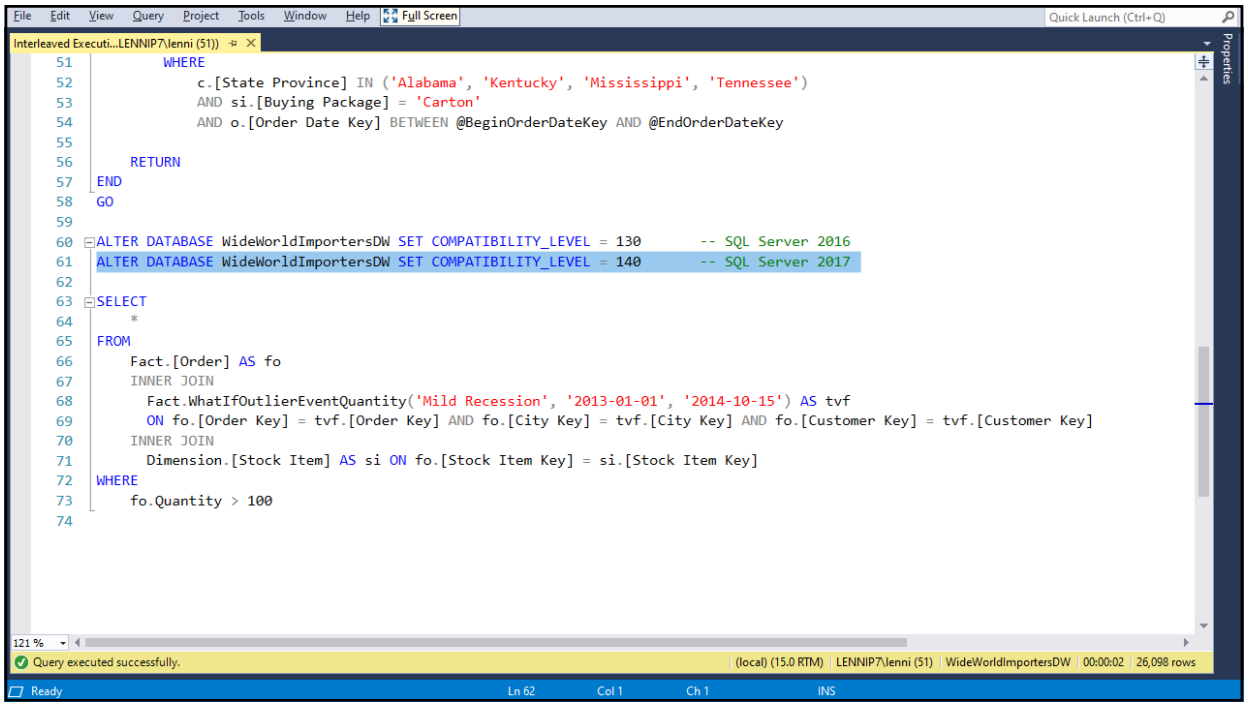

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Interleaved Executi...LENNIP7\lenni (51))
51 WHERE
52 c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53 AND si.[Buying Package] = 'Carton'
54 AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
55
56 RETURN
57 END
121 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Fact.[Order] AS fo INNER JOIN Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf ON fo.[Order Key...
SELECT Cost: 0 %
Sequence Cost: 0 % 1.328s 26098 of 11 (237254%)
Table Valued Function [WhatIfOutlierEventQuantity] [tvf] Cost: 0 % 0.370s 0 of 100 (0%)
Nested Loops (Inner Join) Cost: 0 % 0.956s 26098 of 11 (237254%)
Hash Match (Inner Join) Cost: 1 % 0.755s 26098 of 11 (237254%)
Table Scan [WhatIfOutlierEventQuantity] [tvf] Cost: 2 % 0.035s 231412 of 100 (231412%)
Columnstore Index Scan (Clustered) [Order].[CCX_Fact_Order] [fo] Cost: 87 % 0.098s 26098 of 3 (869933%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImportersDW 00:00:02 26,098 rows
Ready
Properties

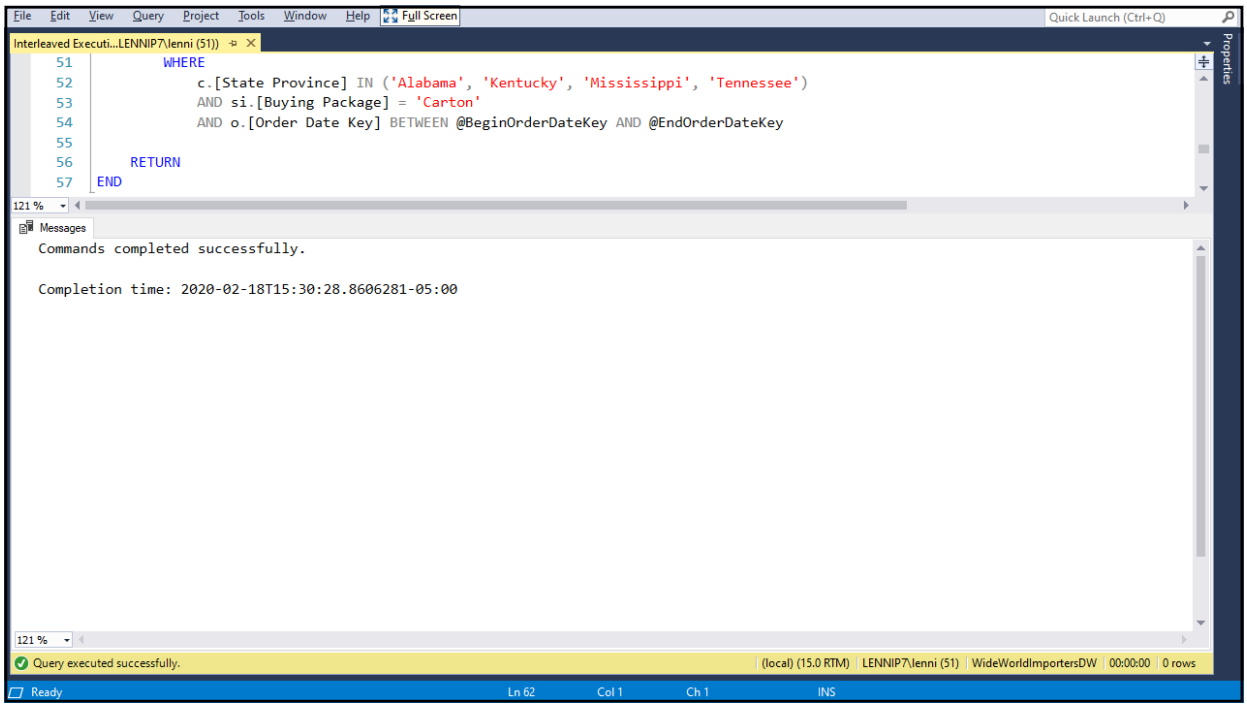

Hash Match
Use each row from the top input to build a hash table, and each row from the bottom input to probe into the hash table, outputting all matching rows.
Physical Operation: Hash Match
Logical Operation: Inner Join
Actual Execution Mode: Batch
Estimated Execution Mode: Batch
Actual Number of Rows: 26098
Actual Number of Batches: 657
Estimated Operator Cost: 0.002073 (1%)
Estimated I/O Cost: 0
Estimated Subtree Cost: 0.142642
Estimated CPU Cost: 0.0019821
Estimated Number of Executions: 1
Number of Executions: 1
Estimated Number of Rows: 11.2484
Estimated Row Size: 280 B
Actual Rebinds: 0
Actual Rewinds: 0
Node ID: 3
Output List
Warnings
Operator used tempdb to spill data during execution with spill level 1 and 1 spilled thread(s), Hash wrote 1047 pages to and read 1047 pages from tempdb with granted memory 1064KB and used memory 1064KB
Hash Keys Probe
Probe Residual
Query executed successfully.
Ready

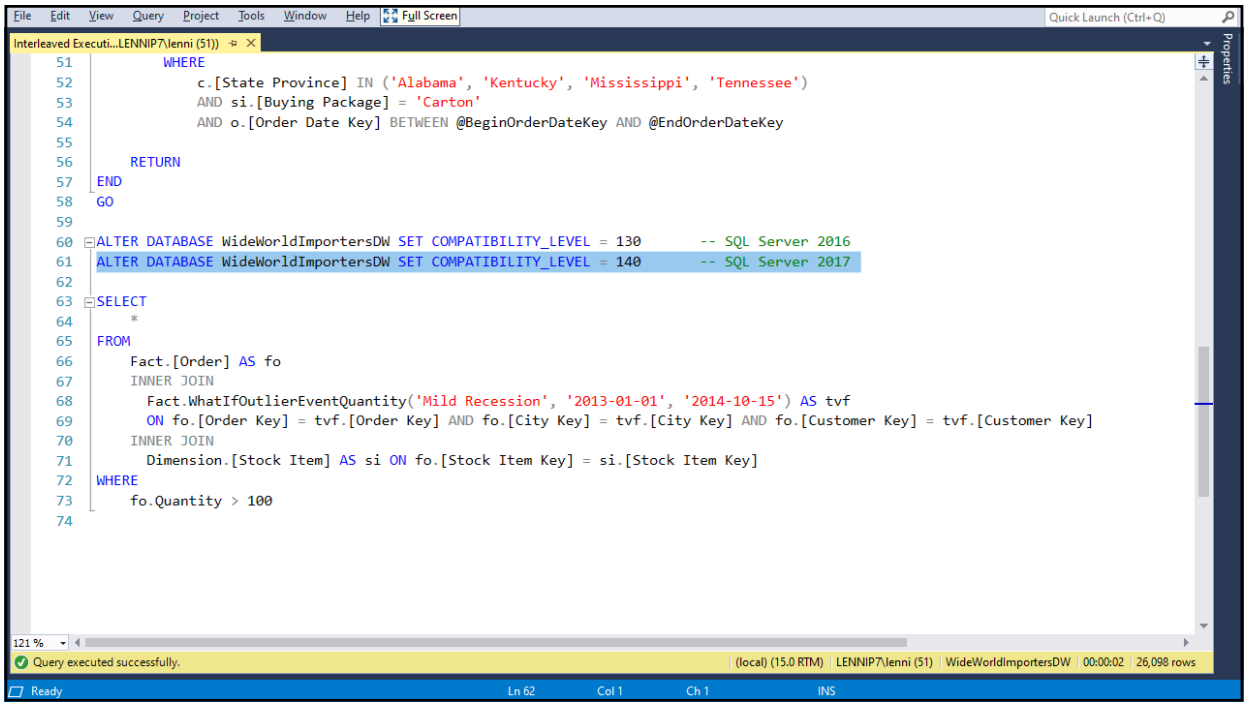

Query executed successfully.
(local) (15.0 RTM)
LENNIP7\lenni (51)
WideWorldImportersDW
00:00:02
26,098 rows
Ready


Nested Loops
For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows.
Physical Operation: Nested Loops
Logical Operation: Inner Join
Actual Execution Mode: Row
Estimated Execution Mode: Row
Actual Number of Rows: 26098
Actual Number of Batches: 0
Estimated Operator Cost: 0.0000473 (0%)
Estimated I/O Cost: 0
Estimated CPU Cost: 0.000047
Estimated Subtree Cost: 0.157822
Number of Executions: 1
Estimated Number of Executions: 1
Estimated Number of Rows: 11.2484
Estimated Row Size: 4736 B
Actual Rebinds: 0
Actual Rewinds: 0
Node ID: 2
Output List
Outer References

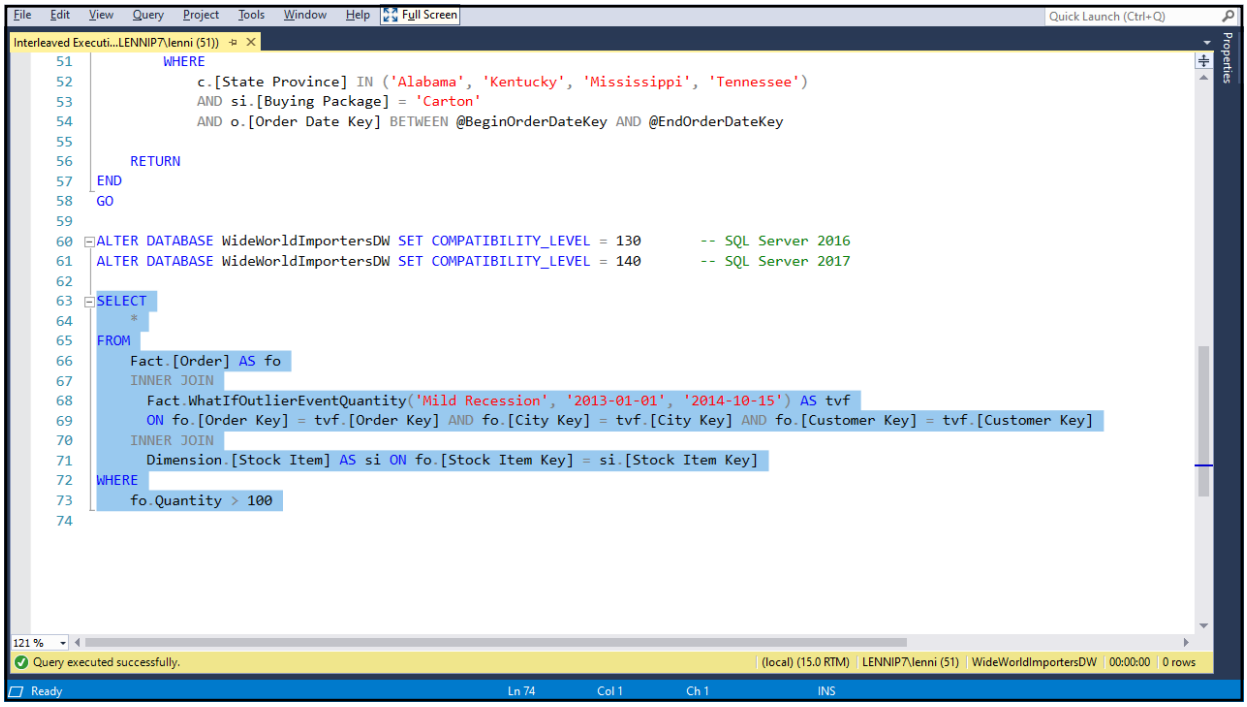

WHERE
c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
AND si.[Buying Package] = 'Carton'
AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
RETURN
END
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Fact.[Order] AS fo INNER JOIN Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf ON fo.[Order Key...
Query executed successfully.
Ready


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Interleaved Executi...LENNIP7\lenni (51))
Properties
51 WHERE
52 c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53 AND si.[Buying Package] = 'Carton'
54 AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
55
56 RETURN
57 END
58 GO
59
60 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 130 -- SQL Server 2016
61 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
62
63 SELECT
64 *
65 FROM
66 Fact.[Order] AS fo
67 INNER JOIN
68 Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf
69 ON fo.[Order Key] = tvf.[Order Key] AND fo.[City Key] = tvf.[City Key] AND fo.[Customer Key] = tvf.[Customer Key]
70 INNER JOIN
71 Dimension.[Stock Item] AS si ON fo.[Stock Item Key] = si.[Stock Item Key]
72 WHERE
73 fo.Quantity > 100
74
121 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImportersDW 00:00:02 26,098 rows
Ready Ln 62 Col 1 Ch 1 INS

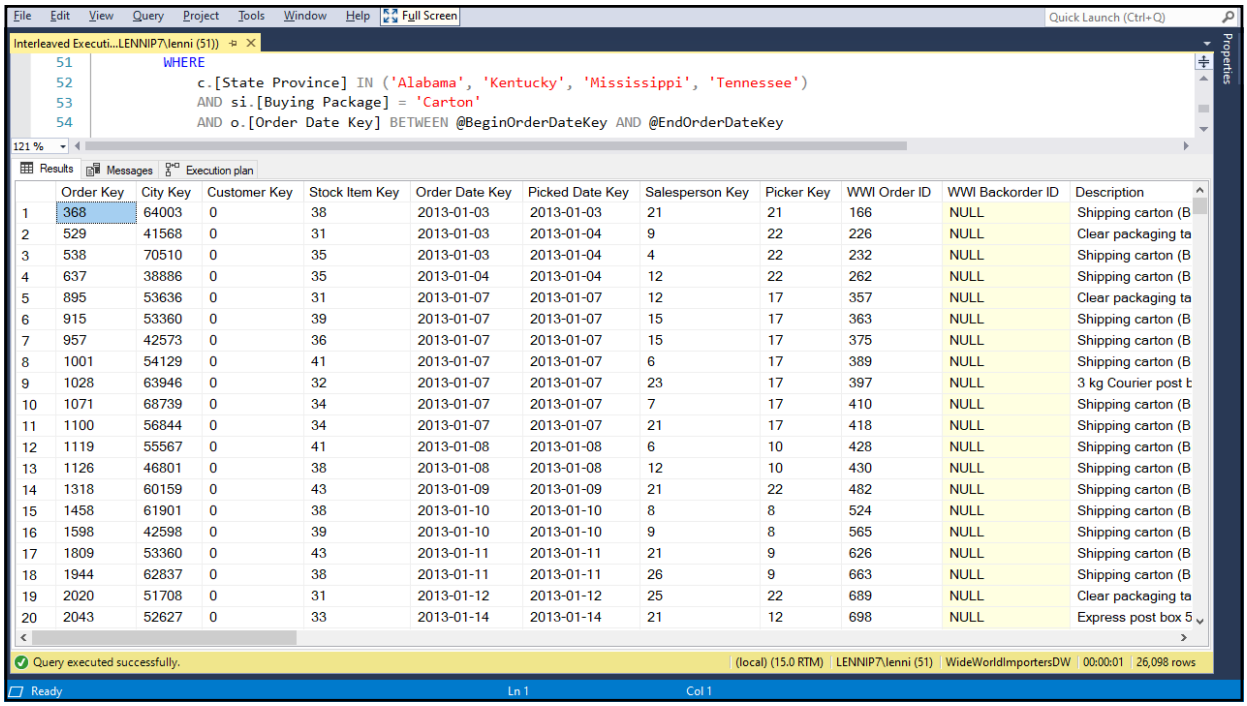

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Interleaved Executi...LENNIP7\lenni (51))
Properties
51 WHERE
52 c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53 AND si.[Buying Package] = 'Carton'
54 AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
55
56 RETURN
57 END
121 %
Messages
Commands completed successfully.
Completion time: 2020-02-18T15:30:28.8606281-05:00
121 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImportersDW 00:00:00 0 rows
Ready Ln 62 Col 1 Ch 1 INS



```
        WHERE
            c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
            AND si.[Buying Package] = 'Carton'
            AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey

    RETURN
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 130       -- SQL Server 2016
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017

SELECT
    *
FROM
    Fact.[Order] AS fo
    INNER JOIN
      Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf
      ON fo.[Order Key] = tvf.[Order Key] AND fo.[City Key] = tvf.[City Key] AND fo.[Customer Key] = tvf.[Customer Key]
    INNER JOIN
      Dimension.[Stock Item] AS si ON fo.[Stock Item Key] = si.[Stock Item Key]
WHERE
    fo.Quantity > 100
```

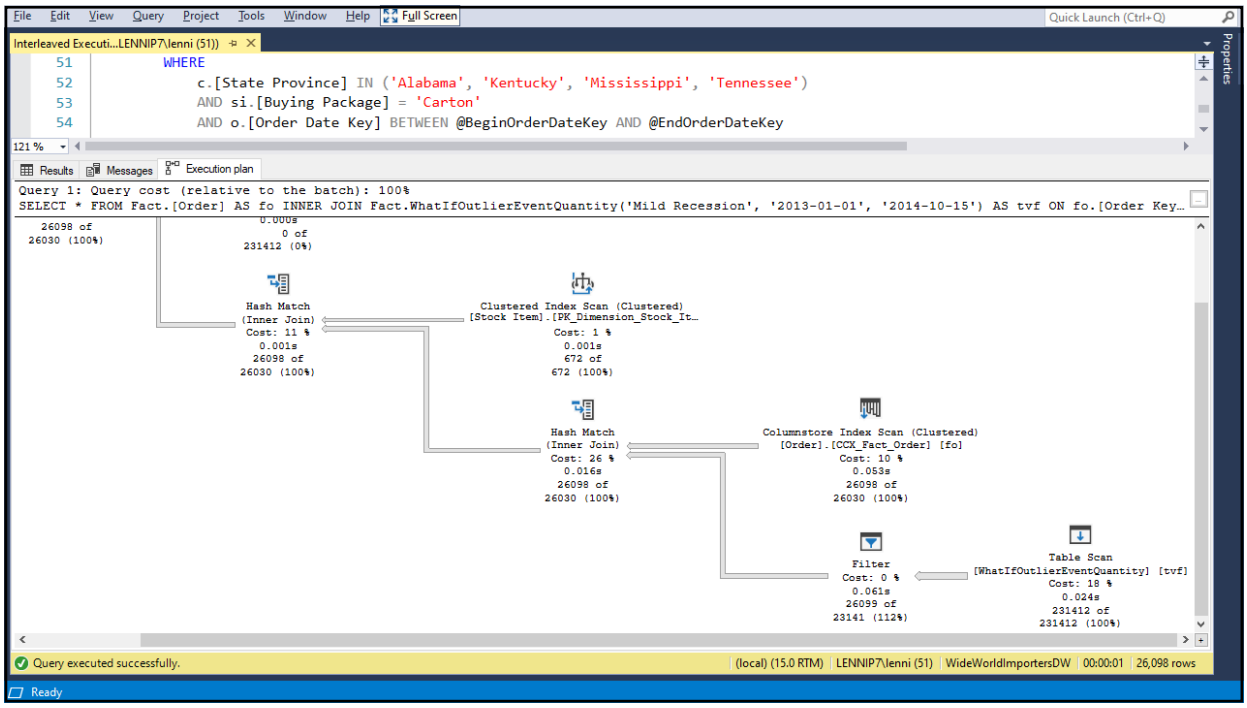


```
        WHERE
            c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
            AND si.[Buying Package] = 'Carton'
            AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey

    RETURN
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 130       -- SQL Server 2016
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017

SELECT
    *
FROM
    Fact.[Order] AS fo
    INNER JOIN
      Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf
      ON fo.[Order Key] = tvf.[Order Key] AND fo.[City Key] = tvf.[City Key] AND fo.[Customer Key] = tvf.[Customer Key]
    INNER JOIN
      Dimension.[Stock Item] AS si ON fo.[Stock Item Key] = si.[Stock Item Key]
WHERE
    fo.Quantity > 100
```



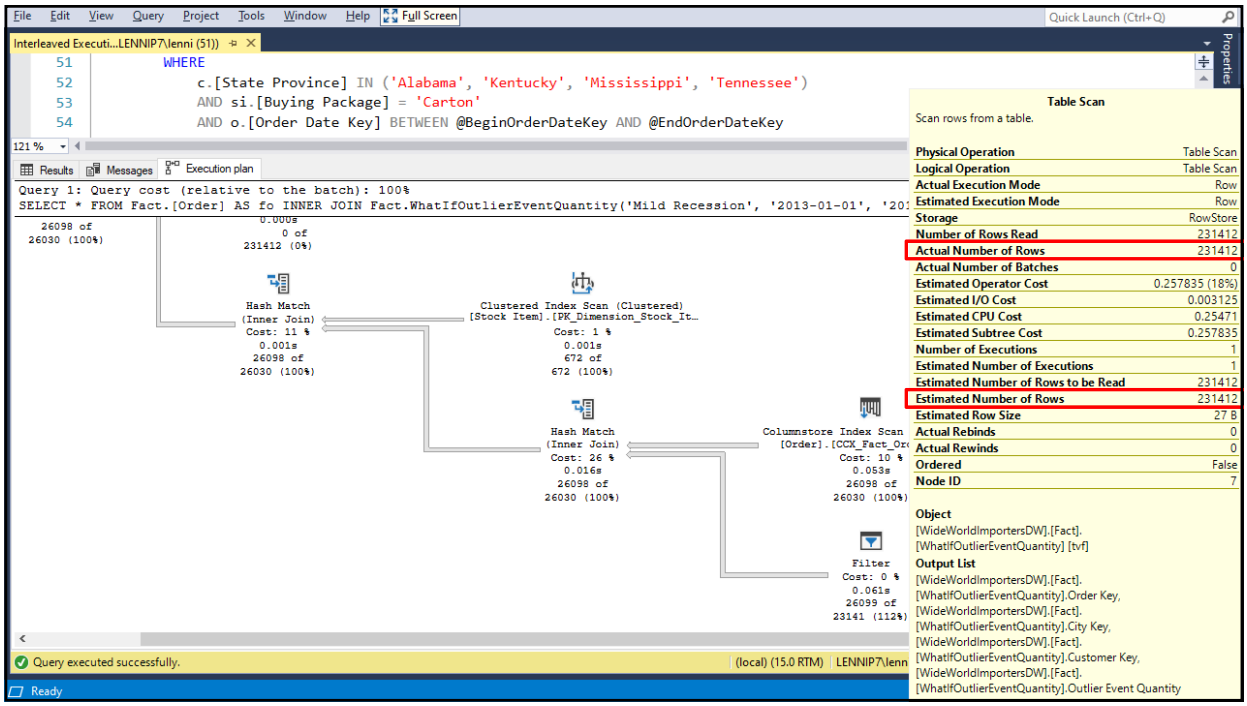
```
51      WHERE
52          c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53          AND si.[Buying Package] = 'Carton'
54          AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
```

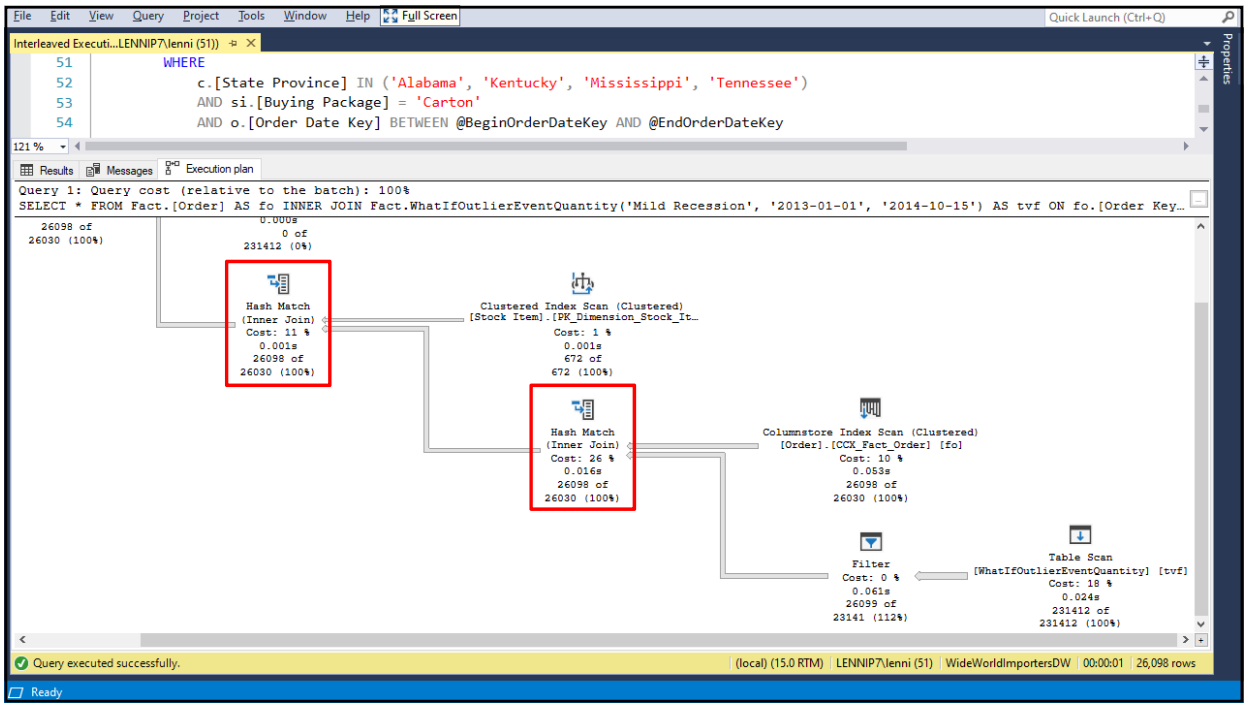
	Order Key	City Key	Customer Key	Stock Item Key	Order Date Key	Picked Date Key	Salesperson Key	Picker Key	WWI Order ID	WWI Backorder ID	Description
1	368	64003	0	38	2013-01-03	2013-01-03	21	21	166	NULL	Shipping carton (B
2	529	41568	0	31	2013-01-03	2013-01-04	9	22	226	NULL	Clear packaging ta
3	538	70510	0	35	2013-01-03	2013-01-04	4	22	232	NULL	Shipping carton (B
4	637	38886	0	35	2013-01-04	2013-01-04	12	22	262	NULL	Shipping carton (B
5	895	53636	0	31	2013-01-07	2013-01-07	12	17	357	NULL	Clear packaging ta
6	915	53360	0	39	2013-01-07	2013-01-07	15	17	363	NULL	Shipping carton (B
7	957	42573	0	36	2013-01-07	2013-01-07	15	17	375	NULL	Shipping carton (B
8	1001	54129	0	41	2013-01-07	2013-01-07	6	17	389	NULL	Shipping carton (B
9	1028	63946	0	32	2013-01-07	2013-01-07	23	17	397	NULL	3 kg Courier post b
10	1071	68739	0	34	2013-01-07	2013-01-07	7	17	410	NULL	Shipping carton (B
11	1100	56844	0	34	2013-01-07	2013-01-07	21	17	418	NULL	Shipping carton (B
12	1119	55567	0	41	2013-01-08	2013-01-08	6	10	428	NULL	Shipping carton (B
13	1126	46801	0	38	2013-01-08	2013-01-08	12	10	430	NULL	Shipping carton (B
14	1318	60159	0	43	2013-01-09	2013-01-09	21	22	482	NULL	Shipping carton (B
15	1458	61901	0	38	2013-01-10	2013-01-10	8	8	524	NULL	Shipping carton (B
16	1598	42598	0	39	2013-01-10	2013-01-10	9	8	565	NULL	Shipping carton (B
17	1809	53360	0	43	2013-01-11	2013-01-11	21	9	626	NULL	Shipping carton (B
18	1944	62837	0	38	2013-01-11	2013-01-11	26	9	663	NULL	Shipping carton (B
19	2020	51708	0	31	2013-01-12	2013-01-12	25	22	689	NULL	Clear packaging ta
20	2043	52627	0	33	2013-01-14	2013-01-14	21	12	698	NULL	Express post box 5



```
51      WHERE
52          c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53          AND si.[Buying Package] = 'Carton'
54          AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
```

```
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Fact.[Order] AS fo INNER JOIN Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf ON fo.[Order Key…
```


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Interleaved Executi...LENNIP7\lenni (51))
51 WHERE
52 c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53 AND si.[Buying Package] = 'Carton'
54 AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
121 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Fact.[Order] AS fo INNER JOIN Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '201
26098 of 26030 (100%)
0.000s 0 of 231412 (0%)
Hash Match (Inner Join) Cost: 11 % 0.001s 26098 of 26030 (100%)
Clustered Index Scan (Clustered) [Stock Item].[PK_Dimension_Stock_It... Cost: 1 % 0.001s 672 of 672 (100%)
Hash Match (Inner Join) Cost: 26 % 0.016s 26098 of 26030 (100%)
Columnstore Index Scan [Order].[CCX_Fact_Or Cost: 10 % 0.053s 26098 of 26030 (100%)
Filter Cost: 0 % 0.061s 26099 of 23141 (112%)
Table Scan
Scan rows from a table.
Physical Operation Table Scan
Logical Operation Table Scan
Actual Execution Mode Row
Estimated Execution Mode Row
Storage RowStore
Number of Rows Read 231412
Actual Number of Rows 231412
Actual Number of Batches 0
Estimated Operator Cost 0.257835 (18%)
Estimated I/O Cost 0.003125
Estimated CPU Cost 0.25471
Estimated Subtree Cost 0.257835
Number of Executions 1
Estimated Number of Executions 1
Estimated Number of Rows to be Read 231412
Estimated Number of Rows 231412
Estimated Row Size 27 B
Actual Rebinds 0
Actual Rewinds 0
Ordered False
Node ID 7
Object
[WideWorldImportersDW].[Fact].[WhatIfOutlierEventQuantity] [tvf]
Output List
[WideWorldImportersDW].[Fact].[WhatIfOutlierEventQuantity].Order Key, [WideWorldImportersDW].[Fact].[WhatIfOutlierEventQuantity].City Key, [WideWorldImportersDW].[Fact].[WhatIfOutlierEventQuantity].Customer Key, [WideWorldImportersDW].[Fact].[WhatIfOutlierEventQuantity].Outlier Event Quantity
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenn
Ready


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Interleaved Executi...LENNIP7\lenni (51))
51 WHERE
52 c.[State Province] IN ('Alabama', 'Kentucky', 'Mississippi', 'Tennessee')
53 AND si.[Buying Package] = 'Carton'
54 AND o.[Order Date Key] BETWEEN @BeginOrderDateKey AND @EndOrderDateKey
121 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT * FROM Fact.[Order] AS fo INNER JOIN Fact.WhatIfOutlierEventQuantity('Mild Recession', '2013-01-01', '2014-10-15') AS tvf ON fo.[Order Key...
26098 of 26030 (100%)
0.000s 0 of 231412 (0%)
Hash Match (Inner Join) Cost: 11 % 0.001s 26098 of 26030 (100%)
Clustered Index Scan (Clustered) [Stock Item].[PK_Dimension_Stock_It... Cost: 1 % 0.001s 672 of 672 (100%)
Hash Match (Inner Join) Cost: 26 % 0.016s 26098 of 26030 (100%)
Columnstore Index Scan (Clustered) [Order].[CCX_Fact_Order] [fo] Cost: 10 % 0.053s 26098 of 26030 (100%)
Filter Cost: 0 % 0.061s 26099 of 23141 (112%)
Table Scan [WhatIfOutlierEventQuantity] [tvf] Cost: 18 % 0.024s 231412 of 231412 (100%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImportersDW 00:00:01 26,098 rows
Ready

Interleaved Execution – Considerations and Limitations

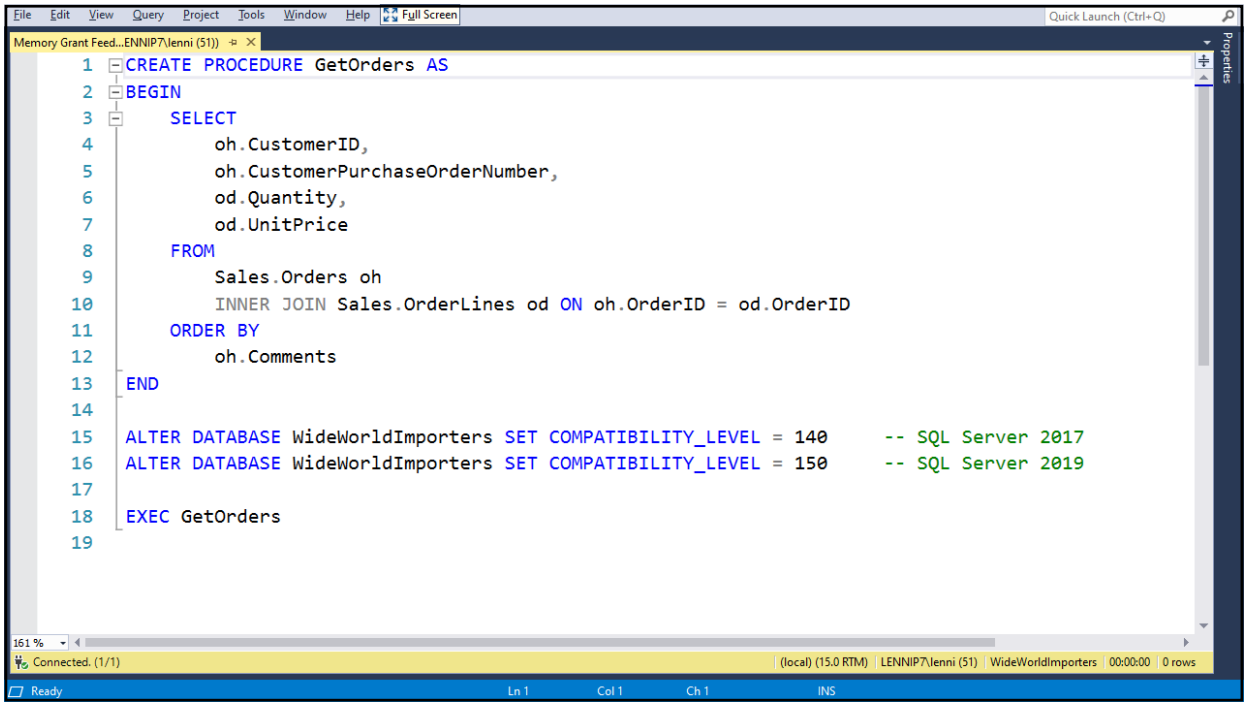
- MSTVF referencing statements
  - Must be read-only
    - Cannot be part of data modification operation
  - Must use runtime constants
    - E.g., TOP 5, not TOP (@Cnt)
- Greatest benefit when:
  - Higher skews between estimated and actual rows
  - Larger number of downstream plan operations

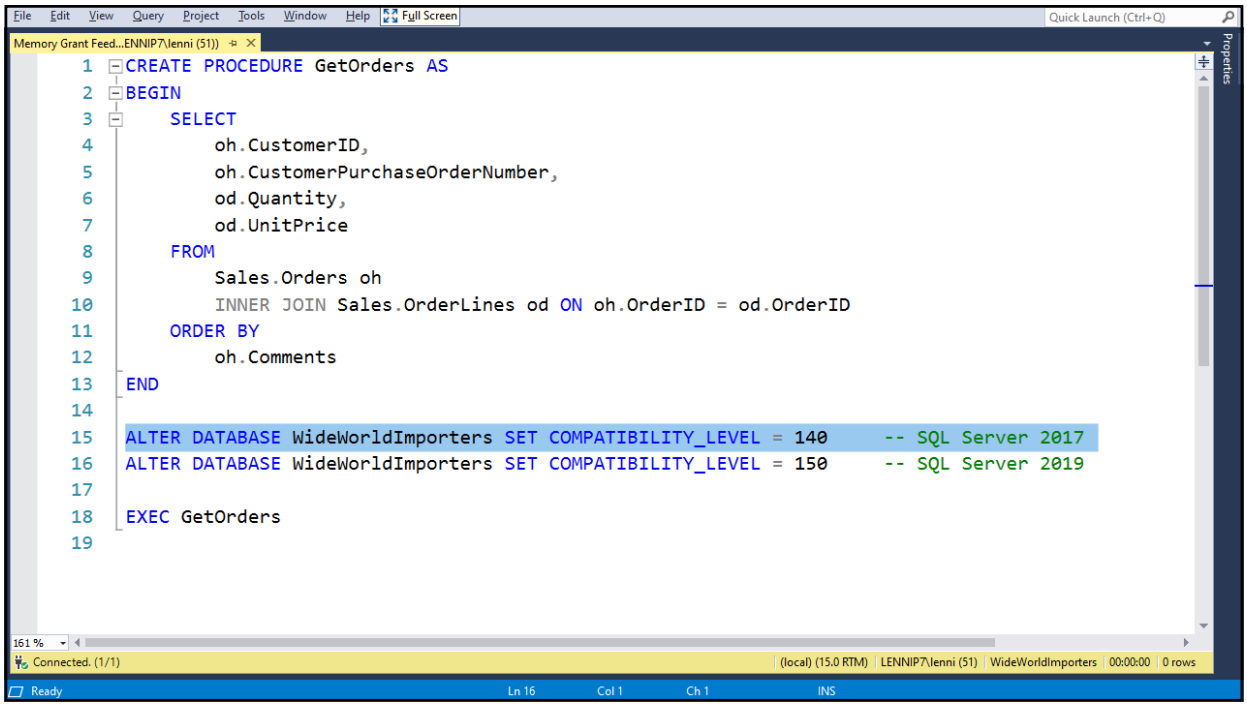
Memory Grant Feedback

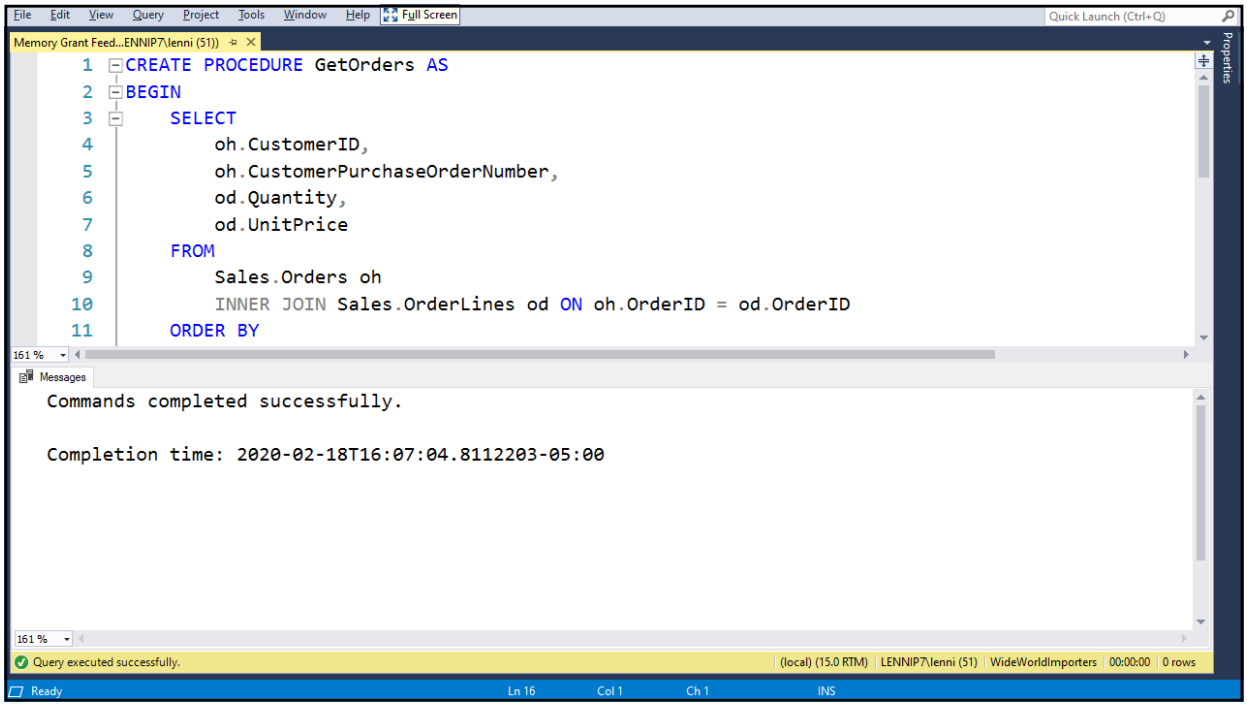
- What is a memory grant?
  - Allocating memory for query operations (e.g., hash joins and sorting)
- Problem:
  - If grant is too small, spills over to tempdb, pages memory to/from disk
  - If grant is too large, squeezes memory pressure, throttles queries
- Detects the misestimation after the first execution
  - Updates the cached plan for future executions
- SQL Server 2019 supports both batch and row modes
  - SQL Server 2017 supported batch mode only

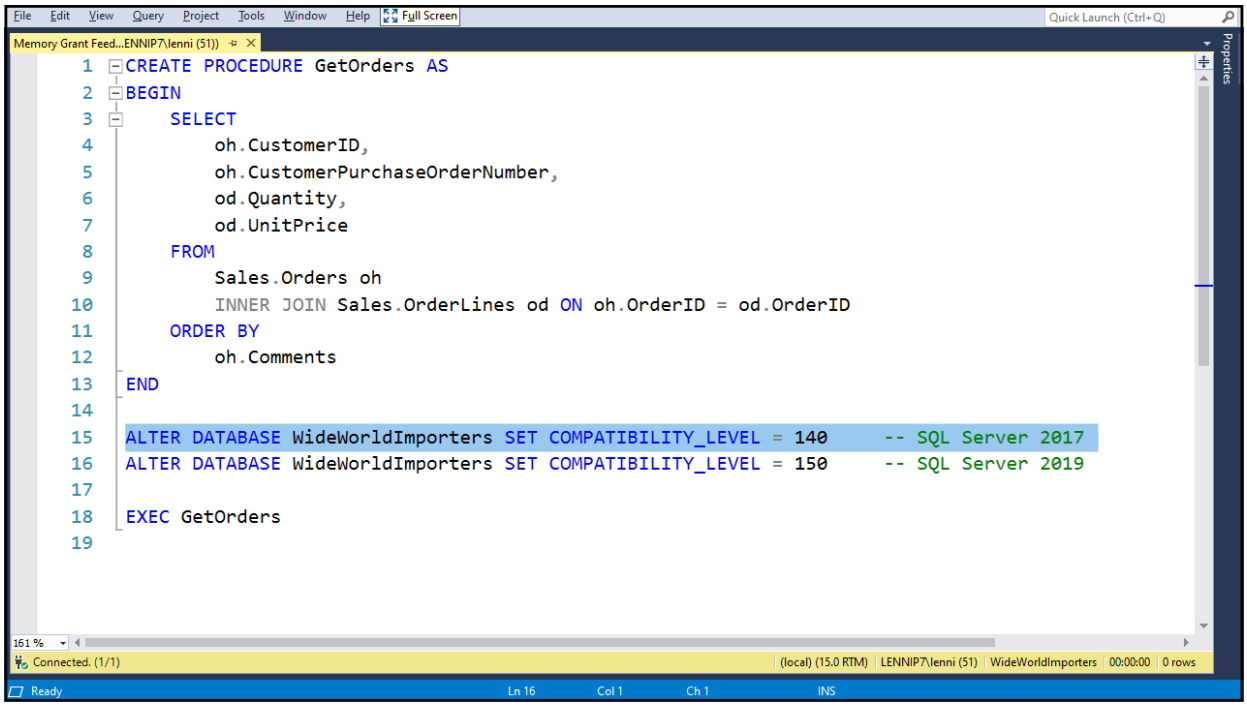
Memory Grant Feedback

demo

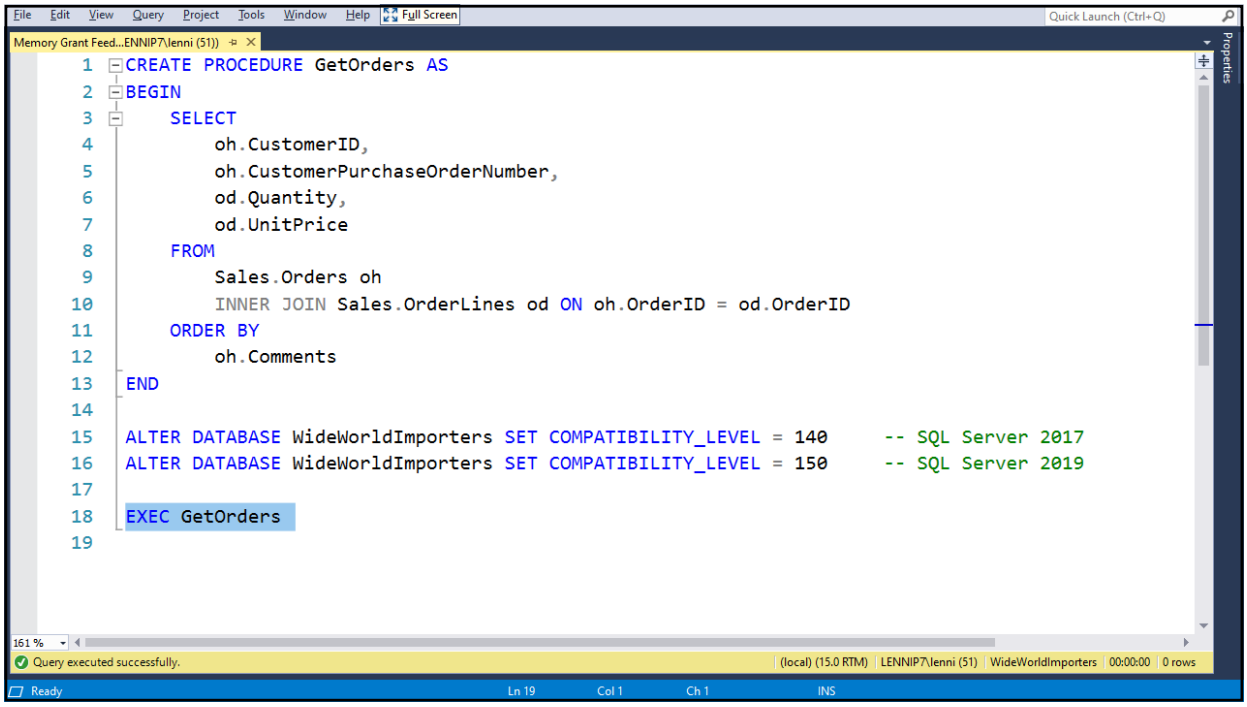



File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
12 oh.Comments
13 END
14
15 ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
16 ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
17
18 EXEC GetOrders
19
161 %
Connected. (1/1)
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:00 0 rows
Ready Ln 16 Col 1 Ch 1 INS

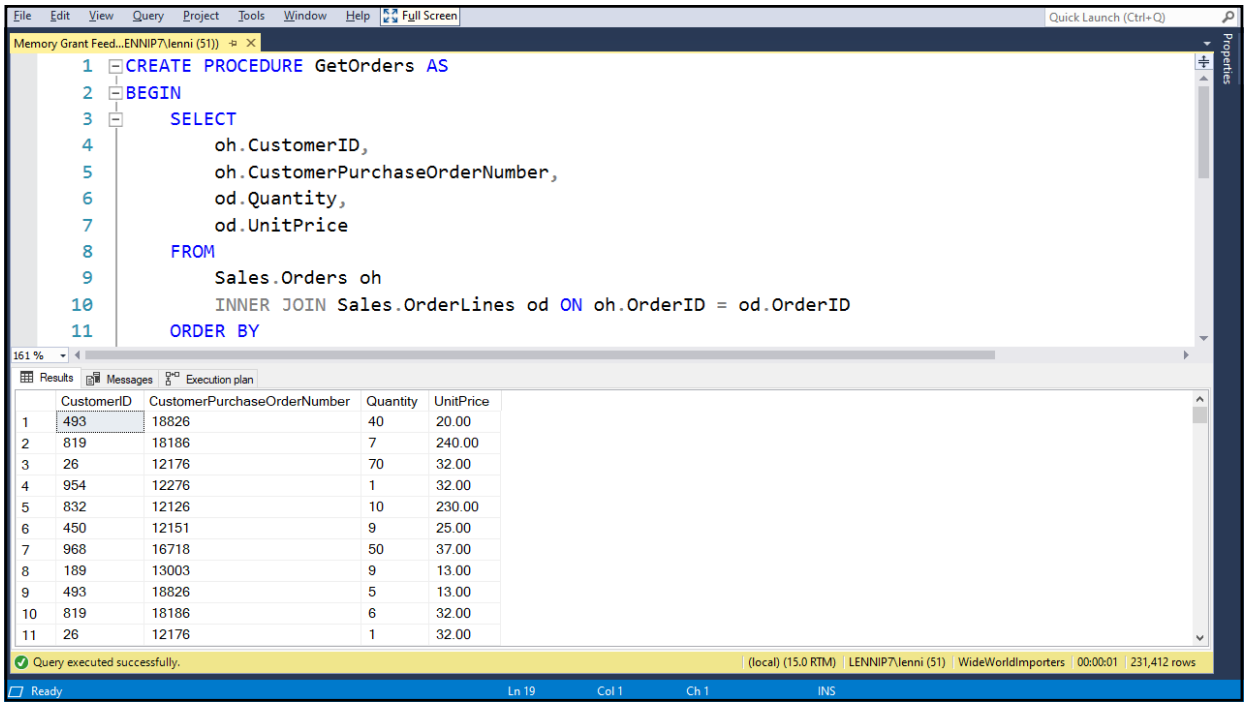

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
161 %
Messages
Commands completed successfully.
Completion time: 2020-02-18T16:07:04.8112203-05:00
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:00 0 rows
Ready Ln 16 Col 1 Ch 1 INS

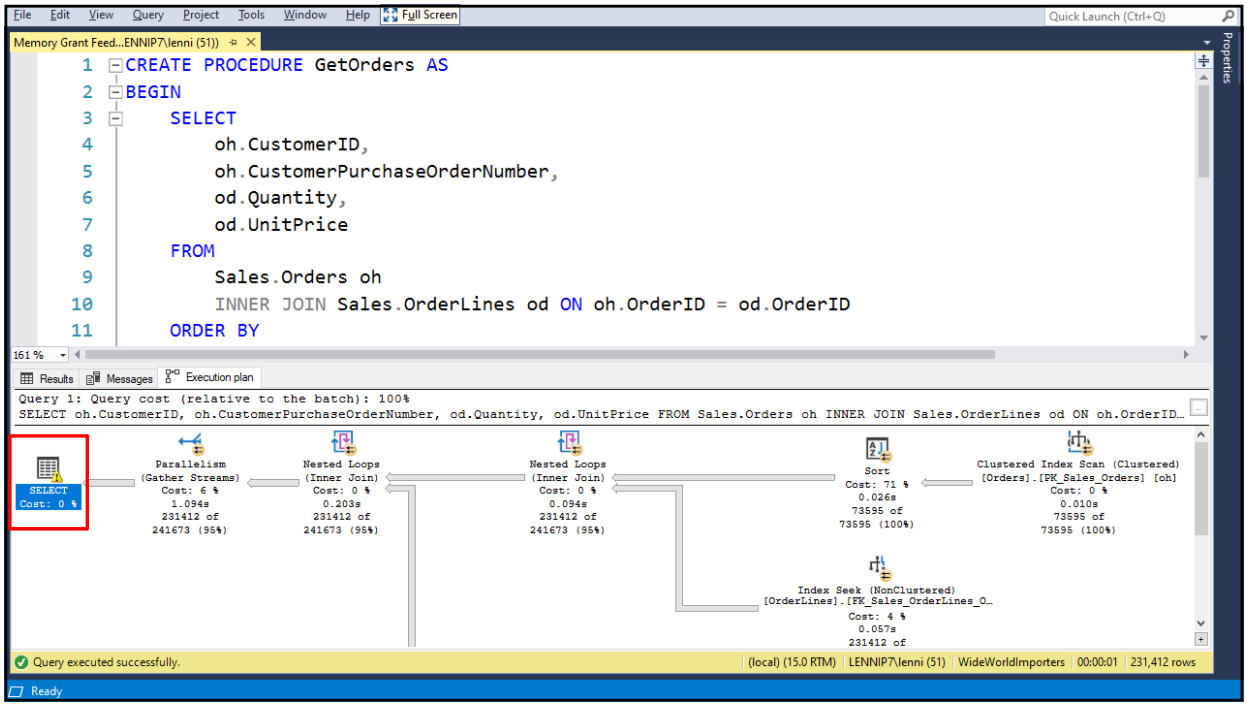


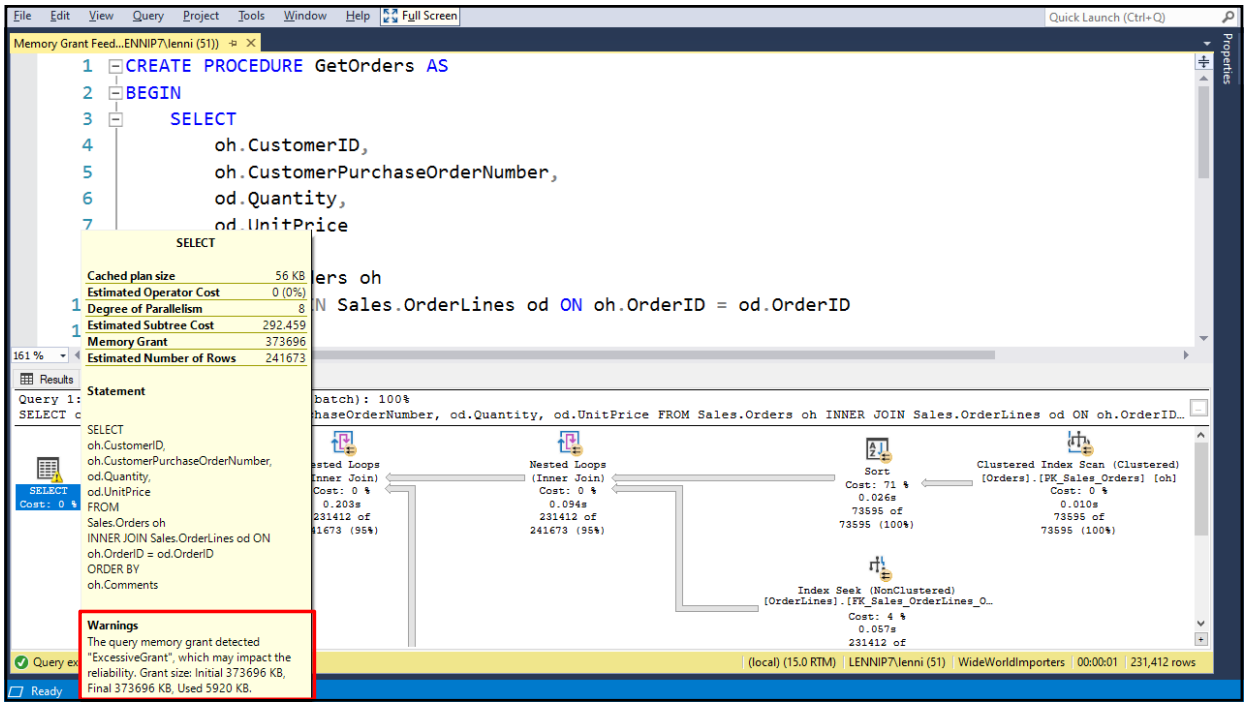
```
CREATE PROCEDURE GetOrders AS
BEGIN
    SELECT
        oh.CustomerID,
        oh.CustomerPurchaseOrderNumber,
        od.Quantity,
        od.UnitPrice
    FROM
        Sales.Orders oh
        INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
    ORDER BY
        oh.Comments
END

ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 140      -- SQL Server 2017
ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 150      -- SQL Server 2019

EXEC GetOrders
```

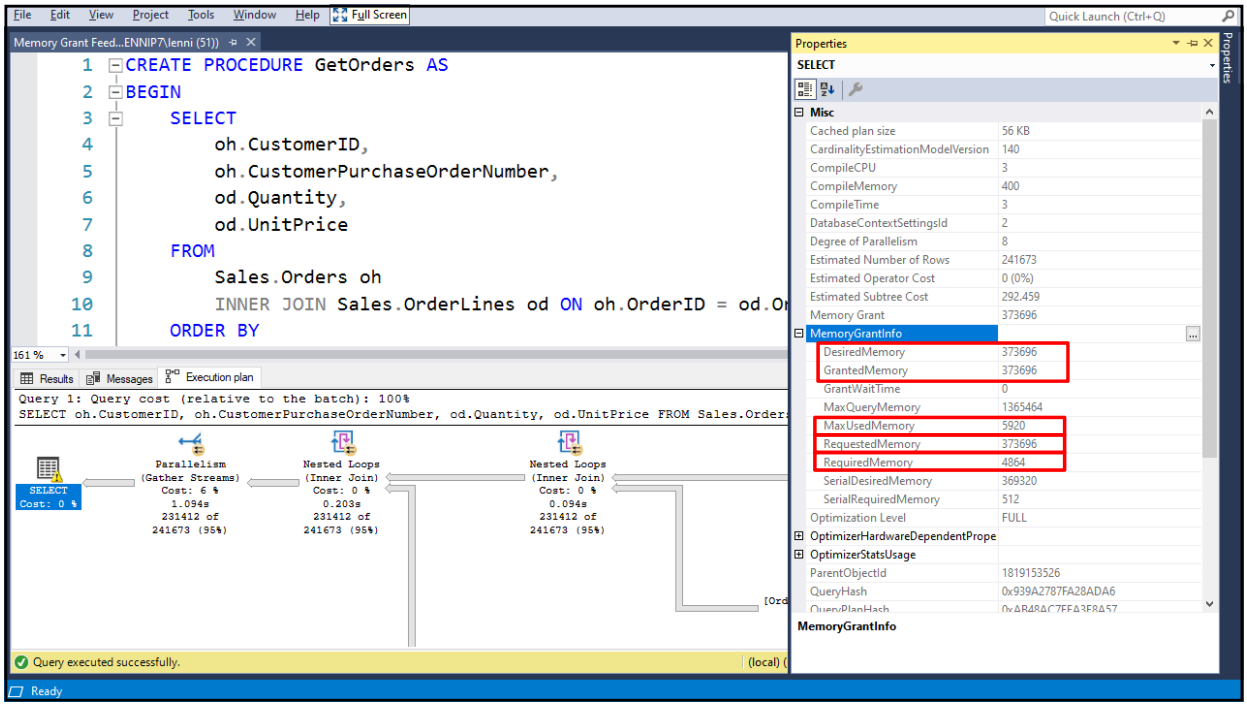


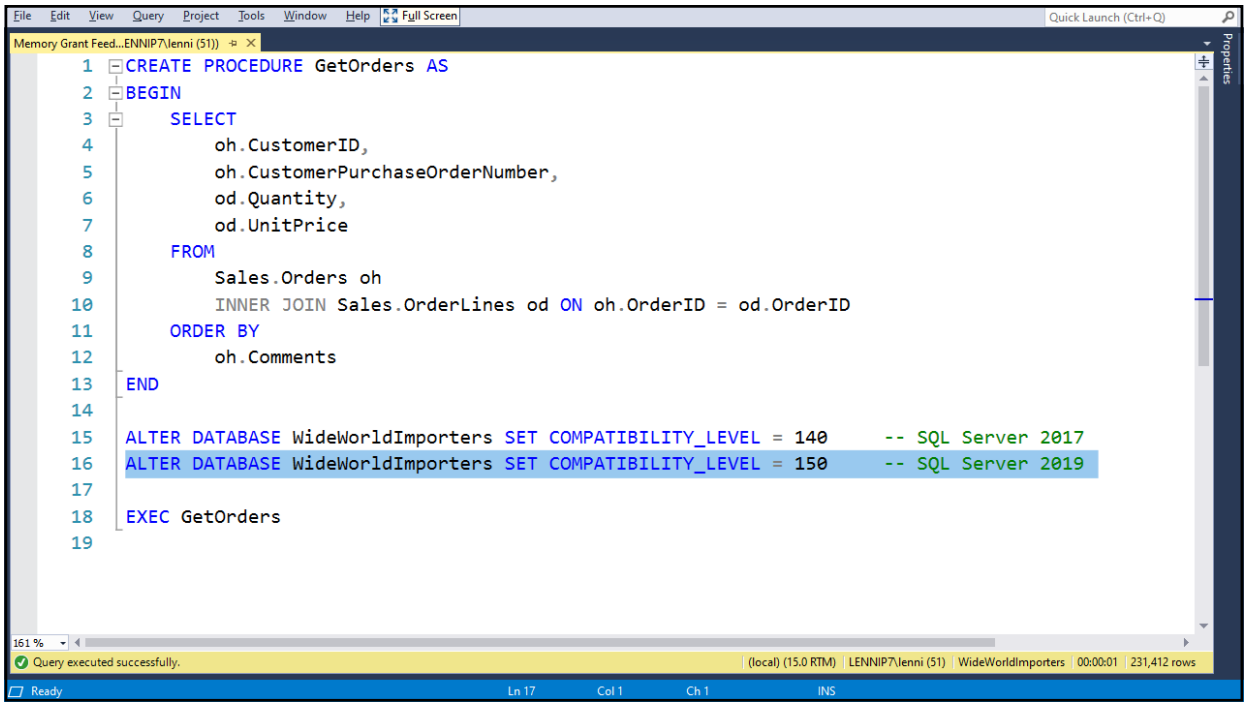
```
CREATE PROCEDURE GetOrders AS
BEGIN
    SELECT
        oh.CustomerID,
        oh.CustomerPurchaseOrderNumber,
        od.Quantity,
        od.UnitPrice
    FROM
        Sales.Orders oh
        INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
    ORDER BY
        oh.Comments
END

ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 140      -- SQL Server 2017
ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 150      -- SQL Server 2019

EXEC GetOrders
```

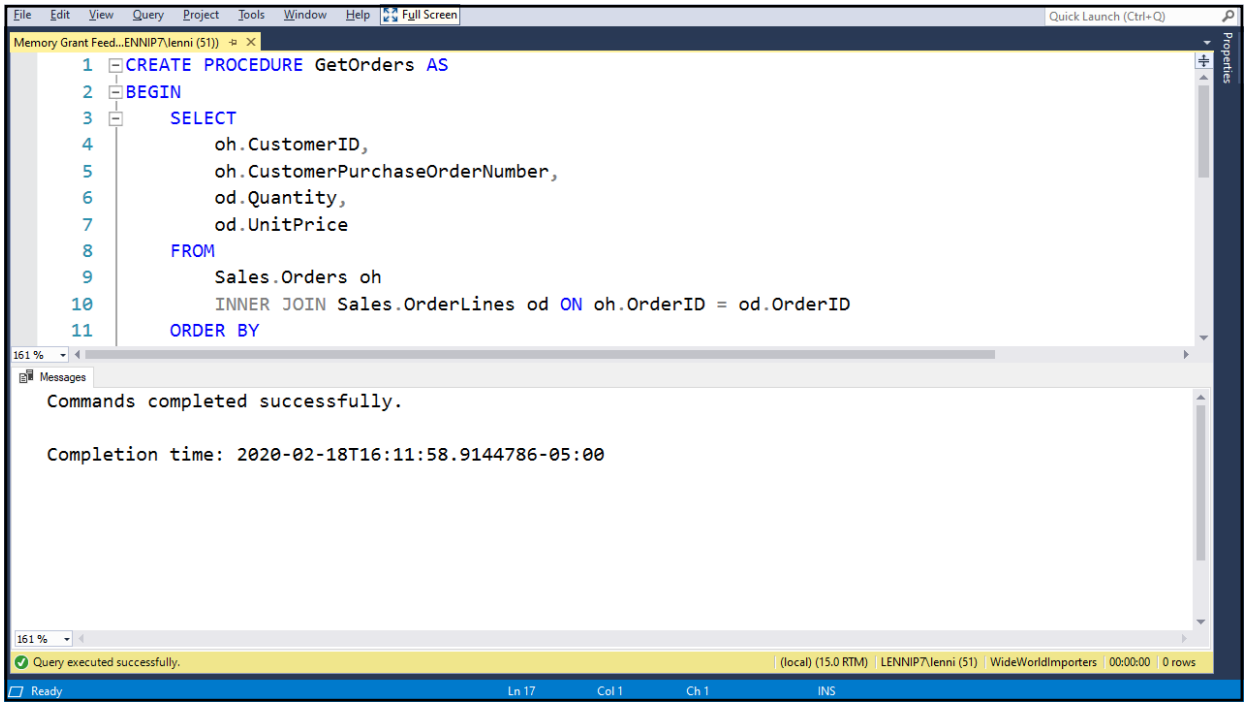

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
161 %
Results
Messages
Execution plan
CustomerID CustomerPurchaseOrderNumber Quantity UnitPrice
1 493 18826 40 20.00
2 819 18186 7 240.00
3 26 12176 70 32.00
4 954 12276 1 32.00
5 832 12126 10 230.00
6 450 12151 9 25.00
7 968 16718 50 37.00
8 189 13003 9 13.00
9 493 18826 5 13.00
10 819 18186 6 32.00
11 26 12176 1 32.00
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:01 231,412 rows
Ready
Ln 19 Col 1 Ch 1 INS

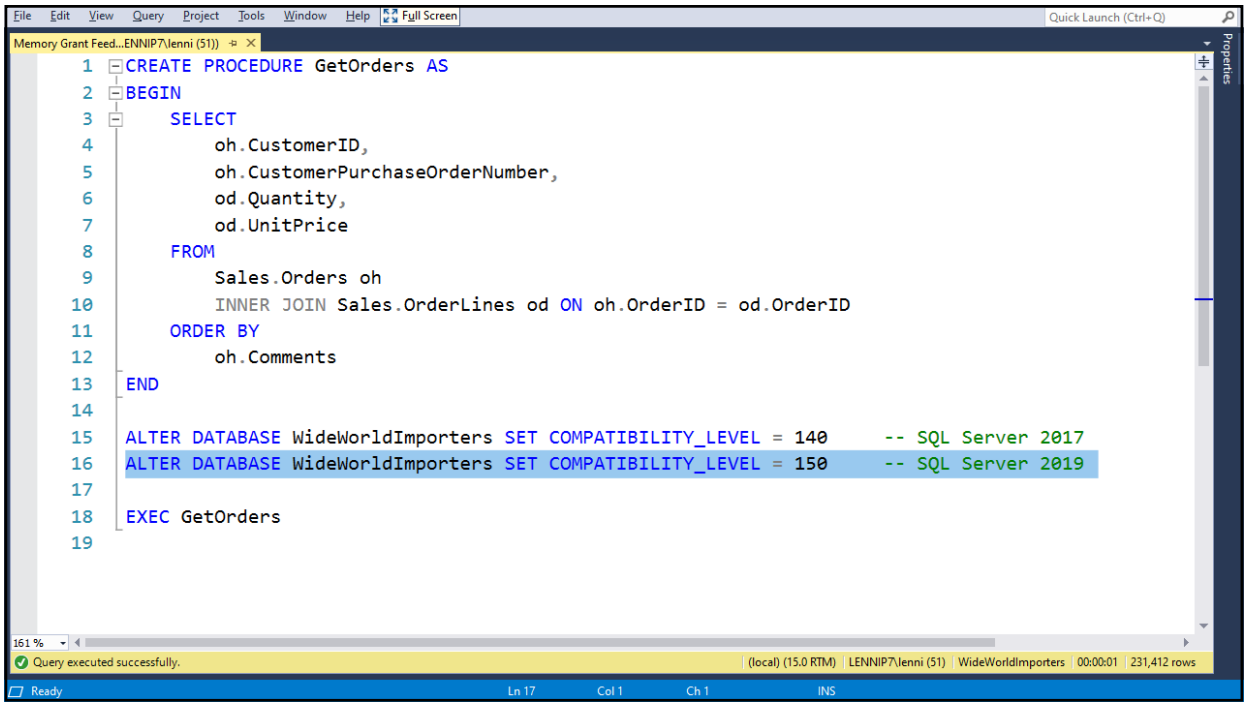

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
161 %
Results
Messages
Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT oh.CustomerID, oh.CustomerPurchaseOrderNumber, od.Quantity, od.UnitPrice FROM Sales.Orders oh INNER JOIN Sales.OrderLines od ON oh.OrderID...
SELECT
Cost: 0 %
Parallelism
(Gather Streams)
Cost: 6 %
1.094s
231412 of
241673 (95%)
Nested Loops
(Inner Join)
Cost: 0 %
0.203s
231412 of
241673 (95%)
Nested Loops
(Inner Join)
Cost: 0 %
0.094s
231412 of
241673 (95%)
Sort
Cost: 71 %
0.026s
73595 of
73595 (100%)
Clustered Index Scan (Clustered)
[Orders].[PK_Sales_Orders] [oh]
Cost: 0 %
0.010s
73595 of
73595 (100%)
Index Seek (NonClustered)
[OrderLines].[FK_Sales_OrderLines_O...
Cost: 4 %
0.057s
231412 of
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:01 231,412 rows
Ready


CREATE PROCEDURE GetOrders AS
BEGIN
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
od.Quantity,
od.UnitPrice
SELECT
Cached plan size 56 KB
Estimated Operator Cost 0 (0%)
Degree of Parallelism 8
Estimated Subtree Cost 292.459
Memory Grant 373696
Estimated Number of Rows 241673
Statement
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
od.Quantity,
od.UnitPrice
FROM
Sales.Orders oh
INNER JOIN Sales.OrderLines od ON
oh.OrderID = od.OrderID
ORDER BY
oh.Comments
Warnings
The query memory grant detected "ExcessiveGrant", which may impact the reliability. Grant size: Initial 373696 KB, Final 373696 KB, Used 5920 KB.
Nested Loops (Inner Join) Cost: 0 % 0.094s 231412 of 241673 (95%)
Sort Cost: 71 % 0.026s 73595 of 73595 (100%)
Clustered Index Scan (Clustered) [Orders].[PK_Sales_Orders] [oh] Cost: 0 % 0.010s 73595 of 73595 (100%)
Index Seek (NonClustered) [OrderLines].[FK_Sales_OrderLines_O... Cost: 4 % 0.057s 231412 of
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:01 231,412 rows
Ready

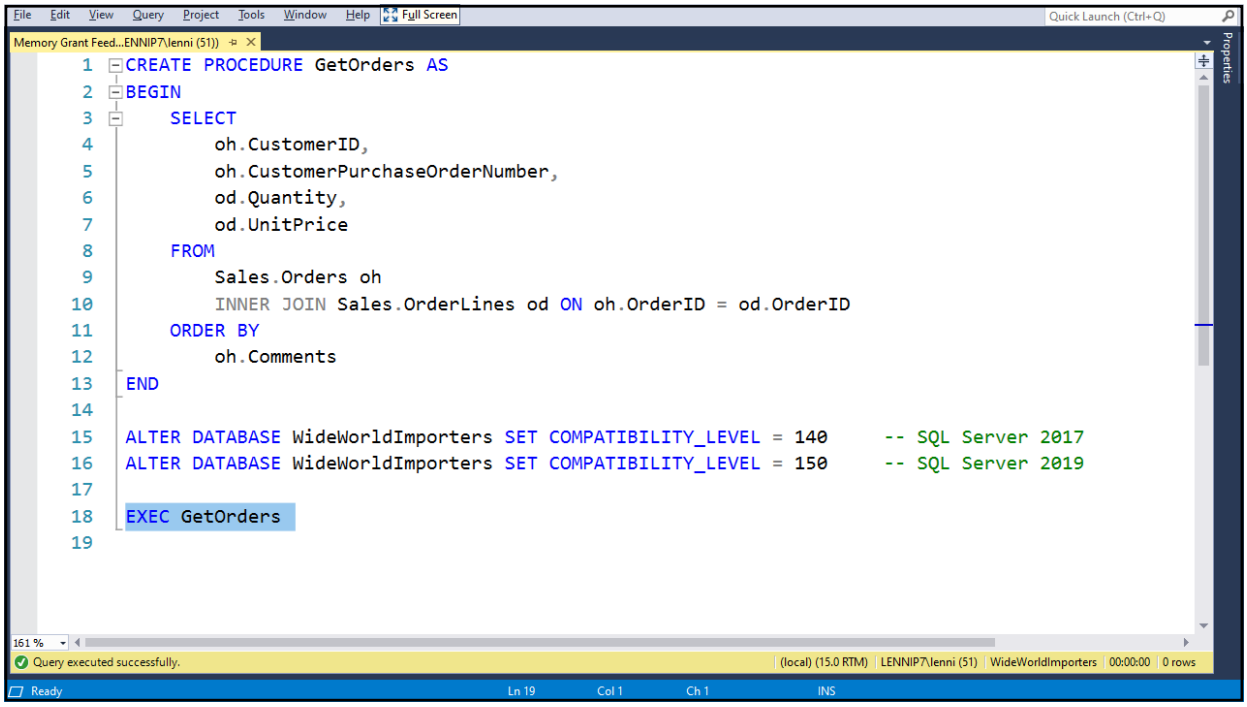

CREATE PROCEDURE GetOrders AS
BEGIN
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
od.Quantity,
od.UnitPrice
FROM
Sales.Orders oh
INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
ORDER BY
Query 1: Query cost (relative to the batch): 100%
SELECT oh.CustomerID, oh.CustomerPurchaseOrderNumber, od.Quantity, od.UnitPrice FROM Sales.Orders
Parallelism (Gather Streams) Cost: 6 % 1.094s 231412 of 241673 (95%)
Nested Loops (Inner Join) Cost: 0 % 0.203s 231412 of 241673 (95%)
Nested Loops (Inner Join) Cost: 0 % 0.094s 231412 of 241673 (95%)
Properties
SELECT
Misc
Cached plan size 56 KB
CardinalityEstimationModelVersion 140
CompileCPU 3
CompileMemory 400
CompileTime 3
DatabaseContextSettingsId 2
Degree of Parallelism 8
Estimated Number of Rows 241673
Estimated Operator Cost 0 (0%)
Estimated Subtree Cost 292.459
Memory Grant 373696
MemoryGrantInfo
DesiredMemory 373696
GrantedMemory 373696
GrantWaitTime 0
MaxQueryMemory 1365464
MaxUsedMemory 5920
RequestedMemory 373696
RequiredMemory 4864
SerialDesiredMemory 369320
SerialRequiredMemory 512
Optimization Level FULL
OptimizerHardwareDependentPrope
OptimizerStatsUsage
ParentObjectId 1819153526
QueryHash 0x939A2787FA28ADA6
MemoryGrantInfo
Query executed successfully.
Ready


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
12 oh.Comments
13 END
14
15 ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
16 ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
17
18 EXEC GetOrders
19
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:01 231,412 rows
Ready Ln 17 Col 1 Ch 1 INS

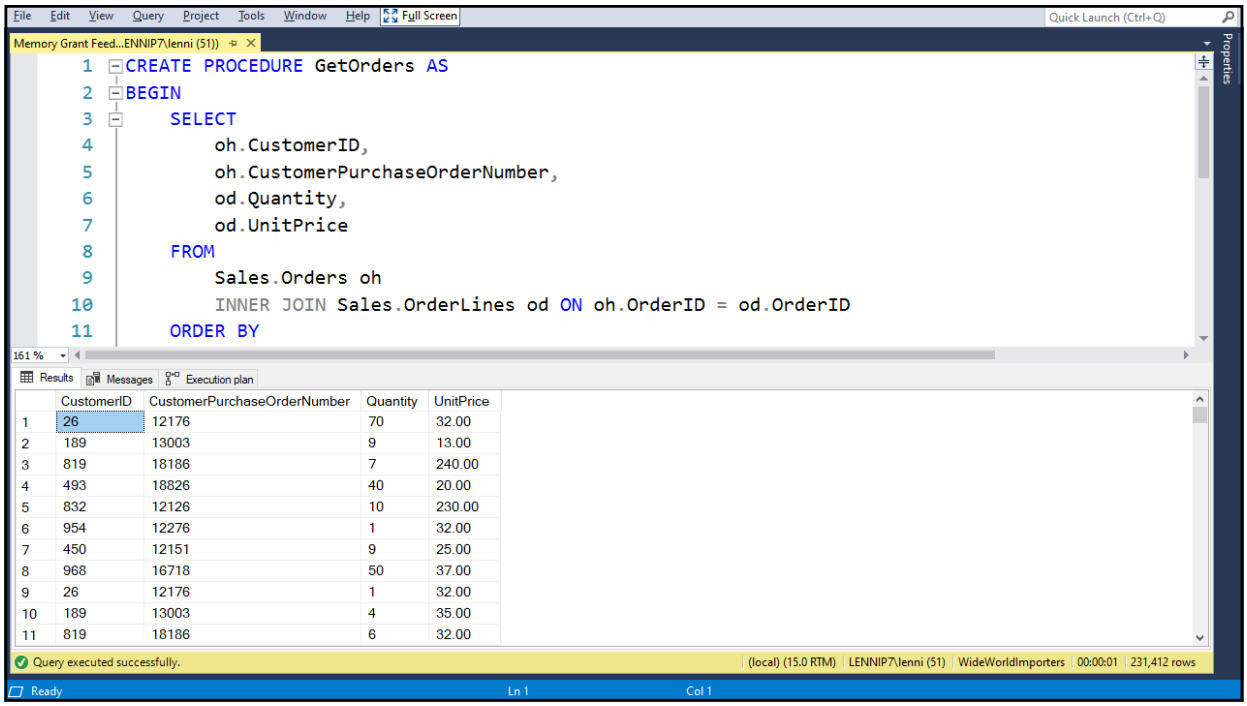

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
161 %
Messages
Commands completed successfully.
Completion time: 2020-02-18T16:11:58.9144786-05:00
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:00 0 rows
Ready Ln 17 Col 1 Ch 1 INS

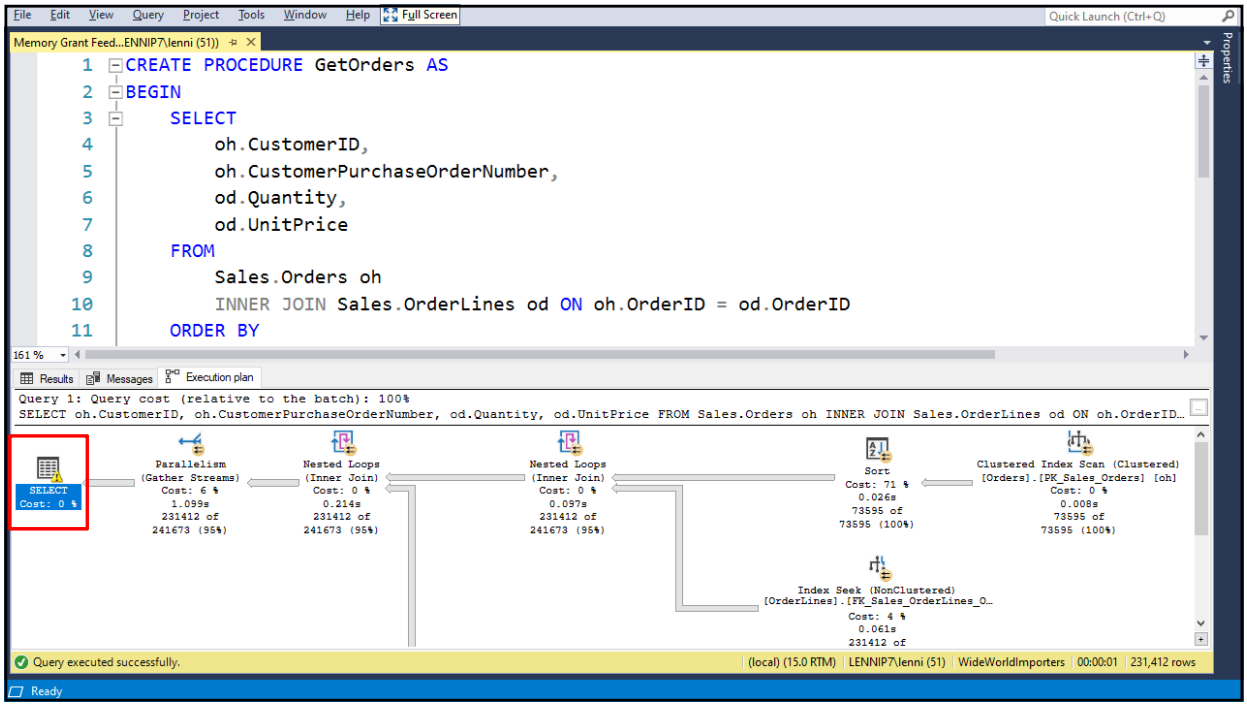


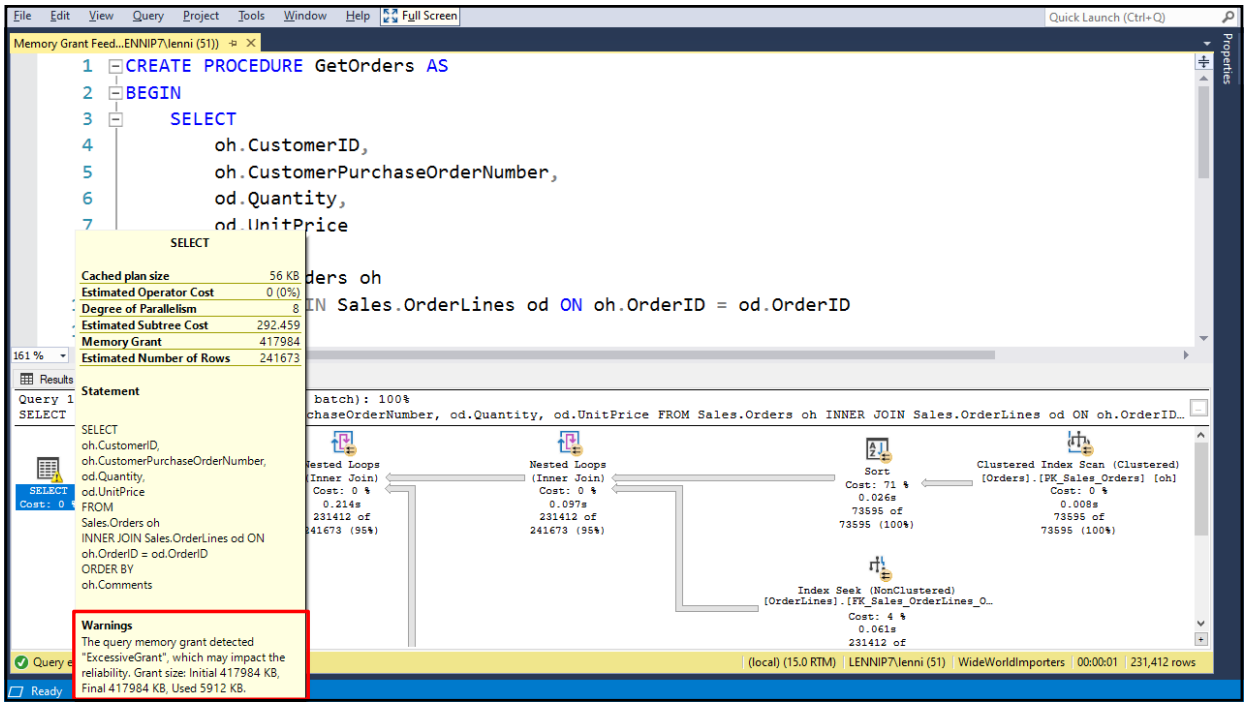
```
CREATE PROCEDURE GetOrders AS
BEGIN
    SELECT
        oh.CustomerID,
        oh.CustomerPurchaseOrderNumber,
        od.Quantity,
        od.UnitPrice
    FROM
        Sales.Orders oh
        INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
    ORDER BY
        oh.Comments
END

ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 140     -- SQL Server 2017
ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 150     -- SQL Server 2019

EXEC GetOrders
```

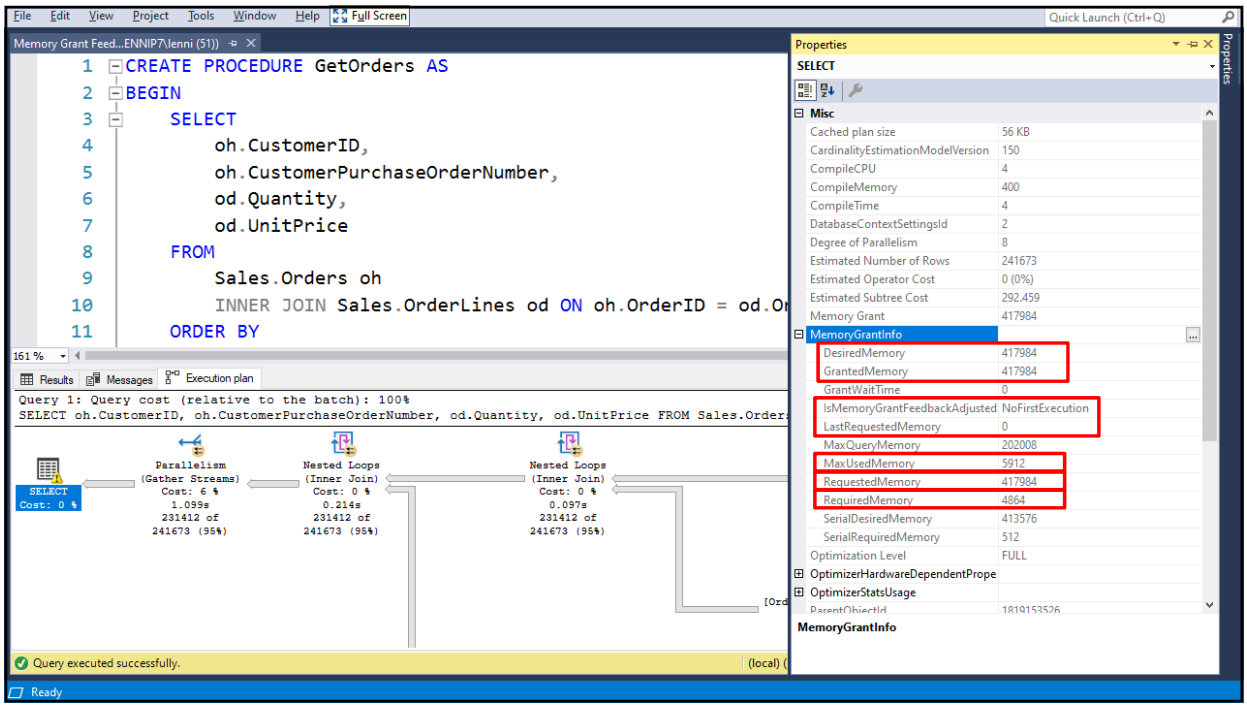



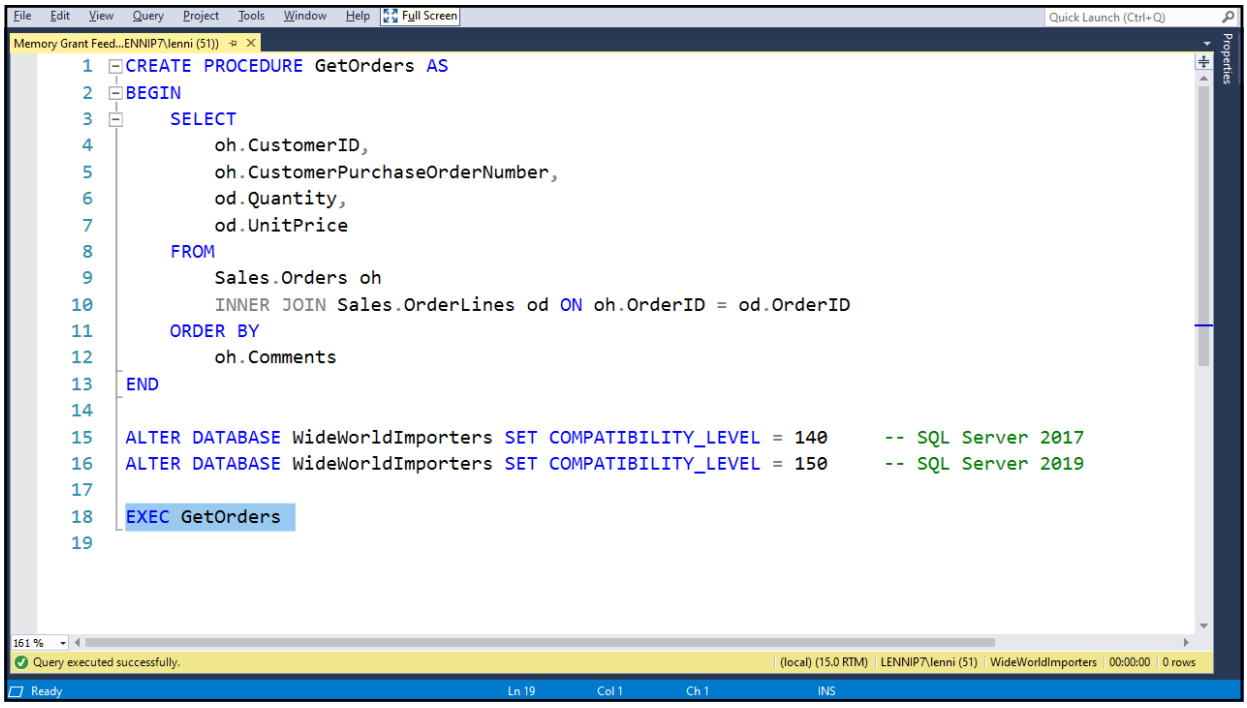
```
CREATE PROCEDURE GetOrders AS
BEGIN
    SELECT
        oh.CustomerID,
        oh.CustomerPurchaseOrderNumber,
        od.Quantity,
        od.UnitPrice
    FROM
        Sales.Orders oh
        INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
    ORDER BY
        oh.Comments
END

ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 140     -- SQL Server 2017
ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 150     -- SQL Server 2019

EXEC GetOrders
```

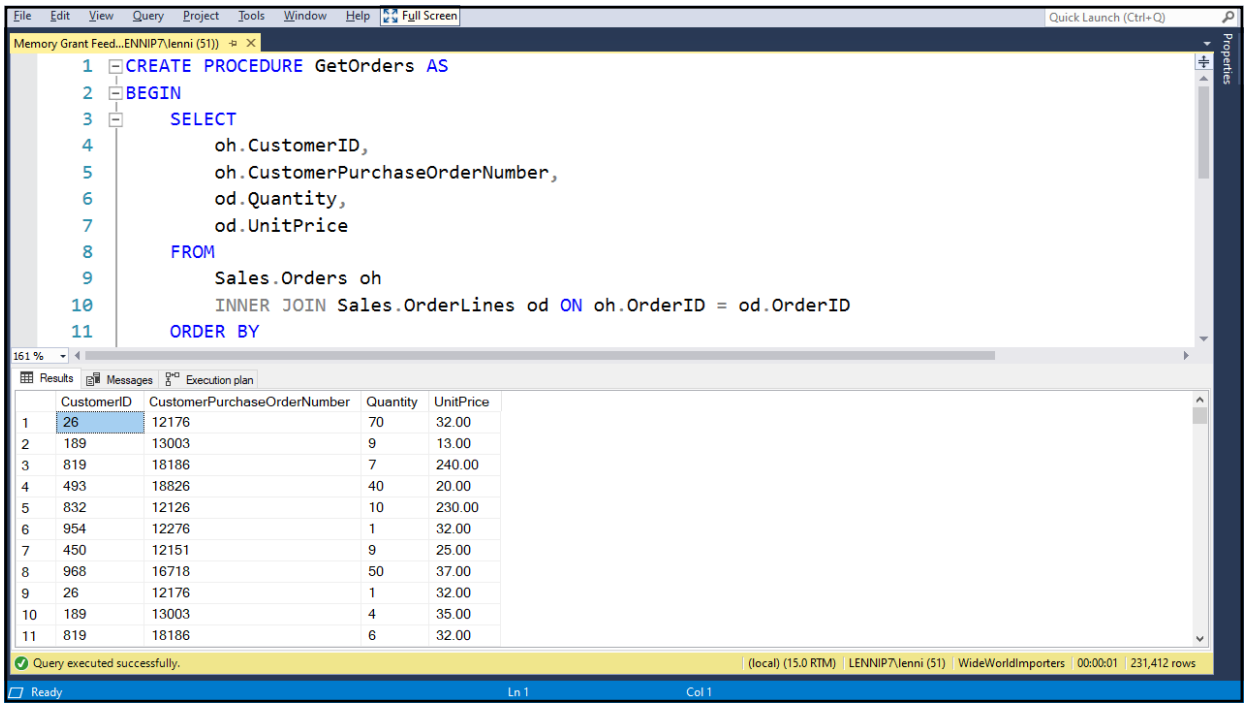

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
161 %
Results
Messages
Execution plan
CustomerID CustomerPurchaseOrderNumber Quantity UnitPrice
1 26 12176 70 32.00
2 189 13003 9 13.00
3 819 18186 7 240.00
4 493 18826 40 20.00
5 832 12126 10 230.00
6 954 12276 1 32.00
7 450 12151 9 25.00
8 968 16718 50 37.00
9 26 12176 1 32.00
10 189 13003 4 35.00
11 819 18186 6 32.00
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:01 231,412 rows
Ready
Ln 1
Col 1

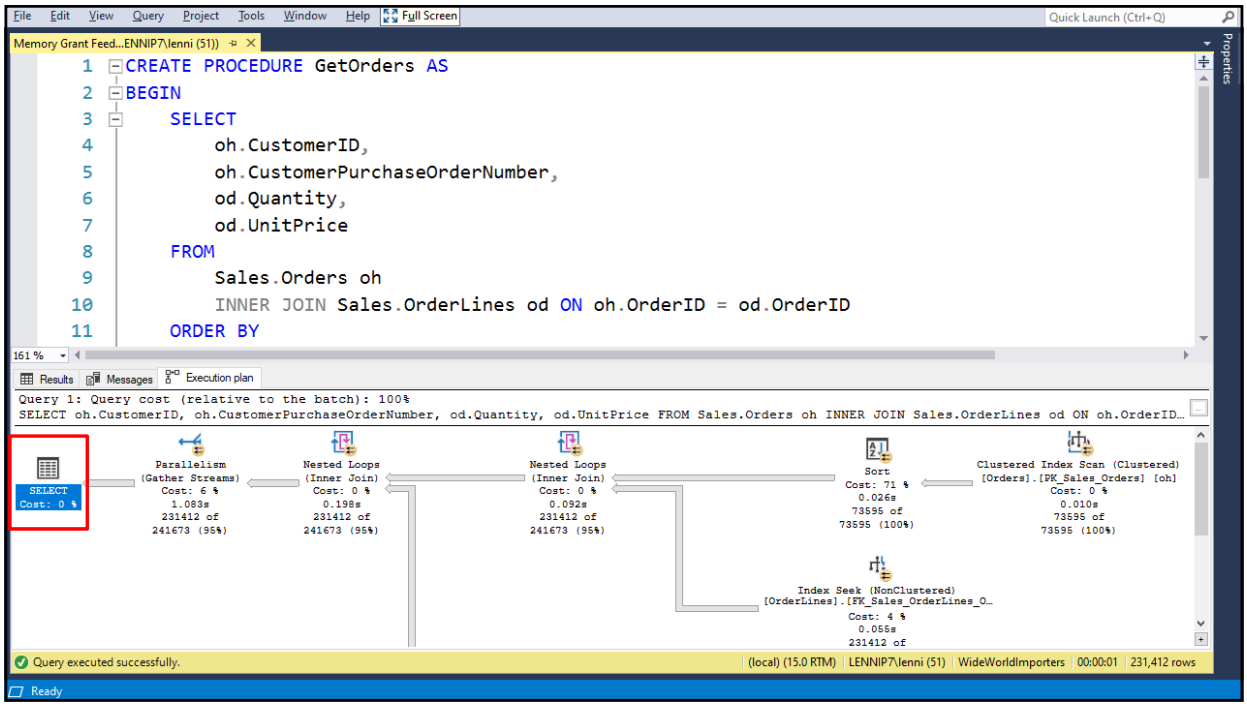

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
161 %
Results
Messages
Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT oh.CustomerID, oh.CustomerPurchaseOrderNumber, od.Quantity, od.UnitPrice FROM Sales.Orders oh INNER JOIN Sales.OrderLines od ON oh.OrderID...
SELECT
Cost: 0 %
Parallelism
(Gather Streams)
Cost: 6 %
1.099s
231412 of
241673 (95%)
Nested Loops
(Inner Join)
Cost: 0 %
0.214s
231412 of
241673 (95%)
Nested Loops
(Inner Join)
Cost: 0 %
0.097s
231412 of
241673 (95%)
Sort
Cost: 71 %
0.026s
73595 of
73595 (100%)
Clustered Index Scan (Clustered)
[Orders].[PK_Sales_Orders] [oh]
Cost: 0 %
0.008s
73595 of
73595 (100%)
Index Seek (NonClustered)
[OrderLines].[FK_Sales_OrderLines_O...
Cost: 4 %
0.061s
231412 of
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:01 231,412 rows
Ready

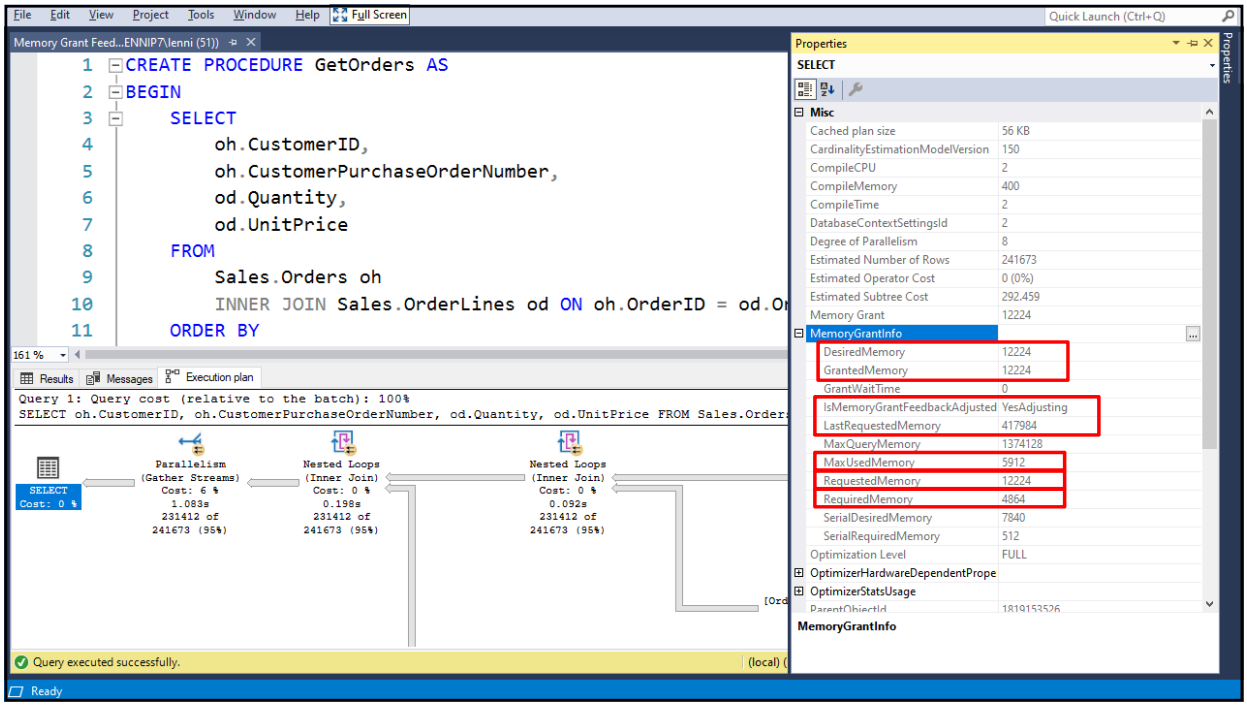

CREATE PROCEDURE GetOrders AS
BEGIN
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
od.Quantity,
od.UnitPrice
SELECT
Cached plan size 56 KB
Estimated Operator Cost 0 (0%)
Degree of Parallelism 8
Estimated Subtree Cost 292.459
Memory Grant 417984
Estimated Number of Rows 241673
Statement
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
od.Quantity,
od.UnitPrice
FROM
Sales.Orders oh
INNER JOIN Sales.OrderLines od ON
oh.OrderID = od.OrderID
ORDER BY
oh.Comments
Warnings
The query memory grant detected "ExcessiveGrant", which may impact the reliability. Grant size: Initial 417984 KB, Final 417984 KB, Used 5912 KB.
Nested Loops (Inner Join) Cost: 0 % 0.214s 231412 of 241673 (95%)
Nested Loops (Inner Join) Cost: 0 % 0.097s 231412 of 241673 (95%)
Sort Cost: 71 % 0.026s 73595 of 73595 (100%)
Clustered Index Scan (Clustered) [Orders].[PK_Sales_Orders] [oh] Cost: 0 % 0.008s 73595 of 73595 (100%)
Index Seek (NonClustered) [OrderLines].[FK_Sales_OrderLines_O... Cost: 4 % 0.061s 231412 of
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:01 231,412 rows
Ready

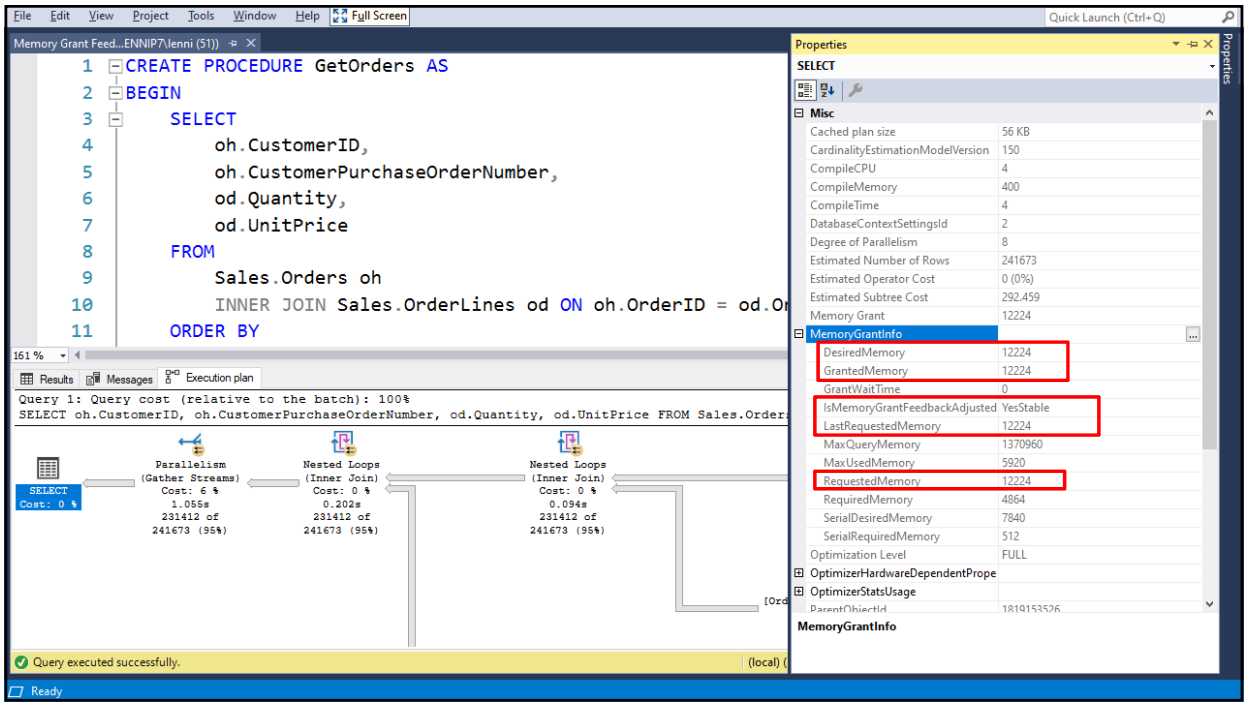

CREATE PROCEDURE GetOrders AS
BEGIN
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
od.Quantity,
od.UnitPrice
FROM
Sales.Orders oh
INNER JOIN Sales.OrderLines od ON oh.OrderID = od.O
ORDER BY
Properties
SELECT
Misc
Cached plan size 56 KB
CardinalityEstimationModelVersion 150
CompileCPU 4
CompileMemory 400
CompileTime 4
DatabaseContextSettingsId 2
Degree of Parallelism 8
Estimated Number of Rows 241673
Estimated Operator Cost 0 (0%)
Estimated Subtree Cost 292.459
Memory Grant 417984
MemoryGrantInfo
DesiredMemory 417984
GrantedMemory 417984
GrantWaitTime 0
IsMemoryGrantFeedbackAdjusted NoFirstExecution
LastRequestedMemory 0
MaxQueryMemory 202008
MaxUsedMemory 5912
RequestedMemory 417984
RequiredMemory 4864
SerialDesiredMemory 413576
SerialRequiredMemory 512
Optimization Level FULL
OptimizerHardwareDependentPrope
OptimizerStatsUsage
MemoryGrantInfo
Parallelism (Gather Streams) Cost: 6 % 1.099s 231412 of 241673 (95%)
Nested Loops (Inner Join) Cost: 0 % 0.214s 231412 of 241673 (95%)
Nested Loops (Inner Join) Cost: 0 % 0.097s 231412 of 241673 (95%)
Query executed successfully.
Ready

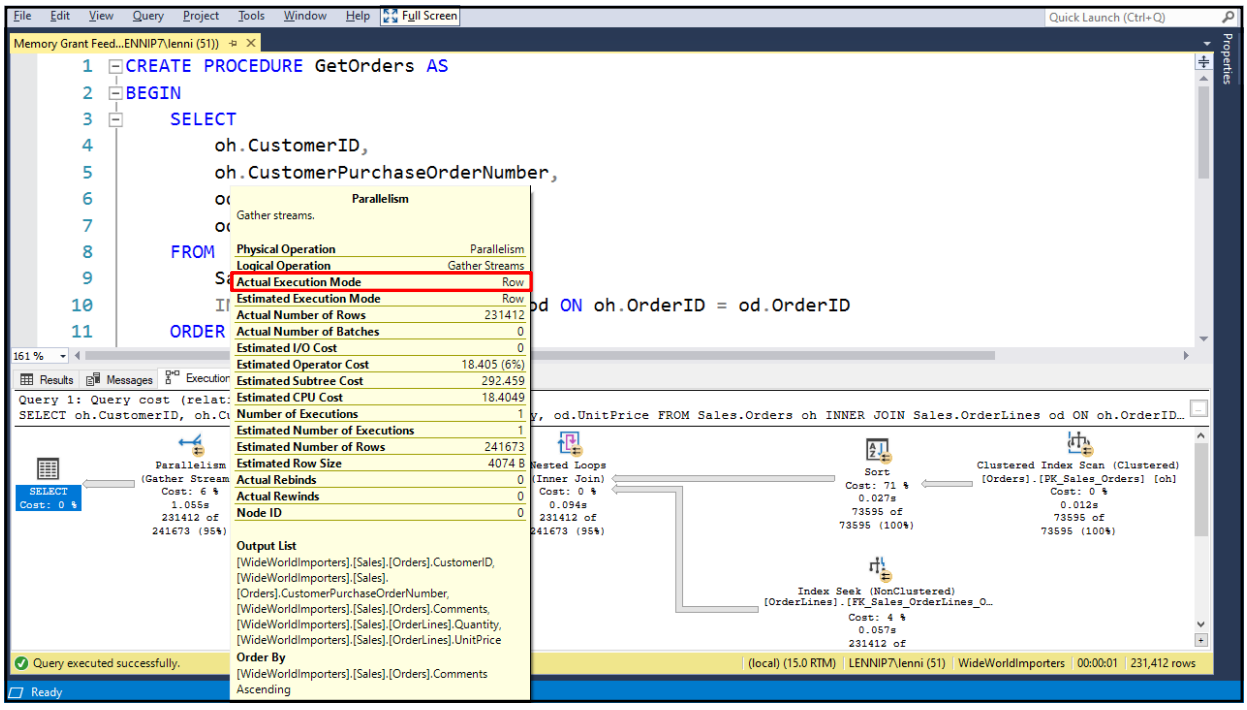

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
12 oh.Comments
13 END
14
15 ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
16 ALTER DATABASE WideWorldImporters SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
17
18 EXEC GetOrders
19
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:00 0 rows
Ready Ln 19 Col 1 Ch 1 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
Properties
1 CREATE PROCEDURE GetOrders AS
2 BEGIN
3 SELECT
4 oh.CustomerID,
5 oh.CustomerPurchaseOrderNumber,
6 od.Quantity,
7 od.UnitPrice
8 FROM
9 Sales.Orders oh
10 INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
11 ORDER BY
161 %
Results Messages Execution plan
CustomerID CustomerPurchaseOrderNumber Quantity UnitPrice
1 26 12176 70 32.00
2 189 13003 9 13.00
3 819 18186 7 240.00
4 493 18826 40 20.00
5 832 12126 10 230.00
6 954 12276 1 32.00
7 450 12151 9 25.00
8 968 16718 50 37.00
9 26 12176 1 32.00
10 189 13003 4 35.00
11 819 18186 6 32.00
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:01 231,412 rows
Ready Ln 1 Col 1


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
CREATE PROCEDURE GetOrders AS
BEGIN
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
od.Quantity,
od.UnitPrice
FROM
Sales.Orders oh
INNER JOIN Sales.OrderLines od ON oh.OrderID = od.OrderID
ORDER BY
161 %
Results
Messages
Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT oh.CustomerID, oh.CustomerPurchaseOrderNumber, od.Quantity, od.UnitPrice FROM Sales.Orders oh INNER JOIN Sales.OrderLines od ON oh.OrderID...
SELECT
Cost: 0 %
Parallelism
(Gather Streams)
Cost: 6 %
1.083s
231412 of
241673 (95%)
Nested Loops
(Inner Join)
Cost: 0 %
0.198s
231412 of
241673 (95%)
Nested Loops
(Inner Join)
Cost: 0 %
0.092s
231412 of
241673 (95%)
Sort
Cost: 71 %
0.026s
73595 of
73595 (100%)
Clustered Index Scan (Clustered)
[Orders].[PK_Sales_Orders] [oh]
Cost: 0 %
0.010s
73595 of
73595 (100%)
Index Seek (NonClustered)
[OrderLines].[FK_Sales_OrderLines_O...
Cost: 4 %
0.055s
231412 of
Query executed successfully.
(local) (15.0 RTM)
LENNIP7\lenni (51)
WideWorldImporters
00:00:01
231,412 rows
Ready
Properties


File Edit View Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Memory Grant Feed...ENNIP7\lenni (51))
CREATE PROCEDURE GetOrders AS
BEGIN
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
od.Quantity,
od.UnitPrice
FROM
Sales.Orders oh
INNER JOIN Sales.OrderLines od ON oh.OrderID = od.Or
ORDER BY
161 %
Results
Messages
Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT oh.CustomerID, oh.CustomerPurchaseOrderNumber, od.Quantity, od.UnitPrice FROM Sales.Order
SELECT
Cost: 0 %
Parallelism
(Gather Streams)
Cost: 6 %
1.083s
231412 of
241673 (95%)
Nested Loops
(Inner Join)
Cost: 0 %
0.198s
231412 of
241673 (95%)
Nested Loops
(Inner Join)
Cost: 0 %
0.092s
231412 of
241673 (95%)
Query executed successfully.
(local)
Ready
Properties
SELECT
Misc
Cached plan size 56 KB
CardinalityEstimationModelVersion 150
CompileCPU 2
CompileMemory 400
CompileTime 2
DatabaseContextSettingsId 2
Degree of Parallelism 8
Estimated Number of Rows 241673
Estimated Operator Cost 0 (0%)
Estimated Subtree Cost 292.459
Memory Grant 12224
MemoryGrantInfo
DesiredMemory 12224
GrantedMemory 12224
GrantWaitTime 0
IsMemoryGrantFeedbackAdjusted YesAdjusting
LastRequestedMemory 417984
MaxQueryMemory 1374128
MaxUsedMemory 5912
RequestedMemory 12224
RequiredMemory 4864
SerialDesiredMemory 7840
SerialRequiredMemory 512
Optimization Level FULL
OptimizerHardwareDependentPrope
OptimizerStatsUsage
MemoryGrantInfo


CREATE PROCEDURE GetOrders AS
BEGIN
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
od.Quantity,
od.UnitPrice
FROM
Sales.Orders oh
INNER JOIN Sales.OrderLines od ON oh.OrderID = od.Or
ORDER BY
Properties
SELECT
Misc
Cached plan size 56 KB
CardinalityEstimationModelVersion 150
CompileCPU 4
CompileMemory 400
CompileTime 4
DatabaseContextSettingsId 2
Degree of Parallelism 8
Estimated Number of Rows 241673
Estimated Operator Cost 0 (0%)
Estimated Subtree Cost 292.459
Memory Grant 12224
MemoryGrantInfo
DesiredMemory 12224
GrantedMemory 12224
GrantWaitTime 0
IsMemoryGrantFeedbackAdjusted YesStable
LastRequestedMemory 12224
MaxQueryMemory 1370960
MaxUsedMemory 5920
RequestedMemory 12224
RequiredMemory 4864
SerialDesiredMemory 7840
SerialRequiredMemory 512
Optimization Level FULL
OptimizerHardwareDependentPrope
OptimizerStatsUsage
MemoryGrantInfo
Query executed successfully.
Ready


CREATE PROCEDURE GetOrders AS
BEGIN
SELECT
oh.CustomerID,
oh.CustomerPurchaseOrderNumber,
FROM
ON oh.OrderID = od.OrderID
ORDER
Parallelism
Gather streams.
Physical Operation Parallelism
Logical Operation Gather Streams
Actual Execution Mode Row
Estimated Execution Mode Row
Actual Number of Rows 231412
Actual Number of Batches 0
Estimated I/O Cost 0
Estimated Operator Cost 18.405 (6%)
Estimated Subtree Cost 292.459
Estimated CPU Cost 18.4049
Number of Executions 1
Estimated Number of Executions 1
Estimated Number of Rows 241673
Estimated Row Size 4074 B
Actual Rebinds 0
Actual Rewinds 0
Node ID 0
Output List
[WideWorldImporters].[Sales].[Orders].CustomerID,
[WideWorldImporters].[Sales].[Orders].CustomerPurchaseOrderNumber,
[WideWorldImporters].[Sales].[Orders].Comments,
[WideWorldImporters].[Sales].[OrderLines].Quantity,
[WideWorldImporters].[Sales].[OrderLines].UnitPrice
Order By
[WideWorldImporters].[Sales].[Orders].Comments
Ascending
Sort
Cost: 71 %
0.027s
73595 of
73595 (100%)
Clustered Index Scan (Clustered)
[Orders].[PK_Sales_Orders] [oh]
Cost: 0 %
0.012s
73595 of
73595 (100%)
Index Seek (NonClustered)
[OrderLines].[FK_Sales_OrderLines_0...
Cost: 4 %
0.057s
231412 of
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (51) WideWorldImporters 00:00:01 231,412 rows
Ready

More Intelligent Query Processing

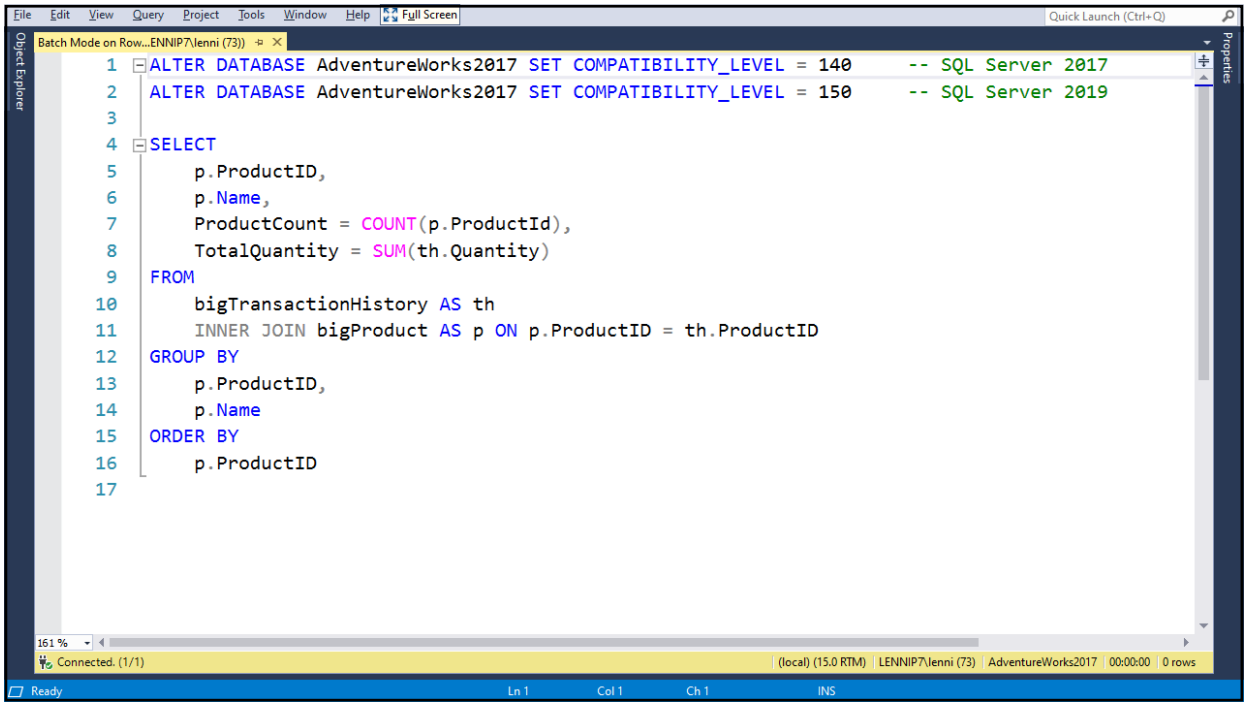
- SQL Server 2019 also adds
  - Batch mode on rowstore
  - Table variable deferred compilation
  - T-SQL Scalar UDF inlining
  - Approximate count distinct

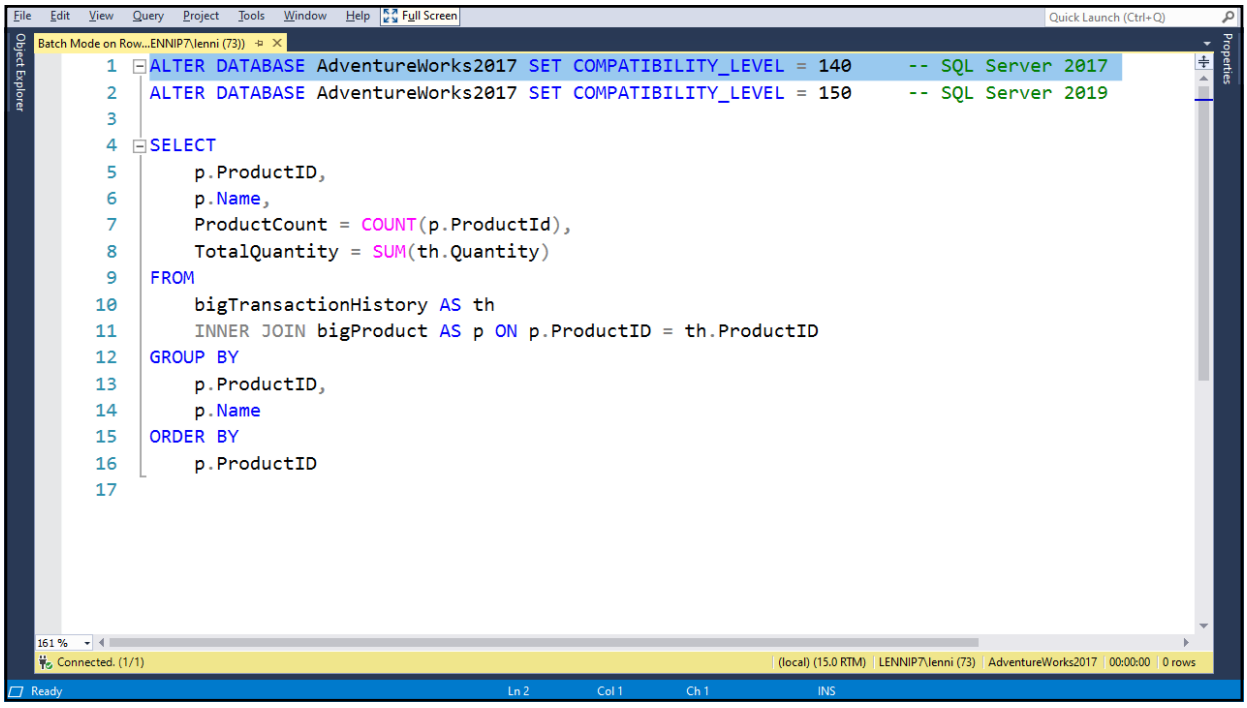
Batch Mode on Rowstore

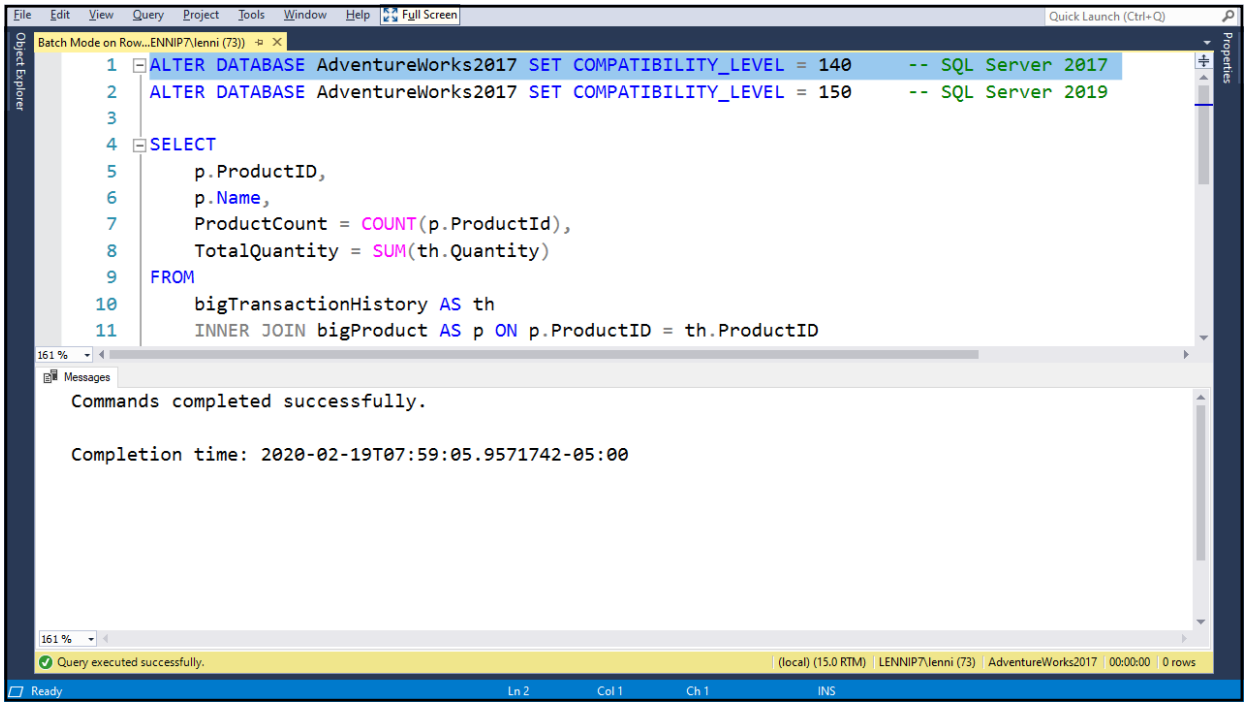
- What is Batch Mode?
  - Process multiple rows at a time
- Available since SQL Server 2012
  - But only on queries using tables with columnstore indexes
- In SQL Server 2019
  - Optimizer detects if the query qualifies for batch mode on a rowstore
    - Large number of rows with aggregations

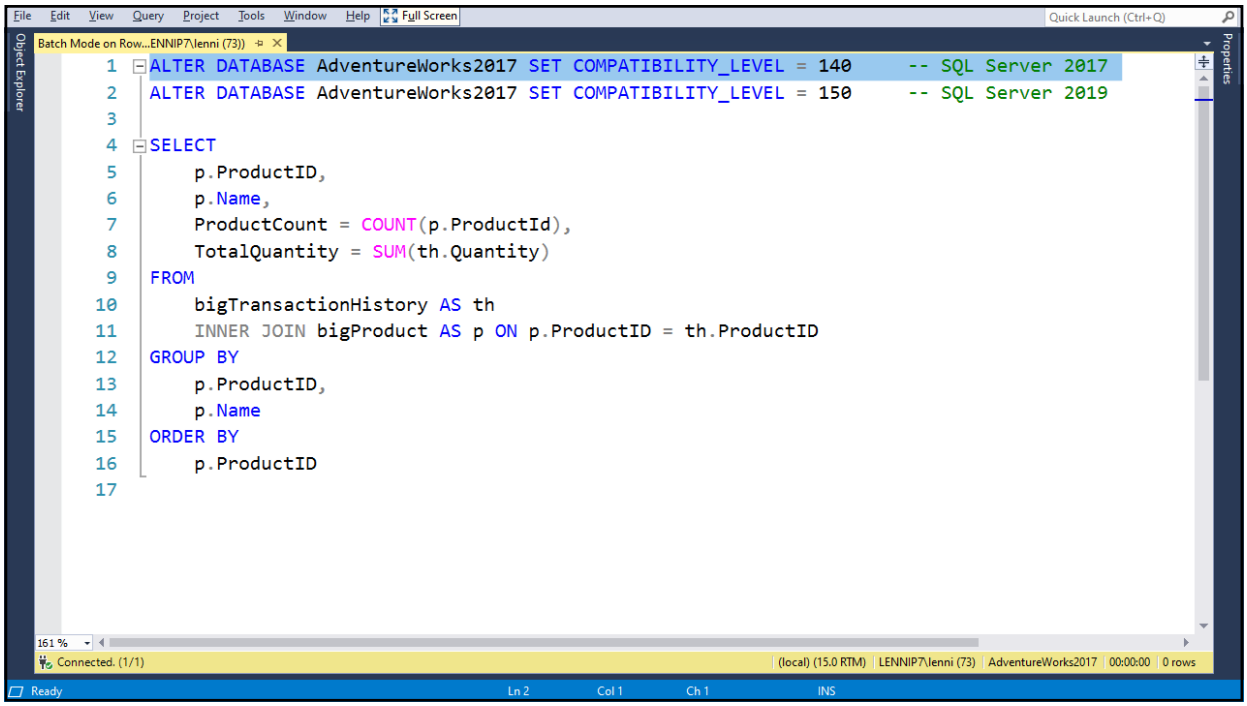
Batch Mode on Rowstore

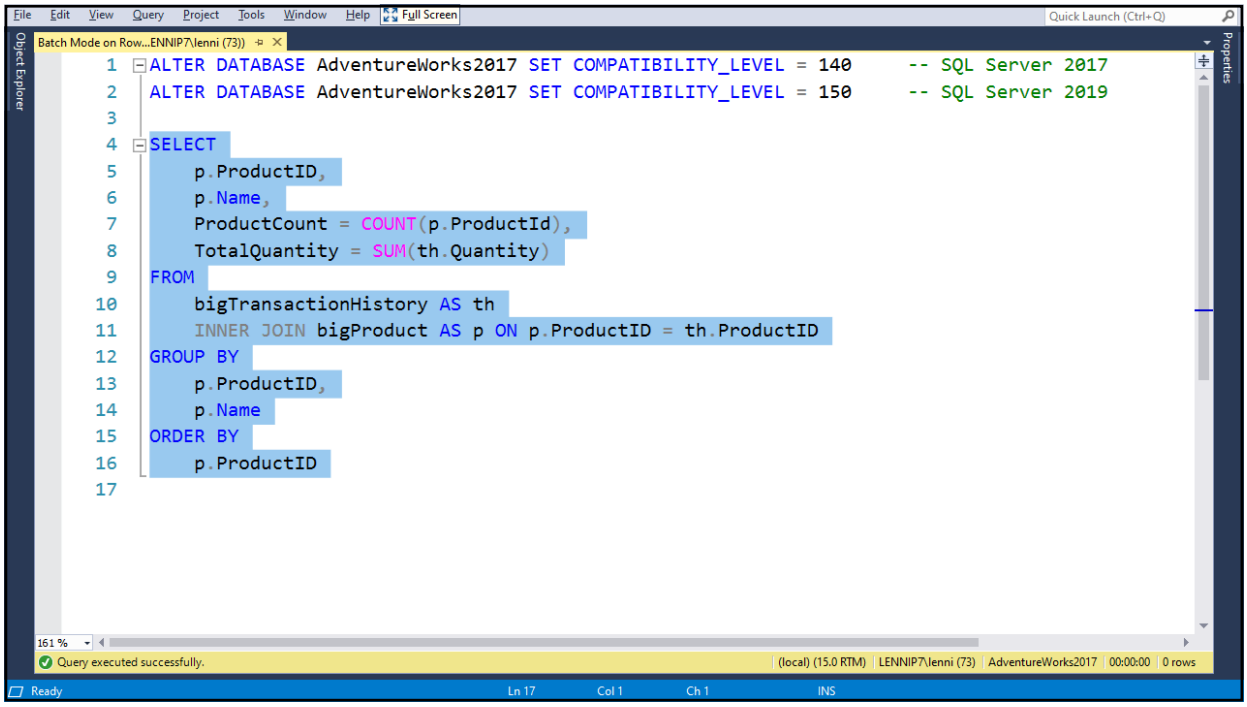
demo

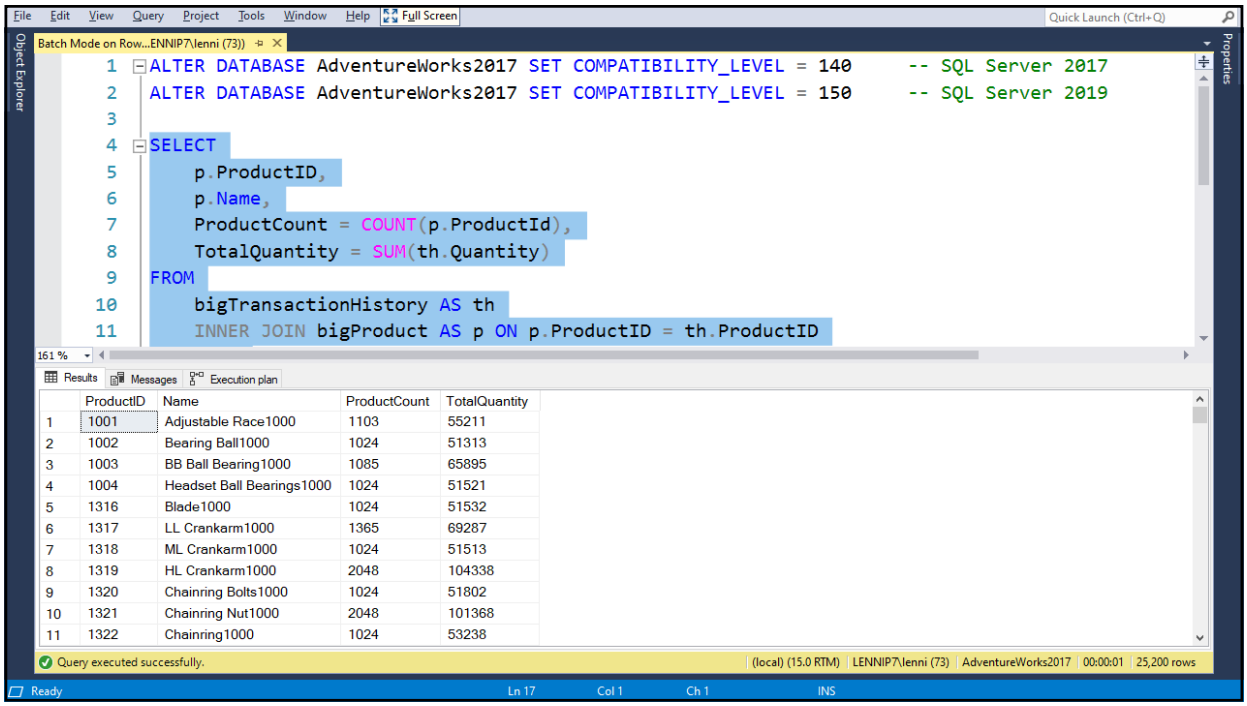


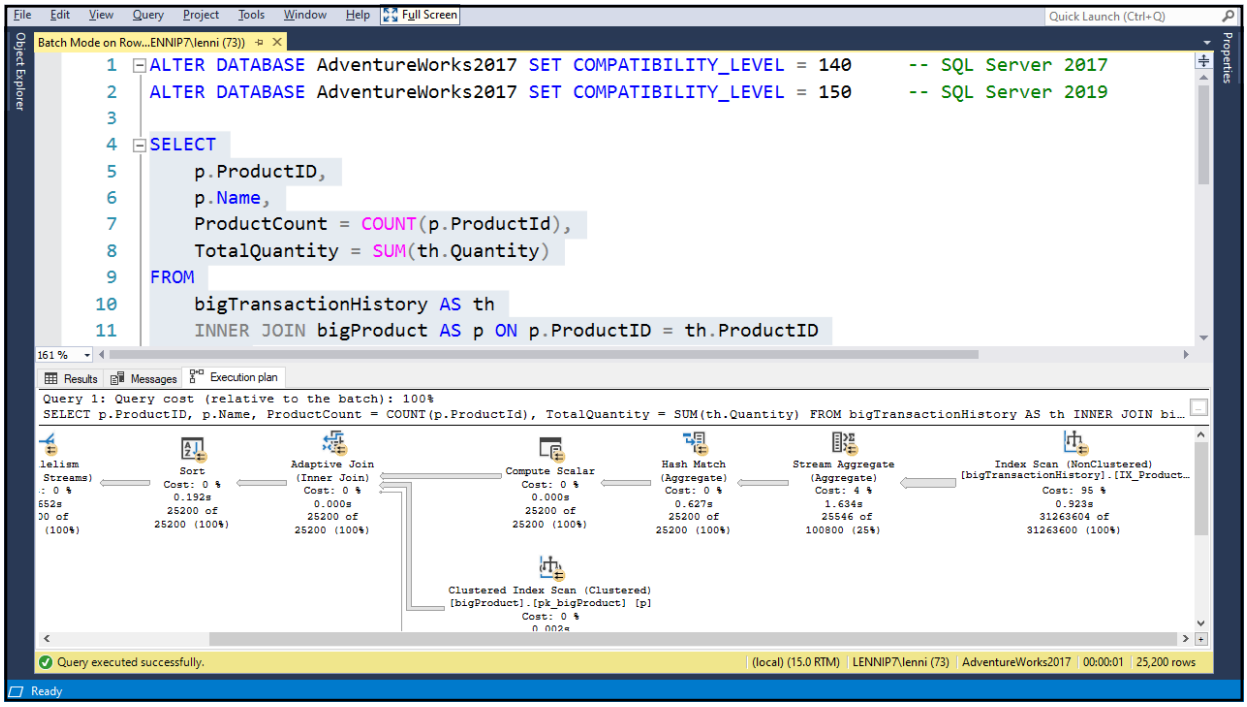

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
SELECT
p.ProductID,
p.Name,
ProductCount = COUNT(p.ProductId),
TotalQuantity = SUM(th.Quantity)
FROM
bigTransactionHistory AS th
INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
GROUP BY
p.ProductID,
p.Name
ORDER BY
p.ProductID
161 %
Connected. (1/1)
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:00 0 rows
Ready Ln 2 Col 1 Ch 1 INS

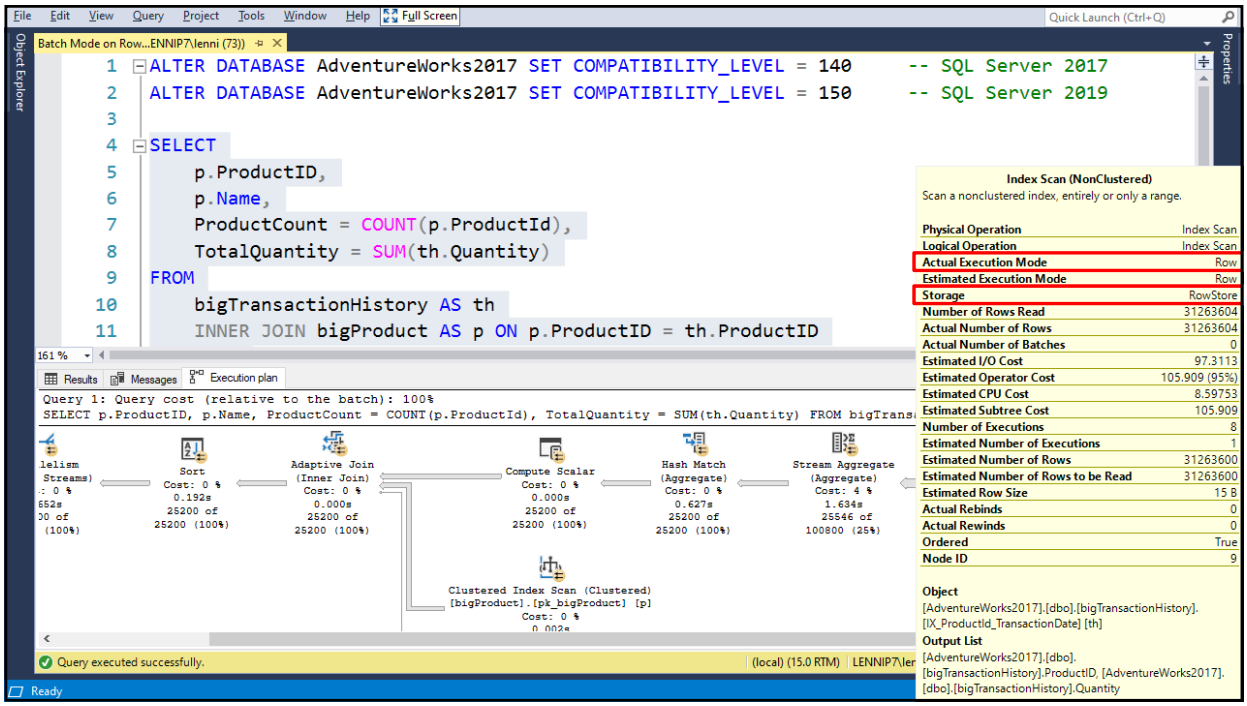

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
SELECT
p.ProductID,
p.Name,
ProductCount = COUNT(p.ProductId),
TotalQuantity = SUM(th.Quantity)
FROM
bigTransactionHistory AS th
INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
161 %
Messages
Commands completed successfully.
Completion time: 2020-02-19T07:59:05.9571742-05:00
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:00 0 rows
Ready Ln 2 Col 1 Ch 1 INS

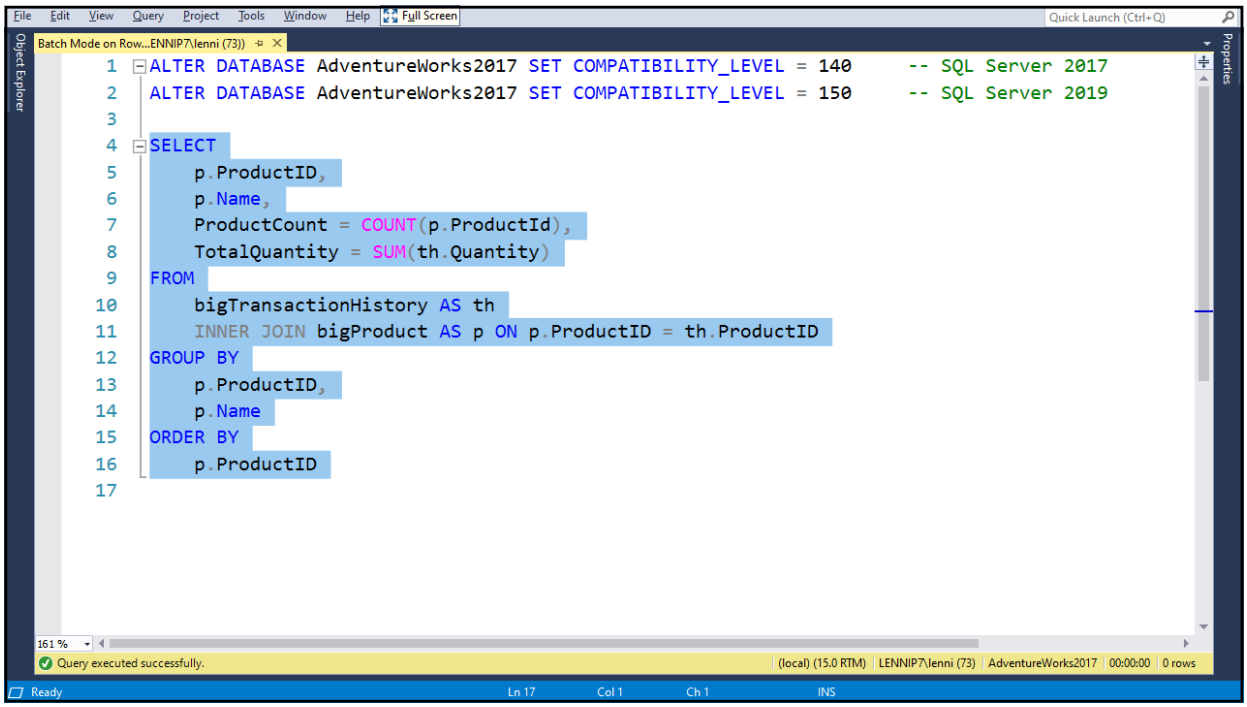

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
12 GROUP BY
13 p.ProductID,
14 p.Name
15 ORDER BY
16 p.ProductID
17
161 %
Connected. (1/1)
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:00 0 rows
Ready Ln 2 Col 1 Ch 1 INS

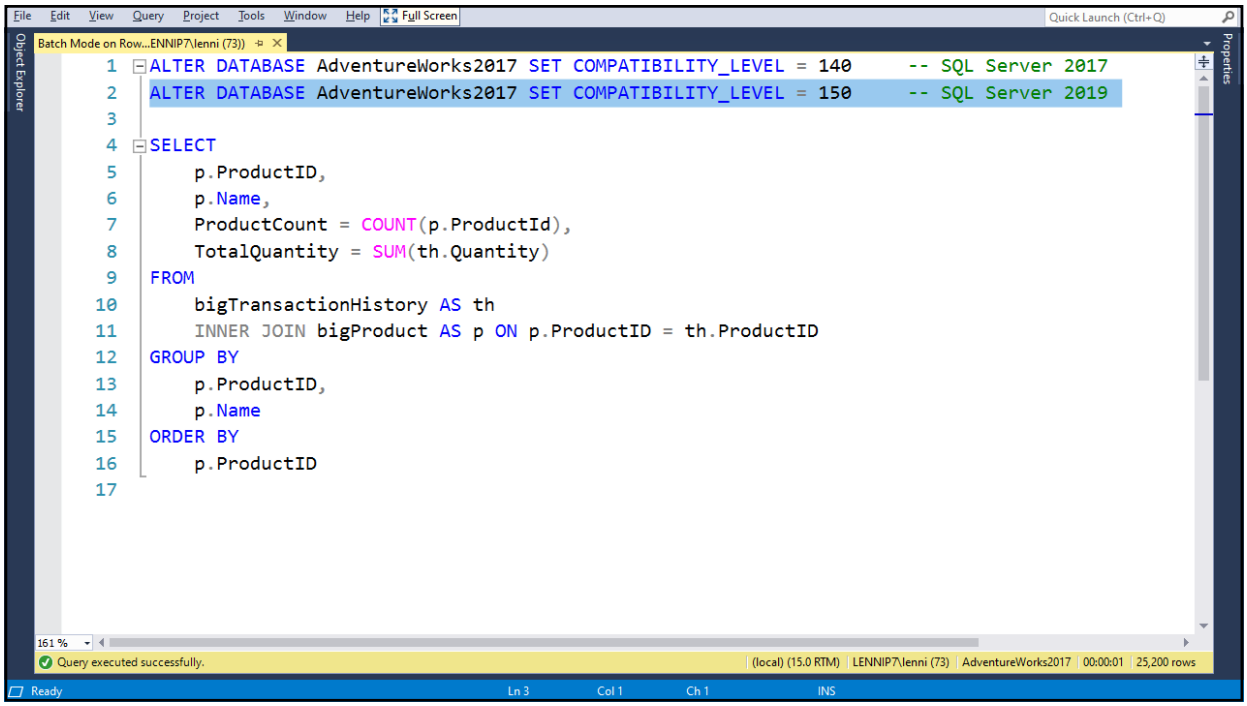

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
12 GROUP BY
13 p.ProductID,
14 p.Name
15 ORDER BY
16 p.ProductID
17
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:00 0 rows
Ready Ln 17 Col 1 Ch 1 INS

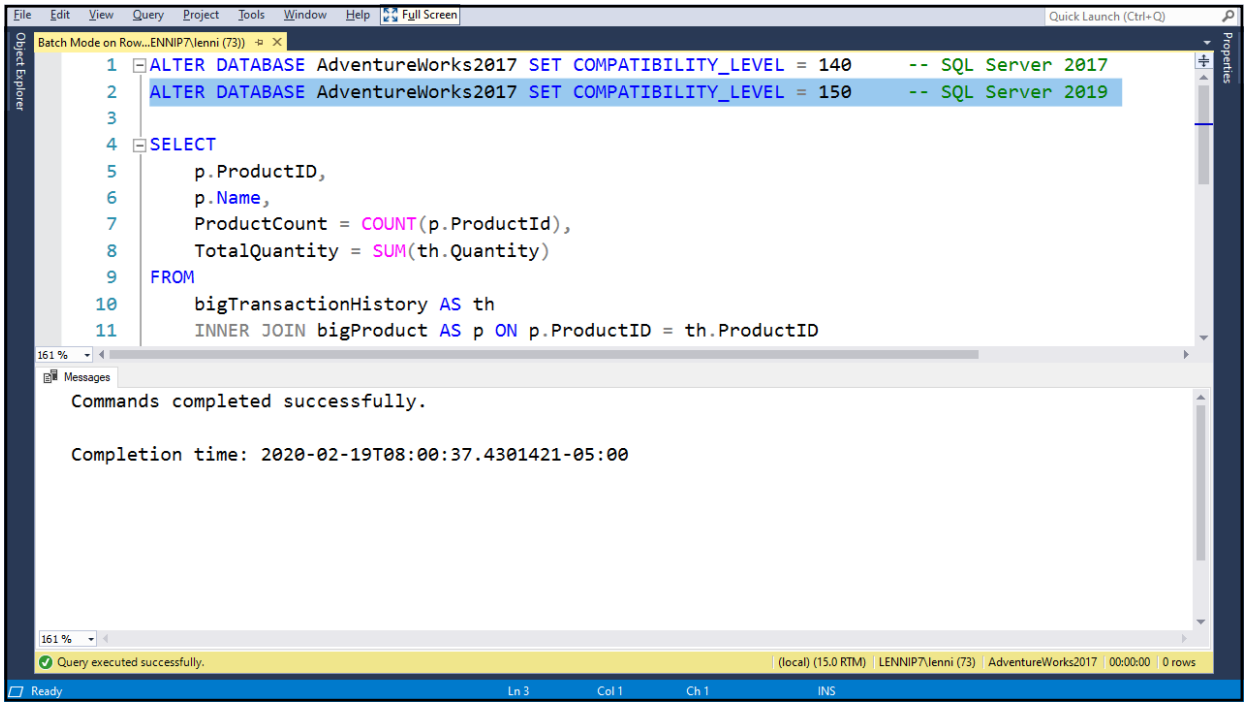

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
161 %
Results Messages Execution plan
ProductID Name ProductCount TotalQuantity
1 1001 Adjustable Race1000 1103 55211
2 1002 Bearing Ball1000 1024 51313
3 1003 BB Ball Bearing1000 1085 65895
4 1004 Headset Ball Bearings1000 1024 51521
5 1316 Blade1000 1024 51532
6 1317 LL Crankarm1000 1365 69287
7 1318 ML Crankarm1000 1024 51513
8 1319 HL Crankarm1000 2048 104338
9 1320 Chainring Bolts1000 1024 51802
10 1321 Chainring Nut1000 2048 101368
11 1322 Chainring1000 1024 53238
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:01 25,200 rows
Ready Ln 17 Col 1 Ch 1 INS

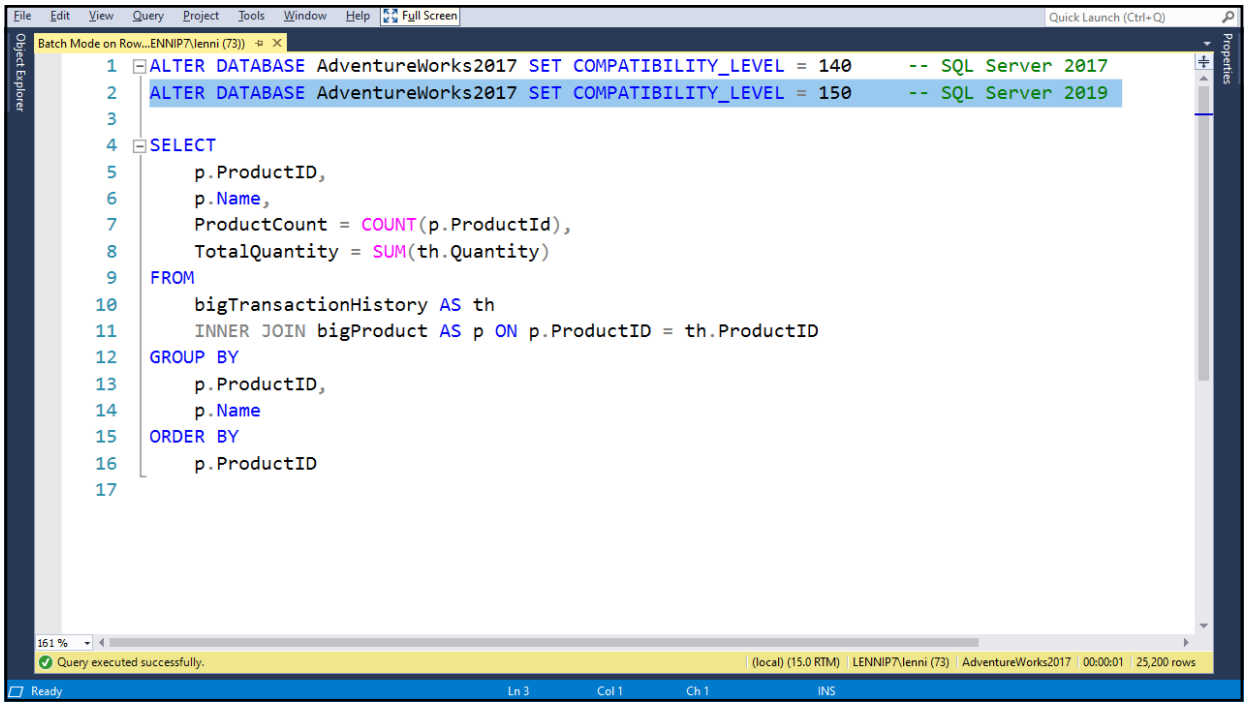

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
161 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT p.ProductID, p.Name, ProductCount = COUNT(p.ProductId), TotalQuantity = SUM(th.Quantity) FROM bigTransactionHistory AS th INNER JOIN bi...
Sort
Cost: 0 %
0.192s
25200 of
25200 (100%)
Adaptive Join
(Inner Join)
Cost: 0 %
0.000s
25200 of
25200 (100%)
Compute Scalar
Cost: 0 %
0.000s
25200 of
25200 (100%)
Hash Match
(Aggregate)
Cost: 0 %
0.627s
25200 of
25200 (100%)
Stream Aggregate
(Aggregate)
Cost: 4 %
1.634s
25546 of
100800 (25%)
Index Scan (NonClustered)
[bigTransactionHistory].[IX_Product...
Cost: 95 %
0.923s
31263604 of
31263600 (100%)
Clustered Index Scan (Clustered)
[bigProduct].[pk_bigProduct] [p]
Cost: 0 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:01 25,200 rows
Ready

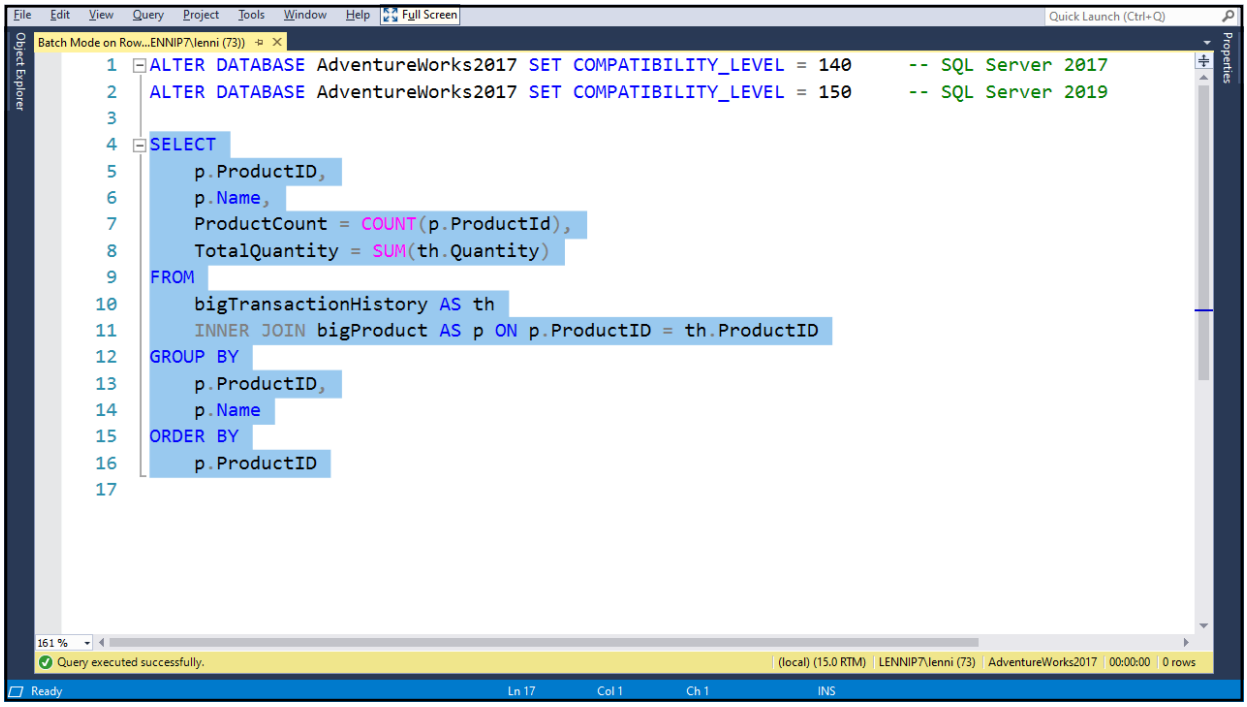

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
161 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT p.ProductID, p.Name, ProductCount = COUNT(p.ProductId), TotalQuantity = SUM(th.Quantity) FROM bigTrans
Sort
Cost: 0 %
0.192s
25200 of
25200 (100%)
Adaptive Join
(Inner Join)
Cost: 0 %
0.000s
25200 of
25200 (100%)
Compute Scalar
Cost: 0 %
0.000s
25200 of
25200 (100%)
Hash Match
(Aggregate)
Cost: 0 %
0.627s
25200 of
25200 (100%)
Stream Aggregate
(Aggregate)
Cost: 4 %
1.634s
25546 of
100800 (25%)
Clustered Index Scan (Clustered)
[bigProduct].[pk_bigProduct] [p]
Cost: 0 %
Query executed successfully.
(local) (15.0 RTM)
Ready
Index Scan (NonClustered)
Scan a nonclustered index, entirely or only a range.
Physical Operation Index Scan
Logical Operation Index Scan
Actual Execution Mode Row
Estimated Execution Mode Row
Storage RowStore
Number of Rows Read 31263604
Actual Number of Rows 31263604
Actual Number of Batches 0
Estimated I/O Cost 97.3113
Estimated Operator Cost 105.909 (95%)
Estimated CPU Cost 8.59753
Estimated Subtree Cost 105.909
Number of Executions 8
Estimated Number of Executions 1
Estimated Number of Rows 31263600
Estimated Number of Rows to be Read 31263600
Estimated Row Size 15 B
Actual Rebinds 0
Actual Rewinds 0
Ordered True
Node ID 9
Object
[AdventureWorks2017].[dbo].[bigTransactionHistory].
[IX_ProductId_TransactionDate] [th]
Output List
[AdventureWorks2017].[dbo].
[bigTransactionHistory].ProductID, [AdventureWorks2017].
[dbo].[bigTransactionHistory].Quantity

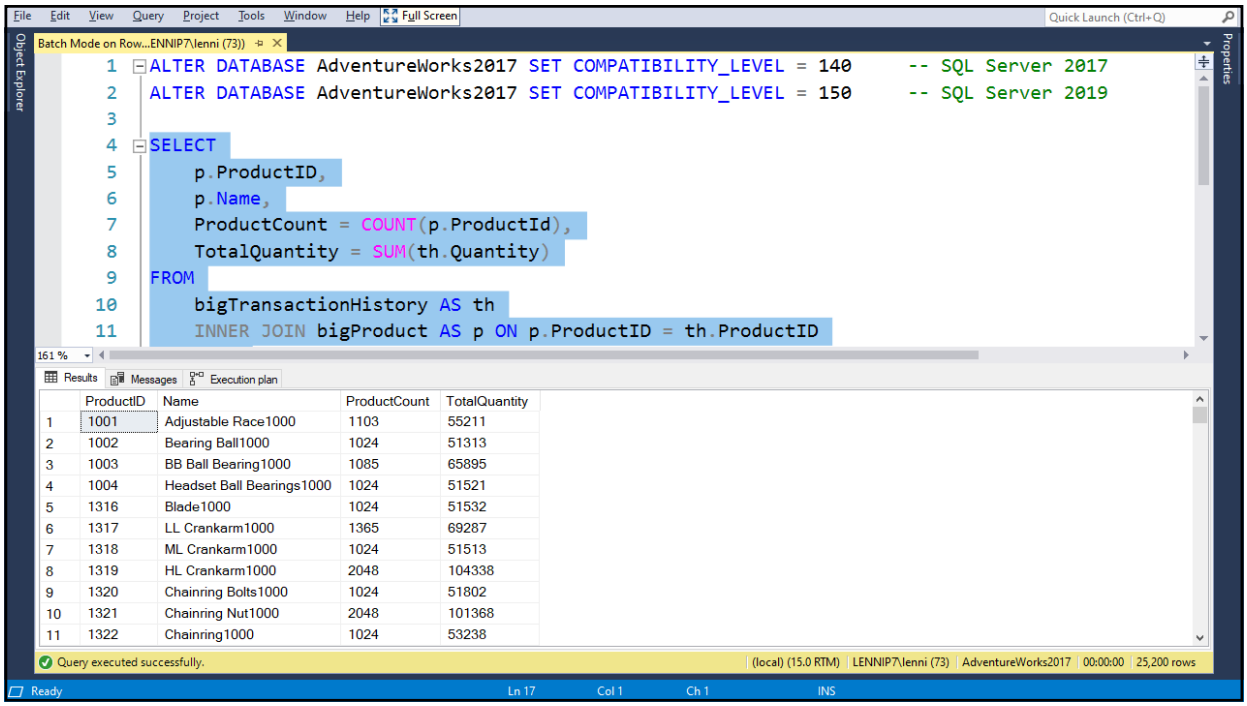

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
12 GROUP BY
13 p.ProductID,
14 p.Name
15 ORDER BY
16 p.ProductID
17
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:00 0 rows
Ready Ln 17 Col 1 Ch 1 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
12 GROUP BY
13 p.ProductID,
14 p.Name
15 ORDER BY
16 p.ProductID
17
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:01 25,200 rows
Ready Ln 3 Col 1 Ch 1 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
161 %
Messages
Commands completed successfully.
Completion time: 2020-02-19T08:00:37.4301421-05:00
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:00 0 rows
Ready Ln 3 Col 1 Ch 1 INS

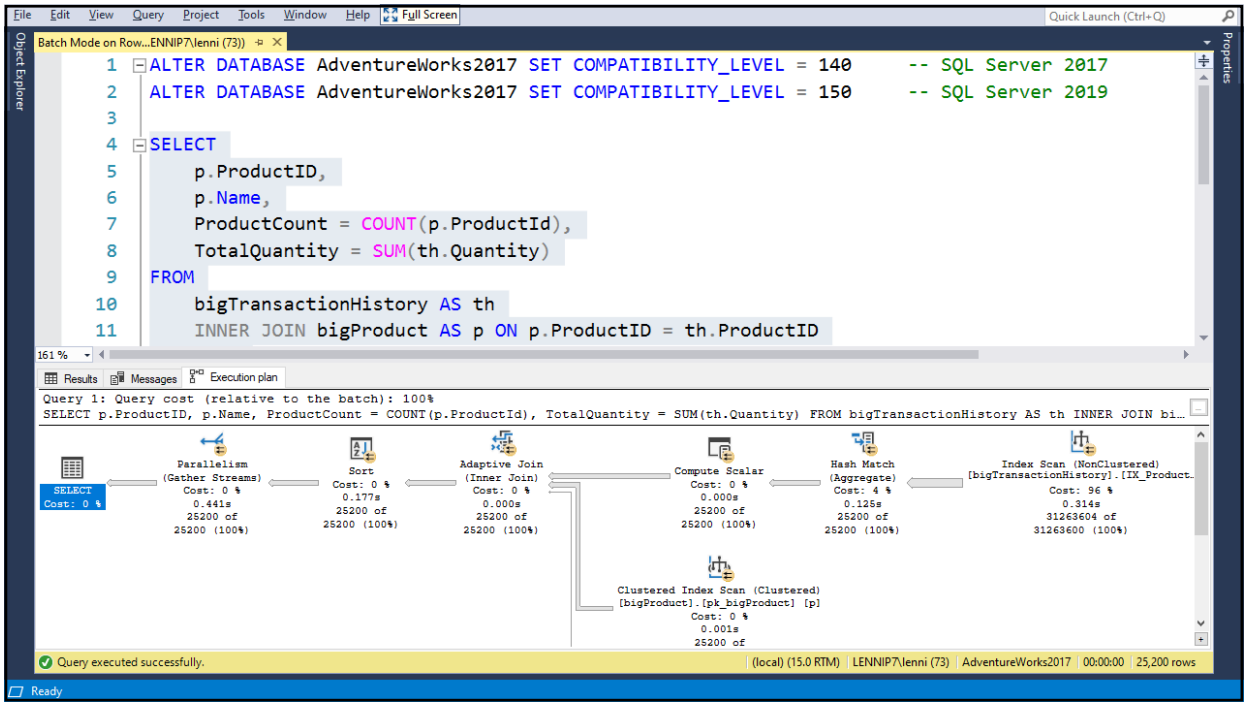

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
12 GROUP BY
13 p.ProductID,
14 p.Name
15 ORDER BY
16 p.ProductID
17
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:01 25,200 rows
Ready Ln 3 Col 1 Ch 1 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Batch Mode on Row...ENNIP7\lenni (73))
Object Explorer
Properties
1 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
2 ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
3
4 SELECT
5 p.ProductID,
6 p.Name,
7 ProductCount = COUNT(p.ProductId),
8 TotalQuantity = SUM(th.Quantity)
9 FROM
10 bigTransactionHistory AS th
11 INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
12 GROUP BY
13 p.ProductID,
14 p.Name
15 ORDER BY
16 p.ProductID
17
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (73) AdventureWorks2017 00:00:00 0 rows
Ready Ln 17 Col 1 Ch 1 INS



```
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140      -- SQL Server 2017
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150      -- SQL Server 2019

SELECT
    p.ProductID,
    p.Name,
    ProductCount = COUNT(p.ProductId),
    TotalQuantity = SUM(th.Quantity)
FROM
    bigTransactionHistory AS th
    INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
```

	ProductID	Name	ProductCount	TotalQuantity
1	1001	Adjustable Race1000	1103	55211
2	1002	Bearing Ball1000	1024	51313
3	1003	BB Ball Bearing1000	1085	65895
4	1004	Headset Ball Bearings1000	1024	51521
5	1316	Blade1000	1024	51532
6	1317	LL Crankarm1000	1365	69287
7	1318	ML Crankarm1000	1024	51513
8	1319	HL Crankarm1000	2048	104338
9	1320	Chainring Bolts1000	1024	51802
10	1321	Chainring Nut1000	2048	101368
11	1322	Chainring1000	1024	53238



```
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 140      -- SQL Server 2017
ALTER DATABASE AdventureWorks2017 SET COMPATIBILITY_LEVEL = 150      -- SQL Server 2019

SELECT
    p.ProductID,
    p.Name,
    ProductCount = COUNT(p.ProductId),
    TotalQuantity = SUM(th.Quantity)
FROM
    bigTransactionHistory AS th
    INNER JOIN bigProduct AS p ON p.ProductID = th.ProductID
```

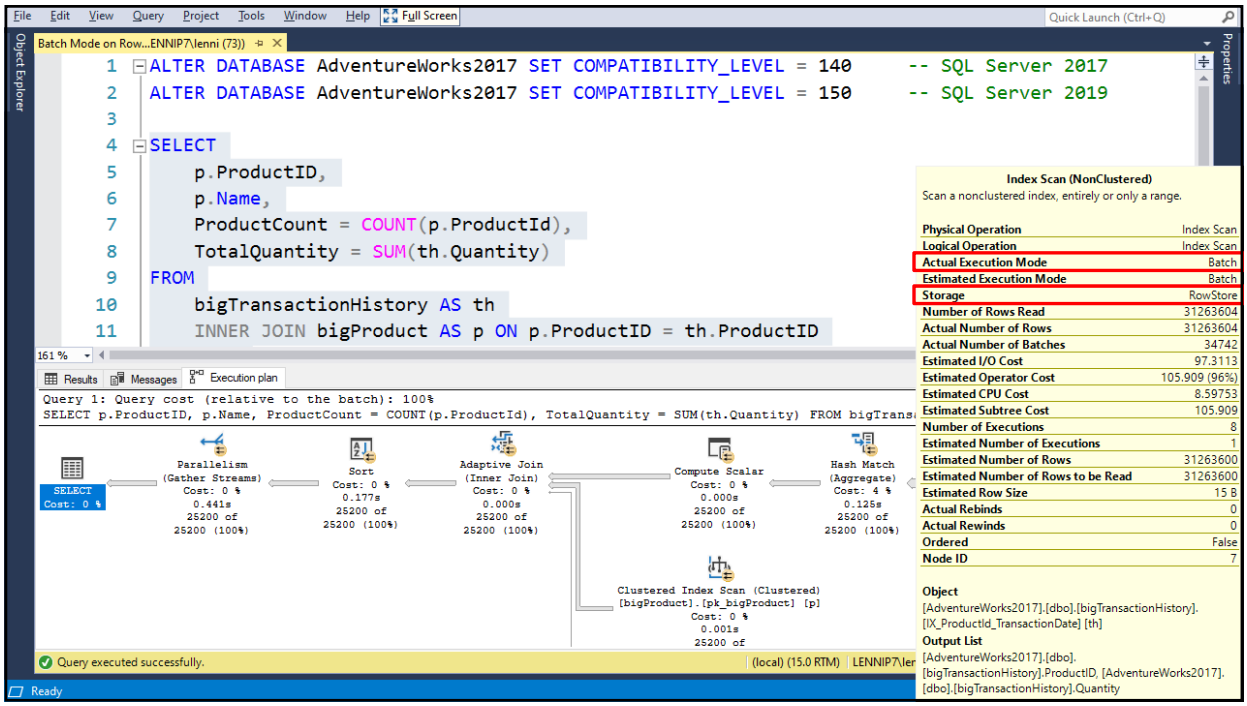
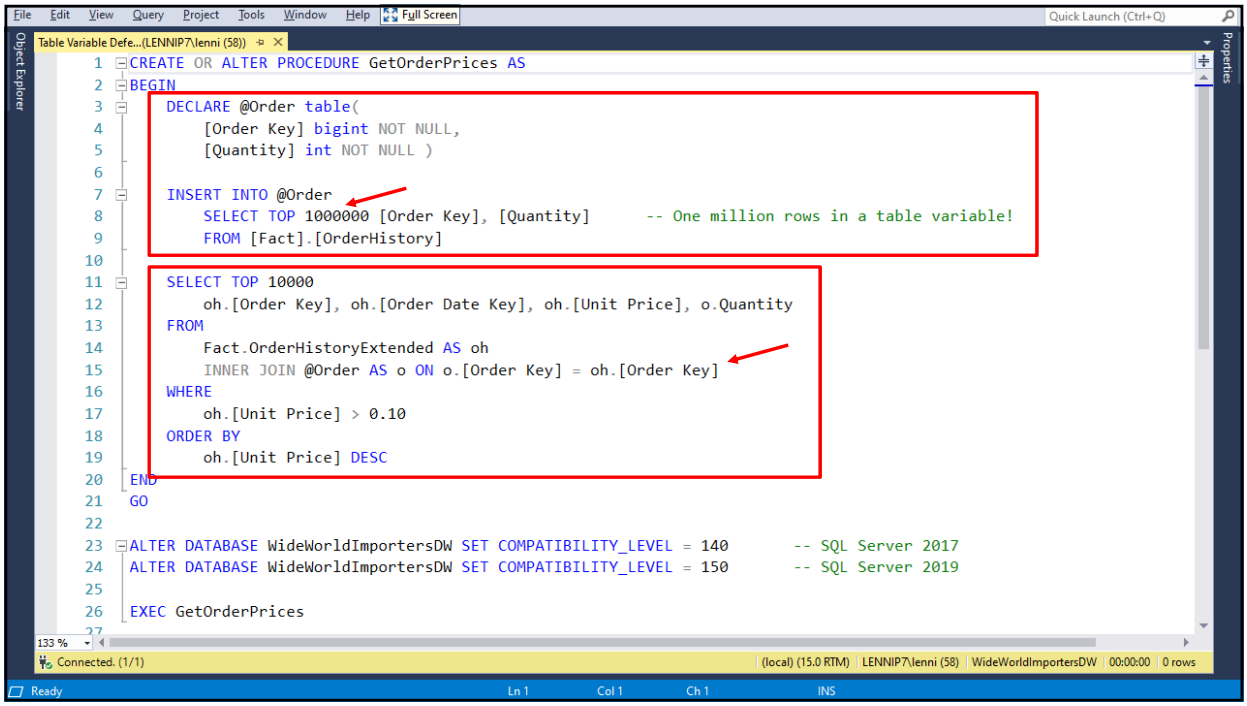


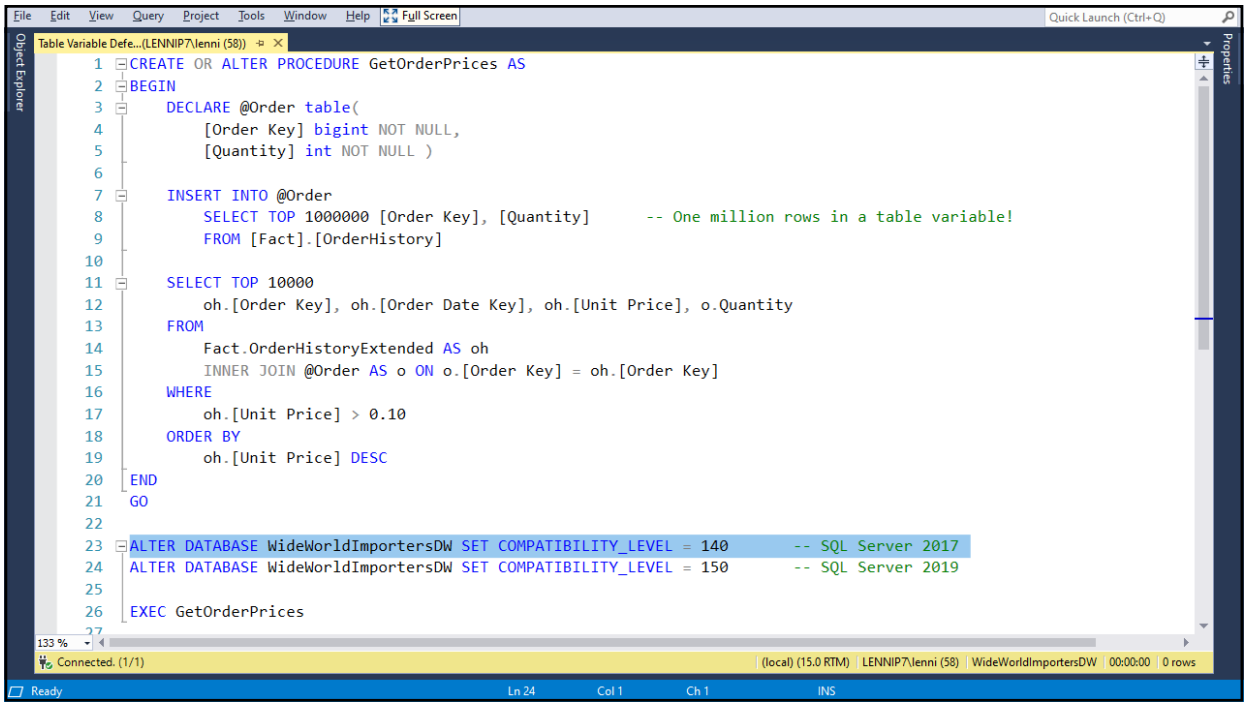
Table Variable Deferred Compilation

- Problem:
  - Historically, CE for table variables is ONE row
    - Optimizer doesn't know how many rows are actually in the table variable
  - Can be disastrous for very large table variables
- Solution:
  - Defer compilation of statements that reference table variables until the first execution
    - Optimizer can then make proper downstream decisions based on the correct cardinality
  - Same behavior as temp tables

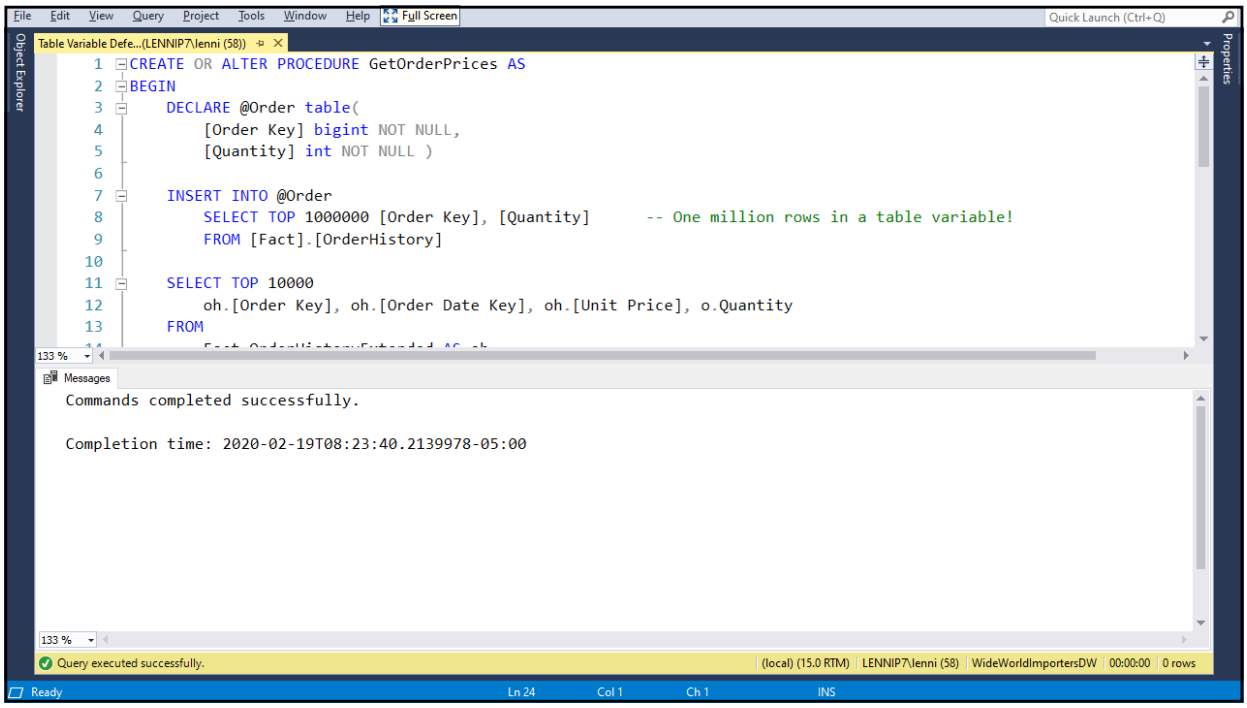
Table Variable Deferred Compilation

demo



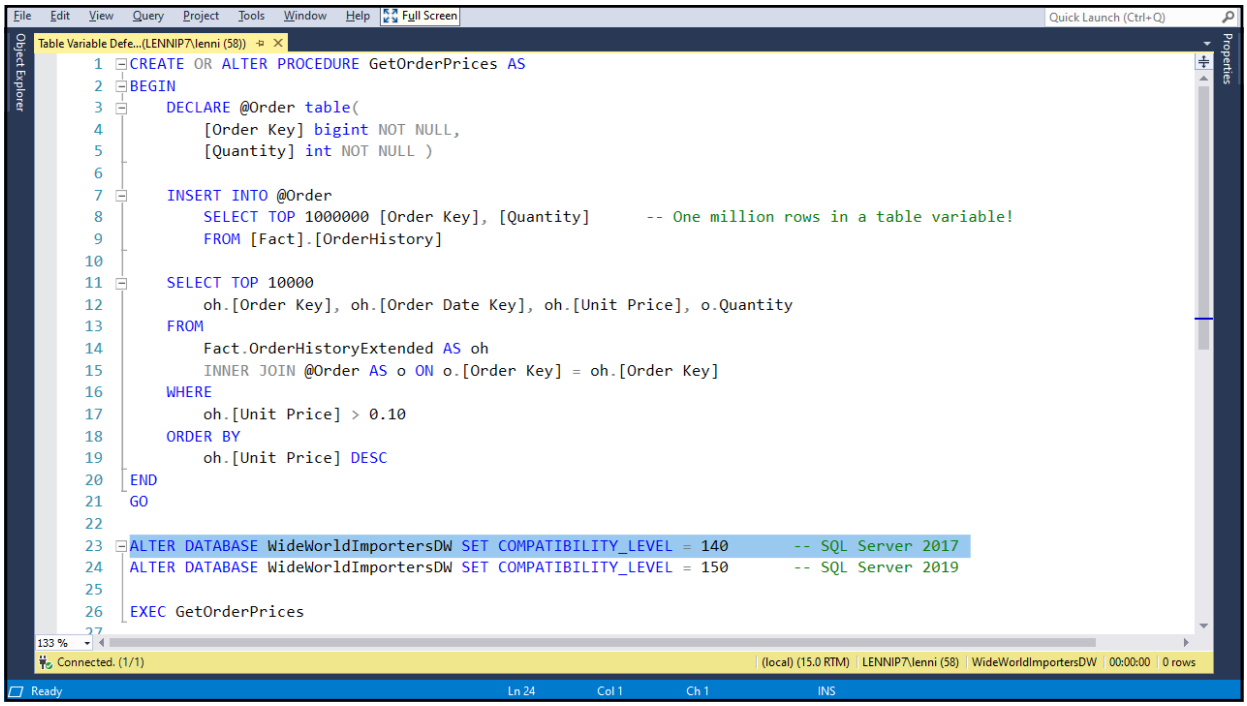


```
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
    DECLARE @Order table(
        [Order Key] bigint NOT NULL,
        [Quantity] int NOT NULL )

    INSERT INTO @Order
        SELECT TOP 1000000 [Order Key], [Quantity]      -- One million rows in a table variable!
        FROM [Fact].[OrderHistory]

    SELECT TOP 10000
        oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
    FROM
        Fact.OrderHistoryExtended AS oh
        INNER JOIN @Order AS o ON o.[Order Key] = oh.[Order Key]
    WHERE
        oh.[Unit Price] > 0.10
    ORDER BY
        oh.[Unit Price] DESC
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150       -- SQL Server 2019

EXEC GetOrderPrices
```

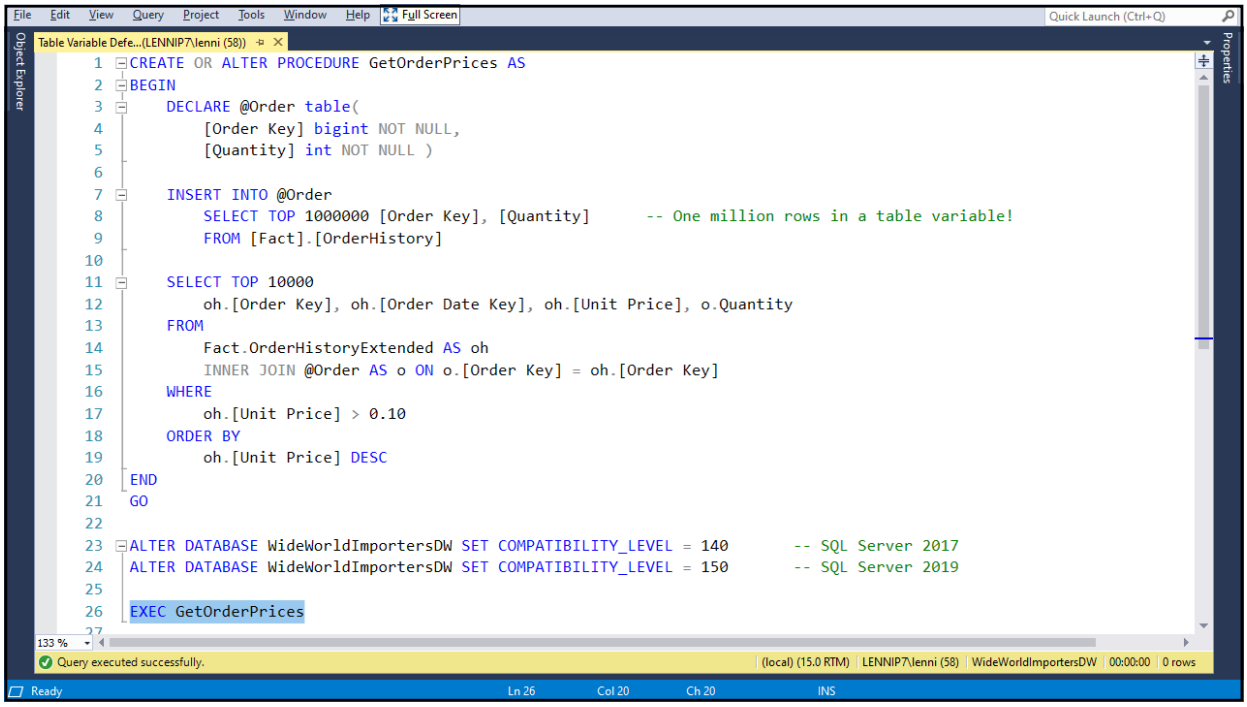


```
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
    DECLARE @Order table(
        [Order Key] bigint NOT NULL,
        [Quantity] int NOT NULL )

    INSERT INTO @Order
        SELECT TOP 1000000 [Order Key], [Quantity]      -- One million rows in a table variable!
        FROM [Fact].[OrderHistory]

    SELECT TOP 10000
        oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
    FROM
```

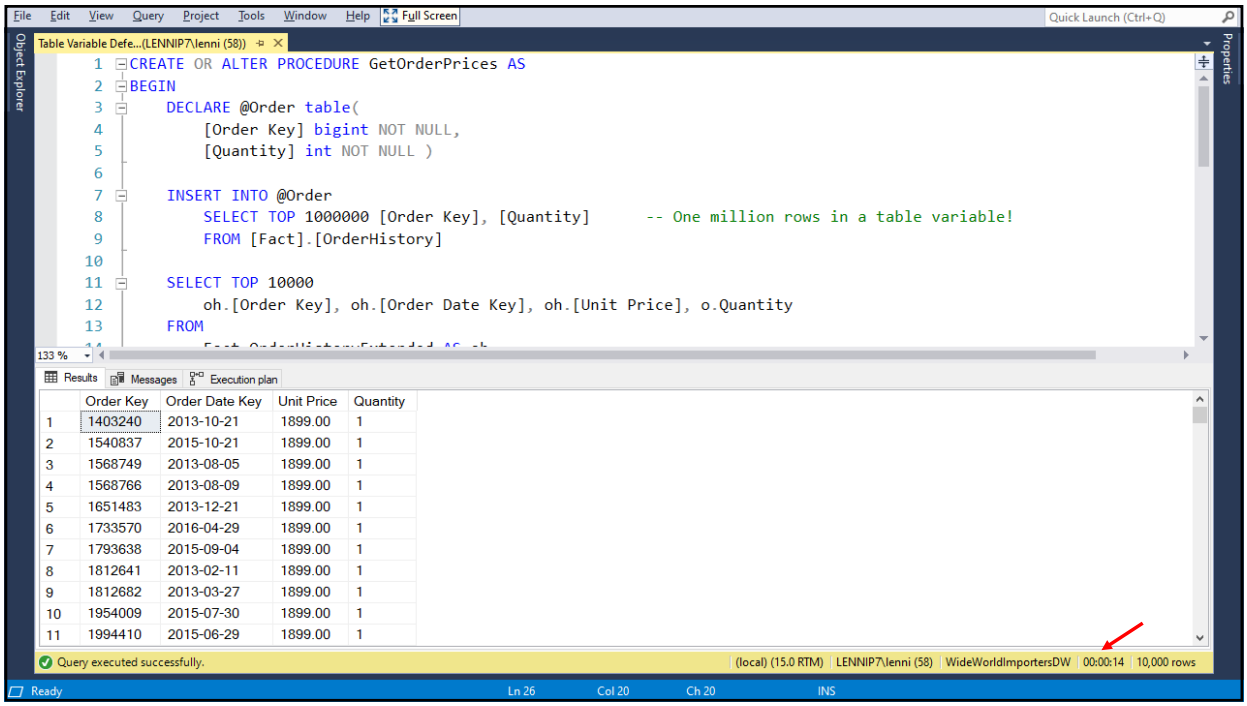
```
Commands completed successfully.

Completion time: 2020-02-19T08:23:40.2139978-05:00
```

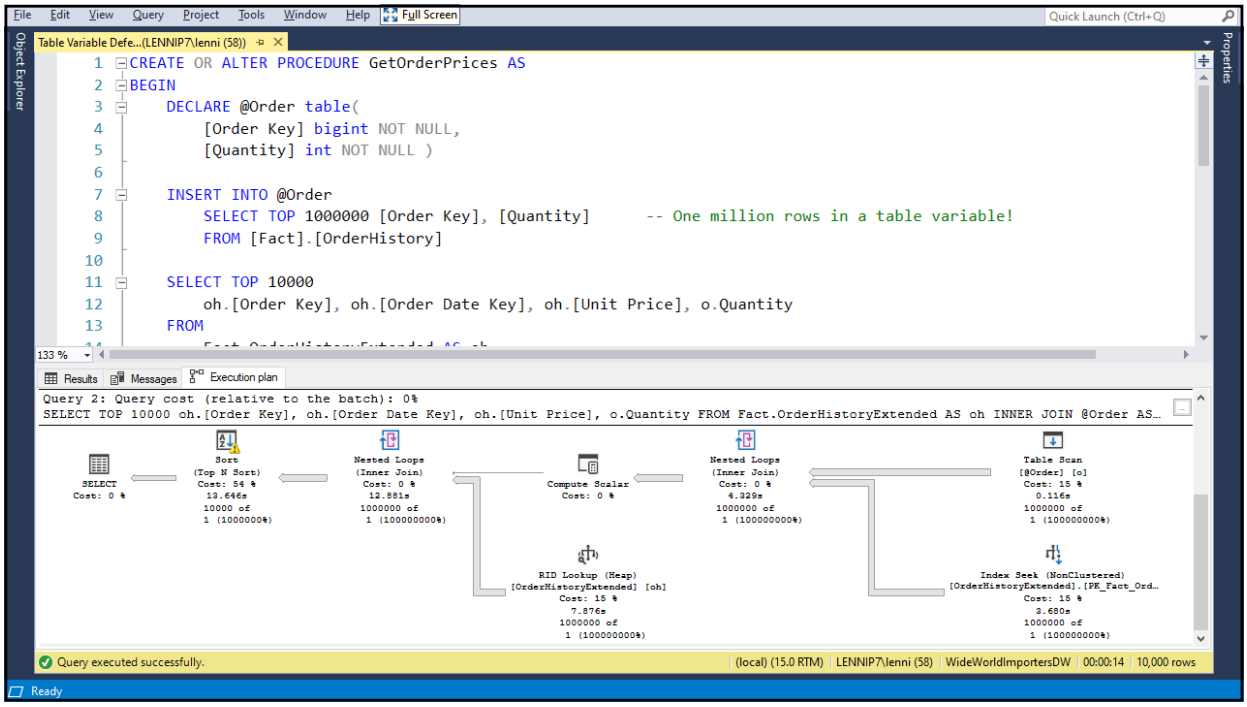


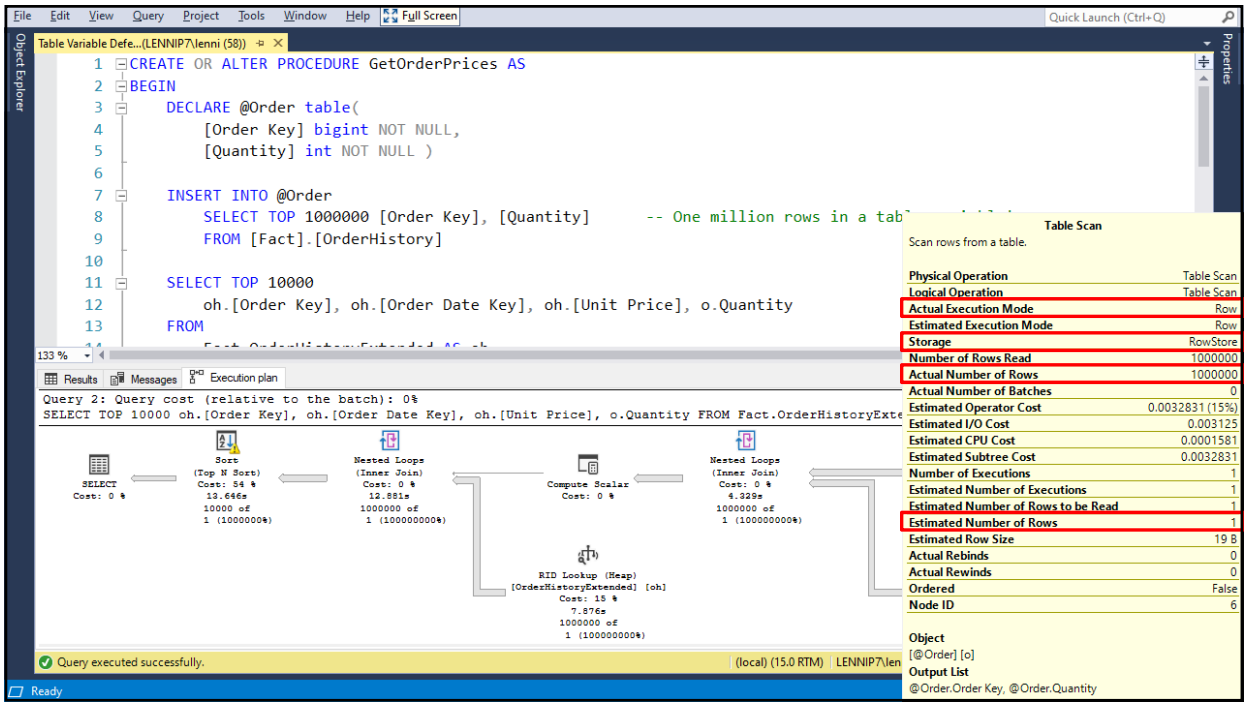
```
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
    DECLARE @Order table(
        [Order Key] bigint NOT NULL,
        [Quantity] int NOT NULL )

    INSERT INTO @Order
        SELECT TOP 1000000 [Order Key], [Quantity]      -- One million rows in a table variable!
        FROM [Fact].[OrderHistory]

    SELECT TOP 10000
        oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
    FROM
        Fact.OrderHistoryExtended AS oh
        INNER JOIN @Order AS o ON o.[Order Key] = oh.[Order Key]
    WHERE
        oh.[Unit Price] > 0.10
    ORDER BY
        oh.[Unit Price] DESC
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150       -- SQL Server 2019

EXEC GetOrderPrices
```

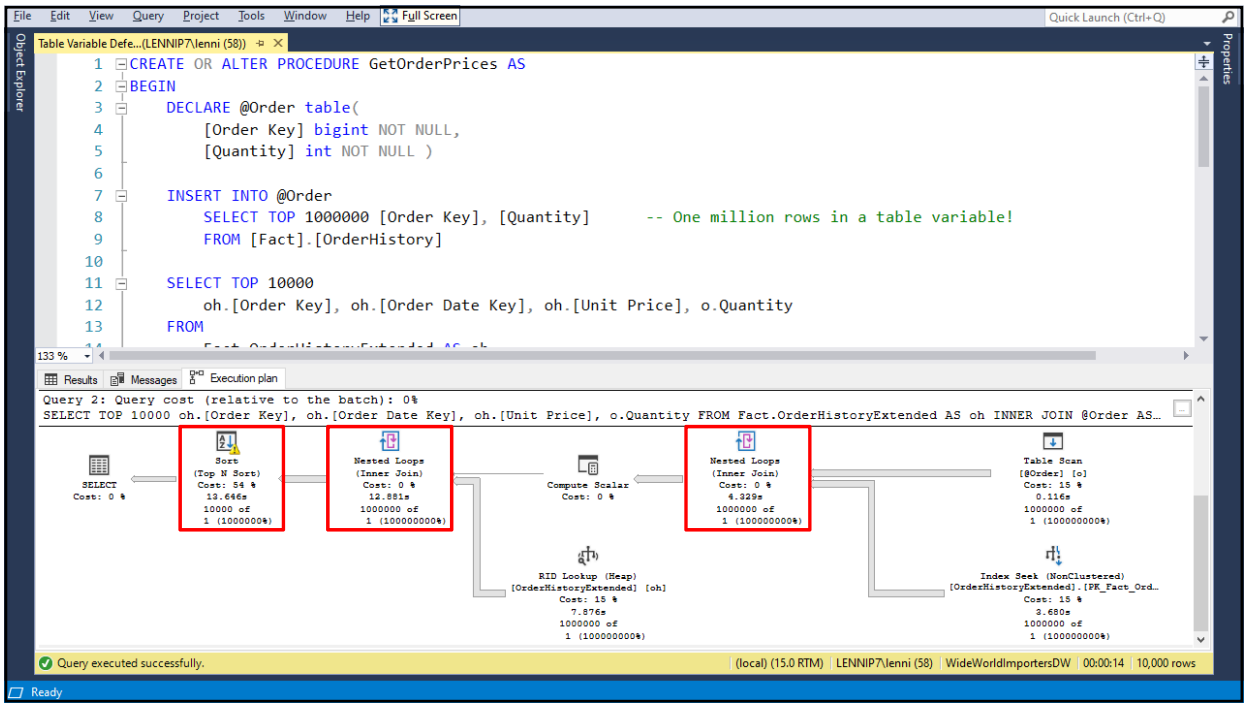


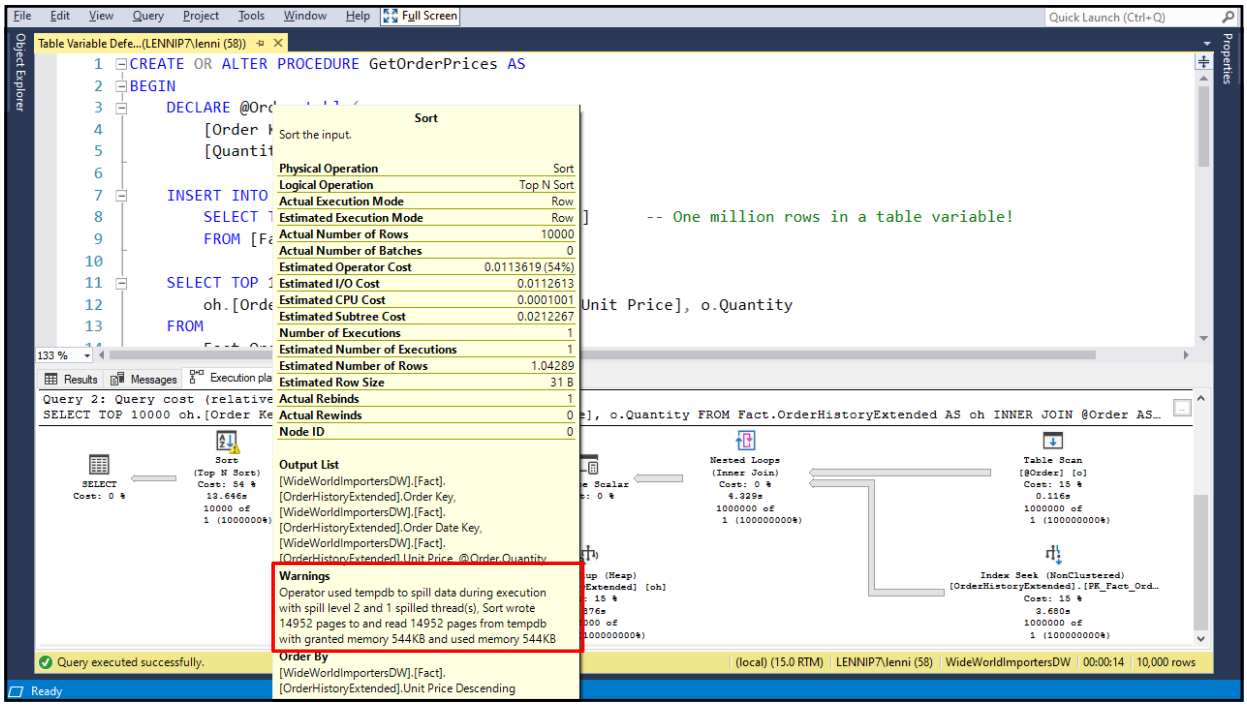
```
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
    DECLARE @Order table(
        [Order Key] bigint NOT NULL,
        [Quantity] int NOT NULL )

    INSERT INTO @Order
        SELECT TOP 1000000 [Order Key], [Quantity]      -- One million rows in a table variable!
        FROM [Fact].[OrderHistory]

    SELECT TOP 10000
        oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
    FROM
        Fact.OrderHistoryExtended AS oh
        INNER JOIN @Order AS o ON o.[Order Key] = oh.[Order Key]
    WHERE
        oh.[Unit Price] > 0.10
    ORDER BY
        oh.[Unit Price] DESC
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150       -- SQL Server 2019

EXEC GetOrderPrices
```

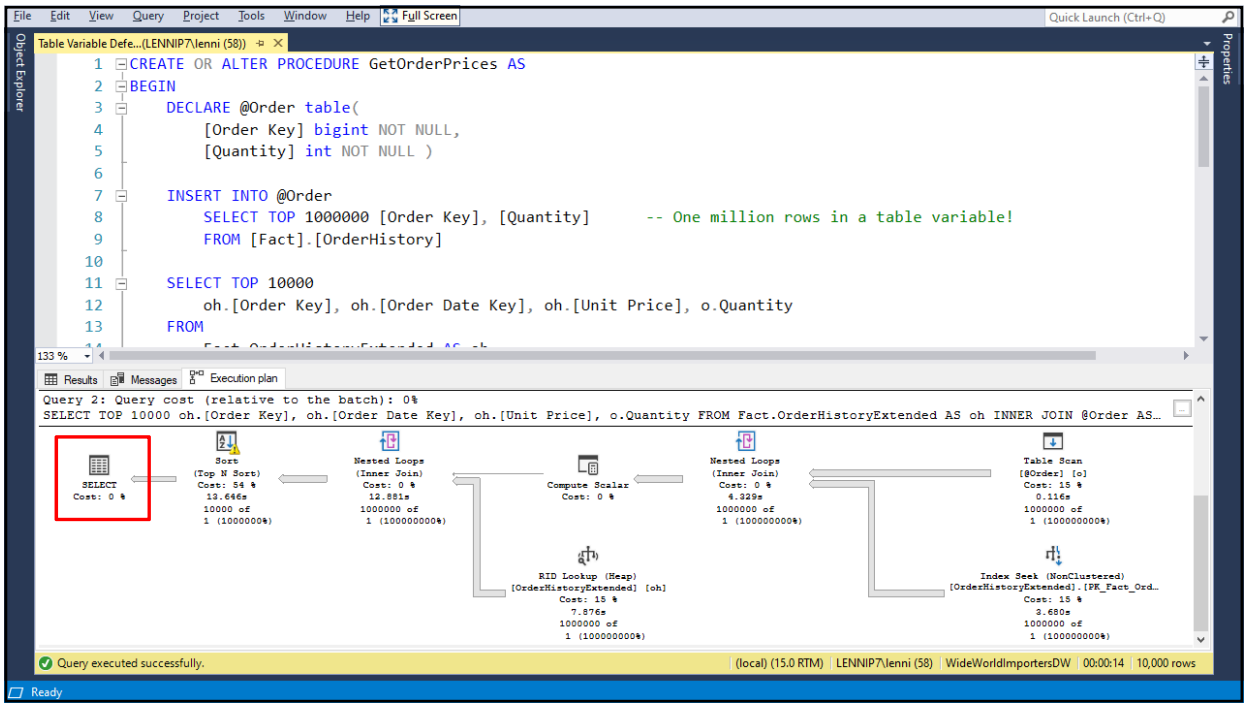

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
Object Explorer
Properties
1 CREATE OR ALTER PROCEDURE GetOrderPrices AS
2 BEGIN
3 DECLARE @Order table(
4 [Order Key] bigint NOT NULL,
5 [Quantity] int NOT NULL)
6
7 INSERT INTO @Order
8 SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
9 FROM [Fact].[OrderHistory]
10
11 SELECT TOP 10000
12 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
13 FROM
133 %
Results Messages Execution plan
Order Key Order Date Key Unit Price Quantity
1 1403240 2013-10-21 1899.00 1
2 1540837 2015-10-21 1899.00 1
3 1568749 2013-08-05 1899.00 1
4 1568766 2013-08-09 1899.00 1
5 1651483 2013-12-21 1899.00 1
6 1733570 2016-04-29 1899.00 1
7 1793638 2015-09-04 1899.00 1
8 1812641 2013-02-11 1899.00 1
9 1812682 2013-03-27 1899.00 1
10 1954009 2015-07-30 1899.00 1
11 1994410 2015-06-29 1899.00 1
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:14 10,000 rows
Ready Ln 26 Col 20 Ch 20 INS

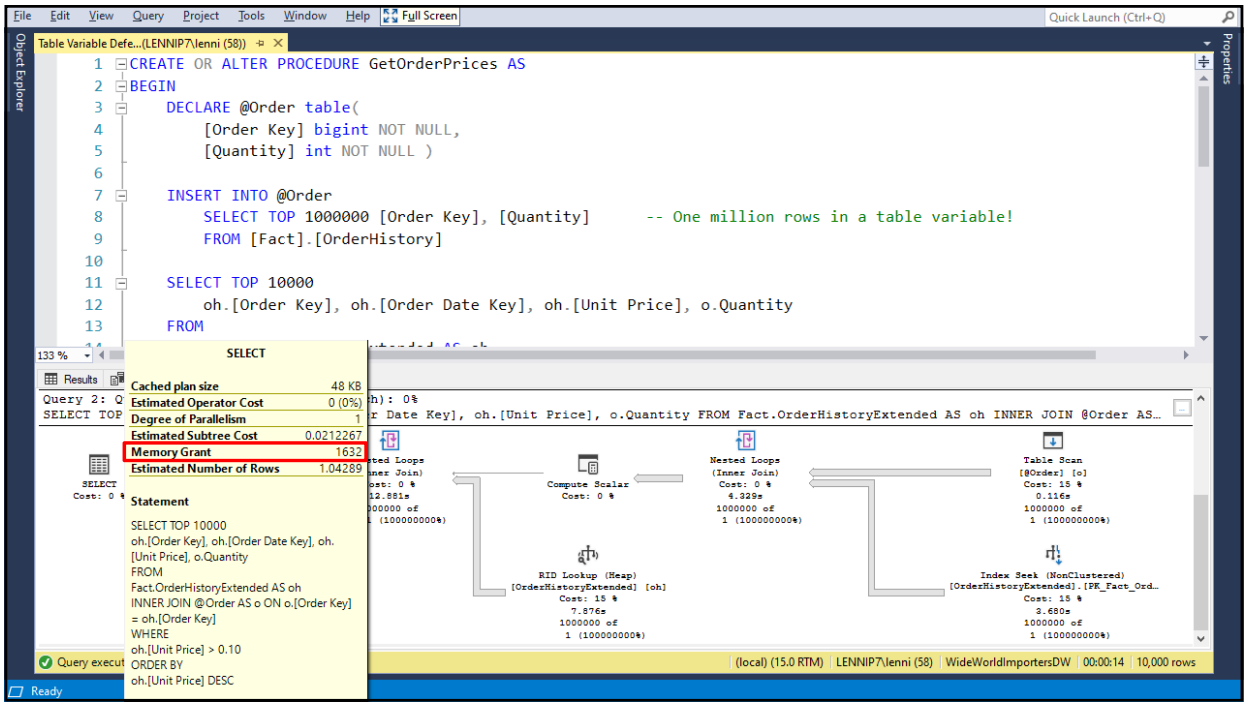

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
Object Explorer
Properties
1 CREATE OR ALTER PROCEDURE GetOrderPrices AS
2 BEGIN
3 DECLARE @Order table(
4 [Order Key] bigint NOT NULL,
5 [Quantity] int NOT NULL)
6
7 INSERT INTO @Order
8 SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
9 FROM [Fact].[OrderHistory]
10
11 SELECT TOP 10000
12 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
13 FROM
133 %
Results Messages Execution plan
Query 2: Query cost (relative to the batch): 0%
SELECT TOP 10000 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity FROM Fact.OrderHistoryExtended AS oh INNER JOIN @Order AS...
SELECT Cost: 0 %
Sort (Top N Sort) Cost: 54 % 13.646s 10000 of 1 (1000000%)
Nested Loops (Inner Join) Cost: 0 % 12.881s 1000000 of 1 (100000000%)
Compute Scalar Cost: 0 %
Nested Loops (Inner Join) Cost: 0 % 4.329s 1000000 of 1 (100000000%)
Table Scan [@Order] [o] Cost: 15 % 0.116s 1000000 of 1 (100000000%)
RID Lookup (Heap) [OrderHistoryExtended] [oh] Cost: 15 % 7.876s 1000000 of 1 (100000000%)
Index Seek (NonClustered) [OrderHistoryExtended].[PK_Fact_Ord... Cost: 15 % 3.680s 1000000 of 1 (100000000%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:14 10,000 rows
Ready

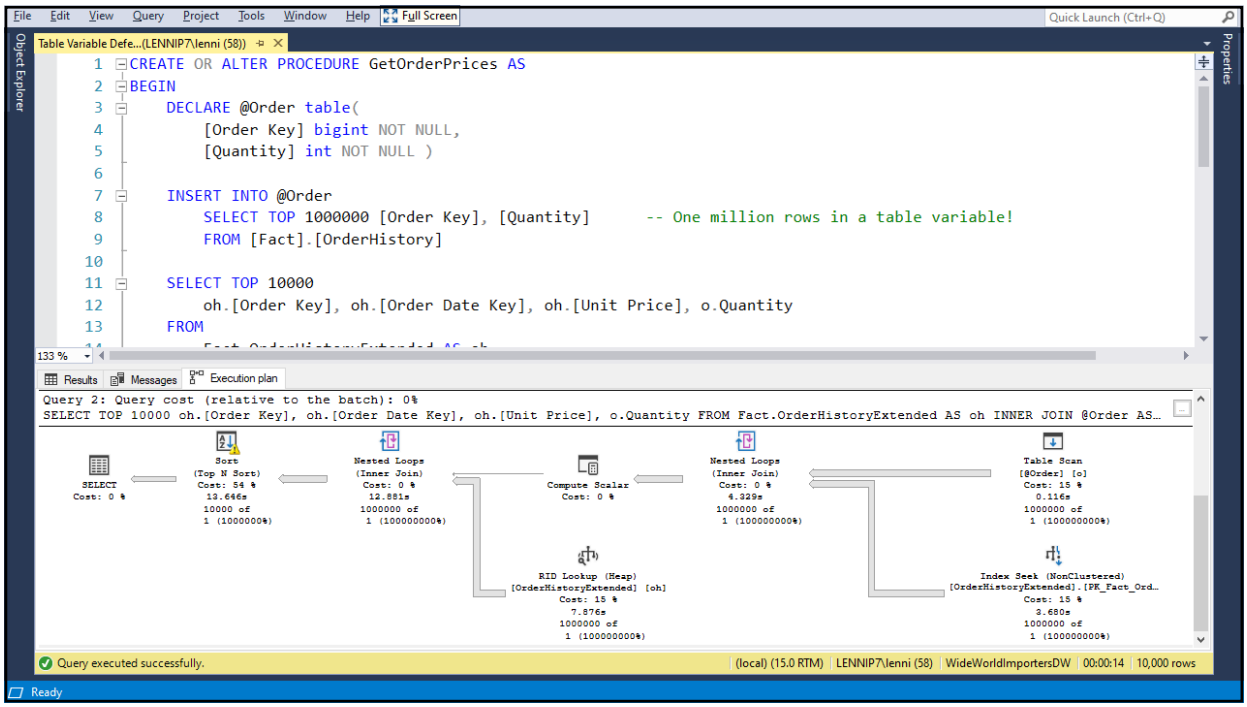

CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
DECLARE @Order table(
[Order Key] bigint NOT NULL,
[Quantity] int NOT NULL)
INSERT INTO @Order
SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a tab
FROM [Fact].[OrderHistory]
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
FROM
Table Scan
Scan rows from a table.
Physical Operation Table Scan
Logical Operation Table Scan
Actual Execution Mode Row
Estimated Execution Mode Row
Storage RowStore
Number of Rows Read 1000000
Actual Number of Rows 1000000
Actual Number of Batches 0
Estimated Operator Cost 0.0032831 (15%)
Estimated I/O Cost 0.003125
Estimated CPU Cost 0.0001581
Estimated Subtree Cost 0.0032831
Number of Executions 1
Estimated Number of Executions 1
Estimated Number of Rows to be Read 1
Estimated Number of Rows 1
Estimated Row Size 19 B
Actual Rebinds 0
Actual Rewinds 0
Ordered False
Node ID 6
Object
[@Order] [o]
Output List
@Order.Order Key, @Order.Quantity
Query 2: Query cost (relative to the batch): 0%
Query executed successfully.
Ready

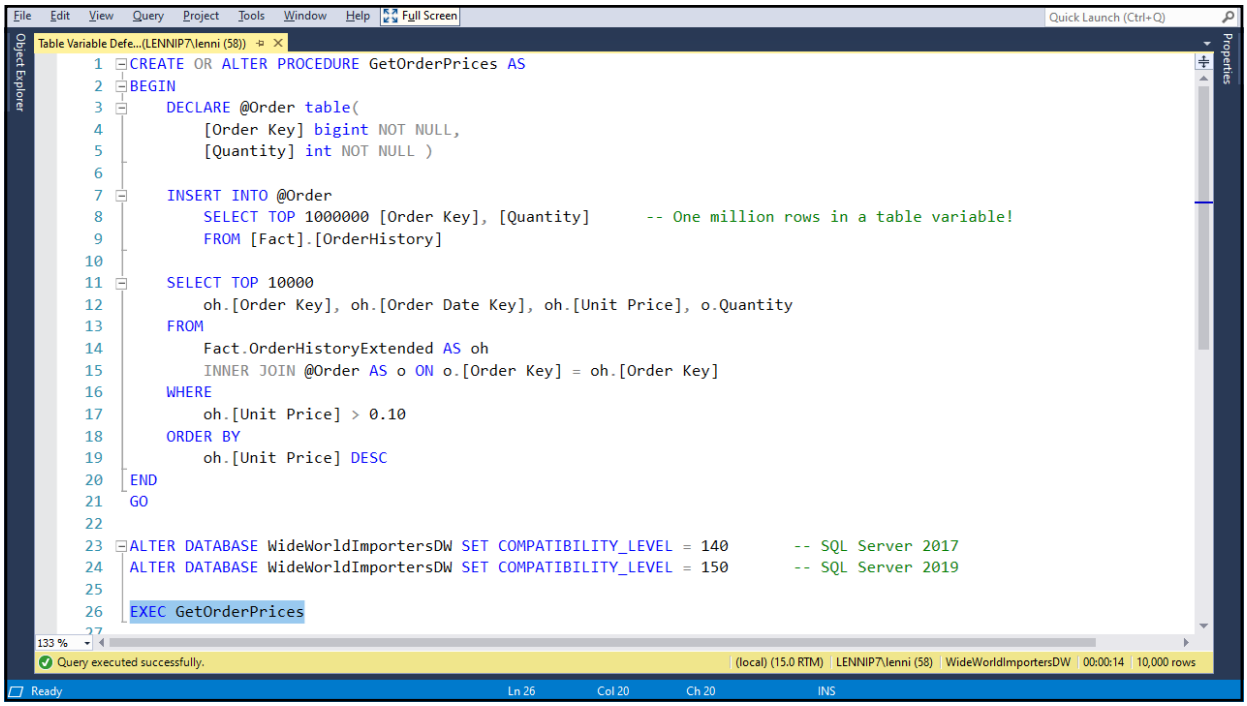

CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
DECLARE @Order table(
[Order Key] bigint NOT NULL,
[Quantity] int NOT NULL)
INSERT INTO @Order
SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
FROM [Fact].[OrderHistory]
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
FROM
Query 2: Query cost (relative to the batch): 0%
SELECT TOP 10000 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity FROM Fact.OrderHistoryExtended AS oh INNER JOIN @Order AS…
Query executed successfully.
Ready


CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
-- One million rows in a table variable!
Unit Price], o.Quantity
Sort
Sort the input.
Physical Operation Sort
Logical Operation Top N Sort
Actual Execution Mode Row
Estimated Execution Mode Row
Actual Number of Rows 10000
Actual Number of Batches 0
Estimated Operator Cost 0.0113619 (54%)
Estimated I/O Cost 0.0112613
Estimated CPU Cost 0.0001001
Estimated Subtree Cost 0.0212267
Number of Executions 1
Estimated Number of Executions 1
Estimated Number of Rows 1.04289
Estimated Row Size 31 B
Actual Rebinds 1
Actual Rewinds 0
Node ID 0
Output List
Warnings
Operator used tempdb to spill data during execution with spill level 2 and 1 spilled thread(s), Sort wrote 14952 pages to and read 14952 pages from tempdb with granted memory 544KB and used memory 544KB
Order By
[WideWorldImportersDW].[Fact].[OrderHistoryExtended].Unit Price Descending
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:14 10,000 rows
Ready

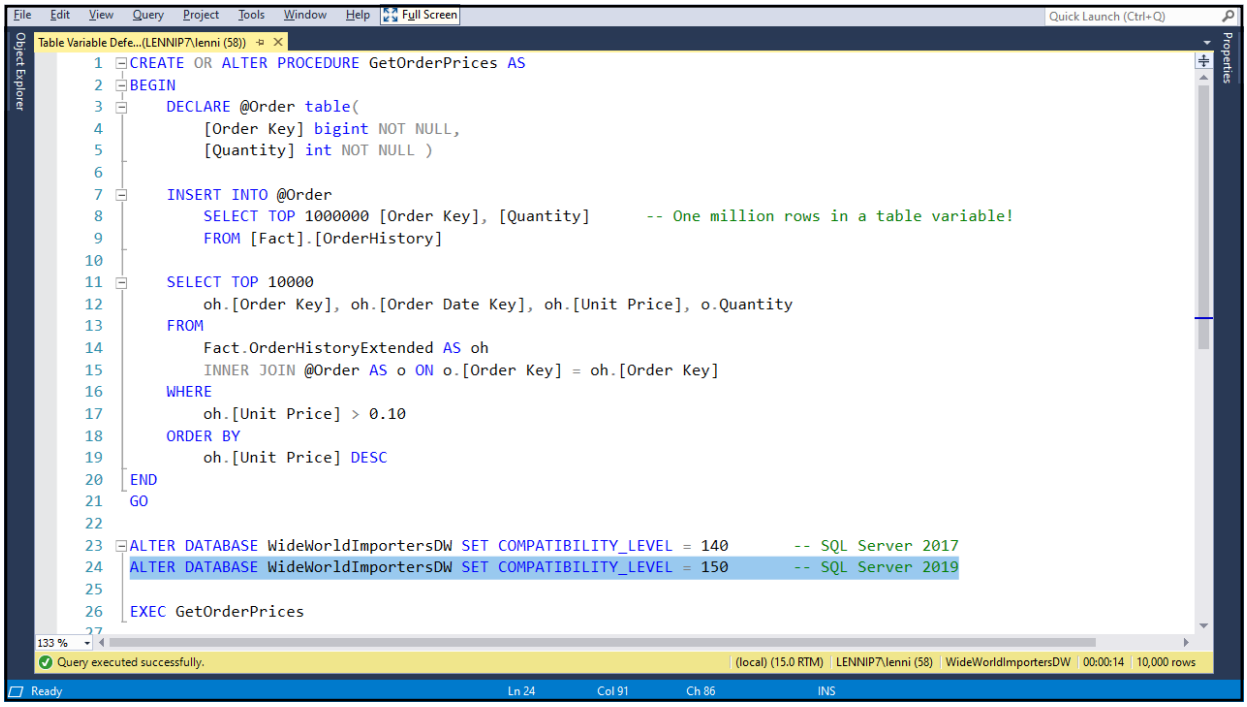

CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
DECLARE @Order table(
[Order Key] bigint NOT NULL,
[Quantity] int NOT NULL)
INSERT INTO @Order
SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
FROM [Fact].[OrderHistory]
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
FROM
Query 2: Query cost (relative to the batch): 0%
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:14 10,000 rows
Ready


CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
DECLARE @Order table(
[Order Key] bigint NOT NULL,
[Quantity] int NOT NULL)
INSERT INTO @Order
SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
FROM [Fact].[OrderHistory]
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
FROM
SELECT
Cached plan size 48 KB
Estimated Operator Cost 0 (0%)
Degree of Parallelism 1
Estimated Subtree Cost 0.0212267
Memory Grant 1632
Estimated Number of Rows 1.04289
Statement
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.
[Unit Price], o.Quantity
FROM
Fact.OrderHistoryExtended AS oh
INNER JOIN @Order AS o ON o.[Order Key]
= oh.[Order Key]
WHERE
oh.[Unit Price] > 0.10
ORDER BY
oh.[Unit Price] DESC
Compute Scalar
Nested Loops
(Inner Join)
Table Scan
[@Order] [o]
RID Lookup (Heap)
[OrderHistoryExtended] [oh]
Index Seek (NonClustered)
[OrderHistoryExtended].[PK_Fact_Ord...
(local) (15.0 RTM)
LENNIP7\lenni (58)
WideWorldImportersDW
00:00:14
10,000 rows
Ready

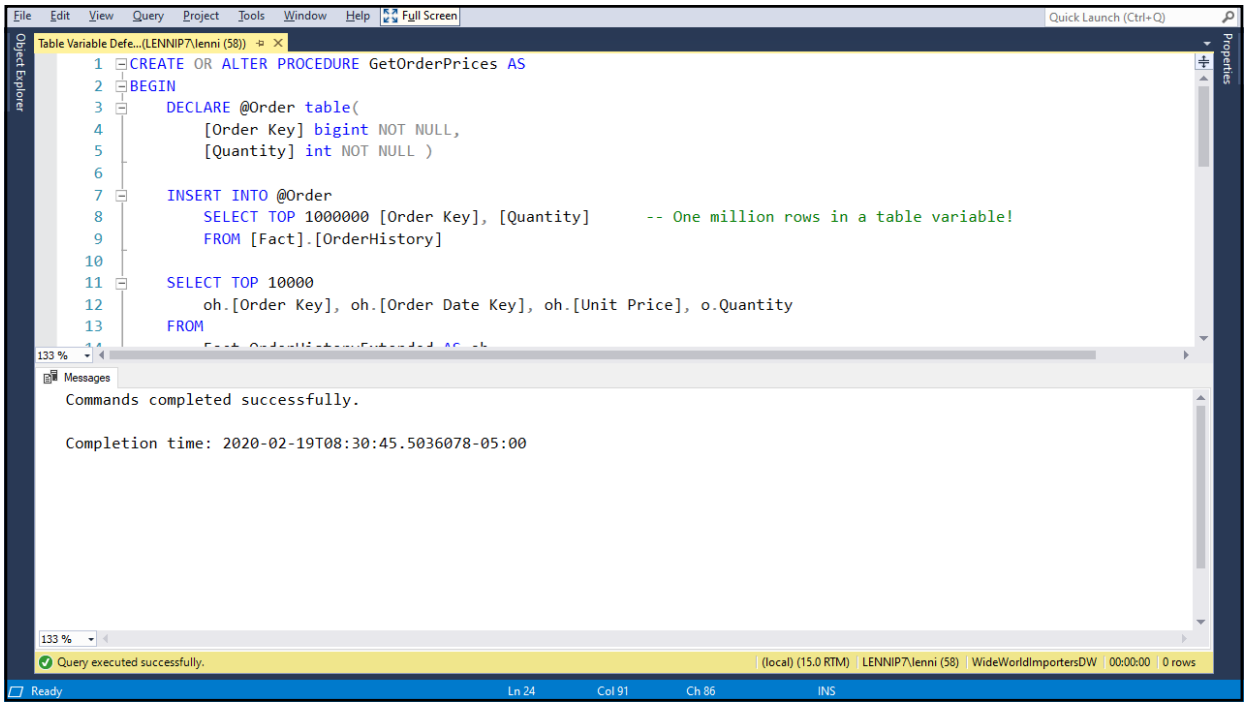

CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
DECLARE @Order table(
[Order Key] bigint NOT NULL,
[Quantity] int NOT NULL)
INSERT INTO @Order
SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
FROM [Fact].[OrderHistory]
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
FROM
Results
Messages
Execution plan
Query 2: Query cost (relative to the batch): 0%
SELECT TOP 10000 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity FROM Fact.OrderHistoryExtended AS oh INNER JOIN @Order AS...
SELECT
Sort
(Top N Sort)
Nested Loops
(Inner Join)
Compute Scalar
Table Scan
[@Order] [o]
RID Lookup (Heap)
[OrderHistoryExtended] [oh]
Index Seek (NonClustered)
[OrderHistoryExtended].[PK_Fact_Ord...
Query executed successfully.
(local) (15.0 RTM)
LENNIP7\lenni (58)
WideWorldImportersDW
00:00:14
10,000 rows
Ready

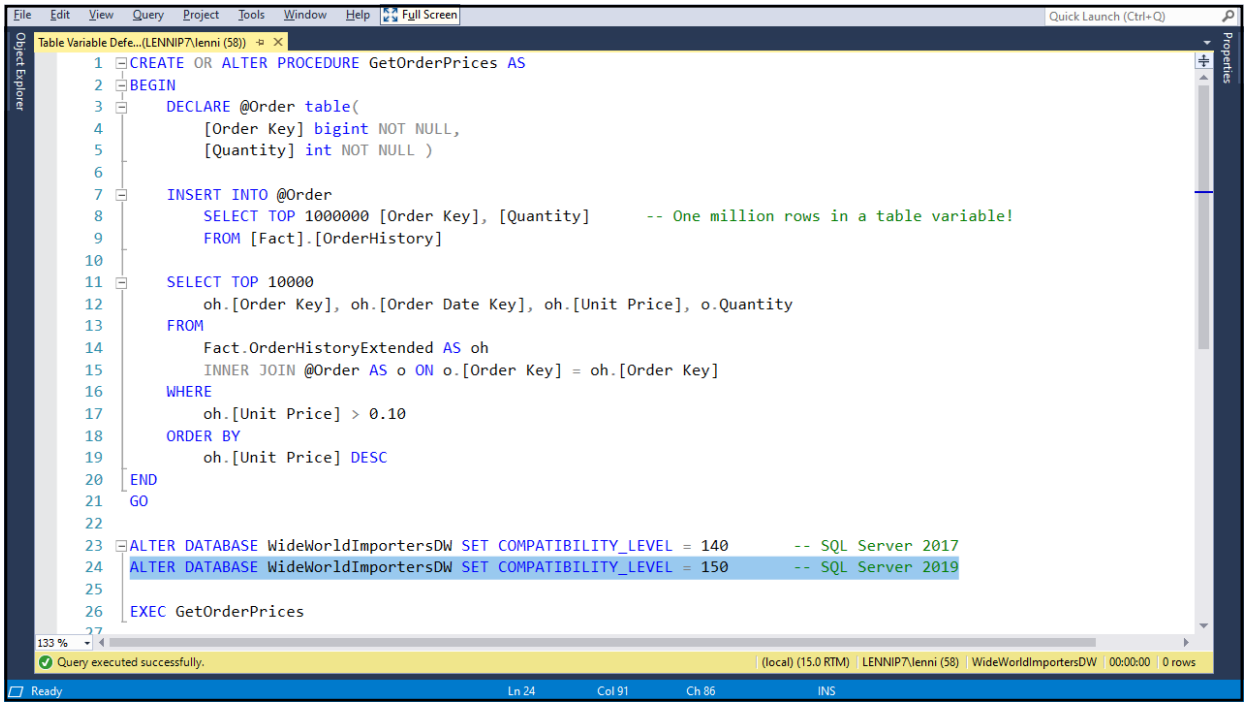


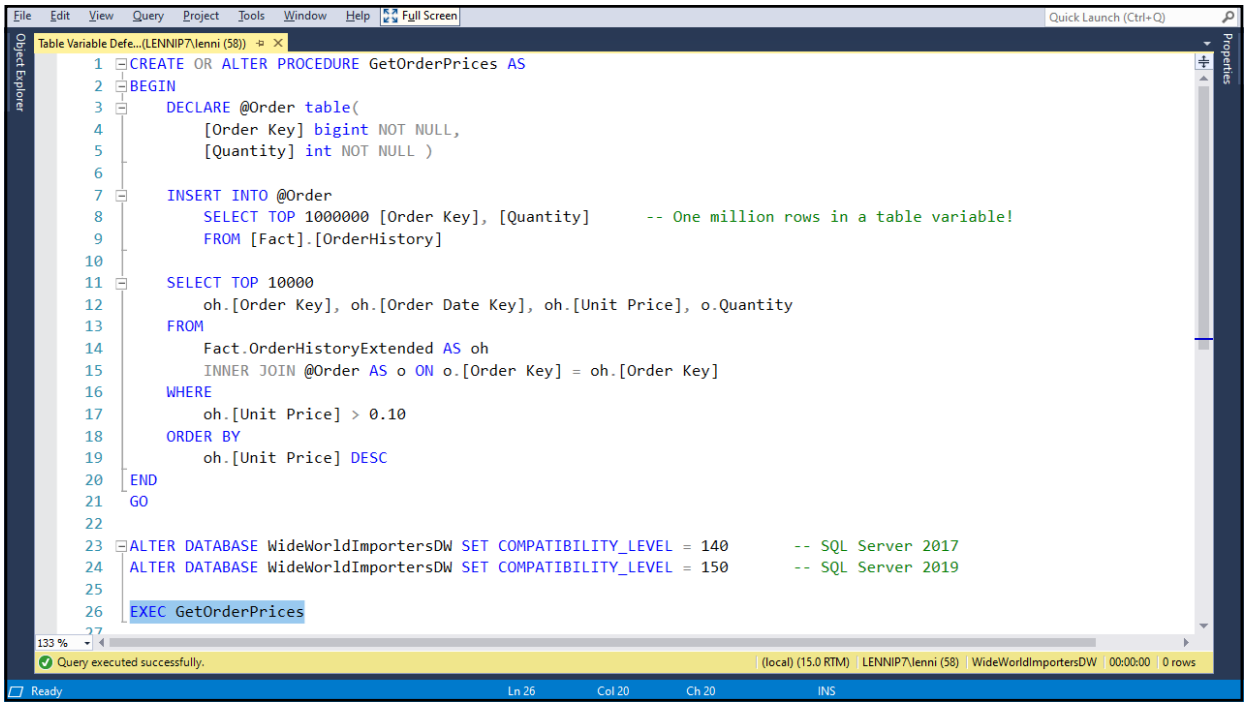
```
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
    DECLARE @Order table(
        [Order Key] bigint NOT NULL,
        [Quantity] int NOT NULL )

    INSERT INTO @Order
        SELECT TOP 1000000 [Order Key], [Quantity]      -- One million rows in a table variable!
        FROM [Fact].[OrderHistory]

    SELECT TOP 10000
        oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
    FROM
        Fact.OrderHistoryExtended AS oh
        INNER JOIN @Order AS o ON o.[Order Key] = oh.[Order Key]
    WHERE
        oh.[Unit Price] > 0.10
    ORDER BY
        oh.[Unit Price] DESC
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150       -- SQL Server 2019

EXEC GetOrderPrices
```

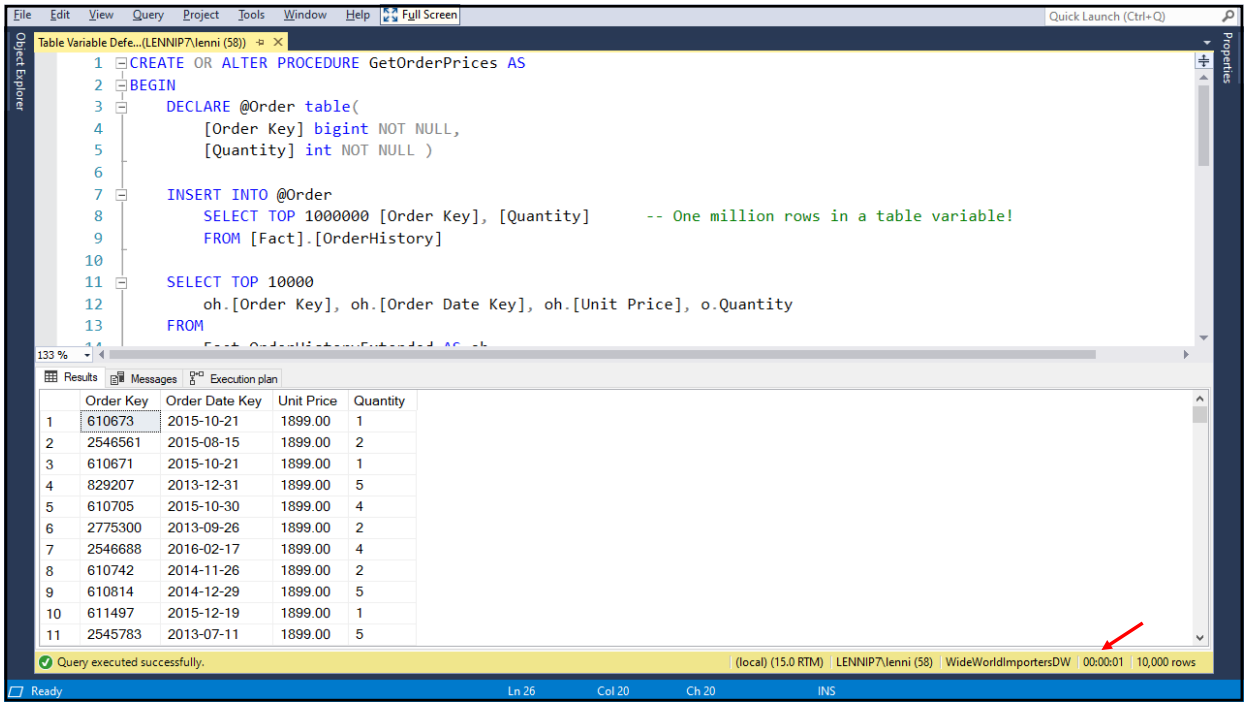


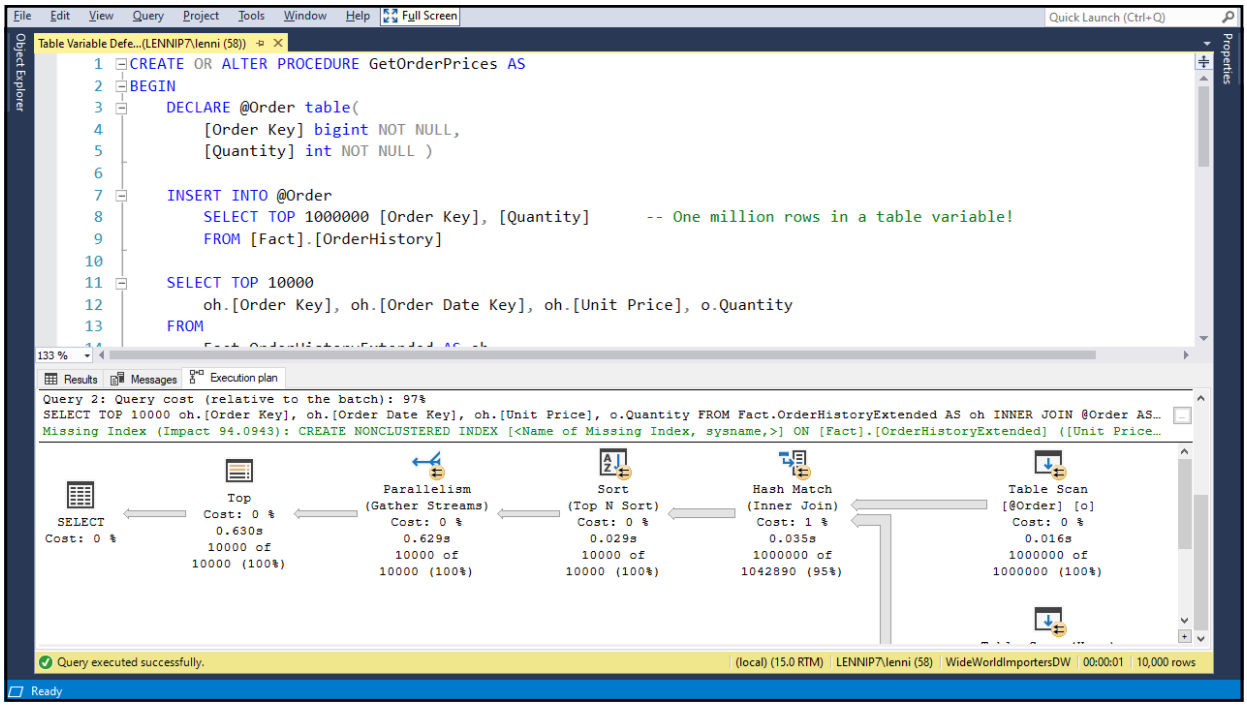
```
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
    DECLARE @Order table(
        [Order Key] bigint NOT NULL,
        [Quantity] int NOT NULL )

    INSERT INTO @Order
        SELECT TOP 1000000 [Order Key], [Quantity]      -- One million rows in a table variable!
        FROM [Fact].[OrderHistory]

    SELECT TOP 10000
        oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
    FROM
        Fact.OrderHistoryExtended AS oh
        INNER JOIN @Order AS o ON o.[Order Key] = oh.[Order Key]
    WHERE
        oh.[Unit Price] > 0.10
    ORDER BY
        oh.[Unit Price] DESC
END
GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150       -- SQL Server 2019

EXEC GetOrderPrices
```

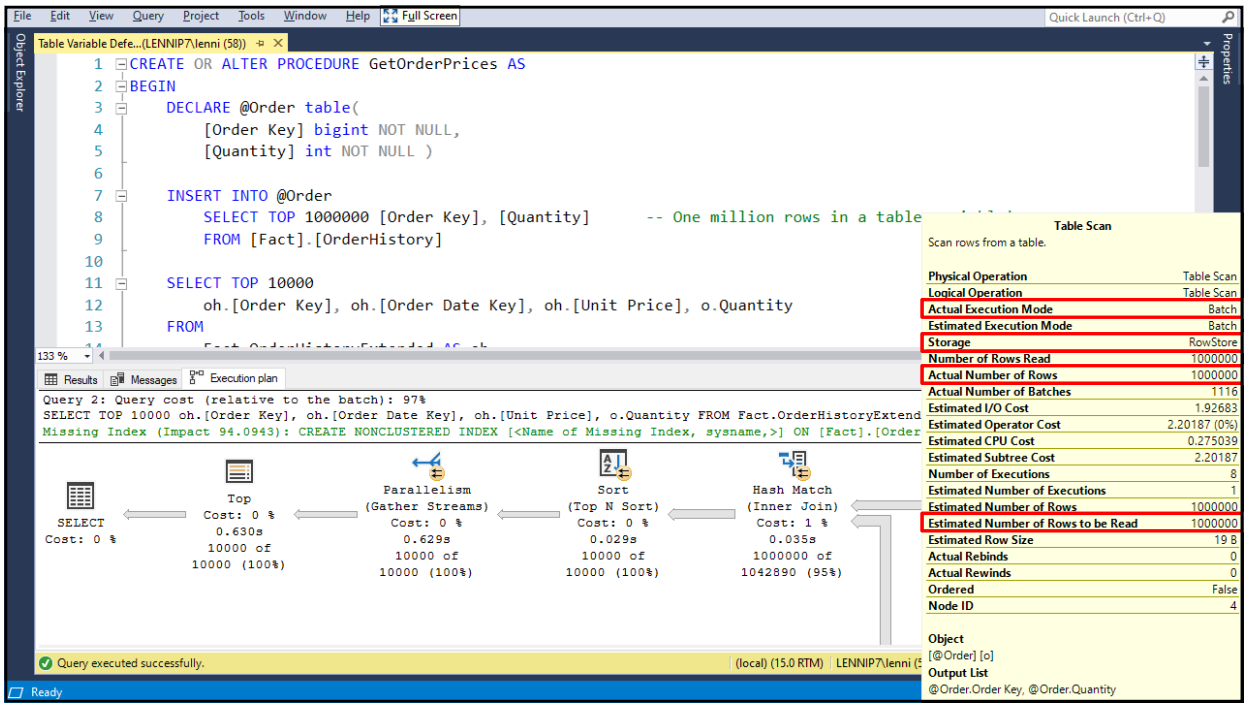

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
Object Explorer
Properties
1 CREATE OR ALTER PROCEDURE GetOrderPrices AS
2 BEGIN
3 DECLARE @Order table(
4 [Order Key] bigint NOT NULL,
5 [Quantity] int NOT NULL)
6
7 INSERT INTO @Order
8 SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
9 FROM [Fact].[OrderHistory]
10
11 SELECT TOP 10000
12 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
13 FROM
133 %
Messages
Commands completed successfully.
Completion time: 2020-02-19T08:30:45.5036078-05:00
133 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 24 Col 91 Ch 86 INS

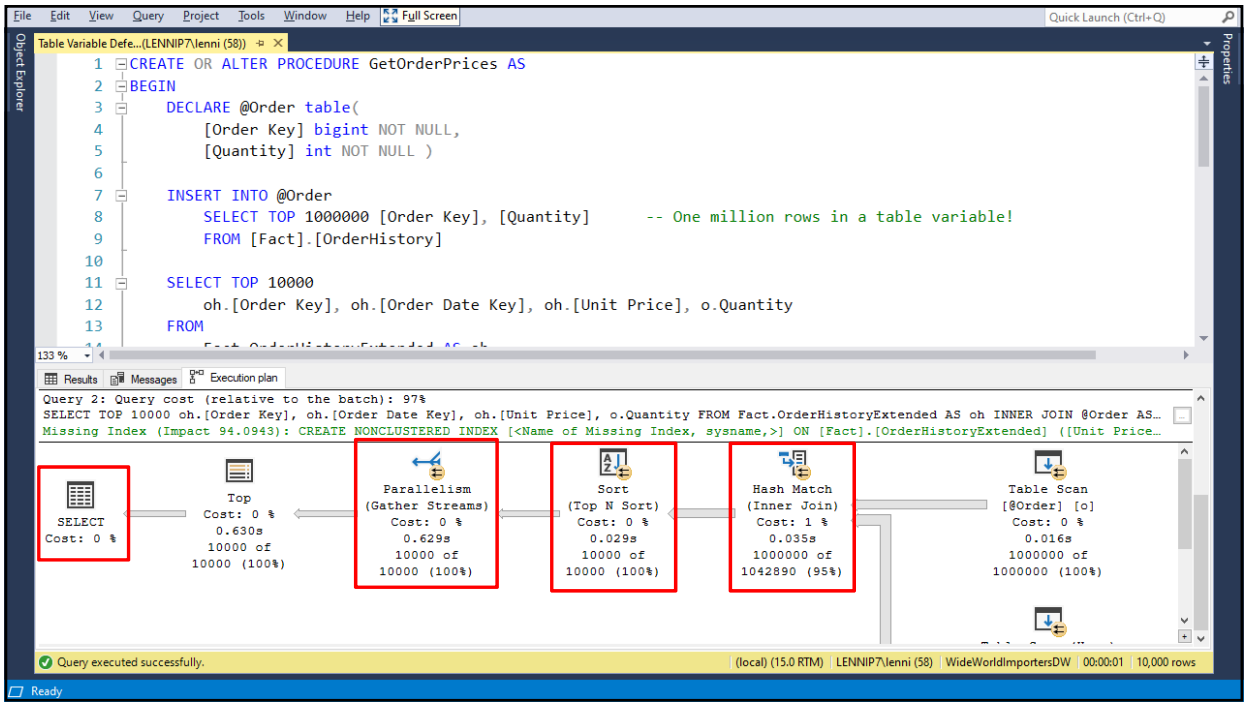

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
Object Explorer
Properties
1 CREATE OR ALTER PROCEDURE GetOrderPrices AS
2 BEGIN
3 DECLARE @Order table(
4 [Order Key] bigint NOT NULL,
5 [Quantity] int NOT NULL)
6
7 INSERT INTO @Order
8 SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
9 FROM [Fact].[OrderHistory]
10
11 SELECT TOP 10000
12 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
13 FROM
14 Fact.OrderHistoryExtended AS oh
15 INNER JOIN @Order AS o ON o.[Order Key] = oh.[Order Key]
16 WHERE
17 oh.[Unit Price] > 0.10
18 ORDER BY
19 oh.[Unit Price] DESC
20 END
21 GO
22
23 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
24 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
25
26 EXEC GetOrderPrices
133 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 24 Col 91 Ch 86 INS

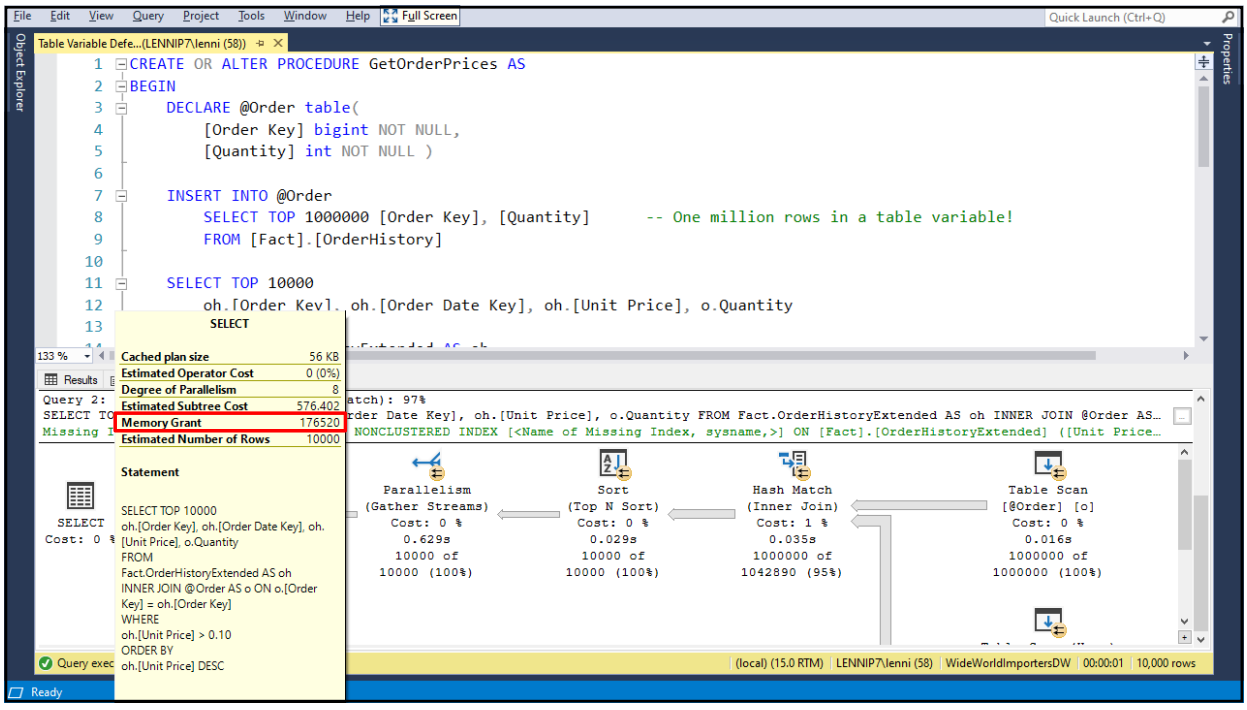

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
Object Explorer
Properties
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
DECLARE @Order table(
[Order Key] bigint NOT NULL,
[Quantity] int NOT NULL)
INSERT INTO @Order
SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
FROM [Fact].[OrderHistory]
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
FROM
Fact.OrderHistoryExtended AS oh
INNER JOIN @Order AS o ON o.[Order Key] = oh.[Order Key]
WHERE
oh.[Unit Price] > 0.10
ORDER BY
oh.[Unit Price] DESC
END
GO
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
EXEC GetOrderPrices
133 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:00 0 rows
Ready Ln 26 Col 20 Ch 20 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
Object Explorer
Properties
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
DECLARE @Order table(
[Order Key] bigint NOT NULL,
[Quantity] int NOT NULL)
INSERT INTO @Order
SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
FROM [Fact].[OrderHistory]
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
FROM
133 %
Results Messages Execution plan
Order Key Order Date Key Unit Price Quantity
1 610673 2015-10-21 1899.00 1
2 2546561 2015-08-15 1899.00 2
3 610671 2015-10-21 1899.00 1
4 829207 2013-12-31 1899.00 5
5 610705 2015-10-30 1899.00 4
6 2775300 2013-09-26 1899.00 2
7 2546688 2016-02-17 1899.00 4
8 610742 2014-11-26 1899.00 2
9 610814 2014-12-29 1899.00 5
10 611497 2015-12-19 1899.00 1
11 2545783 2013-07-11 1899.00 5
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:01 10,000 rows
Ready Ln 26 Col 20 Ch 20 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
Object Explorer
Properties
1 CREATE OR ALTER PROCEDURE GetOrderPrices AS
2 BEGIN
3 DECLARE @Order table(
4 [Order Key] bigint NOT NULL,
5 [Quantity] int NOT NULL)
6
7 INSERT INTO @Order
8 SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
9 FROM [Fact].[OrderHistory]
10
11 SELECT TOP 10000
12 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
13 FROM
133 %
Results Messages Execution plan
Query 2: Query cost (relative to the batch): 97%
SELECT TOP 10000 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity FROM Fact.OrderHistoryExtended AS oh INNER JOIN @Order AS…
Missing Index (Impact 94.0943): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [Fact].[OrderHistoryExtended] ([Unit Price…
SELECT Cost: 0 %
Top Cost: 0 % 0.630s 10000 of 10000 (100%)
Parallelism (Gather Streams) Cost: 0 % 0.629s 10000 of 10000 (100%)
Sort (Top N Sort) Cost: 0 % 0.029s 10000 of 10000 (100%)
Hash Match (Inner Join) Cost: 1 % 0.035s 1000000 of 1042890 (95%)
Table Scan [@Order] [o] Cost: 0 % 0.016s 1000000 of 1000000 (100%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:01 10,000 rows
Ready


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
Object Explorer
Properties
1 CREATE OR ALTER PROCEDURE GetOrderPrices AS
2 BEGIN
3 DECLARE @Order table(
4 [Order Key] bigint NOT NULL,
5 [Quantity] int NOT NULL)
6
7 INSERT INTO @Order
8 SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table
9 FROM [Fact].[OrderHistory]
10
11 SELECT TOP 10000
12 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
13 FROM
133 %
Results Messages Execution plan
Query 2: Query cost (relative to the batch): 97%
SELECT TOP 10000 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity FROM Fact.OrderHistoryExtend
Missing Index (Impact 94.0943): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [Fact].[Order
SELECT Cost: 0 %
Top Cost: 0 % 0.630s 10000 of 10000 (100%)
Parallelism (Gather Streams) Cost: 0 % 0.629s 10000 of 10000 (100%)
Sort (Top N Sort) Cost: 0 % 0.029s 10000 of 10000 (100%)
Hash Match (Inner Join) Cost: 1 % 0.035s 1000000 of 1042890 (95%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (5
Ready
Table Scan
Scan rows from a table.
Physical Operation Table Scan
Logical Operation Table Scan
Actual Execution Mode Batch
Estimated Execution Mode Batch
Storage RowStore
Number of Rows Read 1000000
Actual Number of Rows 1000000
Actual Number of Batches 1116
Estimated I/O Cost 1.92683
Estimated Operator Cost 2.20187 (0%)
Estimated CPU Cost 0.275039
Estimated Subtree Cost 2.20187
Number of Executions 8
Estimated Number of Executions 1
Estimated Number of Rows 1000000
Estimated Number of Rows to be Read 1000000
Estimated Row Size 19 B
Actual Rebinds 0
Actual Rewinds 0
Ordered False
Node ID 4
Object
[@Order] [o]
Output List
@Order.Order Key, @Order.Quantity

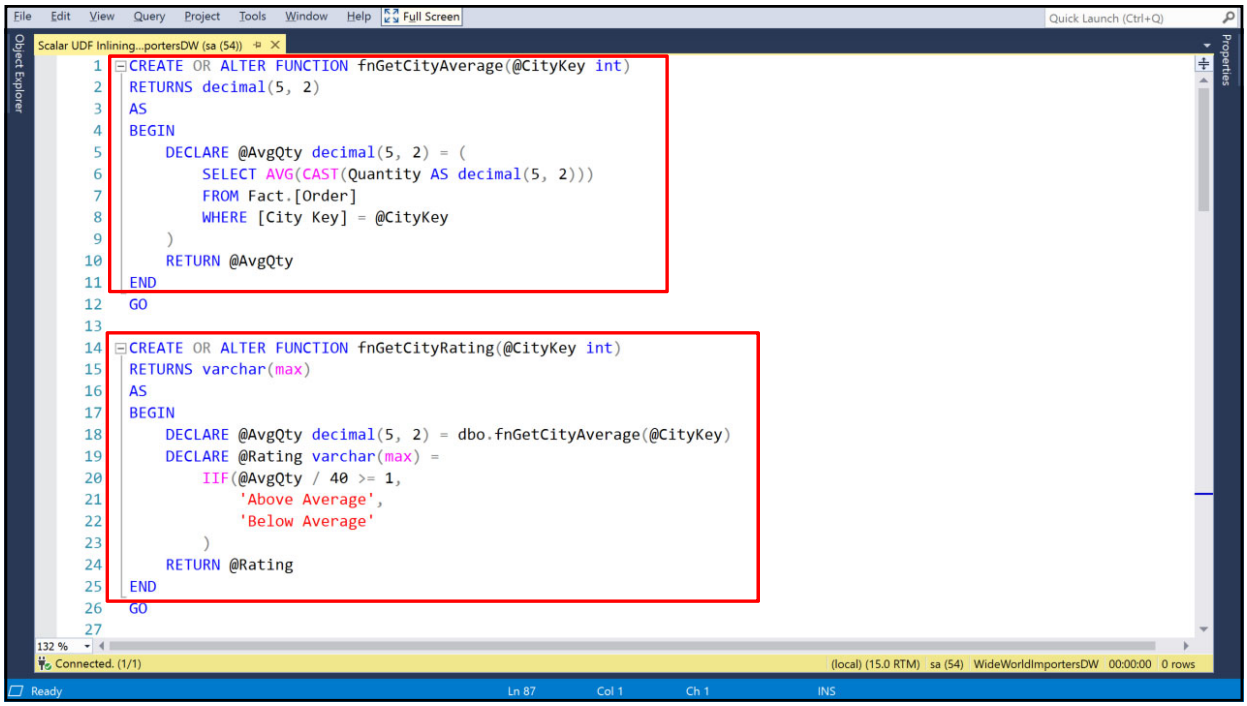

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
DECLARE @Order table(
[Order Key] bigint NOT NULL,
[Quantity] int NOT NULL)
INSERT INTO @Order
SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
FROM [Fact].[OrderHistory]
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
FROM
Results Messages Execution plan
Query 2: Query cost (relative to the batch): 97%
SELECT TOP 10000 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity FROM Fact.OrderHistoryExtended AS oh INNER JOIN @Order AS…
Missing Index (Impact 94.0943): CREATE NONCLUSTERED INDEX [<Name of Missing Index, sysname,>] ON [Fact].[OrderHistoryExtended] ([Unit Price…
SELECT Cost: 0 %
Top Cost: 0 % 0.630s 10000 of 10000 (100%)
Parallelism (Gather Streams) Cost: 0 % 0.629s 10000 of 10000 (100%)
Sort (Top N Sort) Cost: 0 % 0.029s 10000 of 10000 (100%)
Hash Match (Inner Join) Cost: 1 % 0.035s 1000000 of 1042890 (95%)
Table Scan [@Order] [o] Cost: 0 % 0.016s 1000000 of 1000000 (100%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:01 10,000 rows
Ready


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Table Variable Defe...(LENNIP7\lenni (58))
CREATE OR ALTER PROCEDURE GetOrderPrices AS
BEGIN
DECLARE @Order table(
[Order Key] bigint NOT NULL,
[Quantity] int NOT NULL)
INSERT INTO @Order
SELECT TOP 1000000 [Order Key], [Quantity] -- One million rows in a table variable!
FROM [Fact].[OrderHistory]
SELECT TOP 10000
oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity
SELECT
Cached plan size 56 KB
Estimated Operator Cost 0 (0%)
Degree of Parallelism 8
Estimated Subtree Cost 576.402
Memory Grant 176520
Estimated Number of Rows 10000
Statement
SELECT TOP 10000 oh.[Order Key], oh.[Order Date Key], oh.[Unit Price], o.Quantity FROM Fact.OrderHistoryExtended AS oh INNER JOIN @Order AS o ON o.[Order Key] = oh.[Order Key] WHERE oh.[Unit Price] > 0.10 ORDER BY oh.[Unit Price] DESC
Parallelism (Gather Streams) Cost: 0 % 0.629s 10000 of 10000 (100%)
Sort (Top N Sort) Cost: 0 % 0.029s 10000 of 10000 (100%)
Hash Match (Inner Join) Cost: 1 % 0.035s 1000000 of 1042890 (95%)
Table Scan [@Order] [o] Cost: 0 % 0.016s 1000000 of 1000000 (100%)
(local) (15.0 RTM) LENNIP7\lenni (58) WideWorldImportersDW 00:00:01 10,000 rows
Ready

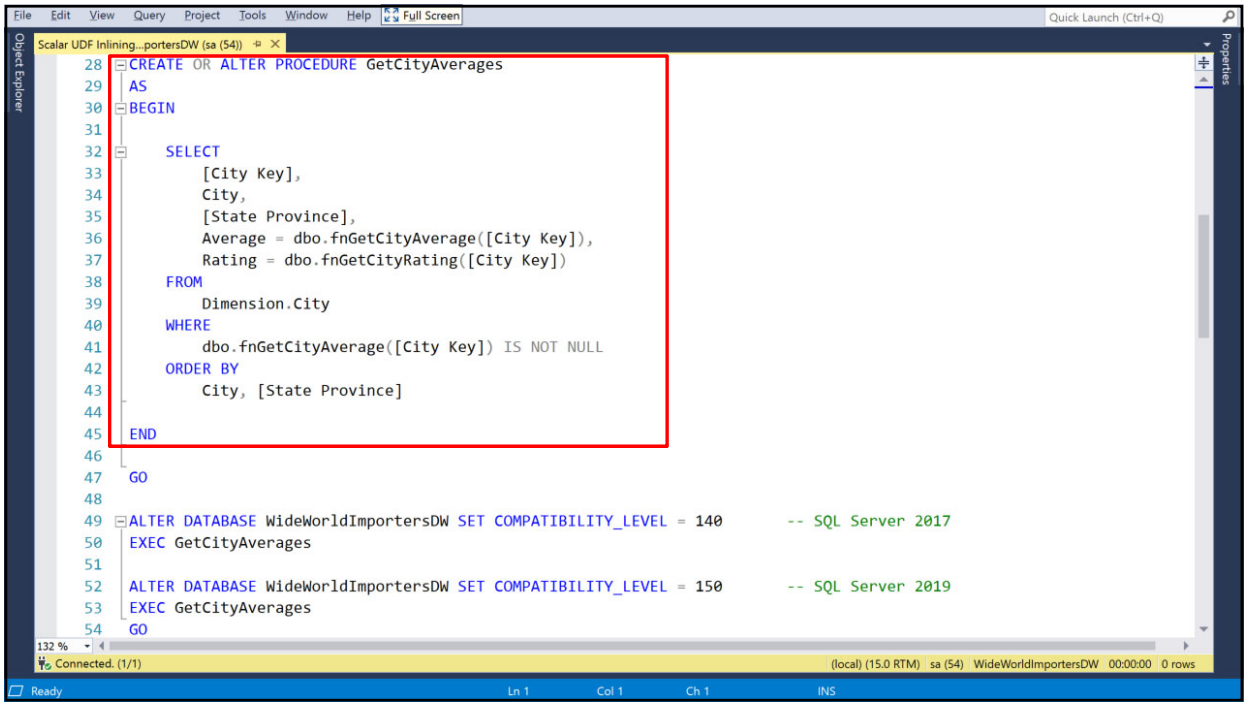
Scalar UDF Inlining

- Scalar UDFs
  - Hugely popular programming feature
    - Encapsulate reusable logic
- Problem
  - Can lead to query performance problems, especially in the WHERE clause
    - UDF code must be applied one row at a time
    - Calling procedure must run on a single thread
- Solution
  - Integrate the UDF code itself into the overall query, if possible
- Not all UDFs will benefit
  - `ALTER DATABASE SCOPED CONFIGURATION SET TSQL_SCALAR_UDF_INLINING = OFF`
  - `OPTION (USE HINT('DISABLE_TSQL_SCALAR_UDF_INLINING'))`
- Still best to avoid UDFs

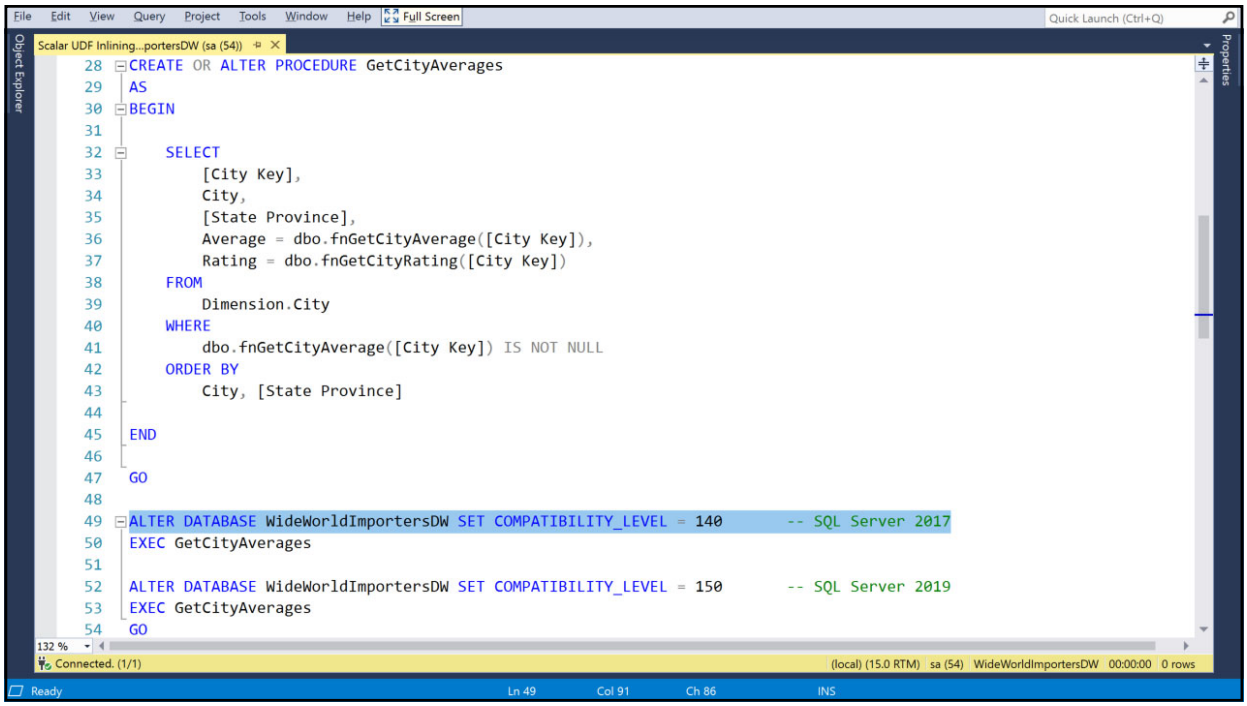
Scalar UDF Inlining

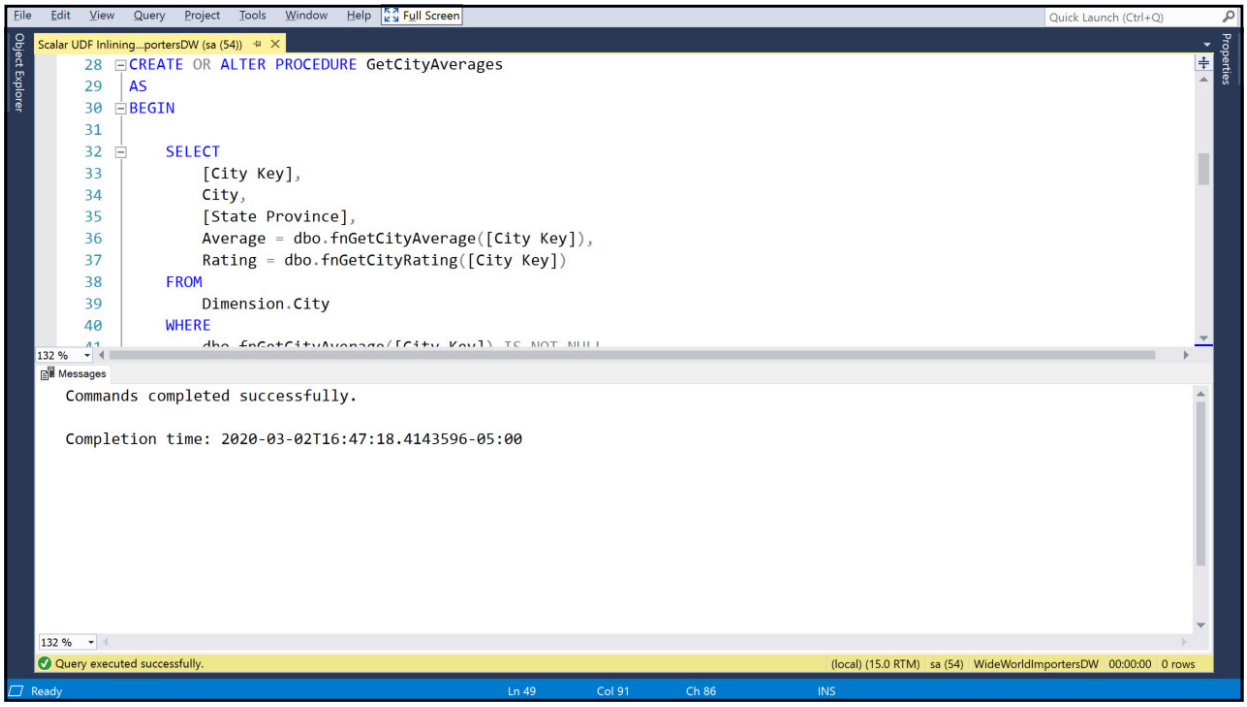


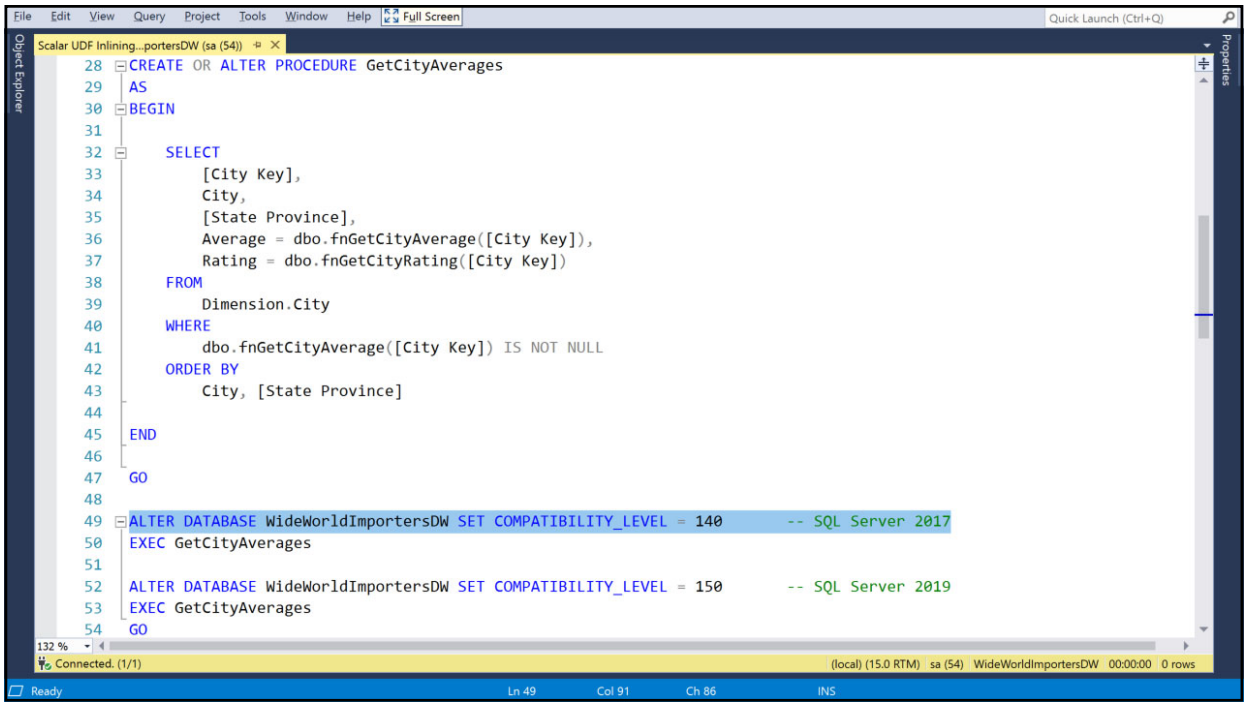
```
CREATE OR ALTER FUNCTION fnGetCityAverage(@CityKey int)
RETURNS decimal(5, 2)
AS
BEGIN
    DECLARE @AvgQty decimal(5, 2) = (
        SELECT AVG(CAST(Quantity AS decimal(5, 2)))
        FROM Fact.[Order]
        WHERE [City Key] = @CityKey
    )
    RETURN @AvgQty
END
GO

CREATE OR ALTER FUNCTION fnGetCityRating(@CityKey int)
RETURNS varchar(max)
AS
BEGIN
    DECLARE @AvgQty decimal(5, 2) = dbo.fnGetCityAverage(@CityKey)
    DECLARE @Rating varchar(max) =
        IIF(@AvgQty / 40 >= 1,
            'Above Average',
            'Below Average'
        )
    RETURN @Rating
END
GO
```

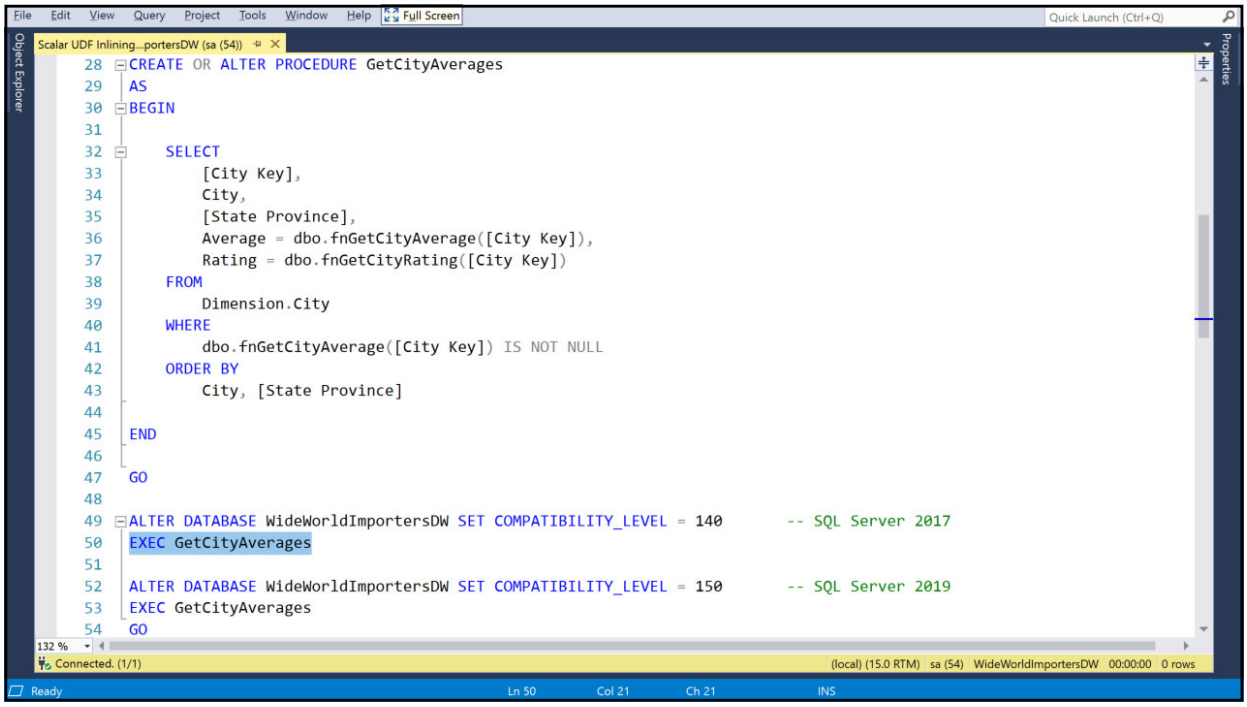


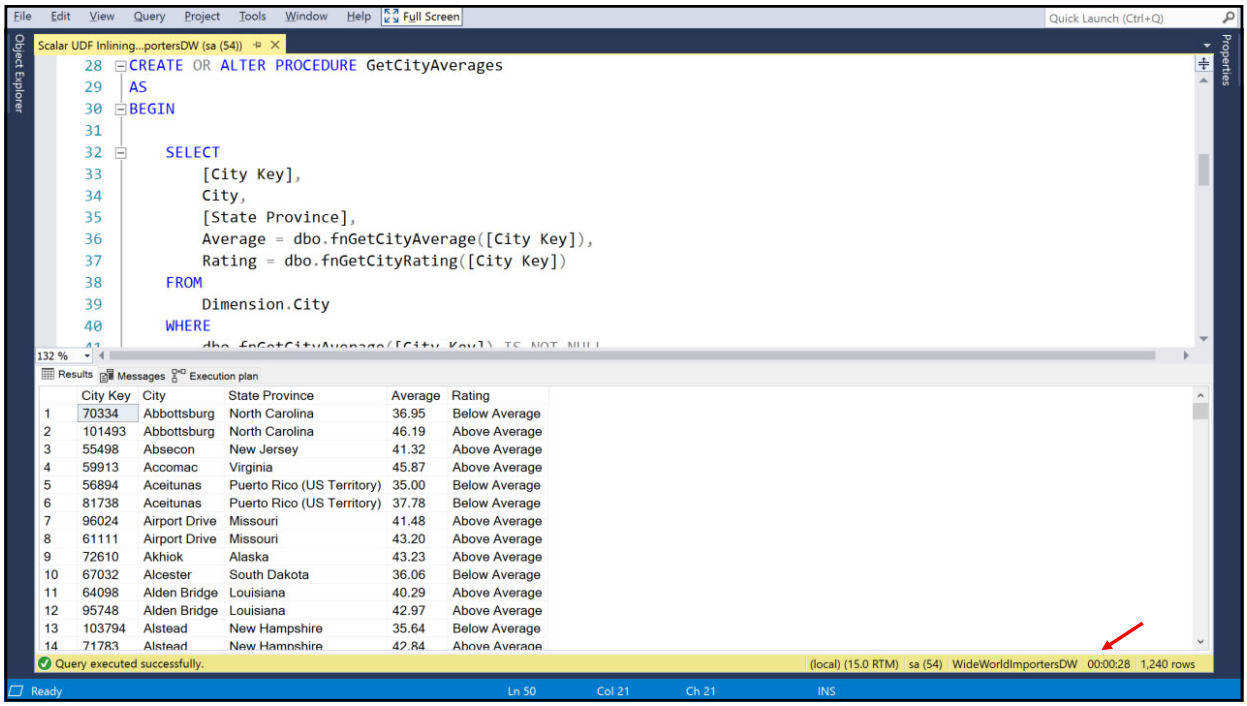
```
CREATE OR ALTER PROCEDURE GetCityAverages
AS
BEGIN

    SELECT
        [City Key],
        City,
        [State Province],
        Average = dbo.fnGetCityAverage([City Key]),
        Rating = dbo.fnGetCityRating([City Key])
    FROM
        Dimension.City
    WHERE
        dbo.fnGetCityAverage([City Key]) IS NOT NULL
    ORDER BY
        City, [State Province]

END

GO

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017
EXEC GetCityAverages

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150       -- SQL Server 2019
EXEC GetCityAverages
GO
```

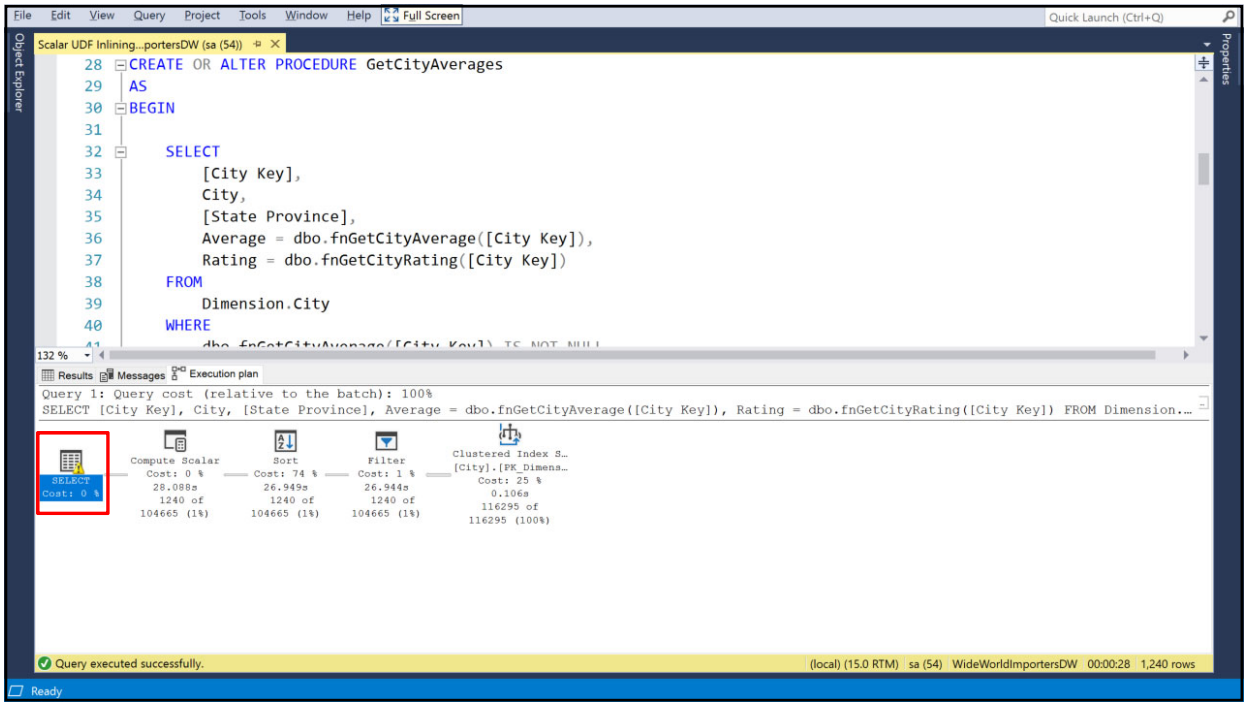

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
41 dbo.fnGetCityAverage([City Key]) IS NOT NULL
42 ORDER BY
43 City, [State Province]
44
45 END
46
47 GO
48
49 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
50 EXEC GetCityAverages
51
52 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
53 EXEC GetCityAverages
54 GO
132 %
Connected. (1/1)
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 49 Col 91 Ch 86 INS

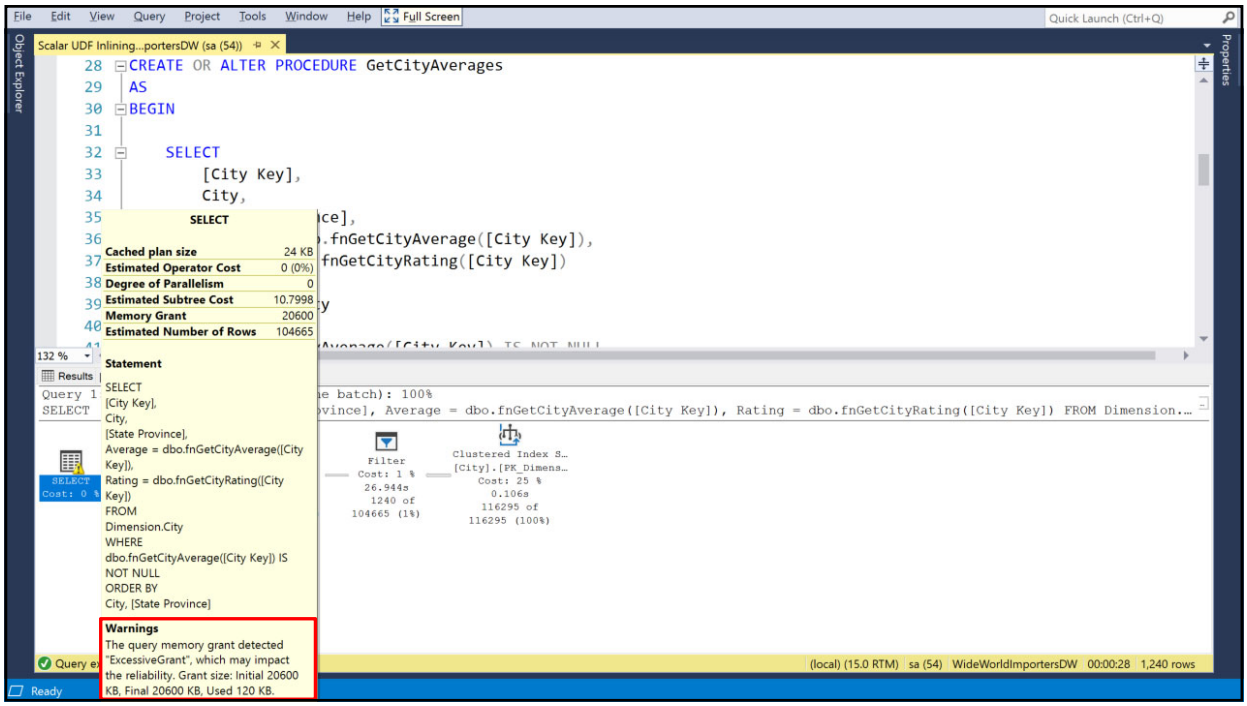

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
132 %
Messages
Commands completed successfully.
Completion time: 2020-03-02T16:47:18.4143596-05:00
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 49 Col 91 Ch 86 INS

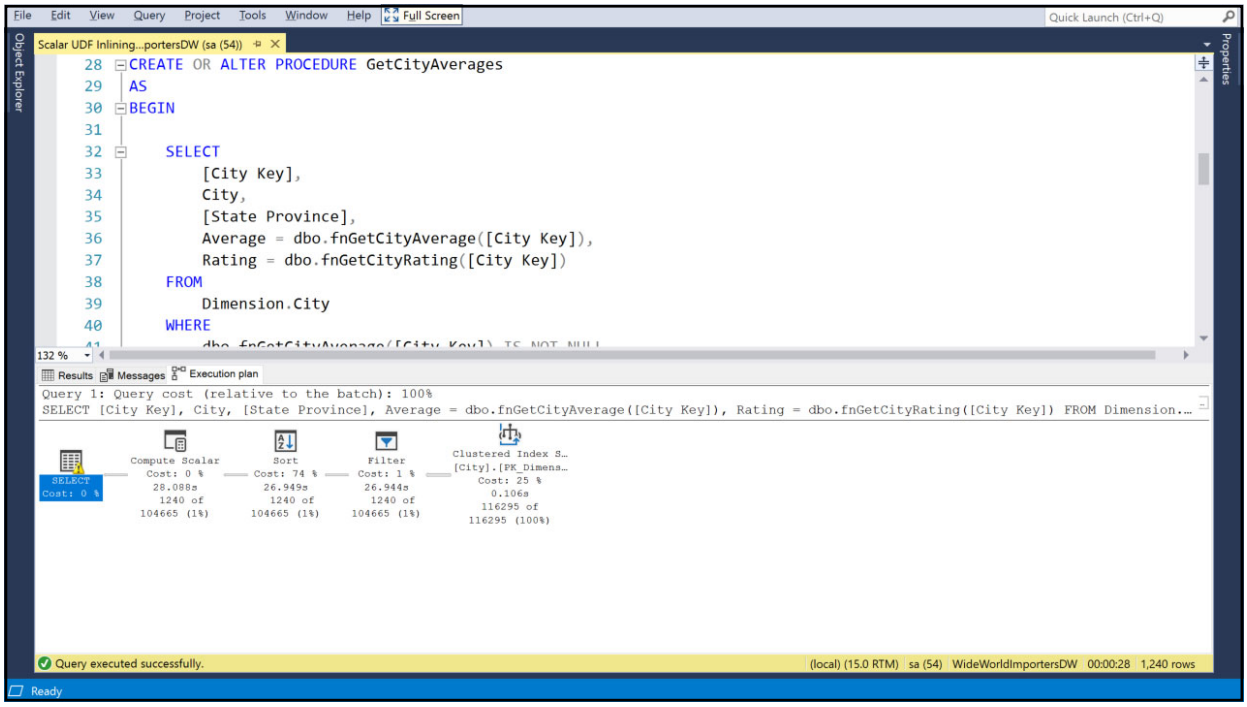

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
41 dbo.fnGetCityAverage([City Key]) IS NOT NULL
42 ORDER BY
43 City, [State Province]
44
45 END
46
47 GO
48
49 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
50 EXEC GetCityAverages
51
52 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
53 EXEC GetCityAverages
54 GO
132 %
Connected. (1/1)
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 49 Col 91 Ch 86 INS

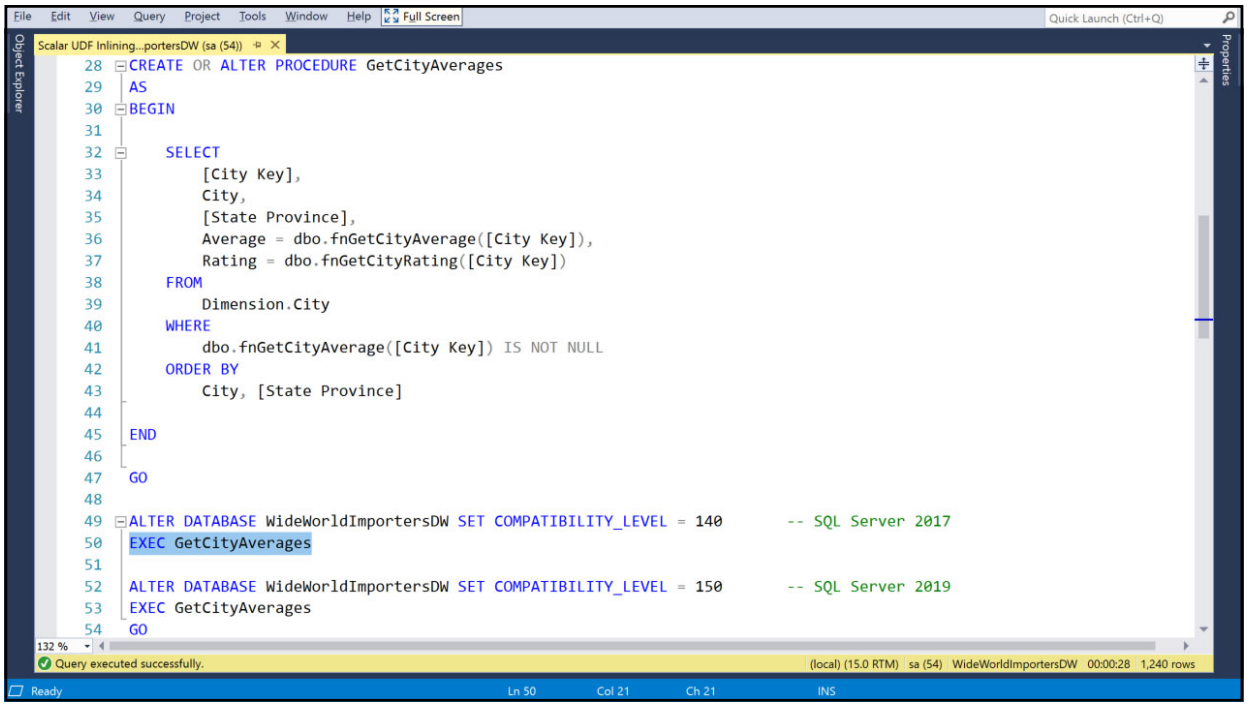

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
41 dbo.fnGetCityAverage([City Key]) IS NOT NULL
42 ORDER BY
43 City, [State Province]
44
45 END
46
47 GO
48
49 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
50 EXEC GetCityAverages
51
52 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
53 EXEC GetCityAverages
54 GO
132 %
Connected. (1/1)
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 50 Col 21 Ch 21 INS

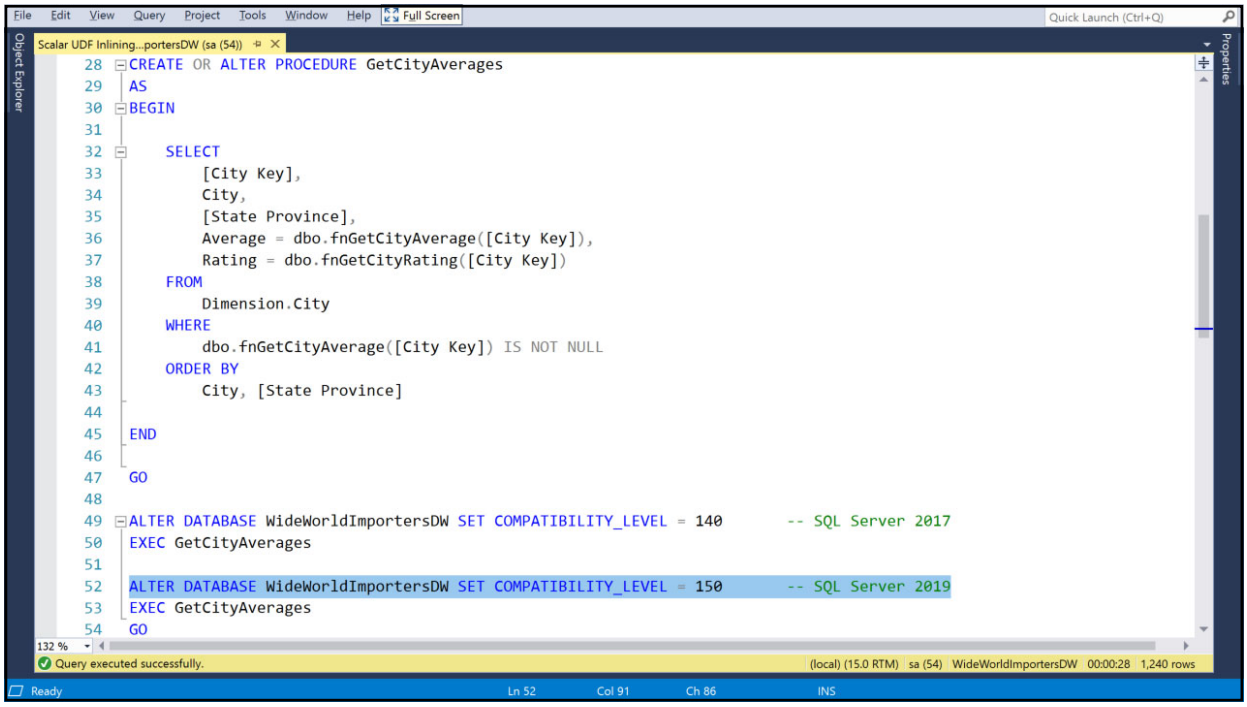

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
132 %
Results Messages Execution plan
City Key City State Province Average Rating
1 70334 Abbottsburg North Carolina 36.95 Below Average
2 101493 Abbottsburg North Carolina 46.19 Above Average
3 55498 Absecon New Jersey 41.32 Above Average
4 59913 Accomac Virginia 45.87 Above Average
5 56894 Aceitunas Puerto Rico (US Territory) 35.00 Below Average
6 81738 Aceitunas Puerto Rico (US Territory) 37.78 Below Average
7 96024 Airport Drive Missouri 41.48 Above Average
8 61111 Airport Drive Missouri 43.20 Above Average
9 72610 Akhiok Alaska 43.23 Above Average
10 67032 Alcester South Dakota 36.06 Below Average
11 64098 Alden Bridge Louisiana 40.29 Above Average
12 95748 Alden Bridge Louisiana 42.97 Above Average
13 103794 Alstead New Hampshire 35.64 Below Average
14 71783 Alstead New Hampshire 42.84 Above Average
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:28 1,240 rows
Ready Ln 50 Col 21 Ch 21 INS

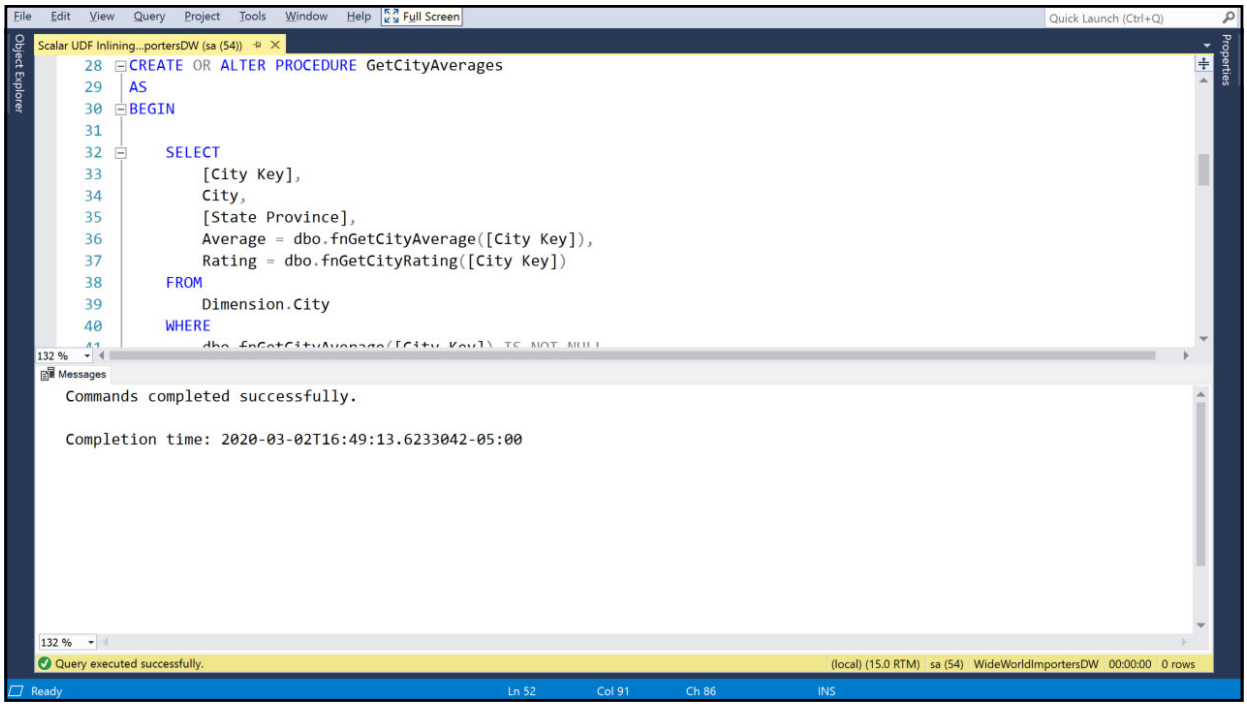

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
132 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT [City Key], City, [State Province], Average = dbo.fnGetCityAverage([City Key]), Rating = dbo.fnGetCityRating([City Key]) FROM Dimension....
SELECT
Cost: 0 %
Compute Scalar
Cost: 0 %
28.088s
1240 of
104665 (1%)
Sort
Cost: 74 %
26.949s
1240 of
104665 (1%)
Filter
Cost: 1 %
26.944s
1240 of
104665 (1%)
Clustered Index S...
[City].[PK_Dimens...
Cost: 25 %
0.106s
116295 of
116295 (100%)
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:28 1,240 rows
Ready

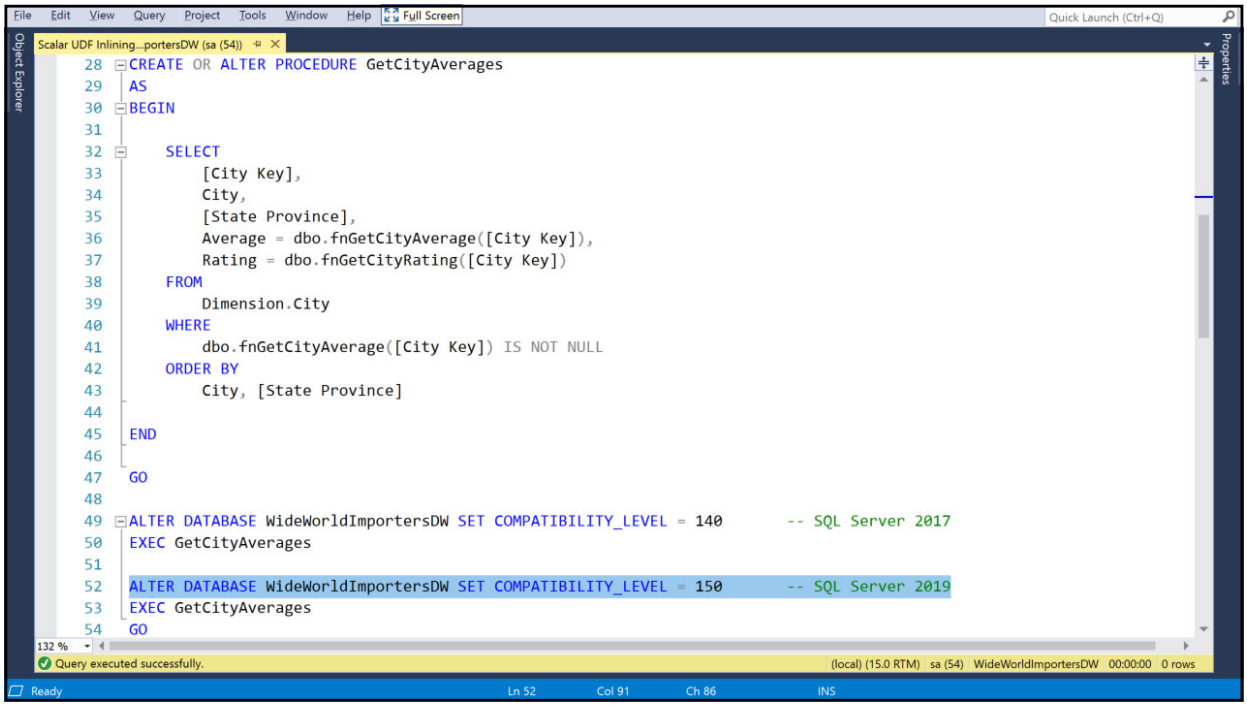

Scalar UDF Inlining...portersDW (sa (54))
CREATE OR ALTER PROCEDURE GetCityAverages
AS
BEGIN
SELECT
[City Key],
City,
SELECT
Cached plan size 24 KB
Estimated Operator Cost 0 (0%)
Degree of Parallelism 0
Estimated Subtree Cost 10.7998
Memory Grant 20600
Estimated Number of Rows 104665
Statement
SELECT [City Key], City, [State Province], Average = dbo.fnGetCityAverage([City Key]), Rating = dbo.fnGetCityRating([City Key]) FROM Dimension.City WHERE dbo.fnGetCityAverage([City Key]) IS NOT NULL ORDER BY City, [State Province]
Warnings
The query memory grant detected "ExcessiveGrant", which may impact the reliability. Grant size: Initial 20600 KB, Final 20600 KB, Used 120 KB.
Filter
Cost: 1 %
26.944s
1240 of
104665 (1%)
Clustered Index S...
[City].[PK_Dimens...
Cost: 25 %
0.106s
116295 of
116295 (100%)
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:28 1,240 rows
Ready

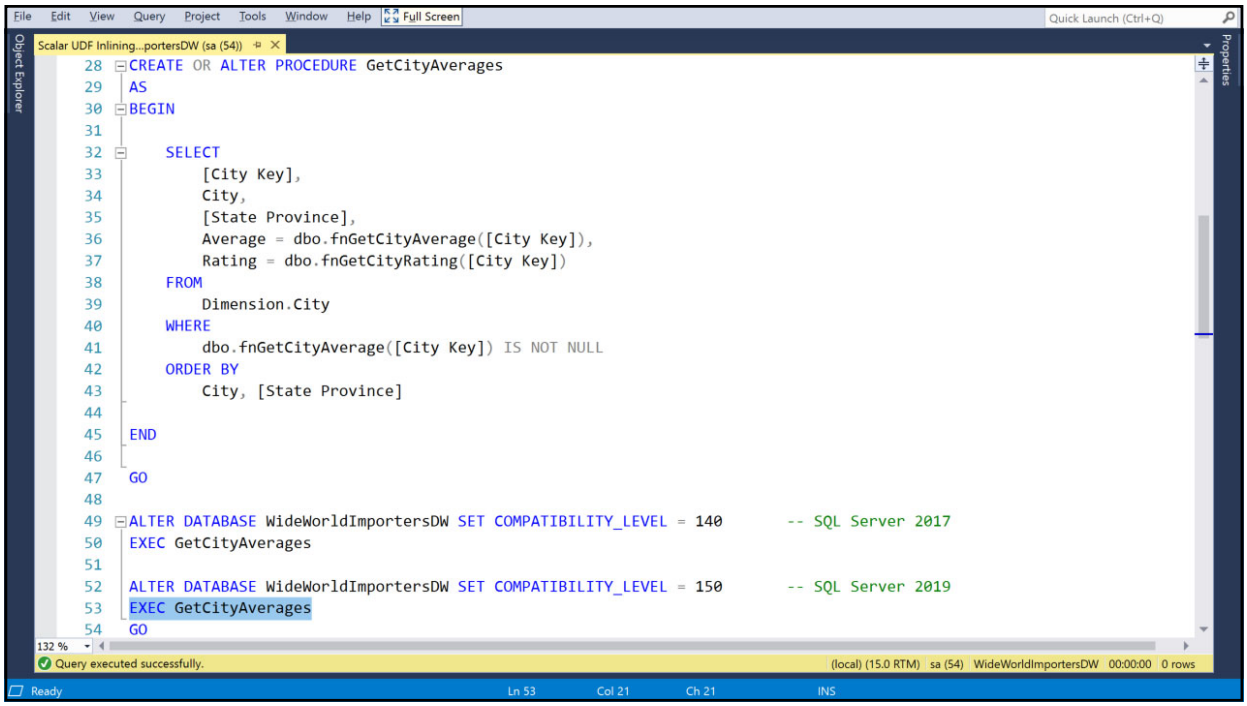

Scalar UDF Inlining...portersDW (sa (54))
CREATE OR ALTER PROCEDURE GetCityAverages
AS
BEGIN
SELECT
[City Key],
City,
[State Province],
Average = dbo.fnGetCityAverage([City Key]),
Rating = dbo.fnGetCityRating([City Key])
FROM
Dimension.City
WHERE
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT [City Key], City, [State Province], Average = dbo.fnGetCityAverage([City Key]), Rating = dbo.fnGetCityRating([City Key]) FROM Dimension....
Compute Scalar
Cost: 0 %
28.088s
1240 of
104665 (1%)
Sort
Cost: 74 %
26.949s
1240 of
104665 (1%)
Filter
Cost: 1 %
26.944s
1240 of
104665 (1%)
Clustered Index S...
[City].[PK_Dimens...
Cost: 25 %
0.106s
116295 of
116295 (100%)
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:28 1,240 rows
Ready

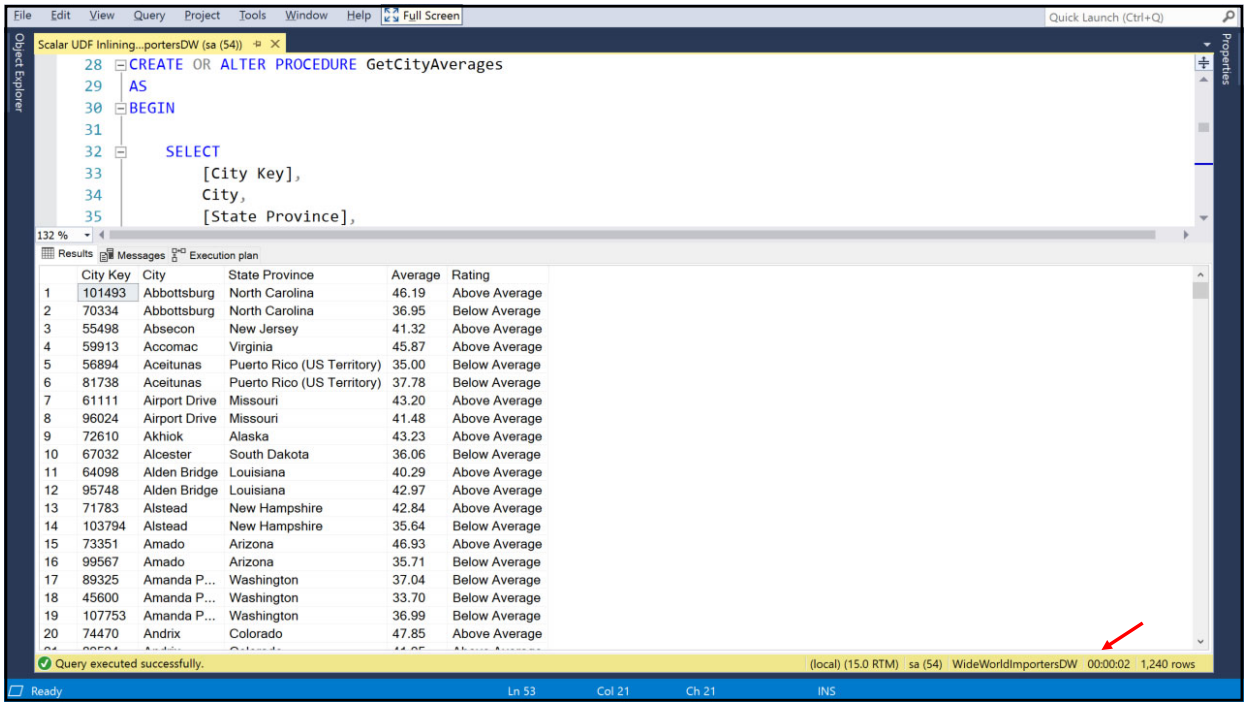

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
41 dbo.fnGetCityAverage([City Key]) IS NOT NULL
42 ORDER BY
43 City, [State Province]
44
45 END
46
47 GO
48
49 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
50 EXEC GetCityAverages
51
52 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
53 EXEC GetCityAverages
54 GO
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:28 1,240 rows
Ready
Ln 50 Col 21 Ch 21 INS

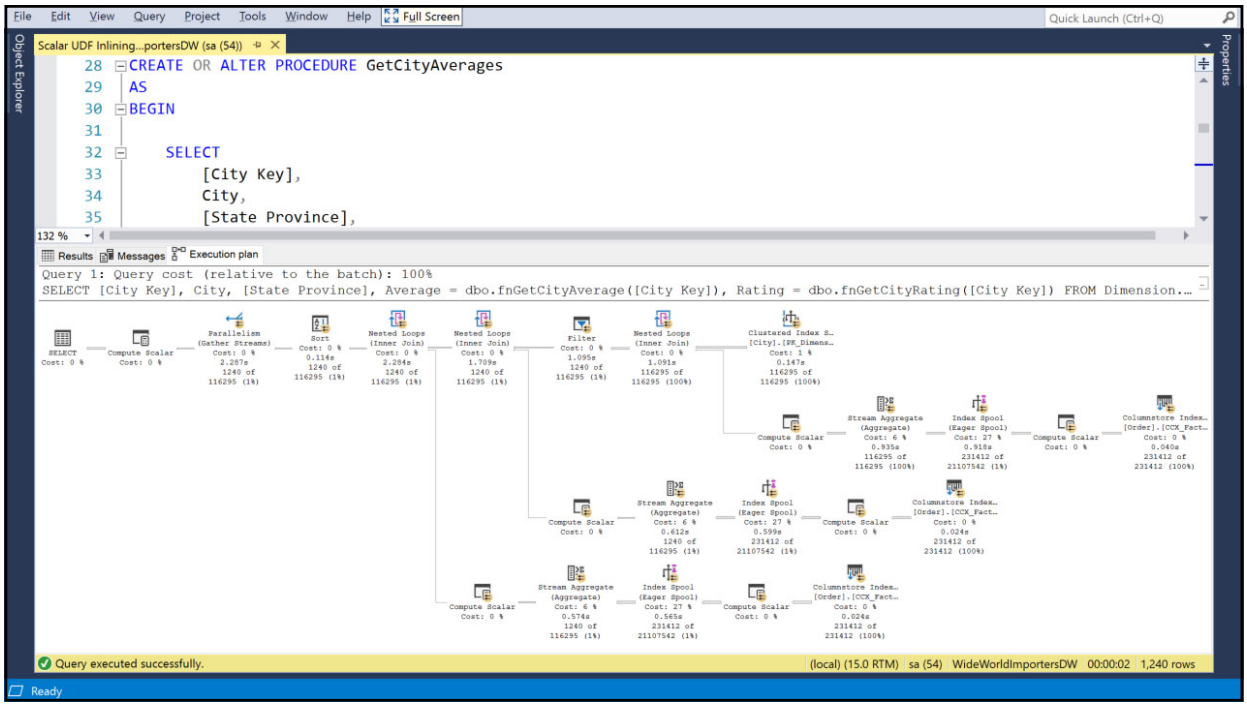

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
41 dbo.fnGetCityAverage([City Key]) IS NOT NULL
42 ORDER BY
43 City, [State Province]
44
45 END
46
47 GO
48
49 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
50 EXEC GetCityAverages
51
52 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
53 EXEC GetCityAverages
54 GO
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:28 1,240 rows
Ready
Ln 52 Col 91 Ch 86 INS

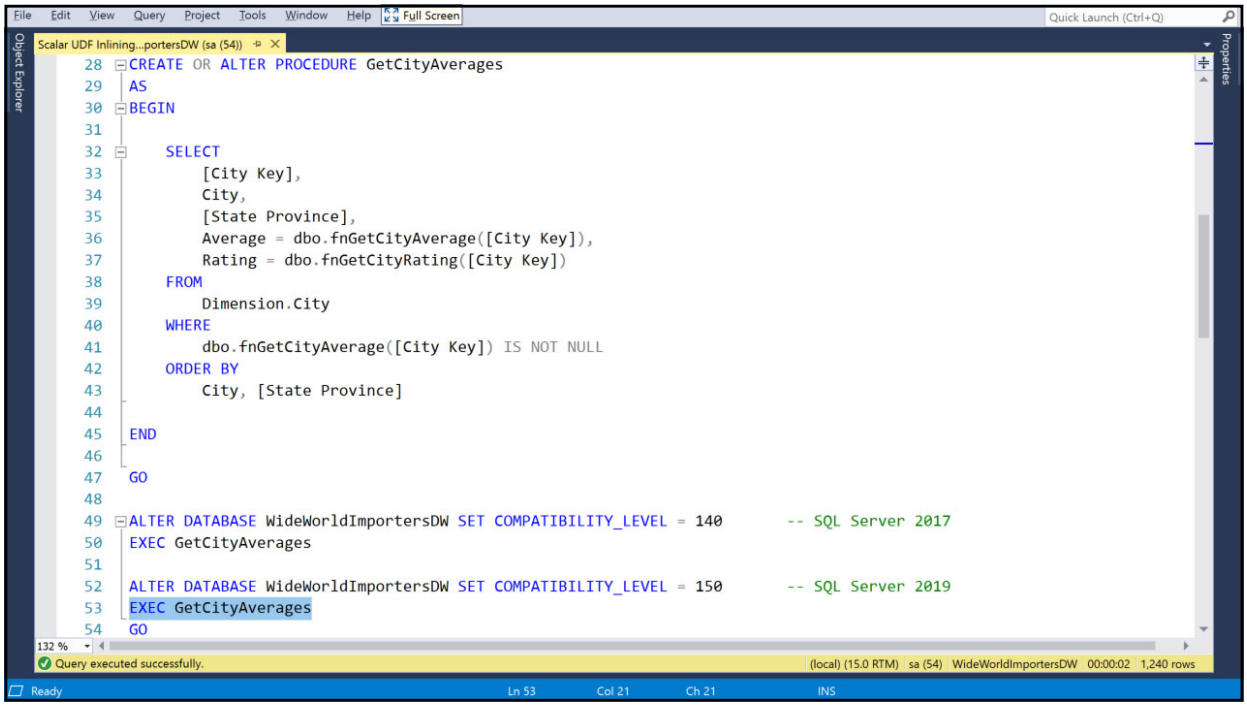

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
132 %
Messages
Commands completed successfully.
Completion time: 2020-03-02T16:49:13.6233042-05:00
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 52 Col 91 Ch 86 INS

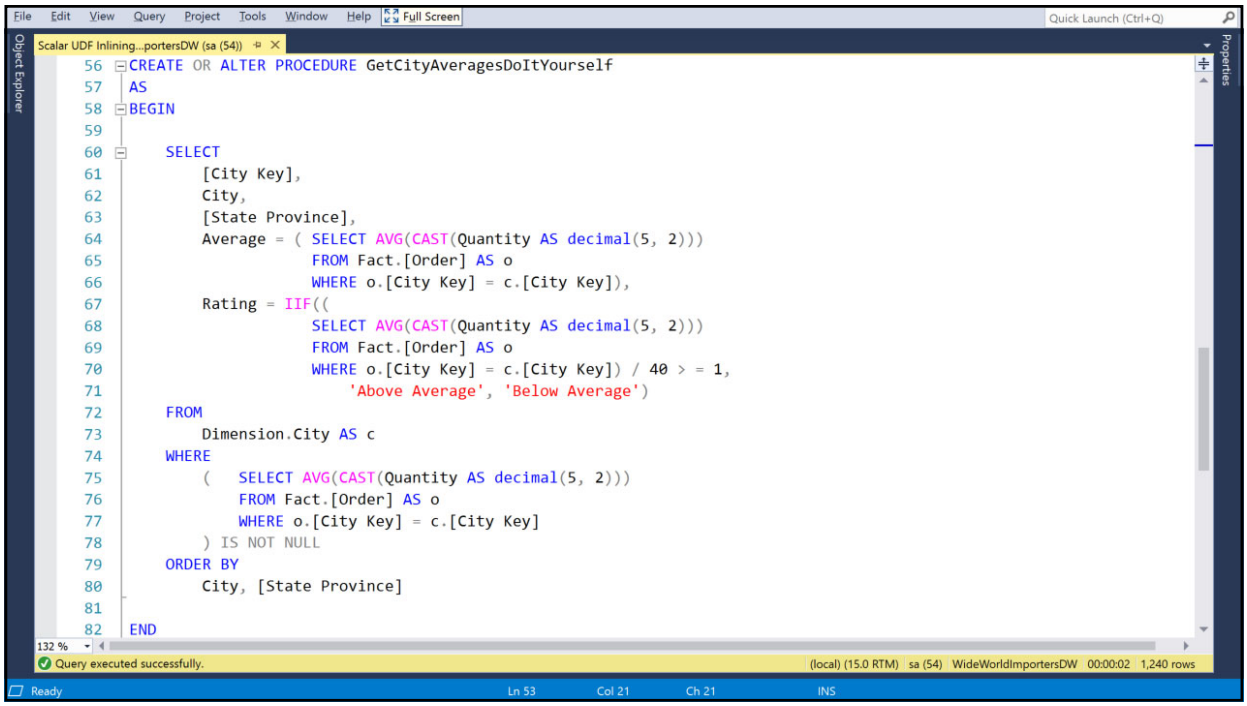

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
41 dbo.fnGetCityAverage([City Key]) IS NOT NULL
42 ORDER BY
43 City, [State Province]
44
45 END
46
47 GO
48
49 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
50 EXEC GetCityAverages
51
52 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
53 EXEC GetCityAverages
54 GO
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 52 Col 91 Ch 86 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
41 dbo.fnGetCityAverage([City Key]) IS NOT NULL
42 ORDER BY
43 City, [State Province]
44
45 END
46
47 GO
48
49 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
50 EXEC GetCityAverages
51
52 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
53 EXEC GetCityAverages
54 GO
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready Ln 53 Col 21 Ch 21 INS

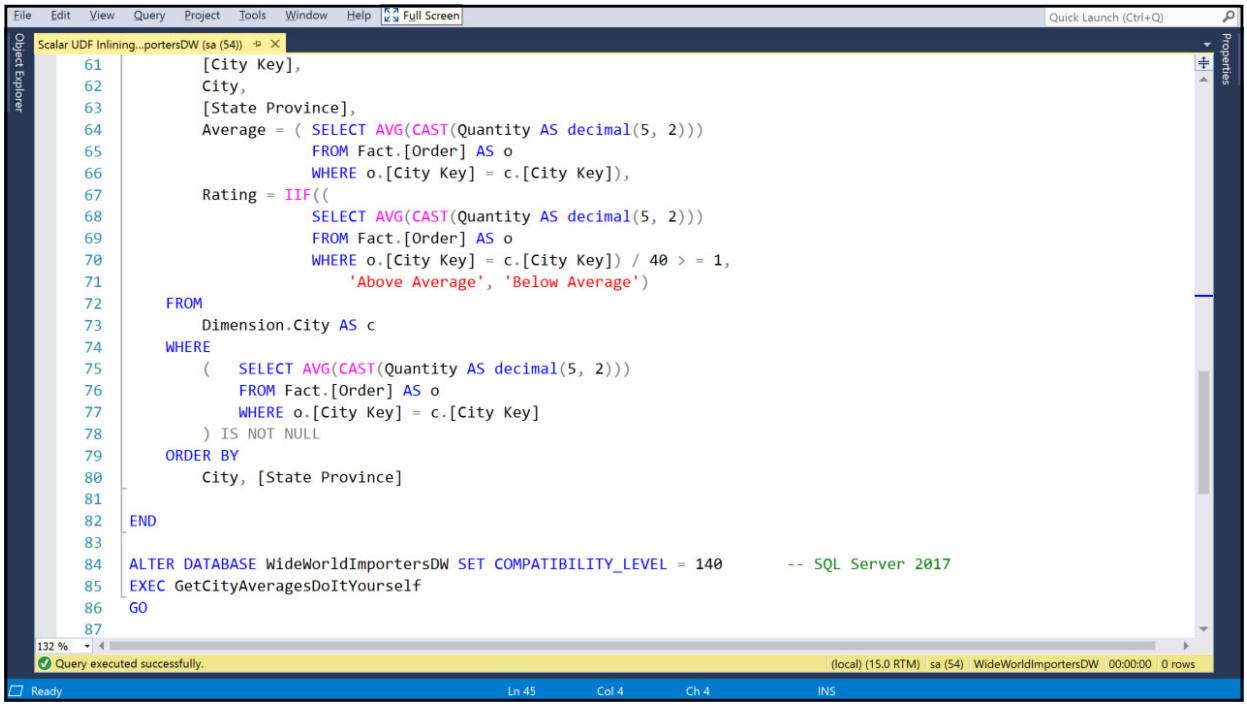

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
132 %
Results Messages Execution plan
City Key City State Province Average Rating
1 101493 Abbottsburg North Carolina 46.19 Above Average
2 70334 Abbottsburg North Carolina 36.95 Below Average
3 55498 Absecon New Jersey 41.32 Above Average
4 59913 Accomac Virginia 45.87 Above Average
5 56894 Aceitunas Puerto Rico (US Territory) 35.00 Below Average
6 81738 Aceitunas Puerto Rico (US Territory) 37.78 Below Average
7 61111 Airport Drive Missouri 43.20 Above Average
8 96024 Airport Drive Missouri 41.48 Above Average
9 72610 Akhiok Alaska 43.23 Above Average
10 67032 Alcester South Dakota 36.06 Below Average
11 64098 Alden Bridge Louisiana 40.29 Above Average
12 95748 Alden Bridge Louisiana 42.97 Above Average
13 71783 Alstead New Hampshire 42.84 Above Average
14 103794 Alstead New Hampshire 35.64 Below Average
15 73351 Amado Arizona 46.93 Above Average
16 99567 Amado Arizona 35.71 Below Average
17 89325 Amanda P... Washington 37.04 Below Average
18 45600 Amanda P... Washington 33.70 Below Average
19 107753 Amanda P... Washington 36.99 Below Average
20 74470 Andrix Colorado 47.85 Above Average
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:02 1,240 rows
Ready Ln 53 Col 21 Ch 21 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
132 %
Results Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT [City Key], City, [State Province], Average = dbo.fnGetCityAverage([City Key]), Rating = dbo.fnGetCityRating([City Key]) FROM Dimension...
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:02 1,240 rows
Ready

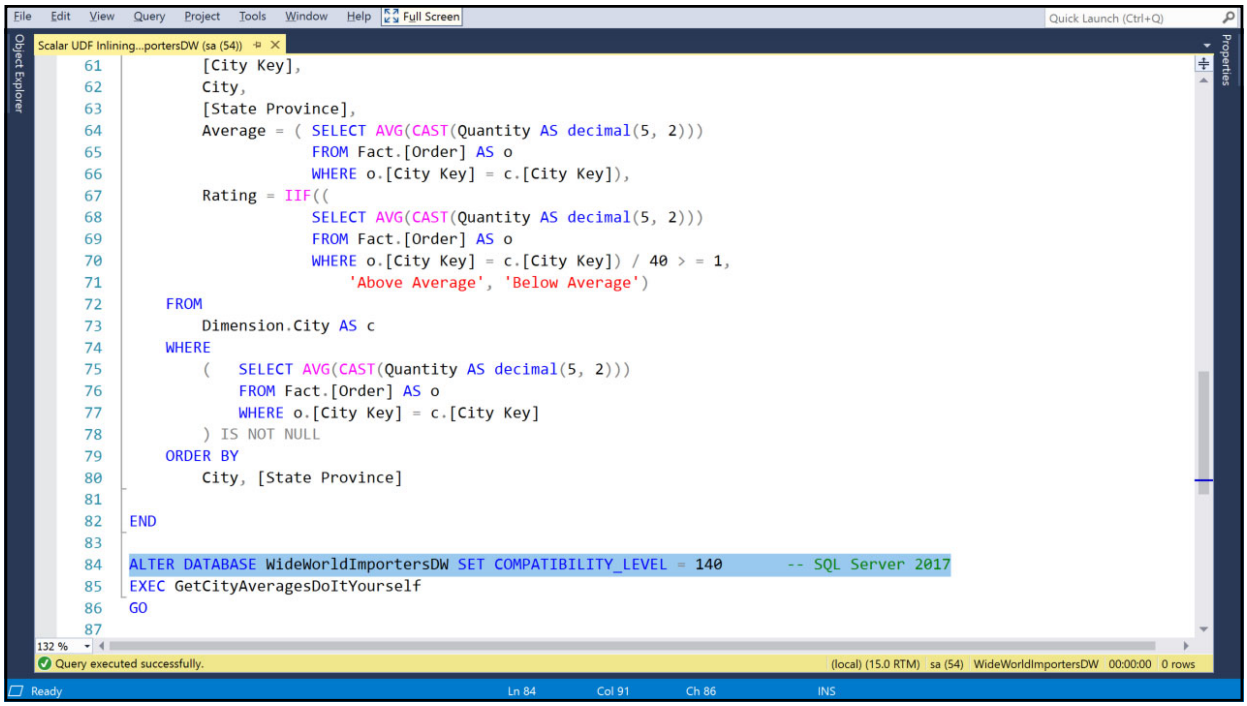

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
28 CREATE OR ALTER PROCEDURE GetCityAverages
29 AS
30 BEGIN
31
32 SELECT
33 [City Key],
34 City,
35 [State Province],
36 Average = dbo.fnGetCityAverage([City Key]),
37 Rating = dbo.fnGetCityRating([City Key])
38 FROM
39 Dimension.City
40 WHERE
41 dbo.fnGetCityAverage([City Key]) IS NOT NULL
42 ORDER BY
43 City, [State Province]
44
45 END
46
47 GO
48
49 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
50 EXEC GetCityAverages
51
52 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
53 EXEC GetCityAverages
54 GO
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:02 1,240 rows
Ready Ln 53 Col 21 Ch 21 INS

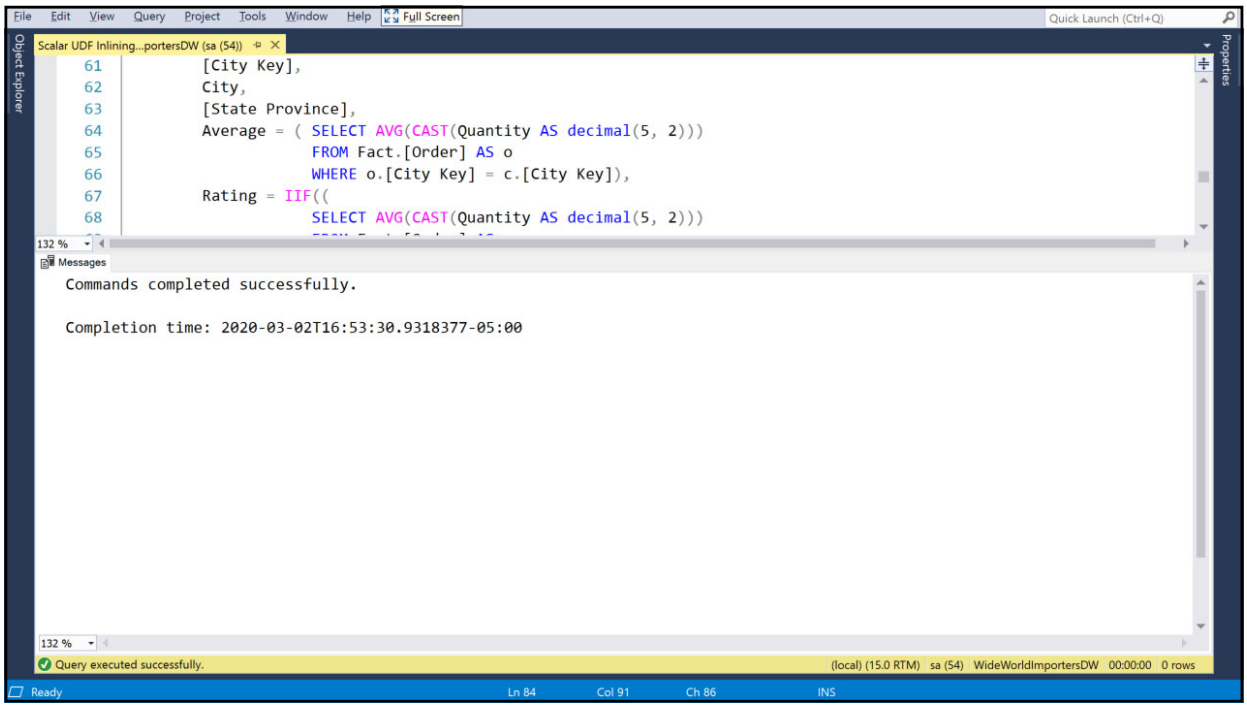


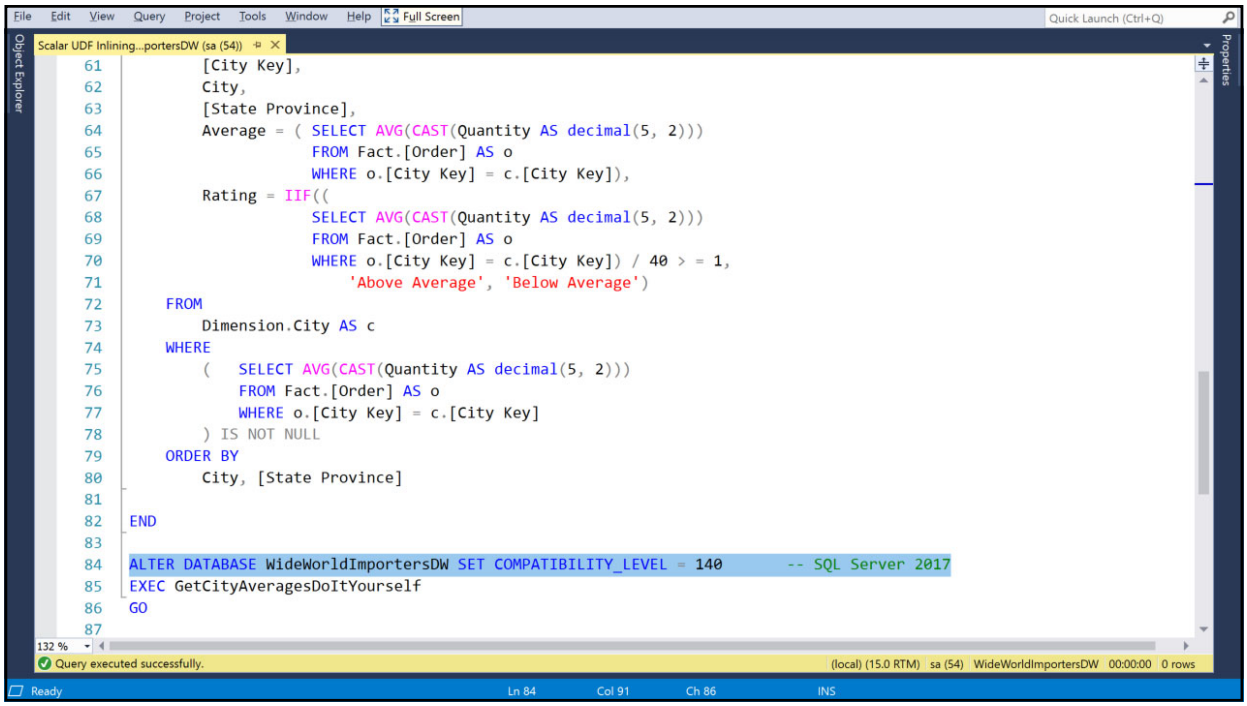
```
CREATE OR ALTER PROCEDURE GetCityAveragesDoItYourself
AS
BEGIN

    SELECT
        [City Key],
        City,
        [State Province],
        Average = ( SELECT AVG(CAST(Quantity AS decimal(5, 2)))
                    FROM Fact.[Order] AS o
                    WHERE o.[City Key] = c.[City Key]),
        Rating = IIF((
                    SELECT AVG(CAST(Quantity AS decimal(5, 2)))
                    FROM Fact.[Order] AS o
                    WHERE o.[City Key] = c.[City Key]) / 40 > = 1,
                        'Above Average', 'Below Average')
    FROM
        Dimension.City AS c
    WHERE
        (   SELECT AVG(CAST(Quantity AS decimal(5, 2)))
            FROM Fact.[Order] AS o
            WHERE o.[City Key] = c.[City Key]
        ) IS NOT NULL
    ORDER BY
        City, [State Province]

END
```

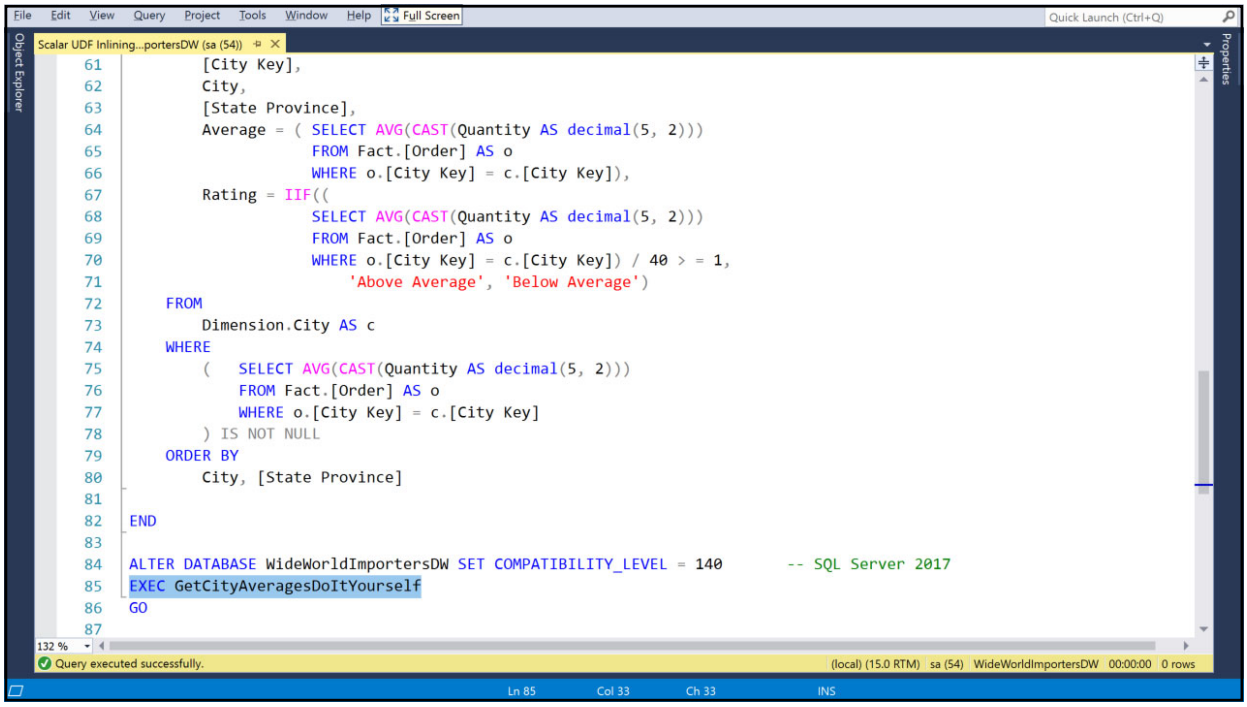


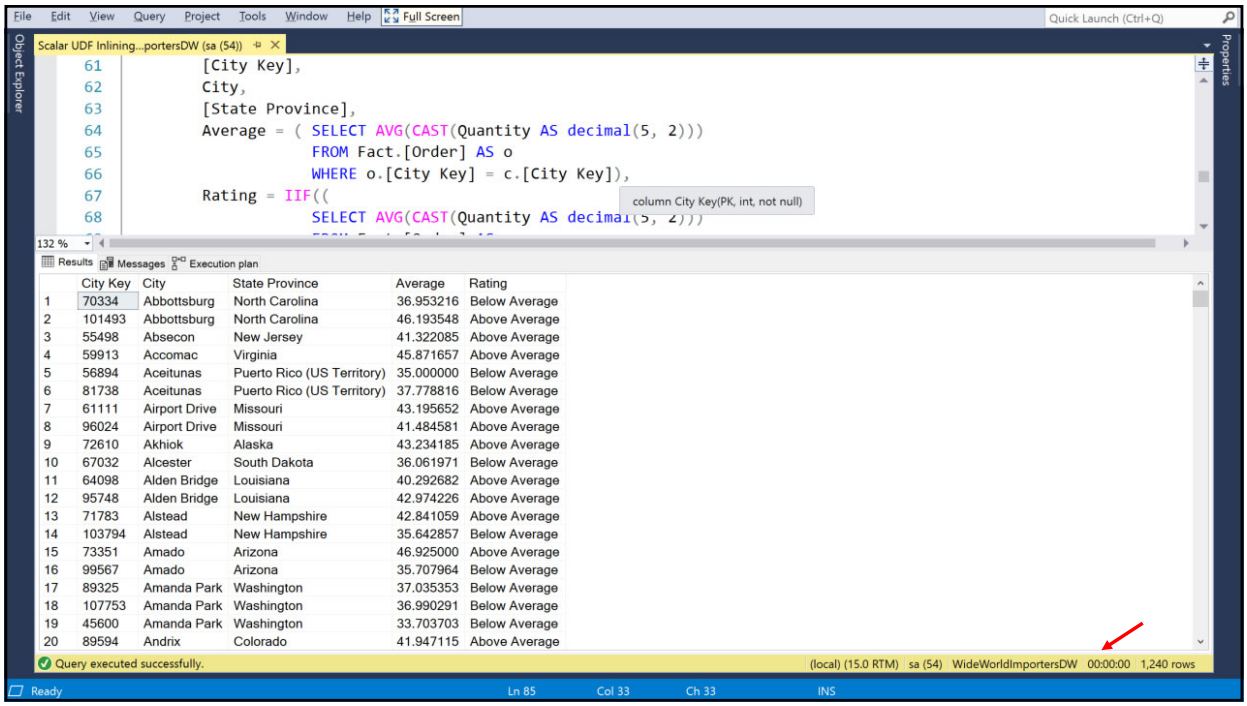
```
        [City Key],
        City,
        [State Province],
        Average = ( SELECT AVG(CAST(Quantity AS decimal(5, 2)))
                    FROM Fact.[Order] AS o
                    WHERE o.[City Key] = c.[City Key]),
        Rating = IIF((
                    SELECT AVG(CAST(Quantity AS decimal(5, 2)))
                    FROM Fact.[Order] AS o
                    WHERE o.[City Key] = c.[City Key]) / 40 > = 1,
                        'Above Average', 'Below Average')
    FROM
        Dimension.City AS c
    WHERE
        (   SELECT AVG(CAST(Quantity AS decimal(5, 2)))
            FROM Fact.[Order] AS o
            WHERE o.[City Key] = c.[City Key]
        ) IS NOT NULL
    ORDER BY
        City, [State Province]

END

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140       -- SQL Server 2017
EXEC GetCityAveragesDoItYourself
GO
```


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
[City Key],
City,
[State Province],
Average = (SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]),
Rating = IIF((
SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]) / 40 > = 1,
'Above Average', 'Below Average')
FROM
Dimension.City AS c
WHERE
(SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]
) IS NOT NULL
ORDER BY
City, [State Province]
END
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
EXEC GetCityAveragesDoItYourself
GO
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 84 Col 91 Ch 86 INS

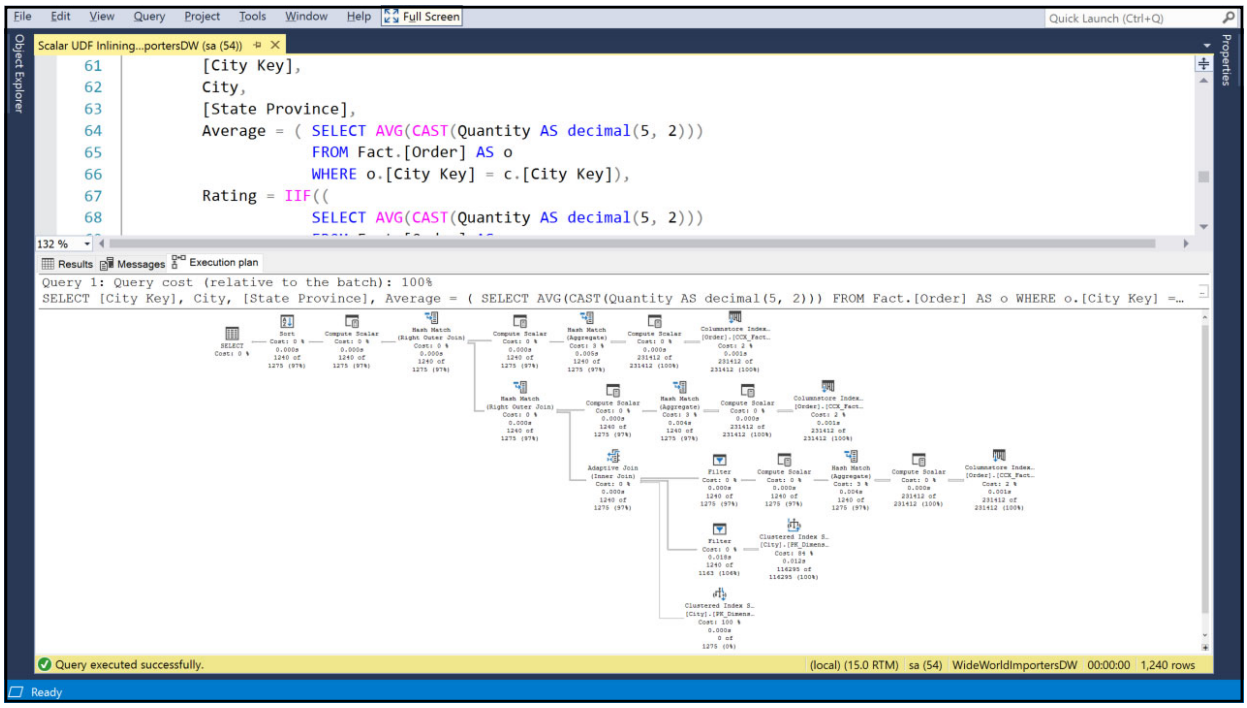

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
[City Key],
City,
[State Province],
Average = (SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]),
Rating = IIF((
SELECT AVG(CAST(Quantity AS decimal(5, 2)))
132 %
Messages
Commands completed successfully.
Completion time: 2020-03-02T16:53:30.9318377-05:00
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 84 Col 91 Ch 86 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
[City Key],
City,
[State Province],
Average = (SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]),
Rating = IIF((
SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]) / 40 > = 1,
'Above Average', 'Below Average')
FROM
Dimension.City AS c
WHERE
(SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]
) IS NOT NULL
ORDER BY
City, [State Province]
END
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
EXEC GetCityAveragesDoItYourself
GO
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ready
Ln 84 Col 91 Ch 86 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Scalar UDF Inlining...portersDW (sa (54))
Object Explorer
Properties
[City Key],
City,
[State Province],
Average = (SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]),
Rating = IIF((
SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]) / 40 > = 1,
'Above Average', 'Below Average')
FROM
Dimension.City AS c
WHERE
(SELECT AVG(CAST(Quantity AS decimal(5, 2)))
FROM Fact.[Order] AS o
WHERE o.[City Key] = c.[City Key]
) IS NOT NULL
ORDER BY
City, [State Province]
END
ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 140 -- SQL Server 2017
EXEC GetCityAveragesDoItYourself
GO
132 %
Query executed successfully.
(local) (15.0 RTM) sa (54) WideWorldImportersDW 00:00:00 0 rows
Ln 85 Col 33 Ch 33 INS



	City Key	City	State Province	Average	Rating
1	70334	Abbottsburg	North Carolina	36.953216	Below Average
2	101493	Abbottsburg	North Carolina	46.193548	Above Average
3	55498	Absecon	New Jersey	41.322085	Above Average
4	59913	Accomac	Virginia	45.871657	Above Average
5	56894	Aceitunas	Puerto Rico (US Territory)	35.000000	Below Average
6	81738	Aceitunas	Puerto Rico (US Territory)	37.778816	Below Average
7	61111	Airport Drive	Missouri	43.195652	Above Average
8	96024	Airport Drive	Missouri	41.484581	Above Average
9	72610	Akhiok	Alaska	43.234185	Above Average
10	67032	Alcester	South Dakota	36.061971	Below Average
11	64098	Alden Bridge	Louisiana	40.292682	Above Average
12	95748	Alden Bridge	Louisiana	42.974226	Above Average
13	71783	Alstead	New Hampshire	42.841059	Above Average
14	103794	Alstead	New Hampshire	35.642857	Below Average
15	73351	Amado	Arizona	46.925000	Above Average
16	99567	Amado	Arizona	35.707964	Below Average
17	89325	Amanda Park	Washington	37.035353	Below Average
18	107753	Amanda Park	Washington	36.990291	Below Average
19	45600	Amanda Park	Washington	33.703703	Below Average
20	89594	Andrix	Colorado	41.947115	Above Average

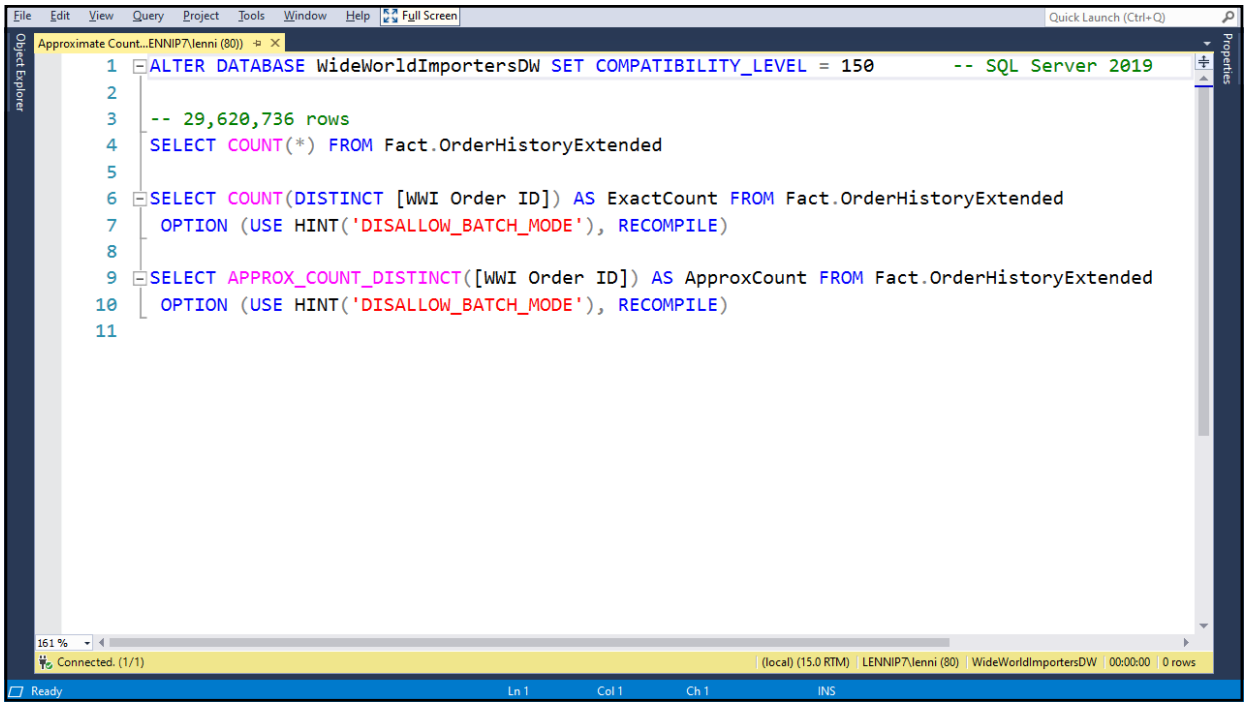


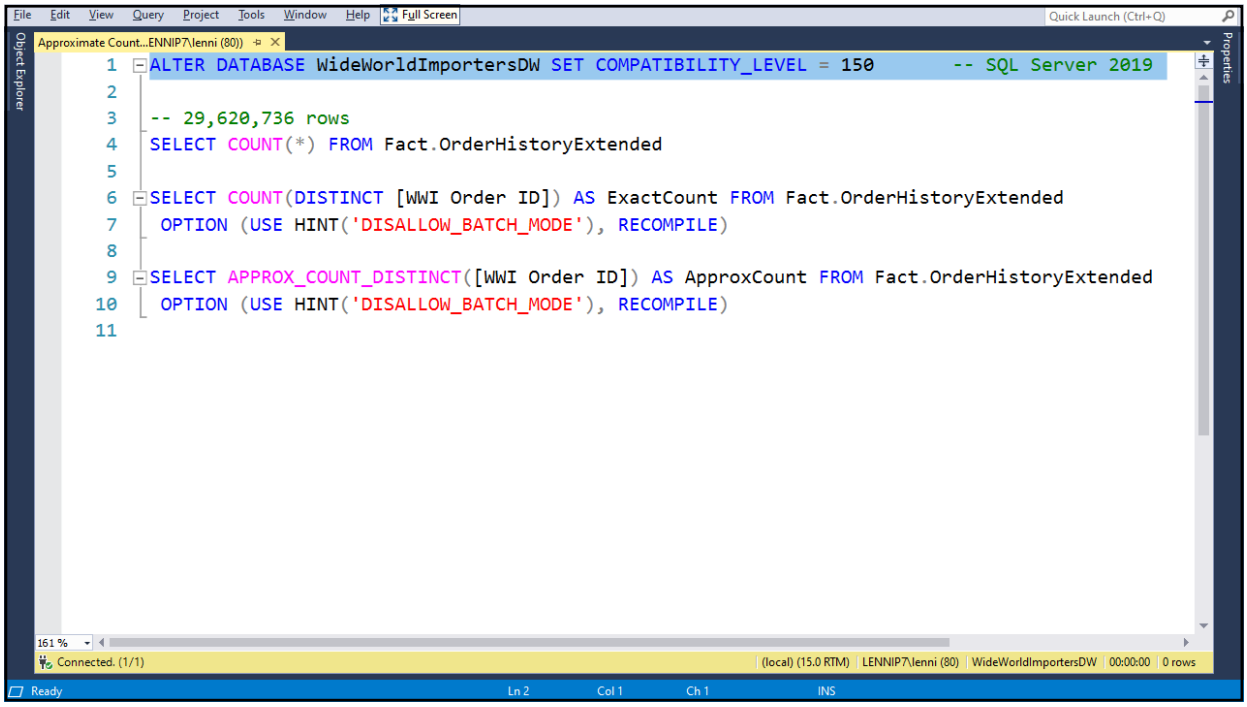
Approximate Count Distinct

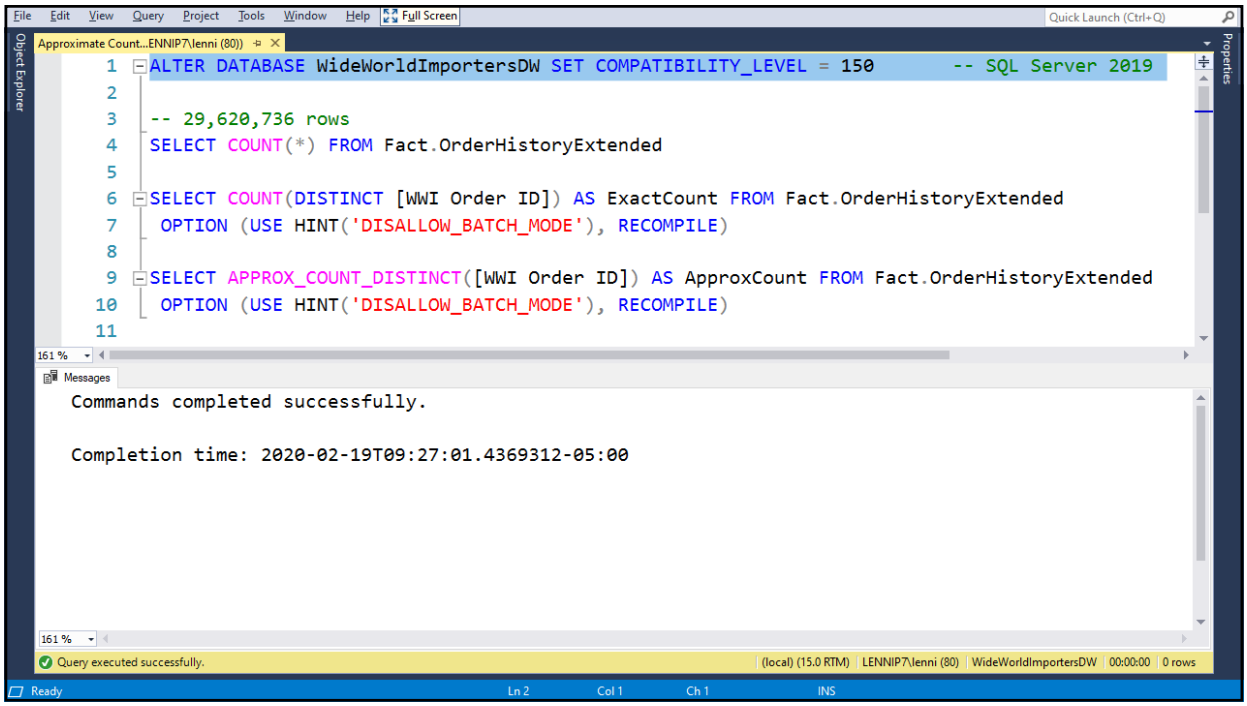
- New function APPROX_COUNT_DISTINCT()
  - Get approximate number of unique non-null
  - Similar to COUNT(DISTINCT)
- When precision doesn't matter
  - Guarantees up to a 2% error rate within a 97% probability
- It's usually faster
  - Always uses significantly less memory
  - Spillover to disk far less likely
- Ideal conditions
  - Millions of rows or more, *and t*here are many distinct values

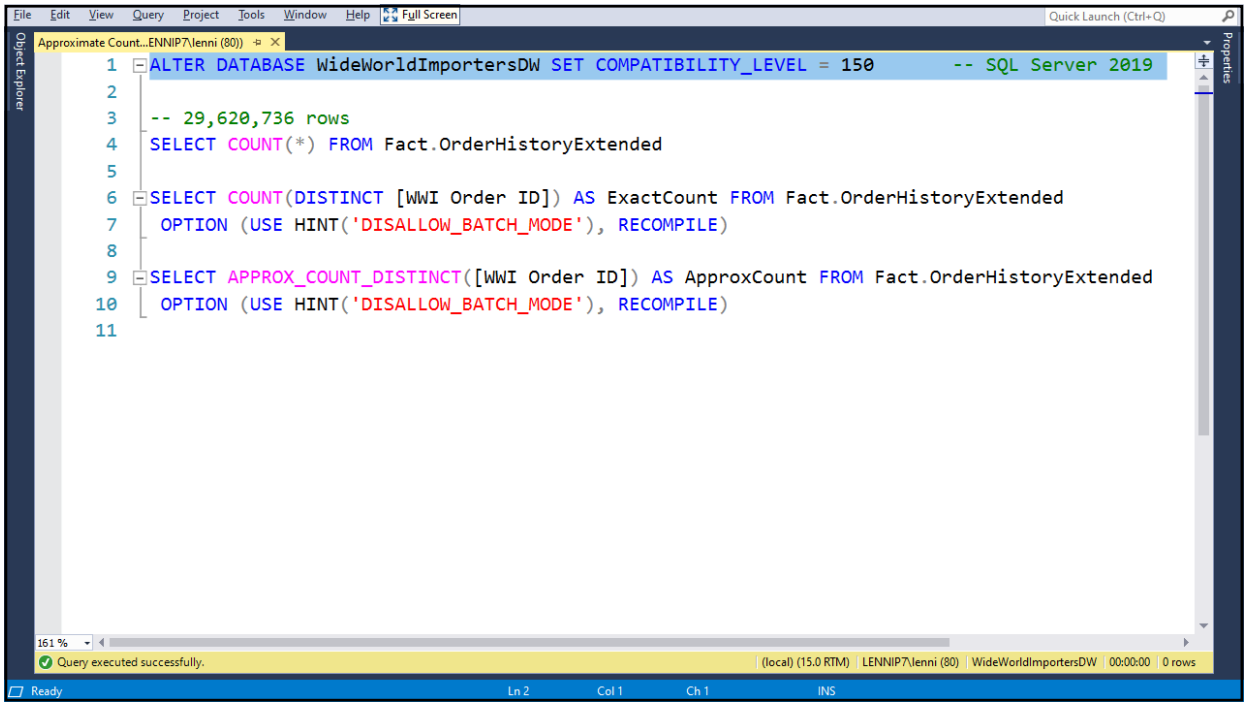
Approximate Count Distinct

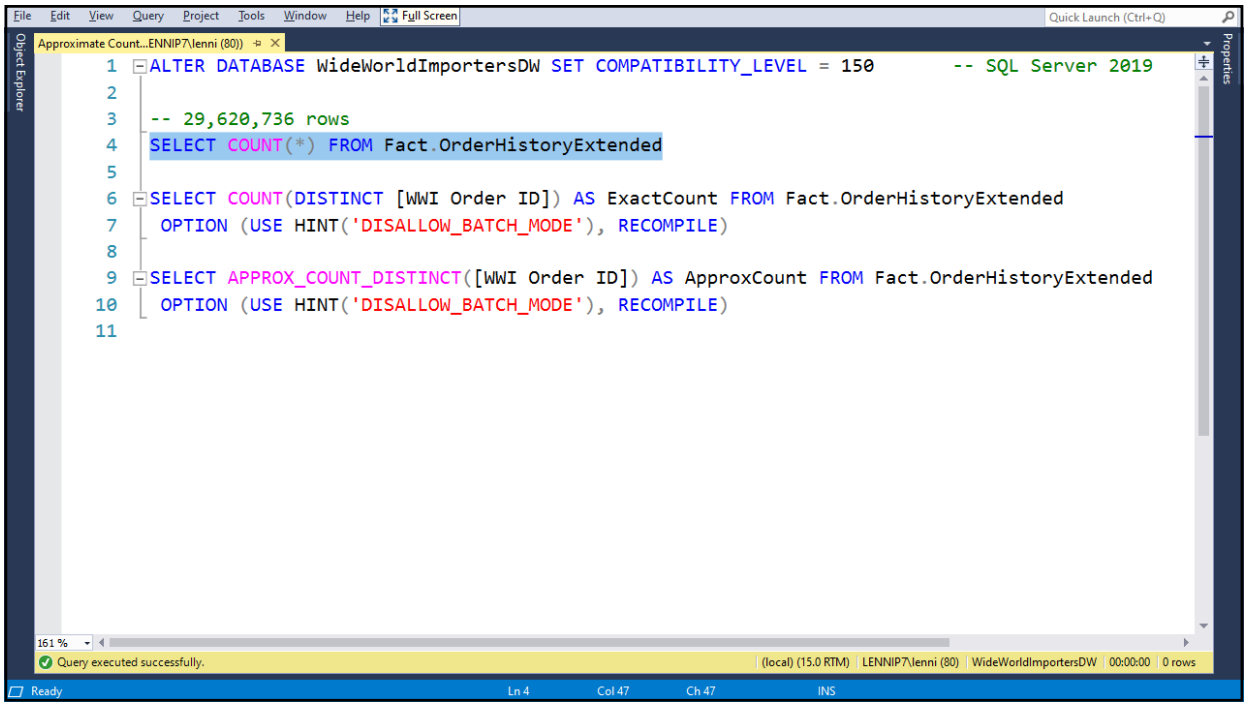
demo

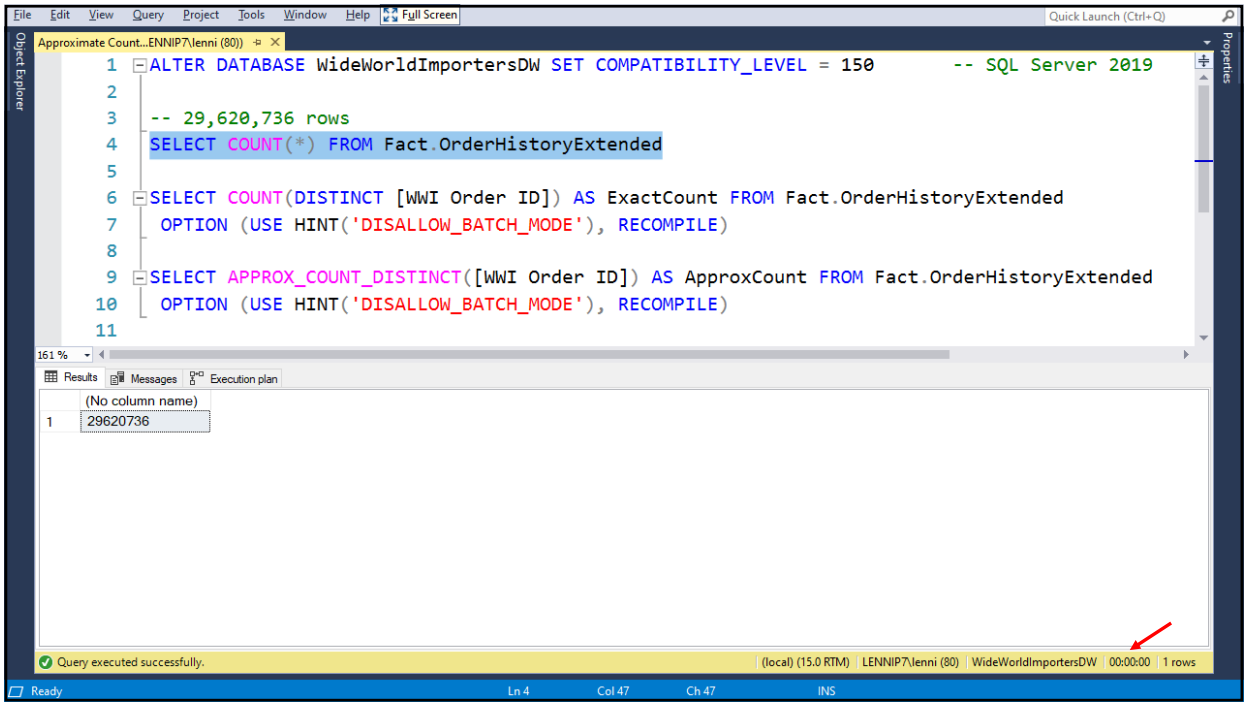

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Connected. (1/1)
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:00 0 rows
Ready Ln 1 Col 1 Ch 1 INS

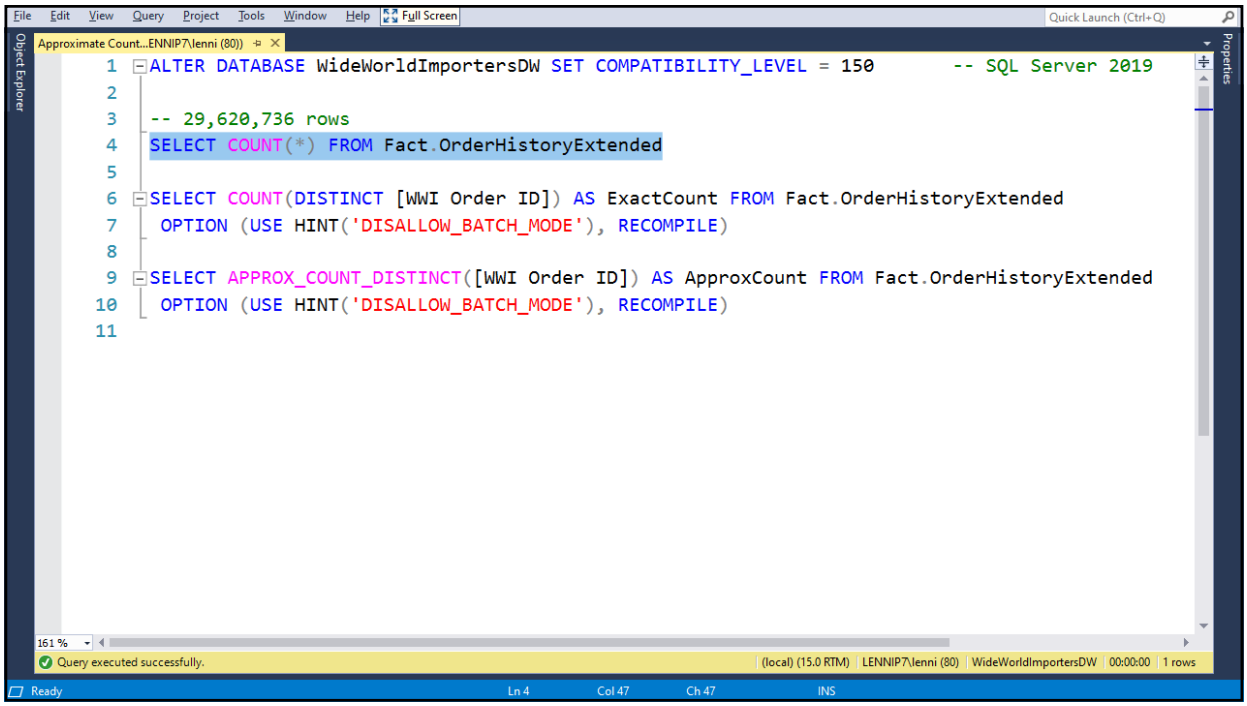

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Connected. (1/1)
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:00 0 rows
Ready Ln 2 Col 1 Ch 1 INS

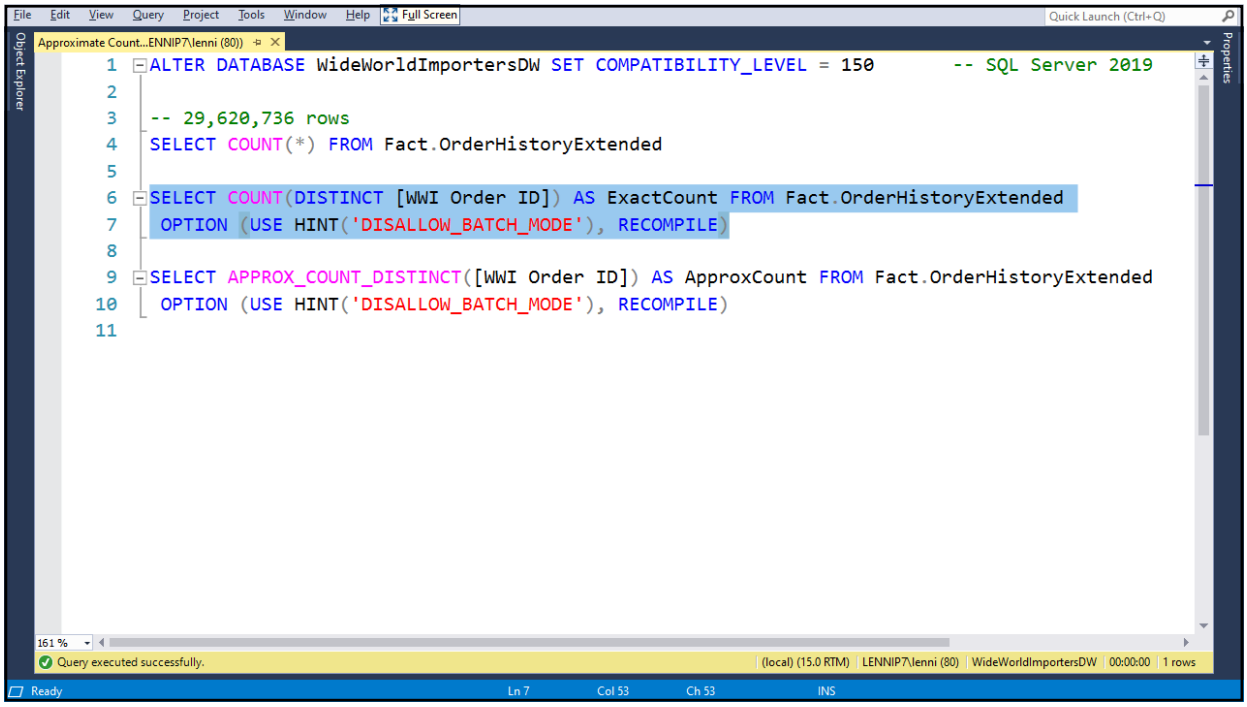

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Messages
Commands completed successfully.
Completion time: 2020-02-19T09:27:01.4369312-05:00
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:00 0 rows
Ready Ln 2 Col 1 Ch 1 INS

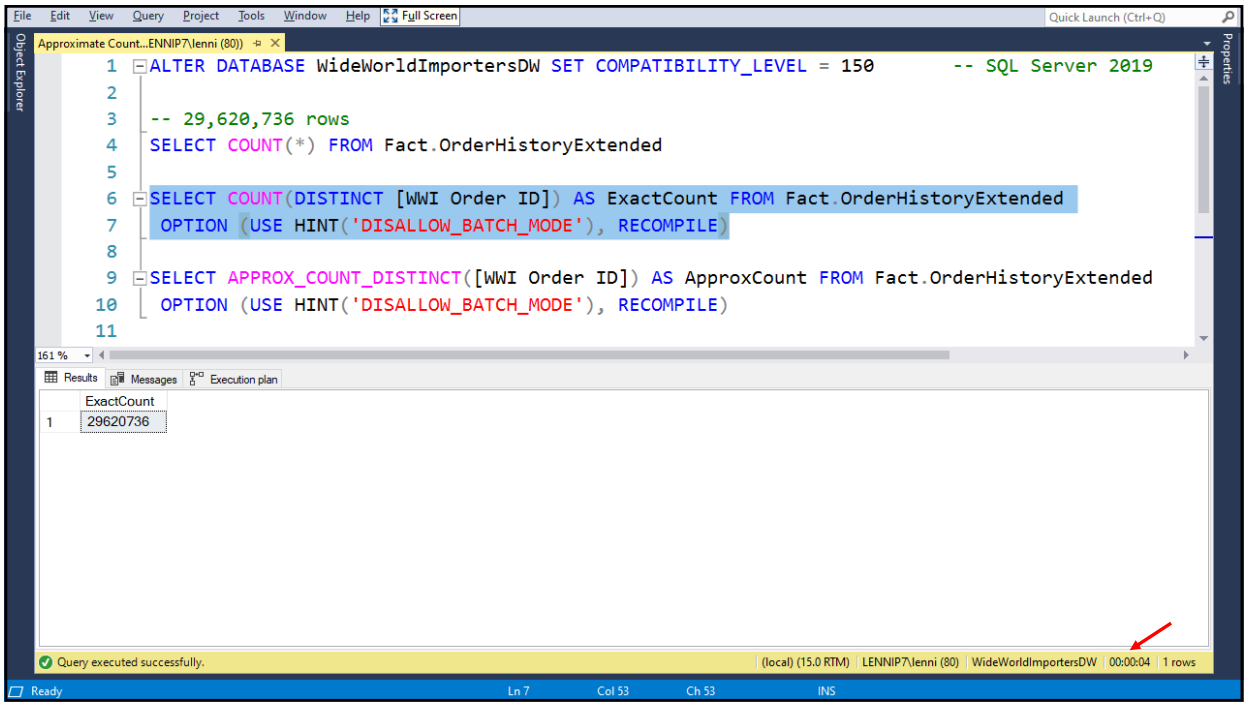

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:00 0 rows
Ready Ln 2 Col 1 Ch 1 INS

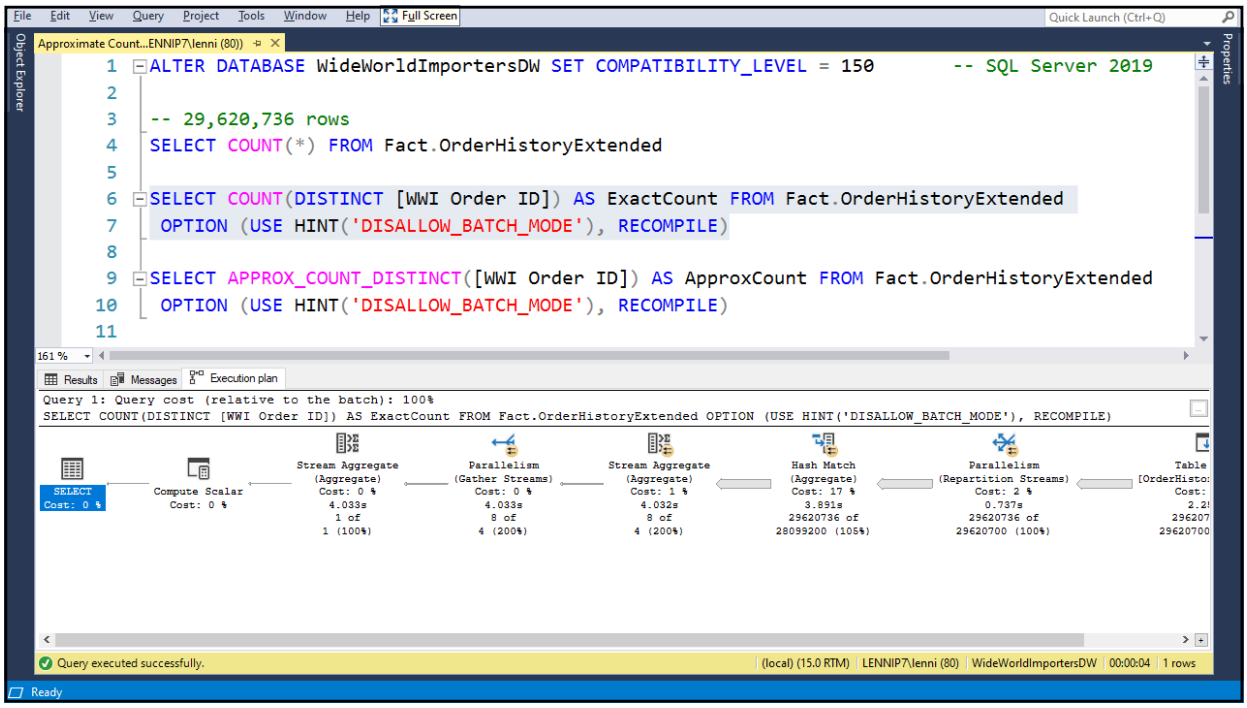

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:00 0 rows
Ready Ln 4 Col 47 Ch 47 INS

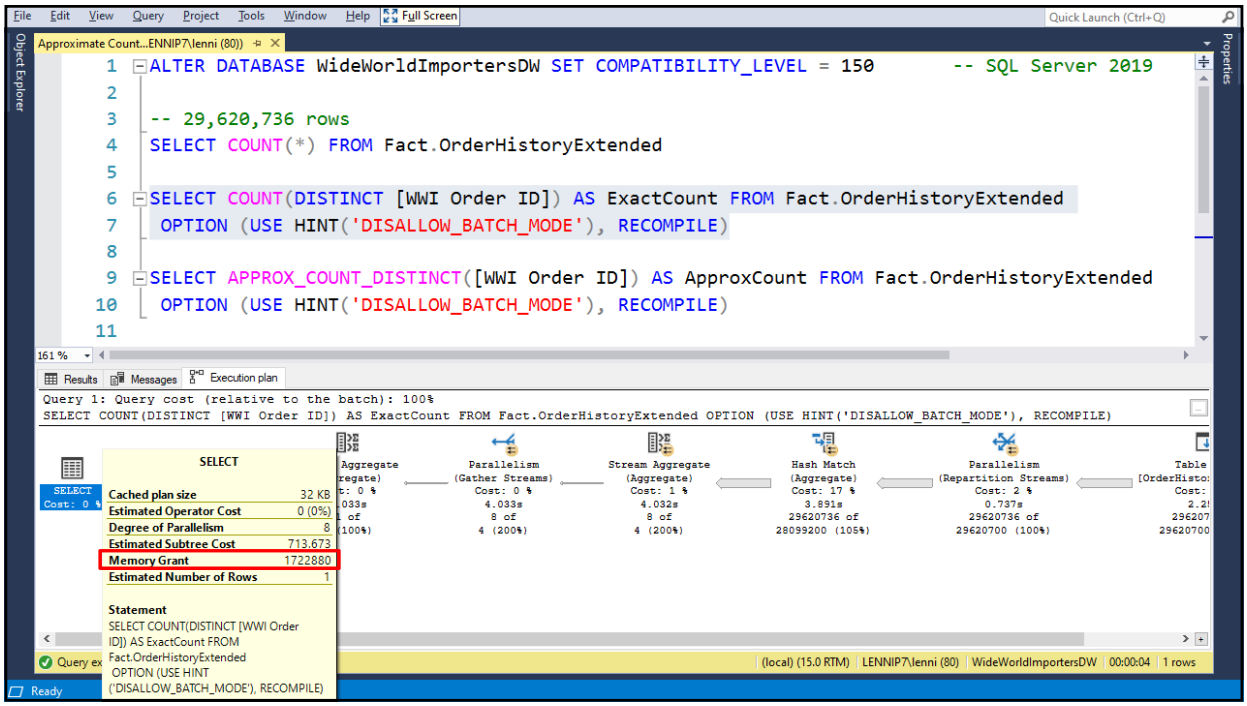

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Results Messages Execution plan
(No column name)
1 29620736
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:00 1 rows
Ready Ln 4 Col 47 Ch 47 INS

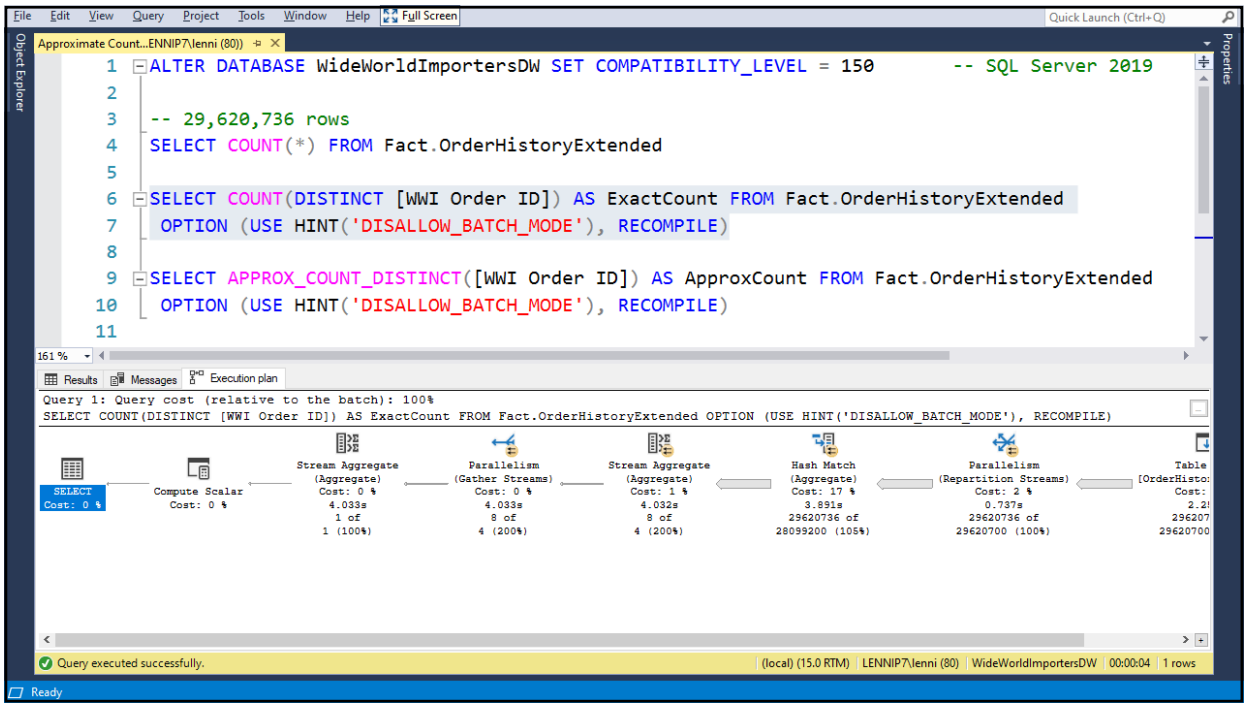

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:00 1 rows
Ready Ln 4 Col 47 Ch 47 INS

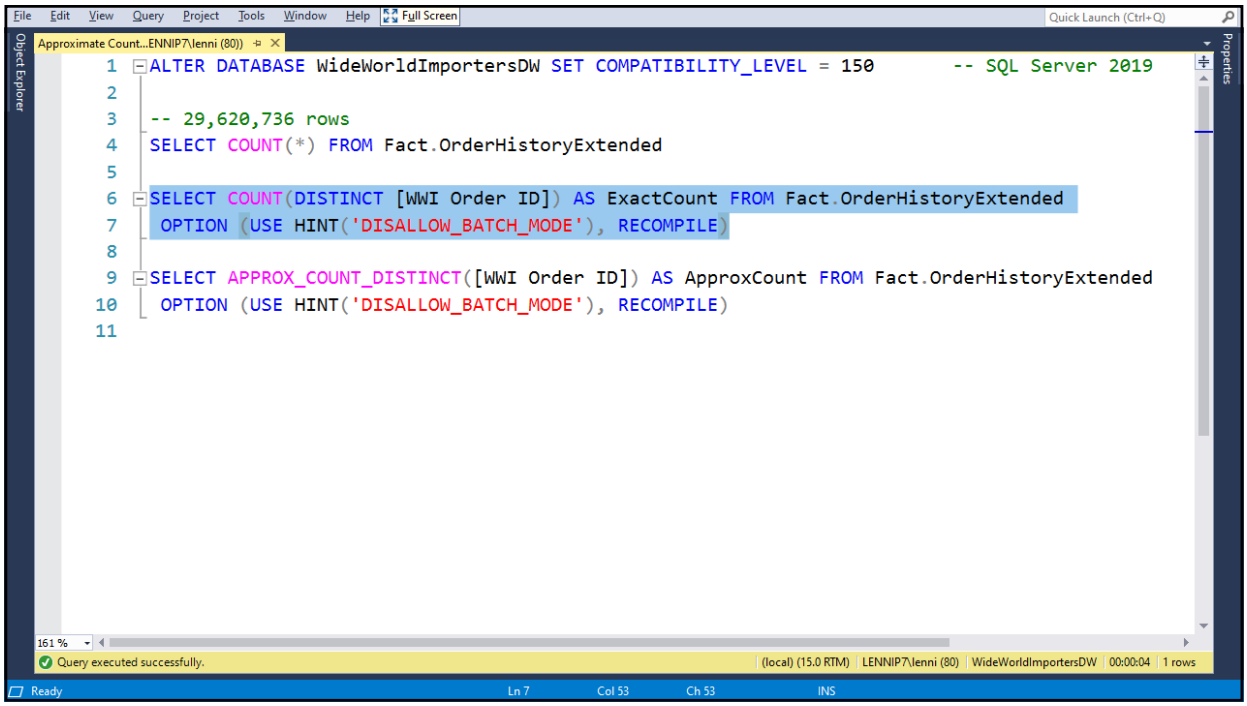

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:00 1 rows
Ready Ln 7 Col 53 Ch 53 INS

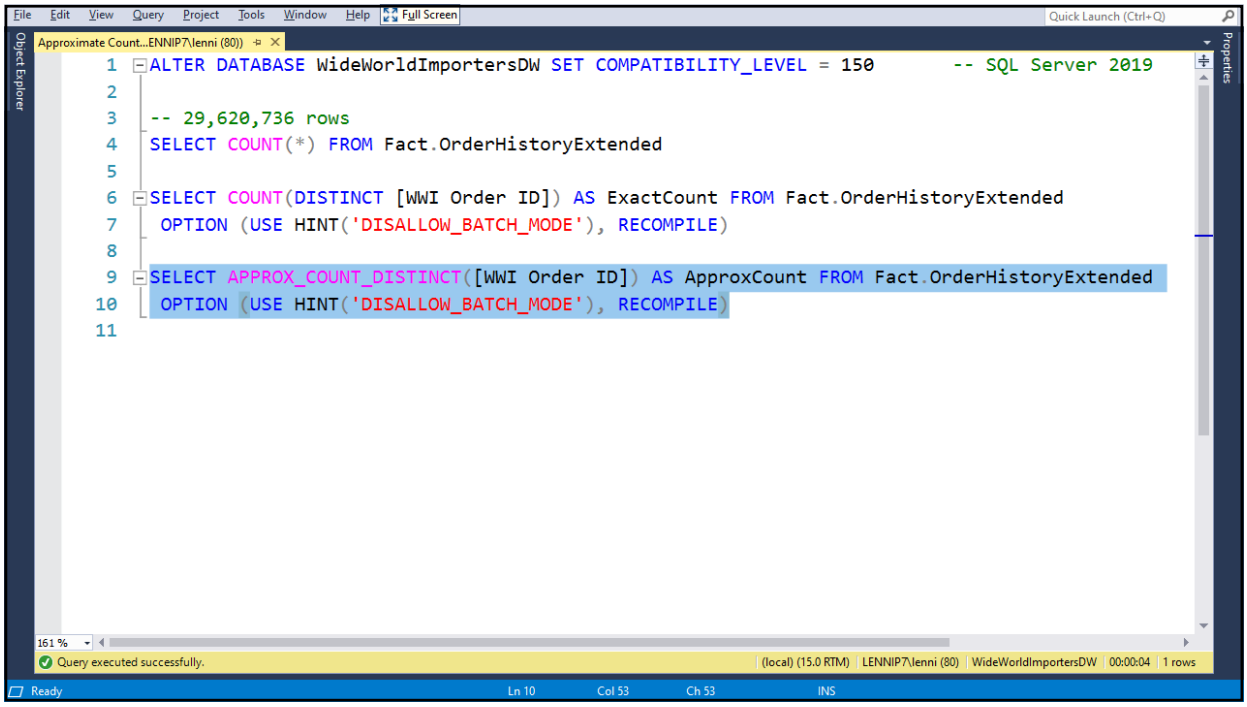

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Results
Messages
Execution plan
ExactCount
1 29620736
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:04 1 rows
Ready
Ln 7 Col 53 Ch 53 INS

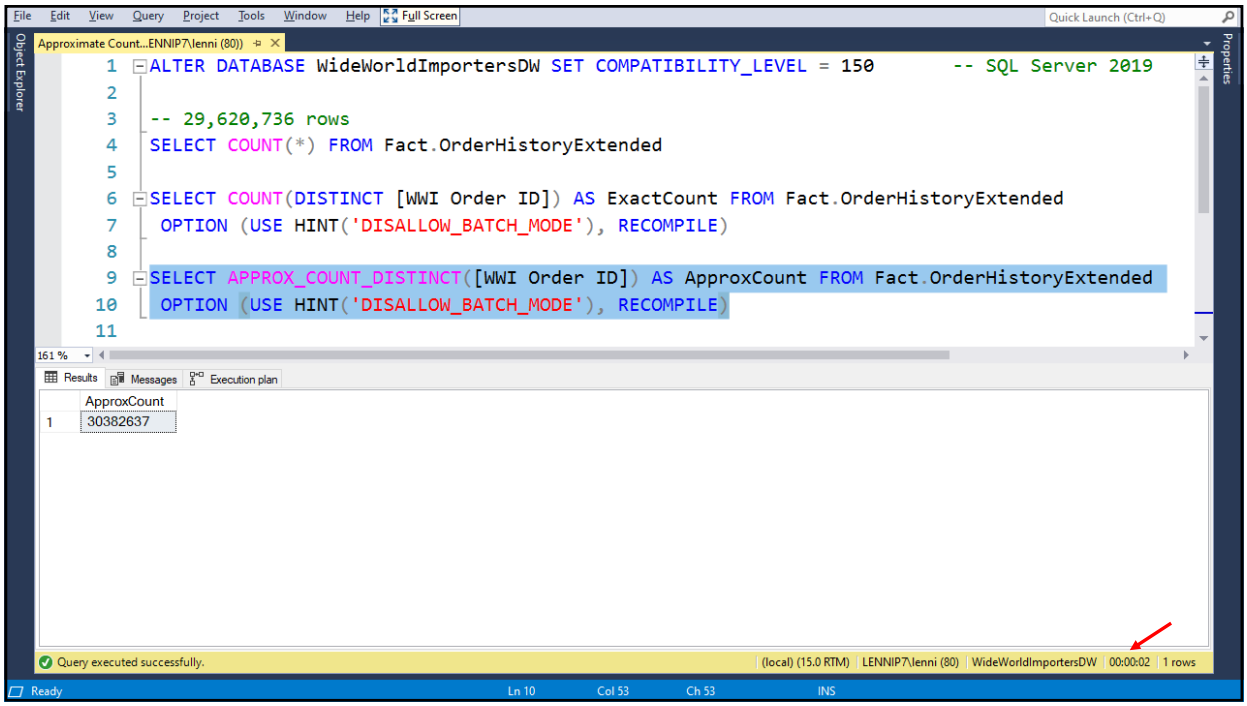

File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Results
Messages
Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
SELECT Cost: 0 %
Compute Scalar Cost: 0 %
Stream Aggregate (Aggregate) Cost: 0 % 4.033s 1 of 1 (100%)
Parallelism (Gather Streams) Cost: 0 % 4.033s 8 of 4 (200%)
Stream Aggregate (Aggregate) Cost: 1 % 4.032s 8 of 4 (200%)
Hash Match (Aggregate) Cost: 17 % 3.891s 29620736 of 28099200 (105%)
Parallelism (Repartition Streams) Cost: 2 % 0.737s 29620736 of 29620700 (100%)
Table [OrderHisto Cost: 2.2 29620 29620700
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:04 1 rows
Ready

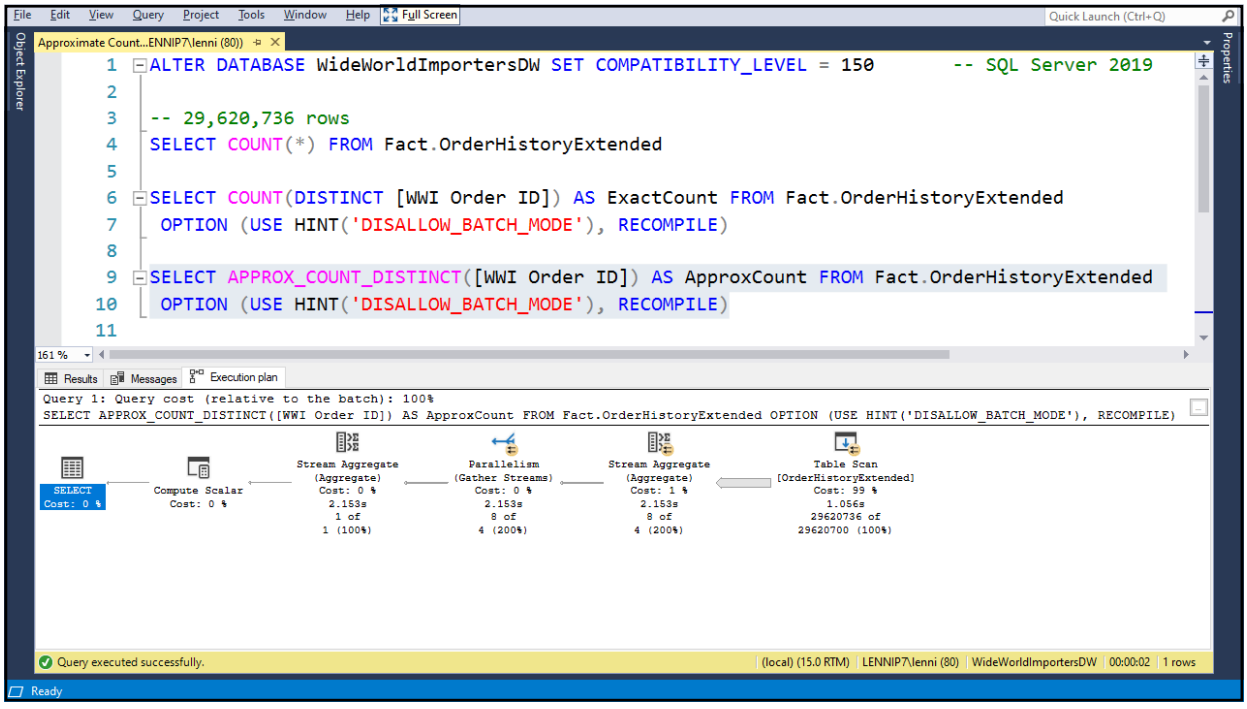

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
-- 29,620,736 rows
SELECT COUNT(*) FROM Fact.OrderHistoryExtended
SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
Query 1: Query cost (relative to the batch): 100%
SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
SELECT
Cached plan size 32 KB
Estimated Operator Cost 0 (0%)
Degree of Parallelism 8
Estimated Subtree Cost 713.673
Memory Grant 1722880
Estimated Number of Rows 1
Statement
SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended OPTION (USE HINT ('DISALLOW_BATCH_MODE'), RECOMPILE)
Parallelism (Gather Streams) Cost: 0 % 4.033s 8 of 4 (200%)
Stream Aggregate (Aggregate) Cost: 1 % 4.032s 8 of 4 (200%)
Hash Match (Aggregate) Cost: 17 % 3.891s 29620736 of 28099200 (105%)
Parallelism (Repartition Streams) Cost: 2 % 0.737s 29620736 of 29620700 (100%)
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:04 1 rows
Ready

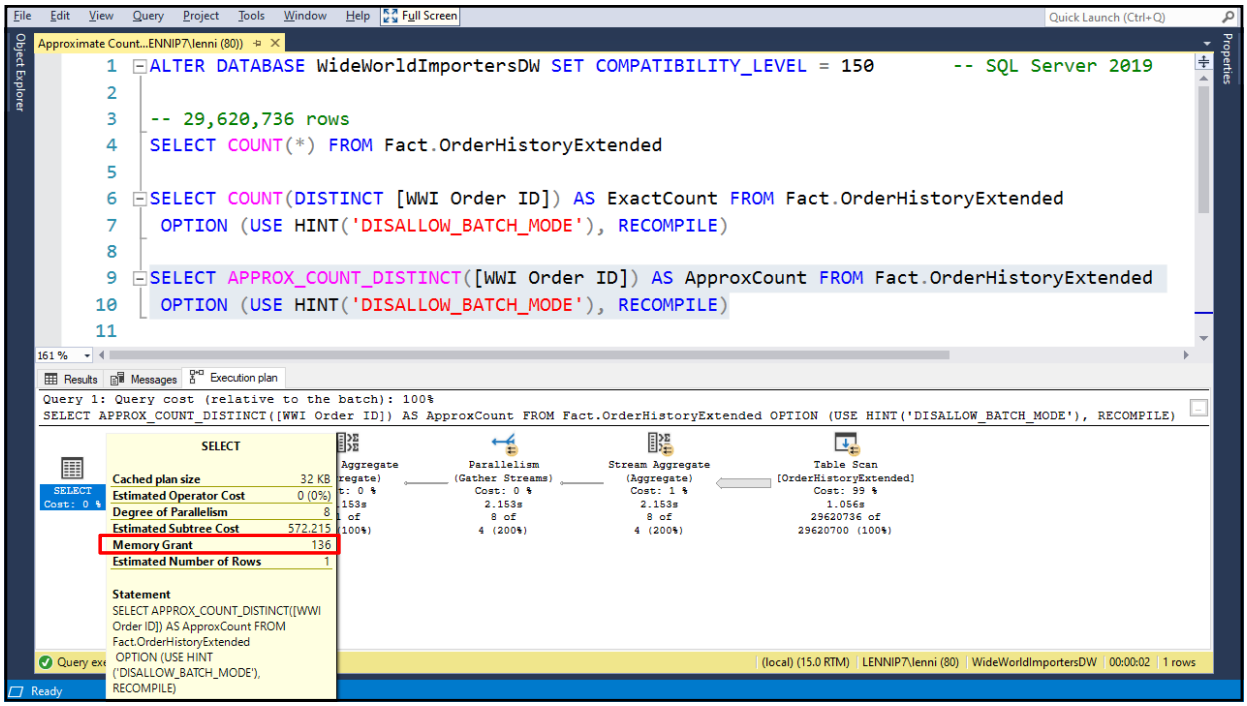

ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
-- 29,620,736 rows
SELECT COUNT(*) FROM Fact.OrderHistoryExtended
SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
Query 1: Query cost (relative to the batch): 100%
SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
SELECT Cost: 0 %
Compute Scalar Cost: 0 %
Stream Aggregate (Aggregate) Cost: 0 % 4.033s 1 of 1 (100%)
Parallelism (Gather Streams) Cost: 0 % 4.033s 8 of 4 (200%)
Stream Aggregate (Aggregate) Cost: 1 % 4.032s 8 of 4 (200%)
Hash Match (Aggregate) Cost: 17 % 3.891s 29620736 of 28099200 (105%)
Parallelism (Repartition Streams) Cost: 2 % 0.737s 29620736 of 29620700 (100%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:04 1 rows
Ready


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:04 1 rows
Ready Ln 7 Col 53 Ch 53 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:04 1 rows
Ready Ln 10 Col 53 Ch 53 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Results
Messages
Execution plan
ApproxCount
1 30382637
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:02 1 rows
Ready
Ln 10 Col 53 Ch 53 INS


File Edit View Query Project Tools Window Help Full Screen
Quick Launch (Ctrl+Q)
Approximate Count...ENNIP7\lenni (80))
Object Explorer
Properties
1 ALTER DATABASE WideWorldImportersDW SET COMPATIBILITY_LEVEL = 150 -- SQL Server 2019
2
3 -- 29,620,736 rows
4 SELECT COUNT(*) FROM Fact.OrderHistoryExtended
5
6 SELECT COUNT(DISTINCT [WWI Order ID]) AS ExactCount FROM Fact.OrderHistoryExtended
7 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
8
9 SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended
10 OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
11
161 %
Results
Messages
Execution plan
Query 1: Query cost (relative to the batch): 100%
SELECT APPROX_COUNT_DISTINCT([WWI Order ID]) AS ApproxCount FROM Fact.OrderHistoryExtended OPTION (USE HINT('DISALLOW_BATCH_MODE'), RECOMPILE)
SELECT Cost: 0 %
Compute Scalar Cost: 0 %
Stream Aggregate (Aggregate) Cost: 0 % 2.153s 1 of 1 (100%)
Parallelism (Gather Streams) Cost: 0 % 2.153s 8 of 4 (200%)
Stream Aggregate (Aggregate) Cost: 1 % 2.153s 8 of 4 (200%)
Table Scan [OrderHistoryExtended] Cost: 99 % 1.056s 29620736 of 29620700 (100%)
Query executed successfully.
(local) (15.0 RTM) LENNIP7\lenni (80) WideWorldImportersDW 00:00:02 1 rows
Ready



Part 3

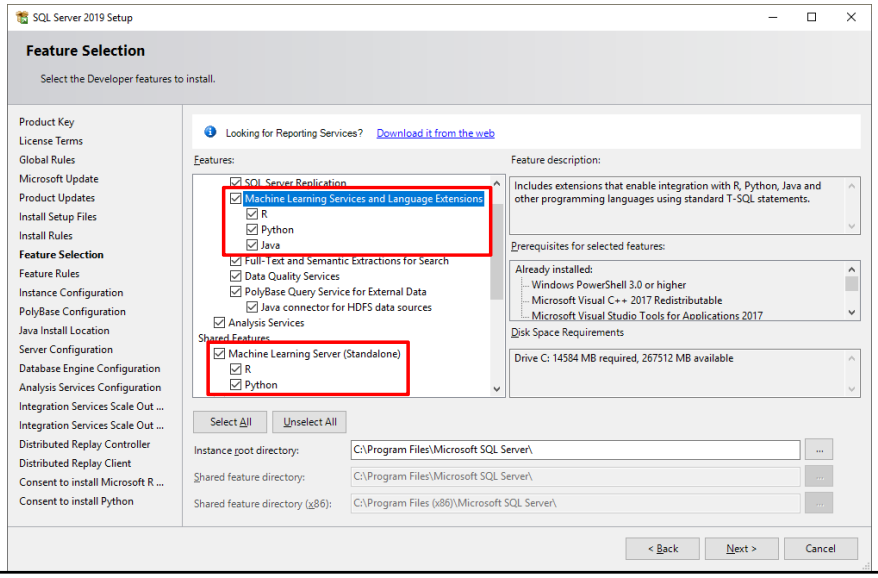
Modern Development Platform

Machine Learning (ML) Services – R and Python

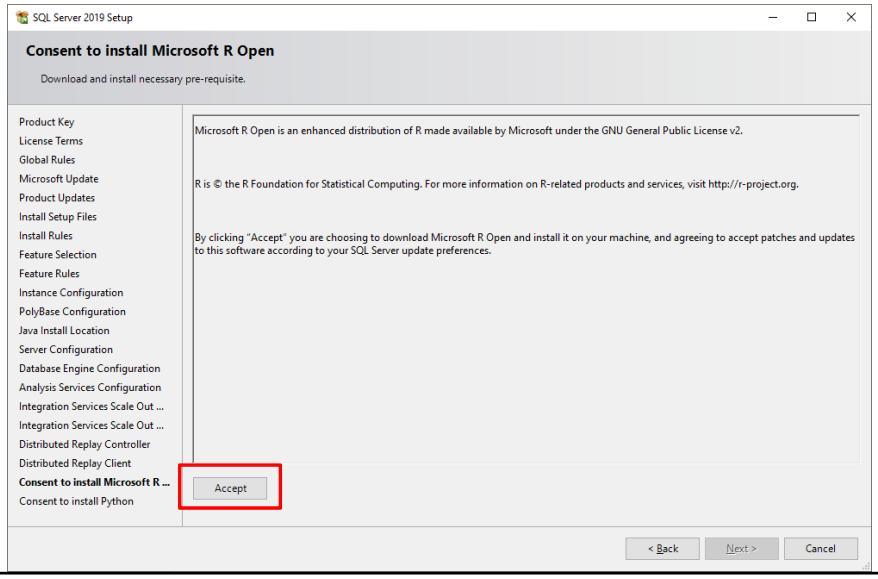
Machine Learning Services + Language Extensions

- Machine Learning Services
  - Incorporate R and/or Python
    - Advanced analytics against big data inside SQL Server
  - Deploy existing code to run inside SQL Server
    - No need to export the data for local processing
- Language Extensions
  - Extend T-SQL with Java
    - E.g., Full Regex to replace LIKE

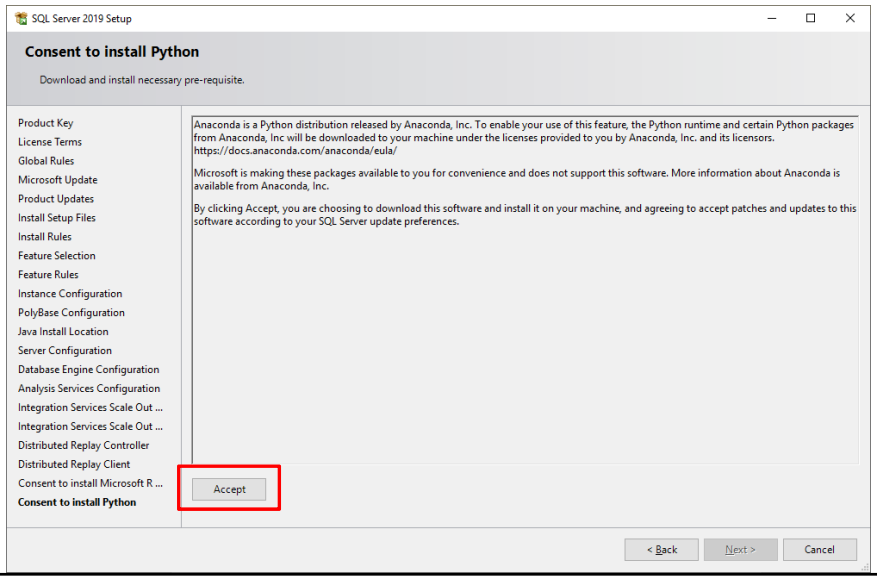
Machine Learning Services + Language Extensions: Setup



Machine Learning Services + Language Extensions: Setup



Machine Learning Services + Language Extensions: Setup



R / Python Integration in SQL Server

- Call **sp_execute_external_script**
  - Choose the language (R or Python)
  - Supply the source code
  - Supply the data source (T-SQL query)
- Example:
  - **EXEC sp_execute_external_script**
    @language = N'R',
    @script = N'*embedded-code*'
    @input_data_1 = 'SELECT …'
- Set the result schema:
  - **WITH RESULT SETS((col1 int, col2 money))**

Importing R packages

- Download **sqlmlutils.zip**
  - https://github.com/Microsoft/sqlmlutils/tree/master/R/dist

```
R -e "install.packages('RODBCext', repos='https://cran.microsoft.com')"
R CMD INSTALL c:\temp\sqlmlutils_0.7.1.zip
```

- Add desired packages to your database

```
library(sqlmlutils)
connection <- connectionInfo(server="server", database="db")
sql_install.packages
 (connectionString=connection, pkgs="package", verbose=TRUE, scope="PUBLIC")
```

Importing Python packages

- Download **sqlmlutils.zip**
  - https://github.com/Microsoft/sqlmlutils/tree/master/Python/dist

```
pip install "pymssql<3.0"
pip install --upgrade --upgrade-strategy only-if-needed
 c:\temp\sqlmlutils_0.7.2.zip
```

- Add desired packages to your database

```
import sqlmlutils
connection = sqlmlutils.ConnectionInfo(server="server", database="db")
sqlmlutils.SQLPackageManager(connection).install("package")
```

Machine Learning Services
R & Python

demo

Built-in JSON Support

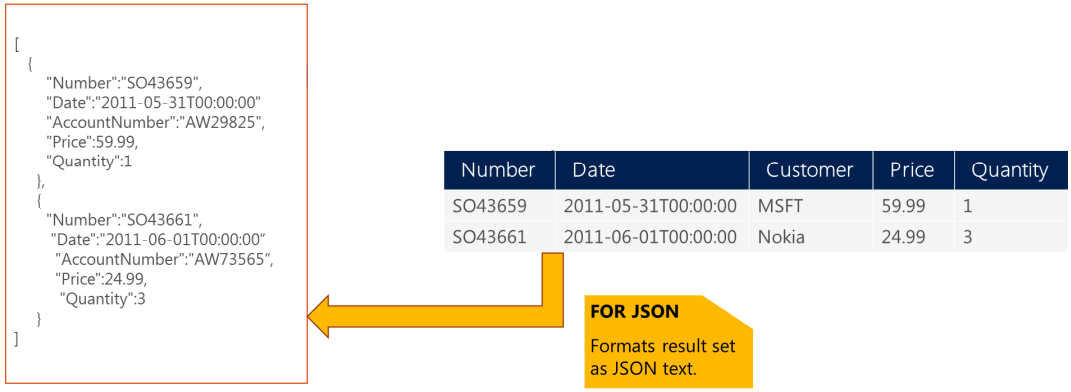
Built-In JSON Support

- Capabilities
  - Format and export JSON from relational queries
  - Store and query JSON inside the database
- Conceptually similar to XML support
  - Simpler model
  - No native “json” data type; uses nvarchar(max)
- Why no native type?
  - Easier migration to leave json columns as ordinary string types
  - Cross-feature compatibility (e.g., Hekaton, temporal)
- No custom JSON indexes
  - Optimize JSON queries using standard indexes
  - Create computed columns over desired properties, and then index the computed columns

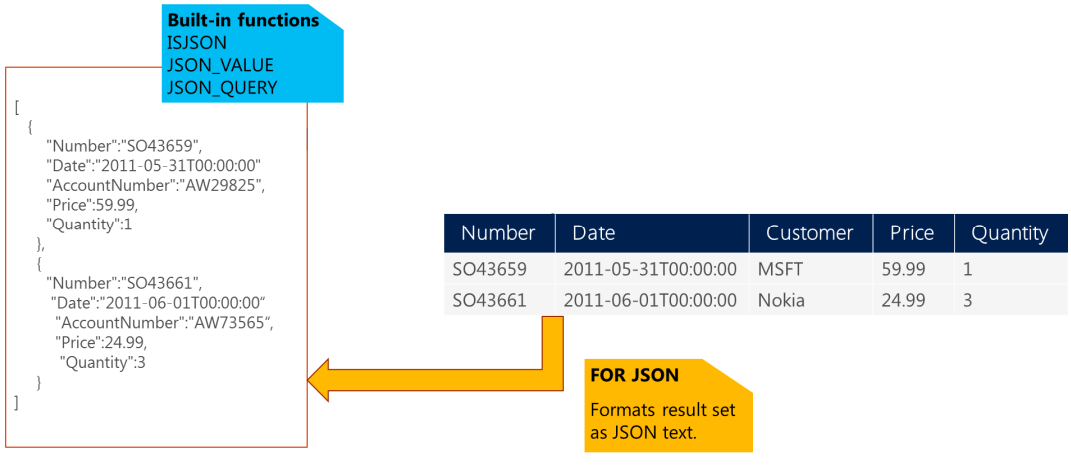
Bidirectional JSON Transformation

Number	Date	Customer	Price	Quantity
SO43659	2011-05-31T00:00:00	MSFT	59.99	1
SO43661	2011-06-01T00:00:00	Nokia	24.99	3

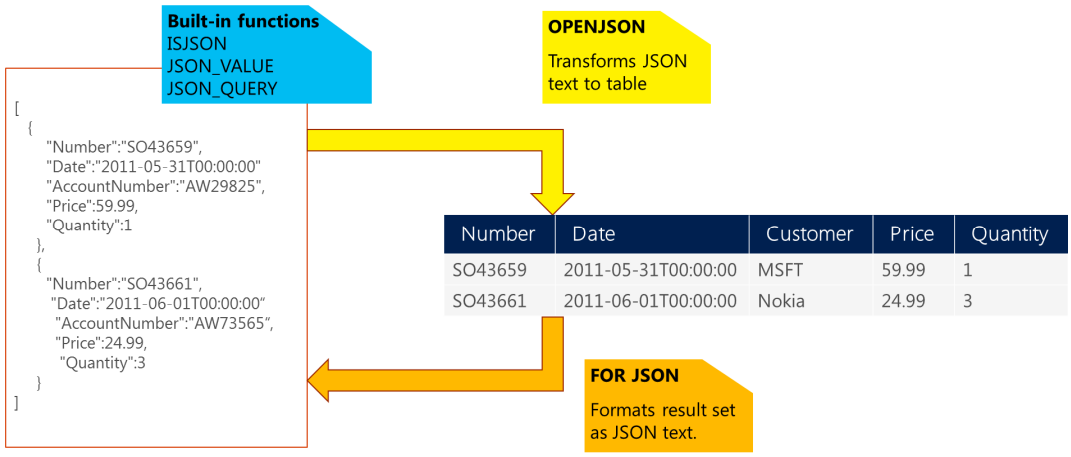
Bidirectional JSON Transformation



Bidirectional JSON Transformation



Bidirectional JSON Transformation



FOR JSON Clause

- Append to SELECT statements to generate results in JSON format
  - Example:

```
SELECT * FROM Customer FOR JSON AUTO
```

- FOR JSON AUTO
  - Creates nested structure based on table hierarchy
- FOR JSON PATH
  - Creates nested structure based on column aliases

FOR JSON Formatting Options

- WITHOUT_ARRAY_WRAPPER
  - Don't generate [] syntax (single JSON object)
- ROOT
  - Generate single root wrapper object around the results
- INCLUDE_NULL_VALUES
  - Generate properties for NULL columns

Generating JSON

demo

Built-in JSON Functions

- ISJSON
  - Validates for well-formed JSON
  - Use in check constraints for NVARCHAR columns containing JSON
- JSON_QUERY
  - Queries by path expression and returns a nested object/array
  - Similar to *xml.query*
- JSON_VALUE
  - Queries by path expression and returns a scalar value
  - Similar to *xml.value*
- JSON_MODIFY
  - Directly modify JSON content
  - Similar to *xml.modify*

JSON Path Expressions

- Reference JSON properties using a JavaScript-like syntax

Syntax	Description
`$`	References the entire JSON object
`$.property1`	References a top-level property in the JSON object
`$[5]`	References the sixth element in the JSON array
`$.property1.property2 .array1[5].property3 .array2[15].property4`	References a complex nested property in the JSON object

JSON Query Example

```
SELECT
 Id,
 OrderNumber,
 OrderDate,
 JSON_VALUE(OrderDetails, '$.Order.ShipDate')
FROM
 SalesOrderRecord
WHERE
 ISJSON(OrderDetails) AND
 JSON_VALUE(OrderDetails, '$.Order.Type') = 'C'
```

Storing and Querying JSON

Shredding JSON

- OPENJSON table-valued function (TVF)
  - Provides a rowset view over a JSON document
  - Shreds single JSON document into multiple rows
- What does it do?
  - Iterates through objects (if JSON array) or properties (if JSON object)
  - Generates a row for each object/property with key, value, and type

Shredding JSON

demo

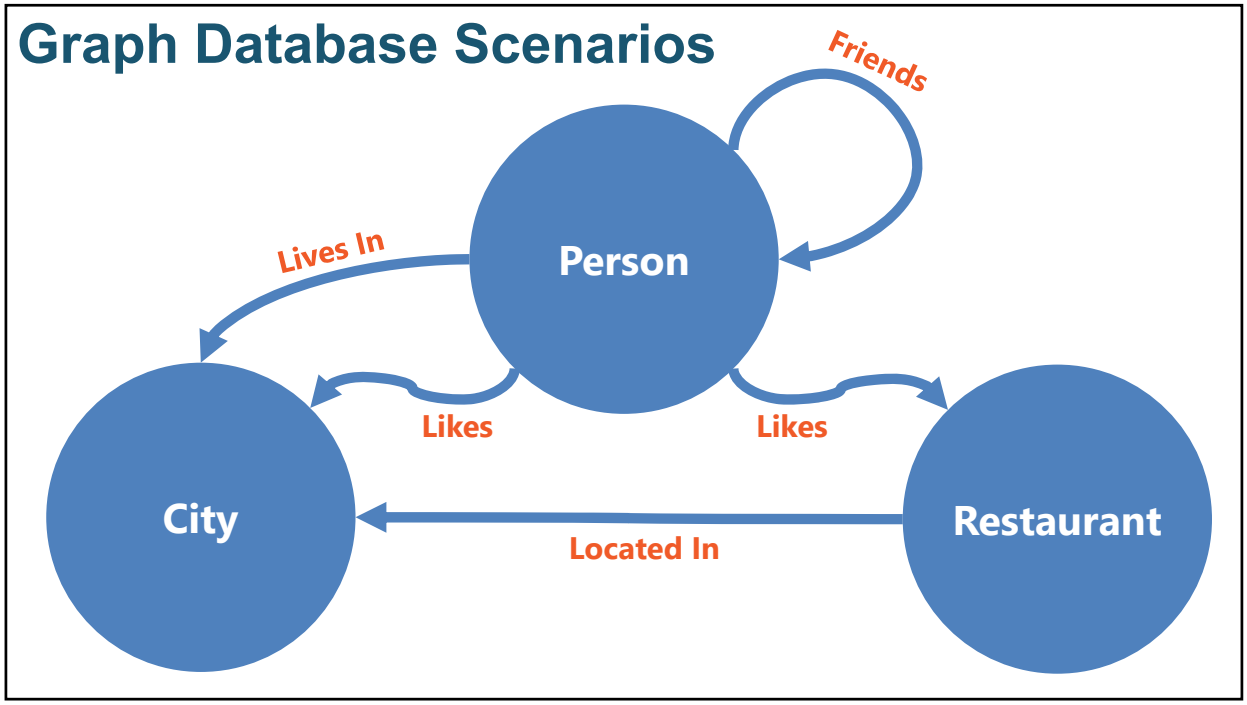
Graph DB

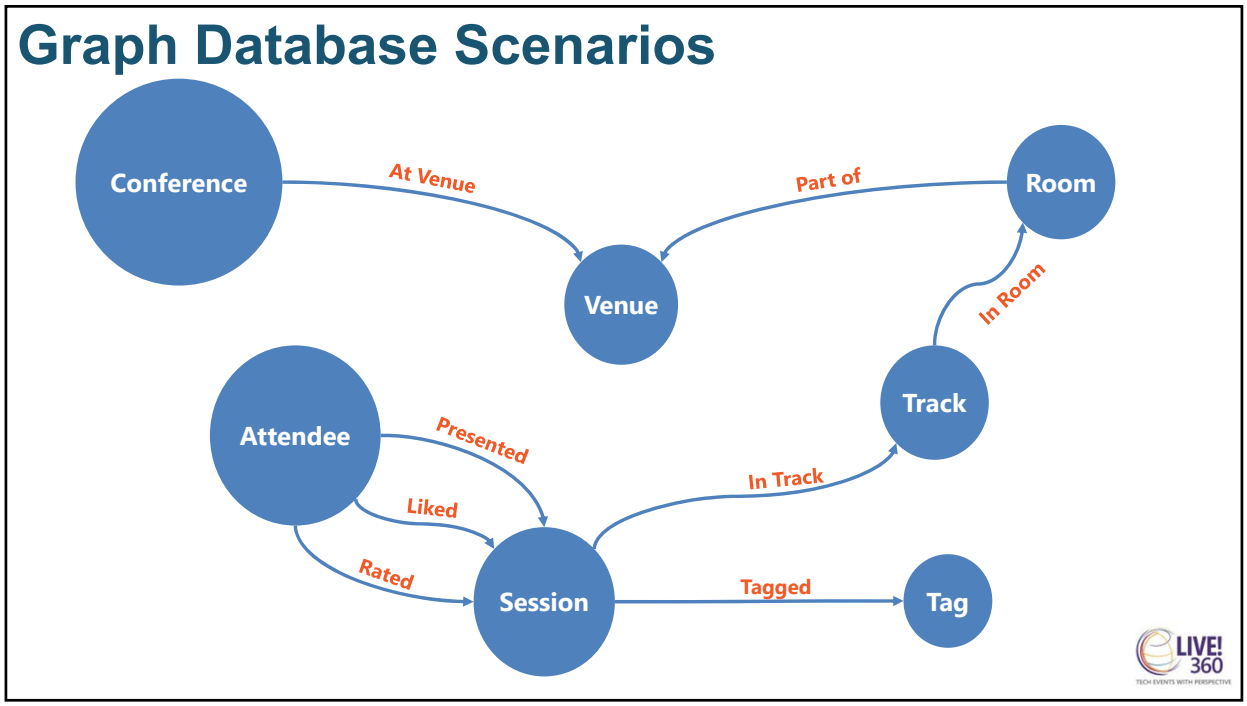
What is a Graph Database?

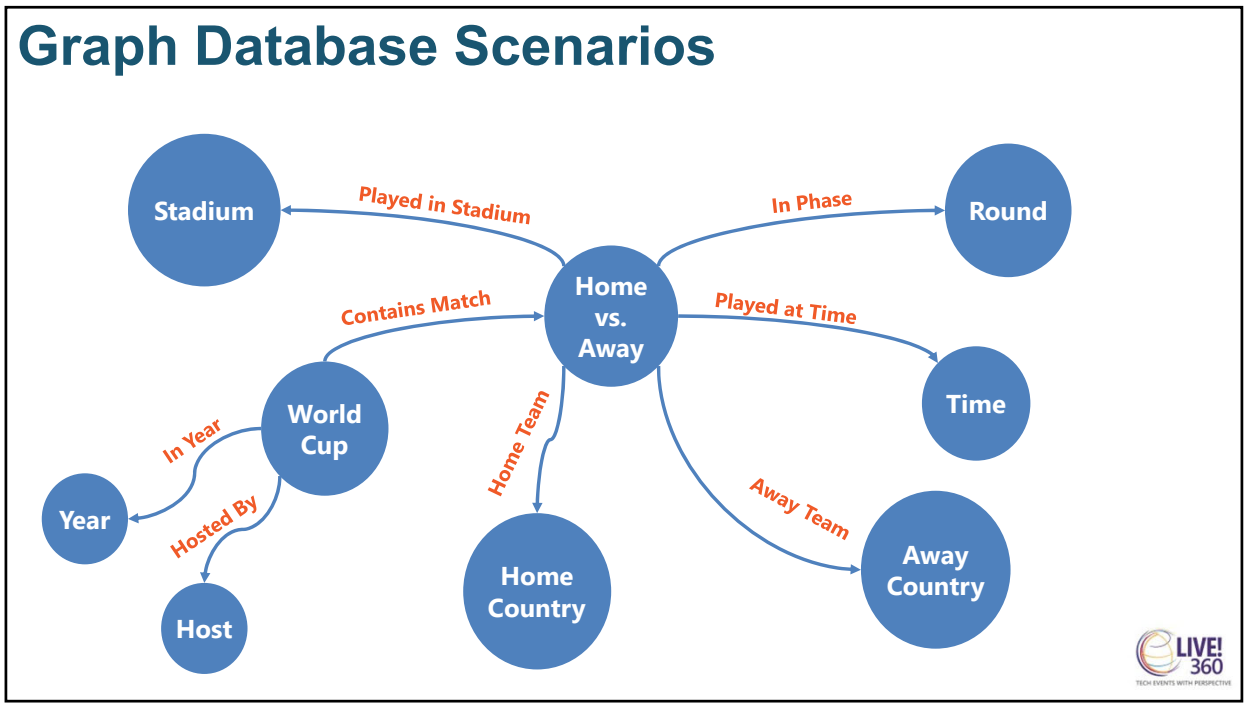
- Focus on relationships
  - Particularly complex many-to-many relationships
- A collection of nodes and edges
  - Node = entity
  - Edge = relationship
- Every edge connects two nodes
  - Flexible relationship model
- Great for analyzing interconnected data
  - Discover non-obvious relationships

Graph DB

- Very popular in the last few years
  - Annual growth of over 300%
- Typical scenarios
  - Supply chain management
  - Dating web sites
  - Recommendation engines (Amazon, eBay)
  - Social networks (Facebook, LinkedIn)




Graph Database Scenarios
Conference
At Venue
Venue
Part of
Room
In Room
Track
Attendee
Presented
Liked
Rated
Session
In Track
Tagged
Tag
LIVE! 360


Graph Database Scenarios
Stadium
Played in Stadium
In Phase
Round
Contains Match
Home vs. Away
Played at Time
Time
World Cup
In Year
Year
Hosted By
Host
Home Team
Home Country
Away Team
Away Country
LIVE! 360

Well-Defined Relationships

- Graph model
  - Relationships (edges) are first-class citizens
- Relational model
  - Relationships inferred by foreign keys embedded into entities

Nodes and Edges

- Node
  - An entity (noun)
    - Landmass, person, business, city, starship
  - Holds properties
    - Like ordinary columns in a table
- Edge
  - A relationship (verb/adjective)
    - WorksAt, LivesIn, Likes
  - Establishes a one-way connection between two nodes
    - Person LivesIn City
  - Can also hold properties
    - Like ordinary columns in a junction table

Graph in SQL Server

- One graph per database
  - Graph contains only node and edge tables
  - Node and edge tables can be created in any schema
    - But they still all belong to one logical graph
- Graph table support
  - Node and edge tables
    - Just add AS NODE or AS EDGE to CREATE TABLE statement
  - Supports most operations as normal tables

Node and Edge Tables

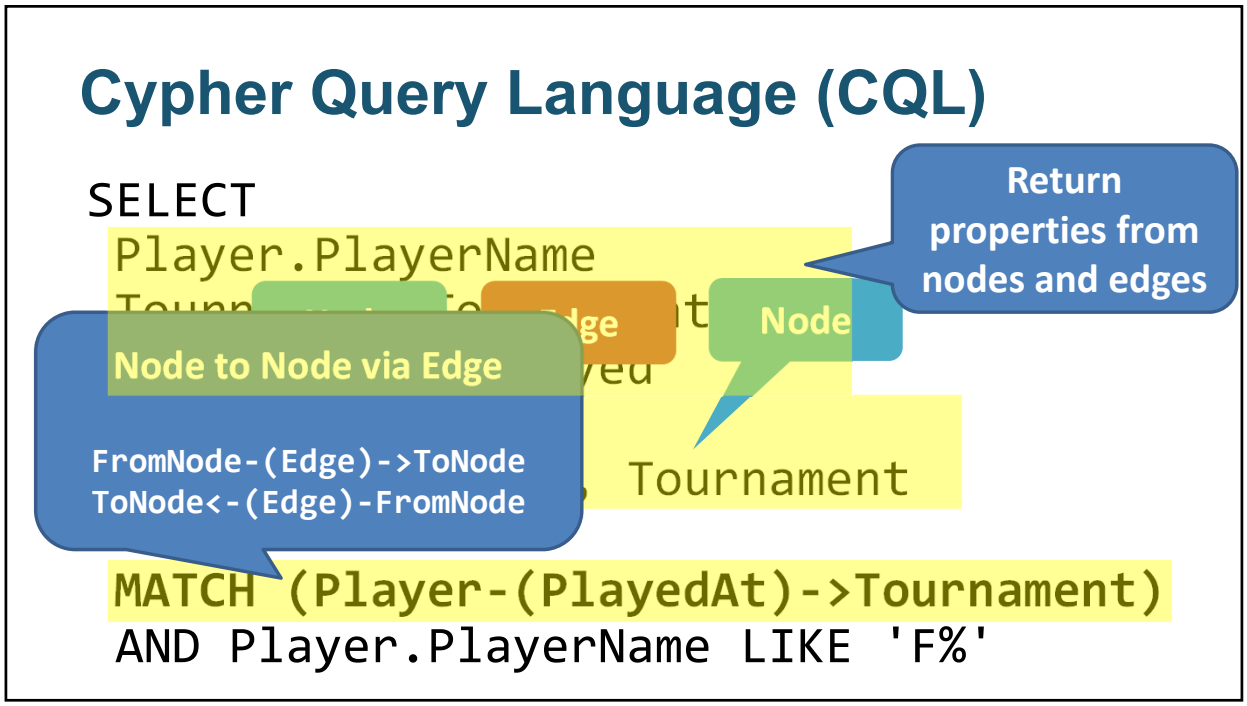
- Node tables
  - CREATE TABLE…AS NODE
    - Adds **$node_id_*xxxx*** column
- Edge tables
  - CREATE TABLE…AS EDGE
    - Adds **$edge_id_*xxxx*** column
    - Adds **$from_id** and **$to_id** columns
      - References any two node IDs

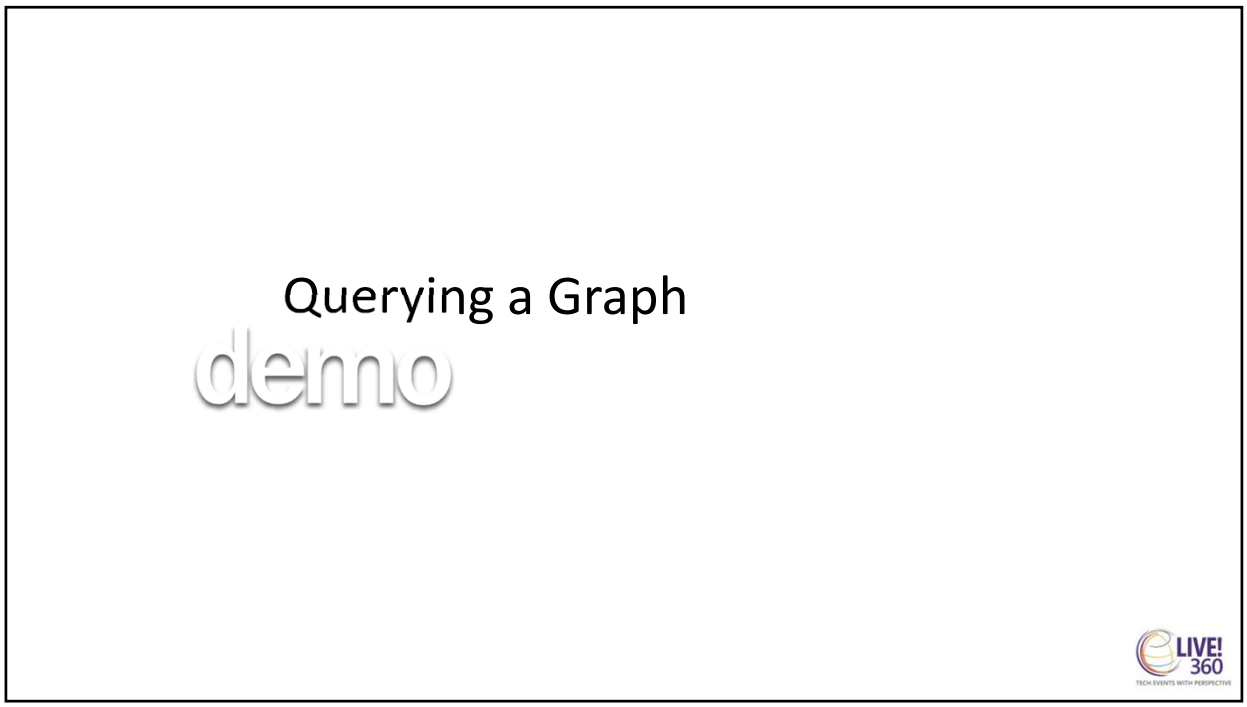
Populating a graph

demo

Cypher Query Language (CQL)

- Originally developed by Neo4j
  - Industry leader in graph database technology
  - Open source since 2015
- MATCH clause
  - Multi-hop navigation
    - From one node to another node through an edge
  - Join-free pattern matching
    - Identifies the nodes that need to be matched


Cypher Query Language (CQL)
SELECT
Player.PlayerName
Node
Return properties from nodes and edges
Node to Node via Edge
FromNode-(Edge)->ToNode
ToNode<-(Edge)-FromNode
Tournament
MATCH (Player-(PlayedAt)->Tournament)
AND Player.PlayerName LIKE 'F%'


Querying a Graph
demo
LIVE! 360

Graph DB Improvements in SQL Server 2019

- SHORTEST_PATH() function
  - Recursive traversals
- Edge constraints
  - Restrict the types of nodes that can be related

Stretch DB

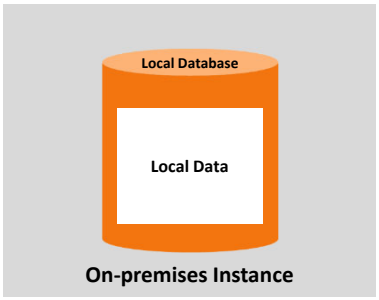
Introducing Stretch Database

- Store portions of a database in the cloud
- Remote Data Archive (RDA)
  - Keep “hot” data in local SQL Server database (on-premises)
  - Seamlessly migrate “cold” data to Azure SQL Database
- Stretch Database Advisor
  - Downloadable as part of the SQL Server 2016 Upgrade Advisor
  - Helps identify tables that are good candidates for stretch
- Enable stretch database
  - Use T-SQL or the Enable Database for Stretch Wizard in SSMS

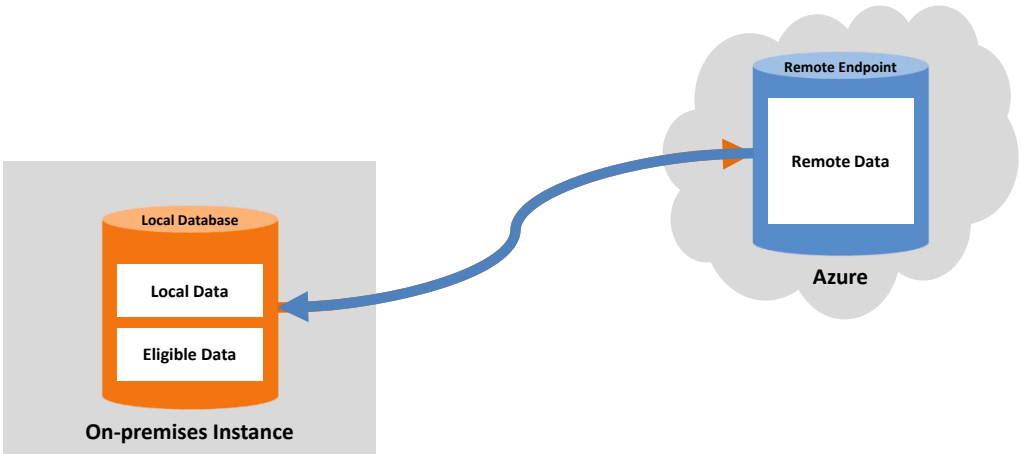
Stretch Database Terminology

- Local database
  - The on-premises SQL Server database being “stretched”
  - Your responsibility to maintain and backup
- Eligible data
  - Data in the local database that has not yet been moved to the cloud
- Remote endpoint
  - The Azure SQL Data Warehouse database in the cloud
- Remote data
  - Data that has already been moved from the local database to the cloud
  - Automatic backups by the stretch database service on Azure

Stretch Database Workflow



Stretch Database Workflow



Stretch Database Workflow

- sys.sp_rda_deauthorize_db
- sys.sp_rda_reauthorize_db

Stretching a table

demo

Stretching Selected Rows

- The entire table doesn't need to be stretched
  - Write a predicate function to filter the rows to be migrated
  - Easily filter by date, status, or other constant
- Filter predicate restrictions
  - Expression must be deterministic
  - Can't inherently implement a sliding window function
- Update filter predicates in place
  - Signature must not change
  - New function must be less restrictive than current one

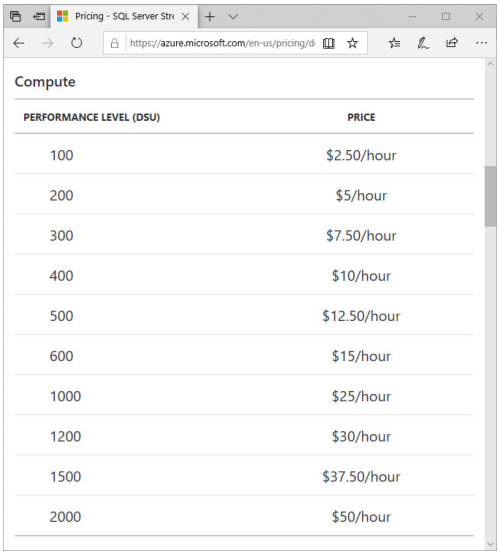
Stretch Filter Predicate Functions

```
CREATE FUNCTION sec.fn_MyStretchPredicate(@Parm1 AS int, ...)
 RETURNS TABLE
 WITH SCHEMABINDING
AS
   -- SQL Server passes in column values of each row via parameters

   RETURN
     SELECT 1 AS IsEligible
     WHERE ...
       -- Custom logic here examines the parameters (column values)
       -- passed in, and determines if the row should be migrated
```

Stretching selected rows

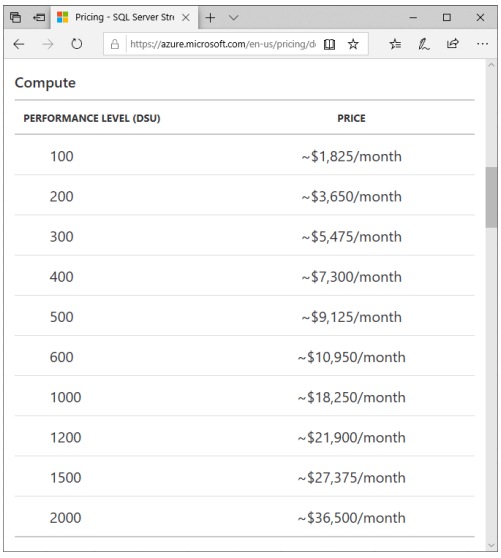
demo

Stretch Pricing



Compute

PERFORMANCE LEVEL (DSU)	PRICE
100	$2.50/hour
200	$5/hour
300	$7.50/hour
400	$10/hour
500	$12.50/hour
600	$15/hour
1000	$25/hour
1200	$30/hour
1500	$37.50/hour
2000	$50/hour



Compute

PERFORMANCE LEVEL (DSU)	PRICE
100	~$1,825/month
200	~$3,650/month
300	~$5,475/month
400	~$7,300/month
500	~$9,125/month
600	~$10,950/month
1000	~$18,250/month
1200	~$21,900/month
1500	~$27,375/month
2000	~$36,500/month

Stretch Database Limitations and Considerations

- Unsupported table types
  - Memory-optimized tables (Hekaton)
  - Replicated tables
  - FILESTREAM/FileTable
  - Tables enabled for Change Tracking or Change Data Capture (CDC)
  - Tables with more than 1023 column or 998 indexes
- Unsupported data types and column properties
  - timestamp, sql_variant, xml, geography, geometry, hierarchyid, CLR UDTs, COLUMN_SET, computed columns
- Unsupported indexes
  - XML, full-text, spatial, indexed views into the table

Stretch Database Limitations and Considerations (cont.)

- Unsupported constraints
  - Check constraints
  - Default constraints
  - Foreign key constraints out of the table (parent tables)
- Uniqueness not enforced UNIQUE constraints
- UPDATE and DELETE not supported
- ALTER TABLE not supported
- Views over stretched tables
  - Can’t create index on the view
  - Can’t update or delete from the view (but you can insert into the view)

Temporal Tables

Temporal Data

- System version tables
- Point-in-time data access
  - Query updated and delete data, not just current data
  - Seamless and transparent
- Primary use cases
  - Time travel
  - Auditing
  - Accidental data loss recovery

Using Temporal

- Create an ordinary table
  - Must have primary key column
  - Must have two period (start and end date) **datetime2** columns
- Enable the table for temporal
  - Creates history table with same schema, but without constraints
  - Automatically records updates and deletes to the history table
- Query to point in time
  - Include **FOR SYSTEM_TIME AS OF** in your SELECT statement
- Manage schema changes
  - ALTER TABLE automatically updates the history table
    - Some schema changes (e.g., new IDENTITY or computed columns) require turning temporal off, applying the changes to both tables, and then turning it back on

Creating a Temporal Table

```
CREATE TABLE Department
(
 DepartmentID    int NOT NULL IDENTITY(1,1) PRIMARY KEY,
 DepartmentName  varchar(50) NOT NULL,
 ManagerID       int NULL,
 ValidFrom       datetime2 GENERATED ALWAYS AS ROW START NOT NULL,
 ValidTo         datetime2 GENERATED ALWAYS AS ROW END   NOT NULL,
 PERIOD FOR SYSTEM_TIME (ValidFrom, ValidTo)
)
WITH (SYSTEM_VERSIONING = ON (HISTORY_TABLE = dbo.DepartmentHist))
```

Querying a Temporal Table

```
DECLARE @ThirtyDaysAgo datetime2 =
 DATEADD(d, -30, SYSDATETIME())

SELECT *
 FROM Employee
 FOR SYSTEM_TIME AS OF @ThirtyDaysAgo
 ORDER BY EmployeeId
```

FOR SYSTEM_TIME Variations

- FOR SYSTEM_TIME **AS OF a**
  - Returns precisely one version of a row from point in time
- FOR SYSTEM_TIME **FROM a TO b**
  - Returns versions overlapping specified range
- FOR SYSTEM_TIME **BETWEEN a AND b**
  - Same as FROM…TO, but includes upper boundary
- FOR SYSTEM_TIME **CONTAINED IN (a, b)**
  - Returns versions fully contained within specified range

Temporal Tables

demo

Managing Temporal History

- History table gets one row per UPDATE and DELETE
  - Can grow very large, very quickly
    - If you update 10 rows 10 times a day, you add 1,000 rows in the history table
- Options for managing growth
  - Stretch DB (requires Azure account)
  - Table Partitioning (requires Enterprise prior to SQL Server 2016 SP1)
  - Custom script (requires maintenance window)
  - Update unneeded columns with NULL (requires maintenance window)
- As of SQL Server 2017
  - Temporal history retention policy

Managing Temporal History
(custom script)

demo

Managing Temporal History
(Stretch DB)

demo

Temporal Limitations and Considerations

- Triggers
  - INSTEAD OF triggers are unsupported
  - AFTER triggers are supported on the current table only
- In-memory OLTP (Hekaton) is not supported
- FILESTREAM/FileTable is not supported
- INSERT and UPDATE statements cannot reference the period columns
- Works with other new SQL Server features
  - DDM, RLS, Always Encrypted, Stretch

Part 4

Security Features

Security Features

- SQL Server 2016
  - Dynamic Data Masking (DDM)
  - Row Level Security (RLS)
  - Always Encrypted
- SQL Server 2019
  - Always Encrypted with Secure Enclaves
  - Data Classification and Auditing
  - TDE suspend and resume

Dynamic Data Masking

Introducing Dynamic Data Masking (DDM)

- Limit exposure to sensitive data by masking
- Four masking functions
  - **default** – full masking; entire column is masked
  - **partial** – show starting and/or ending characters of the column data, mask the rest with a custom string
  - **email** – show the first character of the column data, mask the rest with XXX@XXXX.com
  - **random** – entire column is replaced by random values
- Reveals masked data to queries
  - Data in the database is not changed
- Enforced at the database level
  - No impact at the application level

Masking Table Columns

```
CREATE TABLE Customer(
  FirstName varchar(20)
   MASKED WITH (FUNCTION='default()'),
  LastName varchar(20),
  Phone varchar(12)
   MASKED WITH (FUNCTION='partial(0, "_-___-", 4)'),
  Email varchar(200)
   MASKED WITH (FUNCTION='email()'),
  Balance money
   MASKED WITH (FUNCTION='random(1000, 5000)'))

ALTER TABLE Customer
 ALTER COLUMN LastName
 ADD MASKED WITH (FUNCTION='default()')
```

Masking Different Data Types

Masking Function	Behavior	Strings	Numbers	Dates	Other Types
default()	Show **xxxx** mask (strings), or minimum value (other types)	Yes	Yes	Yes	Yes
partial(*a*, 'x', *b*)	Show first *a* characters, custom mask, and last *b* characters	Yes	No	No	No
email()	Show first character and **XXX@XXXX.com**	Yes	No	No	No
random(*a*, *b*)	Show random value between *a* and *b*	No	Yes	No	No

Discovering Masked Columns

- sys.columns
  - is_masked
  - masking_function
- sys.masked_columns
  - Inherits from sys.columns
  - Filters to show only masked columns
    - WHERE is_masked = 1

Mask Permissions

- DDM is based on user permissions
- Create a table with masked columns
  - No special permission required
- Add, replace, or remove a column mask
  - Requires **ALTER ANY MASK** permission
- View unmasked data in masked columns
  - Requires **UNMASK** permission
- Updating data in a masked column
  - No special permission

Dynamic Data Masking (DDM)

demo

DDM Limitations and Considerations

- DDM cannot be used with
  - FILESTREAM columns
  - COLUMN_SET, or a sparse column that's part of a COLUMN_SET
  - Computed columns
    - But will return masked data if it depends on a masked column
  - Key for FULLTEXT index
  - Encrypted columns (Always Encrypted)
- Masking is a one-way street
  - Once masked, the actual data can never be obtained
  - An ETL process from a source with masked columns results in an irreversible data loss when loaded into the target environment

Row Level Security (RLS)

Introducing Row Level Security (RLS)

- Restrict access to individual rows in a table
  - Create predicate functions (inline TVF)
  - Write custom logic to control user access to every row
- Security policy
  - Bind the functions to tables as a filter or block predicate
  - SQL Server filters and blocks user access to individual rows
  - Can enable/disable the policy as desired

Filter and Block Predicates

- Filter predicate
  - `SELECT, UPDATE, DELETE`
    - Can't select, update, or delete rows that violate the predicate
- Block predicate
  - `AFTER INSERT, AFTER UPDATE`
    - Can't insert or update rows to values that would violate the predicate
  - `BEFORE UPDATE, BEFORE DELETE`
    - Can't update or delete rows that violate the predicate
    - Implied when combined with filter predicate

RLS Security Policy

Predicate	SELECT/UPDATE/DELETE rows that violate the predicate	INSERT rows with violating values	UPDATE rows to violating values
Filter	No	Yes	Yes
AFTER INSERT block	Yes	No	Yes
AFTER UPDATE block	Yes	Yes	No
BEFORE UPDATE block	No	N/A	N/A
BEFORE DELETE block	No	N/A	N/A

Creating Security Predicate Functions

- Write a security predicate function
  - Ordinary inline table-valued function (TVF)
    - Must be schema-bound
  - Accept any parameters of any type
    - Map these parameters to column values
- Implement your own custom logic in T-SQL
  - Examine the row via the columns passed in as parameters
    - Determine if access should be allowed or denied
  - Return a scalar 1 (allow) or nothing at all (deny)
  - Encapsulate logic inside WHERE clause of a single SELECT statement inside the TVF

Creating Security Predicate Functions

```
CREATE FUNCTION sec.fn_MySecurityPredicate(@Parm1 AS int, ...)
 RETURNS TABLE
 WITH SCHEMABINDING
AS
  -- SQL Server passes in column values of each row via parameters

  RETURN
    SELECT 1 AS Result
    WHERE ...
      -- Custom logic here examines the parameters (column values)
      -- passed in, and determines the row's accessibility
```

RLS Security Policy

- Create a security policy
  - Add filter and block predicates to the policy
- Bind each predicate function to a table
  - Map table columns to the TVF parameters
    - SQL Server will call the TVF to determine the accessibility of each row

RLS Security Policy Examples

- With filter predicate

```
CREATE SECURITY POLICY sec.MySecurityPolicy
 ADD FILTER PREDICATE sec.fn_MySecurityPredicate(Col1, ...)
  ON dbo.MyTable
 WITH (STATE = ON)
```

- With AFTER INSERT and AFTER UPDATE block predicates

```
CREATE SECURITY POLICY sec.MySecurityPolicy
 ADD BLOCK PREDICATE sec.fn_MySecurityPredicate(Col1, ...)
  ON dbo.MyTable AFTER INSERT,
 ADD BLOCK PREDICATE sec.fn_MySecurityPredicate(Col1, ...)
  ON dbo.MyTable AFTER UPDATE,
 WITH (STATE = ON)
```

Getting started with
Row-Level Security (RLS)

Identifying Users for RLS

- Credentials supplied for the database connection
  - SQL Server login (username and password)
  - Windows authentication
  - Obtain the username from DATABASE_PRINCIPAL_ID
- Different strategy required for n-tier applications
  - Typically, all users connect to the database using the same service account from the application tier
  - DATABASE_PRINCIPAL_ID is the same for every user
- Solution: Use new SESSION_CONTEXT feature
  - Store the application level user ID as a readonly value in session context

Introducing SESSION_CONTEXT

- What is “session context”?
  - Stateful dictionary (key/value pair) object
    - Key is a Unicode string
    - Value is a sql_variant (any data type)
  - Retains state for the lifetime of the connection
- SESSION_CONTEXT()
  - Built-in function that returns a value from session context by key
- sp_set_session_context
  - System stored procedure that stores a value to session context
  - Specify **@key**, **@value**, and optionally, **@read_only**

Using Row-Level Security (RLS) with SESSION_CONTEXT() for N-Tier Applications

demo

Always Encrypted

Traditional SQL Server Encryption Features

- Column (cell-level) encryption
  - Uses certificates or symmetric keys
- Database (page-level) and backup encryption
  - Transparent Data Encryption (TDE)
  - Uses TDE certificate with database encryption keys (DEKs)
- Keys and certificates are stored in the database
  - Risk of security breach at the database level
- Data is only encrypted “at rest”
  - Risk of security breach while “in flight”

Introducing Always Encrypted

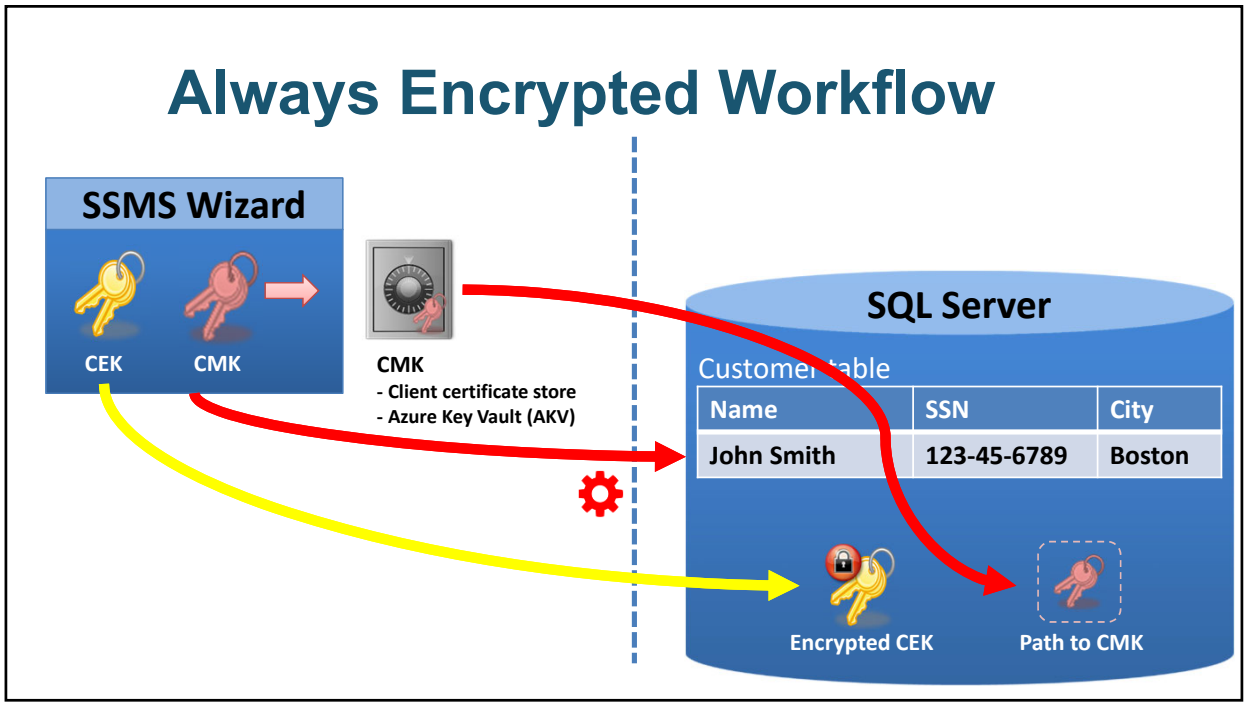
- Always Encrypted in SQL Server 2016
  - Based on keys managed outside the database
  - Keys are never revealed to SQL Server
- Separating those who *own* the data from those who *manage* it
  - Uses client side drivers to encrypt/decrypt on the fly
- SQL Server is incapable of decrypting on its own
  - Data is always encrypted in flight
- Enable Always Encrypted
  - Use T-SQL or the Always Encrypted Wizard in SSMS

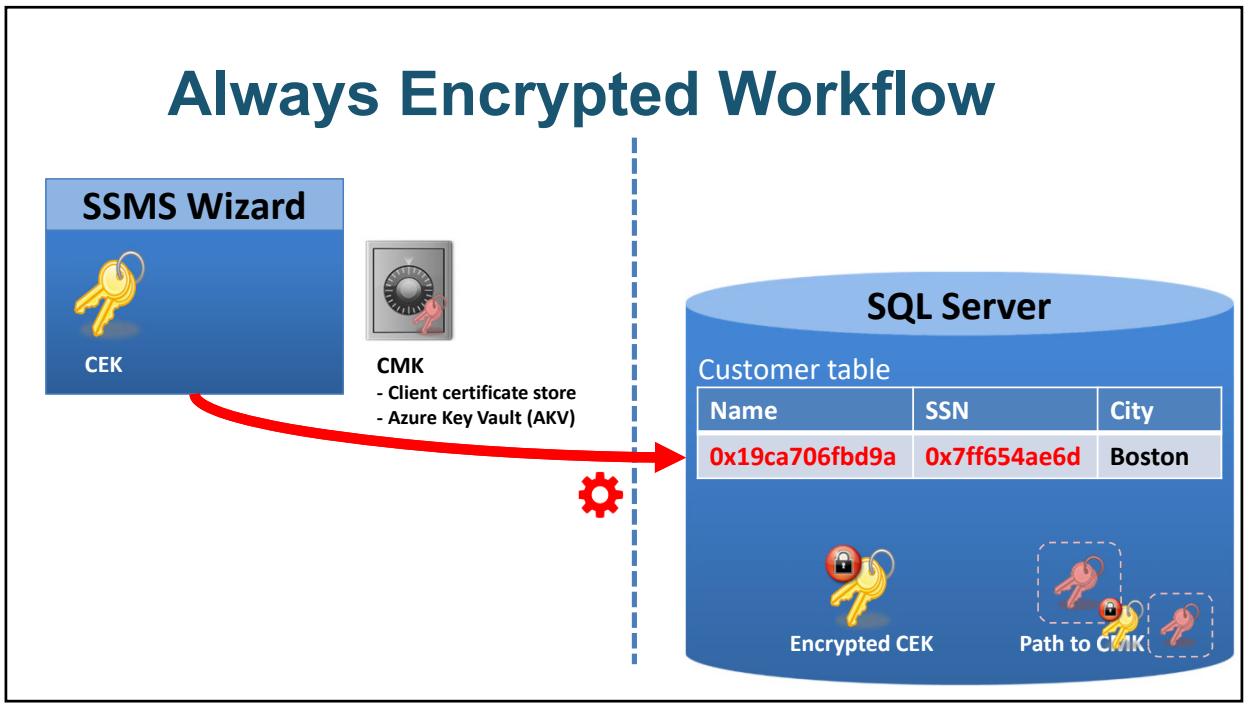
Encryption Types

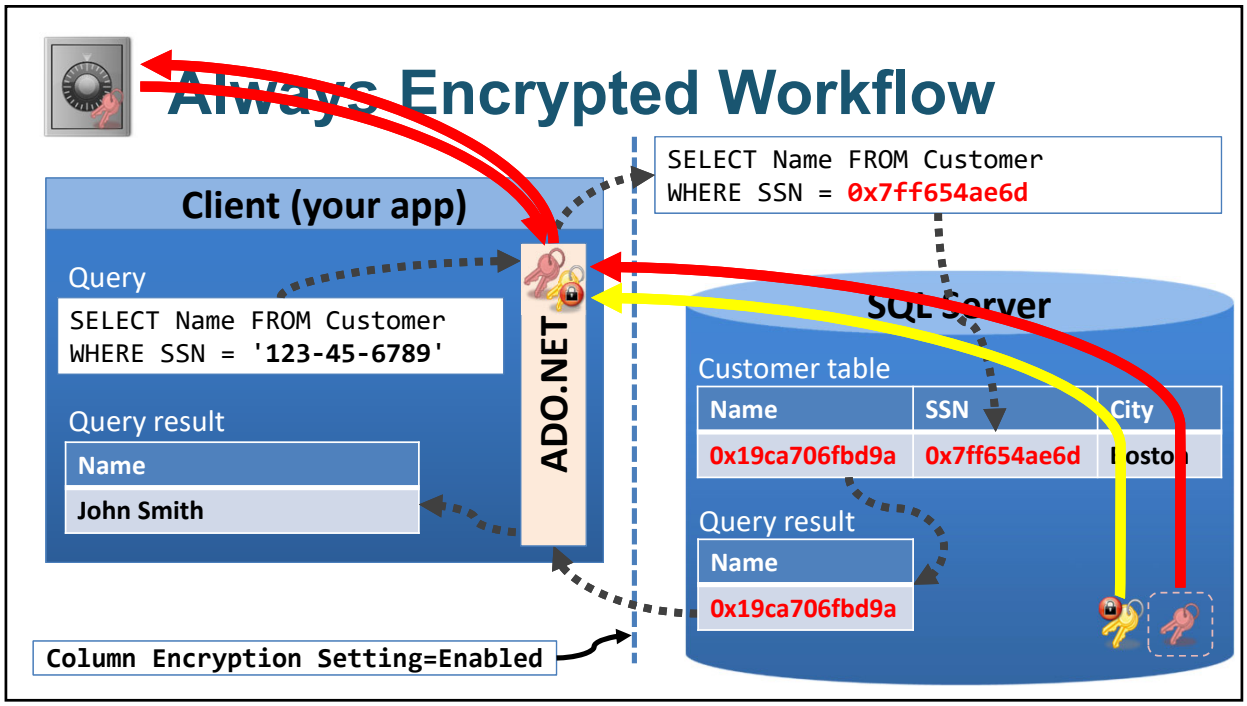
- Randomized
  - Unpredictable, more secure
  - No support for equality searches, joins, grouping, indexing
  - Use for data that is returned but not queried
- Deterministic
  - Predictable, less secure
  - Use for data that must be queried
  - Easier to guess by examining encryption patterns
    - Increased risk for small value sets (e.g., True/False)

Encryption Keys

- Column Encryption Keys (CEK)
  - Used to encrypt values in specific columns
  - Encrypted versions of each CEK is stored in the database
- Column Master Keys (CMK)
  - Used to encrypt all the CEKs
  - Must be stored externally in a secure key store
    - Key store providers: Azure Key Vault, Certificate store, HSM
- CMK rotation
  - Each CEK can have two encrypted values from two CMKs


Always Encrypted Workflow
SSMS Wizard
CEK
CMK
CMK
- Client certificate store
- Azure Key Vault (AKV)
SQL Server
Customer table
Name
SSN
City
John Smith
123-45-6789
Boston
Encrypted CEK
Path to CMK


Always Encrypted Workflow
SSMS Wizard
CEK
CMK
- Client certificate store
- Azure Key Vault (AKV)
SQL Server
Customer table
Name
SSN
City
0x19ca706fbd9a
0x7ff654ae6d
Boston
Encrypted CEK
Path to CMK



Always Encrypted Wizard (SSMS)

- Creates CMK in either:
  - Local Windows Certificate Store
  - Azure Key Vault
- Creates CEKs
  - Then encrypts them from the CMK
- Deploys to database:
  - Encrypted CEKs
  - Path to CMK
- Runs encryption migration
  - Queries the unencrypted table
  - Encrypts client-side (within SSMS)
  - Creates new encrypted temp table
  - Swaps in the new temp table to replace the old unencrypted table

Always Encrypted Catalog Views

- **`sys.column_master_keys`**
  - Identifies each CMK
  - Contains external path to CMK location
- **`sys.column_encryption_keys`**
  - Identifies each CEK
- **`sys.column_encryption_key_values`**
  - Contains CMK-encrypted values of each CEK
- **`sys.columns`**
  - New metadata columns to identify encrypted columns

Always Encrypted (AE)

demo

AE Limitations and Considerations

- Unsupported data types
  - xml, rowversion, image, ntext, text, sql_variant, hierarchyid, geography, geometry
- Also not supported for
  - FILESTREAM, ROWGUIDCOL, sparse, or partitioning columns
  - Fulltext indexes
  - Columns with default constraints
  - Temporal tables
  - Stretch database

Always Encrypted with Secure Enclaves

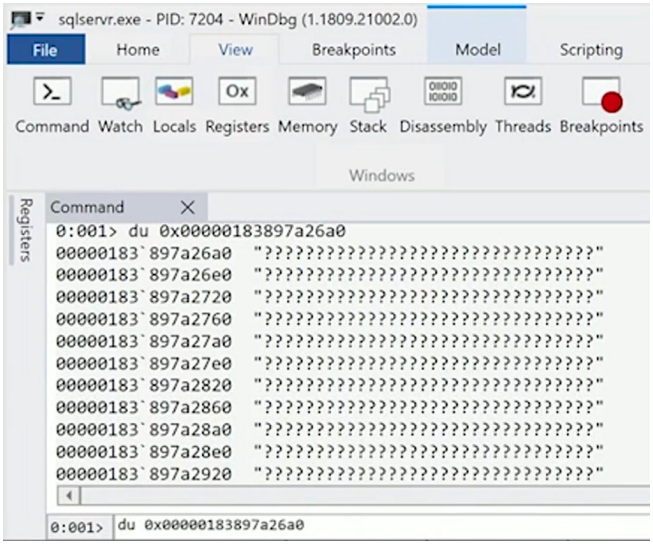
Always Encrypted with Secure Enclaves

- Secure data in use
  - Protect sensitive data from high privileged but unauthorized users
  - Ensure this data is accessible only to trusted code
- Improve functionality
  - Deterministic-based equality queries are very limited
  - Client/server roundtrips are very expensive
- Support rich query processing
  - Enable range queries, sorting, pattern matching
- Provide in-place (server-side) encryption
  - Handle large scale data processing requirements
- Solution:
  - Always Encrypted with Secure Enclaves (SQL Server 2019 for Windows only)

What is an Enclave?

- Isolated protected region of memory
- Normal memory belonging to a normal process
  - But not normal at all
- Region cannot be accessed
  - Not by the containing process or any other process
  - Not by administrators
  - Not by the OS itself
- Trusted execution environment for both code and data
  - Data inside the enclave cannot be accessed from outside
  - Code running in the enclave must be signed in a special way and cannot be modified
- Secure isolation provided by
  - Hardware (e.g., Intel SGX, Arm TrustZone)
  - Hypervisor (e.g., virtualization based security in Windows Server 2019, Windows 10 v1809)

Enclave Data Protection



- Impervious to memory scanning attacks
  - Not present in full memory dump
- Extremely attractive technology
  - Can serve as trusted execution environment for processing sensitive data

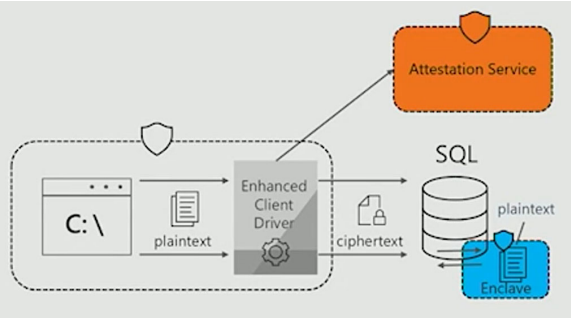
AE w/Secure Enclaves: Server Requirements

https://docs.microsoft.com/en-us/sql/relational-databases/security/tutorial-getting-started-with-always-encrypted-enclaves?view=sql-server-ver15

- Two servers
  - Both can be Azure VMs
- Database Server
  - SQL Server 2019
  - Windows 10 Enterprise 1809+ or Windows Server 2019 Datacenter
  - Can be an VM that supports nested virtualization
    - E.g., Standard D2s_v3 on Azure
- Attestation Server
  - Windows Server 2019 Standard or Datacenter
  - Can be a VM in the same resource group
  - Requires 2 CPUs and 8 GB RAM
    - E.g., Standard D2s_v3 on Azure

Enclave Attestation

- How does your app know the enclave can be trusted?
  - By using an attestation protocol and an attestation service
- Client application and SQL Server both interact with an external attestation service
  - Prove to the client that the enclave is trustworthy, before the client authorizes the use of the enclave



AE with Secure Enclaves: How it Works

- SQL Server loads a VBS enclave on startup
  - Enclave acts as an extension of the client on the server machine
- During query processing
  - SQL Server delegates operations requiring decryption to the enclave
- Creates new possibilities
  - Rich computation
    - Starting with LIKE, range, indexing
- Also delegates cryptographic operations
  - Provides in-place encryption
- How can the enclave encrypt/decrypt data?
  - SQL Server never has the CEK
- When attestation succeeds
  - Client driver negotiates a shared secret over a secure tunnel to the enclave
  - CEK is encrypted/decrypted using the shared secret

Always Encrypted (AE)
with Secure Enclaves

demo

Data Classification

Data Classification in SQL Server 2017

- Address the growing trend for privacy
  - HIPPA, GDPR
- New SSMS tooling (SQL Server 2017)
  - Data Classification Wizard
- Analyzes column names
  - Make recommendations to classify columns via a *label* and *information type*
    - Information type; e.g., contact info, name, financial
    - Sensitivity Label; e.g., Confidential, Confidential-GDPR, HIPAA
- Limitations of the 2017 tooling
  - No built-in classifications (relies on extended properties)
  - No built-in auditing (required for GDPR)

Data Classification in SQL Server 2019

- Built-in sensitivity classifications
- New T-SQL statements
  - ADD SENSITIVITY CLASSIFICATION
  - DROP SENSITIVITY CLASSIFICATION
- New catalog view
  - sys.sensitivity_classifications
- New audit property
  - data_sensitivity_information

Data Classification and Auditing

demo

TDE Suspend and Resume

TDE Suspend and Resume

- What is Transparent Data Encryption?
  - Introduced in SQL Server 2008 (still Enterprise-only)
- Encrypts the entire database automatically
  - Data files, log files, and backups too
- Data is encrypted/decrypted on the fly
  - Encryption occurs at the page level on background thread
- Enabling TDE can be intensive
  - SQL Server reads and writes every page to/from disk
  - Very large databases can impact mission-critical applications

TDE Suspend and Resume

- Minor but important SQL Server 2019 enhancement
- New T-SQL statements:
  - `ALTER DATABASE db_name SET ENCRYPTION SUSPEND`
  - `ALTER DATABASE db_name SET ENCRYPTION RESUME`
- New DMV columns in sys.dm_database_encryption_keys
  - `encryption_scan_state`
  - `encryption_scan_state_desc`
    - RUNNING, SUSPENDED, COMPLETE
  - `encryption_scan_modify_date`

Part 5

Data Virtualization

PolyBase

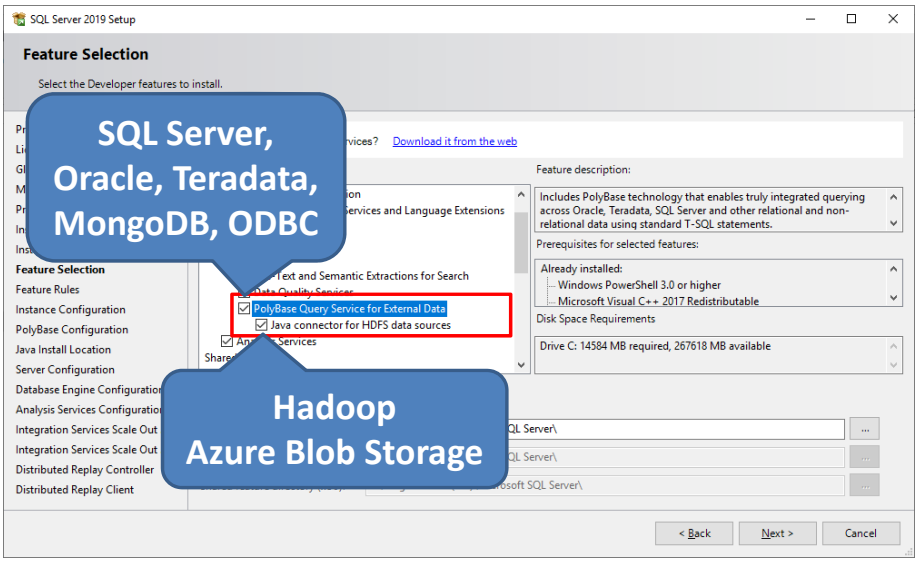
- Query external data
  - Supported data sources (2016)
    - Hadoop
    - Azure blob storage
  - Supported data sources (2019)
    - SQL Server
    - Oracle
    - Teradata
    - MongoDB (Cosmos DB)
    - ODBC
- Use ordinary T-SQL queries against “external” tables
  - Can be relational or non-relational
  - Seamlessly JOIN with relational SQL Server data
  - Can push computation down to external data source

PolyBase vs. Linked Servers

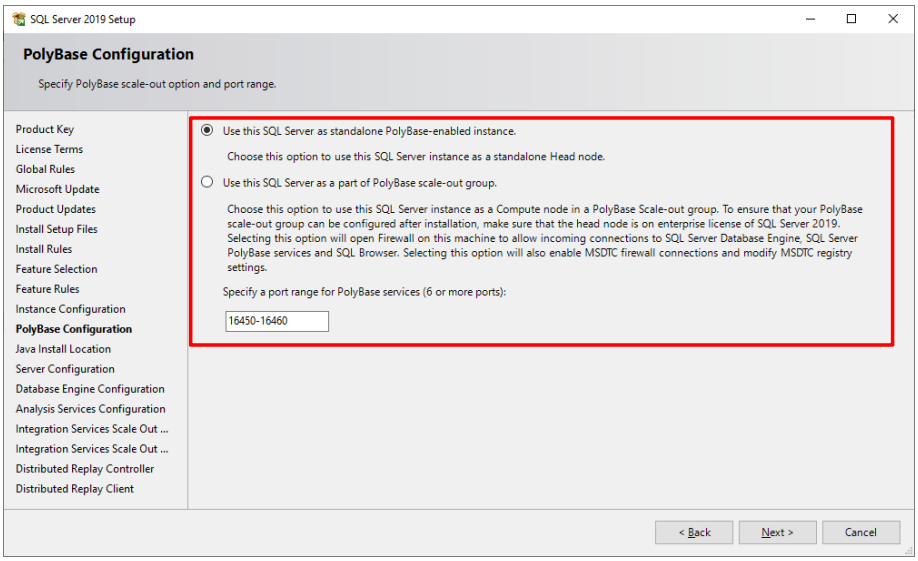
PolyBase	Linked Servers
ODBC	OLE DB
Read/Write for Hadoop only Read-only for other data sources	Read/Write
Horizontally scalable	Single connection
Analytic queries over large rowsets	OLTP queries over small rowsets

- https://docs.microsoft.com/en-us/sql/relational-databases/polybase/polybase-faq?view=sql-server-ver15

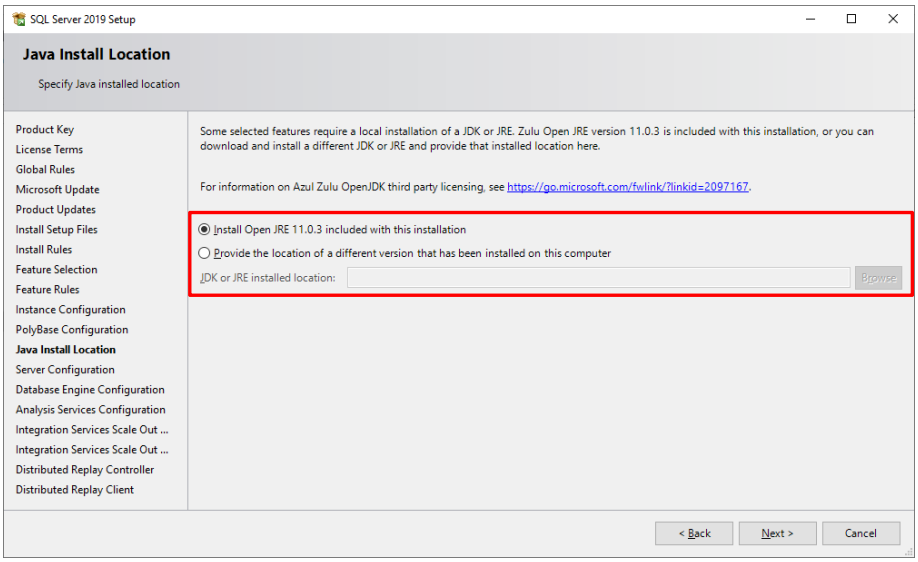
PolyBase: Setup



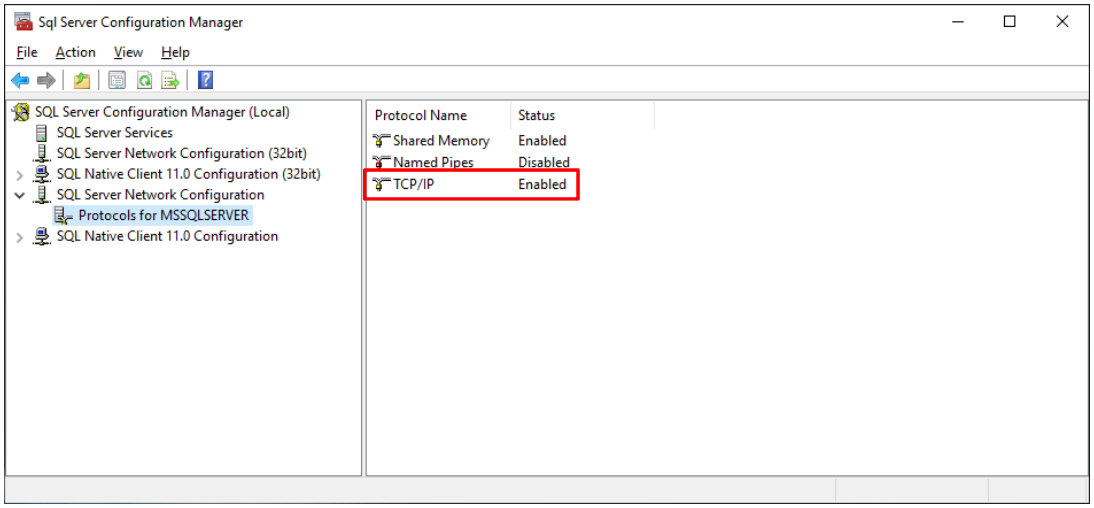
PolyBase: Setup



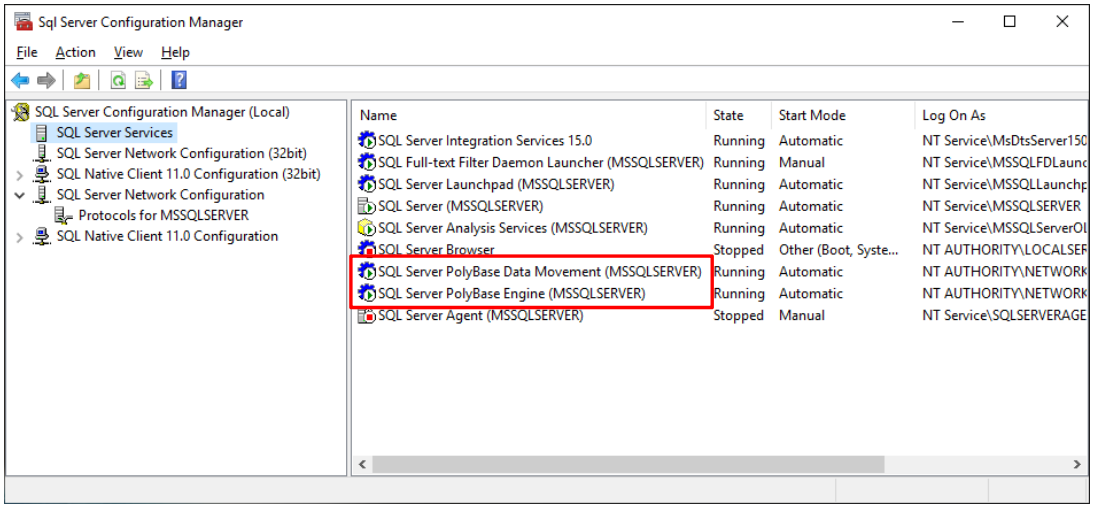
PolyBase: Setup



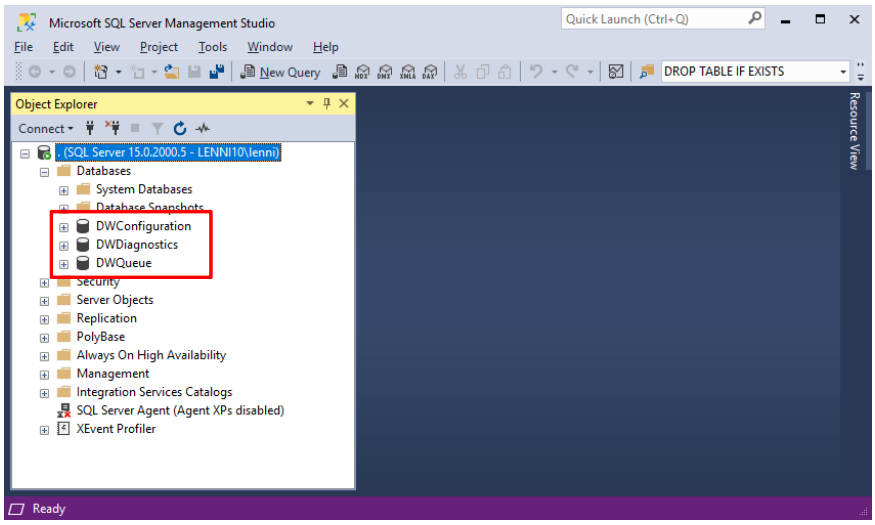
PolyBase: Setup



PolyBase: Setup



PolyBase: Setup



PolyBase: The Steps

- Create master key
- Store credentials for external data source
  - Encrypted by master key
- Define the external data source
  - Point to location
  - Include credentials
  - Enable pushdown computations
- Create an external table
  - Map a SQL Server table definition to the external schema

PolyBase T-SQL

- **oracle://***host:port*
- **sqlserver://***server*.database.windows.net
- **mongodb://***uri:port*
- **wasbs://***storageacct*.blob.core.windows.net
- **odbc://***hostname* +
  CONNECTION_OPTIONS="Driver=…"

```
CRE
 EN                                                    l'
CRE
 WI                                                    d'
CREATE EXTER   DATA SOURCE ExtDS WITH
 (LOCATION='path'                    DENTIAL=MyCreds
CREATE EXTERNAL TABL  table (columns…)
 WITH (LOCATION = 'path', DATA_SOURCE = ExtDS)
```

database.owner.table
database.container

PolyBase With Hadoop/ABS

- Example:

```
SELECT *
 FROM SqlServerTable AS s
 INNER JOIN HadoopFile AS h ON h.key = s.key
```

- SQL Server does a cost-based analysis
  - Stream the dataset back from Hadoop, or
  - Run MapReduce job on Hadoop (parallel processing) and stream back just the results

PolyBase Scale-Out Groups

- Hadoop and ABS data can be extremely large
  - Tens of terabytes, or even petabytes
  - How can a single SQL Server instance handle all that volume?
- PolyBase scale-out group
  - Multiple SQL Server instances linked together
    - One head node, multiple compute nodes
  - Enables massive parallel processing

PolyBase

demo

Thank You!

- Contact me
  - **lenni.lobel@sleektech.com**
- Visit my blog
  - **lennilobel.wordpress.com**
- Follow me on Twitter
  - **@lennilobel**
- Thanks for coming! ☺