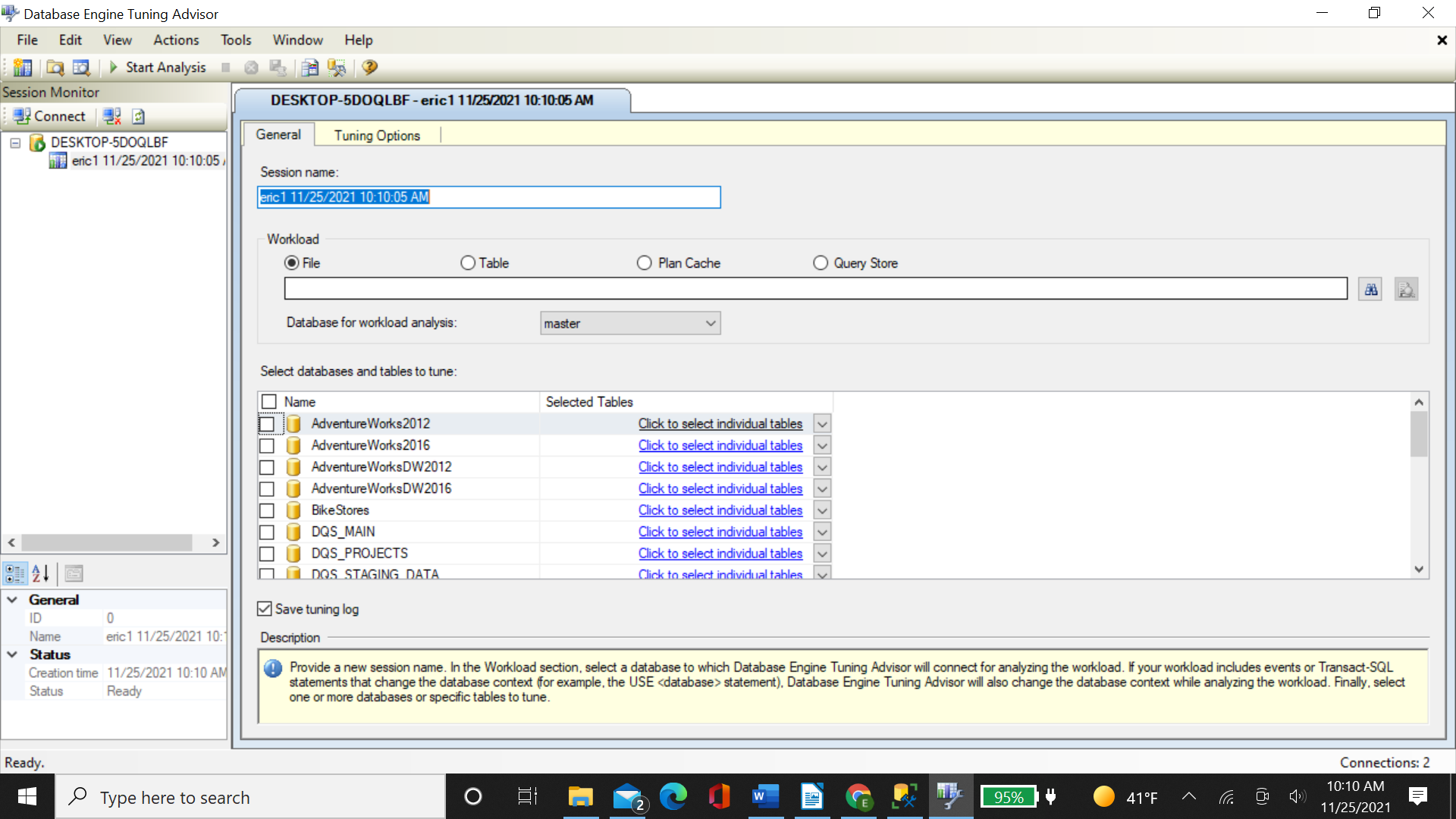
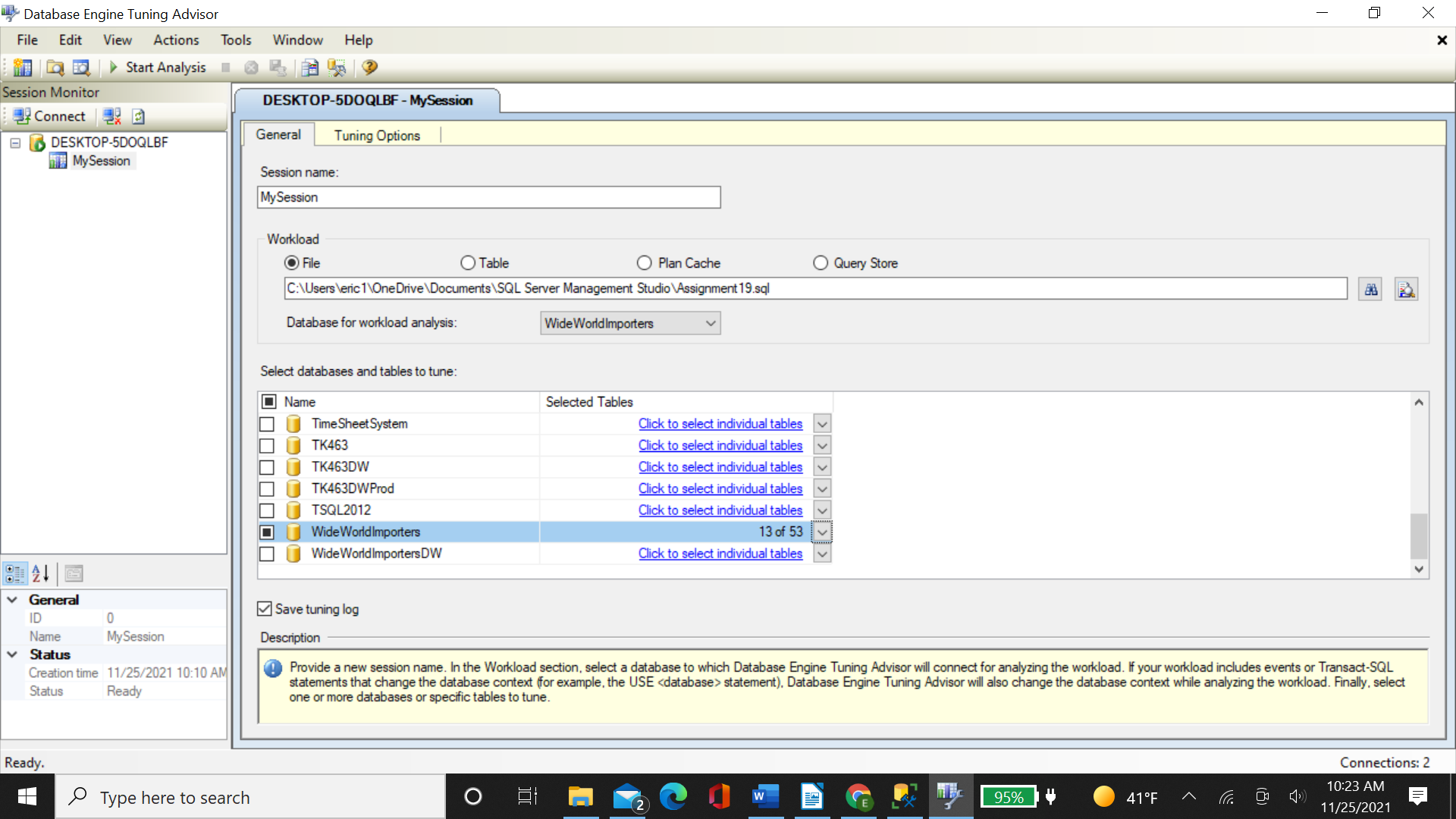
SQL server performance tuning is aimed to make queries of a relational database run as fast as possible. Although performance tuning is very complicated, this task was made easier with tools provided by SQL server management studio, such as *Tuning Advisor, Extended Events, Dynamic Management Views and Functions, Logs, and Execution Plan*.

*Database Engine Tuning Advisor* tool has a user-friendly interface. It examines how queries are processed in the database and then makes recommendations on database structure modification such as changing indexes, indexed view, and partitioning.

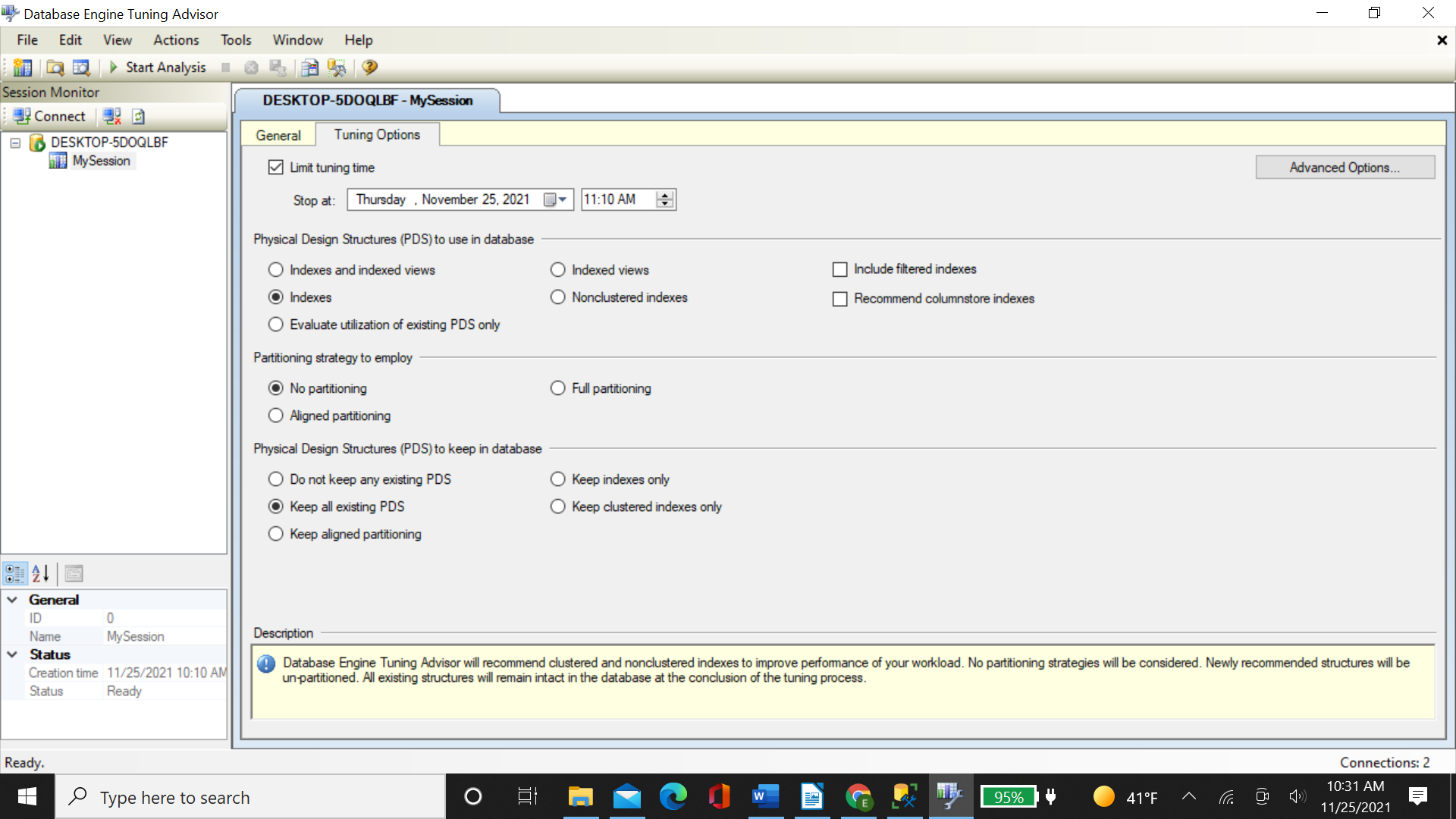
First, load the *Advisor* from SSMS Tools menu on the top of window. The *connect to server* interface will pop up. Click *connect* and a dialogue window as shown below will appear:



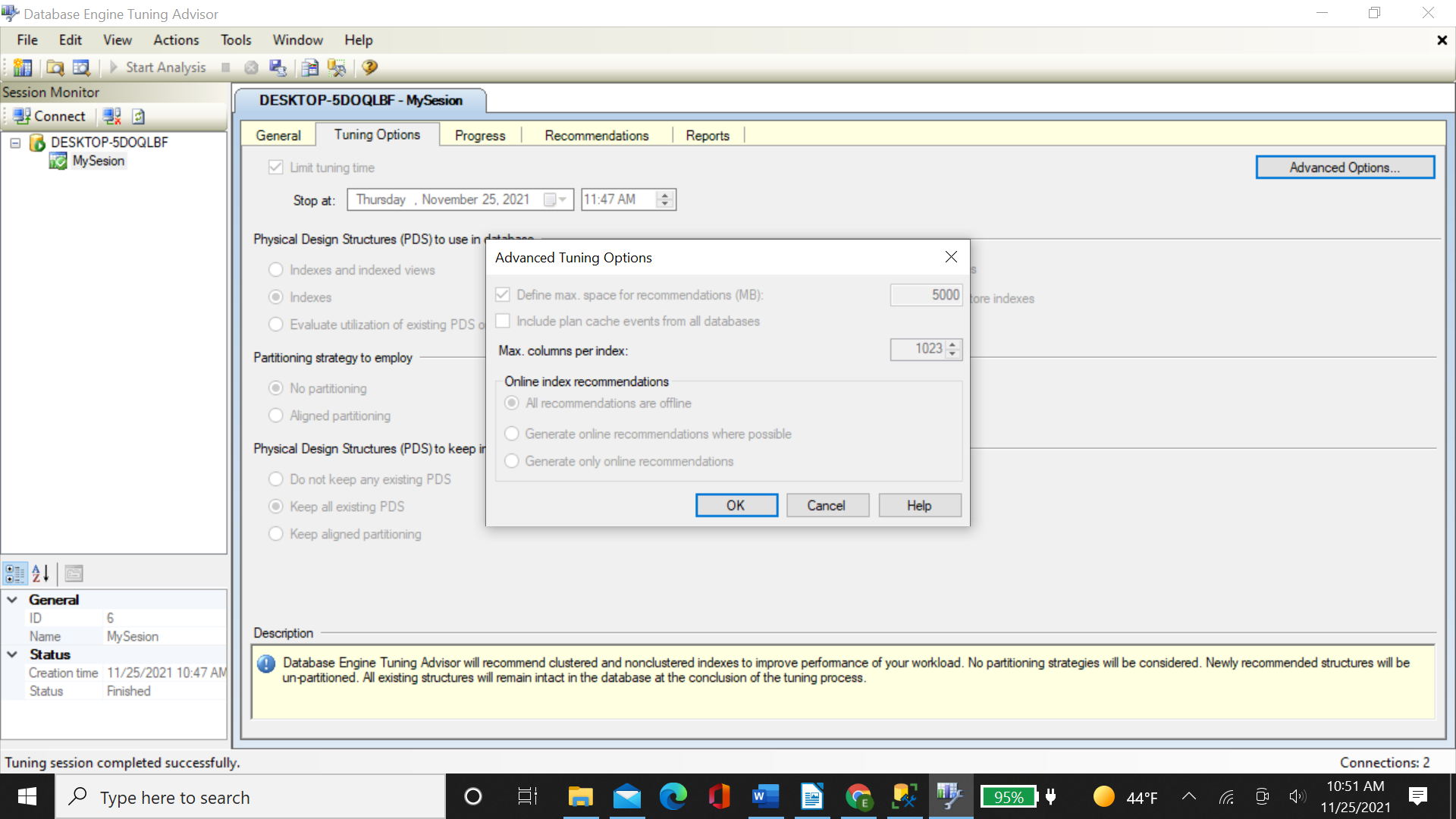
We rename the session to *MySession*, choose File as *Workload*, input the file path, select *WideWorldImporters* database for analysis, and then choose this database again as tuning target. We can also narrow it down to tables under WWI database by *Selected Table* menu on the right side. Down at the bottom, check *Save tuning log*. All steps above are shown in the figure below:



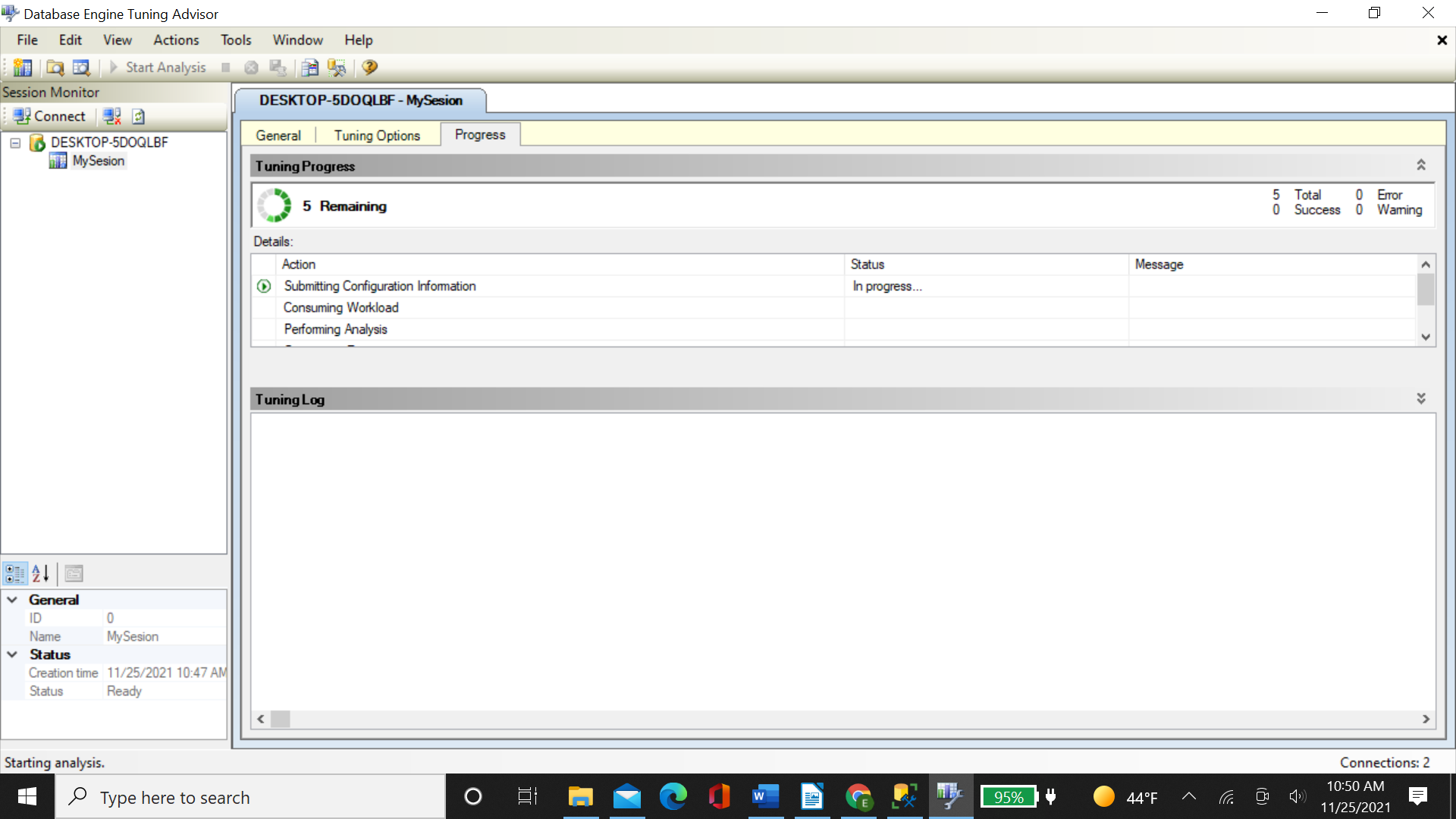
Then we move to the *tuning option*, the second tab on the top of window. Here we limit the tuning time to 10 minutes, select *indexes for PDS* to use option, select *No partitioning* for Partitioning option, and keep *all existing PDS* for *PDS to keep in database* (see the figure below):



Then click *Advance Options* on the top right corner and define the *max space for recommendation* as 5000 MB. At this point we have finished the setting. Click the *Start Analysis* menu to kick off the Advisor.



When Advisor is running, the progress is shown on the screen as:



When it is done, the *Recommendations and Reports* tabs show up on the top of session window. Under Recommendations tab, we can find all the actions recommended by *advisor*. By clicking the *Definition* column, we can find the T-SQL script code that *Advisor* created for every action. Under *Reports* tab, *tuning summary* presents some statistics of tuning process and how much your query performance will be improved by percentage. *Tuning reports* shows the details about your query cost, index usage, etc. For our example, it recommends creating one non clustered index and the performance will by increased by 30% following this action.

Graphical user interface, text, application, email

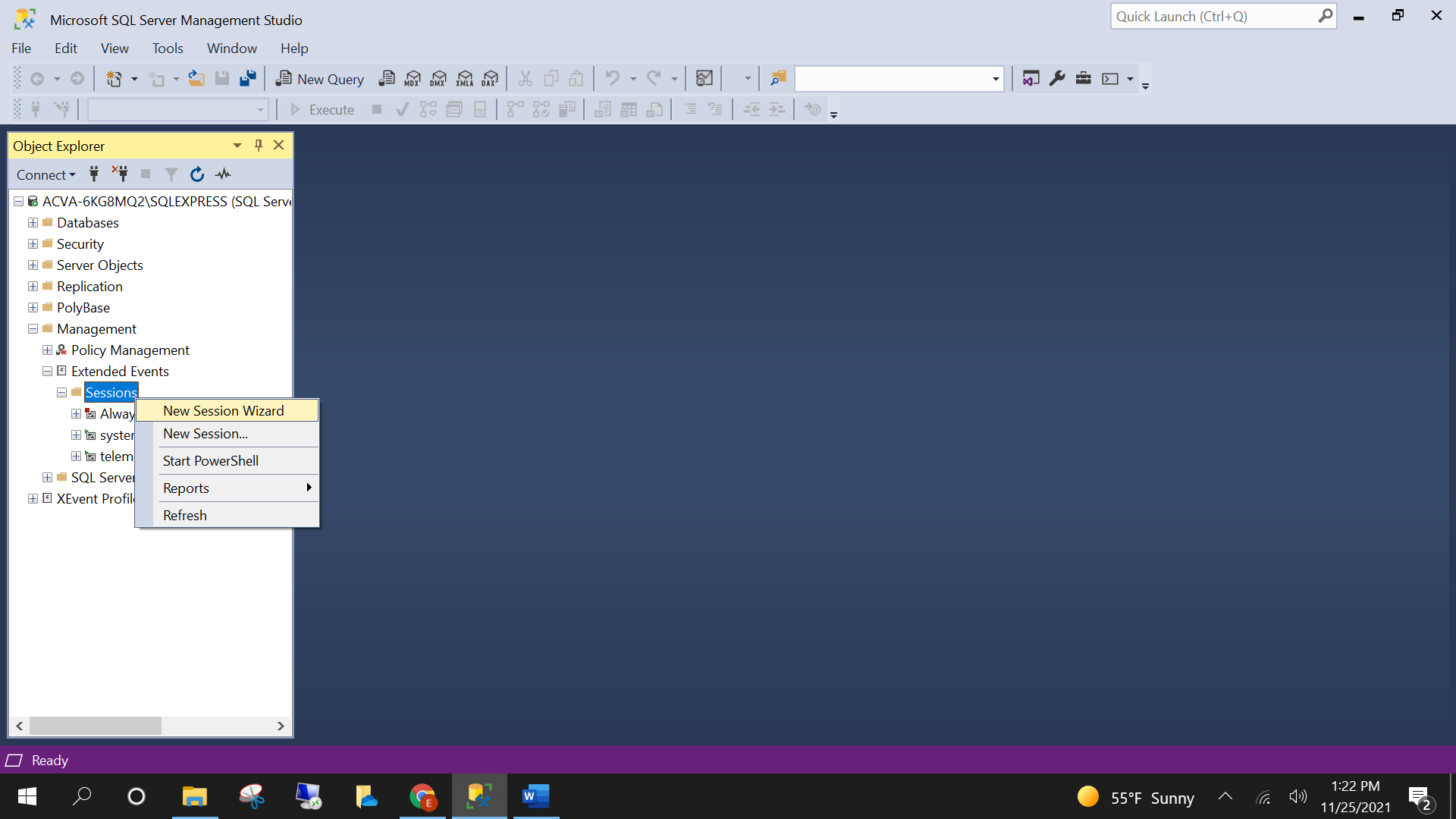
Description automatically generated

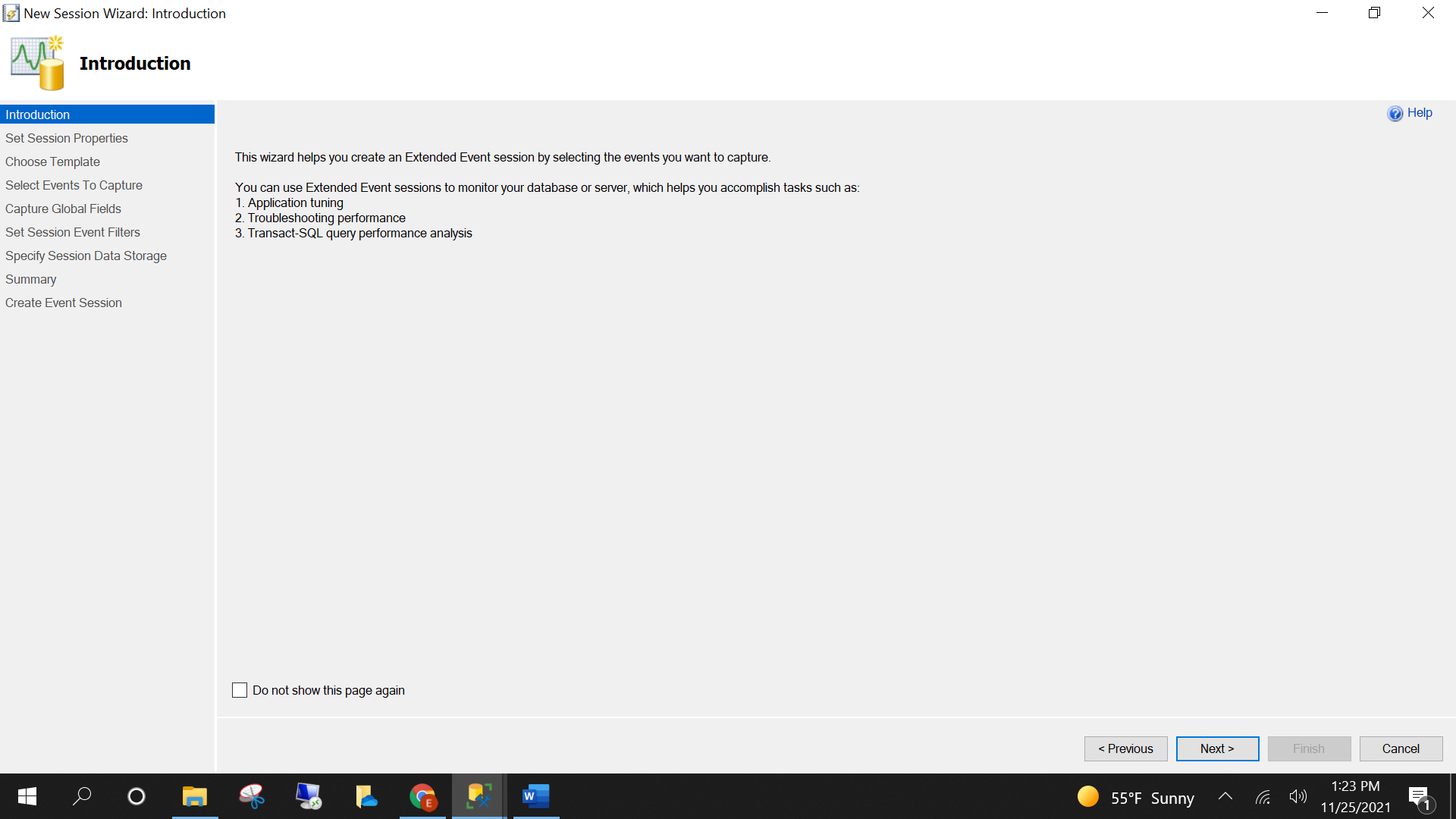
Graphical user interface, text, application, email

Description automatically generated  
  
Graphical user interface, text, email

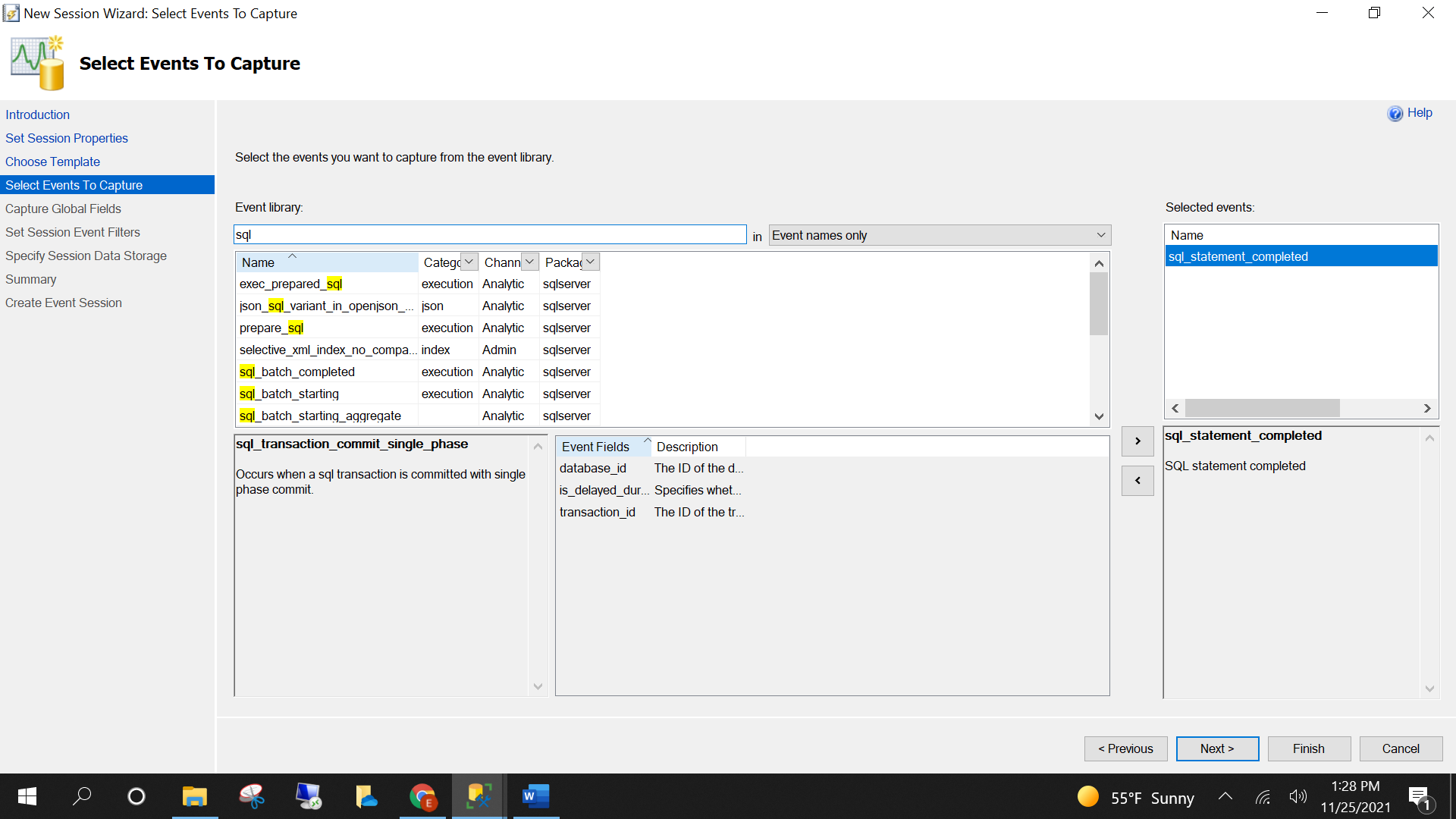
Description automatically generated

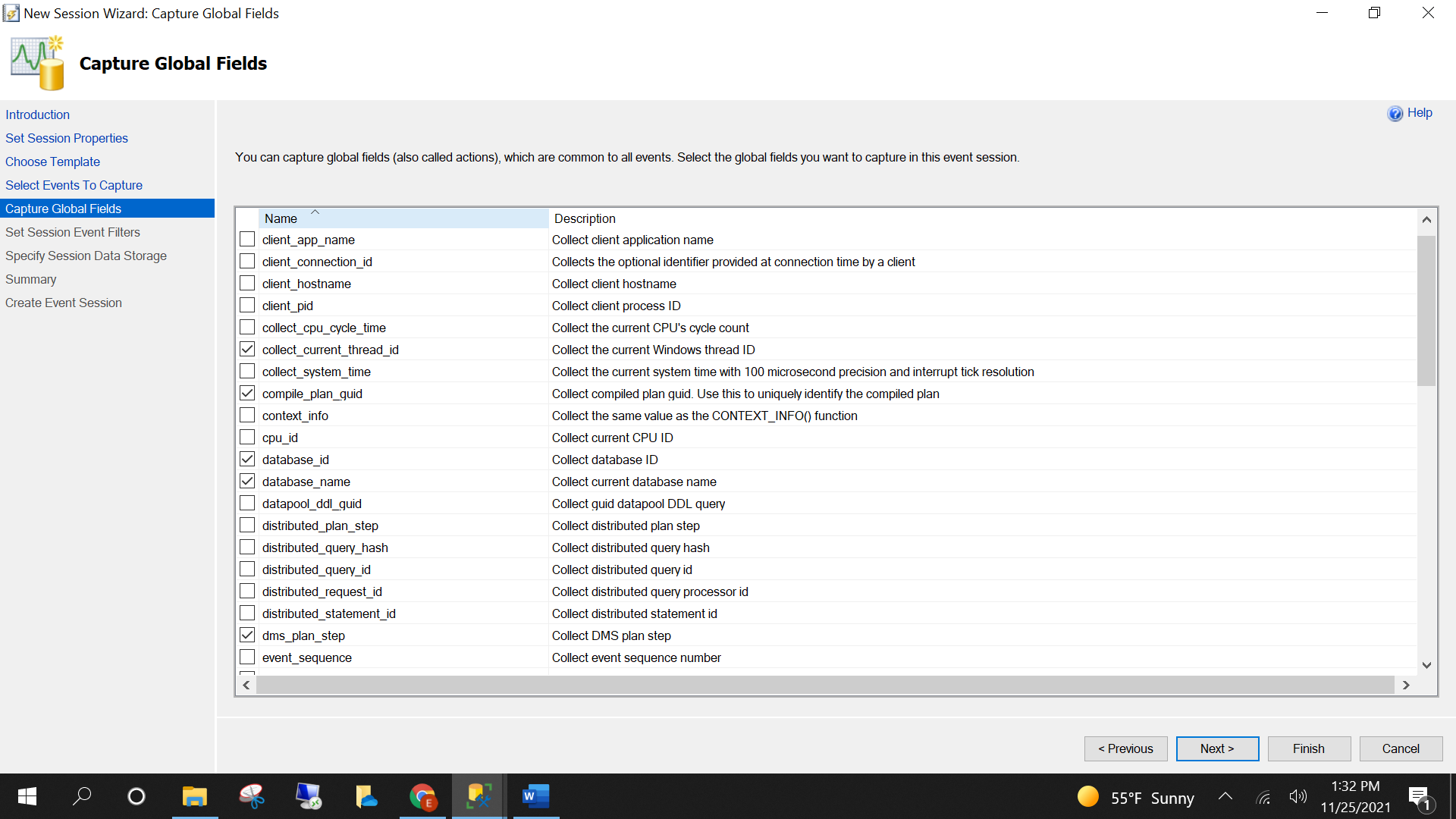
*Extended Events* is a lightweight performance monitoring system which can gather information needed to improve query performance. Like *Tuning Advisor*, it has a Wizard to help you easily create a session. As shown in the figure below, we need to go to Object Explorer >Management>Extended Events>Sessions. After right clicking *Sessions*, *New Session Wizard* menu will pop up.

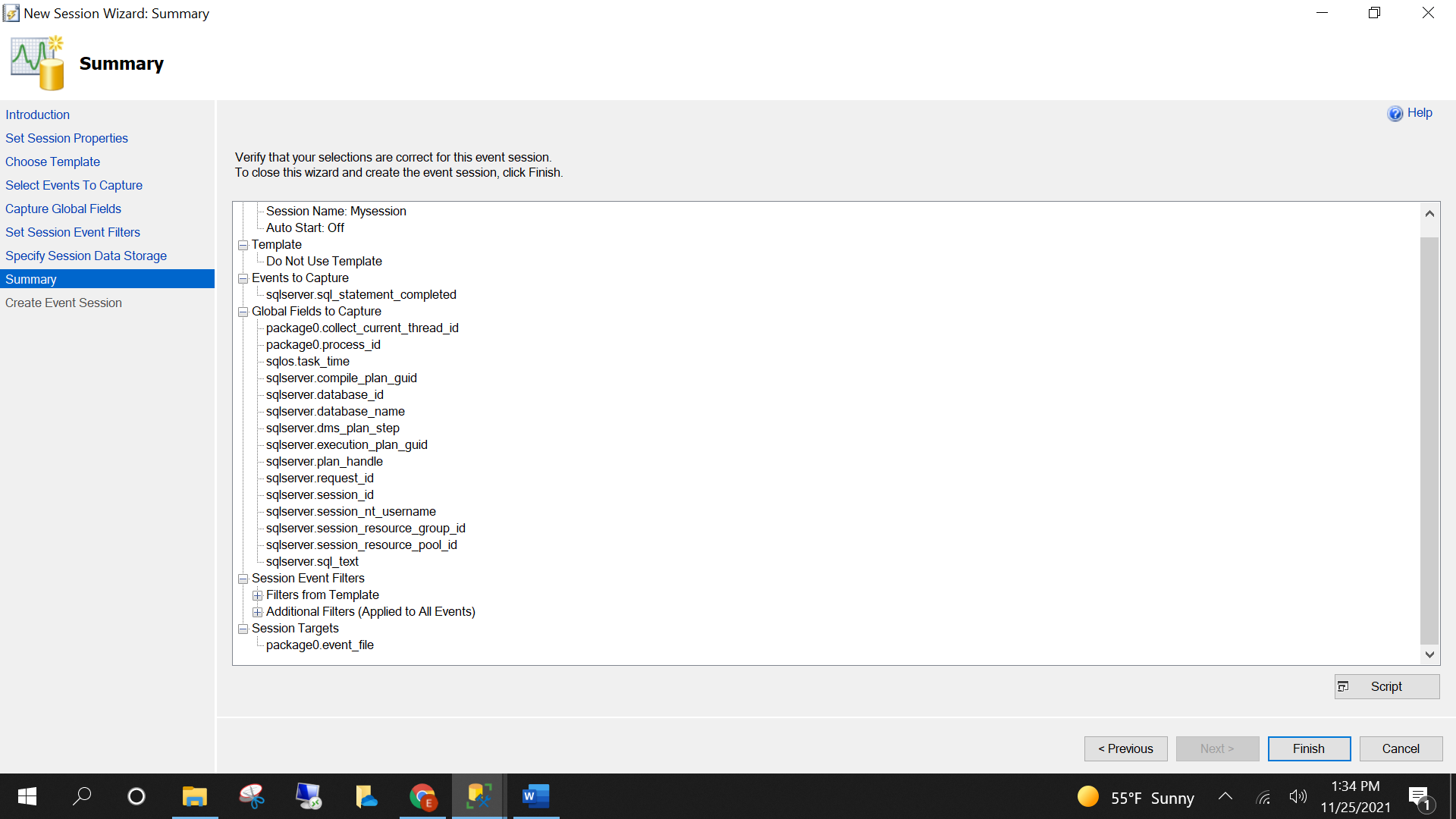


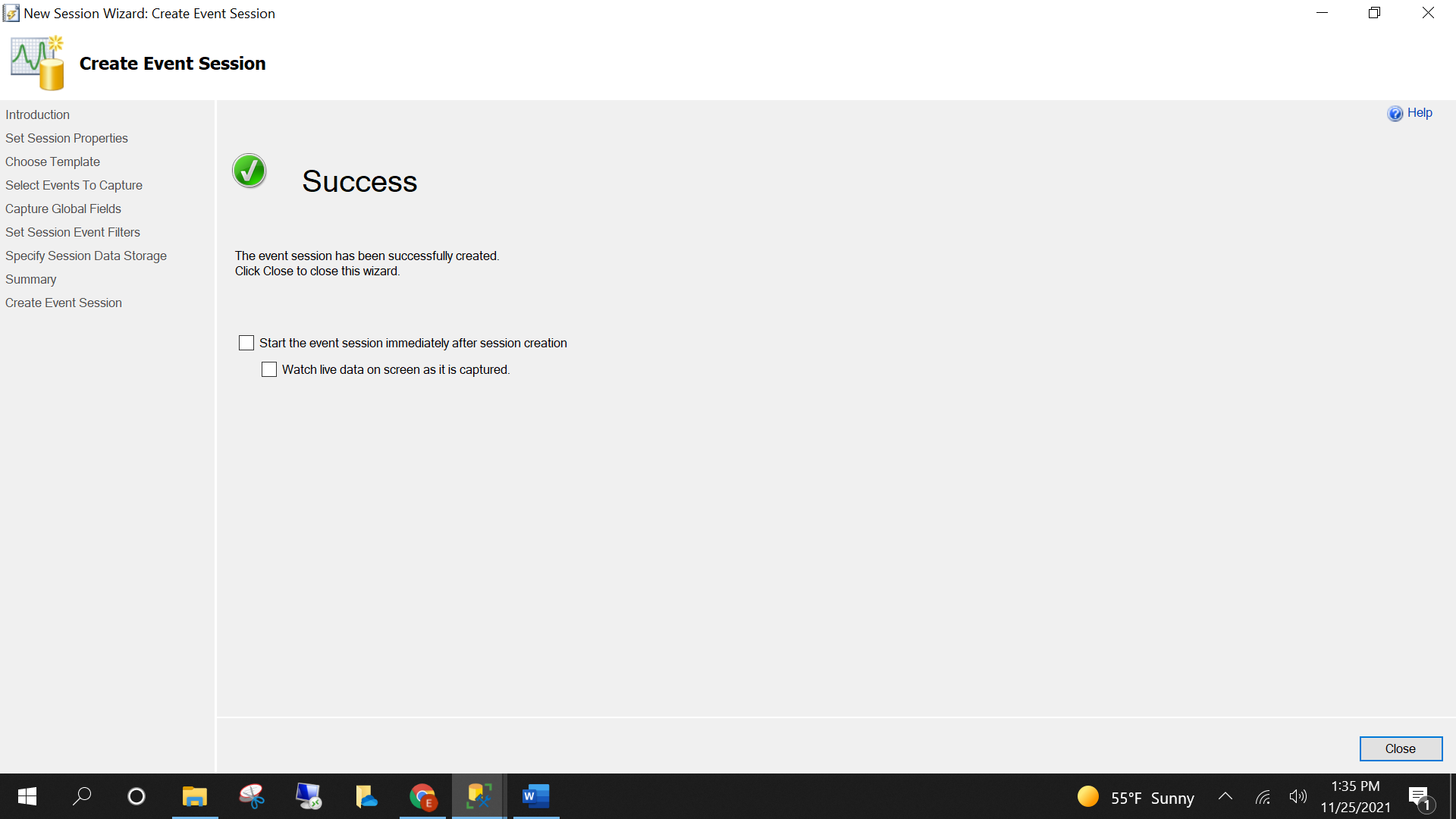


After setting the session name and selecting *do not use template* option, we move to phase 3 - *select the events to capture*. Here we choose sql\_statement\_completed event.

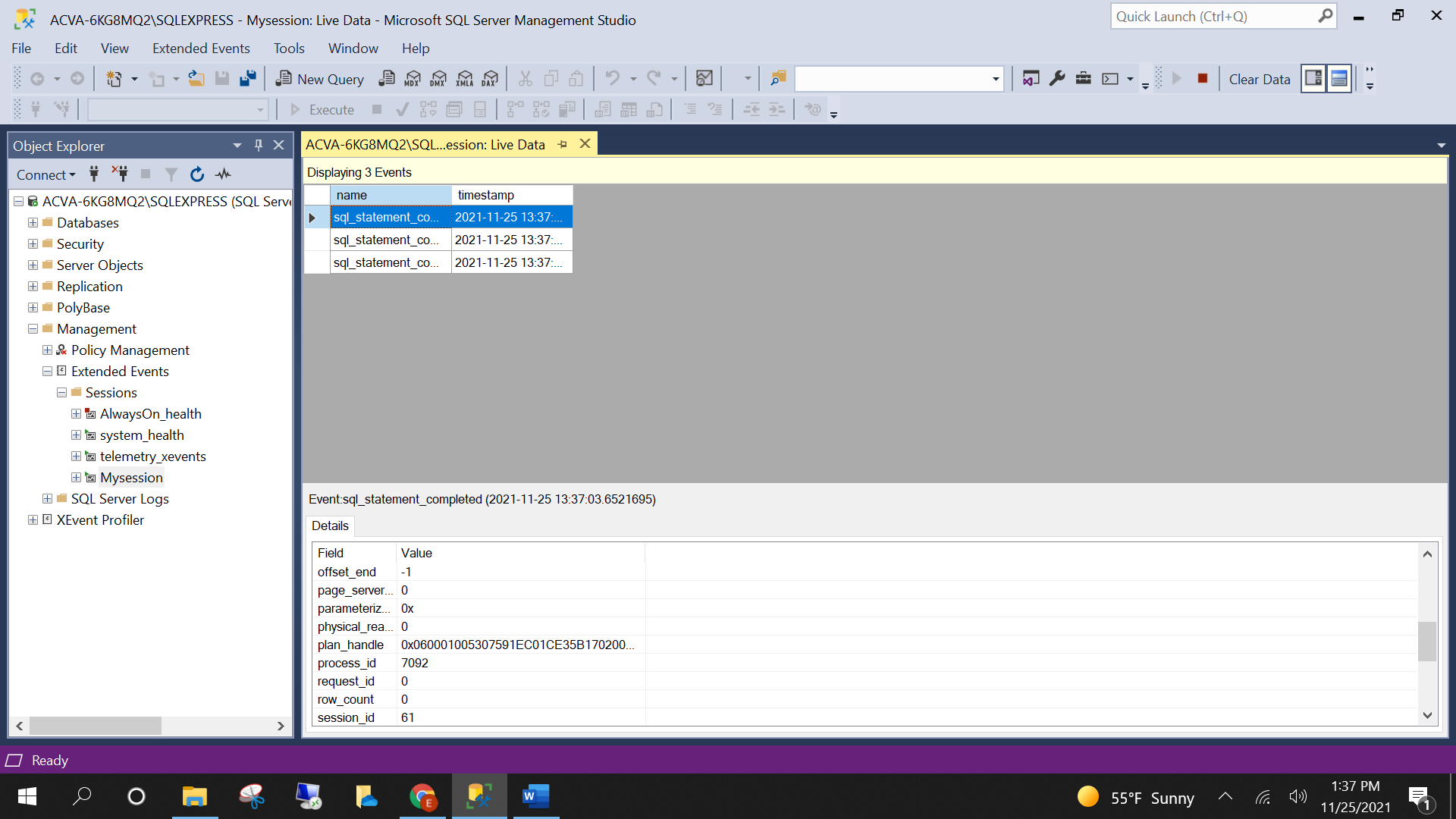




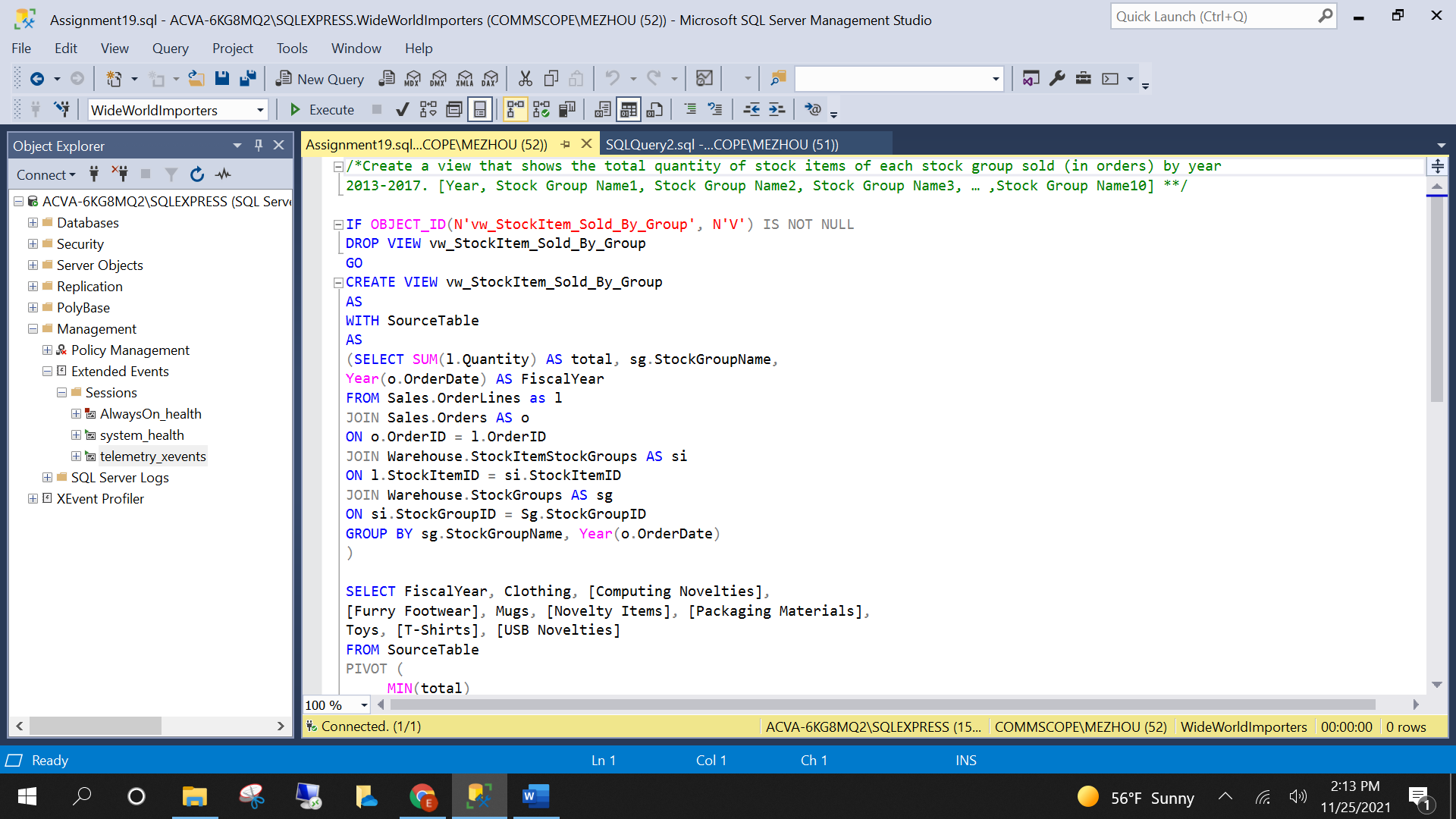


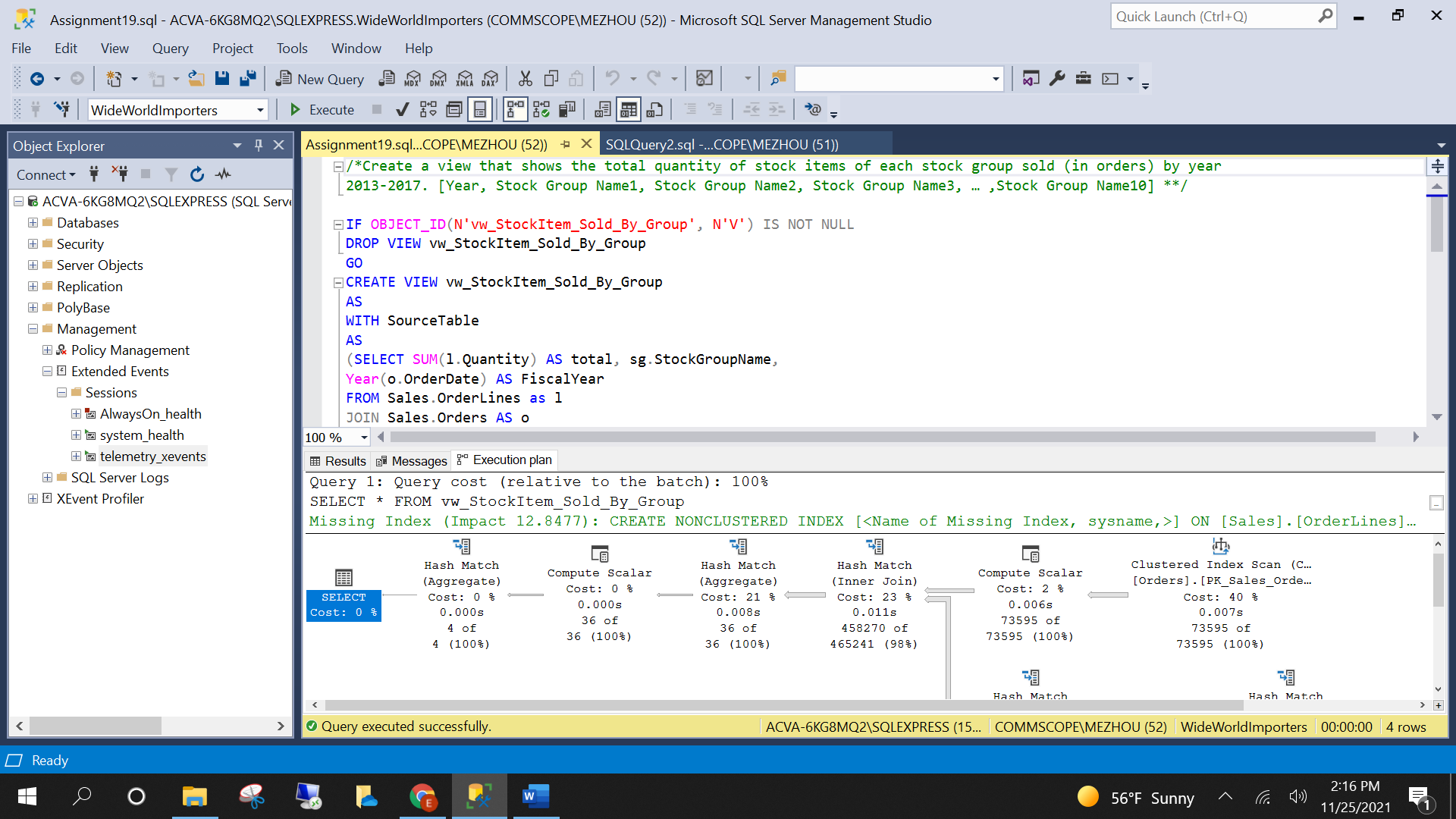


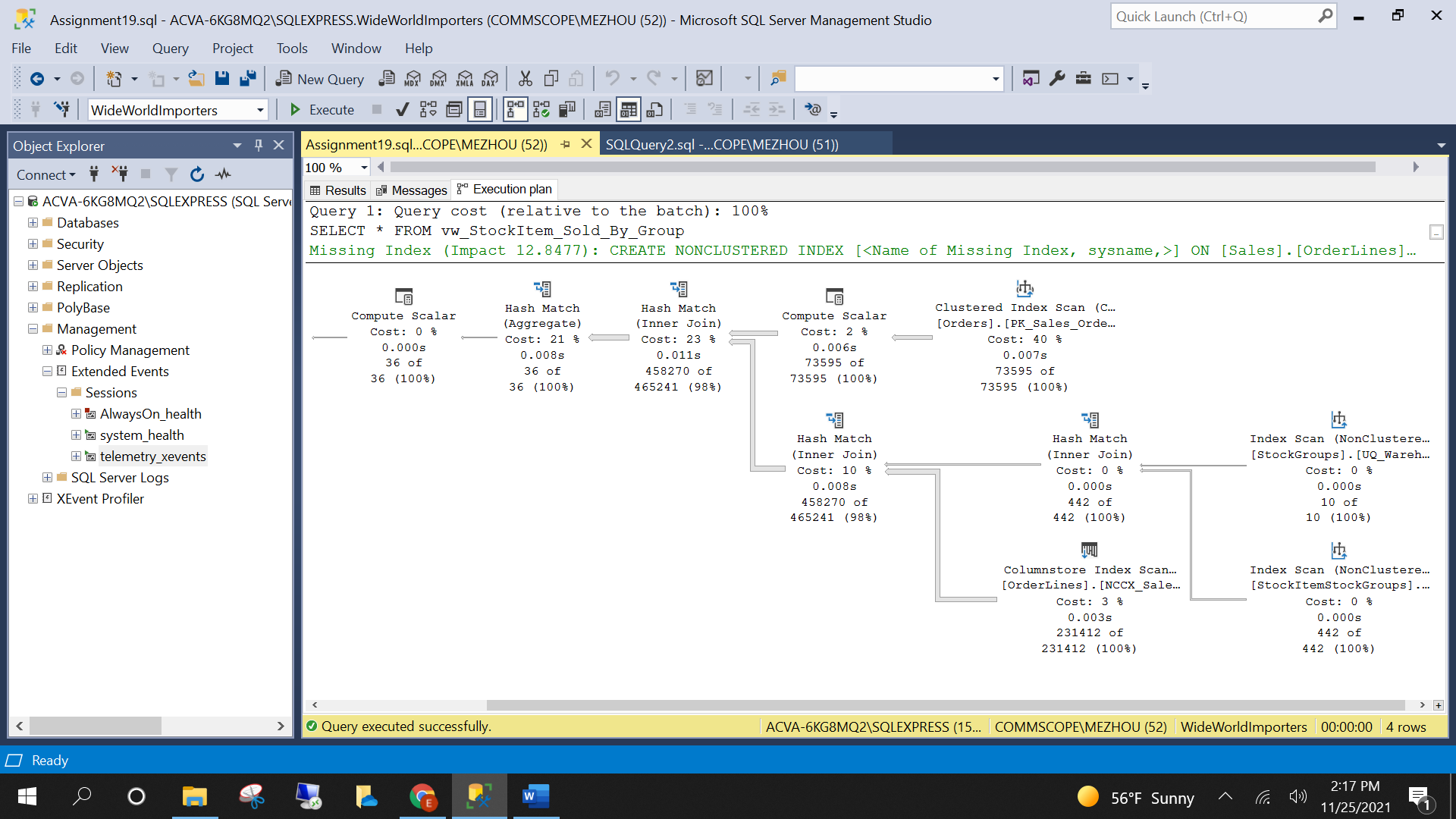
The figure below shows *Extended Events* is running and collecting the live data. We can select each event to check the statistics of those fields you picked before. With the information collected by Extended Events, you can have better chance to solve your query performance issue.



Another useful feature that comes with SSMS is *Execution Plan*. This tool provides the most accurate data on how the query is executed. By checking the diagram of executed plan, you can find what to tune. Sometimes SSMS will recommend what action to take in the middle window. After selecting the *include actual execution plan* in the top window, the query will be executed. The SSMS will give the result back with the information on how it was done (see the figure below).

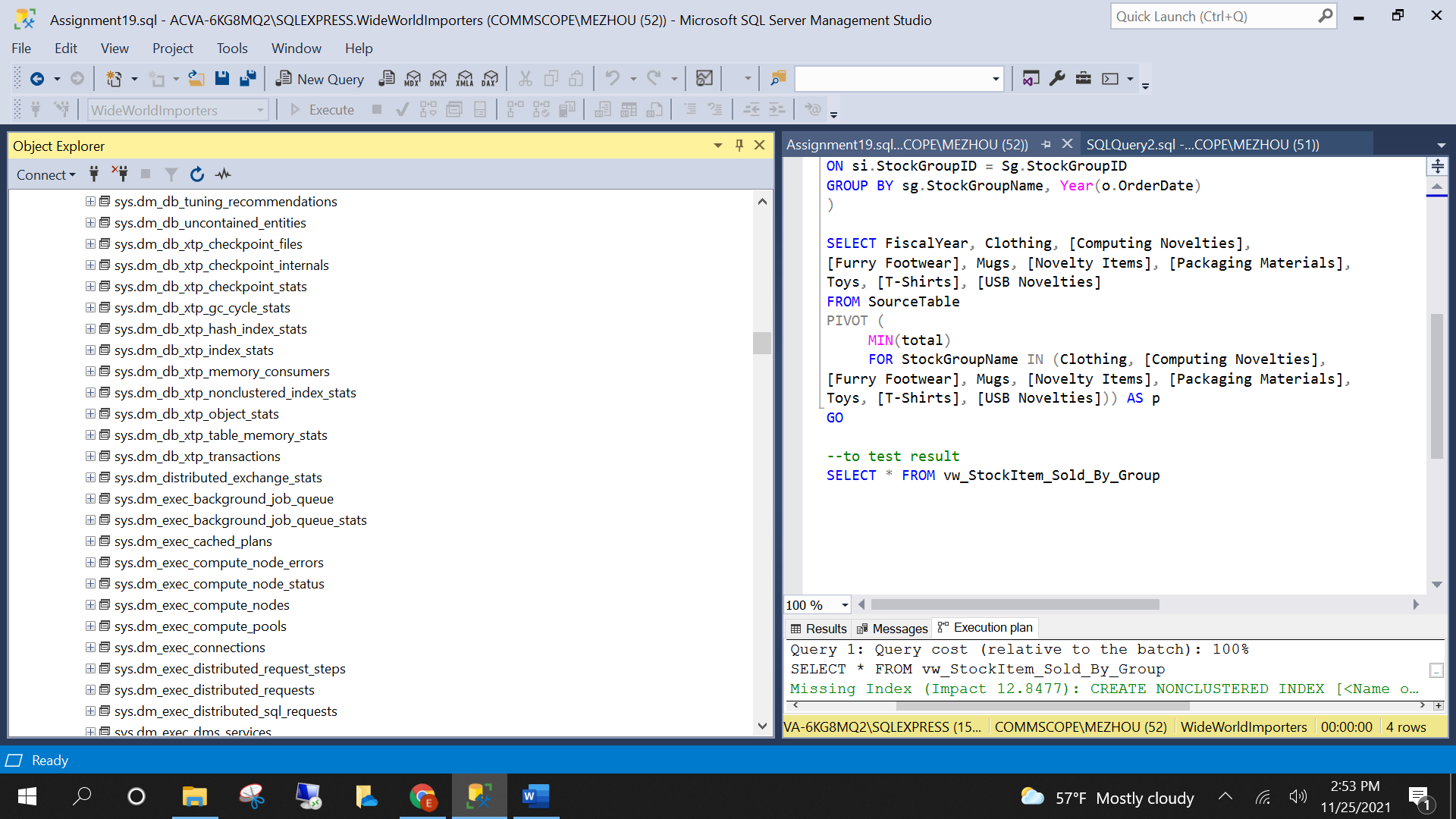






The diagram shows the most expensive part of the query is *clustered index scan*. The recommended action is to create no clustered index, which is consistent with what Tuning Advisor recommended.

DMV contains not only views, but functions as well. They return server state information that can be used to monitor the health of a server instance, diagnose problems, and tune performance. With this information, we can build different queries to collect the data needed. Because there are so many views and functions, it is impossible to list all of them. The figure below shows part of them:



The DMVs starting with *sys.dm* returns information on server scope and the ones starting with *sys.dm\_db* returns information on database scope. For tuning performance, the most popular DMVs are: *sys.dm\_exec\_sessions, sys.dm\_exec\_connections, sys.dm\_db\_index\_usage\_stats, sys.dm\_exec\_sql\_text, sys.dm\_os\_wait, sys.dm\_exec\_query\_plan, sys.dm\_os\_performance\_counters, sys.dm\_exec\_catched\_plans, sys.dm\_exec\_session\_wait\_stats*.

We can build a query based on DMV to list top 10 queries that have the most reads:

1. SELECT TOP(10) DB\_NAME(t.[dbid]) AS [Database],
2. REPLACE(REPLACE(LEFT(t.[text], 255), CHAR(10),''), CHAR(13),'') AS [ShortQueryTXT],
3. qs.total\_logical\_reads AS [TotalLogicalReads],
4. qs.min\_logical\_reads AS [MinLogicalReads],
5. qs.total\_logical\_reads/qs.execution\_count AS [AvgLogicalReads],
6. qs.max\_logical\_reads AS [MaxLogicalReads],
7. qs.min\_worker\_time AS [MinWorkerTime],
8. qs.total\_worker\_time/qs.execution\_count AS [AvgWorkerTime],
9. qs.max\_worker\_time AS [MaxWorkerTime],
10. qs.min\_elapsed\_time AS [MinElapsedTime],
11. qs.total\_elapsed\_time/qs.execution\_count AS [AvgElapsedTime],
12. qs.max\_elapsed\_time AS [MaxElapsedTime],
13. qs.execution\_count AS [ExecutionCount],
14. CASE WHEN CONVERT(nvarchar(max), qp.query\_plan) LIKE N'%%' THEN 1 ELSE 0 END AS [HasMissingIX],
15. qs.creation\_time AS [CreationTime]
16. ,t.[text] AS [Complete Query Text], qp.query\_plan AS [QueryPlan]
17. FROM sys.dm\_exec\_query\_stats AS qs WITH (NOLOCK)
18. CROSS APPLY sys.dm\_exec\_sql\_text(plan\_handle) AS t
19. CROSS APPLY sys.dm\_exec\_query\_plan(plan\_handle) AS qp
20. ORDER BY qs.total\_logical\_reads DESC OPTION (RECOMPILE)

Last, you can use DBCC command to tune performance too. SQLPERF command returns the transaction-log space usage for all databases and the active threads on the server. PERFMON returns information about the I/O work that SQL Server has been performing, the page cache state and operation, and network statistics. SHOWCONTIG command shows the internal state of extents and pages. They are helpful in determining how SQL Server is likely to perform when reading data from a table.