1. Write a short essay talking about your understanding of transactions, locks and isolation levels.

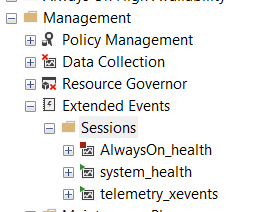
Transaction is a set of commands that must be executed as a set. If one of the commands in the transaction cannot be executed, the whole transaction will not be executed. Transactions have four characteristics: Atomicity, Consistency, Isolation and Durability.

Locks are used in Pessimistic Concurrency Control to prevent users from modifying data in a way that affects other users. If a lock is applied, other users need to wait until the transaction is committed/rollbacked and then can start their own transactions. Lock types include shared lock (allow multiple read transactions, but prevent transactions that modify the data), update lock (prevent other transactions from simultaneously updating the data), exclusive lock (prevent all other transactions to access the data until the transaction commit/rollback)

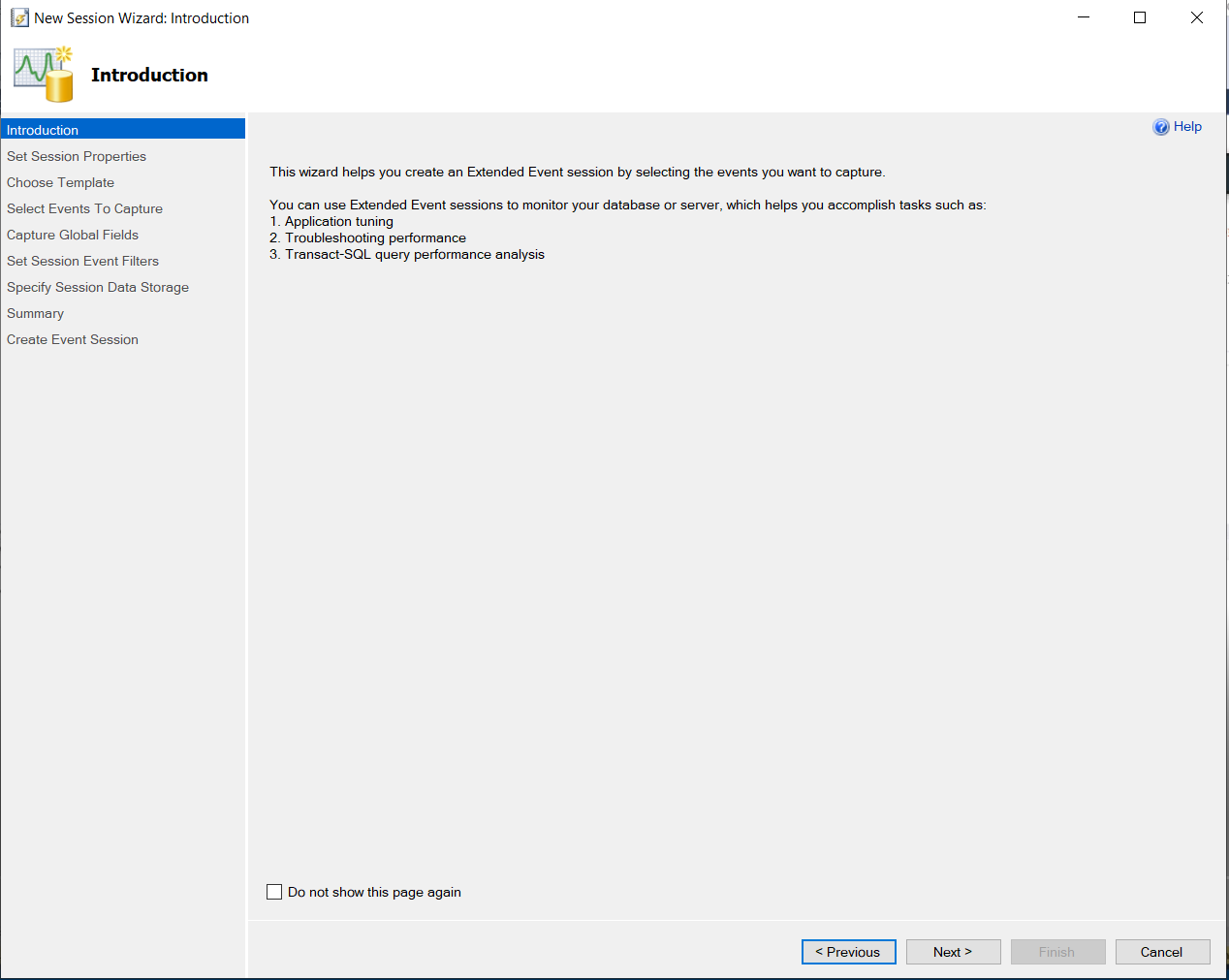
There are four isolation levels, all of which are used to guarantee the accuracy of the data (i.e., make sure the data is the most up-to-date version). The four levels are: Read Uncommitted, Read Committed, Repeatable Read, and Serializable. Read Uncommitted means there’s no lock applied when reading the data. Even if someone alter the data at the same time when reading the data, the data read will be the old version. Read Committed let the transaction holds a read or write lock on the current row, and thus prevent other transactions from reading, updating or deleting it. Repeatable Read has the locks applied to Read Committed, and the only difference is that the shared lock will be there until the transaction end, instead of gone soon after the data is read. Serializable is the highest level of isolation. It has all constraints in Repeatable Read plus the constraint that when data is locked with shared lock, we cannot insert anything until the transaction ends.

1. Write a short essay, plus screenshots talking about performance tuning in SQL Server. Must include Tuning Advisor, Extended Events, DMV, Logs and Execution Plan.
2. Extended Events:

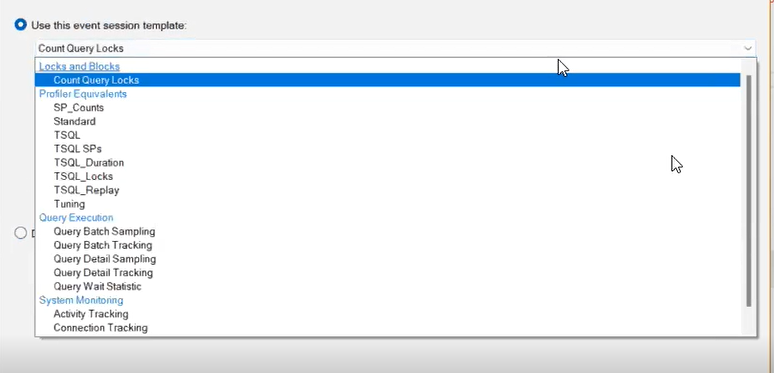
Using SQL Server Event Bubbling System



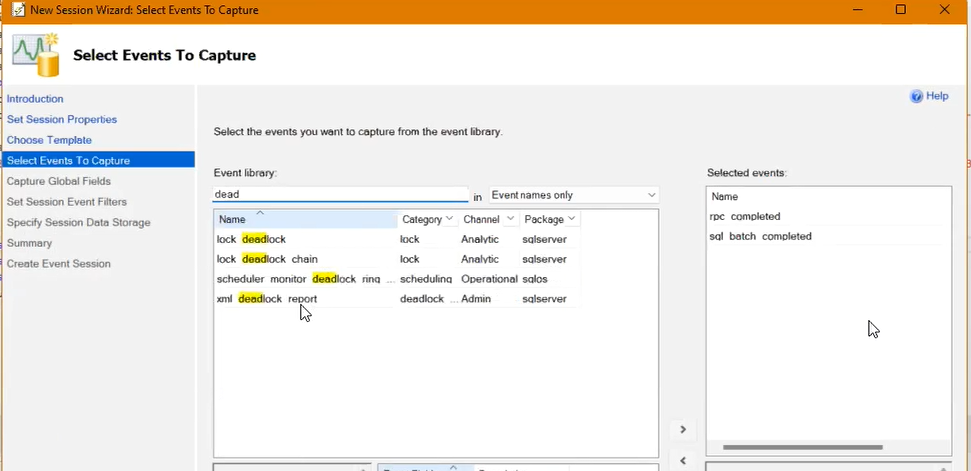
Start a new session using New Session Wizard:



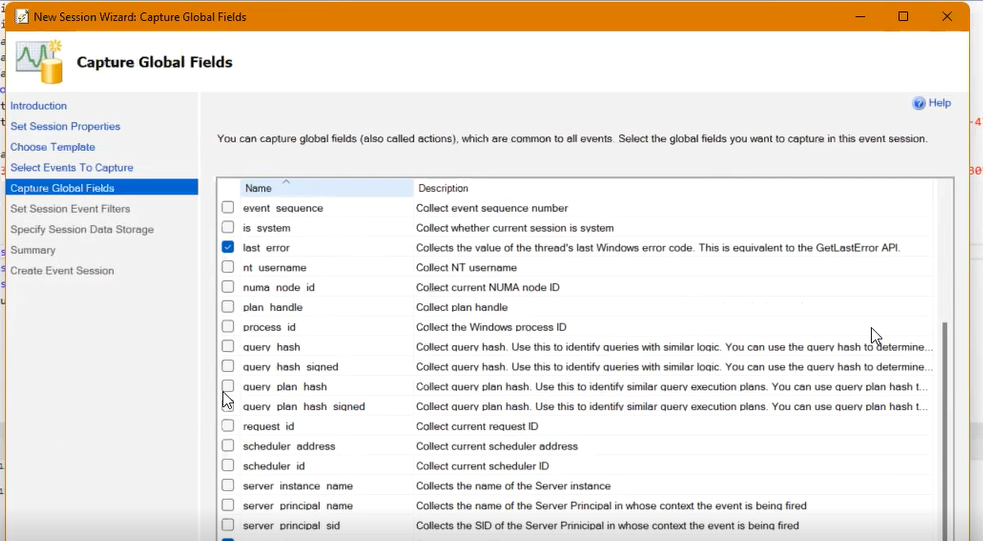
Choose a template:



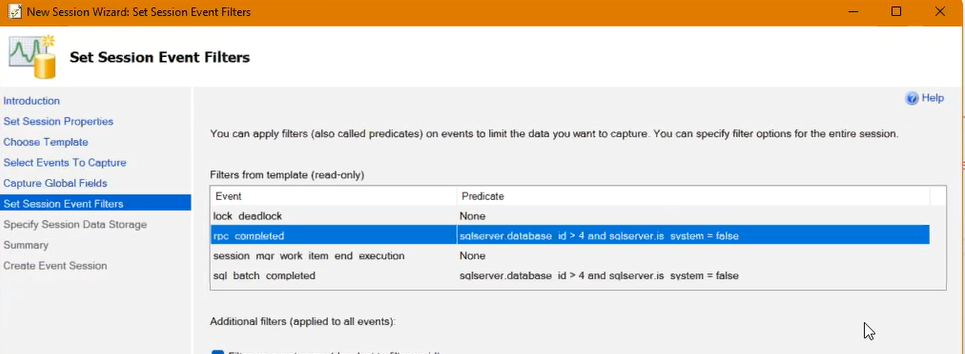
Select Events to capture, the right column are events that are already selected:



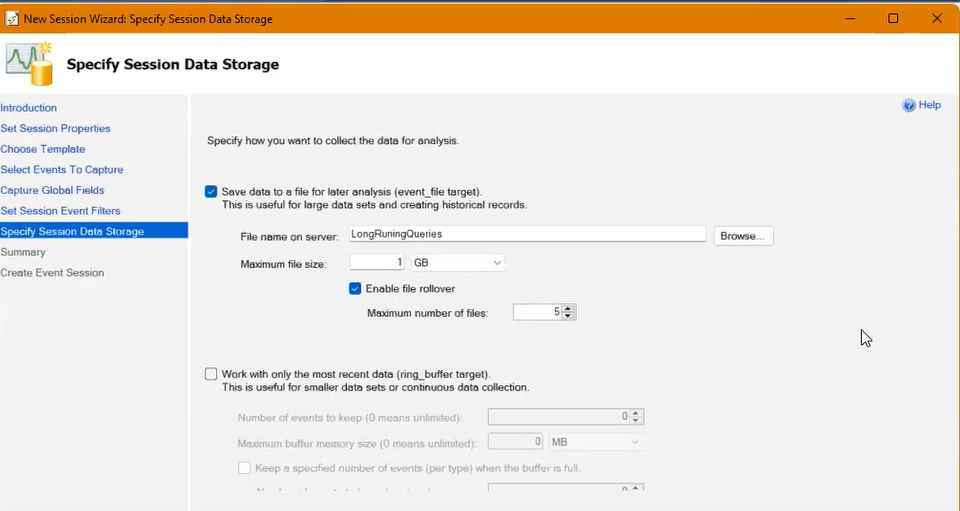
Choose data other than events that you want to capture:



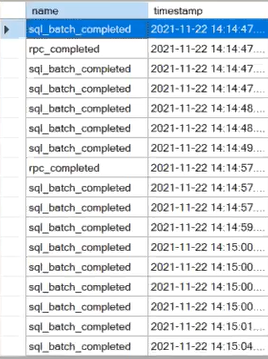
Specify filter so that can customize threshold (e.g., specify what should be the maximum time before the query is captured as long running query.



Choose if want to store the data for later analysis:

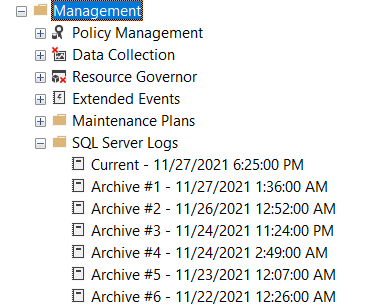


After setting up and running some queries, there will be logs that captures all the actions:



Stop session after finishing.

1. Logs



Use DBCC queries to turn on flag for certain events

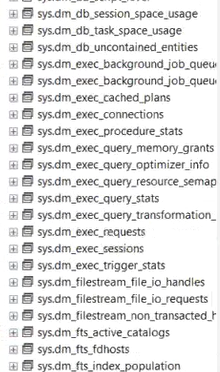
Language : DBCC TRACEON(error\_code, -1);

So that when the certain error happens, the log will catch it.

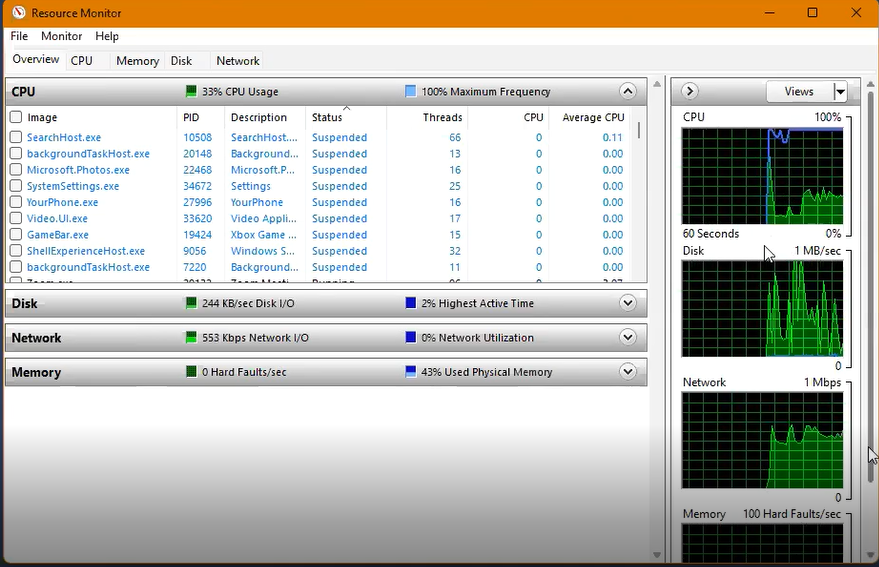
1. DMV

“Dynamic Management Views”

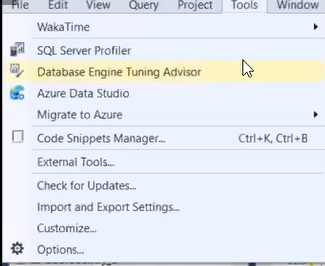
DMV is a collection of views in the masters database, and will tell you all the current status of the system. Start with sys.XXX

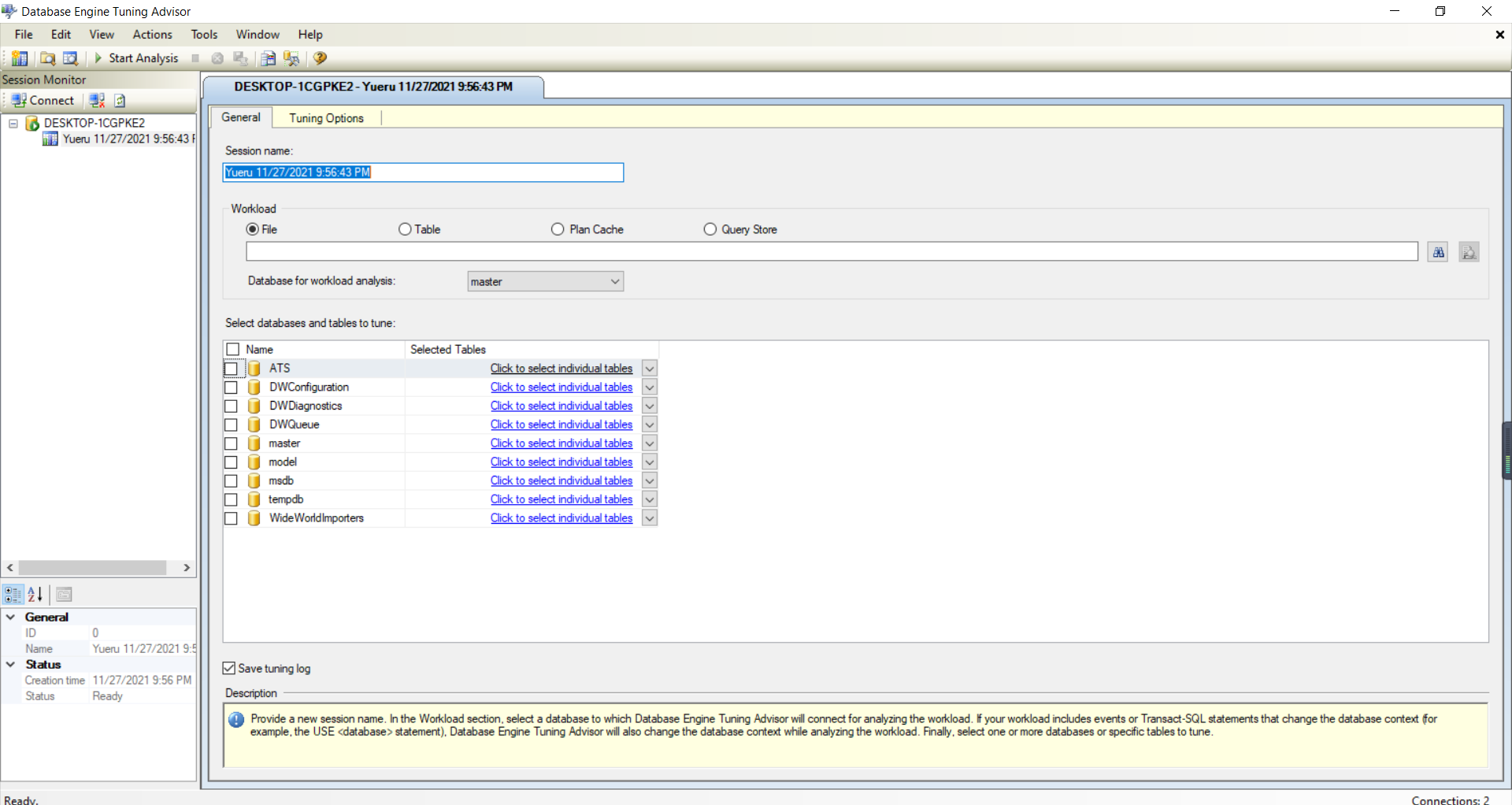


1. Resource Monitor (from Windows System)



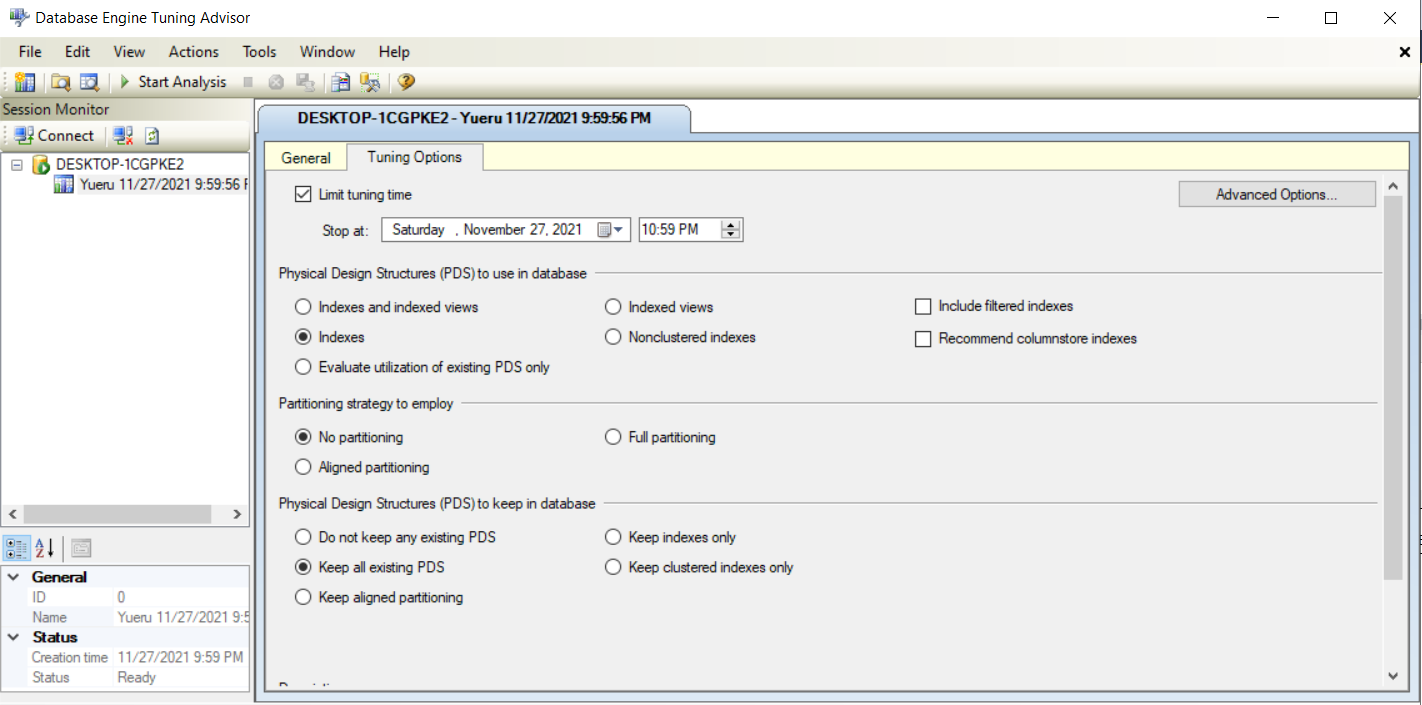
1. Tuning Advisor



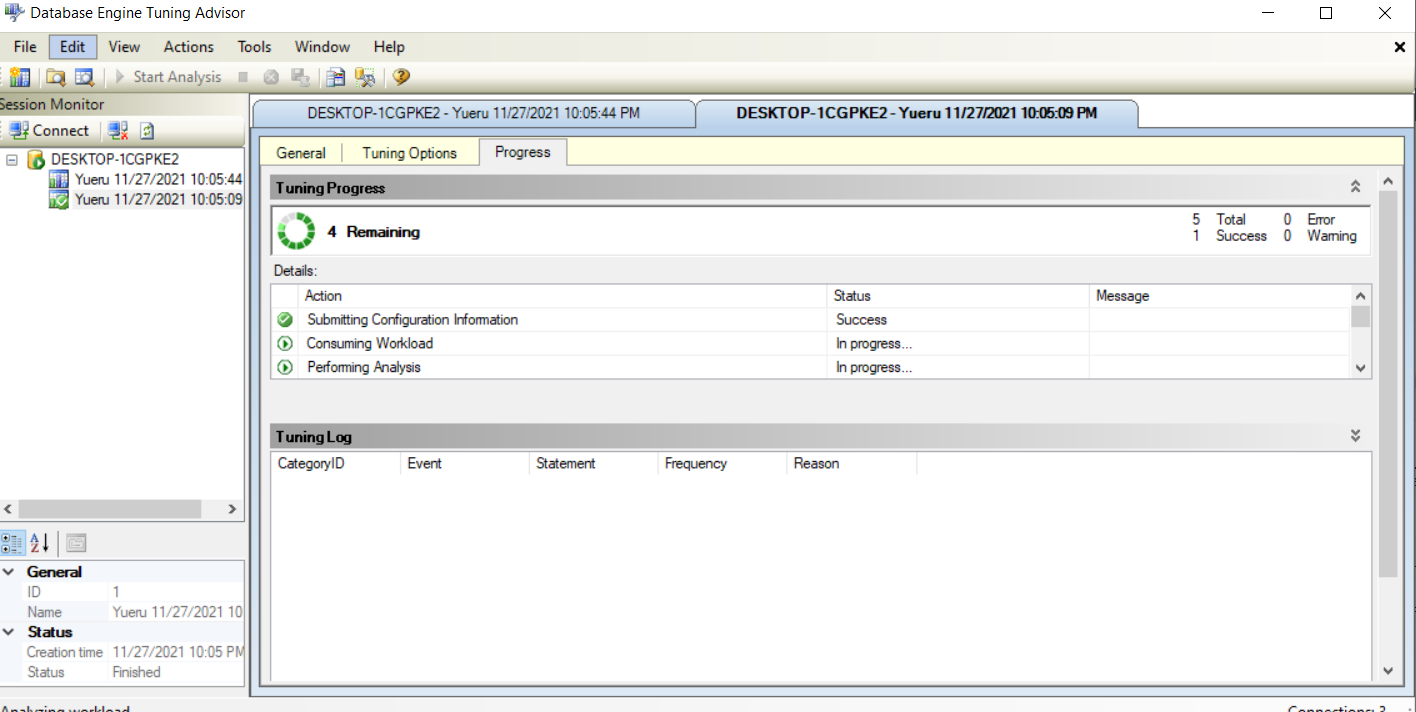


The “Plan Cache” is all the execution plans from stored procedure in the db. “Query Store” stores several queries ran in the past.

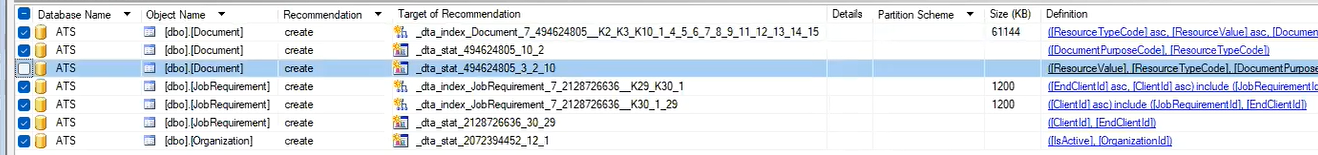
Go to tuning options, and you can specify what kind of index/views you would like to use. Usually will choose “Keep all existing PDS” for the last tab.



Click “start analyze”, then will get a report:



The report will give partition and index recommendations, so one can automatically implement the recommendation via the “action” tab in the tuning advisor.



1. Execution Plan

If click this icon before running multiple queries, there will be a window called execution plan, which gives information about the workflow of each query and how much resources that query cost in terms of percentage out of all queries.

* Can be used to compare two queries that do the same thing. The query with lower percentage will have higher efficiency, thus is more desired.
* Can be used to see if an index is missing. If in the node of each execution plan, there’s something called “Table Scan”, it means there’s an index missing.

