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

**Extended Events** is a helpful and convenient solution designed to monitor your system performance. By using extended events, you can see details about the inner operations of the sql system and your application. There are four scenarios to set, general: to name your session; events is the place you choose the event you want to monitor, the following image telling that I am going to use the sql\_text package to test the select and having clause in sql

Graphical user interface, text, application

Description automatically generated

Then the storage part is the place you set up the storage information. The advance button can constraint the result you want; for example in the following gives you the running time maximum of 30 seconds. I used a system sample from the Microsoft website.

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

It provides multiple information about the query.

**Tuning advisor** is another way to analyze databases and give some recommendations; for example, I will use the problem as an example because I created 4 ctes in that table, and I believe that it can be improved a lot.

I clicked the tools – Database Engine Tuning Advisor. Select the configuration in the following.

Graphical user interface, text, application, email

Description automatically generated

Then go back to my query and right-click using analyze query in DTA and press the start button; the recommendations are in the following.Graphical user interface, application

Description automatically generated

The analyzer gave me an estimated 70% improvement over the performance of my queries.

There are multiple reports that I can choose from, including cost, indexes information in the following.

Graphical user interface, text, application, email

Description automatically generated

Then I clicked on the definition of the table; I got a code that can create an index that could improve my time.

Graphical user interface, text, application

Description automatically generated

After I copied, pasted, and ran the code recommendation, my result is in the following, this time, the performance has been improved.

Graphical user interface, application, Word

Description automatically generated

**Dynamic management views** can be referenced in Transact-SQL statements by using two-part, three-part, or four-part names. On the other hand, dynamic management functions can be referenced in Transact-SQL statements by using either two-part or three-part names. Dynamic management views and functions cannot be referenced in Transact-SQL statements by using one-part names.

All dynamic management views and functions exist in the sys schema and follow this naming convention dm\_\*. When you use a dynamic management view or function, you must prefix the name of the view or function by using the sys schema. For example, to query the dm\_os\_wait\_stats dynamic management view, run the following query:

Background pattern

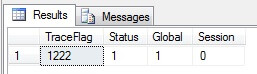
Description automatically generated with medium confidence

Graphical user interface, application, Word

Description automatically generated

Here is the result that I use a constraint saying my wait time is greater than 0.

**SQL Server log**: Database Console Commands, aka DBCC, is one of the tools for tuning queries. It can turn on a flag for a certain event. One application uses the DBCC Traceon(1222,-1) command after the query has started to find the deadlock. You can check the status of the trace flag using the DBCC TRACESTATUS (1222, -1) command. You can see by the following results that the trace flag is enabled and that it is enabled globally. You can turn off the trace flag at any time by simply issuing the DBCC TRACEOFF (1222,-1) command.



Now that we know the trace flag is enabled, all we have to do is wait for a deadlock to get logged and analyze the output.

**Execution Plan:** The Actual Execution Plan is generated after the query has been executed. It shows the actual operations and steps involved while executing the query. An execution plan in SQL Server is a simple graphical representation of the operations that the query optimizer generates to calculate the most efficient way to return a set of results.

First select turn the button on.

**A screenshot of a game

Description automatically generated with medium confidence**

Then run the query, and there will be a column named execution plan near the results

Diagram, schematic

Description automatically generated

This is the example from problem 14. First thing, there’s no table scan which is good; this is the part that anyone should avoid. Having multiple index scans is good, but most have high costs; thanks to the Database Engine Tuning Advisor in previous steps, the result is pretty fine. Check the number that costs more and find the possible way to lower the cost percentage.