



BRENT OZAR
UNLIMITED®

Using AI for Query Tuning

Module 8 Slide 1

This info
expires.



The future you expect

AI directly available inside SSMS, Azure Data Studio.

Right-click on a query plan and get advice.

The AI's advice is based on your actual:

- Tables, and their indexes
- Views
- Code inside stored procs, functions, triggers

The AI's advice is good, because it's trained on your database platform best practices and insight.

Module 8 Slide 3



That's not what we have today.

I'll explain why.

Module 8 Slide 4



AI stacks are complex and changing.

Module 8 Slide 5



Web app stack components

Component	Choice
Database	Microsoft SQL Server
Queries	Stored procedures
App connectivity layer	Entity Framework
App language	C#
Front end dev tool	Visual Studio

Module 8 Slide 6



Another example

Component	Choice
Database	Microsoft SQL Server
Queries	Done in app
App connectivity layer	Entity Framework
App language	C#
Front end dev tool	Visual Studio

Module 8 Slide 7



The slightest
stack change
changes how you work
and what you learn.

Module 8 Slide 8



AI stack components

Component	Choice
AI vendor	OpenAI (ChatGPT)
Large language model	gpt-4o 2024-08-06
Context	(things you automatically pass in as part of your prompt)
Prompt	(what you ask for)
AI front end dev tool	Microsoft Copilot in Azure Portal

Module 8 Slide 9



AI vendors

OpenAI (ChatGPT)

Meta

Google

Anthropic

xAI/Twitter

Cohere

(Note who's not in this list: Microsoft.)

Module 8 Slide 10



Each vendor has their own LLMs

OpenAI: gpt-4, gpt-4o, gpt-4o-mini, o1-preview...

Meta: Llama 3 70B, Llama 3 8B, Llama 3.1 405B...

Google: Gemini 1.5 Pro, 1.5 Flash, 1.5 Flash-8B...

Anthropic: Claude 3.5 Sonnet, 3.0 Opus, 3.0 Haiku...

xAI: Grok 2 Beta, Grok 1.5 Vision, Grok 1...

Cohere: Command R+, Command R, Command...

Module 8 Slide 11



Context

Background information about your particular needs

For query tuning purposes, that means your:

- Tables, and their indexes
- Views
- Code inside stored procs, functions, triggers

The larger your database schema is,
and the more objects it contains,
the more text is in your context.

Module 8 Slide 12



Prompt: what you're asking for

“I’m attaching a query plan file. Please explain what I should do to improve performance.”

The larger your query plan is,
the more text is in your prompt.

Plus, it’s up to *you* to include *all necessary context*, which means the underlying tables, indexes, called procs/views/functions, database settings...

Module 8 Slide 13



AI front end dev tool

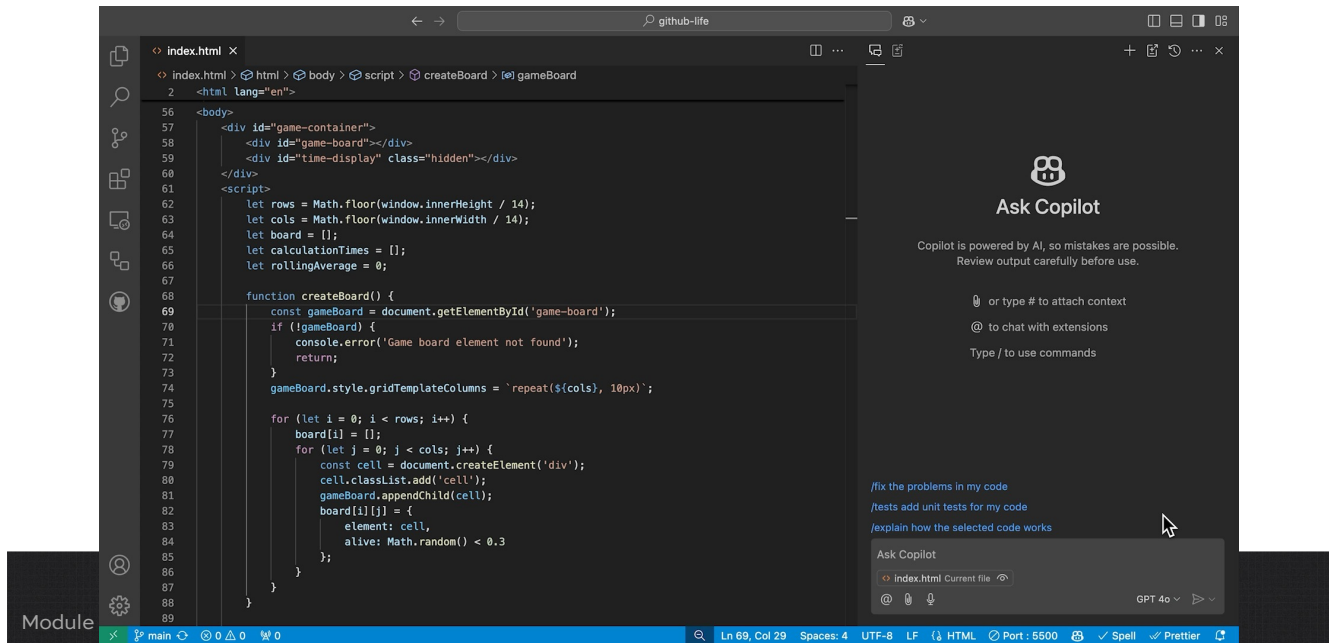
You want one app that puts it all together:

- Lets you pick the vendor (OpenAI, Meta, Google)
- Lets you pick an LLM model from that vendor
- Connects to your database and exports all relevant context, like table definitions, indexes, procs, etc
- Sends it up to the LLM along with your prompt
- Renders the results to you

Module 8 Slide 14

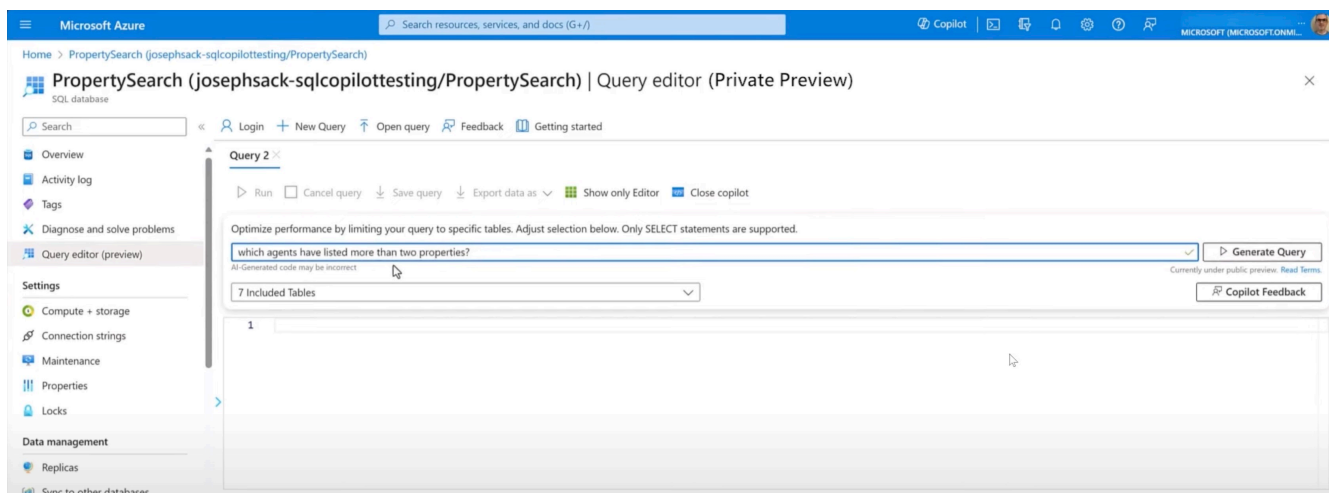


Github Copilot in VS Code (preview)



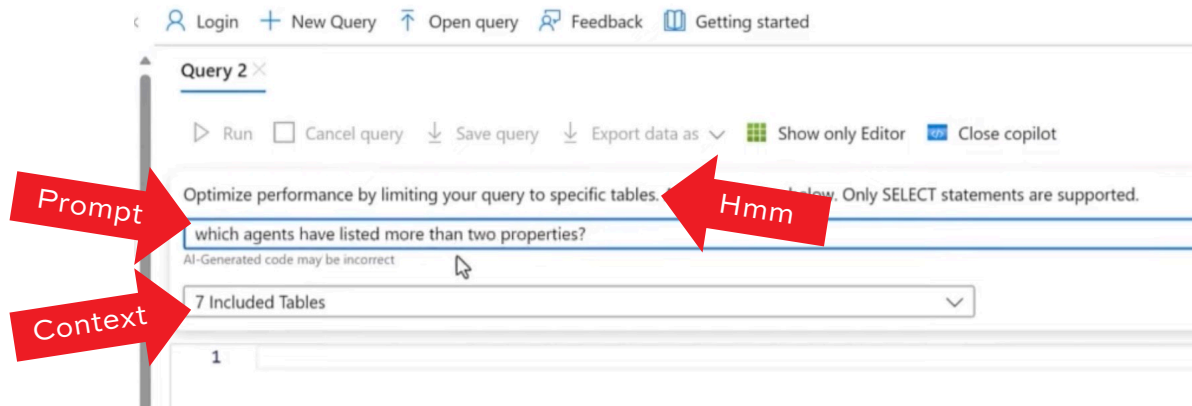
Azure Portal (preview)

https://www.youtube.com/watch?v=B_96_R5oNs4



You see some parts

osephsack-sqlcopilottesting/PropertySearch) | Query editor (Private Preview)



Module 8 Slide 17



The context problem

AI apps don't exist yet to connect to your database and export all of the relevant context

Even if they did, the context size would be huge. Think several megabytes worth of text.

If you're using a hosted AI solution like ChatGPT, you pay per kilobyte of text in & out, and many of them have very limited context sizes.

The larger your context per request, the more you pay, and the slower your request comes back.

Module 8 Slide 18



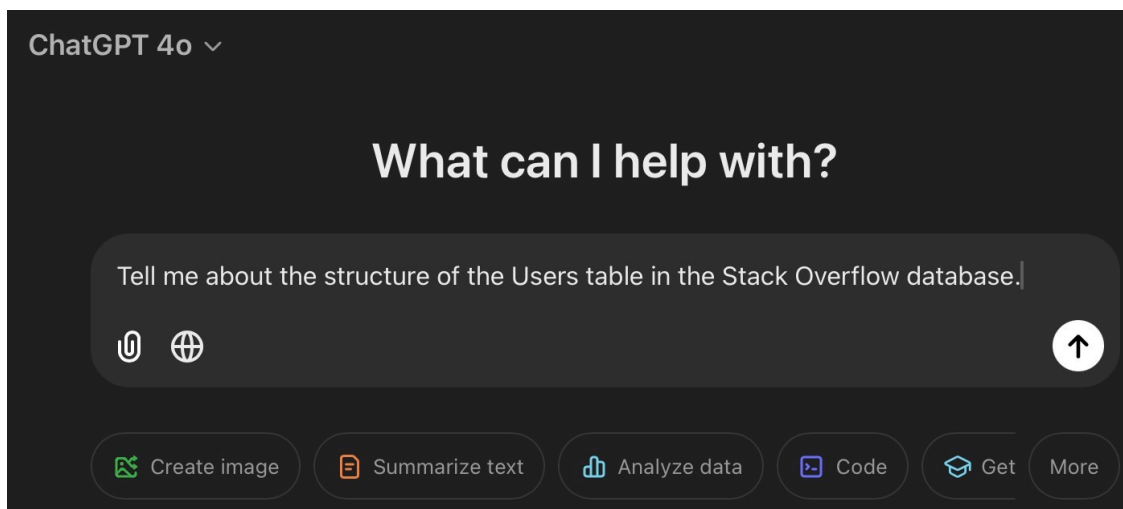
Problems at each layer

Component	Choice
AI vendor	Not a lot of cooperation here
Large language model	Changes extremely quickly, most aren't tuned for T-SQL
Context	Large database = large context, which also means \$\$\$ and slow
Prompt	Entirely up to you to write
AI front end dev tool	Just in preview, focusing on devs not DBAs for now

Module 8 Slide 19



Some demos make it seem easy



Module 8 Slide 20



ChatGPT 4o Share

Tell me about the structure of the Users table in the Stack Overflow database.

The **Users** table in the Stack Overflow database is central to managing information about the users who participate on the platform. This table captures essential details that uniquely identify users, track their contributions, and support various engagement metrics.

Here's an overview of the key columns in the **Users** table:

1. **Id (Primary Key)**
 - An integer that uniquely identifies each user.
2. **Reputation**
 - An integer indicating the user's reputation score, which reflects their engagement level and contributions to the community. Reputation is earned through upvotes on posts, accepted answers, and other interactions.
3. **CreationDate**
 - A datetime value that records when the user account was created.
4. **DisplayName**
 - A string representing the username or display name chosen by the user.
5. **LastAccessDate**
 - A datetime value noting the last time the user accessed the platform.
6. **WebsiteUrl**

Module 8 Slide 2


“Wow! That’s amazing!”

No, it’s not.

Large language models (LLMs) were trained on publicly visible information, like blogs & demos.

The instant you start asking for non-public info...

Module 8 Slide 22



What indexes exist on the StackOverflow.Users table?



The **Users** table in the Stack Overflow database typically has several indexes to optimize various types of queries related to user data. Here are some of the key indexes you might expect to find on this table:

1. **Primary Key Index on Id**

- This is a clustered index on the **Id** column, providing fast access to user records by their unique identifier. It is typically the primary key, making lookups by user ID efficient.

2. **Index on Reputation**

- An index on the **Reputation** column can be useful for queries that retrieve high-reputation users, such as leaderboards or sorted lists of top users. This is often a non-clustered index that helps with ordering users by reputation.

3. **Index on CreationDate**

- This index is helpful for queries involving user registrations over time, such as reports on new users in a given period or for fetching recently joined users.

4. **Index on LastAccessDate**

- An index on **LastAccessDate** can help with queries that need to retrieve users based on recent activity. For instance, if the system needs to find active users or sort users by their last access date, this index improves performance.

5. **Index on DisplayName**

Module 8 Slide 23



It's hallucinating.

Create a widescreen 1280x720 image showing a high androgynous person hallucinating.



Here is the image of a high androgynous person experiencing a vivid hallucination. Let me know if there's anything specific you'd like to adjust!

Module 8 Slide 25



Create a widescreen 1280x720 image showing a high androgynous person hallucinating.



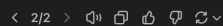
Here is the image of a high androgynous person experiencing a vivid hallucination. Let me know if there's anything specific you'd like to adjust!

Can you make it widescreen 1280x720 pixels?



I've created this image at 1280x720 pixels as requested, ensuring it's in widescreen format. Let me know if there's anything else you'd like to adjust!

Module 8 Slide 26



Hallucinations & overconfidence

This is a real, extensive problem.

AI companies have no idea how to fix it right now.

When you get anything out of an LLM, it's up to you to fact-check all of it.

I routinely use ChatGPT in my live Postgres streams, just to show the quality, and it's usually *terrible*.

Module 8 Slide 27



Back to the future you expect

AI directly available inside SSMS, Azure Data Studio.

Right-click on a query plan and get advice.

The AI's advice is based on your actual:

- Tables, and their indexes
- Views
- Code inside stored procs, functions, triggers

The AI's advice is good, because it's trained on your database platform best practices and insight.

Module 8 Slide 28



We won't have that in 2025.

Apps like SSMS and Azure Data Studio can't evolve quickly enough to keep up with AI industry changes.

LLMs aren't trained on your database schema. (They can be – but that requires LLM expertise, and in your company, that's too hard to get, and that expertise is focused on other business areas.)

RAGs will aim to fix that, but easy RAG implementations that involve your database are still year(s) away.

Hallucinations are too big of a problem today.

Module 8 Slide 29



Where today's AI rocks for SQL

“This stored procedure uses nested cursors. Can you refactor it into a set-based operation?”

“This stored procedure doesn't have any error handling. Can you add a try/catch block and retry deadlocks automatically?”

“This query produces this error. Can you tell me why, and rewrite it to avoid the error?”

Today's AI is awesome at manual labor.

Module 8 Slide 30



You'll hear a lot about Copilot.

“It’s going to change the way you work!”

“It’s going to take your job!”

“If you don’t learn AI now, you’re doomed!”

And if you have spare time, and you want to learn how to do your job more quickly, you should learn how to use today’s AI tools. (I use ‘em every single day!)

Module 8 Slide 31



But if you don’t have spare time...

That’s okay.

AI tools are changing so *quickly*.

AI info you learn today will expire very quickly.

There are no “AI for Database Tuners” resources.

There aren’t even good “AI for C# Developers” resources, and that market is way bigger.

The market needs to mature before learning gets easier.

Module 8 Slide 32



If you do want to learn:

<https://www.reddit.com/r/LocalLLM/>

<https://www.reddit.com/r/LocalLLaMA/>

<https://www.reddit.com/r/ChatGPT/>

<https://www.reddit.com/r/StableDiffusion/>

<https://news.ycombinator.com/>

Don't buy expensive video cards for LLMS. Use services like Runpod.io and Vast.ai. They light VC money on fire.

