

SPECIAL ANNIVERSARY EDITION  
LET'S MEET FOR THE 10<sup>TH</sup> TIME!



EVENTION  
CZAS ZAANGAZOWANY



APRIL 10<sup>TH</sup>-11<sup>TH</sup>  
**2024**

# Your personal LLM and RAG-backed Data Copilot – lessons learned

Marek Wiewiórka, Phd

GetInData | Part of Xebia



Marek Wiewiórka  
PhD | Chief Data Architect at  
GetInData | Cloud and Big Data A...



- Chief Data Architect @**GetInData | Part of Xebia**
- PhD(2023), Research Assistant at the Warsaw University of Technology
- Personally, a keen long-distance runner and gravel bike enthusiast

# ...how to turn best practices into AI coding assistant

1. Why do we need yet another (open-source ) Copilot?
2. How can we build one?
3. Architecture and evaluation
4. DEMO



# (Data) Context is king!

- Explicit and precise data context of your whole data platform
- Data transformation codebase
- Data models with comments and table relationships
- Other user queries
- Lineage and human curated dataset descriptions from data catalogs



# Customized and specialized models are the future.



We believe that in the future, the vast majority of organizations will develop customized models that are personalized to their industry, business, or use case.

With a variety of techniques available to build a custom model, organizations of all sizes can develop personalized models to realize more meaningful, specific impact from their AI implementations. The key is to clearly scope the use case, design and implement evaluation systems, choose the right techniques, and be prepared to iterate over time for the model to reach optimal performance.

With OpenAI, most organizations can see meaningful results quickly with the self-serve fine-tuning API. For any organizations that need to more deeply fine-tune their models or imbue new, domain-specific knowledge into the model, our Custom Model programs can help.



## Build high-quality generative AI applications with DBRX customized for your unique data

by [Jonathan Frankle](#), [Ali Ghodsi](#), [Naveen Rao](#), [Hanlin Tang](#), [Abhinav Venigalla](#) and [Matei Zaharia](#)

March 27, 2024 in [Company Blog](#)

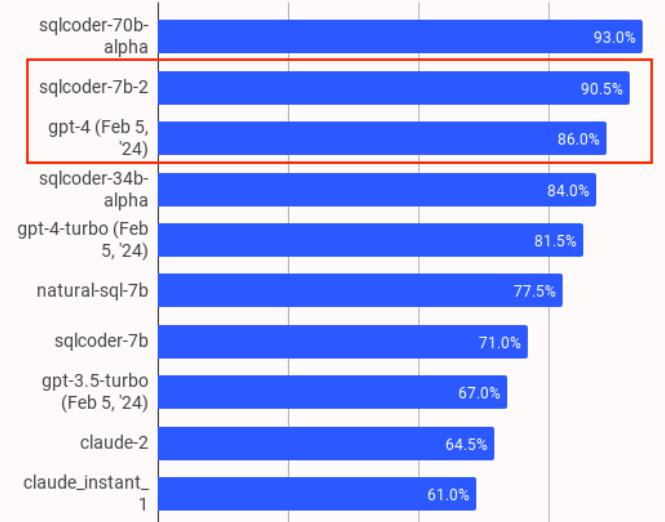
Share this post



Databricks' mission is to deliver data intelligence to every enterprise by allowing organizations to understand and use their unique data to build their own AI systems. Today, we are excited to advance our mission by open sourcing DBRX, a general purpose large language model (LLM) built by our [Mosaic Research](#) team that outperforms all established open source models on standard benchmarks. We believe that pushing the boundary of open source models enables generative AI for all enterprises that is customizable and transparent.

- Many other small (7-34b) models licensed for commercial use, e.g.:
  - ✓ starcoder2
  - ✓ dolphincoder
  - ✓ deepseek-coder
  - ✓ opencodeinterpreter

Percentage of correctly generated SQL queries on novel schemas not seen in training (n = 200), with 4 beams



When quantized can be even run locally!

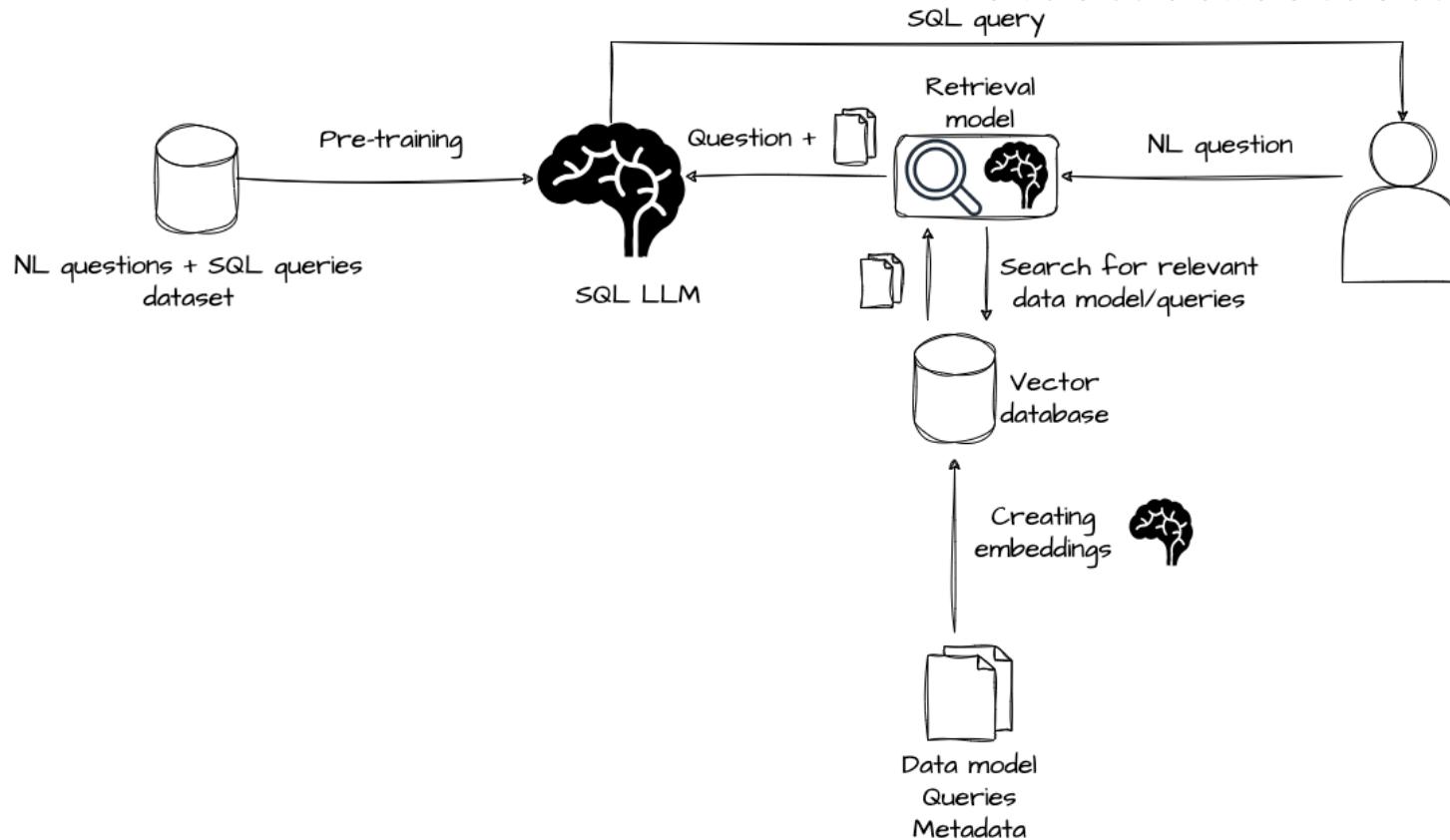
# How turn your best practices into Copilots ?

- Vector database as a knowledge base - what ?
- Prompts as instructions following best practices - how ?
- LLM to combine both...

## Retrieval-Augmented Generation(RAG)

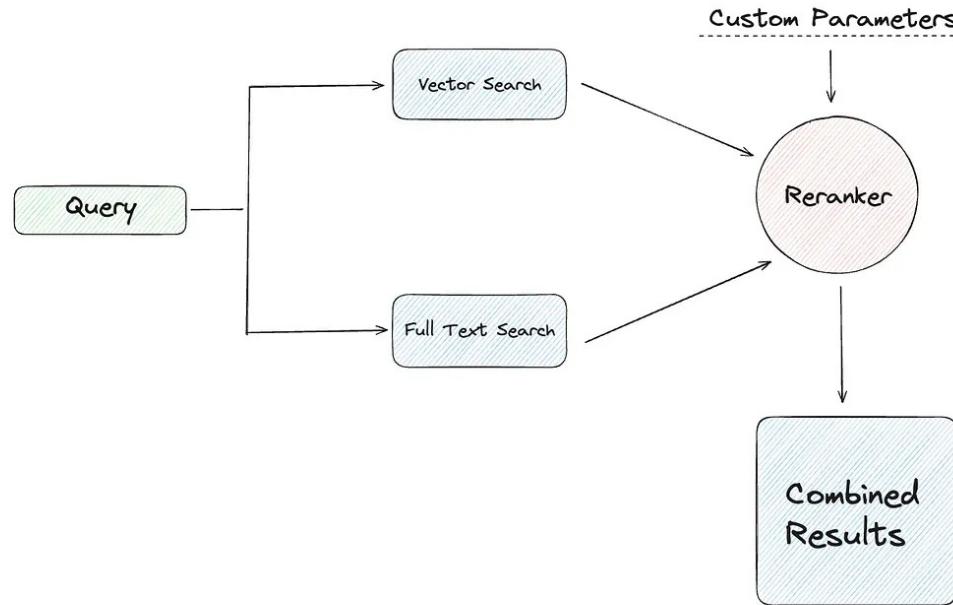


# RAG for Text-to-SQL



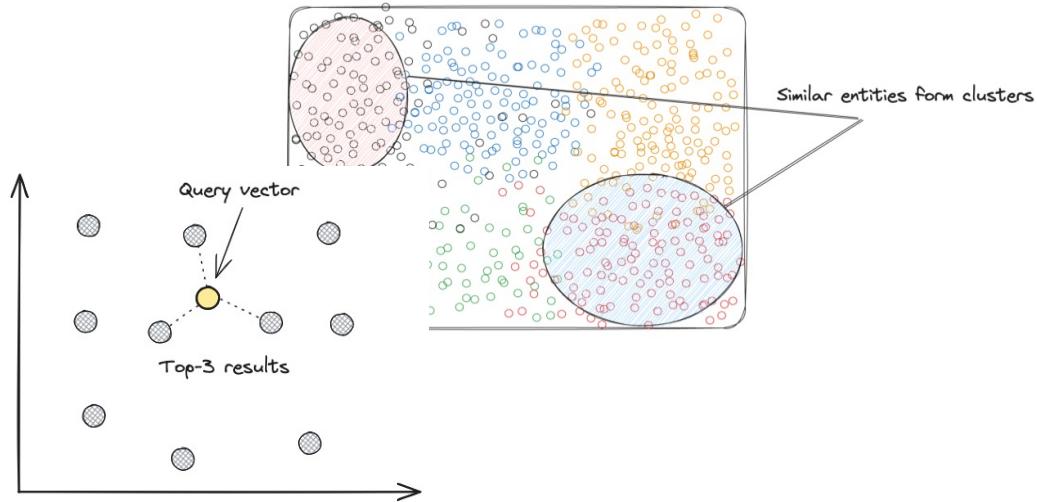
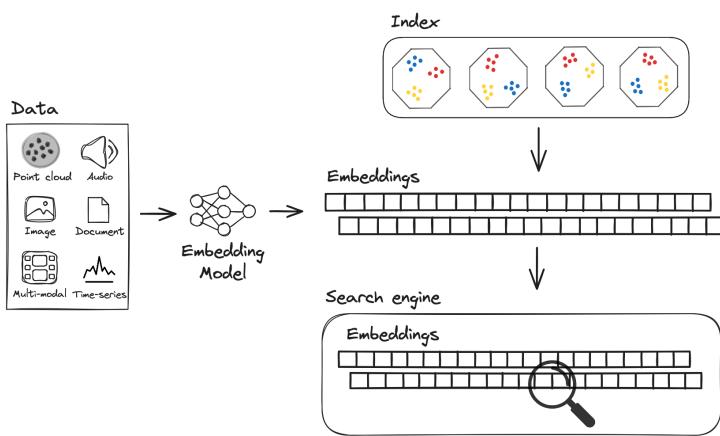
# Hybrid search

- combination of keyword and vector search

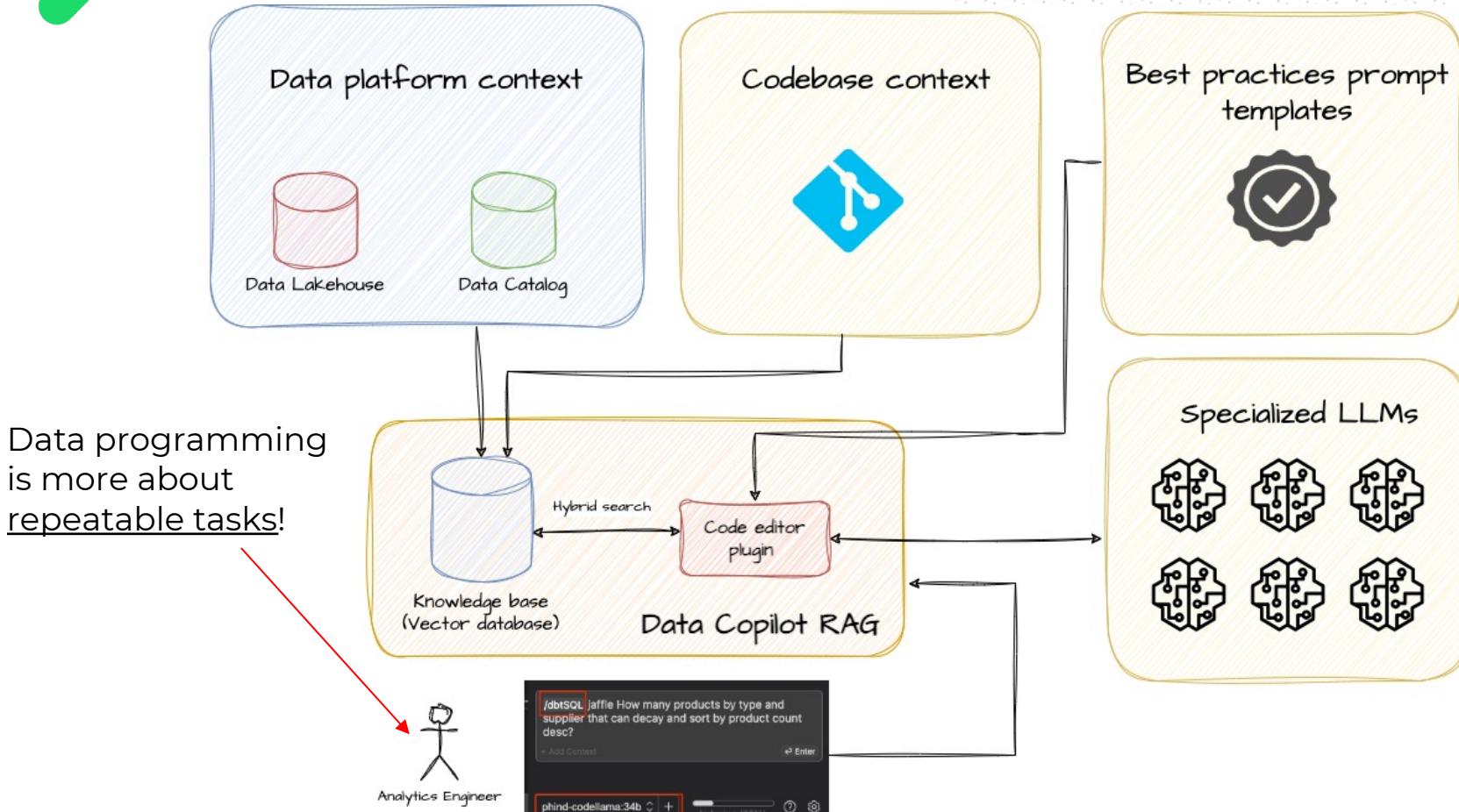


# Vector search

- a technique used to search for similar items based on their vector representations, called embeddings
- Approximate Nearest Neighbours algorithms

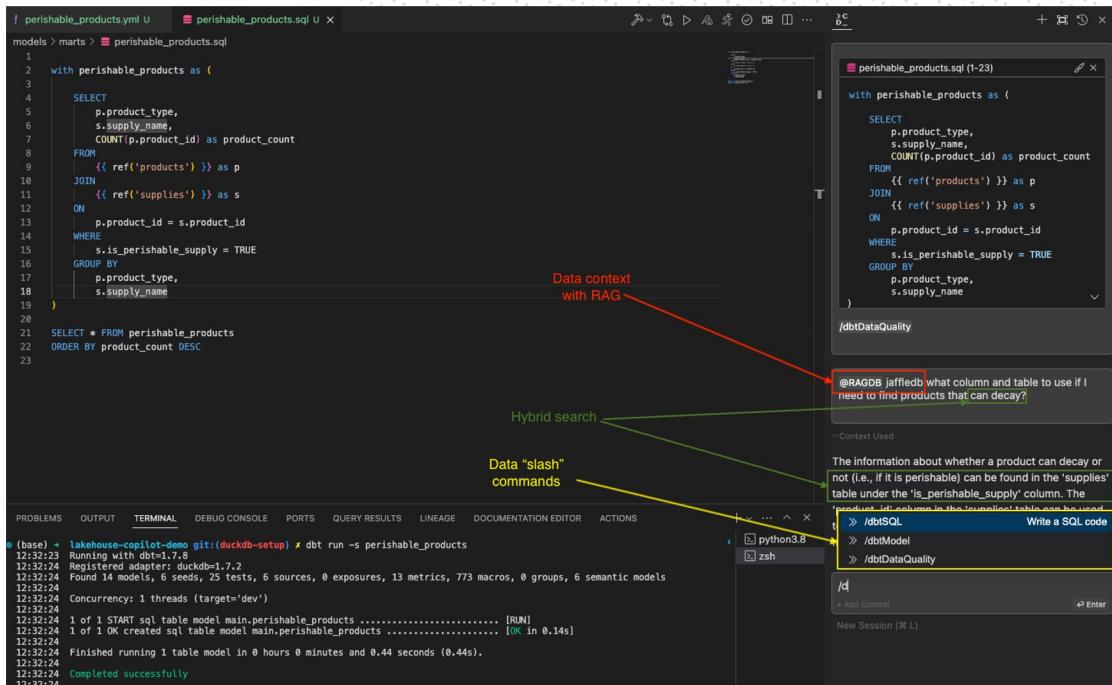


# Data Copilot RAG architecture



# GID Data Copilot (GDC)

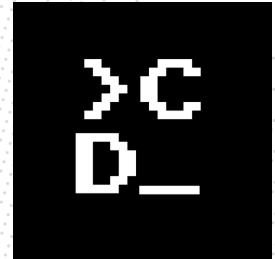
- An extensible AI **programming assistant for SQL and dbt code**
- Powered by:
  - **Large Language Models** (SOTA LLMs)
  - **Robust Retrieval Augmented Generation** (RAG) architecture
  - **Hybrid search** techniques
  - **Fast Vector Database**
  - **Curated Prompts**
  - **Builtin Data commands**



The screenshot illustrates the GID Data Copilot interface, which integrates multiple tools for data engineering tasks:

- Code Editor:** Displays two files: `perishable_products.yml` and `perishable_products.sql`. The `sql` file contains a query to calculate product counts based on supply type and perishability.
- Terminal:** Shows the output of running the `dbt run -s perishable_products` command, indicating the creation of a table model named `main.perishable_products`.
- Search Bar:** A prominent search bar at the top right is labeled "Data context with RAG". It shows a query: "@RAGDB laffedb what column and table to use if I need to find products that can decay?". Below the search bar, a tooltip explains: "The information about whether a product can decay or not (i.e., if it is perishable) can be found in the 'supplies' table under the 'is\_perishable\_supply' column. The 'product\_id' column in the 'supplies' table can be used".
- Autocomplete:** A dropdown menu shows suggestions for "slash commands": `>> /dbtSQL`, `>> /dbtModel`, and `>> /dbDataQuality`.

# Continue - an open-source autopilot



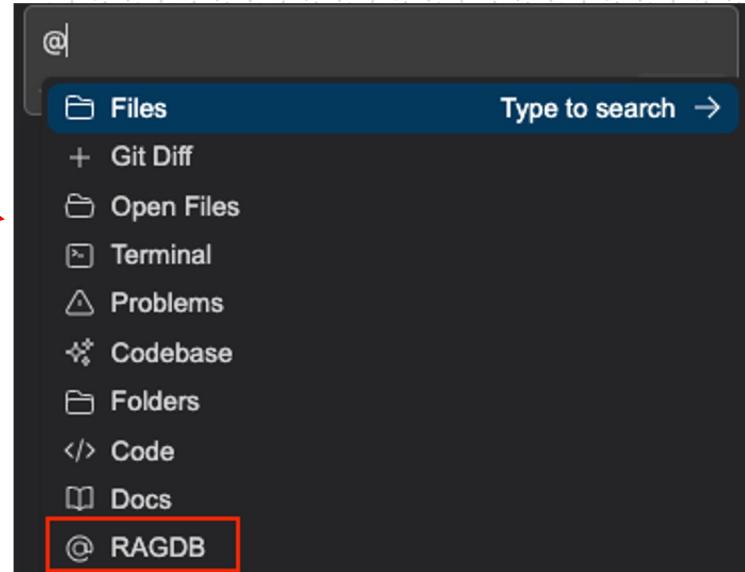
- support for both tasks and tab autocomplete
- highly extensible
  - use any LLM model you wish - also ***multiple, specialized models*** for different languages or tasks
  - support for many ***model providers***, such as Ollama, vLLM, LM Studio
  - custom ***context providers*** for more control over LLMs augmentation
  - custom ***slash commands*** that can combine own *prompts*, *contexts* and *models* for specialized, reusable tasks
- support for VSCode and Jetbrains
- secure (i.e. can be run locally, on-premise or cloud VPC)
- translate *your best practices* into "slash data commands"

# Continue - a custom context provider

```

const RagContextProvider: CustomContextProvider = {
  title: "ragdb",
  displayTitle: "RAGDB",
  description:
    "Retrieve DB schema from our vector database of internal documents",
  type: "normal",
  getContextItems: async (query: string, extras: ContextProviderExtras) => {
    console.info(extras.fullInput)
    const inputArray = extras.fullInput.split(' ');
    const db = inputArray[0];
    const userQuestion = inputArray.slice(2).join(' ');
    const response = await fetch("http://localhost:8000/retrieve", {
      method: "POST",
      headers: {
        'content-type': 'application/json;charset=UTF-8',
      },
      body: JSON.stringify({ query: userQuestion }),
    });
    const results = await response.json();
    return results.map(result: { title: any; contents: any } => ({
      name: result.title,
      description: result.title,
      content: result.contents,
    }));
  },
};

```



# dbtSQL task = custom(context + prompt + model)

```
export function modifyConfig(config: Config): Config {
```

```
  config.slashCommands7.push({
```

```
    name: "dbtSQL",
    description: "Write a SQL code",
    run: async function*( sdk) {
```

```
      const inputArray = sdk.input.split(' '),
      const db = inputArray[0];
      const userQuestion = inputArray.slice(2).join(' '),
      const response = await fetch("http://localhost:8000/retrieve", {
```

```
        method: "POST",
        headers: {
          'content-type': 'application/json; charset=UTF-8',
        },
        body: JSON.stringify({ query: userQuestion }),
      });
    
```

```
    const results = await response.json();
```

```
    const ragResponse = results.map((result: { title: any; contents: any; }) => ({
      name: result.title,
      description: result.title,
      content: result.contents,
    }));
  
```

```
  const PROMPT = `
```

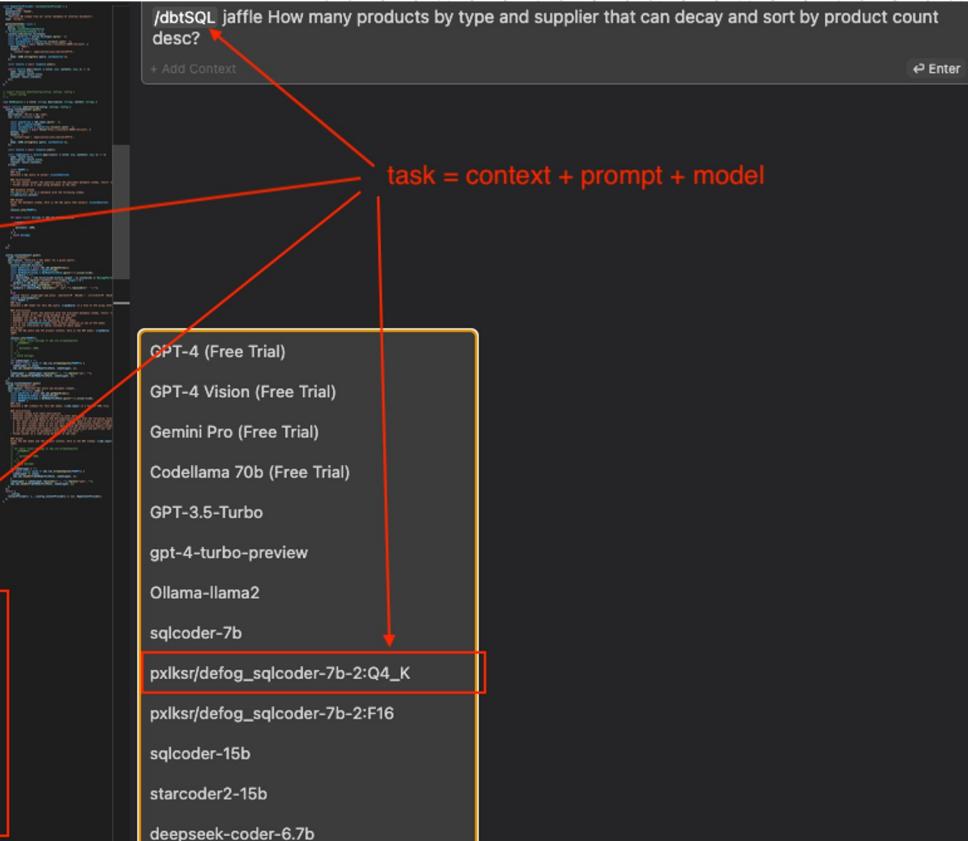
```
### Task
Generate a SQL query to answer: ${userQuestion}
```

```
### Instructions
```

- If you cannot answer the question with the available database schema, return 'I'
- Format answer as a code using markdown in the chat.

```
### Database Schema
```

The query will run on a database with the following schema:  
`$(ragResponse.content)`



- fast and easy self-hosting of LLMs almost everywhere
- hybrid CPU+GPU inference
- powered by [llama.cpp](#)
- rich library of existing LLMs in different flavours
- GGUF - fast and memory efficient quantization for GPU
- Serve model with one-liner:  

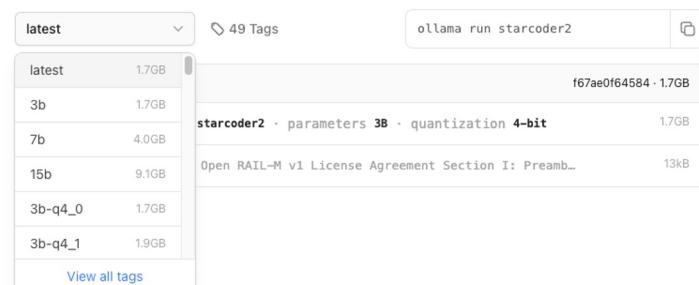
```
ollama run starcoder2:7b
```
- vLLM for production deployments

([Our video tutorial](#))

#### starcoder2

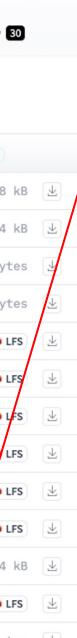
StarCoder2 is the next generation of transparently trained open code LLMs that comes in three sizes: 3B, 7B and 15B parameters.

 21.2K Pulls  Updated 4 weeks ago



The screenshot shows the Ollama interface with the 'starcoder2' model selected. On the left, there's a dropdown menu set to 'latest' and a list of available models: latest (1.7GB), 3b (1.7GB), 7b (4.0GB), 15b (9.1GB), 3b-q4\_0 (1.7GB), and 3b-q4\_1 (1.9GB). To the right, there's a search bar containing 'ollama run starcoder2'. Below the search bar, a card displays the 'starcoder2' model with 'parameters 3B', 'quantization 4-bit', and a file size of '1.7GB'. A link to 'Open RAIL-M v1 License Agreement Section I: Preamb...' is also visible. At the bottom, there's a button labeled 'View all tags'.

# Ollama - custom model in 4 steps



Hugging Face Search models, datasets, users...

defog/sqlcoder-7b-2 like 194

Text Generation Transformers Safetensors GGUF llama

Model card Files and versions Community 30

main sqCoder-7b-2

rishdotblog Update README.md 7e5b6f7 VERIFIED

.gitattributes 1.58 kB

README.md 3.54 kB

config.json 691 Bytes

generation\_config.json 111 Bytes

label\_mask.npy 458 kB LFS

labeled\_matches.npy 458 kB LFS

labels.npy 3.66 MB LFS

model-00001-of-00003.safetensors 4.94 GB LFS

model-00002-of-00003.safetensors 4.95 GB LFS

model-00003-of-00003.safetensors 3.59 GB LFS

model.safetensors.index.json 24 kB

predictions.npy 3.66 MB LFS

special\_tokens\_map.json 515 Bytes

sqlCoder-7b-q5\_k\_m.gguf 4.78 GB LFS

1. Download a model in the GGUF format
2. Create a Modelfile, e.g.:

```
FROM ./sqlcoder-7b-q5_k_m.gguf
TEMPLATE """{{ .Prompt }}"""
PARAMETER stop "<|endoftext|>"
```

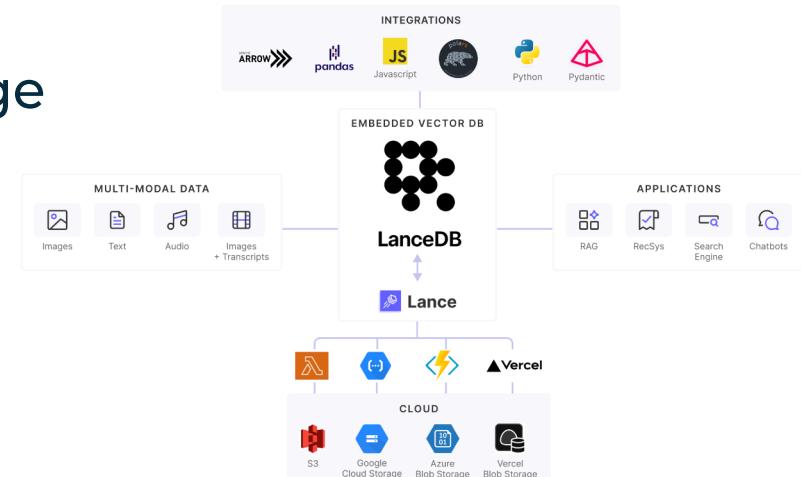
3. Create a model with Ollama

```
ollama create sqCoder-7b-2 -f Modefile
```

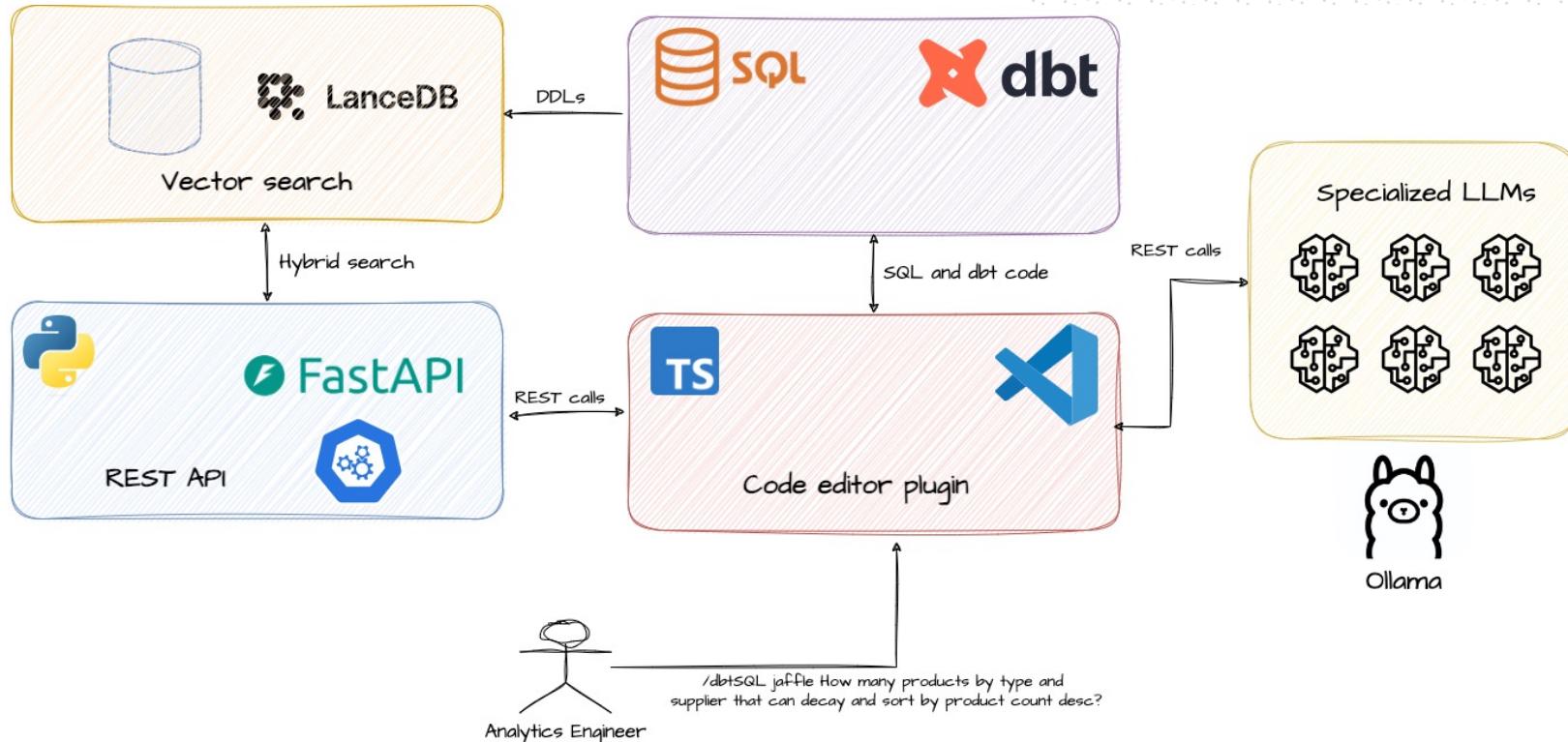
4. Serve it

```
ollama run sqCoder-7b-2
```

- fast (Rust ❤️), serverless and embeddable - DuckDB for ML
- powered by [Lance](#) file format for ML (versioning, zero-copy)
- multi-modal
- support for hybrid (semantic + keyword) search
- [Llamaindex](#) integration
- Python API and fast data exchange with [polars](#) and [Arrow](#)



# Technical architecture



# Question representation<sup>1</sup>

```

Table continents, columns = [ContId, Continent]
Table countries, columns = [CountryId, CountryName,
↳ Continent]
Q: How many continents are there?
A: SELECT
  
```

**Listing 1: Example of Basic Prompt**

```

1 ### Complete sqlite SQL query only and with no
↳ explanation
2 ### SQLite SQL tables, with their properties:
3 #
4 # continents(ContId, Continent)
5 # countries(CountryId, CountryName, Continent)
6 #
7 ### How many continents are there?
8 SELECT
  
```

**Listing 3: Example of OpenAI Demostration Prompt**

```

1 Given the following database schema:
2 continents: ContId, Continent
3 countries: CountryId, CountryName, Continent
4
5 Answer the following: How many continents are there?
6 SELECT
  
```

**Listing 2: Example of Text Representation Prompt**

```

1 /* Given the following database schema: */
2 CREATE TABLE continents(
3     ContId int primary key,
4     Continent text,
5     foreign key(ContId) references countries(Continent)
6 );
7
8 CREATE TABLE countries(
9     CountryId int primary key,
10    CountryName text,
11    Continent int,
12    foreign key(Continent) references continents(ContId)
13 );
14
15 /* Answer the following: How many continents are there?
16 ↳ */
17
18 SELECT
  
```

**Listing 4: Example of Code Representation Prompt**

<sup>1</sup>Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation

- Not meant to be yet another benchmark, such as: [Spider](#) or [Bird-SQL](#)
- [Jaffle Shop](#) example - simple but not trivial
- 4 typical data tasks
  - Data model exploration/discovery
  - SQL: an easy one (single table) and more complex (joins with sorting and aggregations)
  - dbt model generation
  - dbt tests generation based on rules

# LLMs evaluation 2/2

Model	Licence	size [b]	Data discovery	SQL - simple	SQL - complex	dbt - model	dbt - tests
deepseek-coder	deepseek	33	+	+	+/-	+/-	-
deepseek-coder	deepseek	6.7	+/-	+/-	+/-	-/+	-
codellama	Llama2	70	+	+	-/+	-	-
starcoder2	bigcode-openrail-m	15	-	-	-	-	-
sqlcoder	CC BY-SA 4.0	7	-	+/-	+/-	N/A	N/A
<b>phind-codellama</b>	Llama2	34	+	+	+	+	+/-
wizardcoder	Llama2	33	+	+	+/-	+/-	-/+
gpt-3.5-turbo	Commercial	N/A	+	+	+	+	+/-
<b>gpt-4</b>	Commercial	N/A	+	+	+	+	+
gpt-4-turbo-preview	Commercial	N/A	+	+	+	+	+/-
Gemini Pro	Commercial	N/A	+	+/-	+	+/-	-
OpenCodeInterpreter	Apache-2.0	33	+	+	-/+	-	-

**+-** perfect or almost perfect

**+/-** - not optimal or some minor tweaks needed

**-/+** - not very helpful, serious hallucinations

**-** - totally useless

# A handful of conclusions...with a grain of salt

- NL-to-SQL and dbt code generation are **challenging**
- **commercial** models are in most cases still better... but
- there are very promising **open-source ~30b alternatives**
- **SQL-dedicated** and fine-tuned models can turn out a bit a disappointing (beam search?), e.g.:
  - unnecessary joins elimination
  - wrong data types inference
  - occasional hallucinations

- Implementation of in-context learning such as Query Similarity Selection (few-shot strategy)
- Model(s) fine-tuning using dbt examples, especially for data quality aspects
- Fine-tuning focused on platform-to-platform migrations



# Welcome to the GID data copilot

## DEMO

# AI & LLM Ops free consultation



# Thank you !