Talk to your database: An open-source in-context learning approach to interact with relational databases through LLMs

Maximilian Plazotta[1,*], Meike Klettke[1]

[1]University of Regensburg, Bajuwarenstraße 4, 93053 Regensburg, Germany

Abstract

With the emergence of large language models, the long studied field of the Text-to-SQL problem was elevated into new spheres. In this paper, we test how our LLM fine-tuning approach performs on two relational databases (small vs. big) and compare it to a default setting. The results are convincing: using in-context learning boosts the performance from a merely 35% (default) to over 85%. Furthermore, we present a detailed architectural framework for such a system, emphasizing its exclusive reliance on open-source components.

Keywords

Text-to-SQL, Large Language Models, Relational Databases

1. Introduction

The underlying idea for this paper is the following (real-world) case: Imagine a business analyst who needs certain information to answer business questions such as ”Which customers should we contact in our next marketing campaign?” or ”Which customers qualify for a discount?” based on data. Furthermore, the business analyst only has very rudimentary knowledge in SQL programming. So, is there a way to create a system that can help this person to gain insights from the data and answer their business questions? With the novel introduction of large language models (LLMs) and a PostgreSQL database, we create a system that builds a bridge between the LLM and the database — with the usage of in-context learning. Moreover, we test the system based on different database sizes and the application of in-context learning vs. default LLM prompting to test the following hypotheses:

- **H1**: In-context learning should perform better than the default.
- **H2**: In-context learning should possess a higher execution time due to having more complex inputs.
- **H3**: The more complex a database is, the lower the accuracy should be.

Furthermore, we are not only testing if the hypotheses are fulfilled for a certain setting, but also determine, up to what degree (changes of accuracy and execution time) they are fulfilled. The rest of the paper is organized as follows: in Section 2 a short overview and explanation of relevant terminology is given. Section 3 is devoted to related work in this field. The main section, Section 4, describes the approach of the experiment, the technical set-up of the system, and gives an overview of the results — a discussion of the results is also included. Section 5 tackles the limitations and gives some ideas how to eventually circumvent them. The last Section 6 concludes the paper and highlights future areas of research.

Large Language Models for Research Data Management!? 2025 (LLMs4RDM 2025), Workshop at the INFORMATIK Festival 2025 (55th Annual Conference of the German Informatics Society), September 18, 2025, Potsdam, Germany

*Corresponding author.

Maximilian.Plazotta@informatik.uni-regensburg.de (M. Plazotta); meike.klettke@informatik.uni-regensburg.de (M. Klettke)

2. Terminology

Within this section, we provide an overview of the central terminology used in this paper: LLMs, fine-tuning, and the Text-to-SQL problem.

2.1. Large Language Models

With the introduction of the first commercially usable large language models in 2022, the whole world around artificial intelligence changed drastically — today the buzzword "AI" (short for artificial intelligence) is everywhere. With the release of GPT-3.5 by OpenAI in late 2022 the traffic and usage of their chatbot with the remarkable name ChatGPT exploded overnight. Since then, many new players have entered the market with their own LLMs for which some need to be paid for and some are open-sourced.

Table 1
Most commonly used LLMs[1]

Closed-source (proprietary) LLMs			
Model	Family	Provider	ELO[2]
Gemini-2.5-Pro	Gemini	Google	1467
o3-2025-04-16	GPT	OpenAI	1451
ChatGPT-4o-latest-20250326	GPT	OpenAI	1442
Claude-Opus-4-20250514	Claude	Anthropic	1418
Grok-3-preview-02-24	Grok	xAI	1409
Mistral-medium-2505	Mistral	Mistral	1381
Open-source LLMs			
Model	Family	Provider	ELO
Deepseek-R1-0528	DeepSeek	DeepSeek	1413
Qwen3-235b-a22b	Qwen	Alibaba	1369
Gemma-3-27b-it	Gemma	Google	1362
Llama-3.1-405b-instruct	Llama	Meta AI	1333
Llama-4-Maverick-17b-128e-instruct	Llama	Meta AI	1329
Qwen2.5-coder-32b-instruct	Qwen	Alibaba	1270
Phi-4	Phi	Microsoft	1257
Llama-3.1-8b-instruct	Llama	Meta AI	1213

[1] As of 30st of June 2025, data source: https://lmarena.ai/leaderboard/text.
[2] ELO based on Chatbot Arena LLM Leaderboard. [1]

Table 1 provides a broad overview of the current LLM market with respective models and their capability scoring called ELO or arena score. This number is derived from a multitude of different tests a LLM is encountered with, e.g., coding-, math-, creative writing-, reasoning-tasks. [1] The most capable models according to the ELO score are currently Google's Gemini model(s) and OpenAI's GPT model(s). Other competitor models like xAI's Grok (backed by Elon Musk), Anthropic (backed by Amazon), and Qwen (Alibaba) also score very highly. As mentioned for these models one needs to pay for an API key to get access — for small endeavors the price is manageable, but for enterprise usage the costs accumulate very quickly as you pay for each prompt. Thankfully, there exist open-source models that have good performance scores and can be used locally given that you possess enough RAM. Meta AI's Llama models are one of the most popular open-source models with different parameter sizes: 405B,

70B, or 8B — the ”B” stands for billion and specifies the model’s number of input parameters. Normally, the more parameters a model has, the better its performance. However, this comes with a downside: the more parameters a model possesses, the more computing power a system needs to deliver. So, for an 8B model, a good graphics processing unit (GPU), such as the Nvidia 30er series or higher, is sufficient, but for instance, a 200B+ models needs an Nvidia A100 or 4x 3090 GPUs (or more) to function. Nevertheless, small models (<14B models) score very well compared to their bigger brothers and sisters: Llama’s 8B model has an ELO of 1213 compared to the 70B (1315), or the 405B (1333) model. So, why is this important: To build such a system, one needs to define where to deploy the LLM. Small models can be accessed from your local PC, bigger models need a data center or cloud environment; or you just simply access it with an API key from a proprietary vendor like OpenAI, Google, or Anthropic. Most recently, DeepSeek-R1 disrupted the AI market with incredible performance (ELO: 1413) and being open-source, but on the other hand being relatively big (671B). More interestingly, the new smaller, open-source model from Google (Gemma) scores very high (ELO: 1362) despite only having 27 billion parameters.

2.2. Improving LLM performance

To improve the accuracy and general performance of LLMs many methods have been established throughout the last years. The most prevalent is retrieval augmented generation (**RAG**) [2]. Within this approach a vector database is used to store external knowledge from sources, e.g., textual, structured data from databases, enterprise systems, and many more. This information is then used by the LLM for augmentation. Another method is **LLM fine-tuning** [3] where a pre-trained LLM is retrained on a smaller, domain-specific dataset. Lastly, **in-context learning**, known as the most straightforward approach, only uses the input given through the context window (prompt) to generate better outputs — the method we test in this paper.

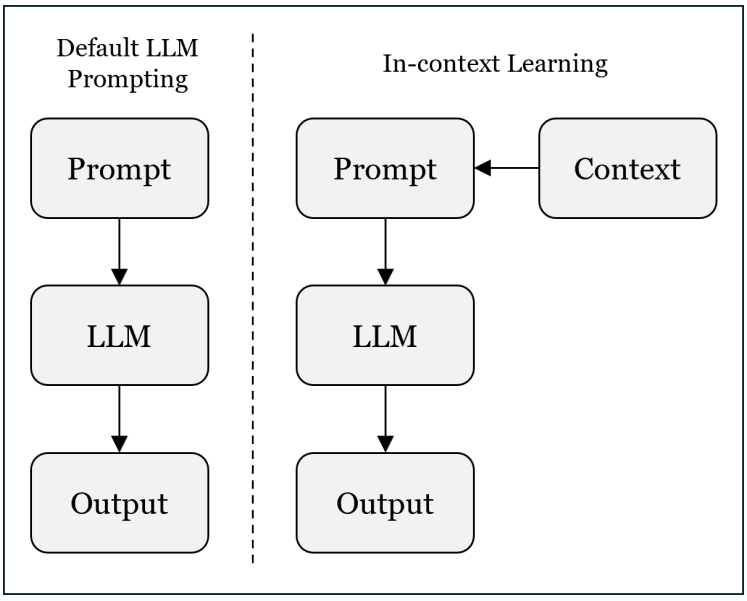

Figure 1: Default LLM Prompting vs. In-context Learning

These approaches have all one thing in common: enriching LLM systems with external knowledge sources to give the model more context (”to make it see”) — see Figure 1.

2.3. The Text-to-SQL problem

First attempts to solve the Text-to-SQL (or NL2SQL) problem were introduced in 2015 and were developed ever since. [4] In 2015, some first rule-based, statistical methods arose to translate natural language into SQL code but lacked performance for complex database systems. In 2019, deep learning modeling

approaches emerged [5], where the long-short-term-memory (LSTM) architecture is the focal point of transformation. This approach improved the accuracy substantially. Nevertheless, the real game changer was just ahead: In 2021, pre-trained language models (PLMs) showed very promising results but they lacked of individual task-oriented fine-tuning. Consequently, LLMs were derived from this around 2022 and are able to transform natural language into executable SQL queries, and the results convince. LLMs are trained on various different datasets such as Wikipedia, Project Gutenberg, or Reddit but more importantly also on GitHub and Kaggle from which the high capability of solving coding problems comes from.

In the next section, we dive deeper into important publications that highlight these three terms **LLM, in-context learning, and Text-to-SQL** from different angles.

3. Related Work

LLMs [6] are in the center of most Text-to-SQL tasks. Hence, a lot of improvements and new implementation methods emerge continuously. On the improvement side, the main players like Google, OpenAI, or Anthropic release frequently, newly trained and improved models. The same goes for open source models.

As mentioned, improving accuracy and therefore also reliability of LLMs is currently one of the biggest topics in the AI system research area. In-context learning was one of the first approaches after LLM were publicly available in 2022. Pourreza and Rafiei [7] show a 5% (from 79.9% to 85.3%) accuracy improvement of their DINSQL (in-context learning) approach against sophisticated fine-tuned models.

Retrieval augmented generation systems also can improve LLM performance. Vichev and Marchev [8] build their own custom evaluation model RAGSQL and test it against the renown BIRD benchmark. It performs very well with accuracy values above 90%. Another very important notion from the paper is *"We demonstrate that much smaller models with efficient fine-tuning can lead to higher performance on a task."* which implies that not only the big models have exceptional performance but also smaller models can play a huge part.

Currently, there is significant hype and enthusiasm surrounding AI agents which is a more direct problem-oriented approach. *Cooperative SQL Generation framework based on Multi-functional Agents (CSMA)* from [9] or *MAC-SQL* from [10] have to be highlighted in this context.

Other mentionable, related work comes from Zaharia et al. [11] on the optimization of LLM queries in relational workloads. The authors state correctly that LLM inference is (currently) very expensive and introduce various techniques to improve the LLM inference process. The already mentioned BIRD (BIg bench for laRge-scale Database grounded in text-to-SQL tasks) benchmark [12] introduces a 33.4 GB database system to hold against newly created systems and test their abilities.

4. Experiment

To validate our initial hypotheses from Section 1, we present an experimental set-up that will be explained in more depth within the following subsections.

4.1. Approach

As described in Sections 2.3 and 3, Text-to-SQL is a wide field and also the focal point here in this experiment. Figure 2 gives a high level overview of the general approach (referred to as in-context learning): We combine the user's question (input) which is of course in natural language and retrieve the current database schema. Consequently, we use both inputs (db_metadata and user_input) and create a prompt for the LLM. Then, based on this information the LLM creates a SQL query and runs it automatically on the database. The output is generated. To compare the performance of the in-context

learning approach, we hold it against our so called default LLM prompting technique (”default”). For this default technique, we only give the LLM the information that it is a customer database with the respective tables and of course the question. For the experiment, we use two sizes of databases db_small (Figure 3) and db_big (Figure 4) with dummy data based on a classical, fictional customer database.

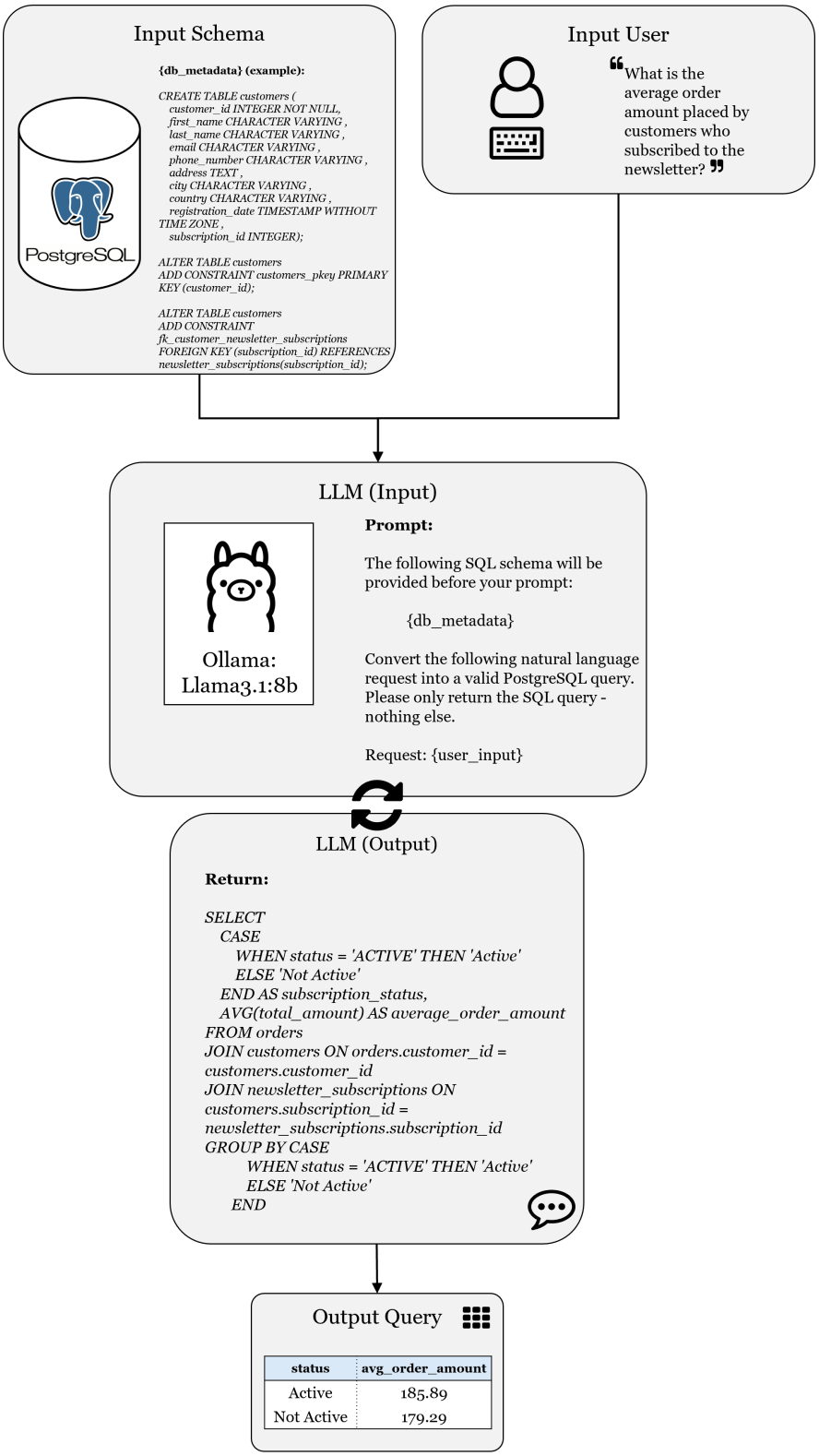

Figure 2: Integration and interplay of all relevant experimental components

The conceptual data models are depicted in the entity relationship diagrams in the appendix. Further-

more, we created 50 real-world questions (Table 3) for the experiment to test accuracy and execution time to compare results based on database size and the usage of in-context learning vs. default. For the questions, we were vigilant about including simple and also more complex questions to test how the system reacts to e.g., simple select statements, over more difficult joins, to complex nested queries. The default tests are done with a simple prompt without the database schema input.

4.2. Technical Setup

The technical setup basically consists of two systems interacting with each other through an API: the relational database management system (RDBMS) and the intelligent AI system.

- **RDBMS**

  We selected PostgreSQL as our relational database system to run this experiment as it is one of the most used database systems with a big and active community. Furthermore, it is open source and adheres to ACID properties.

- **AI system**

  The AI system is a LLM from Meta AI called 'meta-llama/Llama-3.1-8B-instruct'. This experiment is conducted on a local machine with a total of 16GB of RAM (8GB GPU + 8GB DDR5 RAM). To calculate which size a certain model must not exceed to be deployed on a machine, the following formula helps to give an estimate: [13] [14]

  $$Required\_RAM = \frac{P*Q}{8} * (1 + \beta)$$

P	Number of model parameters in billion
Q	Quantization of the model in bit (fp32, fp16, int8, int4)
8	No. of bits per byte
β	buffer capacity (20% is a common estimate)

  $$Required\_RAM = \frac{26*4}{8} * 1,2 = 15,6 \textit{ billion bytes}$$

So, the upper theoretical boundary to run the system locally, is a 26B model (assuming 4-bit quantization). Ceteris paribus, the required RAM on the local machine is 15,6 GB (<16GB). It is important to mention that models with > 8GB of RAM usage will be slower as it will access the DDR5 RAM after the GPU's RAM is maxed out — for trial purposes, a 24B model ran extremely slow on this machine. However, taking the strong ELO-Score of 1213 (see Section 2.1) into account, the 3.1:8B Meta model is a reasonable selection for this experiment while it only needs 4,8 GB of RAM.

4.3. Experiment Results

Overall, the results from our experiments (see Table 2) support our initial hypotheses:

Table 2
Experiment results

Default			**In-context learning**		
Database size	Accuracy	Execution time (sec)	Database size	Accuracy	Execution time (sec)
db_small	33.33%	2.9052	db_small	90.91%	3.1987
db_big	36.00%	2.7780	db_big	86.00%	3.2652

The first hypothesis **H1** *"in-context learning should perform better than the default."* holds for both db_small and db_big with accuracy values for the default vs. in-context learning of 33.33% vs. 90.91% (small) and 36.00% vs. 86.00% (big), respectively. Taking the average execution time into account, the second hypothesis **H2** *"in-context learning should possess a higher execution time due to having more complex inputs."* is also fulfilled: in-context learning possesses an average execution time of 3.1987 seconds (small) and 3.2652 seconds (big) versus 2.9052 seconds (small) and 2.7780 seconds (big). This comes as no surprise as the LLM must process the metadata from the database which logically takes longer than not applying this step. We were also able to observe some differences in database sizes. Bigger databases mean longer execution times and lower accuracy which is partially in line with **H3** *"The more complex a database is, the lower the accuracy should be."* For the default tests this hypothesis does not uphold as the results are reversed: the accuracy is higher for db_big. The accuracy outlier can be explained due to the randomized nature of the queries meaning that e.g., the right column name (total_amount vs. amount) is returned randomly from the LLM as it does not know the column names. For the in-context learning tests the hypothesis upholds with 86,00% (big) being lower than 90,91% (small). Another noticeable finding is also a relative simple one: Spelling. The LLM does not know from the provided metadata how certain things are spelled meaning e.g., question 27 "Retrieve a list of customers who have opted out of the newsletter." — from first glance a relative simple question (query) — failed only because the condition in the query for status = 'active' was misspelled as 'Active'. The same observation also occurred for the definition of the payment_method (credit card vs. Credit Card). In these two specific cases the ENUM() data type (instead of TEXT() or VARCHAR()) would have solved the problem, but if the possible values are not limited, the correctness of the results of these types of queries are random based on how the LLM decides to spell the word. This simple error represents the majority of the errors in our experiment — the LLM was not able to handle these type of questions. Another observation comes from question 25 "How many percent of customers come from Europe?": This one returned always an error independent of database size and method due to the fact that there is no information in the dataset which country belongs to Europe. An easy solution would have been to add a column "continent" in the customers table. To summarize the experiment results, it is important to mention that the in-context learning prompting technique worked surprisingly well with accuracy values around 90% even with some shortcomings when it comes to spelling or metrics definitions. The mentioned solutions to these shortcomings would have increased the accuracy even further.

5. Limitations

The technical implementation of this system comes along with a few limitations that are listed below:

5.1. Computing resources

As mentioned earlier in Subsection 4.2, we run this experiment on a local machine with a total RAM of 16 GB (8GB GPU + 8GB DDR5) that limits us to the usage of 26B-models (assuming 4-bit quantization). Alternatively, such experiments can also be run in the cloud where much more performant GPUs or even GPU clusters are available, but of course more costly — currently a memory-optimized virtual machine with ca. 500 GB of RAM (to run the most sophisticated open source LLMs) costs roughly **10$ per hour** at the major cloud providers.

5.2. Model selection

The computing resources directly impact the models that can be selected. As a rule of thumb, the more parameters a model has, the better it performs (see Table 1). But in our case it might not have changed the outcome as much as the errors arose from the structure of the data and not the created SQL code.

5.3. Sample size questionnaire

For the experiment, we used 50 questions to test the system. The questions attempt to be as close to the real-world as possible. One could argue why not more questions were used, but further questions would have been a derivation of the already existing ones resulting in similar SQL statements only with some very small changes.

6. Conclusion and Outlook

To come back to our initial, real-world notion about the business analyst from the introduction: *"So, is there a way to create a system that can help this person to gain insights from the data and answer their business questions?"*: Short answer — yes, but with some limitations. We showed that especially the in-context learning performs much better than default prompting. In addition, the more complex a database is, the lower the accuracy will be. Apart from this, we also compare execution times of the queries and find that in-context learning possesses higher execution times due to having to process more complex inputs. The most mistakes made by our system result from the fact that it simply cannot query what it does not know, e.g., spelling differences in the data values ("Active vs. active") or definitions ("Europe"). Future research should focus on solving these gaps. Also, the application of such systems to bigger and more complex data ecosystems like data warehouses or even data lakes might be interesting to address in the future. In addition to this, it might be interesting to take a deeper look into the security aspects: as this system runs automatically queries on a database, there is room for errors, e.g., unwanted deletion or unauthorized change of data.

References

[1] W. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, B. Zhu, H. Zhang, M. I. Jordan, J. E. Gonzalez, I. Stoica, Chatbot arena: An open platform for evaluating LLMs by human preference, Forty-first International Conference on Machine Learning (2024).

[2] P. S. H. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.

[3] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, J. Yi, W. Zhao, X. Wang, Z. Liu, H.-T. Zheng, J. Chen, Y. Liu, J. Tang, J. L. . M. Sun, Parameter-efficient fine-tuning of large-scale pre-trained language models., Nature Machine Intelligence 5 (2023) 220–235.

[4] A. Mohammadjafari, A. S. Maida, R. Gottumukkala, From Natural Language to SQL: Review of LLM-based Text-to-SQL Systems, CoRR abs/2410.01066 (2024).

[5] G. Katsogiannis-Meimarakis, G. Koutrika, A survey on deep learning approaches for text-to-sql, The VLDB Journal 32.4 (2023) 905–936.

[6] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J. Nie, J. Wen, A survey of large language models, CoRR abs/2303.18223 (2023).

[7] M. Pourreza, D. Rafiei, DIN-SQL: Decomposed in-context learning of text-to-sql with self- correction., Advances in Neural Information Processing Systems 36 (2023) 36339–36348.

[8] S. Vichev, A. Marchev, RAGSQL: Context Retrieval Evaluation on Augmenting Text-to-SQL Prompts, in: IEEE 12th International Conference on Intelligent Systems, IS, 2024, pp. 1–6.

[9] Z. Wu, F. Zhu, X. Shang, Y. Zhang, P. Zhou, Cooperative SQL Generation for Segmented Databases By Using Multi-functional LLM Agents., CoRR abs/2412.05850 (2024).

[10] B. Wang, C. Ren, J. Yang, X. Liang, J. Bai, Q. Zhang, Z. Yan, Z. Li, MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL, CoRR abs/2312.11242 (2023).

[11] S. Liu, A. Biswal, A. Kamsetty, A. Cheng, L. G. Schroeder, L. Patel, S. Cao, X. Mo, I. Stoica, J. E.

Gonzalez, M. Zaharia, Optimizing LLM queries in relational workloads., CoRR abs/2403.05821 (2024).

[12] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, X. Zhou, M. Chenhao, G. Li, K. Chang, F. Huang, R. Cheng, Y. Li, Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-sqls, Advances in Neural Information Processing Systems 36 (2023) 42330–42357.

[13] Q. Anthony, S. Biderman, H. Schoelkopf, Transformer math 101, https://blog.eleuther.ai/transformer-math/, 2023.

[14] C. Chen, Transformer inference arithmetic, https://kipp.ly/blog/transformer-inference-arithmetic, 2022.

Declaration on Generative AI

The authors hereby declare that no GenAI was used to generate text etc. following the guidelines and policy by CEUR-WS Policy on AI-Assisting Tools (https://ceur-ws.org/GenAI/Policy.html).

A. Entity relationship diagrams

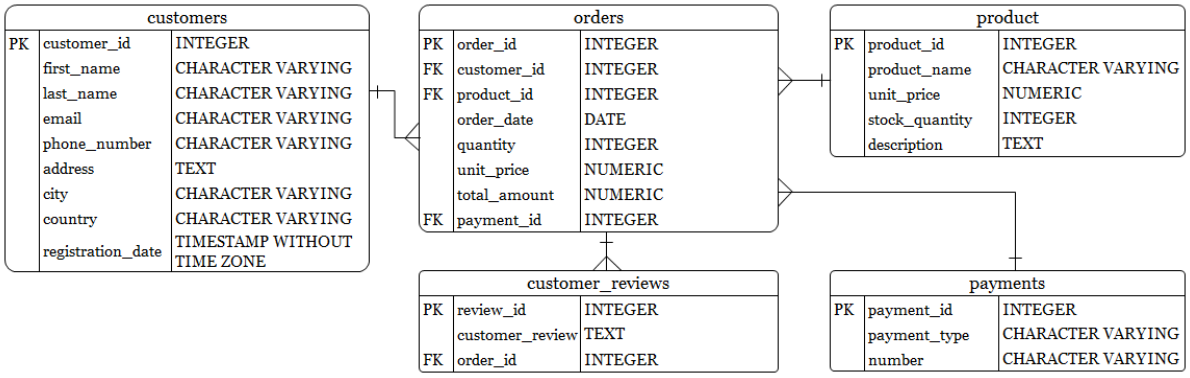

Figure 3: Entity Relationship Diagram DB Small

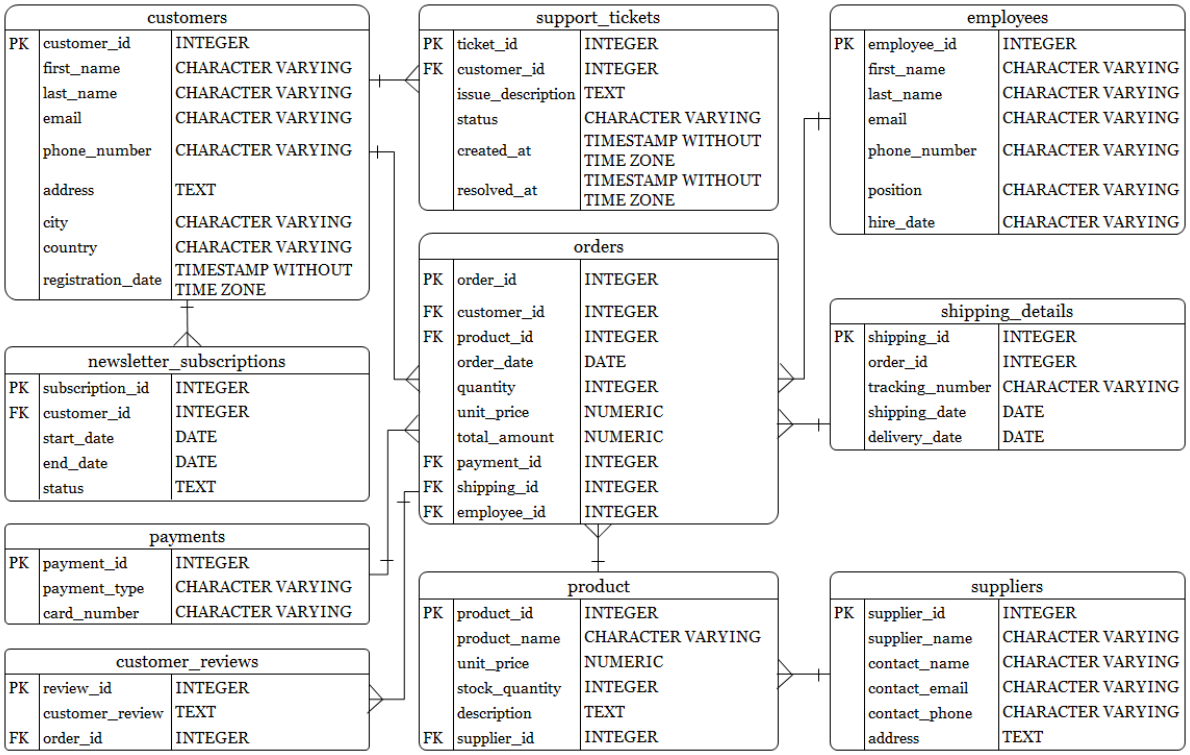

Figure 4: Entity Relationship Diagram DB Big

B. Sample questions

Table 3
Test questions

No.	Question
1	Who are our top 10 highest-spending customers?
2	What is the average order value across all customers?
3	Which country has the highest sales in total?
4	Which month had the highest sales?
5	What is the most common payment type used by customers?
6	Which payment type has the highest average order value?
7	What is the average time between purchase and payment completion?
8	What percentage of transactions are completed using credit card?
9	Which payment type is most commonly used by high-spending customers?
10	From which country are the most customers?
11	Which review was given for the highest order value?
12	How much stock of the product do we have in our warehouse?
13	Give me all e-mail addresses of every customer.
14	How often does the word 'great' occur in the customer reviews?
15	How long was the time when no orders occured?
16	Give me the ratios of all payment methods in percent. (CREDIT, DEBIT, PAYPAL)
17	What was the average quantity of order volume from customers from Germany?
18	What was the most common order value among all customers?
19	Which country has the highest spending single order customer?
20	What percentage of all customers left a review after their purchase?
21	How many customers placed their order within the first week of 2024?
22	Which customers left the longest review (word count)?
23	From which city are the most customers?
24	How many percent of customers come from Europe?
25	What is the most common last name of our customers?
26	Find the top 5 customers (full name, e-mail, amount spent) who have spent the most money.
27	Retrieve a list of customers who have opted out of the newsletter.
28	Retrieve the number of orders handled by each employee.
29	Retrieve the total revenue generated each month.
30	Find the most common payment type used by customers who spent more than $390.

No.	Question
31	Find the average time taken to resolve support tickets.
32	Retrieve a breakdown of revenue by country.
33	Find the employee who has resolved the most support tickets.
34	Find the average delivery time for all shipped orders.
35	Find the percentage of support tickets that were resolved within 24 hours.
36	How many support tickets are currently open?
37	What is the average order amount placed by customers who subscribed to the newsletter?
38	What is the average order value for orders placed on weekends vs weekdays?
39	Which countries have the highest proportion of customers subscribing to the newsletter?
40	Which top 5 countries have the most customers subscribing to the newsletter?
41	How many orders were placed in each week of the year?
42	What is the average sales amount for each day of the week?
43	How many support tickets are currently 'In Progress' and were created in calendar week 1?
44	How many products are supplied by suppliers with 'gmail.com' in their contact email?
45	How many orders used a payment type that is NOT 'CREDIT'?
46	List the first and last names of employees with the word 'Manager' in their position?
47	Which orders from which customers took longer than 7 days to deliver?
48	List the first and last names, and E-Mail address of customers whose email addresses end with '.net'?
49	List the customers' first and last name, and order dates for orders with a total amount between $100 and $200?
50	Provide a table with the employee's first, last name, and the customers they are associated with. Sort them by employee.