Bridging the Gap with RedSQL: A Russian Text-to-SQL Benchmark for Domain-Specific Applications

Brodskaia Irina
MIPT
brodskaiairina@gmail.com

Tutubalina Elena
AIRI, HSE University
tutubalina@airi.net

Somov Oleg
AIRI, MIPT
somov@airi.net

Abstract

We present the first domain-specific text-to-SQL benchmark in Russian, targeting fields with high operational load where rapid decision-making is critical. The benchmark spans across 9 domains, including healthcare, aviation, and others, and comprises 409 curated query pairs. It is designed to test model generalization under domain shift, introducing challenges such as specialized terminology and complex schema structures. Evaluation of state-of-the-art large language models (LLM) reveals significant performance drop in comparison to open-domain academic benchmarks, highlighting the need for domain-aware approaches in text-to-SQL. The benchmark is available at: https://github.com/BrodskaiaIrina/functional-text2sql-subsets

1 Introduction

Text-to-SQL parsing—the task of translating natural language questions into executable SQL queries over relational databases—has emerged as a core component of database question answering systems. These systems promise intuitive, NL-based interfaces for interacting with structured data, powering applications in customer support, business analytics, healthcare, and beyond (Abbas et al., 2022). This vision has fueled rapid progress in the field, driven by large-scale datasets such as Spider (Yu et al., 2018), WikiSQL (Zhong et al., 2017), and BIRD (Li et al., 2024), and advances in LLMs and semantic parsing techniques (Pourreza and Rafiei, 2023; Li et al., 2023; Gao et al., 2023; Somov and Tutubalina, 2023, 2025; Somov et al., 2024; Somov, 2025).

However, despite strong performance on academic benchmarks, state-of-the-art text-to-SQL models remain brittle when deployed in real-world domains. Practical applications often involve domain-specific terminology (e.g., ICD codes in healthcare, technical abbreviations in aviation), complex legacy schemas, and queries that arise under strict time and accuracy constraints. In such settings, even small misinterpretations—such as confusing "cycle time" with "lead time"—can lead to costly errors. Unfortunately, existing benchmarks prioritize breadth over depth, and typically exclude the very characteristics that make real-world deployment challenging: domain shift, schema ambiguity, and naturally occurring language.

Benchmarks like EHRSQL (Lee et al., 2021) and KaggleDBQA (Lee et al., 2022) has highlighted the mismatch between academic datasets and industrial environments. Real-world databases often contain opaque column names, sparse documentation, and organically evolving schema structures, none of which are well represented in academic benchmarks Spider or WikiSQL. Moreover, NL queries in practice are less schema-aware and more linguistically varied than those in curated datasets. As a result, models trained on general-domain benchmarks struggle to generalize to the distributions seen in production.

Currently, the only available Russian-language benchmark for the text-to-SQL task is PAUQ (Bakshandaeva et al., 2022), which focuses on academic, general-purpose queries. To bridge the gap between academic settings and domain-specific, real-world applications, we introduce **RedSQL**—the first Russian-language benchmark tailored to domain-specific text-to-SQL tasks. RedSQL comprises 409 carefully curated natural language–SQL query pairs spanning nine high-impact domains, including *healthcare*, *logistics*, and *aviation*. Each example is grounded in realistic schema structures, incorporates domain-specific terminology, and captures multi-step reasoning typical of operational environments. Our evaluation demonstrates a substantial decline in performance for general-purpose LLMs when applied to these domain-specific scenarios. By focusing on Russian-language usage and real-world complexity, RedSQL comple-

Proceedings of the 10th Workshop on Slavic Natural Language Processing (Slavic NLP 2025), pages 76–83

Dataset	# Examples	# DB	# Tables/DB	# Rows/Table	# Tables/Query
EHRSQL	24000	2	13.5	108000	2.4
KaggleDBQA	300	8	2.3	280000	1.2
RedSQL	409	9	15.4	338	4.6

Table 1: RedSQL statistics comparison with EHRSQL and KaggleDBQA.

Domain	Avg. Question Length	Avg. Query Length	Avg. Tables per DB	Avg. Columns per Table	Avg. Values per Query	Avg. Rows per Table	% Executed Queries (non-null)
banking	43	93	16	10	3	378	91
aviation	11	44	15	10	1	387	98
medicine	38	88	16	11	2	336	100
logistic	30	85	15	10	1	376	89
jurisprudence	20	69	15	9	1	366	73
architecture	27	88	17	9	2	336	93
energy	22	117	15	12	1	301	82
science	45	116	15	12	2	281	87
engineering	49	95	15	13	2	278	89

Table 2: Summary statistics of RedSQL across domains.

ments existing benchmarks and provides a valuable testbed for studying model robustness under domain shift—particularly in low-resource and non-English contexts.

2 RedSQL Benchmark Construction

We construct the **RedSQL** benchmark, a collection of domain-specific text-to-SQL datasets in Russian spanning **nine high-impact domains**: *banking, aviation, medicine, logistics, jurisprudence, architecture, energy, science and engineering*. These domains were selected due to their complex schema structures, specialized terminology, and high operational demands in real-world settings, where Text-To-SQL application would be really usefull. Despite the growing interest in text-to-SQL modeling, there remains a significant lack of domain-specific evaluation datasets in the Russian language. RedSQL addresses this gap by providing realistic, executable SQL queries paired with Russian natural language questions grounded in domain-aware relational databases.

Table 1 compares RedSQL with two widely used relevant domain-specific Text-To-SQL benchmarks: EHRSQL and KaggleDBQA. While EHRSQL provides a large number of examples, it is limited to only two databases, reducing schema diversity. KaggleDBQA includes more databases but operates over simplified schemas with fewer tables per query. In contrast, RedSQL strikes a balance between size and complexity: it spans nine distinct domains, features the highest average number of tables per database (15.4), and requires more complex queries involving an average of 4.6 tables per query (most queries refer to 3-6 tables - see Appendix B). These characteristics make RedSQL more reflective of real-world complexity in domain-specific applications and better suited for evaluating generalization under schema and linguistic shift.

The dataset construction pipeline generates natural language–SQL pairs and corresponding relational databases for query execution. The process has four major steps:

1. **Domain-Specific Schema Design:** For each domain, a database schema was manually constructed based on an analysis of key entities and their relationships. For example, the medical domain includes interlinked entities such as *doctors, patients, diagnoses*, and *prescriptions*, while the aviation domain connects *airports, pilots, flights*, and *aircraft*. These conceptual mappings were encoded into SQL using domain-representative DDL (Data Definition Language). LLMs were also prompted to assist in schema generation where appropriate.

2. **Data Population:** The constructed schemas were instantiated as SQLite databases and populated with synthetic data. For generic fields (e.g., names, addresses, transaction logs), we used the Python `Faker`[1] library. Domain-

[1] https://pypi.org/project/Faker/

specific content (e.g., clinical diagnoses or flight plans in aviation domain) was generated using LLMs such as GPT-4o and DeepSeek (Liu et al., 2024), producing realistic, context-aware data entries.

3. **SQL Query Generation:** Given the populated databases, SQL queries of varying complexity were generated. Basic queries were synthesized using GPT models, while more complex queries requiring multi-table joins, nested subqueries, or temporal reasoning were created using Cursor AI, leveraging models such as Claude and Gemini 2.5. Queries were manually reviewed to ensure they are executable and semantically valid.

4. **Natural Language Question Formulation:** For each SQL query, a corresponding natural language question was generated in Russian. This step employed a mix of GPT models and Cursor AI to ensure fluency, domain specificity, and alignment with realistic user queries.

The full pipeline was manually reviewed by graduate computer science student to validate database structure, SQL correctness, and natural language alignment. The resulting benchmark includes diverse domains with varying schema complexity, query types, and linguistic patterns. Dataset statistics are provided in Table 2.

3 Experiments

To assess the complexity of the RedSQL benchmark, we conducted an evaluation using several popular LLMs under a few-shot prompting setting. For each domain, the prompt included the corresponding database schema, a small sample of representative data, and five reference text-to-SQL pairs. Each query from the benchmark was evaluated using two separate prompts: one in English and one in Russian. This evaluation aims to address the following research questions:

1. How well do modern LLMs generalize to unseen domain-specific text-to-SQL tasks?

2. What is the impact of prompt language (Russian vs. English) on model performance?

We adopt the **execution match** metric for evaluation. A prediction is considered correct if the result

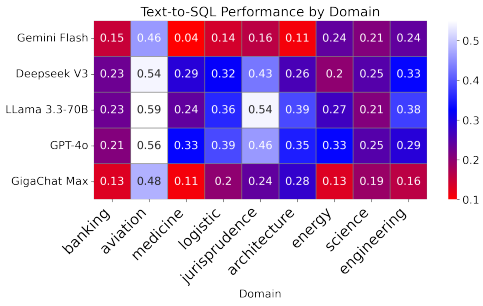

Figure 1: Performance of LLMs across RedSQL domains. The performance is measured via Execution Match between generated query and gold query.

returned by executing the predicted SQL query is identical to that of the gold (reference) query. We further extend the metric to tolerate predictions that include superfluous attributes, as long as the required answer can be unambiguously inferred from the returned result set. It has been found that models often fail to display all required columns. If at least one column is missing after the execution of the predicted query, the metric value becomes zero. As a result, the overall metric values were quite low with an average accuracy of 28% across all spheres and models. To address this issue, we have also calculated an additional **soft execution match** metric that measures the proportion of correctly displayed columns in the predicted query output, which reports the average accuracy of 41%.

The following LLMs were included in the evaluation – Gemini Flash, DeepSeek V3 (Liu et al., 2024), Meta LLaMA 3.3 70B Instruct (Grattafiori et al., 2024), OpenAI GPT-4o, GigaChat Max (Russian LLM)[2]. All models were prompted under the same configuration, with temperature fixed at 0 to ensure deterministic outputs. The result, with English prompting, is presented in Figure 1. Full Execution Match results are available at Table 3. The same table encompasses the performance metrics of identical models on the PAUQ dataset. The table demonstrates the divergence in the model's performance on the existing academic dataset and on a domain-specific benchmark, highlighting the gap between the existing datasets and our newly introduced one.

To answer our first research question—*how well do modern LLMs generalize to unseen domain-specific text-to-SQL tasks?*—we find that general-purpose LLMs experience significant performance degradation in domains with specialized terminol-

[2] https://giga.chat/

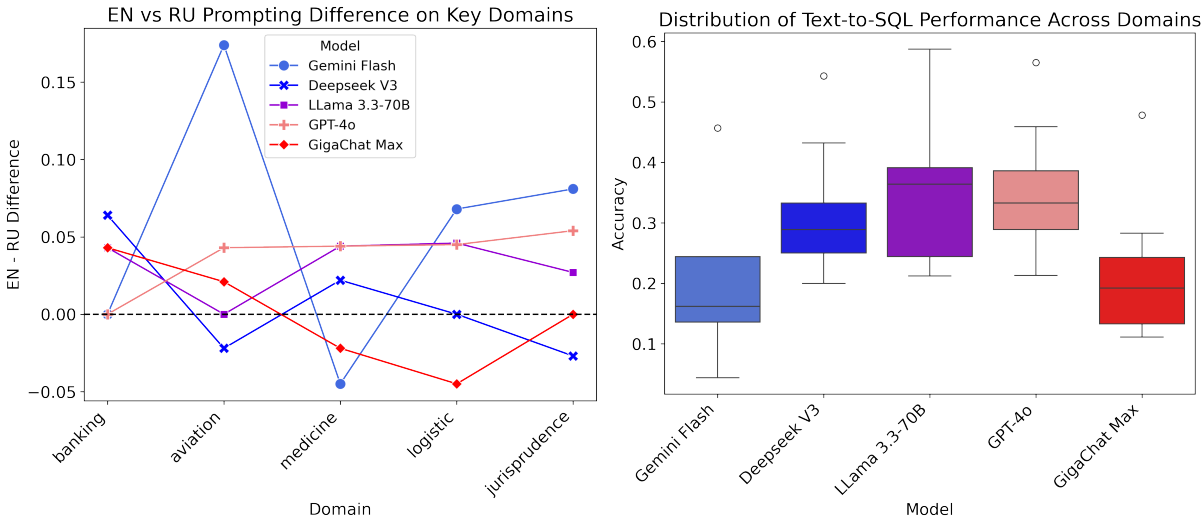

Figure 2: Comparison of prompt language sensitivity **(left)** and domain-wise model variability **(right)**. **(Left)** Points above 0 mean English version outperforms Russian, points below 0 mean Russian version outperforms English, dashed line at 0 marks parity. **(Right)** The box plots are based on experiments using English prompts, which generally yielded higher performance compared to Russian prompts.

ogy and complex schema structures. In particular, the **medicine**, **science**, and **banking** domains consistently yielded the lowest execution accuracy across all models, with Gemini Flash performing notably poorly (e.g., 0.044 in medicine with English prompts). In contrast, domains such as **aviation** and **jurisprudence** proved easier, achieving higher scores, especially for Meta LLaMA and DeepSeek, likely due to more regular schema structures and training-aligned terminology.

Comparing the hard execution match metrics (Table 3) and soft execution match (Table 4), we see that it is difficult for the models to identify all the required columns. In cases such as our benchmark, where each query requires the return of a large number of columns, and they are not always clearly stated in the question, the soft execution match metric may provide better understanding of models performance.

We also conducted an analysis of the errors committed by the models when generating queries. The error rates were computed for each component of the SQL query (Appendix A.1), and precision and recall metrics of tables and columns prediction were determined across all evaluated models (Appendix A.2).

The results indicate that the models have the worst performance in predicting complex logic, with the error rate for operations such as SUBQUERIES and HAVING being the highest. On the other hand, simple components such as LIMIT and ORDER BY were predicted with the best accuracy.

The precision and recall metrics analysis demonstrated that models generally perform better at **column identification** than **table identification** across most domains. This pattern suggests that once the relevant tables are identified, models are more successful at selecting appropriate columns within those tables. The gap between table and column performance is most pronounced in complex domains like medicine and engineering, indicating that schema understanding remains a bottleneck.

Addressing our second research question—*what is the impact of prompt language (Russian vs. English) on model performance?*—we observe that prompt language can significantly influence results, particularly in complex domains. As shown in Figure 2 (left), performance gaps between English and Russian prompts vary by model and domain. Gemini Flash and GPT-4o perform better with English prompts, while DeepSeek V3 shows more balanced results, and GigaChat Max appears better tuned to Russian-language instructions.

Figure 2 (right) further illustrates that model robustness also varies: **GPT-4o** exhibits the most stable and consistently high performance, while models such as **Gemini Flash** and **GigaChat Max** show greater variability and underperformance in challenging domains.

Domain	Gemini Flash		Deepseek V3		Llama 3.3-70B		GPT-4o		GigaChat Max	
	EN	RU	EN	RU	EN	RU	EN	RU	EN	RU
banking	0.149	0.149	0.234	0.170	0.234	0.191	0.213	0.213	0.128	0.085
aviation	0.457	0.283	0.543	0.565	0.587	0.587	0.565	0.522	0.478	0.457
medicine	0.044	0.089	0.289	0.267	0.244	0.200	0.333	0.289	0.111	0.133
logistic	0.136	0.068	0.318	0.318	0.364	0.318	0.386	0.341	0.205	0.250
jurisprudence	0.162	0.081	0.432	0.459	0.541	0.514	0.459	0.405	0.243	0.243
architecture	0.109	0.152	0.261	0.261	0.391	0.348	0.348	0.304	0.283	0.261
energy	0.244	0.267	0.200	0.244	0.267	0.289	0.333	0.378	0.133	0.222
science	0.212	0.192	0.250	0.231	0.212	0.250	0.250	0.269	0.192	0.173
engineering	0.244	0.178	0.333	0.333	0.378	0.222	0.289	0.244	0.156	0.156
PAUQ	0.785	0.772	0.747	0.759	0.715	0.719	0.737	0.747	0.700	0.711

Table 3: Model Execution Accuracy on Functional Subsets and PAUQ dataset (English vs. Russian Prompts).

4 Conclusion

This study introduces **RedSQL**, the first benchmark for evaluating Text-To-SQL systems in domain-specific settings using the Russian language. Covering nine high-impact domains, RedSQL provides a realistic and linguistically diverse evaluation environment that exposes important limitations in modern LLMs.

Through systematic evaluation across English and Russian prompts, we observe that model performance varies significantly depending on both the domain and prompt language. Domains such as *medicine*, *science*, and *banking* emerge as particularly challenging due to their complex schema structures and domain-specific terminology. In contrast, *aviation* and *jurisprudence* show relatively higher performance, likely due to more regular schemas and simpler question patterns.

We also find that prompt language plays a non-trivial role: certain models, particularly Gemini Flash and GigaChat Max, exhibit higher sensitivity to Russian prompting, while models like GPT-4o demonstrate more consistent cross-lingual performance. Box plot analysis further reveals that models differ not only in average accuracy but also in robustness across domains, with GPT-4o showing the most stable results.

These findings highlight the need for stronger domain adaptation, more effective multilingual prompting strategies, and robust evaluation benchmarks in non-English and domain-specific scenarios. RedSQL offers a foundation for advancing these goals and improving the reliability of text-to-SQL systems in realistic deployments.

5 Limitations

While RedSQL supports evaluation in domain-specific and Russian-language settings, it has several limitations. First, despite covering nine domains, the datasets are synthetically generated and may not reflect the full complexity or noise of real-world databases and queries. All tables and contents are AI-generated and, though human-verified, may lack real-world diversity. Second, our evaluation uses few-shot prompting without fine-tuning or retrieval, potentially underestimating the performance achievable with specialized adaptation. We leave these improvements for future work.

Acknowledgments

This research was supported in part through computational resources of HPC facilities at HSE University (Kostenetskiy et al., 2021).

References

Shanza Abbas, Muhammad Umair Khan, Scott Uk-Jin Lee, Asad Abbas, and Ali Kashif Bashir. 2022. A review of nlidb with deep learning: findings, challenges and open issues. *IEEE Access*, 10:14927–14945.

Daria Bakshandaeva, Oleg Somov, Ekaterina Dmitrieva, Vera Davydova, and Elena Tutubalina. 2022. Pauq: Text-to-sql in russian. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2355–2376.

Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *CoRR*, abs/2308.15363.

Domain	Gemini Flash		Deepseek V3		Llama 3.3-70B		GPT-4o		GigaChat Max	
	EN	RU	EN	RU	EN	RU	EN	RU	EN	RU
banking	0.231	0.212	0.387	0.336	0.368	0.272	0.334	0.330	0.251	0.214
aviation	0.529	0.467	0.609	0.609	0.620	0.634	0.601	0.591	0.514	0.504
medicine	0.170	0.143	0.434	0.449	0.446	0.402	0.499	0.472	0.244	0.256
logistic	0.224	0.195	0.344	0.329	0.366	0.305	0.404	0.342	0.257	0.323
jurisprudence	0.402	0.356	0.614	0.604	0.622	0.631	0.629	0.618	0.392	0.405
architecture	0.285	0.318	0.428	0.414	0.470	0.486	0.486	0.461	0.361	0.420
energy	0.350	0.411	0.367	0.470	0.371	0.465	0.529	0.520	0.310	0.394
science	0.444	0.422	0.497	0.468	0.406	0.421	0.472	0.500	0.365	0.353
engineering	0.380	0.377	0.541	0.56	0.533	0.508	0.457	0.425	0.347	0.321

Table 4: Model Soft Execution Accuracy on Functional Subsets (English vs. Russian Prompts).

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

P. S. Kostenetskiy, R. A. Chulkevich, and V. I. Kozyrev. 2021. HPC Resources of the Higher School of Economics. *Journal of Physics: Conference Series*, 1740(1):012050.

Chia-Hsuan Lee, Oleksandr Polozov, and Matthew Richardson. 2021. KaggleDBQA: Realistic evaluation of text-to-SQL parsers. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2261–2273, Online. Association for Computational Linguistics.

Gyubok Lee, Hyeonji Hwang, Seongsu Bae, Yeonsu Kwon, Woncheol Shin, Seongjun Yang, Minjoon Seo, Jong-Yeup Kim, and Edward Choi. 2022. Ehrsql: A practical text-to-sql benchmark for electronic health records. *Advances in Neural Information Processing Systems*, 35:15589–15601.

Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. 2023. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13067–13075.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. 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.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems*, 36:36339–36348.

Oleg Somov. 2025. The generalization and error detection in llm-based text-to-sql systems. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, WSDM '25, page 1077–1079, New York, NY, USA. Association for Computing Machinery.

Oleg Somov, Alexey Dontsov, and Elena Tutubalina. 2024. AIRI NLP team at EHRSQL 2024 shared task: T5 and logistic regression to the rescue. In *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 431–438, Mexico City, Mexico. Association for Computational Linguistics.

Oleg Somov and Elena Tutubalina. 2023. Shifted PAUQ: Distribution shift in text-to-SQL. In *Proceedings of the 1st GenBench Workshop on (Benchmarking) Generalisation in NLP*, pages 214–220, Singapore. Association for Computational Linguistics.

Oleg Somov and Elena Tutubalina. 2025. Confidence estimation for error detection in text-to-sql systems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(23):25137–25145.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Error type	Gemini Flash		Deepseek V3		Llama 3.3-70B		GPT-4o		GigaChat Max	
	EN	RU	EN	RU	EN	RU	EN	RU	EN	RU
SELECT	0.172	0.175	0.151	0.159	0.162	0.152	0.156	0.146	0.212	0.181
FROM	0.175	0.177	0.155	0.160	0.159	0.150	0.154	0.144	0.220	0.187
WHERE	0.250	0.261	0.196	0.208	0.237	0.228	0.215	0.212	0.263	0.235
JOIN	0.129	0.121	0.113	0.110	0.115	0.114	0.104	0.108	0.138	0.132
ORDER BY	0.079	0.062	0.055	0.045	0.086	0.086	0.051	0.055	0.123	0.099
JOIN_TYPES	0.170	0.162	0.143	0.143	0.135	0.134	0.134	0.136	0.178	0.189
GROUP BY	0.141	0.136	0.157	0.156	0.163	0.150	0.132	0.129	0.212	0.183
AGGREGATE	0.138	0.141	0.107	0.107	0.105	0.106	0.102	0.115	0.117	0.122
LIMIT	0.059	0.036	0.024	0.032	0.059	0.059	0.016	0.016	0.059	0.059
SUBQUERY	0.347	0.347	0.340	0.386	0.317	0.301	0.336	0.324	0.351	0.359
HAVING	0.299	0.280	0.309	0.315	0.375	0.379	0.303	0.307	0.395	0.352
DISTINCT	0.236	0.232	0.227	0.217	0.205	0.192	0.229	0.236	0.278	0.280

Table 5: Error rates by model and SQL component (English vs. Russian prompts).

Domain	Gemini Flash		Deepseek V3		Llama 3.3-70B		GPT-4o		GigaChat Max	
	Tables	Columns	Tables	Columns	Tables	Columns	Tables	Columns	Tables	Columns
banking	0.51 / 0.54	0.55 / 0.61	0.52 / 0.51	0.55 / 0.57	0.51 / 0.50	0.58 / 0.60	0.50 / 0.52	0.55 / 0.58	0.47 / 0.46	0.43 / 0.48
medicine	0.41 / 0.45	0.75 / 0.71	0.48 / 0.53	0.77 / 0.72	0.50 / 0.54	0.79 / 0.77	0.51 / 0.55	0.83 / 0.80	0.29 / 0.32	0.55 / 0.55
aviation	0.79 / 0.81	0.92 / 0.92	0.87 / 0.87	0.94 / 0.94	0.84 / 0.86	0.92 / 0.95	0.83 / 0.84	0.92 / 0.95	0.71 / 0.74	0.89 / 0.94
science	0.65 / 0.71	0.65 / 0.85	0.67 / 0.68	0.68 / 0.84	0.65 / 0.70	0.66 / 0.85	0.65 / 0.71	0.68 / 0.83	0.54 / 0.57	0.56 / 0.74
engineering	0.44 / 0.39	0.73 / 0.67	0.47 / 0.45	0.69 / 0.62	0.58 / 0.56	0.82 / 0.78	0.50 / 0.45	0.77 / 0.76	0.33 / 0.32	0.39 / 0.41
jurisprudence	0.65 / 0.69	0.80 / 0.90	0.72 / 0.75	0.81 / 0.84	0.77 / 0.77	0.84 / 0.90	0.76 / 0.79	0.81 / 0.87	0.54 / 0.54	0.61 / 0.73
logistic	0.63 / 0.66	0.74 / 0.69	0.71 / 0.73	0.68 / 0.67	0.67 / 0.69	0.67 / 0.67	0.69 / 0.69	0.68 / 0.67	0.51 / 0.54	0.52 / 0.58
architecture	0.64 / 0.65	0.70 / 0.75	0.68 / 0.70	0.79 / 0.75	0.69 / 0.65	0.77 / 0.73	0.74 / 0.73	0.77 / 0.79	0.63 / 0.65	0.70 / 0.76
energy	0.41 / 0.41	0.40 / 0.46	0.55 / 0.52	0.55 / 0.53	0.48 / 0.42	0.46 / 0.45	0.52 / 0.51	0.52 / 0.51	0.45 / 0.50	0.46 / 0.52

Table 6: Model Precision (left) and Recall (right) Metrics on Functional Subsets (Tables vs. Columns).

A Error Analysis

A.1 SQL Component Error Analysis

To gain deeper insights into model failures, we conducted a detailed error analysis focusing on specific SQL components, using the Python `sqlparse`[3] library. Table 5 presents error rates for different SQL components across all evaluated models. The analysis reveals several key patterns:

- **Complex constructs are most problematic:** SUBQUERY and HAVING clauses consistently show the highest error rates across all models (30-39%), indicating that models struggle with nested logic and conditional aggregation.

- **WHERE clause challenges:** WHERE clauses show relatively high error rates (20-26%), suggesting difficulties in correctly translating natural language conditions into SQL predicates.

- **Basic operations are more reliable:** Simple constructs like LIMIT and ORDER BY show lower error rates (2-12%), indicating that models handle straightforward sorting and limiting operations more successfully.

[3] https://pypi.org/project/sqlparse/

A.2 Classification Metrics Analysis

To better understand model performance, we analyzed precision and recall for table and column identification across domains. True positives are matches between gold and predicted queries; false positives and other cases are defined accordingly. Results are shown in Table 6.

B Distribution of the number of tables

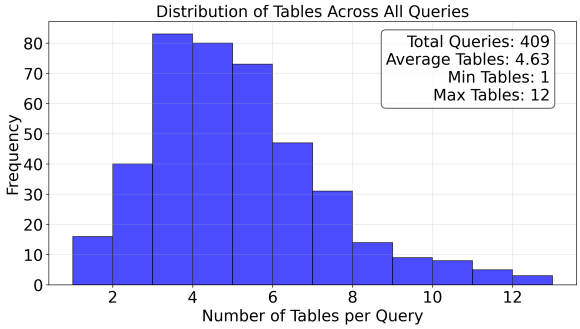

Figure 3: Distribution of the number of tables among all queries in the benchmark.