CenQuery: A Domain-Specific Text-to-SQL Dataset and Baseline Analysis for Indian Census Data

Sourish Kanna
Department of Computer Science
SIES Graduate School of Technology
Navi Mumbai, India

Maharajan Konar
Department of Computer Science
SIES Graduate School of Technology
Navi Mumbai, India

Nandini Shende
Department of Computer Science
SIES Graduate School of Technology
Navi Mumbai, India

G U Gopikha
Department of Computer Science
SIES Graduate School of Technology
Navi Mumbai, India

Suvarna Chaure
Department of Computer Science
SIES Graduate School of Technology
Navi Mumbai, India

***Abstract*—Accessing Indian Census data traditionally requires proficiency in complex SQL queries, creating a barrier for policymakers. While Large Language Models (LLMs) show promise in Text-to-SQL tasks, their effectiveness on secure, offline infrastructure remains under-explored for specific administrative domains. This paper presents two key contributions: (1) A comparative baseline analysis of state-of-the-art models (Llama-3-70B, SQLCoder-8B) on standard benchmarks (Spider, WikiSQL), revealing a significant performance gap between cloud-based models (69.9%) and privacy-preserving local models (52.28%), and (2) The construction of Census-650, a novel, execution-validated dataset specifically tailored to the Indian Census domain to address these gaps. We identify critical challenges in domain-specific schema linking and propose the CenQuery architecture, a roadmap for a fine-tuned, privacy-preserving interface. This work establishes the necessary data foundation for developing secure government query systems.**

***Index Terms*—Text-to-SQL, Transformer Models, Low-Rank Adaptation (LoRA), Schema-Aware Prompting, Query Optimization, Indian Census**

I. Introduction

In the era of big data, relational databases serve as the backbone of modern information systems, storing vast amounts of structured data critical for analysis and decision-making. However, accessing this data traditionally requires proficiency in Structured Query Language (SQL), a skill often lacking among business analysts, policymakers, and domain experts. This technical barrier creates a bottleneck in which non-technical users must depend on data engineers to retrieve even simple insights.

Natural Language to SQL (NL2SQL) interfaces aim to address this challenge by enabling users to query databases using plain English. While recent advances in Large Language Models (LLMs), such as GPT-4, have demonstrated strong language understanding capabilities, their deployment in enterprise and government settings presents several challenges: (i) hallucinations in the form of non-existent column or table names, (ii) data privacy concerns arising from transmitting sensitive schemas to external cloud-based APIs, and (iii) operational costs that scale linearly with usage.

A. Problem Statement

The Indian Census represents one of the largest and most complex administrative data collections globally. Despite its societal importance, extracting actionable insights from census data remains challenging due to highly normalized schemas, hierarchical table relationships, and inconsistent naming conventions. Existing general-purpose NL2SQL models often fail to generalize to such domain-specific structures, frequently producing syntactically valid but semantically incorrect queries (e.g., confusing `State_Code` and `District_Code` across joined tables).

B. Contributions

This paper introduces *CenQuery*, a domain-specific NL2SQL system tailored to the complex and privacy-sensitive schema of the Indian Census. Unlike generic approaches that rely on large, closed-source models, CenQuery adopts a Parameter-Efficient Fine-Tuning (PEFT) strategy. The key contributions of this work are as follows:

- A schema-aware NL2SQL architecture based on the Defog Llama-3-SQLCoder-8B model, adapted using Low-Rank Adaptation (LoRA) and quantization-aware fine-tuning.
- A curated domain-specific dataset comprising 650 natural language–SQL training pairs derived from Indian census tables.
- A systematic baseline analysis demonstrating the performance gap of off-the-shelf local models and motivating the need for domain-specific adaptation on consumer-grade hardware.

The remainder of this paper is organized as follows. Section II reviews related work on Text-to-SQL systems and repair-based approaches. Section III presents the proposed privacy-preserving system architecture. Section IV describes the methodology, including the LoRA adaptation strategy and the end-to-end implementation pipeline. Section V provides the baseline performance analysis that motivates our approach.

Finally, Section VI concludes the paper and outlines directions for future research.

II. Literature Survey

A. Repair-based Approaches

One line of work focuses on repairing failed SQL queries generated by LLMs. For example, mutation and structural repair techniques analyse errors and attempt to fix them without retraining [1]. While effective in salvaging outputs, these approaches operate post-hoc and do not improve the core generation ability of the model.

B. Ontology and Rule-based Systems

Earlier systems, such as ontology-driven query mapping pipelines, relied on linguistic resources and rule-based transformations [2]. These methods demonstrated robustness in structured domains but lacked scalability and generalization in diverse, real-world schemas. In contrast, our fine-tuned Llama-3 approach leverages large-scale datasets and schema-aware prompting, offering greater flexibility across domains.

C. Deep Learning and Decoding Innovations

Comprehensive surveys have catalogued seq-to-seq models, syntax-tree decoders, and execution-guided methods [3]. While providing valuable background, they often do not propose deployable architectures. Similarly, recent work explores hybrid decoding techniques that combine sketch-based and sequential generation to reduce syntax errors [4]. These innovations improve decoding quality but are often model-agnostic.

D. Reliability and Benchmarks

Operational frameworks such as the LLM maturity model emphasize accuracy, consistency, and transparency in LLM-driven applications [5]. Complex benchmarks like TPC-DS have been used to stress-test these models, showing that state-of-the-art LLMs still fail on structurally complex queries involving multiple joins and subqueries [6].

E. Domain-Specific Applications

Evaluating Text-to-SQL for Software Engineering needs shows that fine-tuning on domain-specific datasets can drastically improve accuracy to as high as 94% [7]. Similar studies in the medical domain using MIMIC-III datasets highlight unique challenges such as specialized terminology [8]. Recent surveys discuss the evolution from rule-based systems to modern LLMs, emphasizing the ongoing challenge of schema linking [9]. Finally, the BIRD benchmark introduces large-scale, diverse datasets to measure performance on unseen databases, a key inspiration for our census-specific validation [10].

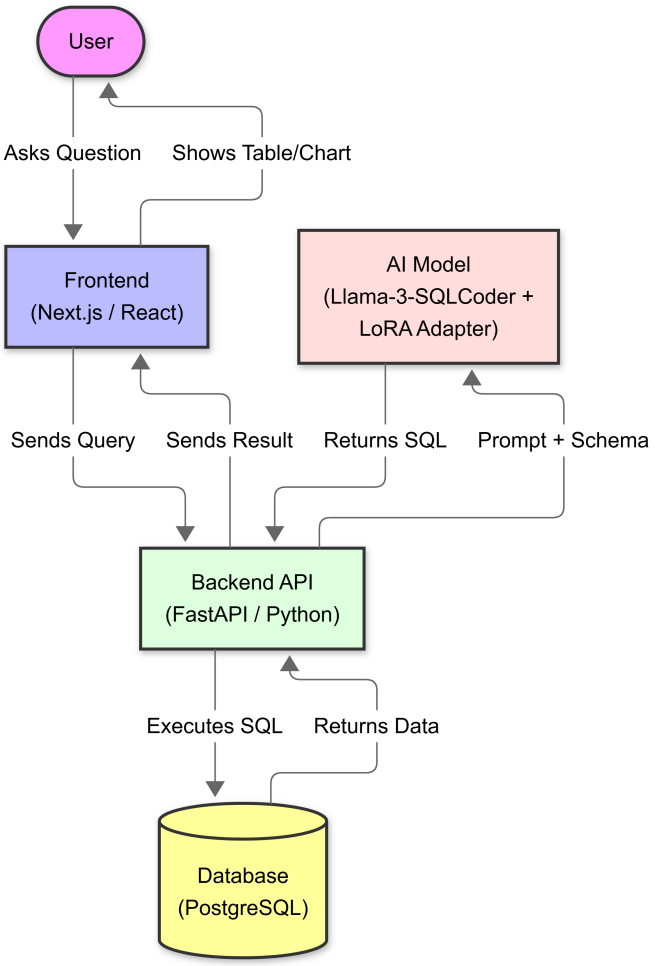

Fig. 1. Proposed System Architecture

III. Proposed System Architecture

The proposed system is built around the Llama-3-SQLCoder-8B transformer model, fine-tuned using LoRA adapters for parameter-efficient adaptation. The architecture comprises the following components (Fig. 1):

- **User Interface Layer:** A Next.js (React) frontend provides a chat interface for users to input natural language queries.
- **Application Layer:** A FastAPI (Python) backend acts as the orchestrator. It handles session management, rate limiting, and forwards requests to the inference engine.
- **Intelligence Layer:** The core of the system is the Fine-Tuned Llama-3-SQLCoder-8B model hosted on a Hugging Face Inference Endpoint. It utilizes a LoRA adapter to convert natural language into SQL.
- **Data Layer:** A PostgreSQL database hosts the 12 normalized census tables. It executes the generated SQL and returns the structured result set to the user.

As illustrated in Fig. 1, the system follows a modular three-tier architecture. The User Layer (Next.js) captures natural

language input and renders visualization components such as tables and charts. The Application Layer (FastAPI) serves as the secure orchestrator, managing session states and routing requests to the Intelligence Layer. This core layer hosts the local inference engine (Llama-3-SQLCoder-8B), which processes the schema-aware prompts without transmitting data to external cloud providers. Finally, the Data Layer executes the resulting SQL queries against the isolated PostgreSQL database, ensuring end-to-end data sovereignty.

IV. Methodology and Algorithms

A. The Census-650 Dataset Construction

Existing Text-to-SQL benchmarks like Spider or WikiSQL lack the specific schema complexity of the Indian Census (e.g., hierarchical geographic regions, caste-based aggregations). To address this, we developed **Census-650**, a domain-specific instruction-tuning dataset constructed through a three-stage pipeline.

1) **Schema-Guided Generation:** We first mapped the 12 normalized tables of the census database (Population, Health, Education, etc.) to a set of 50 core analytical intents (e.g., "Compare literacy rates between districts"). This ensured diverse coverage of SQL operations, including `JOIN`, `GROUP BY`, and window functions.
2) **Human-in-the-Loop Refinement:** A team of domain experts generated Natural Language Questions (NLQ) corresponding to these SQL templates, ensuring linguistic diversity that reflects real-world user queries.
3) **Execution-Based Validation:** A critical limitation of many synthetic datasets is invalid SQL. We implemented a strict *Execution Validity Check* pipeline. As illustrated in our generation logs, every query in the dataset was executed against the live PostgreSQL backend. Only queries that returned non-empty, valid result sets were retained in the final corpus.

Dataset Statistics:

- **Total Pairs Generated:** 650
- **Validation Criteria:** Syntax correctness + Non-empty return set.

This dataset serves as a proposed benchmark for future domain adaptation, addressing the specific linguistic and structural complexities identified in our baseline analysis.

B. Complexity Analysis

The resulting dataset differs from generic benchmarks in its reliance on specific Indian administrative terms (e.g., "Tehsil", "NCT of Delhi", "SC/ST") and deep join depths (up to 3 tables), which we identify as the primary failure points for generic LLMs.

C. Low-Rank Adaptation (LoRA)

Training a Large Language Model (LLM) with 8 billion parameters is computationally prohibitive. We employ Low-Rank Adaptation (LoRA), which freezes the pre-trained model weights and injects trainable rank decomposition matrices into the Transformer's attention layers. [11]

$$h = W_0 x + \Delta W x = W_0 x + \frac{\alpha}{r}(BA)x \tag{1}$$

Let $W_0 \in \mathbb{R}^{d \times k}$ represent the frozen pre-trained weights. Instead of updating W_0 directly, we constrain the weight update ΔW by representing it as the product of two low-rank matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$, where the rank $r \ll \min(d, k)$. The forward pass for a hidden layer h is modified as shown above, where:

- x is the input vector,
- r is the rank (set to 16 in our experiments),
- α is the scaling factor (set to 32).

This technique reduces the number of trainable parameters by approximately 98%, allowing us to fine-tune the model on a single consumer-grade GPU (T4).

D. Schema-Aware Instruction Tuning

Generic LLMs often fail to distinguish between similar column names (e.g., `tot_p` vs `total_population`). To mitigate this, we implemented a Schema-Aware Prompting Strategy. The input sequence X is constructed dynamically for each query:

$$X = \{\text{Instruction} \oplus \text{Schema Context} \oplus \text{Natural Language Question}\} \tag{2}$$

The model is trained to maximize the conditional probability of the target SQL query Y:

$$\max_{\Phi} \sum_{(X,Y) \in D} \log P(Y \mid X; \Phi) \tag{3}$$

E. Query Generation Algorithm

Given a user question Q and database schema S, the system generates an optimized SQL query Y using the following procedure:

1) Extract schema metadata: $S_{\text{context}} \leftarrow \texttt{ExtractTableMetadata}(S)$
2) Construct schema-aware prompt: Prompt $\leftarrow$ ``### Task: '' $Q$ `` ### Schema: '' S_{context} `` ### SQL''
3) Load base model and adapter: Load `SQLCoder-8B` in 4-bit mode and attach the `CenQuery-LoRA` adapter.
4) Generate SQL tokens with deterministic decoding: $\text{Tokens}_{\text{out}} \leftarrow \texttt{Model.generate}(\text{Prompt}, temperature = 0)$
5) Decode and sanitize output: $Y \leftarrow \texttt{RemoveMarkdownFormatting}(\texttt{Decode}(\text{Tokens}_{\text{out}}))$

F. End-to-End Implementation Pipeline

To operationalize the fine-tuned model, we developed a three-tier architecture ensuring secure, offline execution:

- **Orchestration Layer (FastAPI):** A Python-based FastAPI backend serves as the central orchestrator. It implements strict validation middleware to sanitize natural language inputs before they reach the inference engine.

TABLE I
BASELINE MODEL PERFORMANCE ON SPIDER BENCHMARK

Model Provider	Model Name	Avg. Latency (s)	Exact Match (%)
Groq	Llama 3.3 70B Versatile	1.00	69.91
Groq	Llama 3.1 8B Instant	1.07	69.03
Groq	OpenAI GPT-OSS 120B	2.01	60.09
Groq	OpenAI GPT-OSS 20B	1.44	57.49
Hugging Face	Llama 3 SQLCoder 8B	2.81	52.28
Hugging Face	SQLCoder 7B-2	3.26	49.51

TABLE II
BASELINE MODEL PERFORMANCE ON WIKISQL BENCHMARK

Model Provider	Model Name	Avg. Latency (s)	Exact Match (%)
Groq	Llama 3.3 70B Versatile	2.25	84.24
Groq	Llama 3.1 8B Instant	2.63	82.96
Groq	OpenAI GPT-OSS 120B	2.95	75.51
Groq	OpenAI GPT-OSS 20B	2.46	73.71
Hugging Face	Llama 3 SQLCoder 8B	2.64	71.79
Hugging Face	SQLCoder 7B-2	3.07	65.38

The backend manages session state and enforces rate limiting to prevent resource exhaustion on local hardware.

- **Inference Engine Integration:** The fine-tuned Llama-3-SQLCoder-8B model is loaded using the `bitsandbytes` library for 4-bit quantization, reducing memory footprint to under 8 GB VRAM. We utilize a custom inference endpoint that loads the LoRA adapters dynamically at runtime, allowing the base model to switch between different schema contexts without reloading full weights.
- **Secure Execution Environment:** Generated SQL queries are executed against a PostgreSQL database configured with strictly read-only permissions. The execution layer wraps all database interactions in transaction blocks with a 5-second timeout to prevent long-running malformed queries from degrading system performance. This human-in-the-loop design allows the frontend (Next.js) to display the generated SQL for user verification before data retrieval.

V. BASELINE ANALYSIS AND DOMAIN GAP

A. Performance on Standard Benchmarks

To quantify the need for a specialized fine-tuned model, we evaluated five state-of-the-art Large Language Models (LLMs) on standard datasets (Spider and WikiSQL). The objective was to strictly quantify the performance gap between privacy-preserving local models and cloud-based commercial models.

Analysis of Results: As observed in Tables I and II, the cloud-based Llama-3.3-70B demonstrates the highest baseline performance with 69.91% accuracy on Spider. However, utilizing such models requires transmitting sensitive schema data to external cloud providers, which violates strict data privacy protocols for census data.

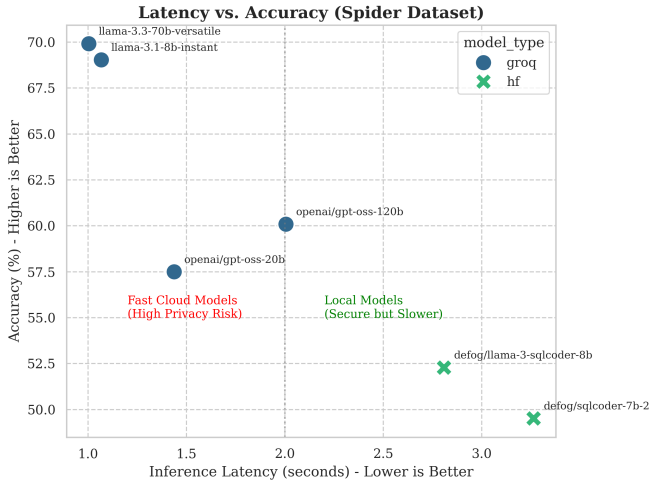

Fig. 2. Latency vs. Accuracy Trade-off Across Evaluated Models

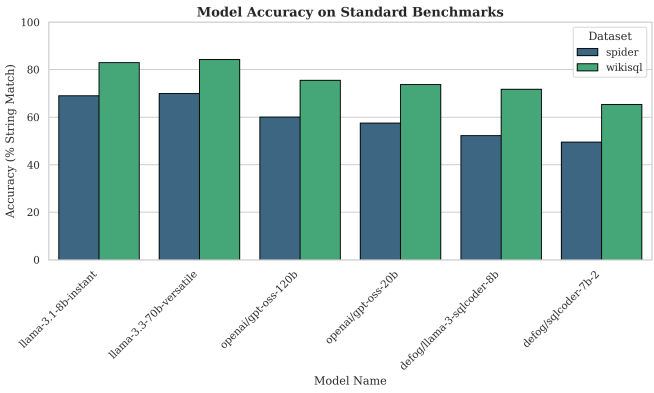

Fig. 3. Exact Match Accuracy on Spider and WikiSQL Benchmarks

In contrast, the locally hosted SQLCoder-8B—selected as the base for CenQuery—achieves only 52.28% accuracy. This reveals a **˜17% accuracy gap** that must be bridged. This limitation is critical because census data requires the strict data sovereignty that only local models can provide, yet their performance on standard benchmarks is insufficient for production use.

Fig. 2 presents the Latency vs. Accuracy trade-off. The scatter plot highlights a distinct clustering: cloud-based models achieve higher accuracy with ultra-low latency but pose privacy risks, while local models are secure but slower and less accurate.

Fig. 3 provides a breakdown of Exact Match (EM) accuracy. The drop in performance for the local SQLCoder-8B is more pronounced on the complex Spider dataset (52.28%) compared to the simpler WikiSQL dataset (71.79%), indicating difficulty in handling multi-table joins.

B. The Census Domain Gap

While standard benchmarks like Spider cover general SQL operations, they fail to capture the specific nuances of the Indian Census. Through qualitative analysis of the schema and initial model outputs, we identified specific complexities that general-purpose models are ill-equipped to handle:

- **Hierarchical Geography:** Census queries often require deep joins across normalized geographic tables (e.g., linking `districts`, `tehsils`, and `states`). Standard models frequently hallucinate join keys or miss intermediate tables.
- **Administrative Jargon:** The schema contains domain-specific terminology such as "Scheduled Tribes (SC/ST)" or distinctions between "Main Workers" and "Marginal Workers," which generic models fail to map to the correct columns (e.g., `main_work_p` vs `marg_work_p`).
- **Privacy Requirements:** The strict requirement for offline inference precludes the use of powerful cloud models (like GPT-4), forcing reliance on smaller, less capable local models which (as shown in Fig. 2) currently exhibit the lowest accuracy.

C. Motivation for Census-650

This baseline analysis confirms that existing off-the-shelf local models are insufficient for the Census domain. To bridge this gap, we constructed the **Census-650** dataset (detailed in Section IV) to serve as the ground truth for fine-tuning privacy-preserving models in future work.

VI. Conclusion and Future Scope

This paper addressed the critical challenge of democratizing access to Indian Census data through secure, natural language interfaces. Our baseline analysis of state-of-the-art Large Language Models (LLMs) revealed a significant trade-off: while cloud-based models like Llama-3-70B achieve reasonable accuracy (69.9%) on standard benchmarks, they are unsuitable for privacy-sensitive government data. Conversely, privacy-preserving local models (SQLCoder-8B) exhibit a 17% performance gap, failing to handle the hierarchical joins and specific terminology inherent to the census domain.

To bridge this gap, we contributed **Census-650**, a rigorously validated dataset of domain-specific SQL-NL pairs. By isolating failure modes such as schema hallucination and join misalignment, this work establishes the necessary data foundation for building specialized government querying systems. We further proposed the **CenQuery architecture**, a fine-tuning roadmap designed to leverage this dataset for adapting local models.

Future Work: Our immediate next step is to execute the full parameter-efficient fine-tuning (LoRA) of the Llama-3-SQLCoder-8B model using the Census-650 dataset. We hypothesize that this domain adaptation will recover the performance deficit observed in our baseline analysis. Additionally, we plan to expand the system to support multilingual queries in Hindi and Marathi, further enhancing accessibility for regional policymakers.

VII. Acknowledgment

The authors would like to thank the open-source research community for providing publicly available benchmarks, tooling, and foundational models that made this work possible. We also acknowledge the developers of the SQLCoder and LoRA frameworks for enabling efficient experimentation with large language models under limited computational resources. No external funding was received for this research.

References

[1] T. Yu, J. Eisner, and B. Van Durme, "On repairing lousy NL-to-SQL queries with pre-trained language models," *arXiv preprint arXiv:2305.*, 2023.

[2] A.-M. Popescu, O. Etzioni, and H. Kautz, "Towards a theory of natural language interfaces to databases," *Proceedings of the International Conference on Intelligent User Interfaces*, 2003.

[3] V. Zhong, C. Xiong, and R. Socher, "Semantic parsing for text-to-sql with schema linking," *Proceedings of ACL*, 2020.

[4] J. Guo, Z. Zhan, Y. Gao, Y. Xiao, J.-G. Lou, and T. Liu, "Towards complex text-to-sql in cross-domain database with intermediate representation," *Proceedings of ACL*, 2019.

[5] I. Raji *et al.*, "A maturity model for operationalizing large language models," *arXiv preprint arXiv:2303.*, 2023.

[6] Transaction Processing Performance Council, "Tpc-ds benchmark specification," *TPC Benchmark Documentation*, 2022.

[7] Z. Chen *et al.*, "Evaluating text-to-sql systems for software engineering," *Proceedings of ICSE*, 2021.

[8] A. Johnson *et al.*, "Natural language querying of clinical databases," *Proceedings of AMIA*, 2020.

[9] R. Zhang and T. Yu, "A survey on text-to-sql parsing," *ACM Computing Surveys*, 2021.

[10] H. Li *et al.*, "Bird: A large-scale benchmark for text-to-sql on real databases," *Proceedings of NeurIPS*, 2023.

[11] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.