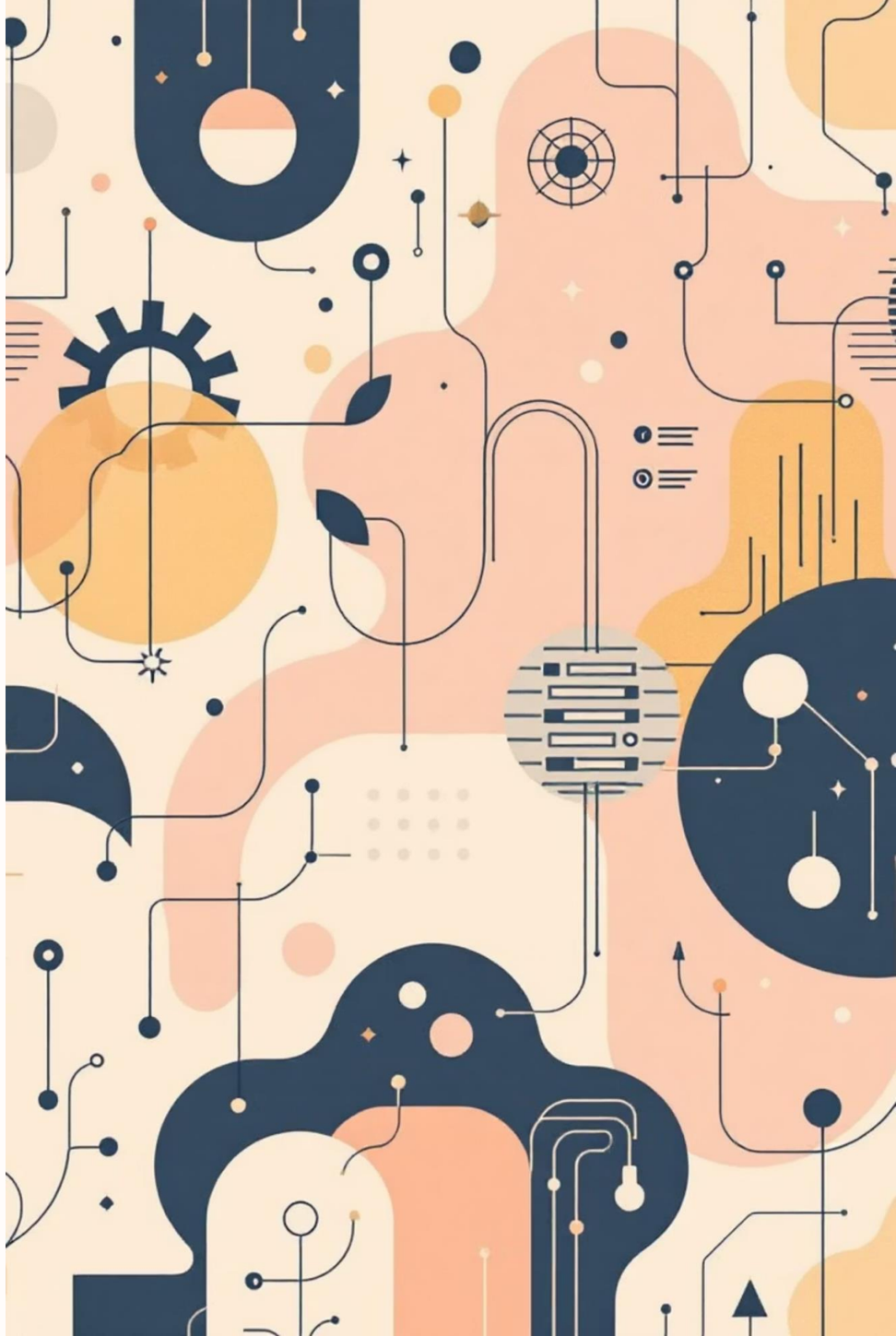
SQL Copilot: Natural Language to SQL using Gemini

DATS 6312 — Applied Natural Language Processing

The George Washington University

Prof. Ning Rui

Shaik Mohammad Mujahid Khalandar • Ayush Meshram



Introduction: Bridging Natural Language and Databases

What is Natural Language to SQL?

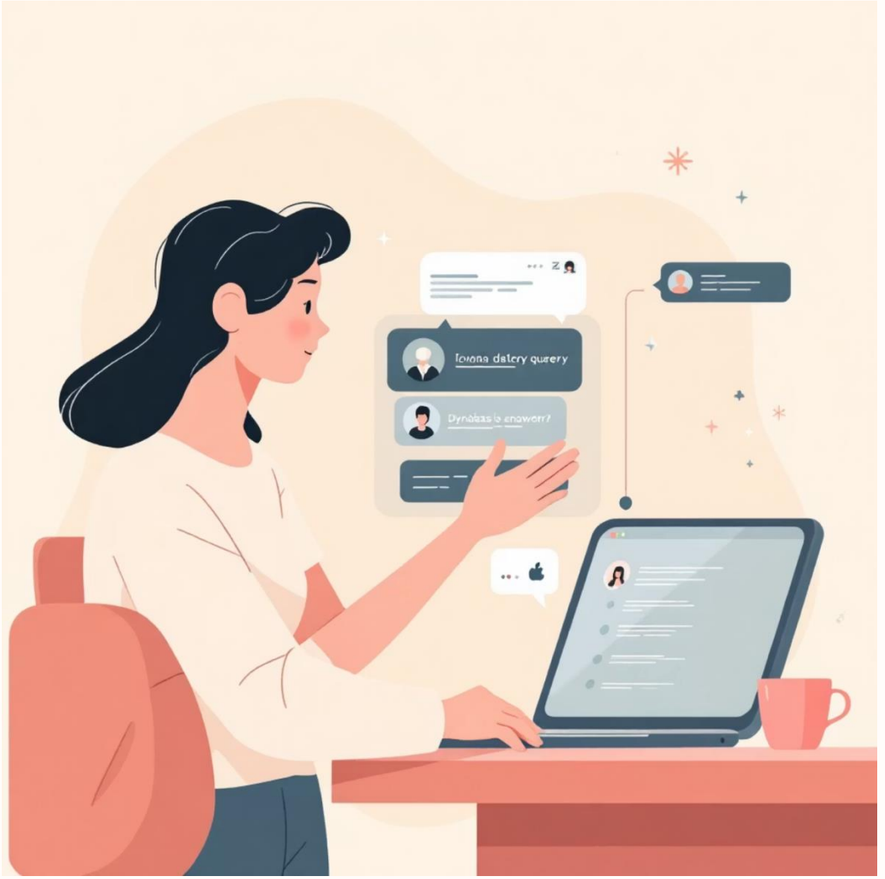
Transforming everyday questions into structured database queries enables non-technical users to extract insights without learning SQL syntax. This democratizes data access across organizations.

Why SQL Copilot?

Traditional business intelligence tools require training and technical expertise. SQL Copilot removes this barrier by interpreting user intent and generating accurate SQL queries instantly.

Our Contribution

We leverage Google Gemini's advanced language understanding to create an accessible, accurate natural language interface for fossil fuel energy data analysis.



Dataset Description

Dataset Name

fossil_trend_countries — annual fossil-energy share by country

Key Features

Country, Year, Fossil Fuel Share (%), Energy Consumption, Carbon Emissions

Data Scale

Multi-year temporal data spanning multiple countries with detailed energy metrics

This structured environmental dataset enables complex analytical queries about global energy trends, making it ideal for testing natural language SQL generation capabilities.

NLP Model Architecture

Google Gemini: Advanced Language Understanding

Natural Language Input

User passes question in plain English

Gemini Processing

LLM translates to SQL with schema context

Query Execution

SQLite processes the generated query

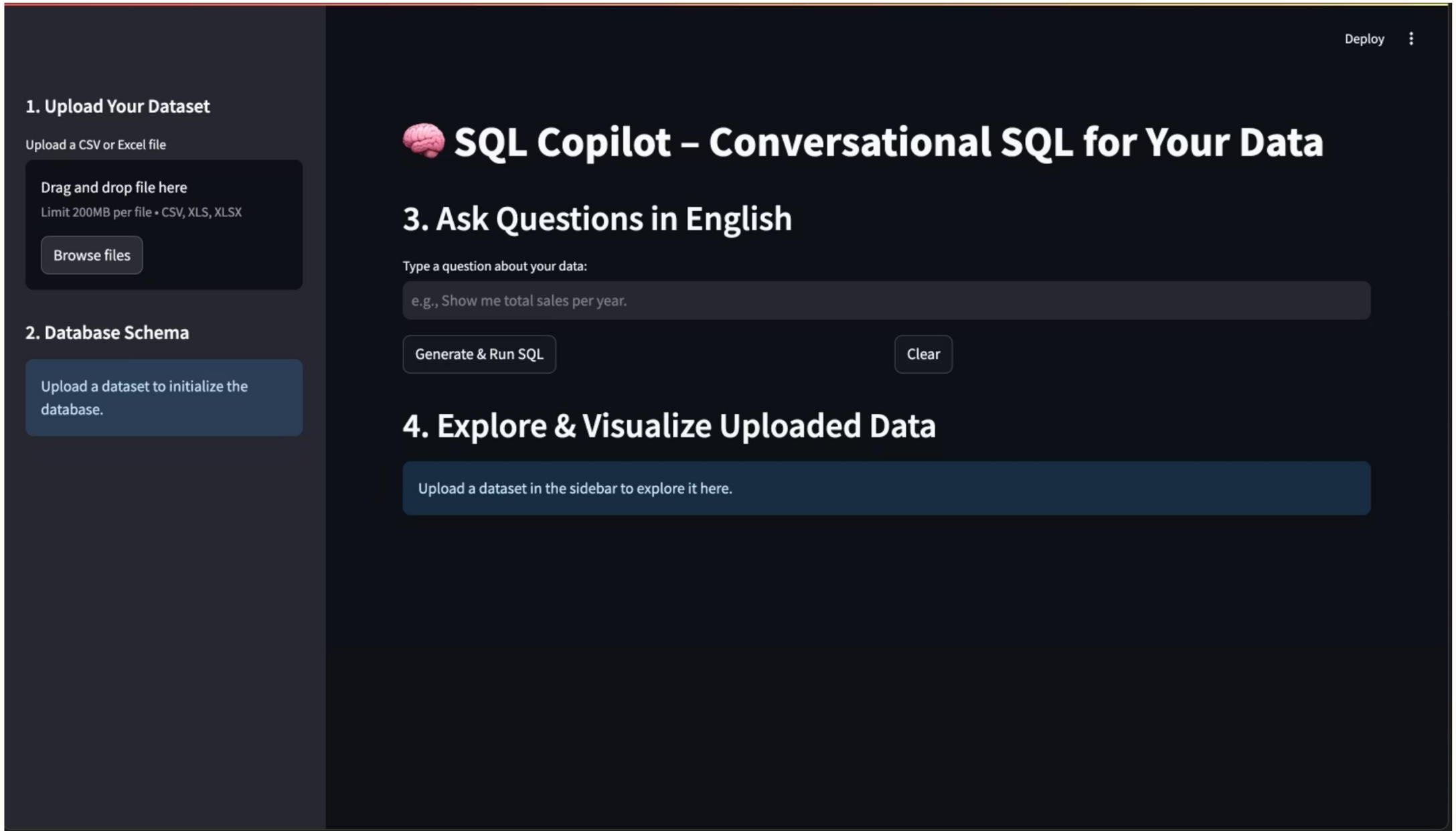
Results Display

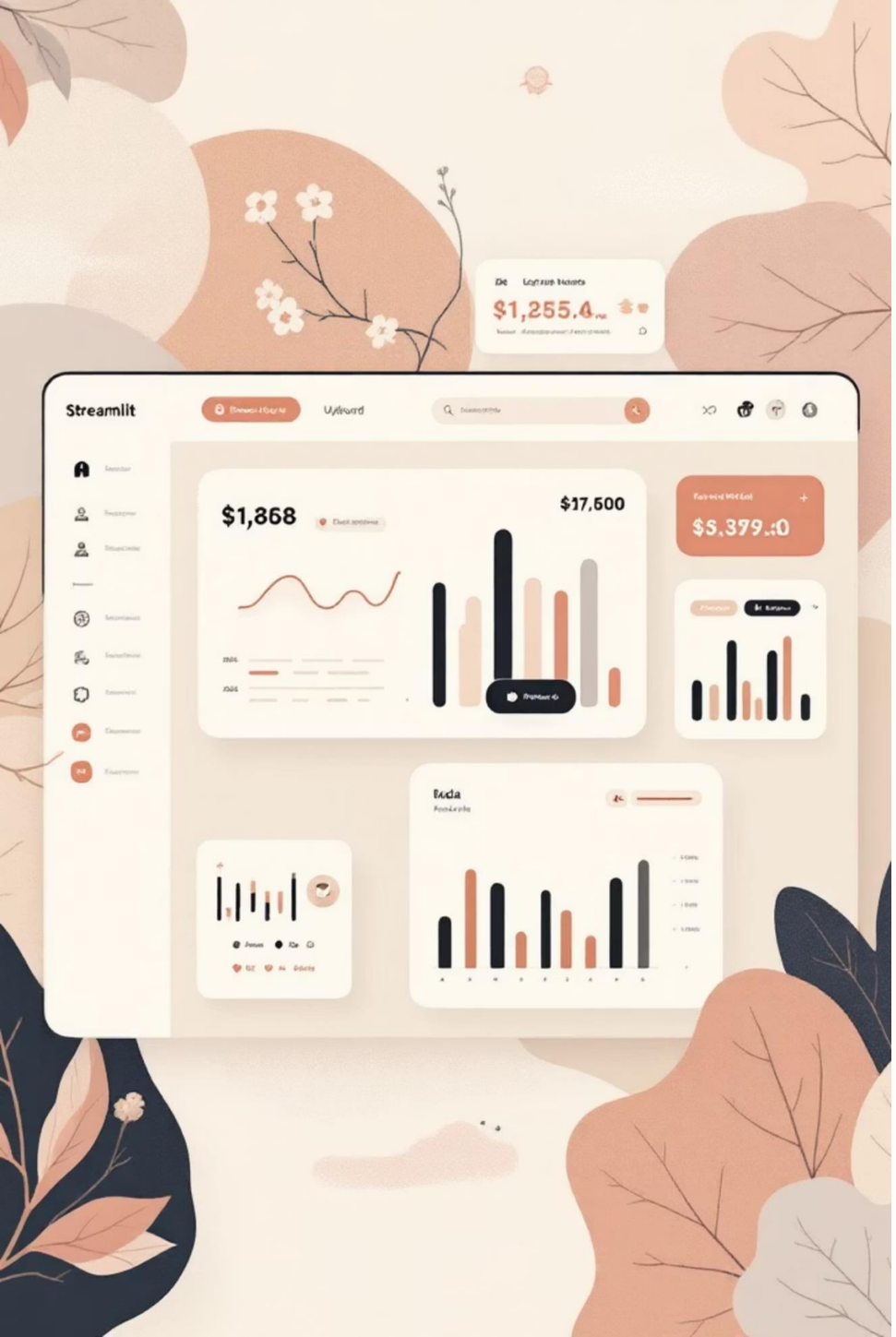
Structured output with visualizations

The system maintains schema awareness throughout the pipeline, ensuring generated queries are valid and semantically aligned with the underlying database structure.

System Interface in Action

How It Looks





Experimental Setup: Streamlit Application Workflow

01

Data Upload

Users upload CSV files containing fossil fuel energy data

02

Natural Language Query

Enter questions in plain English about the dataset

03

SQL Execution

Gemini generates and executes validated SELECT queries

04

Visual Results

Interactive tables and charts display query results

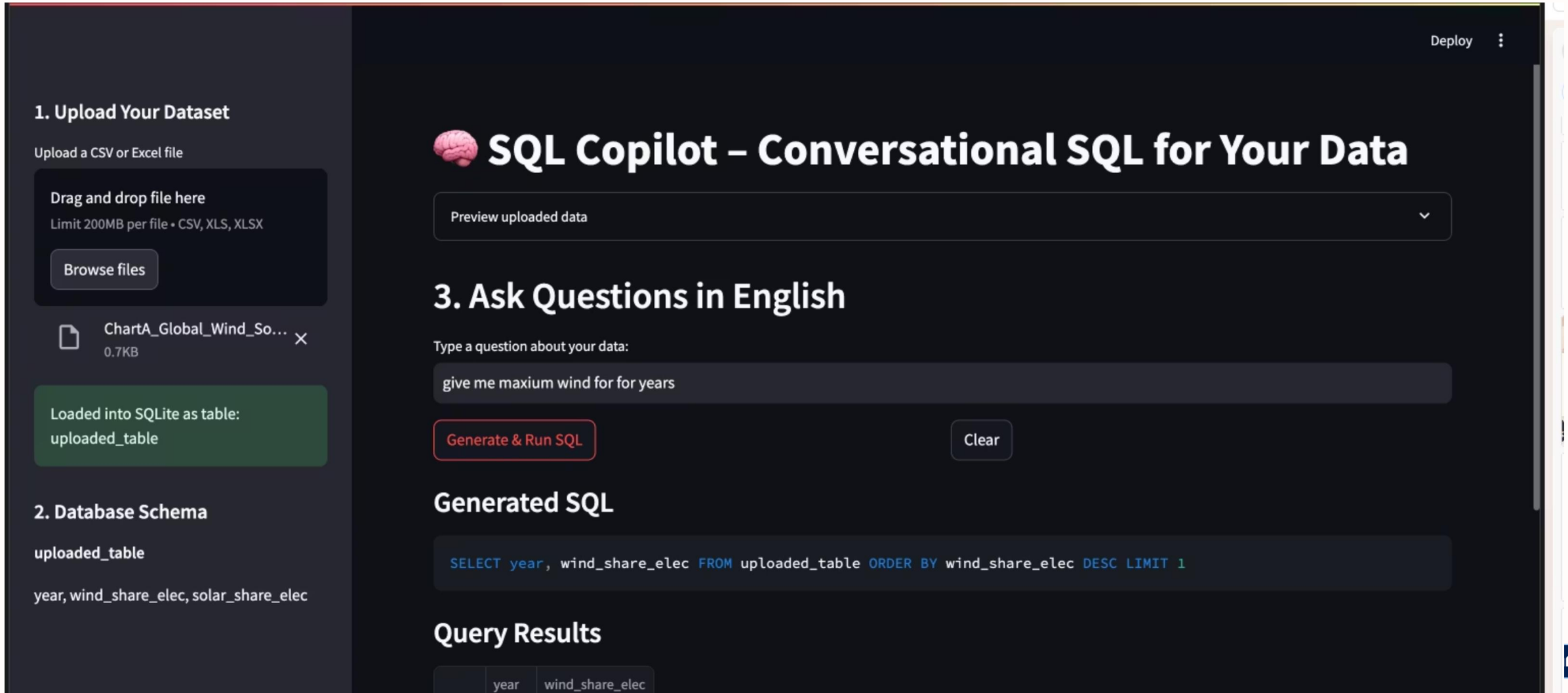
Evaluation Methodology

Query correctness assessed using Execution Success (does the SQL run) and Result-Set Accuracy (do the results match the ground-truth output), rather than exact string-level SQL matching.

Streamlit Application Workflow

The Complete Running Setup

This screenshot demonstrates the complete end-to-end workflow of the SQL Copilot application, showing all key components in action: dataset upload, schema integration, natural language query input, SQL generation, and interactive results display.



Prompt Design & Safety Guardrails

Model Configuration

Fixed LLM Configuration

> We treat **Gemini 2.5 Flash** as a frozen API model – there is **no fine-tuning or training**, which avoids overfitting to our small NL→SQL benchmark and keeps behavior stable across datasets.

Schema-Guided Prompting

> Before each query, the backend calls SQLite `PRAGMA` commands to extract the table and column names. This schema string is injected into the Gemini prompt so generated SQL is **grounded in the actual table structure** instead of hallucinated columns.

Synonym Normalization

> A lightweight synonym map rewrites user phrases (e.g., *"emissions"* → `co2_emissions`) before prompting. This reduces ambiguity and helps the model consistently map natural-language terms to the correct columns.

Validation Layer

Results & Performance Metrics

100%

Execution Success Rate

Queries executed without errors

90%

SQL Accuracy

Semantic match with ground truth

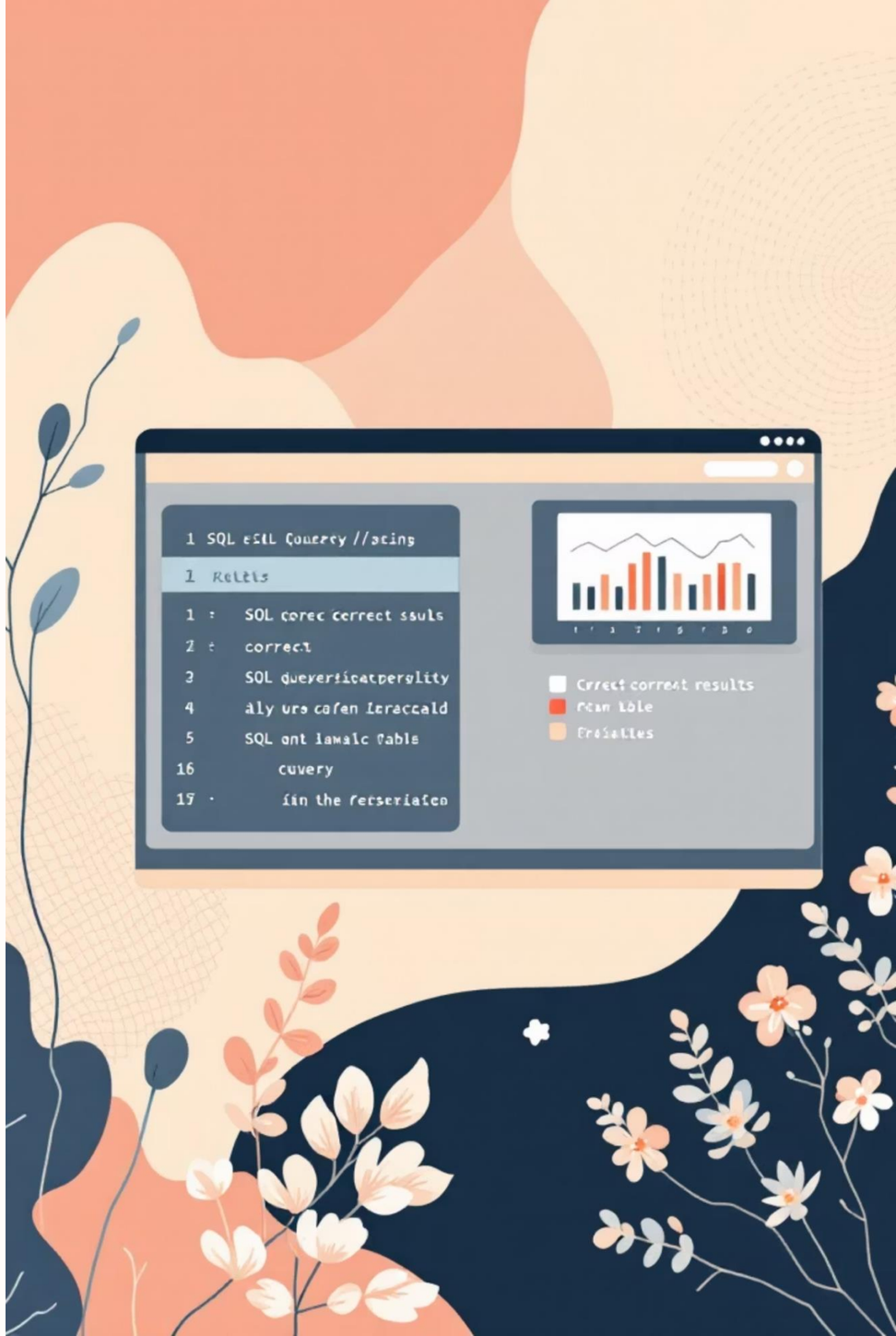
0.82s

Average Latency

End-to-end query generation time

Key Insights

- **High accuracy on temporal queries:** Year-based filtering and aggregations perform exceptionally well
- **Strong country-specific analysis:** Geographic queries demonstrate robust semantic understanding
- **Complex aggregations:** Successfully handles AVG, SUM, MAX calculations with appropriate GROUP BY clauses



Ablation Study: Design Choices Impact

To quantify the effect of our design choices, we ran a small ablation study. It compares variants with and without schema awareness, synonym normalization, and safety checks. Schema guidance provides the largest improvement in execution and result quality, while synonyms offer a smaller but consistent boost. Safety checks do not directly change ExecSucc but ensure that all generated queries are valid and bounded.

Variant	ExecSucc	ResultAcc	Notes
No schema, no synonyms	0.80	0.62	Basic prompting; several queries misaligned
Schema-aware prompt	0.90	0.78	Schema guidance improves table/column grounding
Schema + synonyms (final)	1.00	0.90	Best performance: all queries execute successfully

Error Analysis & System Limitations

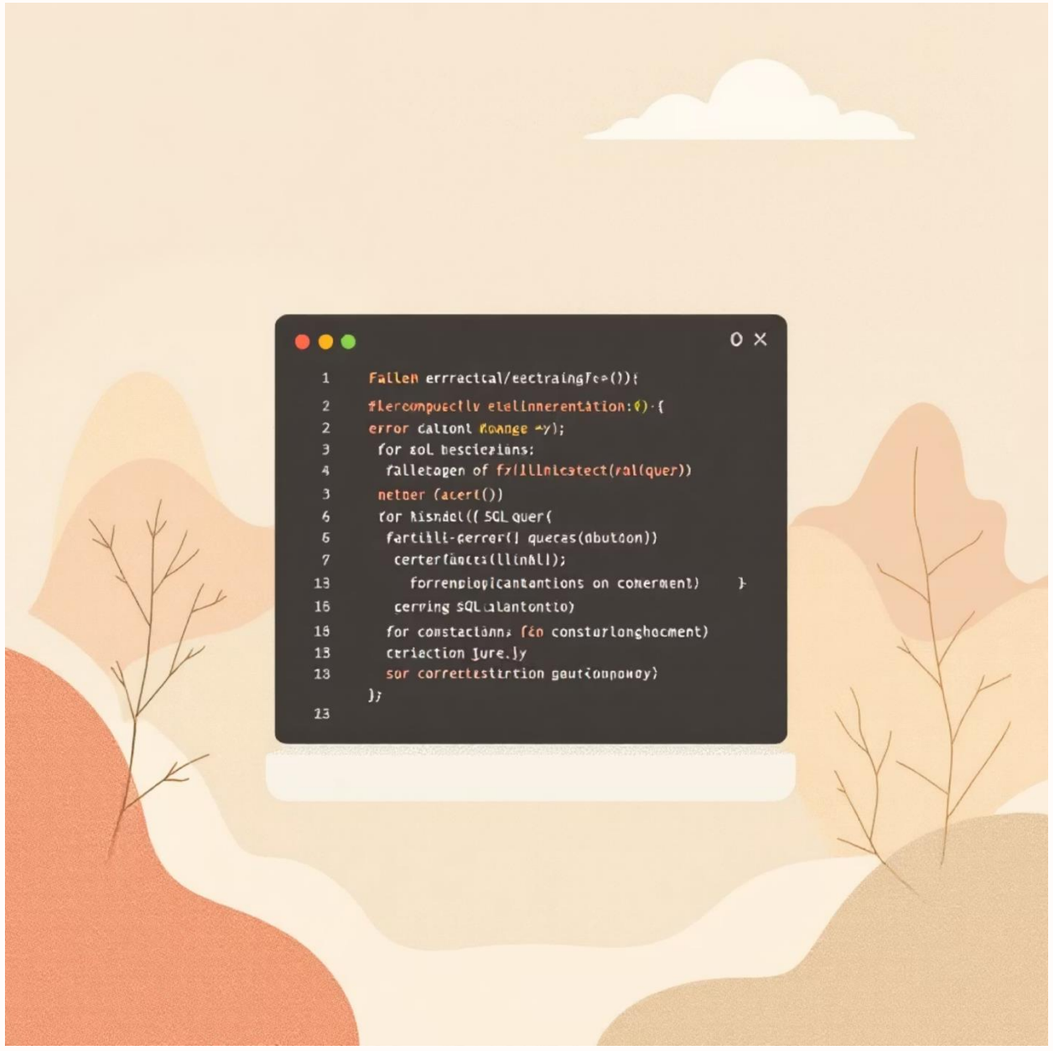
Common Failure Patterns

Multi-Step Reasoning Failures

The model performs well on simple aggregations, but breaks on queries that need *temporal or sequential reasoning* (e.g., “years where fossil share increased every year”), where it fails to encode the monotonic trend correctly.

Single-Table Schema Limitation

The current system only supports a *single table* and cannot answer questions that require joins or relationships across multiple tables, which limits how complex the queries can be.



Mitigation Strategies: Add few-shot examples and better prompt templates for temporal and multi-step reasoning.

Summary & Future Directions

Key Achievements

Democratized Data Access

Non-technical users can now query complex energy datasets using natural language

Reliable NL→SQL Performance

Achieved **100% execution success** and **90% result accuracy** on a 50-question benchmark, with about **0.8s** end-to-end latency per query.

Production-Ready Safety

Read-only query restrictions and validation layers ensure secure deployment

Lessons Learned

Explicit schema-aware prompting plus a read-only safety wrapper (forbidden-keyword filter + automatic LIMIT) were essential to keep Gemini from hallucinating columns and to reach 100% execution success and 90% result accuracy on our NL→SQL benchmark.

Future Enhancements

- **Multi-table JOIN support:** Expand to complex relational schemas
- **Multilingual capabilities:** Support non-English natural language inputs
- **Query optimization:** Performance tuning for large-scale datasets

Conclusion

Key Takeaways

Our natural language to SQL translation system demonstrates the viability of LLM-powered database interfaces, achieving strong performance on standard queries while identifying clear paths for improvement.

By democratizing database access, this technology has the potential to transform how organizations leverage their data assets, enabling broader participation in data-driven decision making.

Impact & Applications

This work contributes to the growing field of natural language data interfaces and provides a foundation for future research in semantic query understanding and generation.