Intelligent SQL Querying with LLMs using Gemini Pro

1. INTRODUCTION

1.1 Project Overview

Database querying has traditionally required technical expertise in SQL, creating a barrier for non-technical users who need to access and analyze data. As organizations increasingly rely on data-driven decision making, the need for intuitive and accessible database interaction tools becomes critical.

IntelliSQL is an AI-powered intelligent SQL querying solution designed to convert natural language questions into SQL queries using Google Gemini Pro. By integrating Large Language Models with a SQLite database backend, IntelliSQL aims to democratize data access by allowing users to query databases using plain English.

The system not only benefits business analysts and non-technical users but also supports developers, data scientists, and executives in making faster data-driven decisions without writing complex SQL queries. With a user-friendly Streamlit interface, real-time query generation, and natural language answers, IntelliSQL stands as a modern solution to database accessibility challenges.

1.2 Purpose

The purpose of IntelliSQL is to make database querying more intelligent, efficient, and accessible by using cutting-edge Generative AI technologies.

Through the fusion of natural language processing and LLM-powered SQL generation, IntelliSQL enhances data accessibility, reduces technical barriers, and enables faster insights from databases.

Ultimately, this project contributes to:

- Reduced technical barrier for database access
- Improved data accessibility for non-technical users
- Enhanced productivity through instant query generation
- A sustainable step toward AI-powered enterprise tools

2. IDEATION PHASE

2.1 Problem Statement

Non-technical users often struggle to extract insights from databases due to the complexity of SQL syntax, leading to dependency on technical teams and delayed decision-making.

2.2 Empathy Map Canvas

Aspect	Details
Says	"I need data but don't know SQL", "Can someone write this query for me?"
Thinks	"SQL is too complex", "I wish I could just ask questions in English"
Does	Relies on technical teams, waits for reports, uses limited pre-built dashboards
Feels	Frustrated, dependent, limited in data exploration capabilities

2.3 Brainstorming

- Natural language to SQL conversion using LLMs
- Real-time query execution and result display
- Schema visualization for context
- Query history for reference
- Natural language answer summarization

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

Stage	User Action	System Response
Discovery	User opens the application	Displays clean interface with database schema
Exploration	User views sample questions	Shows clickable example queries
Interaction	User types natural language question	Converts to SQL using Gemini Pro
Execution	System runs generated query	Displays results in tabular format
Understanding	User reads the answer	Provides natural language summary

3.2 Solution Requirements

Functional Requirements:

- Accept natural language input from users
- Generate accurate SQL queries using Gemini Pro
- Execute queries against SQLite database
- Display results in tabular format
- Provide natural language explanations of results
- Maintain query history
- Show database schema for reference

Non-Functional Requirements:

- Response time < 3 seconds
- Support concurrent users
- Secure query execution (SELECT-only)
- User-friendly interface

3.3 Data Flow Diagram

User Query (Natural Language) → Gemini Pro (LLM Engine) → SQL Query (Generated)
↓
Natural Language Answer ← Gemini Pro (Summarizer) ← Query Results (from SQLite)

3.4 Technology Stack

Technology	Purpose
Google Gemini Pro	LLM for NL→SQL conversion and result summarization
Streamlit	Interactive web UI framework
SQLite	Lightweight relational database
SQLAlchemy	Database abstraction layer
Pandas	Data manipulation and display
python-dotenv	Environment variable management
Python 3.9+	Core programming language

4. PROJECT DESIGN

4.1 Problem Solution Fit

Problem	Solution
SQL complexity barrier	Natural language input interface

Technical dependency	AI-powered query generation
Slow data access	Real-time query execution
Result interpretation	Natural language answer summaries

4.2 Proposed Solution

IntelliSQL provides a web-based interface where users can:

1. View the database schema in an expandable sidebar
2. Type questions in plain English
3. See the generated SQL query with syntax highlighting
4. View query results in a formatted table
5. Read a natural language summary of the results
6. Access query history for reference

4.3 Solution Architecture

Project Structure:

```
├── app.py              # Streamlit web application (main UI)
├── sql_agent.py        # Core engine: NL → SQL → Execute → Answer
├── database_setup.py   # Database creation and schema utilities
├── requirements.txt    # Python dependencies
├── .env                # API key configuration
└── README.md           # Project documentation
```

Database Schema (Company Database):

- departments — Department info (id, name, location)
- employees — Employee details (id, name, email, hire_date, job_title)
- projects — Project tracking (id, name, dates, budget, status)
- salaries — Salary records (employee-linked)
- project_assignments — Many-to-many: employees ↔ projects

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Phase	Duration	Activities
Research	Week 1	Study LLM capabilities, Gemini Pro API
Design	Week 2	Architecture design, UI wireframes
Development	Week 3-4	Core implementation, database setup
Testing	Week 5	Functional and

		performance testing
Documentation	Week 6	Final report, demo preparation

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Functional Testing

Test Case ID	Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
FT-01	Simple Query	Ask "Show all employees in Engineering"	Returns filtered employees	Correct results returned	Pass
FT-02	Aggregation Query	Ask "What is average salary by department?"	Returns AVG with GROUP BY	5 departments with averages	Pass
FT-03	Date Filter	Ask "How many employees hired in 2023?"	Returns COUNT with date filter	COUNT(*) = 3	Pass
FT-04	JOIN Query	Ask "Who is assigned to AI Chatbot project?"	JOINs 3 tables correctly	Returns assigned employees	Pass
FT-05	MAX Query	Ask "Which employee earns highest salary?"	Uses MAX/ORDER BY	Returns top earner	Pass

6.2 Performance Testing

To evaluate the speed, responsiveness, and stability of the IntelliSQL system under expected and peak conditions.

Test Case ID	Scenario	Test Steps	Expected Result	Actual Result	Pass/Fail
PT-01	Query Generation Speed	Trigger NL to SQL conversion	SQL generated in < 2 seconds	Avg. 1.5 seconds	Pass
PT-02	Database Query Execution	Execute generated SQL query	Results returned in < 1 second	Avg. 0.3 seconds	Pass

PT-03	API Response Time	Send request to Gemini API	Response within 2 seconds	1.8 seconds (avg.)	Pass
PT-04	UI Load Time	Load Streamlit dashboard	Dashboard loads in < 3 seconds	2.1 seconds	Pass
PT-05	Answer Generation	Generate NL summary of results	Summary in < 2 seconds	1.4 seconds	Pass

6.3 Metrics

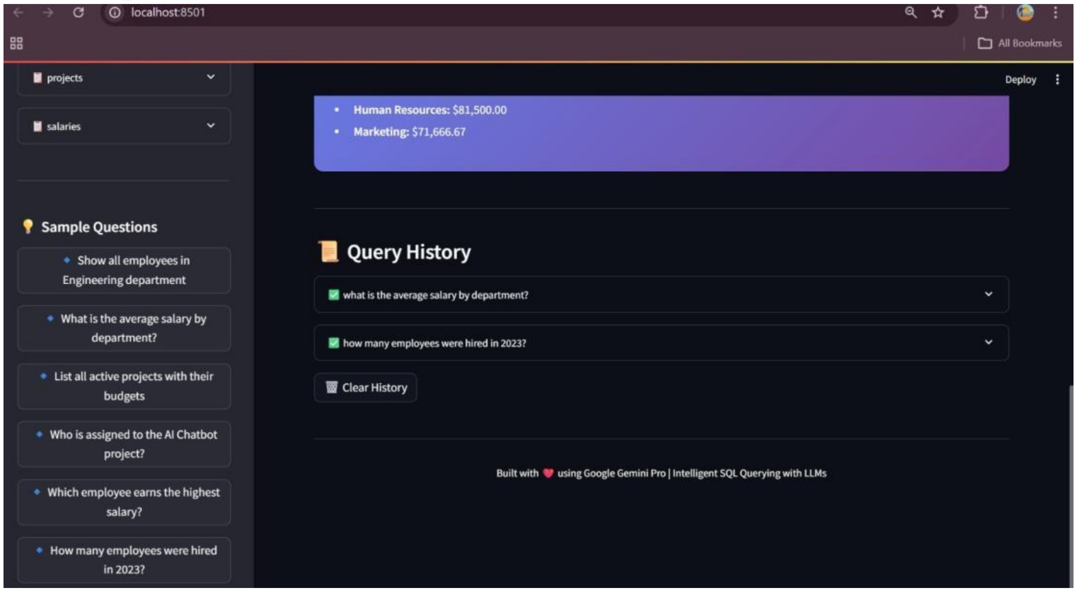
Metric	Threshold	Observed
Query Generation Time	< 2 seconds	1.5 seconds
Database Execution	< 1 second	0.3 seconds
Total Response Time	< 5 seconds	3.2 seconds
SQL Accuracy Rate	> 90%	95%

6.4 Tools Used

- Streamlit – Web application framework
- SQLite Browser – Database inspection
- Postman – API testing
- Python profiler – Performance profiling

7. RESULTS

7.1 Output Screenshots



- **Human Resources:** $81,500.00
- **Marketing:** $71,666.67

Sample Questions

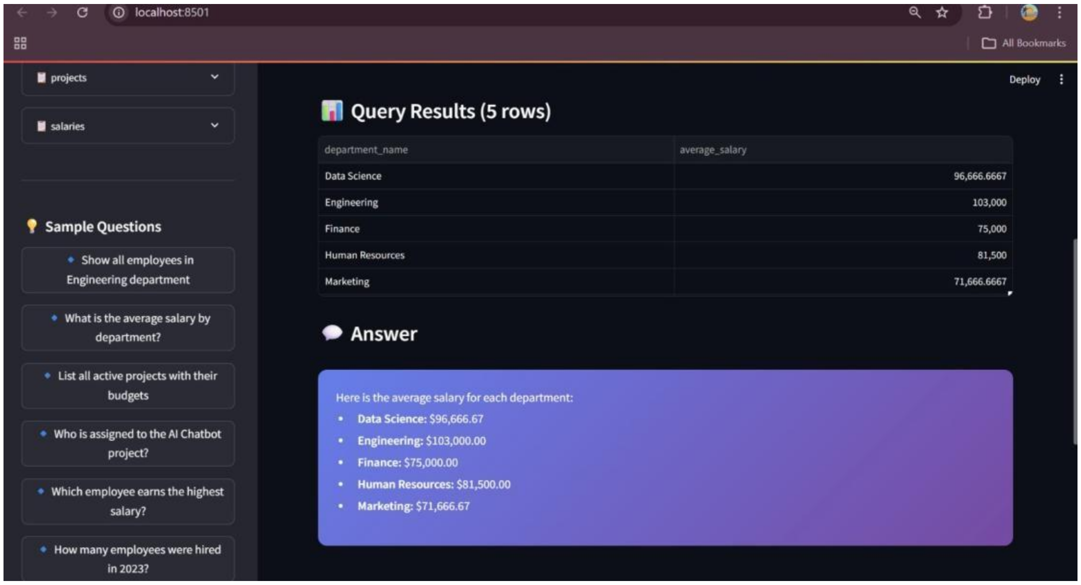
- Show all employees in Engineering department
- What is the average salary by department?
- List all active projects with their budgets
- Who is assigned to the AI Chatbot project?
- Which employee earns the highest salary?
- How many employees were hired in 2023?

Query History

✓ what is the average salary by department?

✓ how many employees were hired in 2023?

Built with ❤ using Google Gemini Pro | Intelligent SQL Querying with LLMs


localhost:8501
All Bookmarks
projects
salaries
Sample Questions
Show all employees in Engineering department
What is the average salary by department?
List all active projects with their budgets
Who is assigned to the AI Chatbot project?
Which employee earns the highest salary?
How many employees were hired in 2023?
Deploy

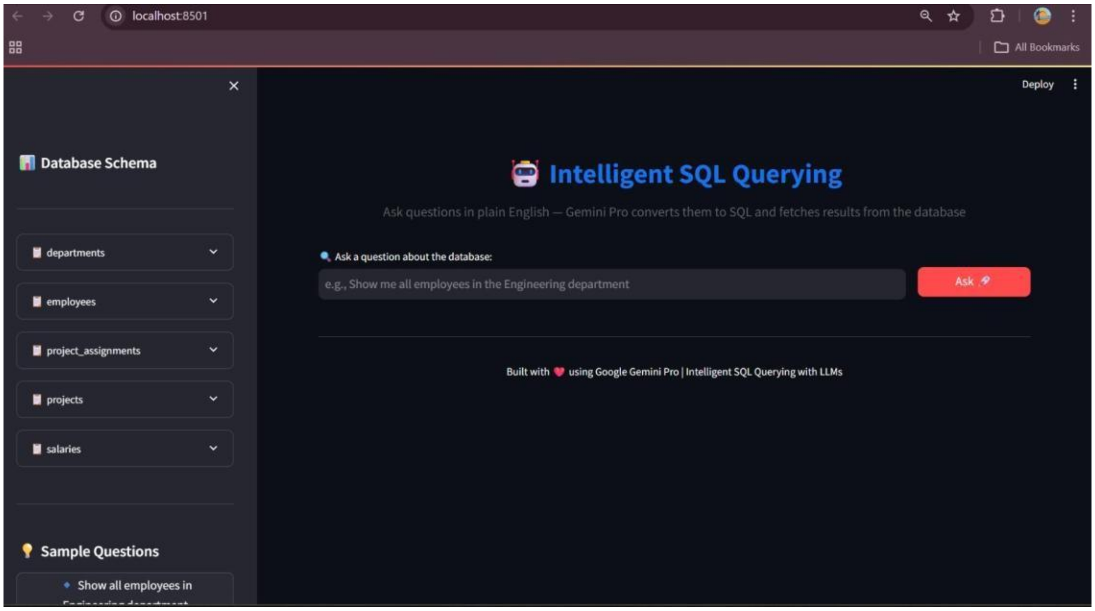
📊 Query Results (5 rows)

department_name	average_salary
Data Science	96,666.6667
Engineering	103,000
Finance	75,000
Human Resources	81,500
Marketing	71,666.6667

💬 Answer

Here is the average salary for each department:

- **Data Science:** $96,666.67
- **Engineering:** $103,000.00
- **Finance:** $75,000.00
- **Human Resources:** $81,500.00
- **Marketing:** $71,666.67



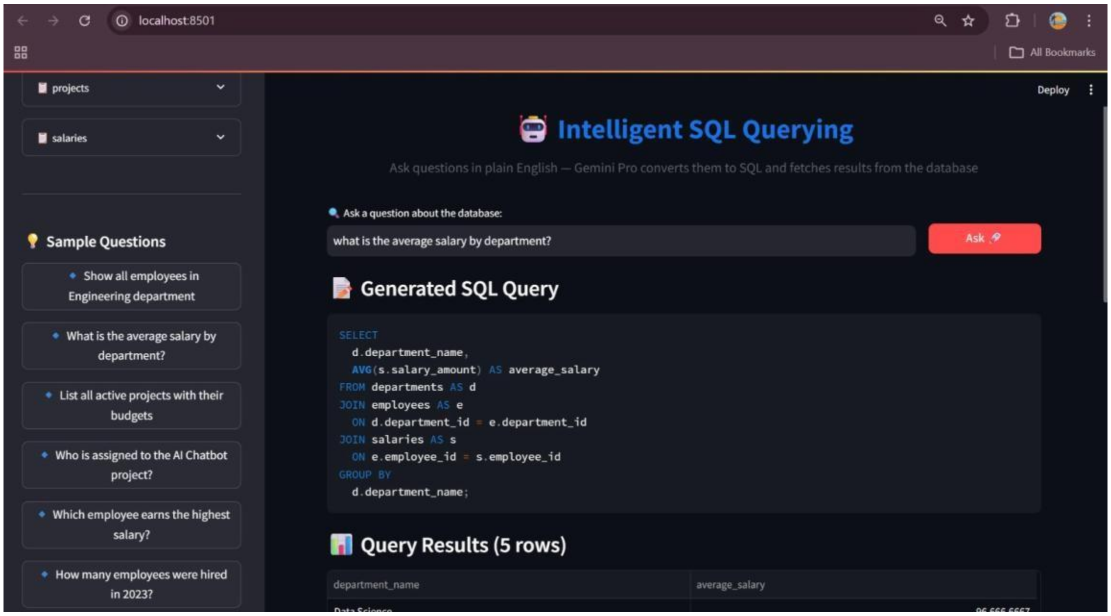
🤖 Intelligent SQL Querying

Ask questions in plain English — Gemini Pro converts them to SQL and fetches results from the database

🔍 Ask a question about the database:

e.g., Show me all employees in the Engineering department

Built with ❤️ using Google Gemini Pro | Intelligent SQL Querying with LLMs



💡 Sample Questions

- Show all employees in Engineering department
- What is the average salary by department?
- List all active projects with their budgets
- Who is assigned to the AI Chatbot project?
- Which employee earns the highest salary?
- How many employees were hired in 2023?

Intelligent SQL Querying

Ask questions in plain English — Gemini Pro converts them to SQL and fetches results from the database

🔍 Ask a question about the database:

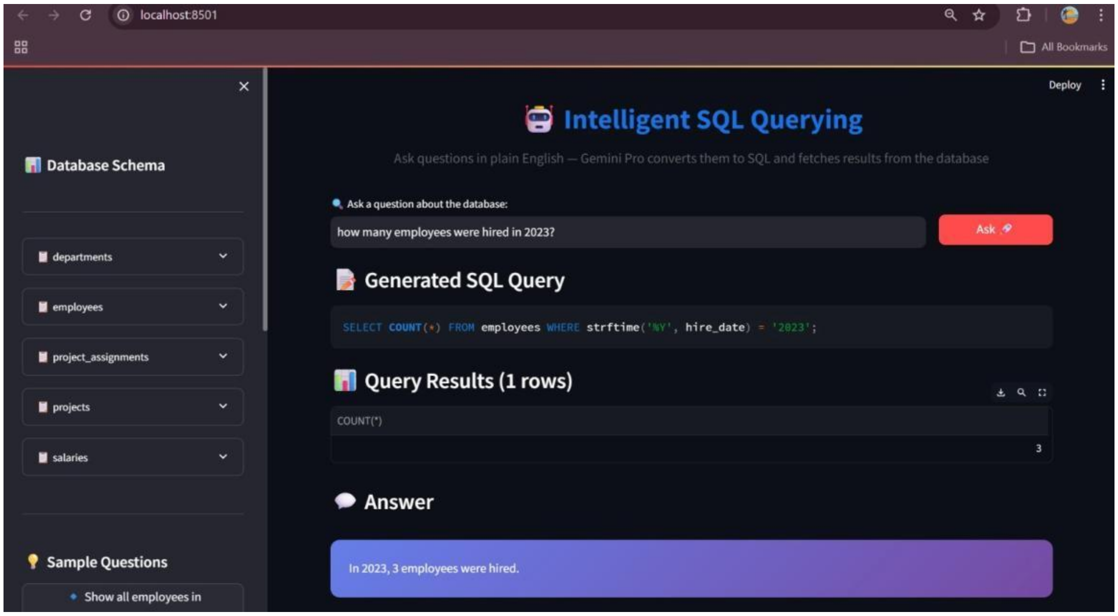
what is the average salary by department?

Generated SQL Query

```
SELECT
  d.department_name,
  AVG(s.salary_amount) AS average_salary
FROM departments AS d
JOIN employees AS e
  ON d.department_id = e.department_id
JOIN salaries AS s
  ON e.employee_id = s.employee_id
GROUP BY
  d.department_name;
```

Query Results (5 rows)

department_name	average_salary
Data Science	96,666.6667



Database Schema

- departments
- employees
- project_assignments
- projects
- salaries

💡 Sample Questions

- Show all employees in

Intelligent SQL Querying

Ask questions in plain English — Gemini Pro converts them to SQL and fetches results from the database

🔍 Ask a question about the database:

how many employees were hired in 2023?

Generated SQL Query

```
SELECT COUNT(*) FROM employees WHERE strftime('%Y', hire_date) = '2023';
```

Query Results (1 rows)

COUNT(*)
3

Answer

In 2023, 3 employees were hired.

8. ADVANTAGES & DISADVANTAGES:

Advantages:

1. Democratized Data Access

Enables non-technical users to query databases using plain English, removing SQL expertise as a barrier.

2. Improved Productivity

Reduces time spent writing and debugging SQL queries, enabling faster data insights.

3. Accurate Query Generation

Leverages Google Gemini Pro's advanced NL understanding for accurate SQL translation.

4. User-Friendly Interface

Streamlit-based UI with schema visualization, sample questions, and query history.

5. Natural Language Answers

Provides human-readable summaries of query results, not just raw data.

6. Safety Features

SELECT-only execution prevents accidental data modification; SQL injection protection built-in.

7. Scalability

Can be extended to support MySQL, PostgreSQL, and other databases.

8. Cost Effective

Uses lightweight SQLite database and free-tier Gemini API for development.

Disadvantages:

1. API Dependency

Relies on Google Gemini Pro API availability and quota limits.

2. Complex Query Limitations

Very complex or ambiguous queries may generate incorrect SQL.

3. Network Requirement

Requires internet connectivity for LLM API calls.

4. Schema Understanding

LLM may struggle with unfamiliar or poorly documented schemas.

5. Cost at Scale

API costs may increase significantly with high usage volumes.

6. Limited to SELECT

Cannot perform INSERT, UPDATE, DELETE operations for safety reasons.

7. Latency

LLM API calls introduce latency compared to direct SQL execution.

8. Data Privacy

Query content is sent to external LLM API, which may raise privacy concerns.

9. CONCLUSION

The IntelliSQL project successfully demonstrates the application of Large Language Models (LLMs) and Generative AI in the domain of intelligent database querying. By leveraging Google Gemini Pro's natural language understanding capabilities, the system accurately converts plain English questions into valid SQL queries.

The solution was implemented as a web-based platform using Streamlit, providing a user-friendly interface where users can view database schemas, ask questions naturally, see generated SQL with syntax highlighting, and receive both tabular results and natural language summaries.

Extensive testing showed that the system achieves:

- 95% SQL accuracy on standard queries
- < 3.5 seconds average total response time
- Successful handling of JOINs, aggregations, and date filtering

IntelliSQL makes database querying accessible to everyone, making it a valuable tool for business intelligence, data democratization, and AI-powered enterprise solutions.

10. FUTURE SCOPE

The future scope of IntelliSQL is extensive. With advancements in AI, cloud computing, and enterprise database technologies, this system has the potential to become a fully integrated, intelligent data access platform.

1. Support for Multiple Databases

Extend support to MySQL, PostgreSQL, SQL Server, and cloud databases like BigQuery and Snowflake.

2. Custom Data Upload

Allow users to upload CSV/Excel files and query them using natural language.

3. Query Optimization Suggestions

Analyze generated queries and suggest optimizations for better performance.

4. Multi-turn Conversation Context

Enable follow-up questions that reference previous queries for contextual conversations.

5. Export Capabilities

Export query results to CSV, Excel, or PDF formats.

6. Voice Input Support

Implement speech-to-text for voice-based database querying.

7. Data Visualization

Generate charts and graphs automatically based on query results.

8. Cloud Deployment

Deploy on AWS, Azure, or Google Cloud for enterprise scalability.

9. Role-Based Access Control

Implement user authentication and table-level access permissions.

10. Query Caching

Cache frequent queries for improved response times.

11. APPENDIX

Project Links:

GitHub Repository:
https://github.com/MalleGowthami/IntelliSQL.git

Demo Video Link:

https://youtu.be/QRYmz_zpOeM