

# Google Cloud Generative AI Project Documentation

## 1. Introduction

- **Project Title:** Intelligent SQL Querying with LLMs using Gemini Pro

- **Team Members:**

1. Malle Gowthami (Team Leader)
2. Y Swetha Reddy
3. Chattukonda Santhosh Raj
4. Yugandhar Kasu

## 2. Project Overview

- **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

## 3. 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

## **4. Setup Instructions**

### **Prerequisites:**

- Python 3.9 or higher
- Google Cloud account (for Gemini API access)
- Internet connection

### **Technical Skills:**

- Basic Python programming
- Familiarity with Streamlit web framework
- Understanding of SQL and database concepts

### **Hardware/Software Requirements:**

- Computer with Windows, macOS, or Linux
- Python installed
- Web browser (for dashboard access)

### **Data Requirements:**

- Access to sample company database (SQLite)
- Database schema includes departments, employees, projects, salaries, project\_assignments

### **Tools and Libraries:**

- Streamlit
- Pandas
- SQLAlchemy
- python-dotenv
- Google Gemini Pro API
- SQLite Browser (for database inspection)
- Postman (for API testing)

### **Domain Knowledge:**

- Database querying and management
- Data analytics

### **Other Requirements:**

- API key for Google Gemini Pro (obtain from Google AI Studio)
- Environment variables configured in .env file

## 5. Running The Application

After setting up the required dependencies and environment variables, follow these steps to run the IntelliSQL application locally:

### Frontend:

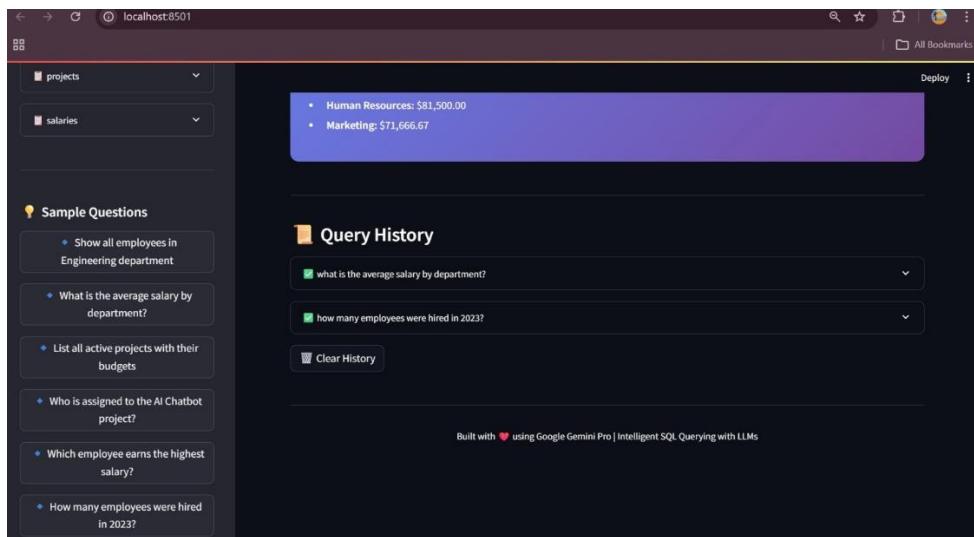
- Streamlit (for interactive web UI)
- HTML, CSS, JavaScript (used within Streamlit components for visualization)

### Backend (app.py server):

1. Python
2. Streamlit
3. Pandas
4. SQLAlchemy

## 6. User Interface:

### 6.1 Output Screenshots



The screenshot shows the Gemini Pro interface at localhost:8501. On the left, there's a sidebar with dropdown menus for "projects" and "salaries". Below this is a "Sample Questions" section with several expandable items:

- 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?

In the center, there's a "Query Results (5 rows)" section with a table:

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

Below the table is an "Answer" section with a purple background containing the following text:

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

The screenshot shows the Gemini Pro interface at localhost:8501. On the left, there's a sidebar with dropdown menus for "departments", "employees", "project\_assignments", "projects", and "salaries". Below this is a "Sample Questions" section with one item:

- Show all employees in Engineering department

In the center, there's a large "Intelligent SQL Querying" section with the following components:

- A title "Intelligent SQL Querying" with a robot icon.
- A subtitle "Ask questions in plain English — Gemini Pro converts them to SQL and fetches results from the database".
- An input field labeled "Ask a question about the database:" with the placeholder "e.g., Show me all employees in the Engineering department".
- A red "Ask" button with a microphone icon.
- A footer note "Built with ❤️ using Google Gemini Pro | Intelligent SQL Querying with LLMs".

The screenshot shows the IntelliSQL interface on a dark-themed browser window. On the left, there's a sidebar with dropdown menus for 'projects' and 'salaries'. Below these are several 'Sample Questions' cards:

- 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?

In the main area, the title 'Intelligent SQL Querying' is displayed above a search bar. The search bar contains the question 'what is the average salary by department?'. To the right of the search bar is a red 'Ask' button. Below the search bar, the heading 'Generated SQL Query' is shown, followed by a block of SQL code:

```

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;

```

Below the SQL code, the heading 'Query Results (5 rows)' is displayed, followed by a table with two columns: 'department\_name' and 'average\_salary'. The table shows one row for 'Data Science' with an average salary of 96,666.6667.

This screenshot shows the same interface but with a different set of sample questions in the sidebar:

- departments
- employees
- project\_assignments
- projects
- salaries

The main area shows the question 'how many employees were hired in 2023?' in the search bar, with a red 'Ask' button to its right. The 'Generated SQL Query' section displays the following SQL code:

```

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

```

The 'Query Results (1 rows)' section shows a table with one row labeled 'COUNT(\*)' containing the value '3'. Below this, the 'Answer' section contains the text 'In 2023, 3 employees were hired.'

## 7. Testing

To ensure that all components of the IntelliSQL system—including the LLM-powered SQL generation, backend APIs, and Streamlit frontend—work correctly, efficiently, and reliably under various database scenarios before deployment.

### Testing Strategy Overview:

- LLM Model Testing:** Validate the accuracy and robustness of natural language to SQL conversion.
- Backend API Testing:** Ensure APIs respond correctly, handle errors gracefully, and perform well under load.

- **Frontend (Web Interface) Testing:** Check UI responsiveness, usability, and correct data visualization.
- **End-to-End Testing:** Simulate real-world scenarios to verify the workflow from user query to SQL execution and result display.

## 8. Known Issues

- Occasional delays in API response due to network or LLM latency
- Limited database schema may affect query flexibility for complex questions
- No mobile application support currently (planned for future development)
- Basic UI design; lacks advanced user customization features
- Model performance may degrade for ambiguous or poorly structured queries

## 9. Future Enhancements

- Machine Learning & Model Enhancements (improve NL→SQL accuracy, add new features)
- Cloud Deployment (for scalability and reliability)
- Integration with enterprise databases (MySQL, PostgreSQL, etc.)
- Advanced AI Models (deep learning, multi-turn conversation context)
- User Personalization (customized query suggestions, saved queries, dashboards)

## 10. Conclusion

The IntelliSQL project demonstrates how natural language processing and large language models can be effectively combined to improve database accessibility. By providing accurate SQL generation, query execution, and result summarization, the system enables non-technical users to interact with databases efficiently. With its scalable architecture and user-friendly interface, IntelliSQL lays the foundation for smarter, more accessible data-driven applications. Future enhancements will further expand its capabilities, making it a valuable tool for business intelligence and enterprise data management.