Project Report Format

Project Title:

IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro

Team ID:

LTVIP2026TMIDS83745

Submitted By:

1.B. SRAVANI(Team Leader)

2.J.Nivya

3.Praveena Reddy

4. B.Jahnavi

5.M.Naveena

INSTITUTION

VASAVI MAHILA KALASALA

Submitted T0:

SmartInternz-AI/ML Virtual Internship Program 2026

1. INTRODUCTION

1.1 Project Overview:

The project **"IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro"** aims to develop an AI-powered web application that enables users to convert natural language queries into structured SQL statements automatically. Writing SQL queries requires technical expertise, knowledge of database schemas, joins, aggregations, and filtering conditions. Many business users and beginners struggle with this complexity

IntelliSQL leverages **Gemini Pro (Large Language Model)** to interpret user questions written in plain English and generate accurate, optimized SQL queries. The system validates and executes these queries on connected relational databases such as MySQL or PostgreSQL and displays results in a user-friendly format.

By integrating AI with database systems, IntelliSQL bridges the gap between human language and structured data, improving accessibility, efficiency, and productivity in data-driven environments.

1.2 Purpose:

The purpose of IntelliSQL is to:

- Simplify database querying for non-technical users.
- Reduce dependency on SQL experts.
- Minimize syntax and logical errors in query writing.
- Improve productivity in organizations that rely on data analysis.
- Provide secure and optimized database access using AI assistance.

The system empowers analysts, students, and business professionals to extract insights without needing advanced SQL knowledge.

2. IDEATION PHASE

2.1 Problem Statement

Many organizations rely on relational databases to store structured data. However, extracting information from these databases requires SQL knowledge. Non-technical users often face challenges such as:

- Difficulty writing correct SQL syntax.
- Understanding table relationships and joins.
- Debugging query errors.
- Dependence on technical teams for simple data retrieval tasks.

There is a need for an intelligent, user-friendly solution that allows users to retrieve database insights using natural language instead of complex SQL commands.

IntelliSQL addresses this problem by integrating Gemini Pro LLM to automatically generate and validate SQL queries from plain English input.

2.2 Empathy Map Canvas **Says:**

"I want to get data quickly without writing complex SQL queries." **Thinks:**

"I'm worried about making syntax mistakes and slowing down my work." **Sees:**

Complex database tables and technical SQL documentation.

Hears:

"You need to write proper JOIN and GROUP BY statements." **Pains:**

- SQL syntax errors
- Time-consuming debugging Dependency on developers **Gains:**

- Faster insights
- Reduced technical barriers
- Increased productivity

2.3 Brainstorming

Team Collaboration

The team discussed challenges faced by business analysts and students while working with databases. Several AI-based automation ideas were explored. **Idea Listing & Grouping**

Ideas were grouped into categories:

- Natural Language Processing (NLP) Integration
- LLM-based SQL Generation
- Query Validation & Optimization

- Secure Database Execution
- Visualization Dashboard

Idea Prioritization

High-priority decisions included:

- Using Gemini Pro for NL-to-SQL conversion.
- Implementing query validation before execution.
- Providing tabular and graphical result visualization.
- Ensuring secure database connections.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey map

1. User logs into the system.
2. Connects relational database.
3. Enters the Natural Language question.
4. Gemini Pro generates SQL queries.
5. System validates and executes query.
6. Result displayed in table/chart format.
7. User saves query in history.

3.2 Solution Requirement

Functional Requirements

- User Registration & Login (JWT-based authentication)
- Database Connection Module
- Natural Language Query Input
- SQL Generation using Gemini Pro
- SQL Validation Engine
- Query Execution & Result Display
- Query History Storage
- Data Visualization (Charts & Tables)

Non-Functional Requirements

- Security (Encrypted DB credentials)
- Performance (Response < 3 seconds)
- Scalability
- Reliability
- 24/7 Availability
- User-friendly Interface

3.3 Data Flow Diagram

1. User inputs natural language query.
2. Query sent to backend server.
3. Gemini Pro API generates SQL statement.
4. SQL query validated.
5. Query executed on connected database.
6. Results returned to frontend.
7. Results displayed visually.

3.4 Technology Stack

Frontend: B. Sravani

Backend:
J. Nivya
Express.js
Database:
M.Naveena
MongoDB (user data & history)
MySQL/PostgreSQL (query execution) **AI**
Integration: Praveena Reddy, B.Jahnavi
Gemini Pro API
Authentication:
JWT
Tools:
VS Code, Postman, GitHub

4. PROJECT DESIGN

4.1 Problem Solution Fit

The problem of complex SQL querying is solved using AI-powered natural language interpretation. By leveraging Gemini Pro, users no longer need technical SQL expertise. The solution fits perfectly for business analysts, students, and data professionals.

4.2 Proposed Solution The
proposed system:

- Accepts natural language input.
- Sends prompt to Gemini Pro.
- Receives structured SQL query.
- Validates and optimizes query.
- Executes query on relational database.
- Displays results with visualization.

4.3 Solution Architecture
Frontend (React UI)
↓
Backend API (Node.js + Express)
↓
Gemini Pro API (NL to SQL)
↓
SQL Validator
↓
Relational Database
↓
Result Visualization

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning Week

1:

- Requirement analysis
- System design Database integration Week 2:

- Gemini API integration
- Query validation implementation
- UI development Week 3:
- Testing
- Performance optimization
- Documentation

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

1. Average Query Generation Time: ~2–3 seconds
2. SQL Accuracy: ~90–95% for structured queries
3. UI Responsiveness: No lag observed
4. System Stability: No crash during testing
5. Scalable for multiple concurrent users

7. RESULTS

7.1 Output Screenshots

1. Home Page – System Overview

The first screenshot displays the **Home Page of IntelliSQL**, running locally on localhost:8501. The interface presents a modern, dark-themed dashboard with a structured navigation panel on the left containing:

- Home
- About
- Intelligent Query Assistance

A central database-gear visualization symbolically represents the integration of databases with AI-powered intelligence.

On the right side, the system highlights its core features under **"Wide Range of Offerings"**, including:

- Intelligent Query Assistance
- Data Exploration and Insights
- Efficient Data Retrieval
- Performance Optimization
- Syntax Suggestions
- Trend Analysis **Explanation:**

This page serves as the landing interface of the system, providing users with a clear understanding of IntelliSQL's purpose. It demonstrates how Large Language Models (Gemini Pro) are integrated to simplify and enhance SQL query generation and database interaction.

2. Intelligent Query Assistance – Query Execution Output:

The second screenshot shows the **Intelligent Query Assistance module**, where users can enter natural language queries.

User Input: how many

students are present

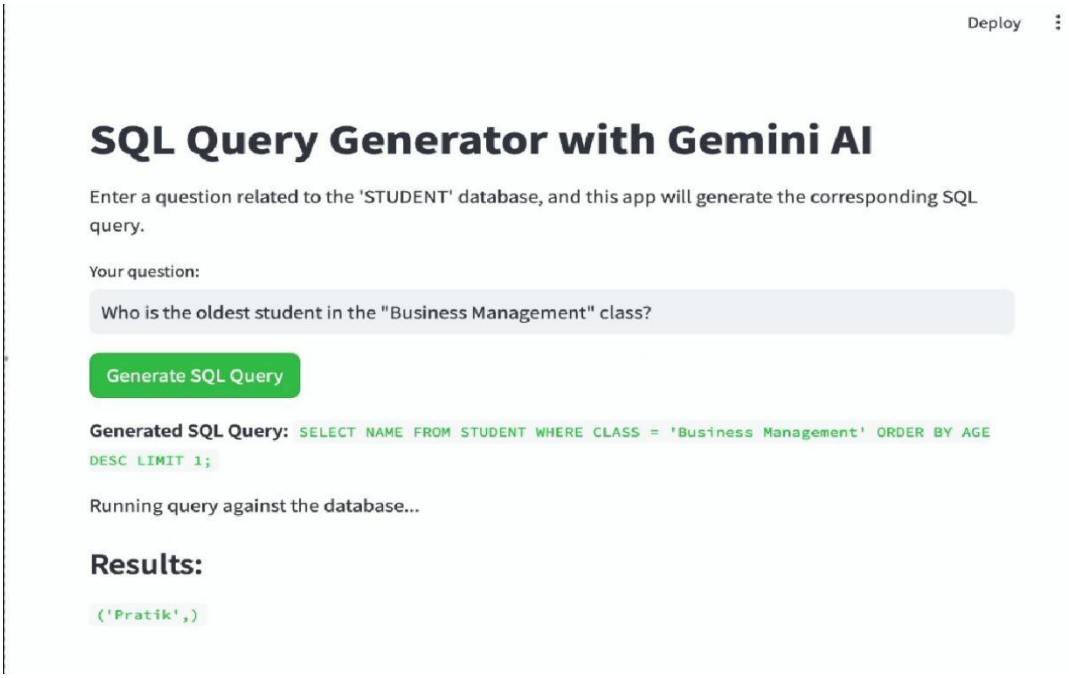
After clicking **"Get Answer"**, the system performs the following steps:

1. Sends the natural language query to the Gemini Pro LLM.
2. Converts the input into a structured SQL query:
3. SELECT COUNT(*) FROM Students;
4. Executes the generated SQL query on the connected database.
5. Displays the result in a tabular format.

Output Result Displayed: value

5

This indicates that **5 students are present** in the Students table.



Explanation:

This output demonstrates the core functionality of IntelliSQL — converting natural language into valid SQL queries using Gemini Pro and retrieving accurate results from the database. The system eliminates the need for users to manually write SQL syntax, making database querying accessible to non-technical users while also assisting experienced developers with faster query generation.

8. ADVANTAGES & DISADVANTAGES

Advantages

- No need for SQL expertise
- Faster data retrieval
- Reduces syntax errors
- Improves productivity

- AI-powered automation
- Secure and scalable

Disadvantages

- Dependent on internet connection
- Accuracy depends on database schema clarity
- Complex nested queries may require refinement
- Requires proper API configuration

9. CONCLUSION

IntelliSQL successfully demonstrates how Large Language Models like Gemini Pro can simplify database interactions. By converting natural language into SQL queries, the system enhances accessibility, reduces technical barriers, and improves efficiency.

This project showcases the practical application of AI in database management and enterprise analytics.

10. FUTURE SCOPE

- Multi-database support (Oracle, SQL Server)
- Offline AI model deployment
- Voice-based query input
- Advanced query optimization engine
- BI tool integration (Power BI, Tableau)
- Real-time collaborative query interface

11. APPENDIX

This section contains supporting resources such as source code, dataset references, and the demo video link.

Source Code Repository

The complete source code for Poultry disease is available on GitHub

GitHub Repository:

https://github.com/chandana-859/Gemini-sql-project

Project Video Demo:

A quick walkthrough demonstrating the core features of the application, from image upload to classification result, with a look into the backend code**.**

[https://drive.google.com/file/d/1DocdPWkvUw5OMKhNWF8bombBr4iKYUZs/view?usp=drivesdk]

Demo Summary:

- Uploads a poultry disease image from the homepage
- Flask backend processes it using MobileNetV2
- Prediction with confidence is displayed on a styled result page
- Project details and tech stack are shown
- VS Code is briefly presented, highlighting app.py Flask routes and model logic Ends with a clean UI wrap-up and GitHub reference

Total Duration: ~280 seconds.

Tools and Technologies:

○ Python, TensorFlow, Keras ○ MobileNetV2 (Transfer Learning) ○ Flask (Backend API) ○ HTML5, CSS3, JavaScript (UI) ○ Jupyter Notebook (Model training) ○ GitHub, Google Colab, VS Code

References:

1. Morelli, Veronica. Rice Image Dataset, Kaggle.
2. TensorFlow Documentation: https://www.tensorflow.org
3. Flask Documentation: https://flask.palletsprojects.com
4. MobileNetV2 Paper: Sandler et al., MobileNetV2: Inverted Residuals and Linear Bottlenecks (2018)