# Generative AI Project

SQL query generator from plain English

Ayush Chichani (Enrollment No. : 0827CS221063)
Trainer : Mr. Aman Dharmendra Sir
1 September 2025

# Abstract

The project SQL Query Generator demonstrates the use of Natural Language Processing (NLP)-inspired techniques and JavaScript logic to convert plain English text into SQL queries. Unlike traditional SQL learning platforms that require strict syntax knowledge, this system allows users to type natural queries (e.g., “Show all users where age is greater than 25”) and instantly generates valid SQL statements (SELECT, INSERT, UPDATE, DELETE).

The project provides a browser-based interface where users can input a query in English, select a query type, and view the corresponding SQL code. The application also includes features like example queries, copy-to-clipboard, and a responsive UI for ease of use.

Developed using HTML, CSS, and JavaScript, the system is lightweight, platform-independent, and requires no external installations. The project serves two purposes: (1) educational, by helping beginners learn SQL query structures, and (2) practical, by accelerating query-writing for quick database operations.

# Project Details

## A. Objective:

The primary objective is to build a simple, interactive tool that transforms natural English statements into SQL queries automatically.

**Goals achieved:**

1. Implemented parsing logic for SELECT, INSERT, UPDATE, DELETE.
2. Allowed condition parsing (WHERE, ORDER BY, greater than, less than, equals).
3. Designed a responsive user interface with examples and auto-generate functionality.
4. Provided a copy-to-clipboard option for direct use in SQL editors.
5. Ensured compatibility across devices and browsers.

## B. Methodology:

**Tools and Technologies Used:**

- HTML: Structure of the interface (form inputs, buttons, examples).
- CSS: Styling for modern, gradient-based UI with responsiveness.
- JavaScript: Core logic for parsing English queries, generating SQL, and interactivity.

**Algorithmic Approach:**

- Keyword Matching: Identify common fields (name, age, city, salary, email).
- Pattern Recognition: Use regex to detect conditions like “greater than 25” → age > 25.
- Query Assembly: Construct SQL based on query type (SELECT/INSERT/UPDATE/DELETE).

**Validation:** Prevent unsafe queries (DELETE without WHERE).

**Development Environment:** Visual Studio Code.

**Testing:** Google Chrome, Mozilla Firefox, Microsoft Edge.

## C.Implementation:

### 1. SELECT Module

- Detects fields (name, age, city, salary).
- Generates SELECT * FROM table WHERE conditions.
- Handles sorting with ORDER BY and conditions with AND.

### 2. INSERT Module

- Extracts column-value pairs (e.g., “Add new user John with age 30”).
- Generates INSERT INTO table (columns) VALUES (values);.

### 3. UPDATE Module

- Identifies update values (e.g., “Update user salary to 60000 where name is Alice”).
- Generates UPDATE table SET column=value WHERE condition;.

### 4. DELETE Module

- Ensures WHERE condition exists before deletion.
- Generates DELETE FROM table WHERE condition;.

**Frontend UI Features:**

- Gradient-styled container with modern design.
- Example queries clickable for auto-generation.
- Copy-to-clipboard with tooltip feedback.
- Fully responsive grid layout.

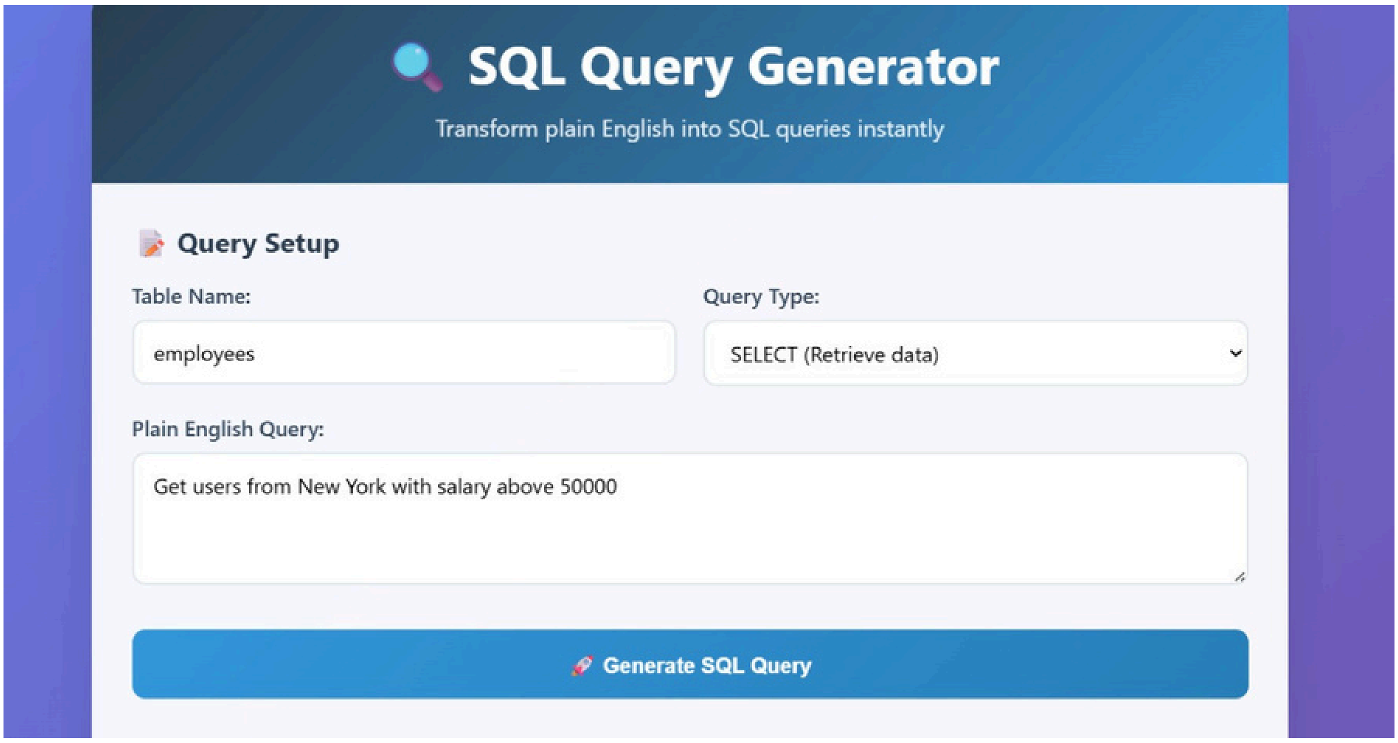
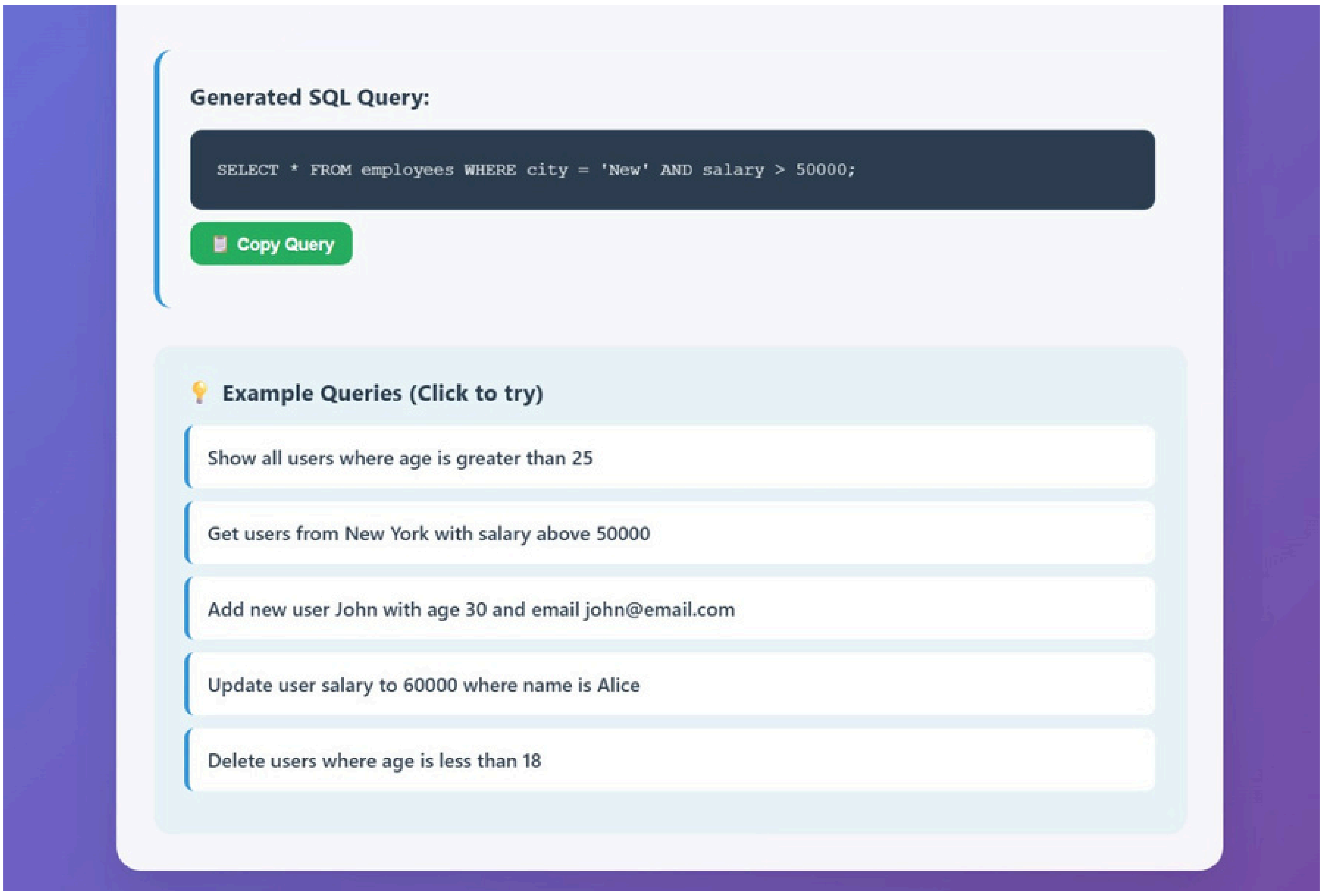


Table Name: employees

Query Type: SELECT (Retrieve data)

Plain English Query: Get users from New York with salary above 50000



```
SELECT * FROM employees WHERE city = 'New' AND salary > 50000;
```

- Show all users where age is greater than 25
- Get users from New York with salary above 50000
- Add new user John with age 30 and email john@email.com
- Update user salary to 60000 where name is Alice
- Delete users where age is less than 18

## D.Results:

### Functional Query Generator

- Successfully converted natural English queries into valid SQL syntax.
- Supported SELECT, INSERT, UPDATE, DELETE.

### Condition Handling

- Age, salary, name, city, and email conditions parsed correctly.
- ORDER BY supported with ascending/descending.

### Browser Compatibility

- Tested on Chrome, Firefox, Edge with consistent performance.

### Usability

- Intuitive interface with examples.
- Responsive design works on desktops and mobiles.
- Offline support (just open index.html).

### Educational Value

- Helps beginners understand SQL query formation.
- Provides an interactive bridge between English and SQL.

# Conclusion

### Key Learnings

1. Learned how NLP-style parsing can be applied to structured query generation.
2. Strengthened skills in JavaScript regex and condition parsing.
3. Designed a professional UI using modern CSS gradients and responsiveness.
4. Understood the importance of safe query validation (e.g., DELETE requiring WHERE).
5. Realized the educational value of tools that simplify database learning.

**Future Improvements:**

1. Extend support to JOIN queries and aggregate functions (SUM, COUNT, AVG).
2. Add natural language support for nested queries.
3. Integrate with an actual database engine (MySQL/SQLite) for execution.
4. Provide voice-to-query support for accessibility.
5. Deploy as a mobile/web app for wider usage.

## References

**[1]** Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach. Pearson.

**[2]** Date, C. J. (2019). An Introduction to Database Systems. Addison-Wesley.

**[3]** W3Schools. SQL Tutorial. https://www.w3schools.com/sql/

**[4]** Mozilla Developer Network. JavaScript Documentation. https://developer.mozilla.org/

**[5]** RegExr. Learn, Build, & Test RegEx. https://regexr.com/

## Code

### GitHub link:

https://github.com/AyushChichani/SQl_generator