TEXT TO SQL - Installation, Setup, and Documentation

1. Project Overview

This project converts natural language queries into SQL queries using a Language Model. It is built with FastAPI and integrates an LLM for natural-to-SQL conversion. The application supports streaming responses and maintains a query history for context-aware interactions.

---

2. Installation & Setup Instructions

2.1 Prerequisites

- Python 3.8+
- FastAPI
- Uvicorn
- Groq API Key
- Dotenv
- Jinja2 Templates

2.2 Clone the Repository

```
git clone <https://github.com/KRISNABADDE/text-sql-assistant.git>
cd text-sql-assistant
```

2.3 Install Dependencies

```
pip install -r requirements.txt
```

2.4 Set Up Environment Variables

Create a .env file in the root directory and add your Groq API key:

```
GROQ_API_KEY=your_api_key_here
```

2.5 Run the Application

```
uvicorn main:app
```

The application will be available at: http://localhost:8000

---

3. How It Works

1. **User Input:** A natural language query is submitted.

2. **Processing:**
    - The system checks query history for context.
    - It constructs messages for the LLM using the database schema.
    - The query is sent to the Groq LLM for SQL conversion.
3. **Streaming Response:** The SQL query is returned in real-time.
4. **Query History:** The query and its response are stored for future context.

---

4. Design Choices

- **FastAPI for API**: Chosen for performance and async support.
- **Streaming Response**: Provides real-time query generation.
- **Memory Service**: Enhances context-aware query generation.
- **Groq API with LLaMA**: Provides efficient text-to-SQL conversion.

---

5. Limitations

- **Dependent on LLM Accuracy**: Results may not always be perfect.
- **Limited Context Memory**: Query history is stored in memory (not persistent storage).
- **Schema Constraints**: Must provide an accurate database schema for optimal results.
