

# AI-Powered Data Query Interface

By Neyati

<https://github.com/Doraemon012/OminiQuery>



# Problem Statement

Organizations often struggle with efficiently accessing and understanding client data stored in internal databases. Traditional methods of data retrieval and analysis can be time-consuming and cumbersome, leading to delays in decision-making and inefficiencies in client management. There is a need for a streamlined and intuitive solution that allows organization members to query and receive timely, accurate insights from the database through a user-friendly interface.

## KEY CHALLENGES FOR AN ORGANIZATION

- Inefficient data access
- Need for understanding of data retrieval queries and visualization
- Complex user interface of databases
- Delays in decision making
- Slow data retrieval




# Solution

Develop a chat interface powered by a Large Language Model (LLM) to read and interpret client data from internal databases. This interface will enable organization members to query the database and receive accurate, contextually relevant responses in real-time.

The sales department has the maximum employees.

Number of employees in sales department = 80000

Department	Name	Salary
Sales	John Smith	80000
Marketing	Jane Doe	75000
Development	Michael Jones	90000
Development	Sarah Lee	85000

 Export to Sheets

What are the top selling products in 2024?



# Traditional Data Retrieval and Visualization Process

1

## Identify Data Requirements

The user identifies what data they need and defines the query parameters.

2

## Access Database

The user logs into the organization's database system (using SQL, NoSQL, or a BI tool).

3

## Write & execute Queries

The user writes a query to fetch the required data. This requires knowledge of SQL or the query language for NoSQL databases.

4

## Export Data, Create Visualizations and Analyze

The retrieved data is exported for further analysis. The user imports the data into a visualization tool, creates charts, graphs, etc; and analyses the visuals.

# AI-Powered Data Query Interface Process

1

## Identify Data Requirements

The user defines what information or insight they need, typically in natural language.

2

## Access Database

The user inputs their query in natural language into the AI-powered interface.

The natural language query is processed to understand the user's intent, identifying relevant data parameters and context.

3

## Generate and execute query

The AI translates the user's query into a structured query, connects to the database and runs the generated query.

The system preprocess the retrieved data, performing cleaning, formatting, and initial analysis as needed.

4

## Visualize and Analyze data

The AI generates visualizations and offers initial insights and patterns

The user can further interact with the AI, refining their query or asking for additional insights and visualizations.

# Features

## Easy-to-use User Interface

Allows users to interact with the database using natural language, making data retrieval as simple as having a conversation.

## Support for both SQL and NoSQL Databases

Ensures compatibility with a variety of database types, enabling seamless integration with existing Systems.

## View and modify the generated queries

Allows users to view and modify the data fetching queries before or after execution.

## Auto Graph Generation

Automatically creates visual representations of data, such as charts and graphs, for easier analysis and insights.

## Displays the Schema Along with the Chat

Provides a real-time view of the database schema within the chat interface, helping users understand the structure and context of the data they are querying.



# Tech Stack

Python

SQLAlchemy

LLM – GPT3.5

Django

Matplotlib

FireBase

Flask

PyPlot



django



*Matplotlib*



Flask

## Code Sample: Auto Graph Generation

```
async def generate_pie_chart(prompt, ai_connection):

    # Build the AI chat/prompts
    builder = [
        '''You are a helpful, cheerful database assistant who gives size and labels to generate pie chart from the database,
        give the labels and size in json format. Do not respond with any information unrelated to databases or queries.
        Use the following database schema when creating your answers:'''
    ]

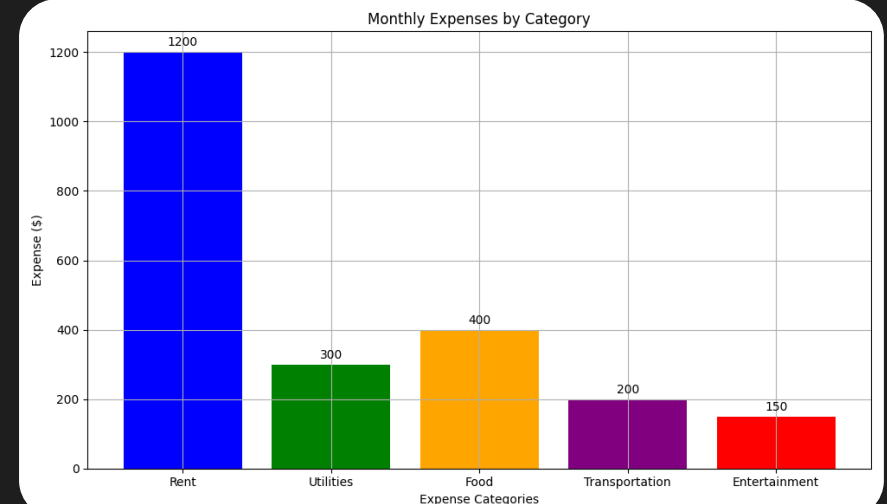
    for table in ai_connection['schema_raw']:
        builder.append(table)

    builder.append("Include column name headers in the query results.")
    builder.append('Always provide your answer in the JSON format below:')
    builder.append('{ "summary": "your-summary", "query": "your-query" }')
    builder.append('Output ONLY JSON formatted on a single line. Do not use new line characters.')
    builder.append('In the preceding JSON response, substitute "your-query" with Microsoft SQL Server Query to retrieve the requested data.')
    builder.append('In the preceding JSON response, substitute "your-summary" with an explanation of each step you took to create this query in a detailed paragraph.')
    builder.append('Do not use MySQL syntax.')
    builder.append('Always limit the SQL Query to 100 rows.')
    builder.append('Always include all of the table columns and details.')

    system_prompt = "\n".join(builder)

    messages = [
        {"role": "system", "content": system_prompt},
        {"role": "user", "content": user_prompt}
    ]

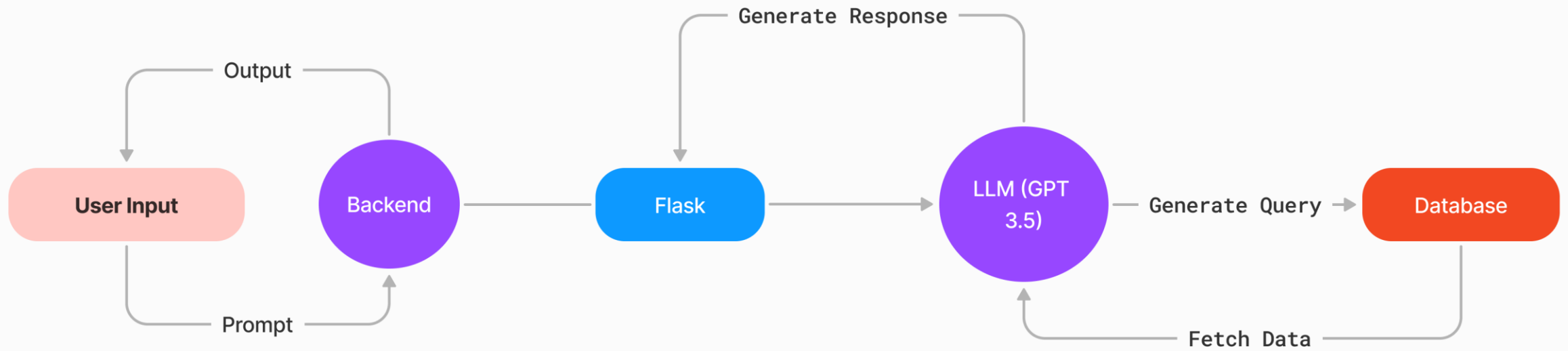
    try:
        # Send request to OpenAI model
        response = openai.chat.completions.create(
            model="gpt-3.5-turbo",
            messages=messages
        )
```





# Flow Diagram

Flow Diagram



# User Interface

The sales department has the maximum employees.

Number of employees in sales department = 80000

Department	Name	Salary
Sales	John Smith	80000
Marketing	Jane Doe	75000
Development	Michael Jones	90000
Development	Sarah Lee	85000

Export to Sheets

## Generated SQL Query

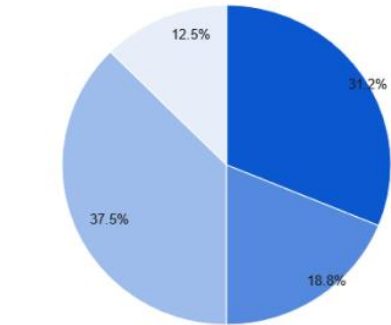
SQL

```
SELECT Department, COUNT(*) AS NumberOfEmployees
FROM Employees
GROUP BY Department;
```

Use code [with caution](#).

## Graph

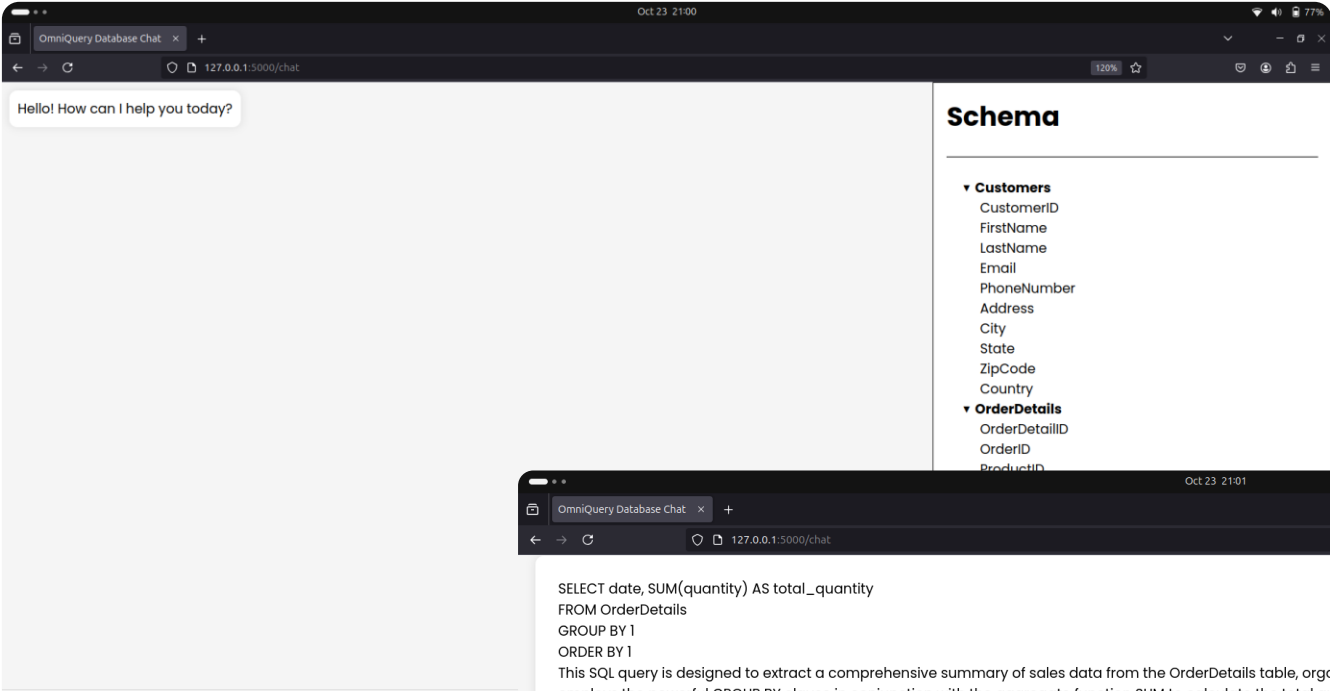
Employee Distribution by Department



What are the top selling products in 2024?



# Screen Shots



MySQL Configuration

Host:

127.0.0.1

Username:

user

Password:

.....

Database:

sales\_db

Connect

OmniQuery Database Chat

127.0.0.1:5000/chat

SELECT date, SUM(quantity) AS total\_quantity  
FROM OrderDetails  
GROUP BY 1  
ORDER BY 1

This SQL query is designed to extract a comprehensive summary of sales data from the OrderDetails table, organized by date. It employs the powerful GROUP BY clause in conjunction with the aggregate function SUM to calculate the total quantity of items sold each day.

By segmenting the data according to the date column, the query gathers sales information for each individual date. The SUM(quantity) function computes the sum of the quantity column, representing the number of items sold on each respective date.

To enhance clarity, the query renames the resulting aggregate column to total\_quantity using the AS keyword. This ensures that the output column is intuitively named and easily interpreted.


The GROUP BY 1 clause might seem redundant if you are not familiar with numeric grouping labels. In SQL, the number 1 is used to refer to the first column specified in the SELECT statement (in this case, the date column). So, GROUP BY 1 is effectively grouping the results by the date column.

Finally, to better facilitate analysis, the query includes an ORDER BY 1 clause. This arranges the results in ascending order based on the date column, thus providing a chronological presentation of sales data.

In summary, when executed, this SQL query will yield a result set consisting of two columns: date and total\_quantity. These columns represent each date and the corresponding total quantity of items sold on that date, respectively.

Result:

Total Sales by Date



Schema

Customers

CustomerID  
FirstName  
LastName  
Email  
PhoneNumber  
Address  
City  
State  
ZipCode  
Country

OrderDetails

OrderDetailID  
OrderID  
ProductID  
Quantity  
Price

Orders

Products

ProductID  
ProductName  
Category  
Price  
StockQuantity

What are the total sales by date?

Send

# Thankyou

