# DSCI-551 ChatDB Project Final Report:

## Trader John's Database Chatbot🤖

Team ChatDB 58:

Chung Sheng (Johnson) Chang

Chu-Huan (Chris) Huang

## GitHub Link

https://github.com/johnson70630/TraderJohn-Database

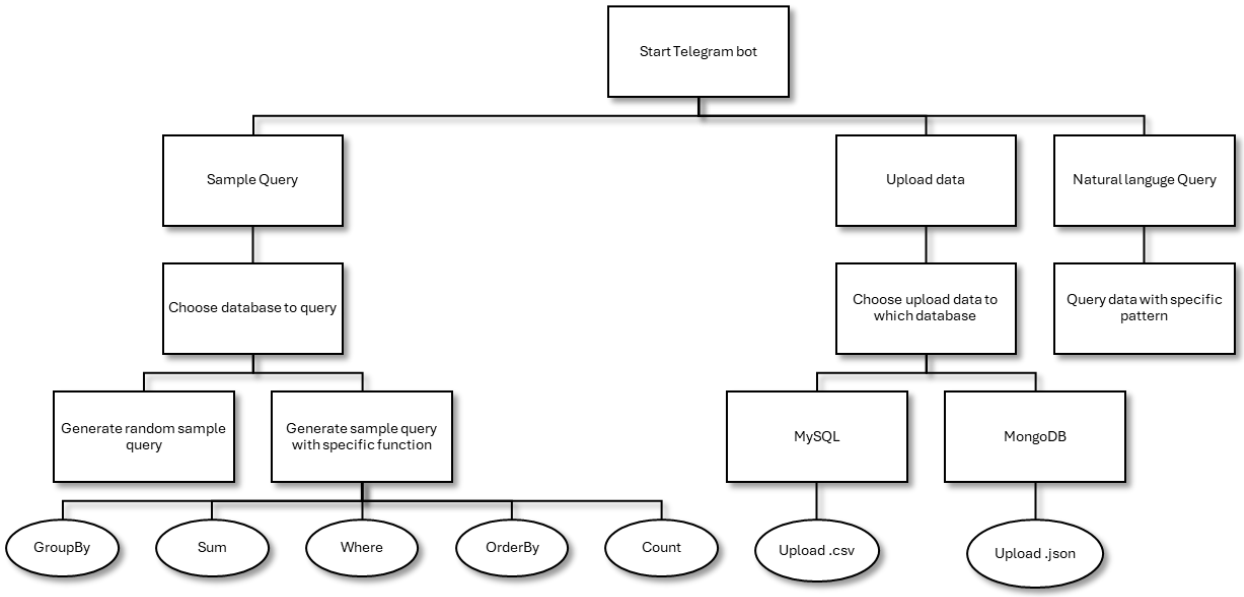
## Introduction

This document presents the final report from Team 58 on the chatDB project. In this report, we will provide a comprehensive overview of how we implemented the essential functionality of chatDB on a Telegram bot using Python. We will also discuss the challenges we faced during the implementation process and share the valuable insights we gained throughout our work.

## Planned Implementation (from proposal)

- Apply the methodologies learned in class and through online resources to use Python for the following purposes: establishing integration between two different types of databases (MySQL and MongoDB) and a Telegram bot and making it easy to retrieve and manipulate data from these databases.
- Enable the Telegram bot to accurately identify specific patterns within users' natural language input, thereby implementing the example query functionality.
- Enhance the Telegram bot's capabilities to respond to inquiries using natural language, generate example queries in programmatic form, and explain each query's functionality in the responses.

# Architecture Design




**Text Version:**

- Run Telegram Bot (Run python main.py, and open the telegram; there would be three buttons)
    - Sample Query (Choose Just give a random sample query or with a specific function)
        - Random Query would randomly give a sample query
        - Choose Specific means it would provide a sample query of the function the user selected

- Upload Data (Choose which database to upload)
  - MySQL → users can upload .csv file to MySQL on AWS RDS
  - MongoDB → users can upload .json file to MongoDB
- Natural Language Query
  - Choose which database to query(it will show up tables/collections in the database)
    - MySQL
    - MongoDB

## Implementation

**Functionalities**

1. Upload data: .csv and .json file
2. Sample Query: Random/Specific Function(orderBy, groupBy, …)
3. Natural Language Query Processing: translate natural language to query language (SQL/ MongoDB)
4. Telegram Bot

**Tech Stack**

[tool.poetry.dependencies]

python = "^3.12"

python-telegram-bot = "^21.6"

pandas = "^2.2.3"

sqlalchemy = "^2.0.35"

mysql-connector-python = "^9.0.0"

python-dotenv = "^1.0.1"

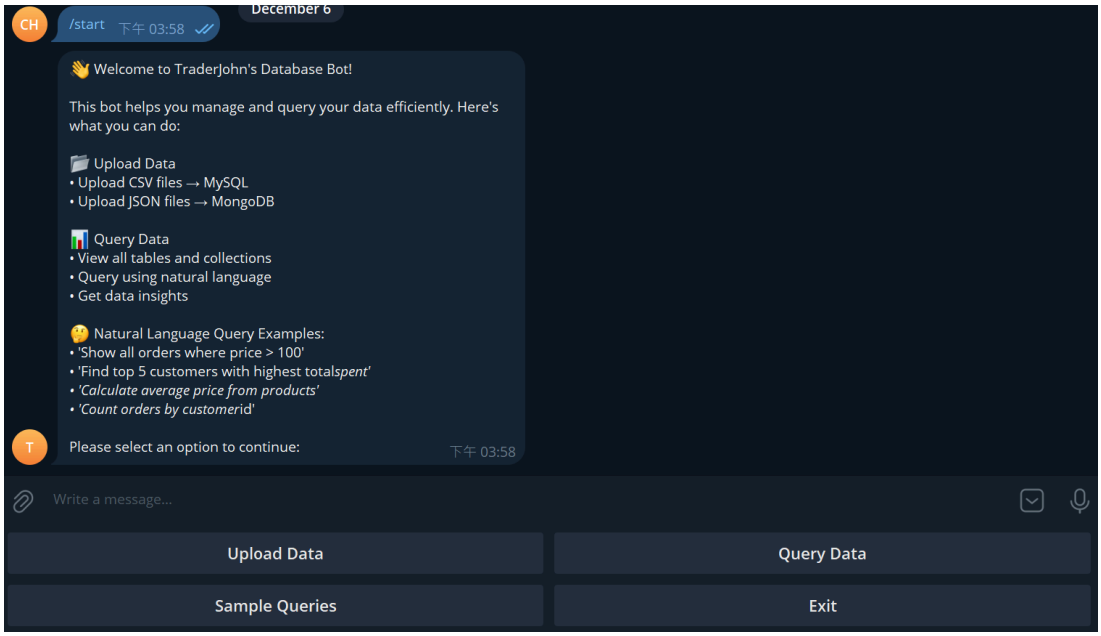
pymongo = "^4.10.1"

nltk = "^3.9.1"

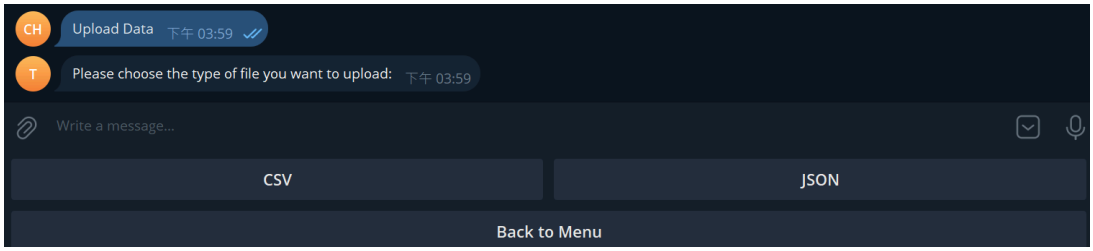
sqlparse = "^0.5.2"

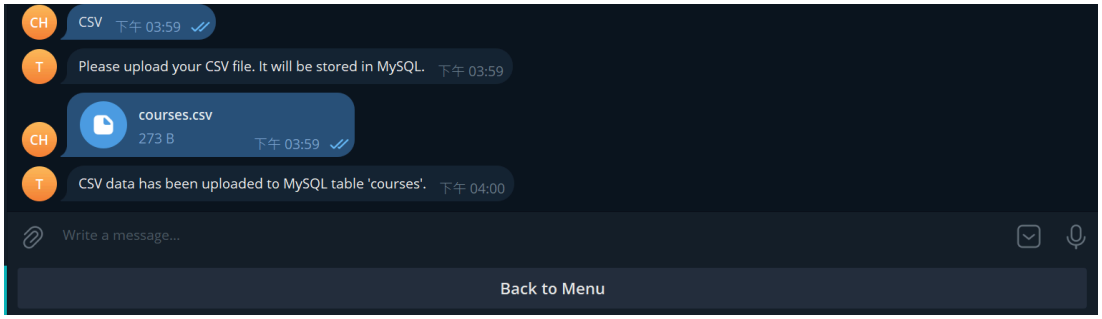
**Implementation Screenshots (Few, not all)**

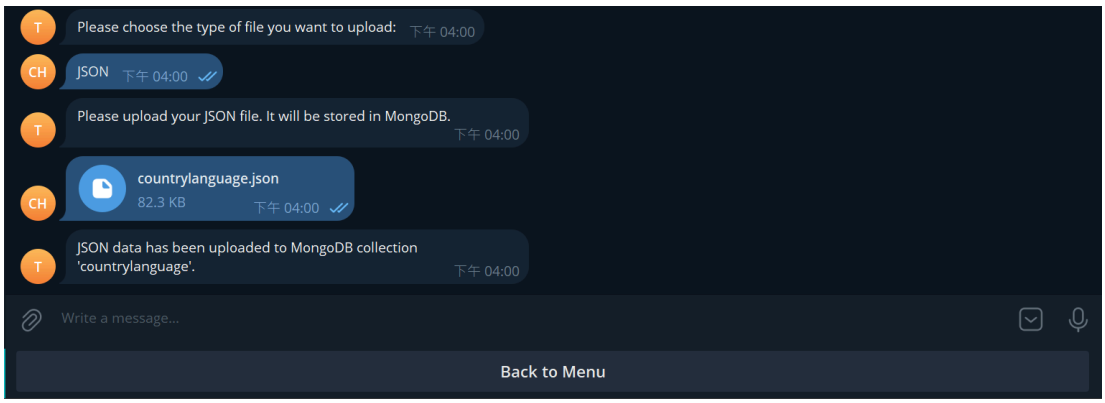
**Main menu:**



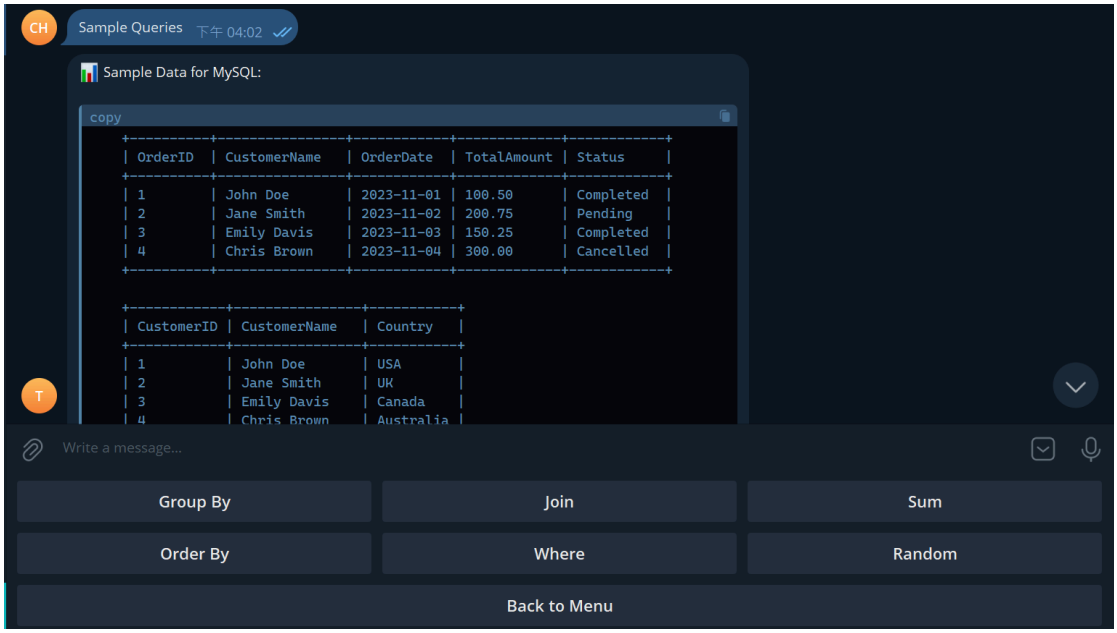
## Upload Data:

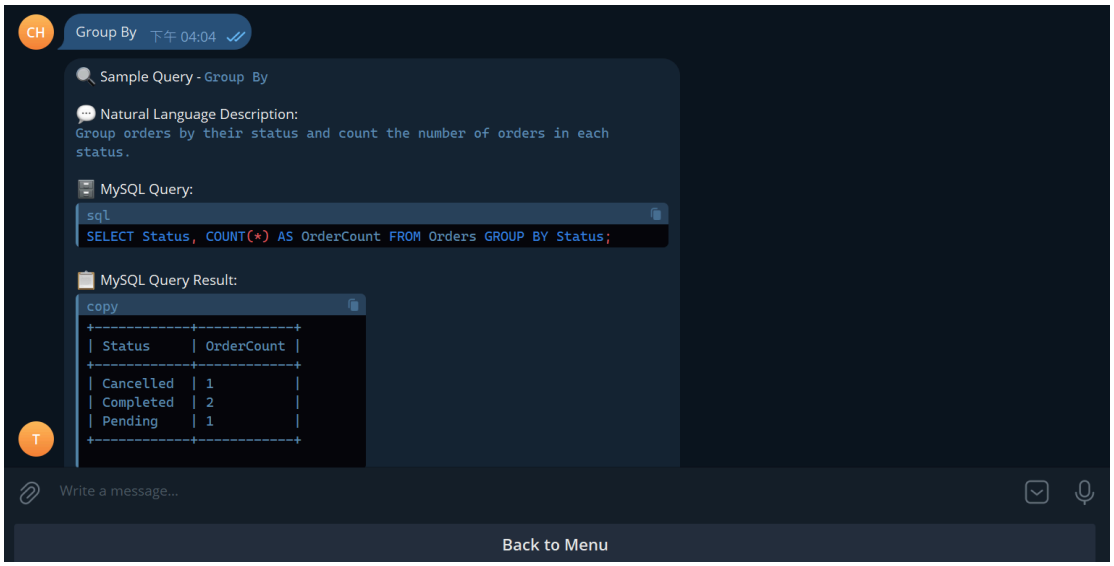

CH
Upload Data 下午 03:59
T
Please choose the type of file you want to upload: 下午 03:59
Write a message...
CSV
JSON
Back to Menu

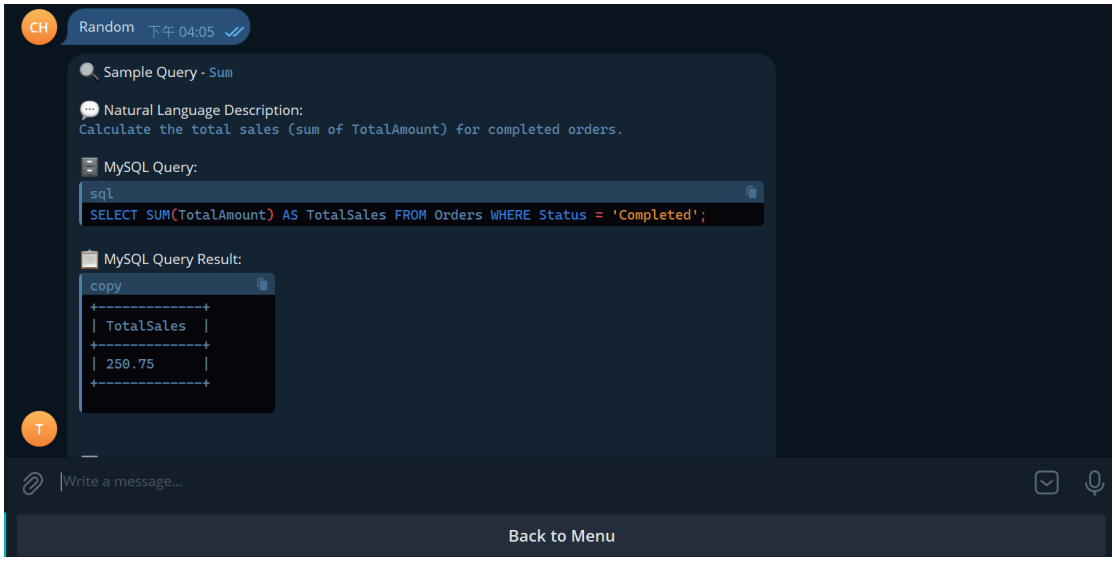

CH
CSV 下午 03:59
T
Please upload your CSV file. It will be stored in MySQL. 下午 03:59
CH
courses.csv
273 B
下午 03:59
T
CSV data has been uploaded to MySQL table 'courses'. 下午 04:00
Write a message...
Back to Menu


T
Please choose the type of file you want to upload: 下午 04:00
CH
JSON 下午 04:00
T
Please upload your JSON file. It will be stored in MongoDB.
下午 04:00
CH
countrylanguage.json
82.3 KB
下午 04:00
T
JSON data has been uploaded to MongoDB collection 'countrylanguage'.
下午 04:00
Write a message...
Back to Menu

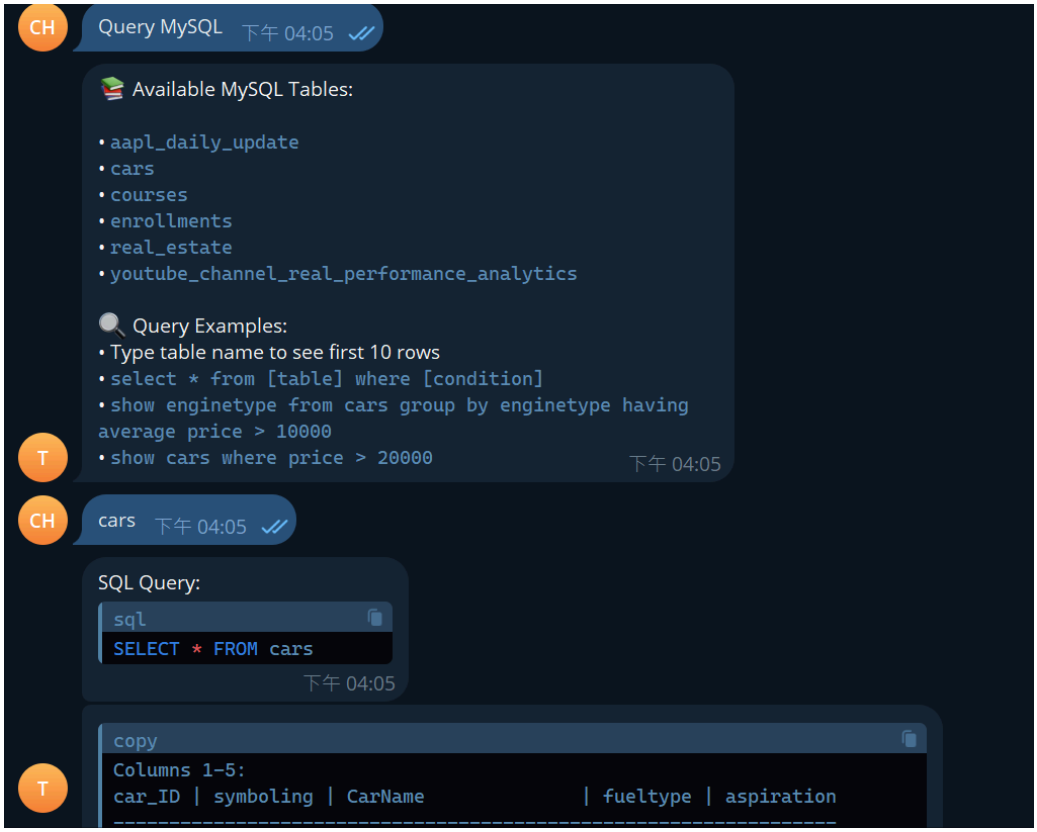
## Sample Query:

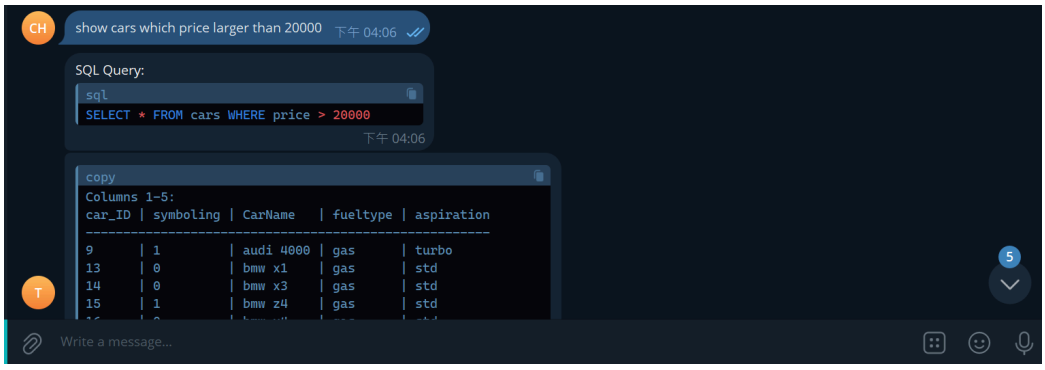




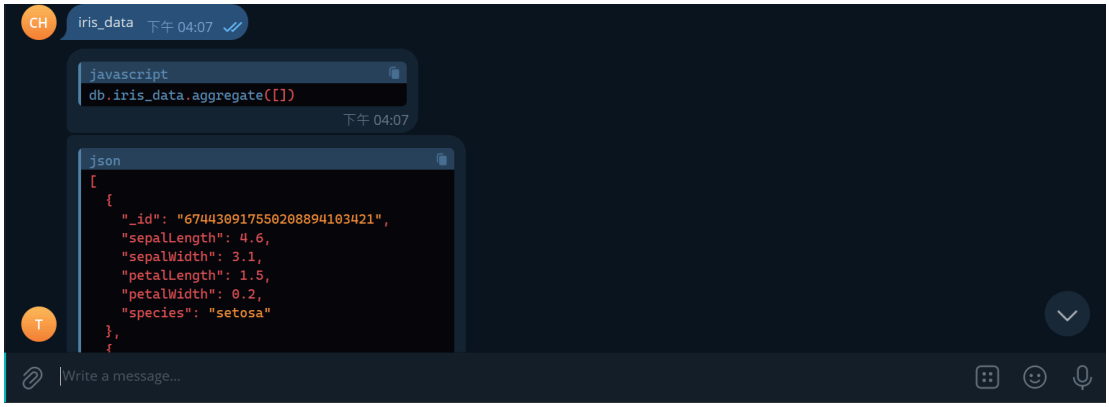



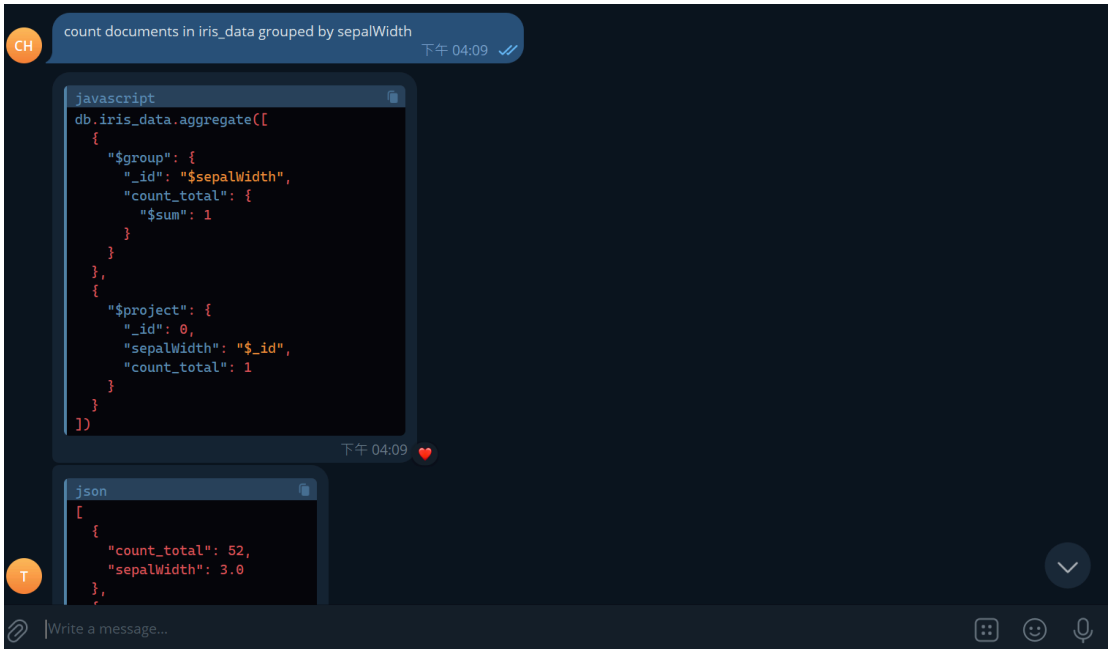

## Natural Language Query:


CH
Query MySQL 下午 04:05
Available MySQL Tables:
• aapl_daily_update
• cars
• courses
• enrollments
• real_estate
• youtube_channel_real_performance_analytics
Query Examples:
• Type table name to see first 10 rows
• select * from [table] where [condition]
• show enginetype from cars group by enginetype having average price > 10000
• show cars where price > 20000
下午 04:05
T
CH
cars 下午 04:05
SQL Query:
sql
SELECT * FROM cars
下午 04:05
T
copy
Columns 1-5:
car_ID | symboling | CarName | fueltype | aspiration


CH
show cars which price larger than 20000 下午 04:06
SQL Query:
sql
SELECT * FROM cars WHERE price > 20000
下午 04:06
copy
Columns 1-5:
car_ID | symboling | CarName | fueltype | aspiration
9 | 1 | audi 4000 | gas | turbo
13 | 0 | bmw x1 | gas | std
14 | 0 | bmw x3 | gas | std
15 | 1 | bmw z4 | gas | std
T
5
Write a message...

## MongoDB:


CH
iris_data 下午 04:07
javascript
db.iris_data.aggregate([])
下午 04:07
json
[
{
"_id": "67443091755020889410342 1",
"sepalLength": 4.6,
"sepalWidth": 3.1,
"petalLength": 1.5,
"petalWidth": 0.2,
"species": "setosa"
},
T
Write a message...


CH
count documents in iris_data grouped by sepalWidth
下午 04:09
javascript
db.iris_data.aggregate([
{
"$group": {
"_id": "$sepalWidth",
"count_total": {
"$sum": 1
}
}
},
{
"$project": {
"_id": 0,
"sepalWidth": "$_id",
"count_total": 1
}
}
])
下午 04:09
json
[
{
"count_total": 52,
"sepalWidth": 3.0
},
T
Write a message...

## Learning Outcomes

**Challenges Faced:**

1. Difficulty building a MySQL database

   → Learned how to set up MySQL on AWS RDS.

2. Challenges in implementing translation from natural language to SQL and MongoDB

   → Learned how to use pattern matching to convert natural language into SQL and translate SQL queries into MongoDB queries.

## Individual Contribution

- Johnson Chang: Telegram Bot implementation, MongoDB, Example Query function implementation, Upload Data function implementation.
- Chris Huang:  Natural language Query function implementation, MySQL Database on AWS RDS.

## Conclusion

While completing this project, we gained valuable experience implementing a chatbot by applying the database knowledge acquired from the course, such as SQL and MongoDB queries. This project allowed us to bridge theoretical concepts with practical applications, reinforcing our understanding of how databases interact with real-world systems. We faced several challenges, including designing efficient database schemas, optimizing query performance, and handling

edge cases in user interactions. Overcoming these obstacles deepened our technical skills and improved our problem-solving abilities and teamwork.

## Future Scope

In the current version of our ChatDB, there are still many queries that cannot be translated from natural language to SQL and MongoDB; we hope to improve and finish these functions in the future.